TUGAS AKHIR – IS184853

RANCANG BANGUN SMART FARMING HIDROPONIK MENGGUNAKAN SENSOR PH, TOTAL DISSOLVED SOLIDS, DAN TEMPERATUR

DESIGN AND DEVELOPMENT HYDROPONICS SMART FARMING USING PH, TOTAL DISSOLVED SOLIDS, AND TEMPERATURE SENSOR

MUHAMMAD ATHMA FARHAN
NRP 05211640000051

Dosen Pembimbing
Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro Dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR - IS184853

RANCANG BANGUN SMART FARMING HIDROPONIK MENGGUNAKAN SENSOR PH, TOTAL DISSOLVED SOLIDS, DAN TEMPERATUR

MUHAMMAD ATHMA FARHAN
NRP 05211640000051

Dosen Pembimbing
Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

UNDERGRADUATE THESIS – IS184853

DESIGN AND DEVELOPMENT HYDROPONICS SMART FARMING USING PH, TOTAL DISSOLVED SOLIDS, AND TEMPERATURE SENSOR

MUHAMMAD ATHMA FARHAN
NRP 05211640000051

Supervisor
Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

DEPARTMENT OF INFORMATION SYSTEMS
Faculty of Electrical and Intelligent Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

RANCANG BANGUN SMART FARMING HIDROPONIK MENGGUNAKAN SENSOR PH, TOTAL DISSOLVED SOLIDS, DAN TEMPERATUR

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer (S.Kom)
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas (ELECTICS)
Institut Teknologi Sepuluh Nopember

Oleh

Muhammad Athma Farhan
05211640000051

Surabaya, 14 Agustus 2020

Kepala Departemen Sistem Informasi

Dr. Muljahidin, ST., MT.
NIP. 197010102003121001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

RANCANG BANGUN SMART FARMING HIDROPONIK MENGGUNAKAN SENSOR PH, TOTAL DISSOLVED SOLIDS, DAN TEMPERATUR

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh :

MUHAMMAD ATHMA FARHAN
NRP. 05211640000051

Disetujui Tim Penguji : Tanggal Ujian : 9 Juli 2020
Periode Wisuda : September 2020

Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom. **(Pembimbing I)**

Bekti Cahyo Hidayanto, S.Si., M. Kom. **(Penguji I)**

Nisfu Asrul Sani, S.Kom., M.Sc. **(Penguji II)**

Halaman ini sengaja dikosongkan

RANCANG BANGUN SMART FARMING HIDROPONIK MENGGUNAKAN SENSOR PH, TOTAL DISSOLVED SOLIDS, DAN TEMPERATUR

Nama : Muhammad Athma Farhan
NRP : 05211640000051
Departemen : Sistem Informasi FTEIC-ITS
Pembimbing : Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

ABSTRAK

Seiring dengan perkembangan luas lahan pertanian kian menurun namun populasi manusia kian meningkat. Meningkatnya populasi manusia juga berbanding lurus dengan meningkatnya permintaan akan pangan. Oleh karena itu dibutuhkan inovasi baru dalam pertanian yaitu budidaya tanaman menggunakan hidroponik. Teknik budidaya tanaman hidroponik ini sangat cocok digunakan pada lahan yang sempit seperti daerah perkotaan yang pada umumnya hanya terdapati sedikit tanah yang baik untuk digunakan pada kegiatan pertanian. Dibandingkan dengan budidaya tanaman koonvesnional, penggunaan air pada teknik hidroponik sangat efisien karena air akan terus menerus digunakan, tidak langsung terbuang. Namun, tidak banyak orang yang mengetahui dan paham cara membudidayakan tanaman menggunakan hidroponik dikarenakan sistem yang tergolong baru dan rumit. Dibutuhkan ilmu khusus dalam perawatan tanaman dengan teknik hidroponik. Dengan berkembangnya teknologi seperti Internet of Things (IOT) dan sensor pertanian yang canggih, dapat dikembangkan sistem smart farming yang dapat melakukan proses pemantauan melalui aplikasi dan proses pengendalian sistem hidroponik dengan alat aktuator yang terhubung dengan Arduino sebagai mikrokontroler secara otomatis. Arduino dapat dikomunikasikan dengan aplikasi menggunakan API melalui pengiriman data JSON. Pengerjaan tugas akhir ini akan menghasilkan sistem yang dapat memantau

kondisi hidroponik terkini menggunakan aplikasi dan sistem yang dapat melakukan pengendalian kondisi hidroponik yang kurang optimal agar kembali ke kondisi yang optimal.

Kata Kunci : API, Arduino, Hidroponik, Internet of Things, Smart Farming

DESIGN AND DEVELOPMENT HYDROPONICS SMART FARMING USING PH, TOTAL DISSOLVED SOLIDS, AND TEMPERATURE SENSOR

Name : Muhammad Athma Farhan
NRP : 05211640000051
Department : Information Systems ELECTICS-ITS
Supervisor : Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

ABSTRACT

Along with the vast growth of agriculture is increasing but the human population is increasing. The increasing human population is also directly proportional to the demand for food. Therefore, new innovations in agriculture are needed in the cultivation of plants using hydroponics. This hydroponic cultivation technique is very suitable for use in narrow areas such as in urban areas which are easier to use on agricultural land. Compared to crop cultivation, the use of water in hydroponic techniques is very efficient because air will continue to be used quickly, not immediately wasted. However, not many people understand and understand how to grow plants using a system that is relatively new and complicated. Special knowledge is needed in treating plants with hydroponic techniques. By developing technologies such as the Internet of Things (IoT) and sophisticated agricultural sensors, intelligent farming systems can be developed that can carry out monitoring processes through applications and the process of controlling a hydroponic system with an actuator device that is connected to Arduino as a microcontroller automatically. Arduino can be communicated with applications using the API through sending JSON data. This final project will produce a system that can replace the updated hydroponic conditions using an application and a system that can make improvements to less optimal hydroponic conditions to return to optimal conditions.

Keywords: API, Arduino, Hydroponicsh, Internet of Things, Smart Farming

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: Muhammad Athma Farhan
NRP	: 05211640000051
Tempat/Tanggal lahir	: Surabaya/9 Oktober 1998
Fakultas/Departemen	: Fakultas Teknologi Elektro dan Informatika Cerdas/Sistem Informasi
Nomor Telp/Hp/email	: farhan.athma@gmail.com

Dengan ini menyatakan dengan sesungguhnya bahwa penelitian/makalah/tugas akhir saya yang berjudul:

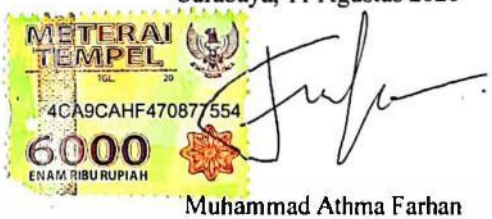
RANCANG BANGUN SMART FARMING HIDROPONIK MENGGUNAKAN SENSOR PH, TOTAL DISSOLVED SOLIDS, DAN TEMPERATUR

Bebas Dari Plagiarisme Dan Bukan Hasil Karya Orang Lain.

Apabila dikemudian hari ditemukan seluruh atau sebagian penelitian/makalah/tugas akhir tersebut terdapat indikasi plagiarisme, maka saya bersedia menerima sanksi sesuai peraturan dan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sesungguhnya dan untuk dipergunakan sebagaimana mestinya.

Surabaya, 11 Agustus 2020



Muhammad Athma Farhan
NRP. 05211640000051

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, Tuhan Semesta Alam yang telah memberikan kekuatan serta hidayah-Nya kepada penulis sehingga penulis dapat menyelesaikan tugas akhir ini yang merupakan salah satu syarat kelulusan di Departemen Sistem Informasi Fakultas Teknologi Elektro dan Informatika Cerdas Institut Teknologi Sepuluh Nopember Surabaya.

Terima kasih penulis sampaikan kepada pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, dan bantuan baik berupa materiil maupun moril demi tercapainya tujuan pembuatan tugas akhir ini. Secara khusus penulis akan menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada :

1. Segenap keluarga besar terutama kedua orang tua dan adik penulis, Bapak Tatang Daryanto, Ibu Sri Andayani, Farah Helga Azzahra, dan Muhammad Ammar Faris yang senantiasa mendoakan, memberikan motivasi dan semangat, sehingga penulis mampu menyelesaikan pendidikan sarjana ini dengan baik.
2. Dr. Mudjahidin, S.T., M.T. selaku Kepala Departemen Sistem Informasi ITS, Bapak Ahmad Mukhlason, S.Kom., M.Sc., Ph.D. selaku Ketua Program Studi Sarjana Departemen Sistem Informasi ITS, serta seluruh dosen pengajar beserta staf dan karyawan Departemen Sistem Informasi ITS selama penulis menjalani perkuliahan.
3. Ibu Feby Artwodini Muqtadiroh S.Kom., MT. sebagai dosen wali penulis selama menempuh pendidikan di Departemen Sistem Informasi ITS.
4. Bapak Dr.Eng. Febriliyan Samopa, S. Kom., M. Kom. selaku dosen pembimbing yang telah banyak meluangkan waktu untuk membimbing, mengarahkan, dan mendukung

dengan memberikan ilmu, petunjuk, dan motivasi dalam penyelesaian tugas akhir ini.

5. Bapak Nisfu Asrul Sani, S. Kom., M.Sc. dan Bapak Bekti Cahyo Hidayanto, S.Si., M. Kom. selaku dosen penguji yang telah memberikan kritik dan saran dalam penyempurnaan tugas akhir ini.
6. Teman-teman Sistem Informasi angkatan 2016 (Artemis) yang senantiasa menemani dan memberikan motivasi bagi penulis selama perkuliahan hingga dapat menyelesaikan tugas akhir ini.
7. Serta seluruh pihak-pihak lain yang tidak dapat disebutkan satu per satu, yang telah banyak membantu penulis selama perkuliahan hingga dapat menyelesaikan Tugas Akhir ini.

Penyusunan laporan tugas akhir ini masih jauh dari kata sempurna, sehingga penulis menerima adanya kritik maupun saran yang membangun untuk perbaikan di masa yang akan datang. Semoga buku tugas akhir ini dapat memberikan manfaat bagi pembaca.

Surabaya, 30 Juni 2020

Penulis

Muhammad Athma Farhan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Pada bagian ini akan diuraikan proses identifikasi masalah penelitian yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan relevansi terhadap pengerjaan tugas akhir. Berdasarkan uraian pada bagian ini, harapannya gambaran umum permasalahan dan pemecahan masalah pada tugas akhir ini dapat dipahami.

1.1 Latar Belakang

Seiring dengan perkembangan zaman yang semakin modern, banyak kaum urban yang pindah ke kota untuk mencari pekerjaan yang lebih bagus dan meningkatkan kesejahteraannya. Tempat tinggal di perkotaan sangat dibutuhkan untuk membangun rumah yang layak dihuni untuk keluarga-keluarga baru. Namun berdasarkan data yang didapatkan dari BPS, luas lahan pertanian kian menurun. Pada tahun 2017 luas lahan masih 7,75 hektare, lalu tahun 2018 mengalami penurunan hingga tinggal 7,1 hektare [1]. Banyak lahan pertanian yang dikonversi sehingga dapat menjadi tersisa hanya 7,1 hektare saja, sedangkan populasi khususnya di perkotaan besar semakin meningkat seiring dengan kebutuhan akan pangan. Dibutuhkan teknik solusi pertanian yang modern untuk mengatasi permasalahan kebutuhan akan pangan.

Cara budidaya tanaman menggunakan teknik hidroponik adalah salah satu solusi untuk memperbaiki masalah kebutuhan pangan pada permasalahan urban [2]. Hidroponik adalah salah satu cara budidaya tanaman yang dapat dilakukan di lahan yang sempit, menggunakan sedikit air, dan cocok digunakan pada tanah yang tandus. Tanaman hidroponik dapat menghasilkan hasil yang lebih banyak dibandingkan tanaman konvensional. Daun yang lebih besar dan lebat dikarenakan akar pada tanaman tersebut langsung menyentuh air, sehingga mempermudah akar dalam menyerap air untuk tumbuh dan kembang tanaman. Beberapa faktor penting yang mempengaruhi tumbuh dan kembangnya tanaman hidroponik adalah berdasarkan satuan *total dissolved*

solids, pH, dan temperatur. Sama seperti tanaman konvensional lainnya, tanaman hidroponik juga membutuhkan zat hara nutrisi makro dan nutrisi mikro. Perbedaan dari tanaman konvensional, tanaman hidroponik menggunakan nutrisi yang berasal dari bubuk nutrisi yang biasa disebut nutrisi hidroponik A dan B. Akan tetapi, pada budidaya tanaman hidroponik yang biasa saja masih dilakukan pemantauan dan pengendalian secara manual. Terlalu tinggi atau terlalu rendah kandungan nutrisi juga dapat mengakibatkan menurun kualitas tanaman yang akan dipanen nantinya.

Dengan meningkatnya penggunaan dan perkembangan internet dan gagasan baru dalam bidang pertanian maka terciptalah smart farming. *Smart farming*, bagian dari *precision agriculture*, pertanian yang memanfaatkan perangkat keras seperti sensor dan aktuator yang terhubung dengan jaringan, seperti menggunakan *internet of things* (*IoT*), untuk membuat keputusan dalam melakukan pengendalian tanaman untuk meningkatkan kuantitas tanaman, kualitas tanaman, serta efisiensi dalam penggunaan sumber daya alam [3]. *Smart farming* ini layak diimplementasikan pada budidaya tanaman hidroponik karena teknik menggukan hidroponik ini dilakukan pada lingkungan yang tidak ekstrim. Melakukan proses pemantauan pun mudah hanya dengan mengakses website dan proses pengendalian otomatis menggunakan aktuator yang terhubung dengan sensor.

Masalah dan tantangan yang dihadapi oleh *smart farming* seperti yang disebutkan dalam [4] yaitu adalah *the right source*, *the right place*, *the right time*, dan *the right amount*. *The right source* yang berarti sumber nutrisi yang tepat sangat dibutuhkan pada tumbuh dan kembangnya tanaman, campuran susunan nutrisi hidroponik harus berasal dari sumber yang tepat juga. *The right pla*ce yang berarti tempat yang tepat juga dibutuhkan dalam mengimplementasi smart farming. Tempat yang baik adalah lahan tanam yang datar agar air yang mengalir pada pipa-pipa sistem hidroponik tidak tumpah. *The right time* yang berarti waktu yang tepat dalam pemberian nutrisi hidroponik pada

larutan yang berdasarkan sensor yang *real-time*, masalah yang sering terjadi adalah sensor tidak menampilkan data *real-time* dikarenakan kurangnya perawatan pada sensor sehingga mengakibatkan kesalahan pada proses pengendalian. Kemudian *the right amount* yang berarti jumlah yang tepat dalam pemberian nutrisi pada larutan hidroponik.

Perbedaan hidroponik yang sudah menggunakan teknologi *smart farming* dengan yang biasa adalah petani hidroponik tidak perlu memeriksa kandungan *total dissolved solids*, pH, dan temperatur berkala secara manual sehingga memudahkan petani dalam menggunakan teknik budidaya tanaman hidroponik ini. Selain itu hidroponik dengan *smart farming* ini memudahkan petani dalam menggunakan sistem hidroponiknya dikarenakan sedikitnya orang yang memahami cara budidaya tanaman menggunakan hidroponik.

Arduino sebagai mikrokontroler dihubungkan ke sensor *total dissolved solids* untuk mengetahui kadar nutrisi larutan hidroponik, sensor pH, dan temperatur. Sensor-sensor tersebut digunakan untuk mengetahui kondisi terkini yang berguna untuk aktuator dalam melakukan proses pengendalian pada sistem hidroponik. Data-data kondisi hidroponik ini kemudian dikirimkan ke aplikasi *REST API* melalui internet. *REST API* sangat dibutuhkan untuk menjembatani antara Arduino dengan aplikasi agar dapat saling berkomunikasi.

Dengan demikian, penelitian tugas akhir ini diharapkan dapat menghasilkan sistem *smart farming* hidroponik yang dapat melakukan proses pemantauan dan pengendalian secara otomatis menggunakan sensor *total dissolved solids*, pH, dan temperatur.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, rumusan masalah pada penelitian tugas akhir ini adalah sebagai berikut:

1. Bagaimana menggunakan sensor *total dissolved solids*, pH, dan temperatur pada larutan hidroponik?
2. Bagaimana menggunakan aktuator untuk mengendalikan indikator *total dissolved solids*, dan pH pada larutan hidroponik?
3. Bagaimana membuat sistem yang menghubungkan sensor dan aktuator dengan Arduino untuk melakukan pengendalian otomatis?
4. Bagaimana membangun API yang dapat terhubung dengan tampilan antar muka untuk proses pemantauan?
5. Bagaimana membuat tampilan antarmuka yang dapat menampilkan visualisasi untuk proses pemantauan?

1.3 Batasan Masalah

Berdasarkan permasalahan yang telah diuraikan sebelumnya, adapun batasan masalah dalam tugas akhir ini adalah sebagai berikut :

1. Perancangan sistem merupakan prototipe yang diuji coba pada budidaya tanaman menggunakan teknik hidroponik di dalam ruangan yang tidak terkena sinar matahari secara langsung.
2. Indikator yang diukur adalah jumlah partikel terlarut, pH, dan temperatur.
3. Antarmuka yang dibuat untuk menampilkan data dibuat dalam bentuk website.
4. Perancangan API dan antarmuka menggunakan bahasa pemrograman PHP.
5. Air yang digunakan adalah air yang memiliki kandungan total dissolved solids mendekati 0 ppm.
6. Tanaman yang digunakan untuk budidaya adalah *Brassica rapa*.

1.4 Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah yang telah dijelaskan sebelumnya, tujuan dari penelitian tugas akhir ini adalah mengembangkan prototipe sistem smart farming hidroponik yang berfungsi untuk mengumpulkan data yang

diambil oleh sensor pada sistem hidroponik, membuat sistem pengendalian hidroponik secara otomatis, pembuatan API, dan antarmuka berbasis web untuk menampilkan riwayat hasil pengolahan data yang didapatkan dari sensor.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dapat diperoleh dari penelitian tugas akhir ini adalah sebagai berikut:

1. Menjadi panduan bagi petani hidroponik yang akan mengimplementasikan sistem *smart farming* hidroponik berbasis Arduino menggunakan sensor *total dissolved solids*, pH, dan temperatur pada sistem hidroponik yang sudah ada untuk membantu melakukan proses pemantauan dan pengendalian kondisi hidroponik.
2. Membantu individu dan atau kelompok untuk mengembangkan sistem *smart farming* hidroponik berbasis Arduino menggunakan sensor *total dissolved solids*, pH, dan temperatur.
3. Menjadi referensi untuk melakukan pengembangan pada bidang *Internet of Things*.

1.6 Relevansi

Tugas akhir ini disusun untuk memenuhi salah satu syarat kelulusan sebagai Sarjana Komputer di Departemen Sistem Informasi Fakultas Teknologi Informasi dan Komunikasi (FTIK) Institut Teknologi Sepuluh Nopember (ITS). Tugas akhir ini sesuai dengan penerapan mata kuliah dari laboratorium Infrastruktur dan Keamanan Teknologi Informasi (IKTI) Departemen Sistem Informasi FTIK ITS, yaitu Internet Untuk Segala. Selain itu, juga berkaitan dengan mata kuliah pada Departemen Sistem Informasi FTIK ITS, yaitu Pemrograman Web, Sistem Basis Data, Teknologi Basis Data, dan Rancang Bangun Perangkat Lunak.

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Pada bagian ini akan dijelaskan tentang studi literatur dari penelitian-penelitian sebelumnya dan dasar teori yang dijadikan acuan atau landasan dalam pengerjaan tugas akhir ini.

2.1 Studi Literatur

Pencarian penelitian-penelitian yang sudah dilakukan dijadikan sebagai referensi dan menjadi pendukung untuk melakukan pengerjaan pengembangan perangkat lunak pada penelitian tugas akhir ini, seperti yang dijelaskan pada Tabel 2.1.

Tabel 2. 1 Studi Literatur

1) An Automated Hydroponics System Based on Mobile Application [5]
Penulis; Tahun K. Kunyanuth, A. Udomlux, K. Nutthaphol; 2019
Pembahasan Penelitian ini menyajikan pengembangan sistem hidroponik otomatis berbasis aplikasi *mobile* menggunakan *Arduino Wemos D1* sebagai mikrokontroler yang memungkinkan dapat terhubung dengan internet melalui jaringan *wi-fi*. Perangkat keras dalam penelitian ini yaitu: *Wemos D1*, modul *relay*, modul sensor ultrasonik, sensor pH, katup solenoid, dan *ambient light sensor*. Untuk menghubungkan mikrokontroller dengan *Cloud* yang berisikan *database*, penelitian ini menggunakan *Message Queuing Telemetry Transport* (MQTT). MQTT sangat cocok digunakan pada alat sensor yang memiliki prosesor kecil dengan penggunaan jaringan jarak jauh yang hanya membutuhkan daya konsumsi rendah dan *bandwidth* terbatas. *Database* yang digunakan adalah *Firebase* karena *database* ini dapat tersinkronisasi secara *real-time*. Otomasi yang dilakukan dibagi menjadi empat bagian. Bagian pertama adalah pengaturan suhu dan kelembaban. Saat sistem mendapatkan nilai dari sensor suhu dan kelembaban. Pompa air akan dialirkan

lebih cepat yang berguna untuk mendinginkan suhu sekitar. Bagian kedua adalah pengaturan intensitas cahaya dengan cara mematikan dan menyalakan lampu LED, sistem akan mengoperasikan modul *relay*. Bagian ketiga adalah pengaturan pH dengan cara sensor mendeteksi pH dan sistem akan mengalirkan larutan yang sesuai untuk mencapai pH yang diinginkan. Bagian keempat adalah pengaturan ketinggian air. Jika sistem mendeteksi nilai yang rendah, air akan dialirkan ke tangki air sehingga tinggi air yang diinginkan tercapai.

Keterkaitan

Penelitian tugas akhir yang dilakukan berkaitan dengan pengembangan *Automated Hydroponics System* menggunakan *Wemos D1*, mikrokontroler Arduino yang terintegrasi dengan *wireless module*, berguna mengirim data yang akan ditampilkan dari sensor ke aplikasi mobile atau website kepada pengguna untuk mengirim data dari sensor ke aplikasi yang akan ditampilkan kepada pengguna. Pengaturan secara otomatis seperti suhu dan kelembaban, pH, tingkat cahaya, dan tingkat ketinggian air. *Monitoring* status kondisi terkini melalui aplikasi *mobile* atau *website*. Perbedaan dari penelitian yang akan dilakukan adalah menggunakan REST API, tidak menggunakan MQTT.

2) Implementation IoT in System Monitoring Hydroponic Plant Water Circulation and Control [6]

Penulis; Tahun

N. Usman, P. Arief, L. Gilang, R. Erfan, P. Hendra; 2018

Pembahasan

Penenlitian ini bertujuan melakukan pemantauan dan pengendalian sirkulasi air pada budidaya tanaman hidroponik. Teknik yang digunakan pada penelitian ini menggunakan teknik *deep flow technique* yang berarti akar tanaman tenggelam di dalam aliran air. Indikator yang dilakukan pemantauan dan pengendalian adalah suhu udara, kelembaban udara, ketinggian air, dan pH. Alat yang digunakan adalah sensor DHT 11 untuk mendeteksi suhu udara dan kelembaban udara, sensor HC-SR04 untuk mendeteksi ketinggian air menggunakan ultrasonik, sensor SEN0161 untuk mendeteksi nilai pH dari larutan hidroponik. Ketiga alat itu kemudian dihubungkan dengan Raspberry Pi melalui alat MCP3008 ADC untuk mengonversi sinyal analog menjadi sinyal digital. Untuk

melakukan pengendalian secara otomatis, penelitian ini menggunakan metode *fuzzy*. Untuk melakukan pemantauan, penelitian ini menggunakan *website* namun tidak dijelaskan secara detil tentang bagaimana *website* tersebut dibuat.

Keterkaitan

Penelitian tugas akhir yang dilakukan berkaitan dengan penggunaan sensor, ketinggian air, pH, kelembaban udara, dan suhu udara sebagai perangkat keras untuk mendeteksi indikator-indikator budidaya tanaman hidroponik. Perbedaan dari penelitian ini dengan penilitian tugas akhir yang dilakukan adalah penggunaan Raspberry Pi dan metode *fuzzy*. Penelitian tugas akhir yang akan dilakukan menggunakan Arduino Mega dan menambahkan sensor suhu air agar air dari suhu larutan hidroponik yang mengalir dapat dipantau. Perbedaan dari penelitian yang akan dilakukan adalah menggunakan Arduino, tidak menggunakan Raspberry Pi dan untuk pengendalian tidak menggunakan metode *fuzzy* namun menggunakan metode perbandingan nilai data sensor dengan nilai data normal yang dikehendaki.

3) Rancang Bangun Sistem Pendeteksi Kesuburan Tanah Menggunakan Arduino Dengan Sensor Electrical Conductivity, Temperatur, Kelembapan Tanah, dan pH [7]

Penulis; Tahun

H. A. Jabbar ; 2019

Pembahasan

Penelitian ini bertujuan untuk memanfaatkan teknologi *Internet of Things* (*IoT)* dan sensor untuk mendukung sektor pertanian dengan menawarkan sistem pengukuran kondisi keadaan tanah pertanian yang dapat dilakukan dengan menggunakan *Arduino* sebagai mikrokontroller. *Arduino* ini yang akan disambungkan dengan *sensor,* sehingga hasil pembacaan *sensor* dapat diolah menjadi informasi yang berguna. Koneksi jaringan yang digunakan untuk mengirimkan data adalah menggunakan *Wi-Fi* dengan modul *ESP8266. Arduino* dikomunikasikan dengan aplikasi melalui pengiriman JSON ke API. API akan berperan sebagai

pengolah data yang telah diterima sebelum ditampilkan ke aplikasi yang akan menampilkan informasi kepada penggunanya. Adapun aplikasi yang digunakan untuk menampilkan hasil informasinya berbasis *website.*
Keterkaitan Penelitian tugas akhir yang dilakukan berkaitan dengan proses pengiriman dan pengolahan data dari *Arduino* hingga ditampilkannya melalui *web.* Sistem menggunakan modul *Wi-Fi ESP8266* sebagai penghubung koneksi jaringan yang dikomunikasikan dengan aplikasi melalui pengiriman *JSON* ke *API. API* akan berperan sebagai pengolah data yang telah diterima sebelum ditampilkan ke aplikasi *web*. Perbedaan dengan penelitian yang dilakukan adalah dengan menggunakan media tanam air yaitu hidroponik dan menambahkan fitur baru yaitu pengendalian otomatis pada sistem hidroponik yang akan dibuat.

2.2 Dasar Teori

2.2.1 Arduino

Arduino adalah mikrokontroler *open source* yang dapat dengan mudah diprogram, dihapus dan diprogram ulang kapan saja. Platform Arduino dirancang untuk membuat perangkat yang berinteraksi dengan lingkungan menggunakan sensor dan aktuator. Arduino juga dapat dijadikan sebagai mikrokontroler dengan mengambil input dari sensor dan mengendalikan output ke aktuator untuk berbagai perangkat elektronik. Arduino menggunakan perangkat keras yaitu Arduino board development salah satunya yaitu Arduino Mega yang dapat dijalankan menggunakan bahasa pemrograman C, C++, dan Arduino. Untuk mengembangkan kode perangkat lunak, Arduino menggunakan Arduino IDE (*Integrated Development Environment*) [8].

Gambar 2. 1 Arduino Mega

2.2.2 Hidroponik, Nutrisi A, dan Nutrisi B

Hidroponik adalah bagian dari hidrokultur, yang merupakan metode menanam tanaman tanpa tanah menggunakan larutan nutrisi mineral yang dilarutkan dalam air. Larutan nutrisi mineral hidroponik yang sering digunakan dan beredar umum adalah nutrisi A dan nutrisi B [9]. Nutrisi A hidroponik terdiri dari nitrat (NO_3^-), kalsium (Ca), dan mikronutrien lain seperti boron (B), tembaga (Cu), besi (Fe), kalium (K), mangan (Mn), molibdenum (Mo), dan seng (Zn). Nutrisi B hidroponik terdiri dari fosfat (PO_4^{3-}), sulfat (SO_4^{2-}), dan magnesium (Mg) [10].

2.2.3 Total Dissolved Solids dan Parts Per Million

Total dissolved solids (TDS) yang terdiri dari garam anorganik (terutama kalsium, magnesium, kalium, natrium, bikarbonat, klorida, dan sulfat) dan sejumlah kecil bahan organik yang larut dalam air [11]. Lazimnya dalam mengukur mengukur TDS menggunakan satuan *parts per million* (ppm). Istilah ppm, yang berarti 10^{-6} nilai relatif atau 1 bagian dalam 10^6 atau *parts per million* juga digunakan dalam pengukuran TDS. Penjelasan ini sejalan dengan arti persen sebagai *parts per hundred* [12].

2.2.4 pH (Power of Hydrogen)

Skala pH bersifat logaritmik dan berbanding terbalik dengan keberadaan konsentrasi ion hidrogen dalam larutan (pH yang lebih rendah menunjukkan konsentrasi ion hidrogen yang lebih tinggi). Hal ini dikarenakan rumus yang digunakan untuk menghitung pH mendekati negatif dari logaritma basa 10

konsentrasi molar ion hidrogen dalam larutan. Lebih tepatnya pH adalah negatif dari logaritma basis 10 dari aktivitas ion hidrogen [13].

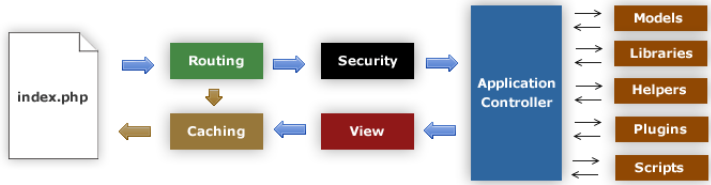
2.2.5 Temperatur

Kuantitas yang ditetapkan sebagai suhu termodinamika didefinisikan oleh hukum termodinamika; ini ditunjukkan oleh simbol T, dan memiliki satuan kelvin, simbol K. Satuan suhu termodinamika didefinisikan sebagai fraksi 1 / 273,16 dari termodinamika suhu titik tripel air. Dalam hal ini biasanya sering dilakukan mengekspresikan suhu dengan selisih dari 273,15 K, nilai untuk titik beku es. Suhu T termodinamika ini dinyatakan dikenal sebagai suhu t Celcius, yang didefinisikan oleh persamaan [14]:

$$\frac{t}{^{\circ}\mathrm{C}} = \frac{T}{\mathrm{K}} - 273.151 \quad (1)$$

2.2.6 CodeIgniter

CodeIgniter adalah *toolkit* kerangka kerja untuk membangun aplikasi web menggunakan PHP dengan cepat. Tujuannya adalah memungkinkan pengembangan proyek lebih cepat daripada menulis baris kode dari awal (*native*), dengan tersedianya pilihan *libraries*, antarmuka yang sederhana dan struktur logis untuk mengakses *libraries*. CodeIgniter menggunakan *pattern* Model-View-Controller, yang dapat melakukan pemisahan antara logika proses kode dan tampilan presentasi *website*. *Model* adalah objek dengan parameter tertentu, *View* adalah tampilan depan *website*, dan *Controller* adalah pusat dari logika yang mengatur alur proses dari *website*. Pola pengembangan seperti ini sangat cocok dengan penggunaan *template* karena tampilan presentasi *website* yang terpisah memudahkan para *designer* untuk merancang tampilan desain. Sistem ini juga dapat ditambahkan dengan berbagai *custom add-on* yang dapat dikonfigurasi secara manual [15].



Gambar 2. 2 Flow Chart Aplikasi CodeIgniter [16]

2.2.7 MySQL

MySQL adalah sistem manajemen basis data *open-source.* - Database adalah kumpulan data yang terstruktur. Data yang disimpan dapat berbagai macam. Sekumpulan data tersebut akan dikelompokkan menjadi sejumlah besar informasi yang berguna bagi perusahaan. MySQL menggunakan basis data relasional yang menyimpan data dalam tabel terpisah daripada meletakkan semua data dalam satu ruang penyimpanan besar. Struktur basis data disusun berdasarkan tabel-tabel lalu dijadikan *file* fisik yang terpisah agar dapat dilakukannya optimalisasi [17].

2.2.8 REST API

Dalam disertasi [18], Roy Fielding memperkenalkan dan mendefinisikan istilah "*representational state transfer*". *Representational state transfer* (*REST*) adalah salah satu metode arsitektur perangkat lunak yang menjelaskan beberapa batasan-batasan yang akan digunakan untuk membuat *web services*. Fielding menjelaskan prinsip-prinsip *REST* yang dikenal sebagai "*HTTP object model*" pada awal 1994. Standar HTTP 1.1 dan *Uniform Resource Identifiers* (*URI*) dibuat berdasarkan rancangan *REST API* ini.

Berikut adalah macam-macam bagaimana metode HTTP 1.1 biasanya digunakan dalam membuat *REST API* berdasarkan [19]:

2.2.8.1 GET
URI ini akan mengambil semua anggota dari *object model* (*member resources*) dari *database* (*collection resources*) yang diletakkan pada sekumpulan *array* sebagai *response body*.

2.2.8.2 POST
Membuat sebuah *object model* (*member resources*) di *database* (*collection resources*) menggunakan metode *POST* pada *request body*.

2.2.8.3 PUT
Menggantikan seluruh representasi *object model* (*member resources*) di *database* (*collection resources*) berdasarkan metode *PUT* pada *request body.*

2.2.8.4 PATCH
Memperbarui seluruh representasi *object model* (*member resources*) di *database* (*collection resources*) berdasarkan metode *PATCH* pada *request body*. Perbedaan dari metode *PUT* yaitu metode *PATCH* hanya memperbarui sebagian *entity* dan hanya memberi parameter *entity* yang mana yang akan diperbarui pada *request body*. Metode *PUT* akan menggantikan sebuah representasi *object model* (*member resources*) dan memberi seluruh parameter pada *request body*.

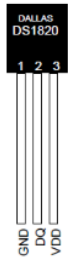
2.2.8.5 DELETE
Menghapus seluruh representasi *object model* (*member resources*) di *database* (*collection resources*) berdasarkan metode *DELETE* pada *request body.*

2.2.9 Pertimbangan Pemilihan Sensor

Pemilihan sensor disesuaikan dengan indikator yang ingin diukur, yaitu pH larutan, nilai *total dissolved solids*, *temperature*, dan tingkat ketinggian air. Jenis sensor yang akan digunakan antara lain:

2.2.9.1 Sensor Suhu Air DS18B20

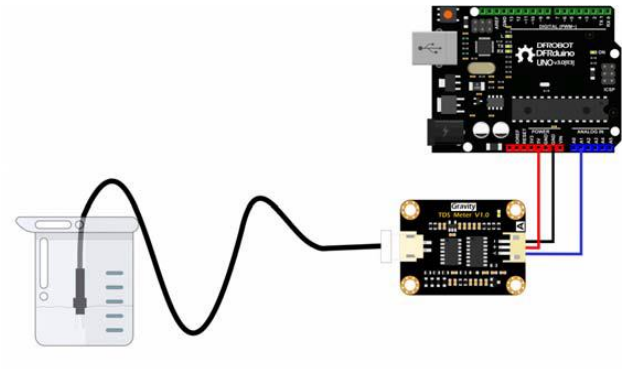
Sensor DS18B20 adalah sensor yang dapat mendeteksi suhu di dalam air menggunakan sinyal digital yang dihubungkan pada mikrokontroler. Tegangan yang dibutuhkan untuk menggunakan sensor ini berkisar antara 3.0V hingga 5V. Sensor ini dapat bekerja pada suhu (temperatur) dari -55°C hingga 125°C. Keakuratan data yang diambil oleh sensor ini adalah ±0.5°C pada suhu -10°C hingga 85°C. Rata-rata sensor ini membutuhkan 10ms untuk mengambil data kondisi suhu untuk dikirimkan melalui sinyal ke Arduino [21]. Sensor ini dibutuhkan untuk melakukan pemantauan yang mendeteksi suhu pada larutan hidroponik.



Gambar 2. 3 Sensor Suhu Air DS18B20

2.2.9.2 Sensor Total Dissolved Solids DFRobot SKU SEN0244

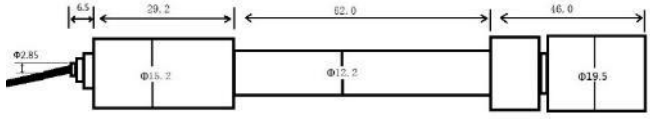
Sensor *Total Dissolved Solids* DFRobot SKU SEN0244 adalah sensor yang dapat mendeteksi jumlah zat padat yang larut pada larutan menggunakan sinyal analog yang dihubungkan pada mikrokontroler. Sensor ini membutuhkan tegangan sebesar 3.3V hingga 5.5V. Sensor ini dapat mengukur *TDS* kisaran 0ppm hingga 1000ppm dengan keakuratan pembacaan sensor sebesar ±10%FS (*Full Scale*) [22]. Sensor ini dibutuhkan dalam pemantauan dan pengendalian nutrisi hidroponik.



Gambar 2. 4 Sensor *TDS* SEN0244 yang terhubung dengan Arduino

2.2.9.3 pH *meter* DFRobot SKU SEN0161

pH meter DFRobot SKU SEN0161 adalah sensor pendeteksi pH menggunakan sinyal analog yang dihubungkan pada mikrokontroler. Tegangan yang dibutuhkan untuk menggunakan sensor ini yaitu 5V. Sensor ini dapat bekerja pada suhu (temperatur) dari 0°C hingga 60°C. Keakuratan data yang diambil oleh sensor ini adalah ±0.1pH pada suhu 25°C [23]. Sensor ini dibutuhkan untuk melakukan pemantauan dan pengendalian pada larutan hidroponik. Tingkat keasaman pada larutan akan mempengaruhi tumbuh dan kembangnya tanaman.



Gambar 2. 5 Dimensi Sensor pH SEN0161 [23]

2.2.9.4 Sensor Ultrasonik HC-SR04

Sensor ultrasonik berguna untuk mendeteksi ketinggian air pada tanki larutan hidroponik menggunakan sinyal digital yang dihubungkan dengan mikrokontroler. Tegangan yang dibutuhkan untuk menggunakan sensor ini yaitu 5V. Sensor ini dapat mengukur jarak dari 2cm hingga 4m. Keakuratan data yang diambil oleh sensor ini adalah ±3mm [24]. Sensor ini

dibutuhkan untuk melakukan pemantauan dan pengendalian pada larutan hidroponik. Sensor ini dibutuhkan dalam pemantauan dan pengendalian ketinggian air atau banyaknya air yang tersisa pada tangki hidroponik.

Gambar 2. 6 Sensor Ultrasonik HC-SR04

2.2.10 Pertimbangan Pemilihan Aktuator

Pemilihan aktuator disesuaikan dengan indikator yang ingin dikendalikan, yaitu nilai *total dissolved solids* dan tingkat ketinggian air. Jenis aktuator yang akan digunakan antara lain:

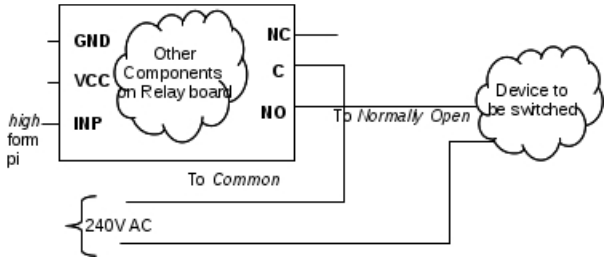
2.2.10.1 Katup Solenoida

Katup solenoida adalah aktuator pada salah satu instrumen otomasiyang memiliki manfaat, seperti proses *switching* yang cepat dan aman, keandalan yang tinggi, umur penggunaan yang lama, sangat cocok bahan yang akan digunakan, memiliki kontrol daya yang rendah, dan desain yang ringkas. Katup solenoida ini sudah banyak digunakan pada bidang seperti industri, pertanian, transportasi, penerbangan, pariwisata, dan ruangan publik [25]. Alat ini memiliki tegangan yang cukup tinggi. Oleh karena itu, pengendalian yang dilakukan dengan cara menghubungkan relai terlebih dahulu, tidak dengan cara menghubungkan ke mikrokontroler Arduino secara langsung. Alat ini berfungsi untuk mengisi air, pH, dan larutan hidroponik.

Gambar 2. 7 Katup Solenoida

2.2.10.2 Relai

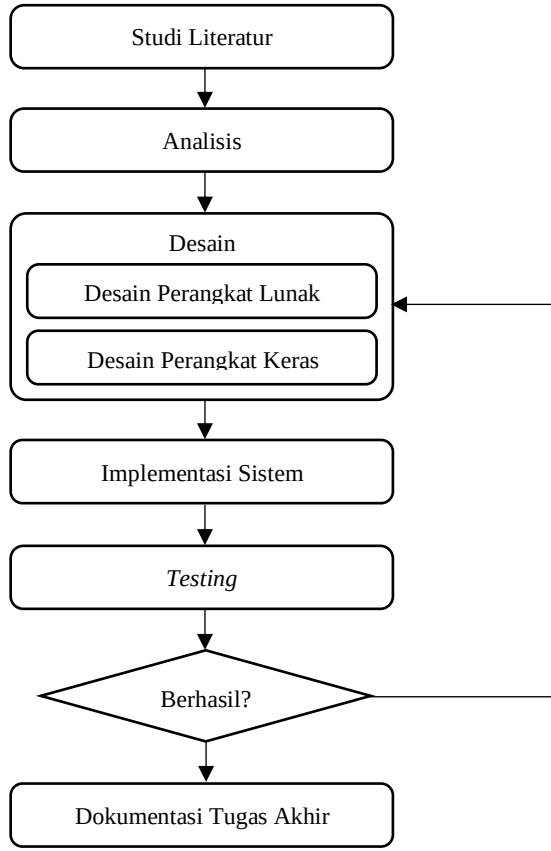
Relai adalah komponen aktuator elekromekanik yang dapat memutuskan atau menghubungkan arus listrik kepada komponen lain yang memiliki perbedaan tegangan. Rangkaian ini digunakan untuk mengontrol satu alat listrik 240V langsung dari mikrokontroler atau sirkuit tegangan rendah. Ada tiga pin pada papan relay yaitu *normally open* (NO), *normally closed* (NC) dan *common* (C). Pin *common* terhubung pada pin NC saat relai mati dan ke pin NO saat relay menyala. "VCC" dan pin "GND" dari relay terhubung ke sumber 5V dan *ground* (GND) masing-masing [26]. Alat ini berfungsi sebagai memutuskan dan menghubungkan katup solenoid agar dapat dikendalikan menggunakan Arduino.



Gambar 2. 8 Relai yang memiliki 1 *channel* [16]

BAB III
METODOLOGI

Pada bagian metodologi ini akan menjelaskan mengenai tahapan langkah yang dilakukan dalam tugas akhir menggunakan *software development life cycle* (SDLC) model *waterfall* yang disesuaikan beserta deskripsi dan penjelasannya.



Gambar 3. 1 Tahapan Pengerjaan Tugas Akhir

3.1 Studi Literatur

Pada tahap ini dilakukan pengumpulan literatur yang mendukung dalam pengerjaan tugas akhir ini. Tugas akhir ini menggunakan literatur yang berisi tentang penjelasan konsep-konsep dasar atau penelitian sebelumnya yang terkait dengan tugas akhir ini dalam bentuk buku, jurnal, ataupun *website*. Studi literatur berguna dalam hal pemahaman terhadap konsep-konsep umum yang akan diterapkan pada tugas akhir serta menjelaskan perbedaan dari penelitian sebelumnya yang telah dilakukan. Luaran dari proses studi literatur ini adalah pengetahuan konsep tentang sistem yang akan dibuat.

3.2 Analisis

Seluruh Berdasarkan konsep pengetahuan yang sudah didapatkan dari tahap studi literatur akan digunakan sebagai panduan tahapan dalam melakukan analisis kebutuhan sistem. Sehingga keluaran dalam tahapan ini adalah terbentuknya spesifikasi dari sistem perangkat lunak dan perangkat keras. Adapun spesifikasi sistem yang akan dibangun sebagai berikut:

3.2.1 Aplikasi

- Dapat menerima dan menyimpan data *total dissolved solids*, pH, dan temperatur yang dikirim dari Arduino ke *database*
- Dapat menampilkan data visualisasi *total dissolved solids*, pH, dan temperatur
- Dapat mengatur konfigurasi batasan normal variabel tiap indikator *total dissolved solids*, pH, dan temperatur
- Dapat menghapus atau menambah alat Arduino yang terhubung

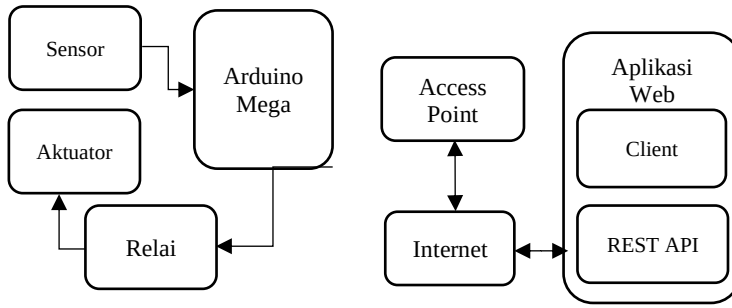
3.2.2 Arduino

- Adaw Dapat menerima nilai pembacaan sensor t*otal dissolved solids*, pH, dan temperatur
- Dapat mengirim data hasil pembacaan sensor ke aplikasi web *API*
- Dapat terhubung dengan internet menggunakan *access point* wi-fi

- Dapat mengendalikan aktuator total dissolved solids, pH, dan temperature secara otomatis

3.3 Desain

Pada tahap ini dilakukan desain sistem yang akan dibuat berdasarkan spesifikasi sistem yang dibutuhkan yang telah dibuat pada tahap analisis. Keluaran dari tahapan ini adalah terbentuknya hasil desain dari sistem. Tahap desain akan dibagi menjadi dua yaitu tahap desain perangkat lunak dan perangkat keras.



Gambar 3. 2 Perancangan Desain Sistem Keseluruhan

Pada gambar 3.2 dijelaskan bagaimana sistem bekerja mulai dari penerimaan data dari pembacaan sensor, lalu aplikasi web menerima data dari pembacaan sensor, hingga Arduino melakukan pengendalian aktuator yang berhubungan dengan sensor masing-masing. Untuk indikator *total dissolved solids*, batas normal dari aplikasi web dikirim melalui *API* lalu diterima oleh Arduino, sensor *TDS* akan melakukan pembacaan untuk dibandingkan dengan batas normal yang diterima dari aplikasi web sebelumnya, jika data pembacaan sensor kurang dari batas normal maka katup solenoid yang berisi nutrisi hidroponik A dan B akan dialirkan menuju tangki larutan hidroponik yang mengalir. Untuk indikator pH akan dilakukan proses yang sama, perbedannya jika pH akan dialirkan larutan asam atau basa ke larutan hidroponik yang mengalir agar mendapatkan nilai yang sesuai dengan batas normal. Untuk indikator suhu yang dilakukan pengendalian yaitu suhu udara sekitar menggunakan

kipas, suhu air hanya dilakukan pemantauan saja. Pembacaan data seluruh sensor akan dikirimkan ke aplikasi web melalui *API* untuk ditampilkan pada tampilan antarmuka *dashboard website*.

3.3.1 Desain Perangkat Lunak

Pada tahap ini dilakukan proses desain pada perangkat lunak yang akan menjelaskan bagaimana alur proses dari aplikasi web berupa desain masing-masing *endpoint API*, desain antarmuka web menggunakan dashboard, dan desain infrastruktur aplikasi web.

3.3.2 Desain Perangkat Keras

Pada Pada tahap ini dilakukan proses desain pada perangkat keras mikrokontroler Arduino menggunakan berbagai sensor dan aktuator. Desain perangkat keras yang akan dibuat adalah desain program yang akan diunggah pada Arduino, rangkaian elektronik yang akan dibuat, serta alur proses bagaimana sistem *smart farming* hidroponik bekerja.

3.4 Implementasi Sistem

Pada tahapan implementasi akan dilakukan proses perakitan sensor dan aktuator pada Arduino pada sistem hidroponik. Untuk mendapatkan nilai sensor yang akurat dibutuhkan kalibrasi terhadap masing-masing sensor dalam merangkai komponen-komponen perangkat keras. Proses kalibrasi dilakukan dengan melakukan pencocokan terhadap hasil pembacaan dari sensor Arduino dengan alat pertanian yang memiliki fungsi yang sama. Cara melakukan kalibrasi masing-masing sensor agar akurat adalah dilakukan dengan cara mencocokkan pengukuran dengan alat pertanian yang sejenis yang sesuai dengan indikator yang akan diukur. Hasil dari proses kalibrasi ini adalah ketepatan dari pembacaan sensor Arduino yang sesuai dengan alat pertanian yang sudah ada. Jika pembacaan sensor yang terhubung dengan Arduino dengan alat pertanian hidroponik yang sejenis sudah cocok, maka proses

kalibrasi berhasil dilakukan. Dalam melakukan proses kalibrasi akan ada beberapa cara dalam melakukan kalibrasi, antara lain :

- Memberikan nilai tertinggi terhadap pembacaan sensor yang dilakukan selama lima detik.
- Memberikan nilai terendah terhadap pembacaan sensor yang dilakukan selama lima detik.
- Melakukan pembacaan sensor secara normal selama lima detik.
- Melakukan pembacaan sensor terhadap sepuluh sampel larutan hidroponik.
- Sehingga pada tahapan ini akan terbentuk rangkaian perangkat keras dari sistem, *API*, serta aplikasi web antarmuka.1

3.5 Testing Aplikasi

Pada tahap testing akan dilakukan pengujian terhadap keseluruhan sistem yang telah dibuat. Pengujian terhadap sistem dilakukan dengan cara melakukan *test case* dimana setiap *case* memiliki skenario yang harus dilakukan. Jika *test case* yang dilakukan terdapat skenario yang tidak sesuai dengan ekspektasi, maka proses pengerjaan kembali ke tahap implementasi dimana di tahap tersebut dilakukan perbaikan *bug* yang ditemukan pada saat *test case* dilakukan. Tahap ini akan selesai jika seluruh *test case* telah berhasil dijalankan tanpa kendala. Luaran dari tahap ini adalah dokumentasi seluruh *test case* yang telah dijalankan.

3.6 Dokumentasi Tugas Akhir

Dokumen tugas akhir ini adalah hasil luaran seluruh tahap proses penelitian yang telah dilakukan sebelumnya, kesimpulan dari penelitian tugas akhir yang telah dikerjakan, serta saran untuk penelitian selanjutnya.

Halaman ini sengaja dikosongkan

BAB IV
PERANCANGAN

Pada bagian ini akan menjelaskan mengenai proses perancangan terhadap sistem yang akan digunakan sebagai acuan dalam proses implementasi.

4.1 Tahap Analisis

Pada tahap awal dari pembuatan sistem dimulai dengan mengidentifikasi seluruh informasi dari sistem yang dibuat. Pengumpulan Informasi dilakukan dengan melakukan wawancara kepada narasumber pegiat hidroponik dan survey langsung ke kebun hidroponik milik narasumber tersebut. Narasumber wawancara adalah Sulihan yang memiliki kebun hidroponik dan jasa pembuatan instalasi hidroponik "Jawara Farm".

Hidroponik adalah salah satu cara menanam di lahan sempit menggunakan media air. Air yang digunakan tidak hanya sekedar air melainkan air yang ditambahkan dengan campuran nutrisi A dan nutrisi B hingga batas *total dissolved solids* yang telah ditentukan. Selain *total dissolved solids*, pH juga mempengaruhi tumbuhnya tanaman hidroponik. Nilai pH berubah relatif naik setiap waktu, kenaikan ini menandakan larutan hidroponik akan semakin bersifat basa. Oleh karena itu butuh pengendalian nilai pH dengan cara menambahkan suatu zat asam.

Sistem dibuat dengan tujuan melakukan pengawasan kualitas pH, *total dissolved* solids, dan temperatur larutan hidroponik dan pengendalian kualitas larutan hidroponik. Tiga parameter tersebut sangat mempengaruhi tumbuhnya tanaman hidroponik. Sistem dapat melakukan pengendalian menggunakan relai yang dihubungkan dengan katup solenoid. Penyimpanan informasi serta visualisasi informasi kualitas larutan hidroponik ini juga menjadi poin penting dalam sistem ini. Untuk dapat menentukan katup solenoid mana saja yang perlu dihidupkan dan dimatikan dibutuhkan informasi kualitas larutan hidroponik yang diambil

dari sensor. Tabel 4.1 menjelaskan pembuatan keputusan yang akan dilakukan jika pH dan *total dissolved solids* melebihi nilai batas atas atau nilai batas bawah yang telah ditetapkan.

Tabel 4. 1 Kondisi Pengambilan Keputusan

Kondisi	Keputusan
pH melebihi nilai batas atas	Menghidupkan katup solenoid melalui relai berisi larutan asam yang terhubung dengan alat untuk menurunkan pH
Nilai t*otal dissolved solids* kurang dari nilai batas bawah	Menghidupkan katup solenoid melalui relai berisi larutan nutrisi A dan B yang terhubung dengan alat untuk menaikkan nilai *total dissolved solids*
Tinggi air kurang dari nilai batas bawah	Menghidupkan katup solenoid melalui relai berisi air yang terhubung dengan alat untuk menambah volume larutan hidroponik hingga ketinggian air mencapai nilai minimum

Sistem terbagi menjadi 3 bagian, yaitu aplikasi web *client* (*frontend)*, aplikasi web *API* (*backend*), dan kode Arduino yang terletak pada perangkat keras. Aplikasi web *client* berguna untuk menampilkan tampilan *dashboard* dan mengambil data dari aplikasi web *API*. Kode yang terletak pada perangkat keras berguna untuk mengambil data dari sensor kemudian dikirimkan ke aplikasi web *API* menggunakan modul *wifi* ESP8266 agar dapat disimpan pada *database* dan berguna untuk mengendalikan temperatur, *total dissolved solids* (TDS), dan pH menggunakan aktuator.

Berdasarkan analisa di atas didapatkan kebutuhan fungsional dan kebutuhan non-fungsional pada Tabel 4. 2 dan Tabel 4. 3 sebagai berikut:

Tabel 4. 2 Kebutuhan Fungsional

No	Kategori	Kebutuhan
1	Perangkat Arduino	Perangkat dapat mengetahui informasi pH, *total dissolved* solids, dan temperatur dengan menggunakan sensor.
2	Perangkat Arduino	Perangkat dapat mengirimkan informasi suhu dan kelembapan secara berkala ke aplikasi web *client* dengan selang waktu.
3	Perangkat Arduino	Perangkat dapat mengatur katup solenoid melalui relai sesuai dengan batas atas dan bawah pH dan *total dissolved solids* yang telah ditentukan pada aplikasi web *client*.
4	Perangkat Arduino	Perangkat dapat menerima informasi konfigurasi sensor yang telah diatur pada aplikasi web *client*.
5	Aplikasi Web	Aplikasi dapat mengelola daftar proyek milik *user* seperti melihat, mengubah, menambahkan, menghapus proyek.
6	Aplikasi Web	Aplikasi dapat mengelola daftar mikrokontroler milik *user* seperti melihat, mengubah, menambahkan, menghapus mikrokontroler.
7	Aplikasi web	Aplikasi dapat menerima informasi pH, *total dissolved* solids, dan temperatur dari tiap mikrokontroler Arduino yang telah terdaftar.
8	Aplikasi web	Aplikasi dapat menyimpan informasi pH, *total dissolved* solids, dan temperatur yang didapat dari mikrokontroler Arduino.

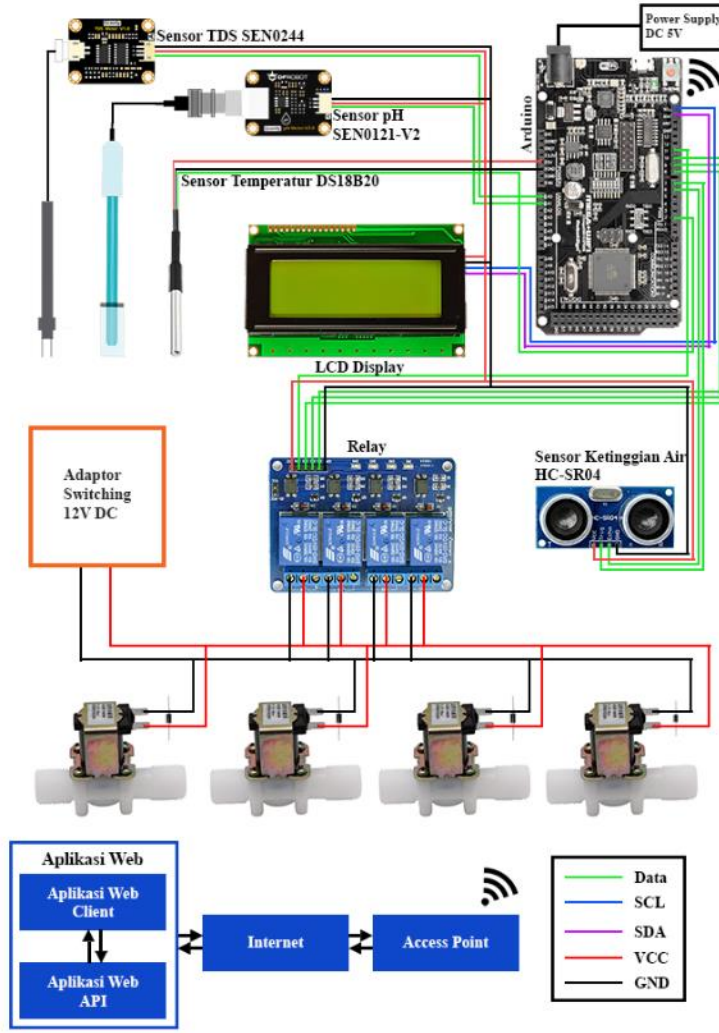
9	Aplikasi web	Aplikasi dapat melakukan visualiasi informasi pH, *total dissolved solids*, dan temperatur dari tiap perangkat.
10	Aplikasi web	Aplikasi dapat mengatur pengaturan tiap sensor pada tiap perangkat Arduino berdasarkan batas bawah dan atas pH, *total dissolved* solids, dan temperatur.
11	Aplikasi web	Aplikasi dapat mengirimkan konfigurasi sensor kepada mikrokontroler Arduino yang terhubung.

Tabel 4. 3 Kebutuhan Non Fungsional

No	Kategori	Kebutuhan
1	Perangkat Arduino	Perangkat dapat terhubung secara otomatis ke jaringan Wi-Fi yang telah ditentukan
2	Perangkat Arduino	Perangkat dapat berjalan dengan stabil
3	Aplikasi web	Aplikasi menerima setiap *request* menggunakan JWT untuk keamanan
5	Aplikasi web	Aplikasi dapat berjalan pada *web browser*
6	Aplikasi web	Aplikasi dapat berjalan dengan stabil

4.2 Desain Sistem

Desain sistem dirancang untuk menjadi acuan pada pengembangan *Smart Farming* hidroponik yang akan dibuat. Desain sistem yang akan dibuat menjelaskan bagimana alur sebuah data dari instalasi hidroponik diambil, lalu dikirimkan ke aplikasi web *API*, kemudian dapat ditampilan pada *dashboard* aplikasi web *client*, hingga dapat mengendalikan keluaran yang diinginkan. Sehingga keluaran pada tahap ini adalah hasil berupa alur dari sistem.



Gambar 4. 1 Desain Sistem Smart Farming

Gambar 4.1 menjelaskan alur data dari instalasi hidroponik berupa temperatur, *total dissolved solids*, dan pH hingga aplikasi web *client*. Terdapat dua pengolahan data. Pengolahan data pertama untuk pemantauan yaitu Arduino mengirimkan *request* POST ke aplikasi web *API* untuk menyimpan data dari

sensor pH (SEN0161-V2), sensor *total dissolved solids* (SEN0244), sensor suhu (DS18B20), dan sensor ketinggian air (HC-SR04). Kemudian setelah data dapat dikirim, Arduino menunggu selang waktu berikutnya untuk melakukan pengiriman data dari nilai sensor. Pengolahan data kedua untuk pengendalian yaitu Arduino mengambil data nilai minimum dan maksimum dengan mengirimkan request GET parameter temperatur, *total dissolved solids*, dan pH melalui aplikasi web *API* kemudian membandingkan dengan nilai sekarang menggunakan sensor pH (SEN0161-V2), sensor *total dissolved solids* (SEN0244), dan sensor ketinggian air (HC-SR04). Jika nilai yang didapatkan lebih rendah dari batas bawah, maka katup solenoid akan membuka dan larutan yang berkaitan akan tercampurkan ke dalam wadah larutan hidroponik. Arduino akan mengulangi proses tersebut hingga nilai yang dibandingkan memenuhi batas bawah dan batas atas. Pengguna dapat melihat informasi nilai sensor dari aplikasi web *client.* Aplikasi web *client* mengambil data dari *database* melalui aplikasi web *API* dengan *request* GET di setiap *endpoint* yang dibutuhkan.

4.3 Desain Perangkat Keras

Desain perangkat keras dbuat dengan tujuan untuk menjadi acuan dalam membuat perangkat Arduino yang sesuai dengan kebutuhan yang diinginkan. Desain yang dibuat akan mampu melakukan pengambilan keputusan dalam membuka/menutup katup solenoid yang berisi larutan untuk mengendalikan nilai yang diinginkan. Desain ini juga mampu melakukan pemantauan seperti nilai pH, *total dissolved solids*, temperatur, dan ketinggian air. Desain perangkat keras secara umum terbagi menjadi empat bagian, yaitu desain instalasi hidroponik, desain wadah larutan hidroponik, desain wadah larutan kontrol beserta katup solenoid, dan kotak Arduino beserta sensor, relay, dan LCD *display.*

4.3.1 Desain Perangkat Arduino

Terdapat beberapa komponen yang dihubungkan dengan mikrokontroler Arduino sebagai data masukan. Berikut

penjelasan masing-masing komponen yang digunakan pada Tabel 4.4 berikut.

Tabel 4. 4 Daftar Komponen Mikrokontroler

No	Nama Komponen	Fungsi
1	Arduino Mega2560 + Wifi ESP8266	Arduino Mega + Wifi ESP8266 berguna sebagai mikrokontroler yang berguna mengolah data, memperoleh data lingkungan menggunakan masing-masing sensor, melakukan keputusan menggunakan aktuator. Perangkat ini juga dapat mengirim dan menerima data dari aplikasi web.
2	Sensor pH SEN0161-V2	Sensor ini berfungsi untuk mendapatkan nilai pH dari larutan hidroponik dan sebagai masukan untuk menentukan keputusan pengoperasian katup solenoid yang berisi larutan asam. Nilai pH ini juga dikirimkan ke aplikasi web *API* untuk divisualisasikan sebagai grafik pada aplikasi web *client*.
3	Sensor *total dissolved solids* SEN0244	Sensor ini berfungsi untuk mendapatkan nilai *total dissolved solids* dari larutan hidroponik dan sebagai masukan untuk menentukan keputusan pengoperasian katup solenoid yang berisi larutan nutrisi A dan B. Nilai *total dissolved solids* ini juga dikirimkan ke aplikasi web *API* untuk divisualisasikan

		sebagai grafik pada aplikasi web *client*.
4	Sensor Ketinggian Air HC-SR04	Sensor ini berfungsi untuk mendapatkan nilai ketinggian air dari larutan hidroponik dan sebagai masukan untuk menentukan keputusan pengoperasian katup solenoid yang berisi air. Nilai ketinggian air ini juga dikirimkan ke aplikasi web *API* untuk divisualisasikan sebagai grafik pada aplikasi web *client*.
4	Sensor Suhu DS18B20	Sensor ini berfungsi untuk mendapatkan data temperatur dari larutan hidroponik. Data temperatur ini juga dikirimkan ke aplikasi web *client* untuk ditampilkan menjadi grafik.
5	LCD Display	Layar LCD digunakan untuk menampilkan data langsung dari Arduino sebelum dikirimkan ke web.
6	Relay dan Katup Solenoid	Relay dan katup solenoid berguna untuk membuka/menutup wadah larutan kontrol. Relay akan berjalan jika Arduino mendapatkan nilai sensor di batas bawah.

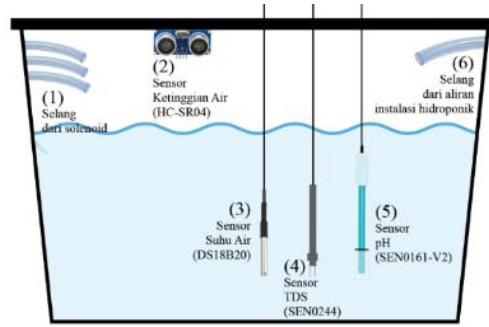
Arduino menggunakan komponen pendukung seperti pada Tabel 4.4. Komponen-komponen tersebut dihubungkan pada Arduino melalui pin-pin yang tersedia, berikut adalah daftar pin pada Tabel 4.5.

Tabel 4. 5 Daftar Pin Sensor

No	Komponen Sensor	Pin Arduino
1	LCD Display	SCL, SDA, Vcc, Gnd
2	DS18B20	2, Vcc, Gnd
3	HC-SR04	6, 7, Vcc, Gnd
4	SEN0244	A0, Vcc, Gnd
5	SEN0161-V2	A1, Vcc, Gnd
6	Relay (Air)	8, Vcc, Gnd
7	Relay (Larutan Basa)	9 Vcc, Gnd
8	Relay (Nutrisi A)	10, Vcc, Gnd
9	Relay (Nutrisi B)	11, 13, Vcc, Gnd

4.3.2 Desain Wadah Larutan Hidroponik

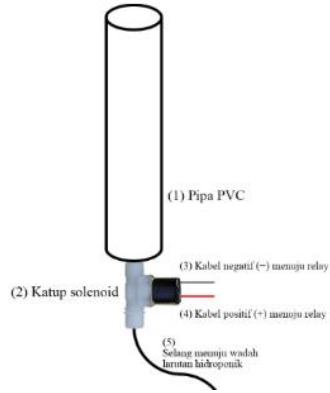
Berdasarkan hasil diskusi dengan Bapak Sulihan, petani dan pengrajin hidroponik, didapatkan desain wadah larutan hidroponik yang ideal agar sensor dapat mendapatkan nilai lingkungan dengan baik. Dalam melakukan pembacaan dari sensor, sensor juga harus diletakkan sedemikan rupa agar rapi dan sensor dapat berjalan dengan baik, berikut adalah desain wadah larutan hidroponik pada Gambar 4.2.



Gambar 4. 2 Desain Wadah Larutan Hidroponik

4.3.3 Desain Wadah Larutan Kontrol

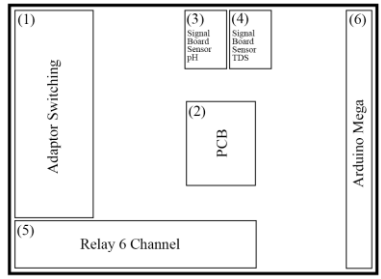
Wadah larutan kontrol hidroponik ini berfungsi sebagai tempat penyimpanan sebelum larutan kontrol dicampurkan pada wadah larutan hidroponik. Terdapat pipa PVC yang dihubungkan dengan katup solenoid diletakkan secara vertikal agar cairan dapat mengalir dengan cepat melalui selang. Berikut adalah desain wadah larutan kontrol pada Gambar 4.3.



Gambar 4. 3 Desain Wadah Larutan Kontrol

4.3.4 Desain Kotak Perangkat Arduino

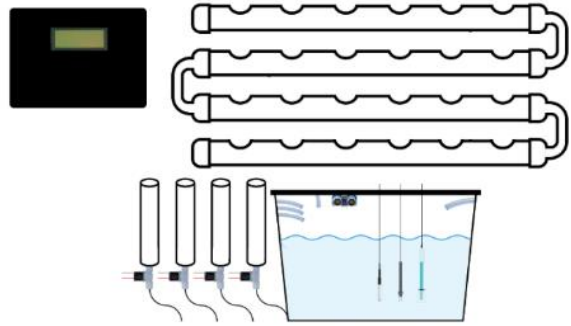
Perangkat keras Arduino akan diletakkan pada kotak agar peletakan komponen-komponen lebih rapi dan terlindung dari korsleting. Kotak terbuat dari plastik yang tebal. Berikut adalah desain dari kotak Arduino pada Gambar 4.4.



Gambar 4. 4 Desain Kotak Perangkat Arduino

4.3.5 Desain Instalasi Hidroponik

Tanaman hidroponik membutuhkan aliran air untuk menyerap nutrisinya. Pipa PVC dibutuhkan untuk menjadikan air dapat mengalir. Berikut adalah desain dari instalasi hidroponik pada Gambar 4.5. Pertumbuhan tanaman hidroponik akan dihitung berdasarkan selisih dari tinggi tanaman sebelum dilakukan proses budidaya dan sesudah dilakukan proses budidaya. Setelah itu hasil tanaman yang sudah dilakukan proses budidaya akan dibandingkan, tanaman yang menggunakan hidroponik dengan sistem *smart farming* otomatis dan tanaman yang menggunakan hidroponik dengan sistem manual.



Gambar 4. 5 Desain Instalasi Hidroponik

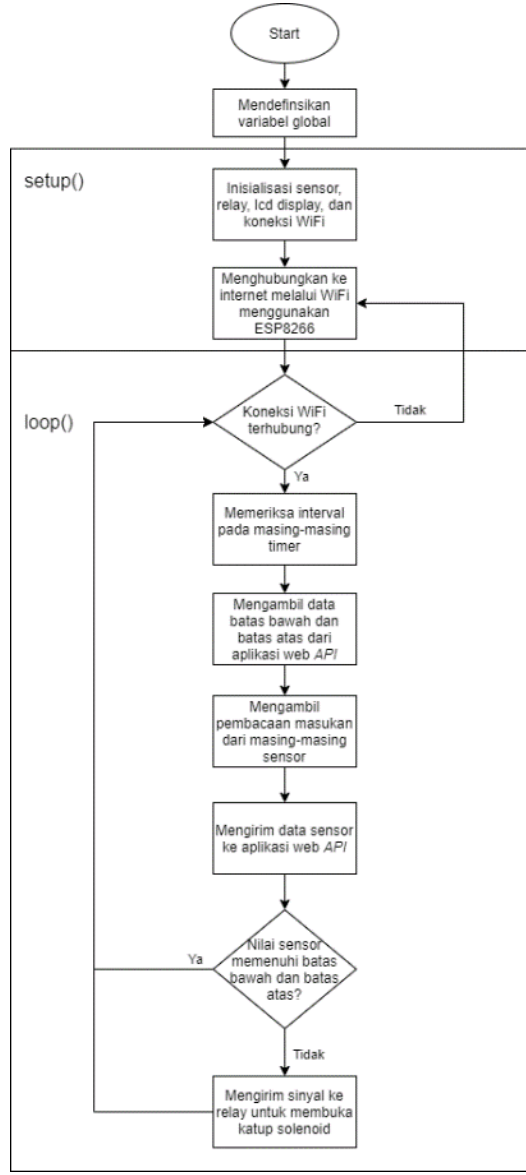
4.3.6 Desain Program Mikrokontroler Arduino

Gambar 4.6 menjelaskan alur dari kode program perangkat keras Arduino secara keseluruhan. Dimulai dengan Arduino menyala, kemudian Arduino terhubung dengan WiFi. Setelah berhasil terhubung, Arduino akan menginisialiasi variabel global seperti SSID, *password*, batas bawah, batas atas, nilai sensor, pin komponen pendukung (relay, sensor, LCD display) agar variabel tersebut dapat diakses di *method* lain.

Method setup() berfungsi sebagai konfigurasi awal Arduino dalam menginisialisasi komponen-komponen pendukung, kecepatan serial baudrate, inisialisasi pin LCD display, sensor suhu DS18B20, sensor pH SEN0161-V2, sensor *total dissolved solids* SEN0244, sensor ketinggian air HC-SR04. Setelah semua

inisialisasi dan konfigurasi pin telah selesai, maka *method* loop() akan berjalan.

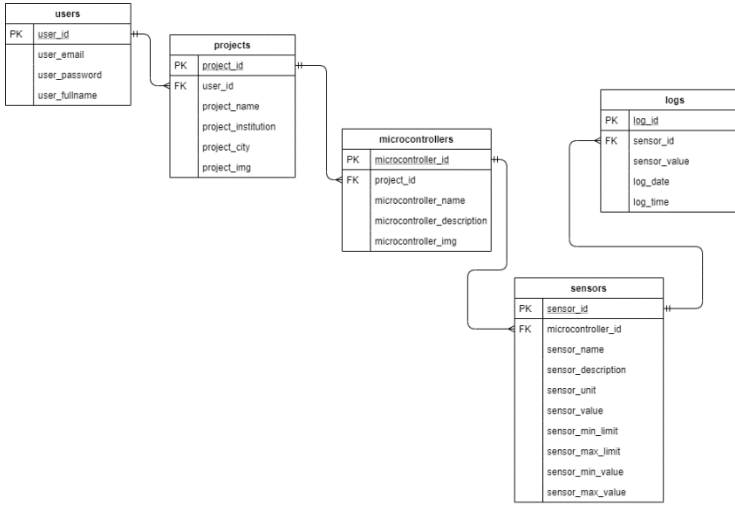
Pada *method* loop() berisi perintah program yang akan berulang terus menerus, diawali dengan pengecekan selang waktu *method* milis() untuk timer masing-masing *method*. Saat *method* loop() berjalan, Arduino akan memanggil *method* getValueFromSensors() untuk mendapatkan seluruh pembacaan dari sensor. Di dalam *method* getValueFromSensors() terdapat *method* lain yaitu: getTemperature() untuk mendapatkan nilai suhu, getTDSValue() untuk mendapatkan nilai *total dissolved solids*, getDistance() untuk mendapatkan nilai ketinggian air, dan getpHValue() untuk mendapatkan nilai pH dari larutan hidroponik. Setelah berhasil mendapatkan berbagai nilai pembacaan, nilai tersebut diset ke variabel global agar dapat dibaca *method* lain untuk melakukan perbandingan nilai batas bawah dan batas atas dan nilai data sensor dapat dikirim ke aplikasi web *API* menggunakan *method* postData(). Arduino melakukan pengiriman data pembacaan sensor menggunakan postData(). M*ethod* tersebut menggunakan metode *request* POST serta berisi sensor_id, id sensor yang telah dikonfigurasi pada aplikasi web *client*, sensor_value, nilai sensor yang telah berhasil diambil oleh Arduino menggunakan sensor, dan JWT untuk keamanan pada masing-masing *request.* Arduino menggunakan *method* getData() untuk mendapatkan nilai batas bawah dan batas atas nilai *total dissolved solids*, ketinggian air, dan pH. *Method* tersebut menggunakan metode *request* GET, dan setiap *request* diberikan JWT untuk keamanan masing-masing *request*. Setelah getData() berhasil dilakukan, Arduino akan membandingkan nilai sensor menggunakan variabel global, jika nilai yang dibaca kurang dari batas bawah yang telah ditentukan maka Arduino memerintah katup solenoid melalui relay untuk membuka katup dengan selang waktu yang ditentukan. Jika nilai yang dibaca tetap kurang dari batas bawah yang telah ditentukan maka Arduino akan menyuruh kembali relay hingga batas bawah terpenuhi dengan selang waktu yang ditentukan. *Flow chart* program Arduino ditunjukkan pada Gambar 4.6 berikut.



Gambar 4. 6 *Flow Chart* Program Arduino

4.4 Desain *Database*

Basis data yang digunakan pada aplikasi web menggunakan basis data mySQL. Gambar 4.7 adalah desain dari basis data yang akan digunakan pada aplikasi web.



Gambar 4. 7 Desain *Database*

Database terdiri dari lima tabel, yaitu *users*, *projects*, *microcontrollers*, *sensors*, dan *logs*. Masing masing dari tabel digunakan untuk menyimpan informasi terkait dari sistem. Untuk rincian kegunaan dari masing-masing tabel akan dijelaskan pada tabel 4.7 sebagai berikut:

Tabel 4. 6 Penjelasan Fungsi Tabel pada Database

No	Nama Tabel	Fungsi
1	*users*	Menyimpan informasi nama, email, password *user* (pengguna) yang dapat mengakses aplikasi web *client*

2	*projects*	Menyimpan informasi daftar proyek pengguna dan menyimpan hubungan relasi tabel antara *users* dan *projects*
3	*microcontrollers*	Menyimpan informasi daftar mikrokontroler pengguna dan menyimpan hubungan relasi tabel antara *projects* dan *microcontrollers*
4	*sensors*	Menyimpan informasi daftar sensor pengguna, menyimpan hubungan relasi tabel antara *microcontrollers* dan *sensors*, menyimpan konfigurasi batas bawah, batas atas, dan satuan pengukuran
5	*logs*	Menyimpan informasi *datalog* pengguna yang kemudian ditampilkan pada *chart* dan menyimpan hubungan relasi tabel antara *sensors* dan *logs*

Tabel 4. 7 Deskripsi Kolom Tabel *users*

Tabel: device	
Nama Kolom	**Deskripsi**
user_id (PK)	*Id Primary Key* pengguna
user_email	Email pengguna
user_password	Password pengguna
user_fullname	Nama lengkap pengguna

Tabel 4. 8 Deskripsi Kolom Tabel *projects*

Tabel: device	
Nama Kolom	**Deskripsi**
project_id (PK)	*Id Primary Key* Proyek pengguna
user_id (FK)	Pemilik proyek menggunakan referensi user_id
project_name	Nama proyek
project_institution	Nama lembaga
project_city	Nama kota
project_img	Nama gambar beserta *format*

Tabel 4. 9 Deskripsi Kolom Tabel *microcontrollers*

Tabel: device	
Nama Kolom	**Deskripsi**
microcontroller_id (PK)	*Id Primary Key* perangkat mikrokontroler pengguna
project_id (FK)	Proyek yang berhubungan dengan mikrokontroler
microcontroller_name	Nama mikrokontroler
microcontroller_description	Nama lembaga
microcontroller_img	Nama gambar beserta *format*

Tabel 4. 10 Deksripsi Kolom Tabel *sensors*

Tabel: device	
Nama Kolom	**Deskripsi**
sensor_id (PK)	*Id Primary Key* perangkat sensor pengguna

microcontroller_id (FK)	Sensor yang berhubungan dengan mikrokontroler
sensor_name	Nama sensor
sensor_description	Deskripsi kegunaan sensor
Sensor_unit	Satuan pengukuran sensor
Sensor_min_limit	Nilai batas bawah suatu pengukuran pada sensor yang akan dikontrol
sensor_max_limit	Nilai batas atas suatu pengukuran pada sensor yang akan dikontrol
sensor_min_value	Nilai terkecil dari pembacaan sensor yang akan ditampilkan pada *chart*
sensor_max_value	Nilai terbesar dari pembacaan sensor yang akan ditampilkan pada *chart*

Tabel 4. 11 Deskripsi Kolom Tabel *logs*

Tabel: device	
Nama Kolom	**Deskripsi**
log_id (PK)	*Id Primary Key datalog* sensor
sensor_id (FK)	*Datalog* yang berhubungan dengan sensor
sensor_value	Nilai pembacaan dari sensor yang terhubung dengan mikrokontroler yang berhubungan
log_date	Tanggal *datalog* yang telah tersimpan
log_time	Waktu *datalog* yang telah tersimpan

4.5 Desain Perangkat Lunak

Desain perangkat lunak digunakan sebagai *guideline* dalam pengimplementasian aplikasi web sesuai dengan kubutuhan yang didapatkan dari analisis kebutuhan sebelumnya.

4.5.1 Use *Case* Aplikasi Web

Perancangan *use case diagram* aplikasi web dibuat untuk menunjukkan fungsi utama pada sistem sesuai dengan masing-masing aktor. Gambar 4.8 di bawah ini menunjukkan *use case diagram* aplikasi web.

Gambar 4. 8 Diagram Use Case Aplikasi We**b**

Dari *use case* model di atas akan dibuat *use case scenario* yang ditunjukkan pada Tabel 4.20 hingga Tabel 4.28 berikut ini.

Tabel 4. 12 Use Case Scenario Mengirim Data Sensor

Use Case* : Mengirim Data Sensor ke Aplikasi Web *API	
Actor	Mikrokontroler Arduino
Description	Aktor mengirim data pH, *total dissolved solids*, temperatur, dan ketinggian air ke aplikasi web *API* yang selanjutnya disimpan ke dalam *database*
Pre-Condition	Aktor telah terhubung ke internet serta JWT, data id masing-masing sensor, data sensor seperti pH, *total dissolved solids*, temperatur, dan ketinggian air sudah ada
Basic Course	1. Aktor mengirimkan *request* HTTP POST ke web yang berisi data dan JWT 2. Sistem melakukan validasi JWT dan kelengkapan data 3. Sistem melakukan penyimpanan data ke dalam database 4. Sistem mengirimkan *response 200 OK*
Alternate Course	2a. Validasi JWT gagal dikarenan token tidak valid, sistem mengirimkan *response 401 Unauthorized* 2b. Data gagal dimasukkan karena *field* data kurang lengkap, sistem mengirimkan *response 500 Internal Server Error*

Tabel 4. 13 Use Case Scenario Menerima Nilai Batas Bawah dan Batas Atas

Use Case* : Menerima Nilai Batas Bawah dan Batas Atas dari Aplikasi Web *API

Actor	Mikrokontroler Arduino
Description	Aktor mengirim *request* GET ke aplikasi web *API* untuk menerima nilai batas bawah dan batas atas dari database
Pre-Condition	Aktor sudah terhubung ke internet dan JWT sudah ada
Basic Course	1. Aktor mengirimkan *request* HTTP GET ke aplikasi web *API* 2. Aplikasi web *API* mengirimkan nilai batas bawah dan batas atas ke Arduino 3. Aplikasi web *API* mengirimkan *response 200 OK*
Alternate Course	2a. Validasi JWT gagal dikarenan token tidak valid, sistem mengirimkan *response 401 Unauthorized*

Tabel 4. 14 Use Case Scenario Mengelola Daftar Proyek

***Use Case* : Mengelola Daftar Proyek**	
Actor	*User*
Description	*User* dapat mengelola daftar proyek pada aplikasi web *client*, seperti tambah, ubah, dan hapus proyek
Pre-Condition	Aktor sudah login dan aplikasi web *client* sudah menyimpan JWT pada *Session*
Basic Course	1. Aktor mengakses halaman Daftar Proyek 2. Aktor melakukan klik tombol tambah, ubah, ataupun hapus proyek 3. Aplikasi web *client* mengirimkan *request* berdasarkan aksi tambah, ubah, atau hapus proyek

	4. Aplikasi web *API* memvalidasi JWT 5. Aplikasi web *API* mengirimkan *response 200 OK*
Alternate Course	4.a Validasi JWT gagal dan aplikasi web *API* akan mengirimkan *response 401 Unauthorized*

Tabel 4. 15 Use Case Scenario Mengelola Daftar Mikrokontroler

***Use Case* : Mengelola Daftar Mikrokontroler**	
Actor	*User*
Description	Aktor dapat mengelola daftar mikrokontroler pada aplikasi web *client*, seperti tambah, ubah, dan hapus mikrokontroler
Pre-Condition	Aktor sudah login dan aplikasi web *client* sudah menyimpan JWT pada *Session*
Basic Course	1. Aktor mengakses halaman Daftar Mikrokontroler 2. Aktor melakukan klik tombol tambah, ubah, ataupun hapus mikrokontroler 3. Aplikasi web *client* mengirimkan *request* berdasarkan aksi tambah, ubah, atau hapus mikrokontroler 4. Aplikasi web *API* memvalidasi JWT 5. Aplikasi web *API* mengirimkan *response 200 OK*
Alternate Course	4.a Validasi JWT gagal dan aplikasi web *API* akan mengirimkan *response 401 Unauthorized*

Tabel 4. 16 Use Case Scenario Mengelola Daftar Sensor

***Use Case* : Mengelola Daftar Sensor**	
Actor	*User*
Description	Aktor dapat mengelola daftar sensor pada aplikasi web *client*, seperti tambah, ubah, dan hapus sensor
Pre-Condition	Aktor sudah login dan aplikasi web *client* sudah menyimpan JWT pada *Session*
Basic Course	1. Aktor mengakses halaman Daftar Sensor 2. Aktor melakukan klik tombol tambah, ubah, ataupun hapus sensor 3. Aplikasi web *client* mengirimkan *request* berdasarkan aksi tambah, ubah, atau hapus sensor 4. Aplikasi web *API* memvalidasi JWT 5. Aplikasi web *API* mengirimkan *response 200 OK*
Alternate Course	4.a Validasi JWT gagal dan aplikasi web *API* akan mengirimkan *response 401 Unauthorized*

Tabel 4. 17 Use Case Scenario Menampilkan Dashboard Proyek

***Use Case* : Menampilkan Dashboard Proyek**	
Actor	*User*
Description	Aktor dapat melihat halaman dashboard Proyek dan melihat daftar proyek

Pre-Condition	Aktor sudah login dan aplikasi web *client* sudah menyimpan JWT pada *Session*
Basic Course	1. Aktor mengakses halaman dashboard proyek 2. Aplikasi web *client* mengirimkan *request* GET beserta JWT pada *header* 3. Aplikasi web *API* memvalidasi JWT 4. Aplikasi web *API* mengirimkan data daftar proyek yang dimiliki oleh aktor dan mengirimkan *response 200 OK* 5. Aplikasi web *client* melakukan *parsing* data 6. Aplikasi web *client* menampilkan tampilan daftar proyek dan aktor dapat melihat halaman dashboard proyek
Alternate Course	3a. Validasi JWT gagal dan aplikasi web *API* akan mengirimkan *response 401 Unauthorized*

Tabel 4. 18 Use Case Scenario Menampilkan Dashboard Mikrokontroler

***Use Case* : Menampilkan Dashboard Mikrokontroler**	
Actor	*User*
Description	Aktor dapat melihat halaman dashboard utama dan halaman detil pada masing-masing kelas
Pre-Condition	Aktor sudah login dan aplikasi web *client* sudah menyimpan JWT pada *Session*

Basic Course	1. Aktor mengakses halaman dashboard mikrokontroler 2. Aplikasi web *client* mengirimkan *request* GET beserta JWT pada *header* 3. Aplikasi web *API* memvalidasi JWT 4. Aplikasi web *API* mengirimkan data daftar mikrokontroler yang dimiliki oleh aktor dan mengirimkan *response 200 OK* 5. Aplikasi web *client* melakukan *parsing* data 6. Aplikasi web *client* menampilkan tampilan daftar mikrokontroler dan aktor dapat melihat halaman dashboard mikrokontroler
Alternate Course	-

Tabel 4. 19 Use Case Scenario Menampilkan Dashboard Sensor

***Use Case* : Menampilkan Dashboard Sensor**	
Actor	*User*
Description	User dapat mendaftar akun dengan mengakses halaman registrasi
Pre-Condition	Aktor sudah login dan aplikasi web *client* sudah menyimpan JWT pada *Session*
Basic Course	1. Aktor mengakses halaman dashboard mikrokontroler 2. Aplikasi web *client* mengirimkan *request* GET beserta JWT pada *header* 3. Aplikasi web *API* memvalidasi JWT 4. Aplikasi web *API* mengirimkan data daftar sensor dan log yang dimiliki

	oleh aktor dan mengirimkan *response 200 OK* 5. Aplikasi web *client* melakukan *parsing* data 6. Aplikasi web *client* menampilkan tampilan daftar sensor, dan log kemudian aktor dapat melihat halaman dashboard sensor
Alternate Course	-

Tabel 4. 20 Use Case Scenario Membuat Akun Baru

***Use Case* : Membuat Akun Baru**	
Actor	*User*
Description	Aktor dapat membuat akun dengan mengakses halaman *sign up*
Pre-Condition	-
Basic Course	1. Aktor mengakses halaman *sign up* 2. Aktor mengisi form pendaftaran 3. Aktor melakukan klik tombol "Sign Up" 4. Aktor sudah dapat menggunakan akun dengan melakukan login pada halaman login
Alternate Course	3a. Muncul notifikasi error karena kesalahan input

4.5.2 Desain Aplikasi Web *Client*

Aplikasi web *client* dibangun dengan bahasa pemrograman PHP menggunakan *framework Codeigniter*. Penggunaan *framework* ini bertujuan untuk memudahakan pembuatan web dan juga dapat mengoptimalkan fitur yang disediakan oleh *framework.*

Framework ini memiliki *design pattern* MVC, yaitu *Model*, *View*, dan *Controller*. *Model* berguna untuk menjalankan *query* langsung pada *database*. *View* berguna menampilkan halaman web kepada pengguna. Sedangkan *Controller* berfungsi melakukan fungsi utama aplikasi web dalam membuat, menerima, memproses, dan mengoordinasi data pada *Model*, *View*, dan *Controller* lainnya. Aplikasi web *client* ini hanya bertindak sebagai *client* (*front-end*) website. Aplikasi web *client* ini menggunakan *library Guzzle* untuk mengirim *request* HTTP ke aplikasi web *API* (*back-end)* website.

4.5.2.1 Detil Desain MVC Aplikasi Web *Client*

Berikut adalah detil dari masing-masing *model*, *view*, dan *controller* pada aplikasi web *client* yang dijelaskan dalam Tabel 4. 21 hingga Tabel 4. 23 di bawah ini.

Tabel 4. 21 Detil Model Aplikasi Web Client

Nama Model	Deskripsi
Project_model	Model yang digunakan untuk penanganan melihat, menambahkan, mengubah, dan menghapus data proyek menggunakan *request* ke aplikasi web *API*
Microcontroller_model	Model yang digunakan untuk penanganan melihat, menambahkan, mengubah, dan menghapus data mikrokontroler menggunakan *request* ke aplikasi web *API*
Sensor_model	Model yang digunakan untuk penanganan melihat, menambahkan, mengubah, dan menghapus data sensor dan log

	menggunakan *request* ke aplikasi web *API*

Tabel 4. 22 Detil View Aplikasi Web Client

Nama *View*	Deskripsi
project/user_id/	*View* yang digunakan untuk menampilkan halaman kelola proyek
microcontroller/project_id	*View* yang digunakan untuk menampilkan halaman kelola mikrokontroler
sensor/microcontroller_id	*View* yang digunakan untuk menampilkan halaman kelola sensor dan menampilkan visualisasi data log sensor menggunakan grafik
login	*View* yang digunakan untuk menampilkan halaman *login*
signup	*View* yang digunakan untuk menampilkan halaman membuat akun baru

Tabel 4. 23 Detil Controller Aplikasi Web Client

Nama Controller	Deskripsi
Auth	*Controller* yang berguna menerima dan menyimpan JWT pada *Session* aplikasi web *client.*
Project	*Controller* yang berguna untuk menampilkan dan mengelola data proyek yang dimiliki oleh *user*.

Microcontroller	*Controller* yang berguna untuk menampilkan dan mengelola data mikrokontroler yang dimiliki oleh *user*.
Sensor	*Controller* yang berguna untuk menampilkan dan mengelola data sensor yang dimiliki oleh *user*.

Dalam menjalankan masing-masing perannya pada sistem, baik *controller* maupun *model* memiliki *function*. Berikut adalah penjelasan *function* yang ditambahkan pada tiap-tiap *controller* yang ditunjukkan pada Tabel 4.X hingga Tabel 4.X.

Tabel 4. 24 Function pada Controller Auth

Controller : **Auth**	
Nama *Function*	**Deskripsi**
index()	Berfungsi untuk menerima JWT dan user_id dari halaman login dan *redirect* ke halaman dashboard proyek berdasarkan user_id
logout()	*Function* yang berfungsi untuk menghancurkan seluruh *Session* yang aktif dan logout *user* menuju halaman login

Tabel 4. 25 Function pada Controller Project

Controller : **Project**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-

	masing *function* di dalam *controller* tersebut
user_id()	*Function* yang berfungsi untuk menampilkan dashboard proyek berdasarkan user_id
add_project()	*Function* yang berfungsi mengirimkan *request* POST ke aplikasi web *API* untuk membuat data proyek baru
edit_poject()	*Function* yang berfungsi mengirimkan *request* PUT ke aplikasi web *API* untuk mengubah data proyek
delete_project()	*Function* yang berfungsi mengirimkan *request* DELETE ke aplikasi web *API* untuk menghapus data proyek

Tabel 4. 26 Function pada Controller Microcontroller

***Controller :* Microcontroller**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
project_id()	*Function* yang berfungsi untuk menampilkan dashboard mikrokontroler berdasarkan project_id

add_microcontroller()	*Function* yang berfungsi mengirimkan *request* POST ke aplikasi web *API* untuk membuat data mikrokontroler baru
edit_microcontroller()	*Function* yang berfungsi mengirimkan *request* PUT ke aplikasi web *API* untuk mengubah data mikrokontroler
delete_microcontroller()	*Function* yang berfungsi mengirimkan *request* DELETE ke aplikasi web *API* untuk menghapus data mikrokontroler

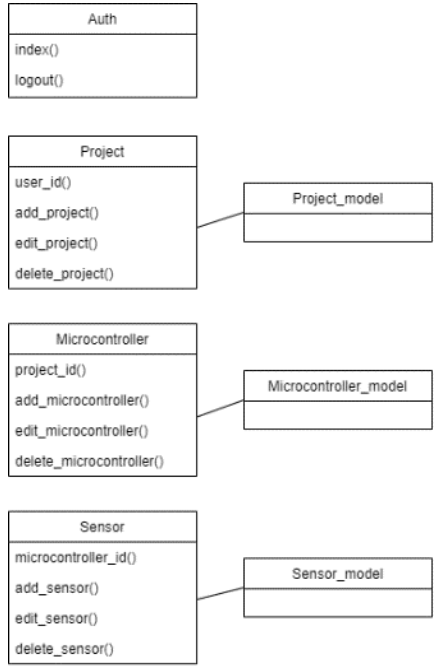
Tabel 4. 27 Function pada Controller Sensor

***Controller :* Sensor**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
microcontroller_id()	*Function* yang berfungsi untuk menampilkan dashboard sensor berdasarkan microcontroller_id
add_sensor()	*Function* yang berfungsi mengirimkan *request* POST ke aplikasi web *API* untuk membuat data sensor baru

edit_sensor()	*Function* yang berfungsi mengirimkan *request* PUT ke aplikasi web *API* untuk mengubah data sensor
delete_sensor()	*Function* yang berfungsi mengirimkan *request* DELETE ke aplikasi web *API* untuk menghapus data sensor

4.5.2.2 Class Diagram Aplikasi Web Client

Desain MVC untuk aplikasi web *client* sudah ditentukan. Agar lebih mudah dipahami, desain rancangan MVC tersebut diterjemahkan ke dalam bentuk gambar *class diagram* untuk memudahkan proses pengembangan aplikasi. Berikut adalah *class diagram* aplikasi web *client* yang ditunjukkan pada Gambar 4.9 di bawah ini.



Gambar 4. 9 Class Diagram Aplikasi Web Client

4.5.3 Desain Aplikasi Web *API*

Aplikasi web *API* juga dibangun menggunakan Bahasa pemrograman PHP dan menggunakan *framework Codeigniter* yang menggunakan *design pattern Model*, *View*, dan *Controller*. Namun aplikasi web *API* ini menggunakan tambahan semacam *library* yaitu REST API JWT. Fungsi Codeigniter sebagai *API* (*back-end*) semakin bagus dikarenakan *library* ini dioptimalkan khusus menerima penanganan *request* REST API seperti GET, POST, PUT, PATCH, OPTIONS, dan DELETE. *Library* juga menambahkan autentikasi verifikasi JWT di setiap masing-masing *request* untuk meningkatkan keamanan dan mencegah *API* diakses orang lain yang tidak berkepentingan.

4.5.3.1 Detil Desain MVC Aplikasi Web *API*

Berikut adalah detil dari masing-masing *model* dan *controller* pada aplikasi web *API* yang dijelaskan dalam Tabel 4. 28 hingga 4. 29 di bawah ini.

Tabel 4. 28 Detil Model Aplikasi Web *API*

Nama Model	Deskripsi
Auth_model	Berfungsi dalam memvalidasi login
User_model	Berfungsi dalam melakukan pembuatan akun *user* baru
Project_model	Model yang digunakan untuk menjalankan *query* melihat, menambahkan, mengubah, dan menghapus data proyek secara langsung ke *database*
Microcontroller_model	Model yang digunakan untuk menjalankan *query* melihat, menambahkan, mengubah, dan menghapus data mikrokontroler secara langsung ke *database*
Sensor_model	Model yang digunakan untuk menjalankan *query* melihat, menambahkan, mengubah, dan menghapus data sensor secara langsung ke *database*
Log_model	Model yang digunakan untuk menjalankan *query* melihat dan menambahkan data log sensor secara langsung ke *database*

Pada aplikasi web *API* tidak terdapat adanya *View* dikarenakan hanya menerima *request* HTTP berupa *method* seperti GET,

POST, PUT, dan DELETE. Kemudian *response* yang diberikan oleh aplikasi web *API* hanya dalam bentuk JSON yang kemudian dilakukan proses *parse* oleh aplikasi web *client* ataupun Arduino.

Tabel 4. 29 Detil Controller Aplikasi Web API

Nama Controller	Deskripsi
Auth	*Controller* yang berguna menerima *request* login
User	*Controller* yang berfungsi dalam membuat akun baru
Project	*Controller* yang berguna untuk melakukan penanganan *method request* GET, POST, PUT, dan DELETE dalam mengelola data proyek.
Microcontroller	*Controller* yang berguna untuk melakukan penanganan *method request* GET, POST, PUT, dan DELETE dalam mengelola data mikrokontroler.
Sensor	*Controller* yang berguna untuk melakukan penanganan *method request* GET, POST, PUT, dan DELETE dalam mengelola data sensor.
Log	*Controller* yang berguna untuk melakukan penanganan *method request* GET dan POST dalam mengelola data log sensor.

Dalam menjalankan masing-masing perannya pada sistem, baik *controller* maupun *model* memiliki *function*. Berikut adalah penjelasan *function* yang ditambahkan pada tiap-tiap *controller* yang ditunjukkan pada Tabel 4.30 hingga Tabel 4.35.

Tabel 4. 30 Function pada Controller Auth

***Controller :* Auth**	
Nama *Function*	**Deskripsi**
login_post()	Berfungsi untuk menerima *request* POST untuk melakukan login

Tabel 4. 31 Function pada Controller User

***Controller :* Auth**	
Nama *Function*	**Deskripsi**
index_post()	Berfungsi untuk menerima *request* POST untuk membuat akun baru

Tabel 4. 32 Function pada Controller Project

***Controller :* Project**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
verify_request()	*Function* yang berfungsi untuk memvalidasi JWT pada *header request*
index_get()	*Function* yang berfungsi menerima *request* GET dari aplikasi web *client* kemudian mengirimkan *response* data proyek

user_id_get()	*Function* yang berfungsi menerima *request* GET dari aplikasi web *client* kemudian mengirimkan *response* data proyek berdasarkan user_id yang dimiliki oleh *user*
index_post()	*Function* yang berfungsi menerima *request* POST beserta JWT dari aplikasi web *client* untuk membuat data Proyek baru
index_put()	*Function* yang berfungsi menerima *request* PUT beserta JWT dari aplikasi web *client* untuk mengubah data Proyek berdasarkan id proyek
index_delete()	*Function* yang berfungsi menerima *request* DELETE beserta JWT dari aplikasi web *client* untuk menghapus data Proyek berdasarkan id proyek

Tabel 4. 33 Function pada Controller Microcontroller

Controller : **Microcontroller**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut

verify_request()	*Function* yang berfungsi untuk memvalidasi JWT pada *header request*
index_get()	*Function* yang berfungsi menerima *request* GET dari aplikasi web *client* kemudian mengirimkan *response* data mikrokontroler
project_id_get()	*Function* yang berfungsi menerima *request* GET dari aplikasi web *client* kemudian mengirimkan *response* data mikrokontroler berdasarkan project_id yang dimiliki oleh *user*
index_post()	*Function* yang berfungsi menerima *request* POST beserta JWT dari aplikasi web *client* untuk membuat data Mikrokontroler baru
index_put()	*Function* yang berfungsi menerima *request* PUT beserta JWT dari aplikasi web *client* untuk mengubah data Mikrokontroler berdasarkan id mikrokontroler
index_delete()	*Function* yang berfungsi menerima *request* DELETE beserta JWT dari aplikasi web *client* untuk menghapus data Mikrokontroler berdasarkan id mikrokontroler

Tabel 4. 34 Function pada Controller Sensor

Controller : **Sensor**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
verify_request()	*Function* yang berfungsi untuk memvalidasi JWT pada *header request*
index_get()	*Function* yang berfungsi menerima *request* GET dari aplikasi web *client* kemudian mengirimkan *response* data sensor
microcontroller_id_get()	*Function* yang berfungsi menerima *request* GET dari aplikasi web *client* kemudian mengirimkan *response* data sensor berdasarkan microcontroller_id yang dimiliki oleh *user*
index_post()	*Function* yang berfungsi menerima *request* POST beserta JWT dari aplikasi web *client* untuk membuat data Sensor baru
index_put()	*Function* yang berfungsi menerima *request* PUT beserta JWT dari aplikasi web *client* untuk mengubah

	data Sensor berdasarkan id sensor
index_delete()	*Function* yang berfungsi menerima *request* DELETE beserta JWT dari aplikasi web *client* untuk menghapus data Sensor berdasarkan id sensor

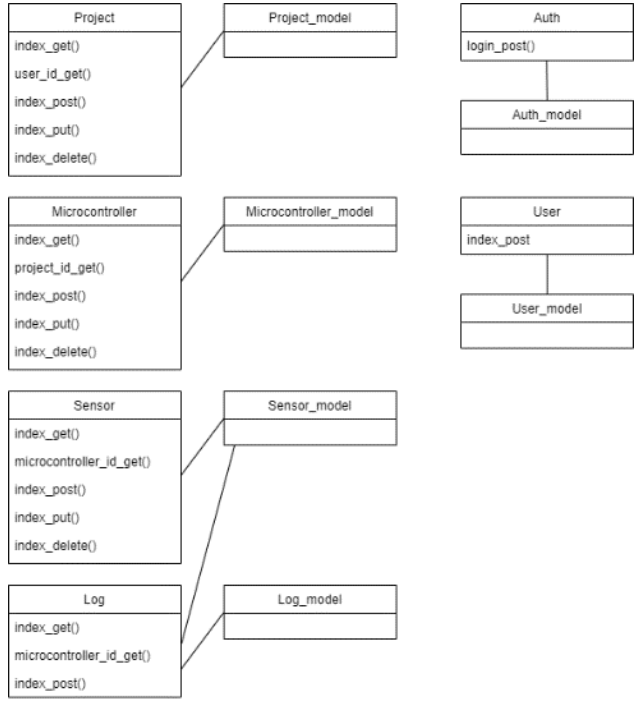
Tabel 4. 35 Function pada Controller Log

***Controller :* Log**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
verify_request()	*Function* yang berfungsi untuk memvalidasi JWT pada *header request*
index_get()	*Function* yang berfungsi menerima *request* GET dari aplikasi web *client* kemudian mengirimkan *response* data log
microcontroller_id_get()	*Function* yang berfungsi menerima *request* GET dari aplikasi web *client* kemudian mengirimkan *response* seluruh data log sensor berdasarkan microcontroller_id yang dimiliki oleh *user*

index_post()	*Function* yang berfungsi menerima *request* POST beserta JWT dari aplikasi web *client* untuk membuat data Log baru

4.5.3.2 Class Diagram Aplikasi Web Client

Desain MVC untuk aplikasi web *client* sudah ditentukan. Agar lebih mudah dipahami, desain rancangan MVC tersebut diterjemahkan ke dalam bentuk gambar *class diagram* untuk memudahkan proses pengembangan aplikasi. Berikut adalah *class diagram* aplikasi web *client* yang ditunjukkan pada Gambar 4.10 di bawah ini.



Gambar 4. 10 Class Diagram Aplikasi Web Client

4.6 Desain *Testing* Sistem

Testing sistem merupakan salah satu tahapan yang penting dalam pengembangan rancang bangun suatu sistem. Hal ini dilakukan untuk mengetahui performa sistem yang dikembangkan dan keluaran masing-masing fungsi sistem. Metode pengujian *system integration testing* dialkukan untuk mengetahui integrasi dari sistem mulai dari perangkat keras,

aplikasi web *client*, dan aplikasi web *API*. Pengujian menggunakan *system integration testing* hampir sama dengan *black box testing*, perbedaannya terletak pada *test case* yang digunakan. Masing-masing *test case* dilengkapi dengan skenario normal untuk menjalankan suatu *test case* dan hasil yang diharapkan (ekspektasi) sebagai syarat lulus uji. Pengujian sistem terbagi menjadi 2, yakni pengujian pada perangkat keras dan perangkat lunak.

4.6.1 Desain *Testing* Perangkat Keras

Testing pada perangkat keras akan dilakukan pada sistem mikrokontroler Arduino beserta komponen lainnya yang berhubungan. Berikut adalah daftar *test case* yang dijelaskan pada Tabel 4.36.

Tabel 4. 36 Test Case Hardware

Test case	Skenario	*Ekspektasi*
Arduino terhubung dengan WiFi	Menghubungkan mikrokontroller Arduino ke WiFi, ssid dan password benar	Arduino dapat terhubung ke jaringan WiFi dan mendapatkan *IP Address*
Pembacaan sensor pH SEN0161-V2	Sensor SEN0161-V2 terhubung dengan arduino dan dicelupkan ke larutan hidroponik	Data pH dapat terbaca oleh perangkat Arduino dari sensor SEN0161-V2 dan ditampilkan pada LCD Display
Pembacaan sensor *total dissolved solids* SEN0244	Sensor SEN0244 terhubung dengan Arduino dan	Data *total dissolved solids* dapat terbaca oleh perangkat

	dicelupkan ke larutan hidroponik	Arduino dari sensor SEN0244 dan ditampilkan pada LCD Display
Pembacaan sensor temperatur DS18B20	Sensor SEN0244 terhubung dengan Arduino dan dicelupkan ke larutan hidroponik	Data temperatur dapat terbaca oleh perangkat Arduino dari sensor DS18B20 dan ditampilkan pada LCD Display
Pembacaan sensor *infrared*	Memancarkan gelombang sinar inframerah melalui *remote control* untuk diterima oleh sensor pembacaan datanya	Arduino dapat menerima data gelombang yang dipancarkan oleh remote dengan berubahnya tampilan layar LCD
Pengiriman data sensor ke aplikasi web *API* melalui HTTP *request* POST	Mengirimkan data hasil pembacaan sensor SEN0161-V2, SEN0244, DS18B20, dan HC-SR04 ke aplikasi web *API* menggunakan HTTP POST	Data berhasil terkirim, HTTP *response code* 201 CREATED
Penerimaan data nilai batas bawah dan batas atas dari aplikasi web *API*	Menerima data nilai batas bawah dan batas atas dari aplikasi web *API*	HTTP *response code* 200 dan mendapatkan data respons dari aplikasi web *API*

melalui HTTP *request* GET	menggunakan HTTP GET	
Respon *relay* untuk membuka/menutup katup solenoiddalam menerima *trigger* dari sensor	Relay terhubung dengan Arduino dan masing-masing sensor	Katup solenoid dapat membuka dan menutup sesuai perintah dari Arduino
Tampilkan data ke LCD	Arduino mengirimkan data pH, *total dissolved solids*, temperatur, dan ketinggian air untuk ditampilkan ke LCD	LCD dapat menampilkan data kelistrikan, suhu, dan mode

4.7 Desain *Testing* Perangkat Lunak

Sama seperti desain *testing* perangkat keras, aplikasi web diuji coba menggunakan metode *system integration testing*. Tabel 4.37 adalah daftar *test case* yang akan diuji pada aplikasi web.

Tabel 4. 37 *Test Case* Perangkat Lunak

Test Case	**Skenario**	**Ekspektasi**
Login	Pengguna mengetik *username* dan *password* dengan benar	Pengguna masuk ke halaman dashboard proyek
	Pengguna mengetik *username* dan atau *password* yang salah	Aplikasi web *client redirect* ke halaman login kembali
Membuat akun	Pengguna memasukkan data yang sesuai pada form dengan benar	Terdapat pemberitahuan akun berhasil dibuat dan

		akan *redirect* ke halaman login
	Pengguna tidak memasukkan data yang sesuai	Aplikasi web *client* akan *redirect* ke halaman *signup* kembali
Melihat dashboard Proyek	Membuka halaman dashboard proyek dan telah login	Aplikasi web *client* menampilkan daftar proyek
	Belum melakukan login	*Redirect* ke halaman login
Mengelola Proyek	Mengklik tombol buat, ubah, atau hapus proyek pada halaman dashboard proyek	*User* dapat membuat, mengubah, atau menghapus proyek
Melihat dashboard Mikrokontroler	Membuka halaman dashboard mikrokontroler dan telah login	Aplikasi web *client* menampilkan daftar mikrokontroler
	Belum melakukan login	*Redirect* ke halaman login
Mengelola Mikrokontroler	Mengklik tombol buat, ubah, atau hapus mikrokontroler pada halaman dashboard mikrokontroler	*User* dapat membuat, mengubah, atau menghapus mikrokontroler
Melihat dashboard Sensor	Membuka halaman dashboard sensor dan telah login	Aplikasi web *client* menampilkan daftar sensor dan menampilkan data log dalam bentuk grafik
	Belum melakukan login	*Redirect* ke halaman login

Mengelola Sensor	Mengklik tombol buat, ubah, atau hapus sensor pada halaman dashboard sensor	*User* dapat membuat, mengubah, atau menghapus sensor

BAB V
IMPLEMENTASI

Pada bagian ini akan menjelaskan mengenai implementasi sistem berdasarkan desain yang telah dibuat pada tahap sebelumnya.

5.1 Lingkungan Implementasi Sistem

Pengembangan program yang digunakan oleh sistem baik program pada perangkat keras atau perangkat lunak menggunakan komputer dengan spesifikasi yang tertera pada tabel 5.1.

Tabel 5. 1 Spesifikasi Komputer Implementasi

Tipe	ASUS ROG GL552VW
Processor	Intel® Core™ i7-6700HQ CPU @2.6GHz
Memory (RAM)	16 GB
Sistem Operasi	Windows 10 Education 64Bit
Graphic Card	Nvidia GeForce GTX 960M

Tabel 5. 2 Spesifikasi Perangkat Keras Implementasi

Tipe	Arduino Mega 2560 + ESP8266 WiFi
Processor	ATmega2560 16 MHz+ ESP8266
Flash Memory	256kb
Memory RAM	8kb
Memory ROM	4kb
Bahasa Pemrograman	C/C++/Arduino
Text Editor (*IDE*)	Arduino IDE

Library	1. WifiEsp v2.2.2 2. ArduinoJson v6.15.2 3. SoftwareSerial 4. LiquidCrystal_I2C 5. OneWire v2.3.5 6. Dallas Temperature v3.8.0 7. DFRobot_PH 8. GravityTDS

Tabel 5. 3 Spesifikasi Aplikasi Web *Client*

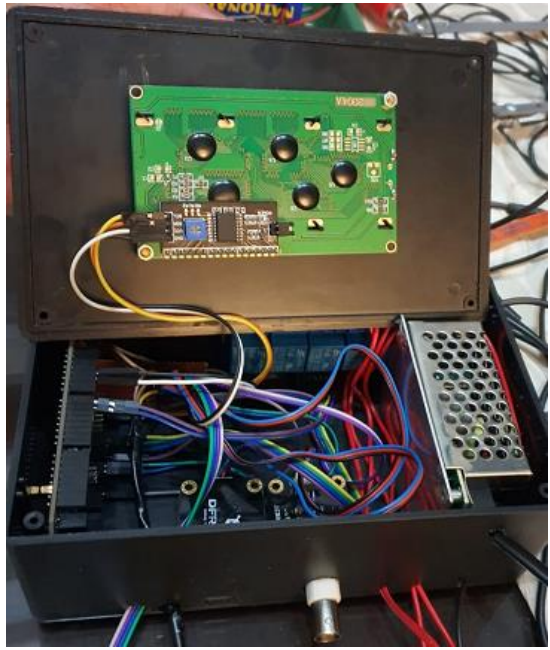
Web Server	Apache
Bahasa Pemrograman	PHP 7.2.10
Framework	Codeigniter 3.10, Bootstrap Material Design 4.1.1
Library	Guzzle, Chart.js
Text Editor (*IDE*)	Visual Studio Code

Tabel 5. 4 Spesifikasi Aplikasi Web *API*

Web Server	Apache
Bahasa Pemrograman	PHP 7.2.10
Framework	Codeigniter REST API JWT 3.10
Library	-
Text Editor (*IDE*)	Visual Studio Code
Database	MySQL

5.2 Implementasi Perangkat Keras

Implementasi perangkat keras dilakukan sesuai dengan rancangan yang telah dibuat pada tahap desain. Tahap awal adalah dengan merangkai seluruh komponen pendukung yaitu, sensor SEN0161-V2, sensor SEN0244, sensor HC-SR04, sensor DS18B20, *LCD Display*, dan relay kemudian diletakkan pada kotak perangkat Arduino.



Gambar 5. 1 Kotak Perangkat Arduino

Setelah semua rangkaian pada Arduino telah terpasang, selanjutnya adalah meletakkan masing-masing ujung sensor ke wadah larutan hidroponik agar nantinya sensor dapat mendapatkan nilai kualitas larutan hidroponik.

Gambar 5. 2 Wadah Larutan Hidroponik

Setelah seluruh komponen pendukung telah dipasang, tahap selanjutnya adalah memasang wadah larutan control hidroponik. Wadah menggunakan pipa PVC ukuran 2,5 inchi. Nepel dengan drat luar ukuran 0,5 inchi dipasang ke pipa PVC untuk menyambung pipa dengan katup solenoida, setelah dipasang kemudian diikat menggunakan *seal tape* khusus untuk pipa agar sambungan pipa tidak terjadi kebocoran. Katup solenoida dialiri listrik 12volt menggunakan adaptor tambahan dan dihubungkan dengan relay.

Gambar 5. 3 Wadah Larutan Kontrol

Instalasi hidroponik juga dibutuhkan untuk menanam tanaman. Desain hidroponik yang digunakan adalah DFT yaitu *deep film technique*. Setelah masing-masing peralatan telah terpasang, selanjutnya memastikan kembali bahwa sensor, aktuator, dan Arduino telah terpasang secara sempurna.

Gambar 5. 4 Instalasi Hidroponik

5.2.1 Implementasi Program Perangkat Keras Arduino

Pemrograman perangkat keras Arduino dibutuhkan untuk melakukan pemantauan dan pengendalian terhadap sistem hidroponik. Pengambilan keputusan akan dilakukan terhadap data batas bawah dan batas atas. Pembuatan kode program dan *upload* dilakukan menggunakan Arduino IDE. Berikut adalah implementasi kode program pada perangkat keras Arduino.

```
1.  #include <ArduinoJson.h>
2.  #include "WiFiEsp.h"
3.  #ifndef HAVE_HWSERIAL3
4.  #endif
5.  #include <SoftwareSerial.h>
6.  #include <OneWire.h>
7.  #include <DallasTemperature.h>
8.  #define ONE_WIRE_BUS 2
9.  OneWire oneWire(ONE_WIRE_BUS);
10. DallasTemperature tempSensors(&oneWire);
11. #include "DFRobot_PH.h"
12. DFRobot_PH ph;
13. #include <EEPROM.h>
14. #include "GravityTDS.h"
15. #define TdsSensorPin A0
16. GravityTDS gravityTds;
17. #include <LiquidCrystal_I2C.h>
18. LiquidCrystal_I2C lcd(0x27, 20, 4);
19. #define pingPin 7
20. #define echoPin 6
21. #define PH_PIN A1
22. #define relay1 8
23. #define relay2 9
24. #define relay3 10
25. #define relay4 11
```

Kode 5. 1 Include Library dan Define Sensor

Kode 5.1 merupakan awal dari konfigurasi *library* dan pin masing-masing komponen pendukung.

```
26. float pHValue, tdsValue, temperature, distance;
27. float pHMaxLimit, pHMinLimit;
28. float tdsMinLimit, tdsMaxLimit;
```

```
29. float distanceMinLimit, distanceMaxLimit;
30. int lines_received = 0;
31.
32. char ssid[] = "namassid";
33. char pass[] = "password";
34. char server[] = "192.168.100.91";
35.
36. long waktuMintaData = 60 * 5 * 1000L, waktuTampil
    kanLCD = 60 * 5 * 1000L;
37. long waktuKirimData = 60 * 60 * 1000L, waktuKontr
    olAktuator = 60 * 10 * 1000L;
38.
39. long waktuMulai;
40. bool responDariServer = false;
41. bool prosesKirimDataKeServer = false;
42. WiFiEspClient client;
43. int status = WL_IDLE_STATUS;
```

Kode 5. 2 Deklarasi Variabel Global

Kode 5.2 merupakan potongan kode melakukan inisialisasi variable global batas bawah, batas atas, koneksi wifi, dan waktu timer seluruh *method*.

```
44.   void setup() {
45.   pinMode(relay1, OUTPUT);
46.   pinMode(relay2, OUTPUT);
47.   pinMode(relay3, OUTPUT);
48.   pinMode(relay4, OUTPUT);
49.   ph.begin();
50.   gravityTds.setPin(TdsSensorPin);
51.   gravityTds.setAref(5.0);
52.   gravityTds.setAdcRange(1024);
53.   gravityTds.begin();
54.   tempSensors.begin();
55.   Serial.begin(115200);
56.   Serial3.begin(115200);
57.   WiFi.init(&Serial3);
58.
59.   if (WiFi.status() == WL_NO_SHIELD) {
60.     Serial.println("WiFi shield not present");
61.     // don't continue
62.     while (true);
63.   }
```

```
64.   lcd.init();
65.   lcd.backlight();
66.   lcd.clear();
67.   lcd.setCursor (0, 0);
68.   lcd.print("Initializing...");
69.   delay(2000);
70.   lcd.clear();
71.   lcd.setCursor (0, 0);
72.   lcd.print("Welcome to");
73.   lcd.setCursor (0, 1);
74.   lcd.print("Smart Farming");
75.   lcd.setCursor (0, 2);
76.   lcd.print("Hidroponik");
77.   delay(2000);
78.
79.   while ( status != WL_CONNECTED) {
80.     Serial.print("Attempting to connect to WPA SS
    ID: ");
81.     Serial.println(ssid);
82.     lcd.clear();
83.     lcd.setCursor (0, 0);
84.     lcd.print("Attempting to");
85.     lcd.setCursor (0, 1);
86.     lcd.print("connect to WPA SSID: ");
87.     lcd.setCursor (0, 2);
88.     lcd.print(ssid);
89.     status = WiFi.begin(ssid, pass);
90.   }
91.   Serial.println("You're connected to the network
    ");
92.   lcd.clear();
93.   lcd.setCursor (0, 0);
94.   lcd.print("You're connected");
95.   lcd.setCursor (0, 1);
96.   lcd.print(" to the network");
97.
98.   printWifiStatus();
99.   waktuMulai = millis();
100.       }
```

Kode 5. 3 Method setup()

Kode 5.3 merupakan kode yang terdapat pada *method* void setup(). Method ini hanya dijalankan satu kali oleh Arduino dinyalakan dan digunakan untuk menghubungkan antara *library*

dan pin komponen pendukung perangkat keras. Pada kode ini konfigurasi relay, sensor pH, *total dissolved solids*, temperatur, dan ketinggian air dilakukan. Selanjutnya koneksi Arduino dengan modul Wifi dilakukan.

```
101.       void loop() {
102.         if (waktuMintaData < millis() - waktuMu
    lai) {
103.           getData(1);
104.           waktuMulai = millis();
105.         }
106.         if (waktuKontrolAktuator < millis() - w
    aktuMulai)  {
107.           getValueFromSensors();
108.           if (pHValue > pHMaxLimit) {
109.             digitalWrite(relay3, HIGH);
110.             delay(1000);
111.             digitalWrite(relay3, LOW);
112.           }
113.           if (tdsValue < tdsMinLimit) {
114.             digitalWrite(relay1, HIGH);
115.             digitalWrite(relay2, HIGH);
116.             delay(1000);
117.             digitalWrite(relay1, LOW);
118.             digitalWrite(relay2, LOW);
119.           }
120.           if (distance < distanceMinLimit) {
121.             digitalWrite(relay4, HIGH);
122.             delay(3000);
123.             digitalWrite(relay4, LOW);
124.           }
125.         }
126.
127.         if (waktuTampilkanLCD < millis() - wakt
    uMulai)  {
128.           delay(2000);
129.           getValueFromSensors();
130.           lcd.clear();
131.           lcd.setCursor(0, 0);
132.           lcd.print("Suhu: ");
133.           lcd.print(temperature);
134.           lcd.print("^C");
135.
136.           lcd.setCursor(0, 1);
```

```
137.            lcd.print("TDS: ");
138.            lcd.print(tdsValue);
139.            lcd.print("ppm");
140.
141.            lcd.setCursor(0, 2);
142.            lcd.print("Tinggi Air: ");
143.            lcd.print(distance);
144.            lcd.print("%");
145.
146.            lcd.setCursor(0, 3);
147.            lcd.print("pH: ");
148.            lcd.print(pHValue);
149.          }
150.
151.          while (client.available()) {
152.            String line = client.readStringUntil(
    '\r\n');
153.            if (lines_received == 14) {
154.              jsonTemp = line;
155.            }
156.            lines_received++;
157.          }
158.
```

Kode 5. 4 Method loop() awal

Pada Kode 5.4 terdapat *method* loop(). Method ini dijalankan oleh Arduino berulang-ulang tanpa berhenti. Kemudian pada kode ini juga dijalankan timer untuk menentukan keputusan berdasarkan batas bawah pada nilai pembacaan sensor. Data pembacaan nilai sensor ditampilkan pada LCD Display.

```
159.          if (jsonTemp.length() > 0) {
160.            Serial.println("from server: " + json
    Temp);
161.            const size_t capacity = JSON_ARRAY_SI
    ZE(4) + JSON_OBJECT_SIZE(2) + 4 * JSON_OBJECT_SIZ
    E(4) + 350;
162.            DynamicJsonDocument doc(capacity);
163.            const char* json;
164.            deserializeJson(doc, json);
165.            JsonArray data = doc["data"];
166.
167.            JsonObject data_0 = data[0];
```

```
168.            const char* data_0_sensor_min_limit =
    data_0["sensor_min_limit"]; //
169.            const char* data_0_sensor_max_limit =
    data_0["sensor_max_limit"]; //
170.
171.            JsonObject data_1 = data[1];
172.            const char* pHMinLimit = data_1["sens
    or_min_limit"]; //
173.            const char* pHMaxLimit = data_1["sens
    or_max_limit"];
174.
175.            JsonObject data_2 = data[2];
176.            const char* tdsMinLimit = data_2["sen
    sor_min_limit"];
177.            const char* tdsMaxLimit = data_2["sen
    sor_max_limit"];
178.
179.            JsonObject data_3 = data[3];
180.            const char* distanceMinLimit = data_3
    ["sensor_min_limit"]; "
181.            const char* distanceMaxLimit = data_3
    ["sensor_max_limit"];
182.          }
```

Kode 5. 5 Parsing Data Batas Bawah dan Batas Atas

Pada Kode 5.5, Arduino mendapatkan nilai sensor dari aplikasi web *API* dalam bentuk JSON dan harus dilakukan *parse* terlebih dahulu menggunakan *library* ArduinoJSON. Setelah berhasil dilakukan *parse* maka nilai tersebut akan dilakukan *assign* pada variabel global.

```
183.        void getValueFromSensors() {
184.          temperature = getTemperature();
185.          tdsValue = getTDSValue();
186.          distance = getDistance();
187.          pHValue = getpHValue();
188.        }
```

Kode 5. 6 Method getValueFromSensors()

Kode 5.6 memperlihatkan penanganan pembacaan nilai sensor terhadap seluruh function.

```
189.          float getpHValue() {
190.          static unsigned long timepoint = millis
    ();
191.          float voltage = analogRead(PH_PIN) / 10
    24.0 * 5000;
192.          if (millis() - timepoint > 1000U)
193.          {
194.            timepoint = millis();
195.            pHValueTemp = ph.readPH(voltage, temp
    erature);
196.           }
197.          ph.calibration(voltage, temperature);
198.            return pHValueTemp;
199.        }
```

Kode 5. 7 Method getpHValue()

Kode 5.7 menunjukkan pembacaan pH dari sensor pH menggunakan *library* khusus.

```
200.          float getTDSValue() {
201.          gravityTds.setTemperature(temperature);
202.          gravityTds.update();
203.          float tdsValueTemp = gravityTds.getTdsV
    alue();
204.          Serial.print(tdsValueTemp);
205.          Serial.println("ppm");
206.          return tdsValueTemp;
207.        }
```

Kode 5. 8 Method getTDSValue()

Kode 5.8 menunjukkan pembacaan *total dissolved solids* dari sensor *total dissolved solids* menggunakan *library* khusus.

```
208.        float getTemperature() {
209.          tempSensors.requestTemperatures();
210.          float temperatureTemp = tempSensors.get
    TempCByIndex(0);
211.          Serial.print("Water Temp: ");
212.          Serial.print(temperatureTemp);
213.          Serial.println("^C");
214.
215.          return temperatureTemp;
```

```
216.       }
```

Kode 5. 9 Method getTemperature()

Kode 5.8 menunjukkan pembacaan *temperatur* dari sensor *total dissolved solids* menggunakan *library* khusus.

```
217.       float getDistance() {
218.         float duration, distanceTemp;
219.         pinMode(pingPin, OUTPUT);
220.         digitalWrite(pingPin, LOW);
221.         delayMicroseconds(2);
222.         digitalWrite(pingPin, HIGH);
223.         delayMicroseconds(20);
224.         digitalWrite(pingPin, LOW);
225.         pinMode(echoPin, INPUT);
226.         duration = pulseIn(echoPin, HIGH);
227.         distanceTemp =  microsecondsToCentimete
    rs(duration);
228.         float percentage = ((28.0 - distanceTem
    p) / 28.0) * 100;
229.         Serial.print(28 - distanceTemp);
230.         Serial.print("cm");
231.         Serial.println();
232.         return percentage;
233.       }
234.
235.       long microsecondsToCentimeters(long micro
    seconds) {
236.         return microseconds / 29 / 2;
237.       }
```

Kode 5. 10 Method getDistance()

Kode 5.8 menunjukkan pembacaan ketinggian air dari sensor ketinggian air menggunakan perhitungan dasar ultrasonik.

```
238.       void postData(float sensor_id, float sens
    or_value) {
239.         Serial.println();
240.         Serial.println("Starting connection to
    server...");
241.         if (client.connect(server, 80)) {
242.           String data;
```

```
243.            data = "sensor_id=";
244.            data += sensor_id;
245.            data += "&";
246.            data += "sensor_value=";
247.            data += sensor_value;
248.            Serial.println("Connected to server")
    ;
249.            //tambahin data dari sensor lain
250.            // Make a HTTP request
251.            client.println("POST /ayotics-
    backend/log HTTP/1.1");
252.            client.print("Host: ");
253.            client.println(server);
254.            client.println("Access-Control-Allow-
    Methods: GET, POST, PUT, DELETE, PATCH, OPTIONS")
    ;
255.            client.println("Access-Control-Allow-
    Headers: Access-Control-Allow-Methods, Access-
    Control-Allow-Origin, Access-Control-Allow-
    Headers, Accept, Accept-  Version,  Content-
    Length, Content-MD5, Content-Type, Date, X-Api-
    Version, x-api-key, X-Response-Time, X-
    PINGOTHER, X-CSRF-Token,Authorization");
256.            client.println("Access-Control-
    Expose-Headers: *");
257.            client.println("Access-Control-Allow-
    Origin: *");
258.            client.println("Access-Control-Allow-
    Methods: GET, POST, PUT, DELETE, PATCH, OPTIONS")
    ;
259.            client.println("Connection: Keep-
    Alive");
260.            client.println("Content-
    Type: application/x-www-form-urlencoded");
261.            client.print("Content-Length: ");
262.            client.println(data.length());
263.            client.println();
264.            client.println(data);
265.            client.stop();
266.
267.            Serial.println("Data Terkirim, Connec
    tion Close");
268.            Serial.println(data.length());
269.            Serial.println(data);
270.            return true;
```

```
271.          } else {
272.            return false;
273.          }
274.        }
```

Kode 5. 11 Pengiriman Data method postData()

Kode 5.11 menjelaskan pengiriman data ke aplikasi web *API* menggunakan *request* POST.

```
275.        void getData(int microcontroller_id) {
276.          Serial.println();
277.          client.stop();
278.          if (client.connect(server, 80)) {
279.            Serial.println("Connecting...");
280.            // send the HTTP PUT request
281.            client.print(F("GET /ayotics-
    backend/sensor/"));
282.            client.print(microcontroller_id);
283.            client.print(F("/min_max_limit"));
284.            client.println(F(" HTTP/1.1"));
285.            client.println(F("Host: 192.168.100.9
    1"));
286.            client.println("Connection: close");

287.            client.println();
288.          }
289.          else {
290.            Serial.println("Connection failed");

291.          }
292.        }
```

Kode 5. 12 Method getData()

Kode 5.12 menjelaskan pengiriman data ke aplikasi web *API* menggunakan *request* GET.

```
293.        void printWifiStatus() {
294.          // print the SSID of the network you're
     attached to
295.          Serial.print("SSID: ");
296.          Serial.println(WiFi.SSID());
297.
```

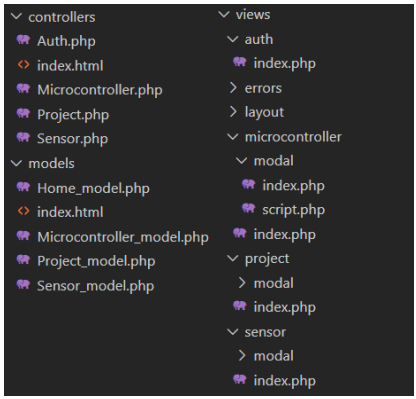
```
298.         // print your WiFi shield's IP address
299.         IPAddress ip = WiFi.localIP();
300.         Serial.print("IP Address: ");
301.         Serial.println(ip);
302.
303.         // print the received signal strength
304.         long rssi = WiFi.RSSI();
305.         Serial.print("Signal strength (RSSI):";
306.         Serial.print(rssi);
307.         Serial.println(" dBm");
308.
309.         IPAddress gateway = WiFi.gatewayIP();
310.         Serial.print("gateway:");
311.         Serial.print(gateway);
312.         Serial.println(" ");
313.       }
```

Kode 5. 13 Method printWifiStatus()

Kode 5.13 menjelaskan status modul WiFi yang sedang berjalan ke *Serial Monitor* agar mudah melakukan proses *debug*.

5.3 Implementasi Aplikasi Web *Client*

Implementasi sistem dilakukan dengan pembuatan aplikasi web *client* sesuai dengan desain yang telah dirancang pada *class diagram* dan bab sebelumnya. Berikut adalah *file* hasil implementasi kode program aplikasi web *client* yang ditunjukkan pada Gambar 5.5



Gambar 5. 5 Hasil Implementasi Aplikasi Web Client

5.3.1 Implementasi *Model* Aplikasi Web *Client*

Model pada aplikasi web *client* bertindak sebagai kode program yang mengirim setiap *request* ke aplikasi web *API*. *Model* juga berfungsi dalam melakukan pengolahan data yang diterima dari aplikasi web *API*. Berikut adalah kode untuk masing-masing *Model*.

```
1.  <?php
2.
3.  use GuzzleHttp\Client;
4.
5.  class Project_model extends CI_Model {
6.   public function get_projects_by_user_id($user_id) {

7.    $client = new Client();
8.    $response = $client->get($this->config-
       >item('backend_url') . 'project/user_id/' . $user
       _id . '', [
9.     'headers' => [
```

```
10.       'Content-Type' => 'application/x-www-form-
          urlencoded',
11.       'Accept' => 'application/json',
12.       'Authorization' => $this->session-
          >userdata('tokenJWT')
13.     ],
14.    ]);
15.    $result = json_decode($response->getBody()-
          >getContents(), true);
16.    return ($result);
17.   }
18.
19.   public function get_projects() {
20.    $client = new Client();
21.
22.    $response = $client->get($this->config-
          >item('backend_url') . 'project/', [
23.     'headers' => [
24.       'Content-Type' => 'application/json',
25.       'Accept' => 'application/json',
26.       'Authorization' => $this->session-
          >userdata('tokenJWT')
27.     ],
28.    ]);
29.    $result = json_decode($response->getBody()-
          >getContents(), true);
30.    return ($result);
31.   }
32.   public function create_project($data) {
33.    $client = new Client();
34.    $response = $client->post($this->config-
          >item('backend_url') . 'project/', [
35.     'headers' => [
36.       'Content-Type' => 'application/x-www-form-
          urlencoded',
37.       'Accept' => 'application/x-www-form-
          urlencoded',
38.       'Authorization' => $this->session-
          >userdata('tokenJWT')
39.     ],
40.     'form_params'     => $data
41.    ]);
42.    $result = json_decode($response->getBody()-
          >getContents(), true);
43.    return $result;
```

```
44.   }
45.
46.   public function edit_project($data) {
47.    $client = new Client();
48.    $response = $client->put($this->config-
        >item('backend_url') . 'project/', [
49.     'headers' => [
50.      'Content-Type' => 'application/x-www-form-
       urlencoded',
51.      'Accept' => 'application/x-www-form-
       urlencoded',
52.      'Authorization' => $this->session-
       >userdata('tokenJWT')
53.     ],
54.     'form_params'     => $data
55.    ]);
56.    $result = json_decode($response->getBody()-
        >getContents(), true);
57.    return $result;
58.   }
59.
60.   public function delete_project($data) {
61.    $client = new Client();
62.    $response = $client->delete($this->config-
        >item('backend_url') . 'project/', [
63.     'headers' => [
64.      'Content-Type' => 'application/x-www-form-
       urlencoded',
65.      'Accept' => 'application/x-www-form-
       urlencoded',
66.      'Authorization' => $this->session-
       >userdata('tokenJWT')
67.     ],
68.     'form_params'     => $data
69.    ]);
70.    $result = json_decode($response->getBody()-
        >getContents(), true);
71.    return $result;
72.   }
73. }
```

Kode 5. 14 Model Project_model

Kode 5. 2 merupakan *Model* pada Project_model.php. Terdapat 5 *function* yaitu get_projects_by_user_id() digunakan untuk

mendapatkan daftar proyek berdasarkan user_id pengguna, get_projects digunakan untuk mendapatkan seluruh proyek pada *database*, create_project() digunakan untuk membuat data proyek baru, edit_project() digunakan untuk mengubah data yang sudah ada menggunakan id, dan delete_project() digunakan untuk menghapus data proyek menggunakan id.

```
1.  <?php
2.
3.  use GuzzleHttp\Client;
4.
5.  class Microcontroller_model extends CI_Model {
6.   public function get_microcontrollers($project_id
     = NULL) {
7.    $client = new Client();
8.    $response = $client->get($this->config-
    >item('backend_url') . 'microcontroller/', [
9.     'headers' => [
10.     'Content-Type' => 'application/json',
11.     'Accept' => 'application/json',
12.     'Authorization' => $this->session-
    >userdata('tokenJWT')
13.    ],
14.   ]);
15.   $result = json_decode($response->getBody()-
    >getContents(), true);
16.   return ($result);
17.  }
18.  public function get_microcontrollers_by_project_
    id($project_id = NULL) {
19.   $client = new Client();
20.   $response = $client->get($this->config-
    >item('backend_url') . 'microcontroller/project_i
    d/' . $project_id . '', [
21.    'headers' => [
22.     'Content-Type' => 'application/json',
23.     'Accept' => 'application/json',
24.     'Authorization' => $this->session-
    >userdata('tokenJWT')
25.    ],
26.   ]);
27.   $result = json_decode($response->getBody()-
    >getContents(), true);
28.   return ($result);
```

```
29.  }
30.  public function create_microcontroller($data) {

31.   $client = new Client();
32.   $response = $client->post($this->config-
    >item('backend_url') . 'microcontroller/', [
33.    'headers' => [
34.     'Content-Type' => 'application/x-www-form-
    urlencoded',
35.     'Accept' => 'application/x-www-form-
    urlencoded',
36.     'Authorization' => $this->session-
    >userdata('tokenJWT')
37.    ],
38.    'form_params'     => $data
39.   ]);
40.   $result = json_decode($response->getBody()-
    >getContents(), true);
41.   return $result;
42.  }
43.  public function edit_microcontroller($data) {
44.   $client = new Client();
45.   $response = $client->put($this->config-
    >item('backend_url') . 'microcontroller/', [
46.    'headers' => [
47.     'Content-Type' => 'application/x-www-form-
    urlencoded',
48.     'Accept' => 'application/x-www-form-
    urlencoded',
49.     'Authorization' => $this->session-
    >userdata('tokenJWT')
50.    ],
51.    'form_params'     => $data
52.   ]);
53.   $result = json_decode($response->getBody()-
    >getContents(), true);
54.   return $result;
55.  }
56.  public function delete_microcontroller($data) {

57.   $client = new Client();
58.   $response = $client->delete($this->config-
    >item('backend_url') . 'microcontroller/', [
59.    'headers' => [
```

```
60.     'Content-Type' => 'application/x-www-form-
    urlencoded',
61.     'Accept' => 'application/x-www-form-
    urlencoded',
62.     'Authorization' => $this->session-
    >userdata('tokenJWT')
63.    ],
64.    'form_params'    => $data
65.   ]);
66.   $result = json_decode($response->getBody()-
    >getContents(), true);
67.   return $result;
68.  }
69. }
```

Kode 5. 15 Model Microcontroller_model

Kode 5.3 merupakan hasil implementasi dari *model* Microcontroller_model.php. Pada *model* ini terdapat 5 function yaitu, get_microcontrollers() berfungsi untuk mendapatkan seluruh data mikrokontroler pada *database*, get_microcontrollers_by_project_id() berfungsi untuk mendapatkan data mikrokontroler berdasarkan project_id pengguna, create_microcontroller() berguna untuk membuat data mikrokontroler baru, edit_microcontroller() berfungsi untuk mengubah data mikrokontroler yang sudah ada menggunakan id, dan delete_microcontroller() digunakan untuk menghapus data mikrokontroler menggunakan id.

```
1.  <?php
2.
3.  use GuzzleHttp\Client;
4.
5.  class Sensor_model extends CI_Model {
6.   public function get_sensors_by_microcontroller_i
    d($microcontroller_id = NULL) {
7.    $client = new Client();
8.    $response = $client->get($this->config-
    >item('backend_url') . 'sensor/microcontroller_id
    /' . $microcontroller_id, [
9.     'headers' => [
10.     'Content-Type' => 'application/json',
11.     'Accept' => 'application/json',
```

```
12.     'Authorization' => $this->session->userdata('tokenJWT')
13.    ],
14.   ]);
15.   $result = json_decode($response->getBody()->getContents(), true);
16.
17.   return ($result);
18.  }
19.
20.  public function get_logs_by_microcontroller_id($microcontroller_id = NULL) {
21.   $client = new Client();
22.   $response = $client->get($this->config->item('backend_url') . "log/microcontroller_id/" . $microcontroller_id . '', [
23.    'headers' => [
24.     'Content-Type' => 'application/json',
25.     'Accept' => 'application/json',
26.     'Authorization' => $this->session->userdata('tokenJWT')
27.    ],
28.   ]);
29.   $result = json_decode($response->getBody()->getContents(), true);
30.   return ($result);
31.  }
32.
33.  public function create_sensor($data) {
34.   $client = new Client();
35.
36.   $response = $client->post($this->config->item('backend_url') . 'sensor/', [
37.    'headers' => [
38.     'Content-Type' => 'application/x-www-form-urlencoded',
39.     'Accept' => 'application/x-www-form-urlencoded',
40.     'Authorization' => $this->session->userdata('tokenJWT')
41.    ],
42.    'form_params'     => $data
43.   ]);
44.   $result = json_decode($response->getBody()->getContents(), true);
```

```
45.   return $result;
46.  }
47.  public function edit_sensor($data) {
48.   $client = new Client();
49.   $response = $client->put($this->config-
    >item('backend_url') . 'sensor/', [
50.    'headers' => [
51.     'Content-Type' => 'application/x-www-form-
    urlencoded',
52.     'Accept' => 'application/x-www-form-
    urlencoded',
53.     'Authorization' => $this->session-
    >userdata('tokenJWT')
54.    ],
55.    'form_params'    => $data
56.   ]);
57.   $result = json_decode($response->getBody()-
    >getContents(), true);
58.   return $result;
59.  }
60.
61.  public function delete_sensor($data) {
62.   $client = new Client();
63.   $response = $client->delete($this->config-
    >item('backend_url') . 'sensor/', [
64.    'headers' => [
65.     'Content-Type' => 'application/x-www-form-
    urlencoded',
66.     'Accept' => 'application/x-www-form-
    urlencoded',
67.     'Authorization' => $this->session-
    >userdata('tokenJWT')
68.    ],
69.    'form_params'    => $data
70.   ]);
71.   $result = json_decode($response->getBody()-
    >getContents(), true);
72.   return $result;
73.  }
74. }
```

Kode 5. 16 Model Sensor_model

Kode 5.3 merupakan hasil implementasi dari *model* Sensor_model.php. Pada *model* ini terdapat 6 function yaitu,

get_logs_by_microcontrollers_id() berfungsi untuk mendapatkan data log berdasarkan microcontroller_id pengguna, get_sensors_by_microcontroller_id() berfungsi untuk mendapatkan data sensor berdasarkan microcontroller_id pengguna, get_sensors() berfungsi untuk mendapatkan seluruh data sensor pada *database*, create_sensor() berguna untuk membuat data sensor baru, edit_sensor() berfungsi untuk mengubah data sensor yang sudah ada menggunakan id, dan delete_microcontroller() digunakan untuk menghapus data sensor menggunakan id.

5.3.2 Implementasi *Controller* Aplikasi Web *Client*

```
1.  <?php
2.  defined('BASEPATH') or exit('No direct script acc
    ess allowed');
3.
4.  class Project extends CI_Controller {
5.   public function __construct() {
6.    parent::__construct();
7.    $this->load->helper('auth');
8.    is_logged_in();
9.    $this->load->model('Project_model');
10.  }
11.
12.  public function index() {
13.   redirect(base_url('project/user_id/' . $this-
    >session->userdata('id')));
14.  }
15.
16.  public function user_id($user_id = NULL) {
17.
18.   if ($user_id == $this->session-
    >userdata('id')) {
19.    $data['title'] = 'Dashboard - Project Ayotics'
    ;
20.    $data['project'] = $this->Project_model-
    >get_projects_by_user_id($user_id);
21.    $data['user'] = $data['project']['user'];
22.    $newdata = array(
```

```
23.     'fullname'  => $data['user']['user_fullname']
    ,
24.     'email'  => $data['user']['user_email'],
25.    );
26.    $this->session->set_userdata($newdata);
27.    $data['userid'] = $this->session-
    >userdata('id');
28.
29.    $this->load->helper('url');
30.    $this->load->view('layout/header', $data);
31.    $this->load->view('layout/sidebar', $data);
32.    $this->load->view('project/index', $data);
33.   } else {
34.    redirect(base_url('project/user_id/' . $this-
    >session->userdata('id')));
35.   }
36.  }
37.
38.  public function add_project() {
39.   $data = [
40.    'user_id' => $this->session->userdata('id'),
41.    'project_name' => $this->input->post('project-
    name'),
42.    'project_institution' => $this->input-
    >post('project-institution'),
43.    'project_city' => $this->input->post('project-
    city'),
44.    'project_img' => $this->input->post('project-
    img),
45.   ];
46.   $this->Project_model->create_project($data);
47.   redirect(base_url('project/user_id/' . $this-
    >session->userdata('id')));
48.  }
49.
50.  public function edit_project() {
51.   $data = [
52.    'project_id' => $this->input->post('project-
    id'),
53.    'user_id' => $this->session->userdata('id'),
54.    'project_name' => $this->input->post('project-
    name'),
55.    'project_institution' => $this->input-
    >post('project-institution'),
```

```
56.    'project_city' => $this->input->post('project-
    city'),
57.    'project_img' => $this->input->post('project-
    img),
58.   ];
59.   $this->Project_model->edit_project($data);
60.   redirect(base_url('project/user_id/' . $this-
    >session->userdata('id')));
61.  }
62.
63.  public function delete_project() {
64.   $data = [
65.    'project_id' => $this->input->post('project-
    id'),
66.   ];
67.   $this->Project_model->delete_project($data);
68.   redirect(base_url('project/user_id/' . $this-
    >session->userdata('id')));
69.  }
70. }
```

Kode 5. 17 Controller Project

Pada Kode 5.5 Controller Project.php, terdapat 6 function yaitu, __consruct() digunakan untuk menginisialisasi *utility function*, index() digunakan untuk bila pengguna mengakses *route* web tanpa parameter khusus, halaman akan *redirect* ke halaman dashboard proyek, user_id() digunakan untuk menampilkan tampilan dashboard proyek dan mengirimkan *request* GET untuk mendapatkan data proyek berdasarkan user_id pengguna, add_project() digunakan untuk menangani FORM yang kemudian akan dilakukan *request* POST ke aplikasi web *API* untuk membuat data proyek baru, edit_project() digunakan untuk menangani FORM yang kemudian akan dilakukan *request* PUT ke aplikasi web *API* untuk mengubah data proyek, delete_project() digunakan untuk menangani FORM yang kemudian akan dilakukan *request* DELETE ke aplikasi web *API* untuk menghapus data proyek.

```
1.  <?php
2.  defined('BASEPATH') or exit('No direct script acc
    ess allowed');
```

```
3.
4.  class Microcontroller extends CI_Controller {
5.   public function __construct() {
6.    parent::__construct();
7.    $this->load->helper('auth');
8.    is_logged_in();
9.    $this->load-
    >model('Microcontroller_model');
10.  }
11.
12.  public function index($project_id = NULL) {
13.   redirect(base_url('microcontroller/project_id/'
    . $this->session->userdata('project_id')));
14.  }
15.
16.  public function project_id($project_id = NULL) {

17.   $newdata = [
18.    'project_id' => $project_id
19.   ];
20.   $this->session->set_userdata($newdata);
21.   if ($project_id == $this->session-
    >userdata('project_id')) {
22.    $data['title'] = 'Dashboard - Microcontroller
    Ayotics';
23.    $data['microcontroller'] = $this-
    >Microcontroller_model-
    >get_microcontrollers_by_project_id($project_id);

24.    $data['user'] = $data['microcontroller']['user
    '];
25.    $this->load->helper('url');
26.    $this->load->view('layout/header', $data);
27.    $this->load-
    >view('layout/sidebar', $data);
28.    $this->load-
    >view('microcontroller/index', $data);
29.   } else {
30.    redirect(base_url('microcontroller/project_id/
    ' . $this->session->userdata('project_id')));
31.   }
32.  }
33.  public function add_microcontroller() {
34.   $data = [
```

```
35.    'project_id' => $this->session->userdata('project_id'),
36.    'microcontroller_name' => $this->input->post('microcontroller-name'),
37.    'microcontroller_description' => $this->input->post('microcontroller-description'),
38.    'microcontroller_img' => $this->input->post('microcontroller-img),
39.   ];
40.   $this->Microcontroller_model->create_microcontroller($data);
41.   redirect(base_url('microcontroller/project_id/' . $this->session->userdata('project_id')));
42.  }
43.
44.  public function edit_microcontroller() {
45.   $data = [
46.    'microcontroller_id' => $this->input->post('microcontroller-id'),
47.    'project_id' => $this->session->userdata('project_id'),
48.    'microcontroller_name' => $this->input->post('microcontroller-name'),
49.    'microcontroller_description' => $this->input->post('microcontroller-description'),
50.    'microcontroller_img' => $this->input->post('microcontroller-img),
51.   ];
52.   $this->Microcontroller_model->edit_microcontroller($data);
53.   redirect(base_url('microcontroller/project_id/' . $this->session->userdata('project_id')));
54.  }
55.
56.  public function delete_microcontroller() {
57.   $data = [
58.    'microcontroller_id' => $this->input->post('microcontroller-id'),
59.   ];
60.   $this->Microcontroller_model->delete_microcontroller($data);
61.   redirect(base_url('microcontroller/project_id/' . $this->session->userdata('project_id')));
62.  }
63. }
```

Kode 5. 18 Controller Microcontroller

Pada Kode 5.6 Controller Microcontroller.php, terdapat 6 function yaitu, __consruct() digunakan untuk menginisialisasi *utility function*, index() digunakan untuk bila pengguna mengakses *route* web tanpa parameter khusus, halaman akan *redirect* ke halaman dashboard mikrokontroler, project_id() digunakan untuk menampilkan tampilan dashboard mikrokontroler dan mengirimkan *request* GET untuk mendapatkan data mikrokontroler berdasarkan project_id pengguna, add_microcontroller() digunakan untuk menangani FORM yang kemudian akan dilakukan *request* POST ke aplikasi web *API* untuk membuat data mikrokontroler baru, edit_project() digunakan untuk menangani FORM yang kemudian akan dilakukan *request* PUT ke aplikasi web *API* untuk mengubah data mikronkontroler, delete_project() digunakan untuk menangani FORM yang kemudian akan dilakukan *request* DELETE ke aplikasi web *API* untuk menghapus data mikrokontroler.

```
1.  <?php
2.  defined('BASEPATH') or exit('No direct script acc
    ess allowed');
3.
4.  class Sensor extends CI_Controller {
5.   public function __construct() {
6.    parent::__construct();
7.    $this->load->helper('auth');
8.    is_logged_in();
9.    $this->load->model('Sensor_model');
10.  }
11.
12.  public function index() {
13.   redirect(base_url('sensor/microcontroller_id/'
    . $this->session-
    >userdata('microcontroller_id')));
14.  }
15.  public function microcontroller_id($microcontrol
    ler_id = NULL) {
16.   $newdata = [
17.    'microcontroller_id' => $microcontroller_id
18.   ];
```

```
19.   $this->session->set_userdata($newdata);
20.   if ($microcontroller_id == $this->session-
    >userdata('microcontroller_id')) {
21.    $data['title'] = 'Dashboard - Data Log Ayotics
    ';
22.    $data['log'] = $this->Sensor_model-
    >get_logs_by_microcontroller_id($microcontroller_
    id);
23.    $data['user'] = $data['log']['user'];
24.    $this->load->helper('url');
25.    $this->load->view('layout/header', $data);
26.    $this->load->view('layout/sidebar', $data);
27.    $this->load->view('sensor/index', $data);
28.   } else {
29.    redirect(base_url('sensor/microcontroller_id/'
     . $this->session-
    >userdata('microcontroller_id')));
30.   }
31.  }
32.  public function add_sensor() {
33.   $data = [
34.    'microcontroller_id' => $this->session-
    >userdata('microcontroller_id'),
35.    'sensor_name' => $this->input->post('sensor-
    name'),
36.    'sensor_description' => $this->input-
    >post('sensor-description'),
37.    'sensor_unit' => $this->input->post('sensor-
    unit'),
38.    'sensor_value' => "0",
39.    'sensor_min_limit' => $this->input-
    >post('sensor-min-limit'),
40.    'sensor_max_limit' => $this->input-
    >post('sensor-max-limit'),
41.    'sensor_min_value' => $this->input-
    >post('sensor-min-value'),
42.    'sensor_max_value' => $this->input-
    >post('sensor-max-value'),
43.   ];
44.   $this->Sensor_model->create_sensor($data);
45.   redirect(base_url('sensor/microcontroller_id/'
    . $this->session-
    >userdata('microcontroller_id')));
46.  }
47.
```

```
48.  public function edit_sensor() {
49.   $data = [
50.    'sensor_id' => $this->input->post('sensor-
    id'),
51.    'microcontroller_id' => $this->session-
    >userdata('microcontroller_id'),
52.    'sensor_name' => $this->input->post('sensor-
    name'),
53.    'sensor_description' => $this->input-
    >post('sensor-description'),
54.    'sensor_unit' => $this->input->post('sensor-
    unit'),
55.    'sensor_value' => "0",
56.    'sensor_min_limit' => $this->input-
    >post('sensor-min-limit'),
57.    'sensor_max_limit' => $this->input-
    >post('sensor-max-limit'),
58.    'sensor_min_value' => $this->input-
    >post('sensor-min-value'),
59.    'sensor_max_value' => $this->input-
    >post('sensor-max-value'),
60.   ];
61.   $this->Sensor_model->edit_sensor($data);
62.   redirect(base_url('sensor/microcontroller_id/'
    . $this->session-
    >userdata('microcontroller_id')));
63.  }
64.
65.  public function delete_sensor() {
66.   $data = [
67.    'sensor_id' => $this->input->post('sensor-
    id'),
68.   ];
69.   $this->Sensor_model->delete_sensor($data);
70.   redirect(base_url('sensor/microcontroller_id/'
    . $this->session-
    >userdata('microcontroller_id')));
71.  }
72. }
```

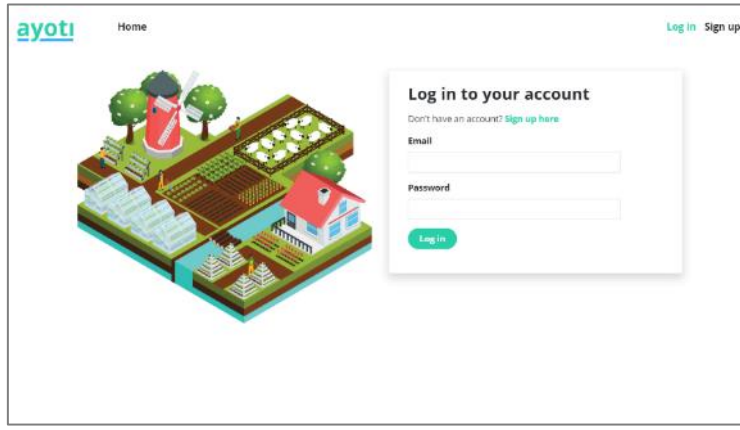
Kode 5. 19 Controller Sensor

Pada Kode 5.7 Controller Sensor.php, terdapat 6 function yaitu, __consruct() digunakan untuk menginisialisasi *utility function*, index() digunakan untuk bila pengguna mengakses *route* web

tanpa parameter khusus, halaman akan *redirect* ke halaman dashboard sensor, microcontroller_id() digunakan untuk menampilkan tampilan dashboard sensor dan mengirimkan *request* GET untuk mendapatkan data sensor berdasarkan microcontroller_id pengguna, add_sensor() digunakan untuk menangani FORM yang kemudian akan dilakukan *request* POST ke aplikasi web *API* untuk membuat data sensor baru, edit_sensor() digunakan untuk menangani FORM yang kemudian akan dilakukan *request* PUT ke aplikasi web *API* untuk mengubah data sensor, delete_project() digunakan untuk menangani FORM yang kemudian akan dilakukan *request* DELETE ke aplikasi web *API* untuk menghapus data sensor.

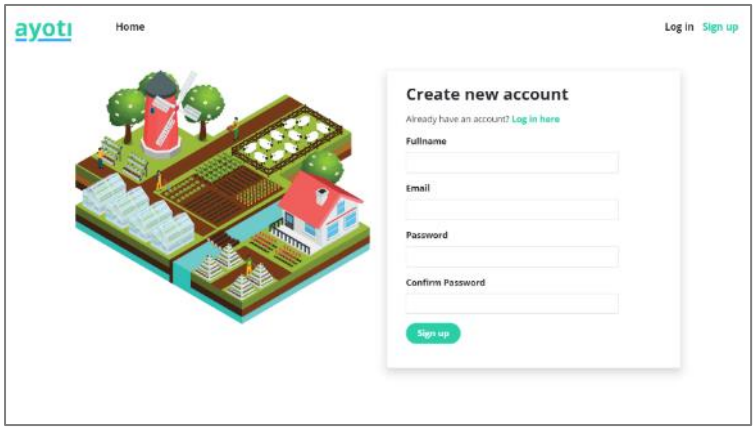
5.3.3 Implementasi *View* Aplikasi Web *Client*

Pada aplikasi web *client* terdapat komponen *view* yang diakses secara langsung oleh pengguna. Masing-masing *controller* yang telah dijelaskan pada bab sebelumnya akan *load* masing-masing komponen *view*. Berikut adalah penjelasan dari masing-masing komponen *view* yang telah diimplementasikan pada aplikasi web *client* yang ditunjukkan pada Gambar 5.6 hingga Gambar 5.19.



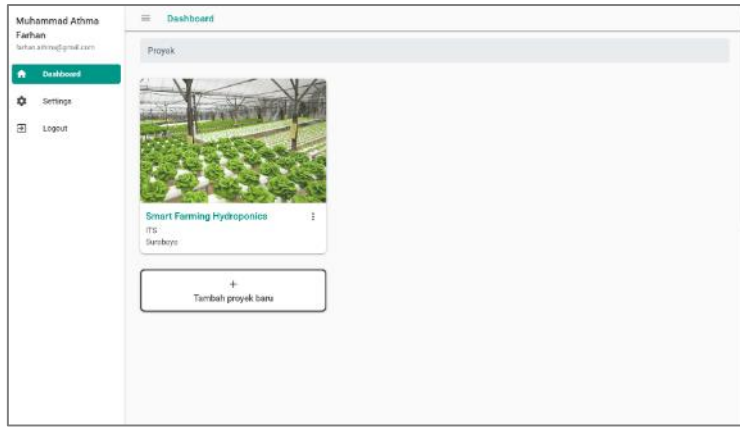
Gambar 5. 6 Tampilan Halaman Login

Pengguna akan mengakses halaman login untuk dapat masuk ke aplikasi web *client* dengan mengisikan email dan *password* kemudian mengeklik tombol “Log in” pada halaman login yang ditunjukkan pada Gambar 5.6.



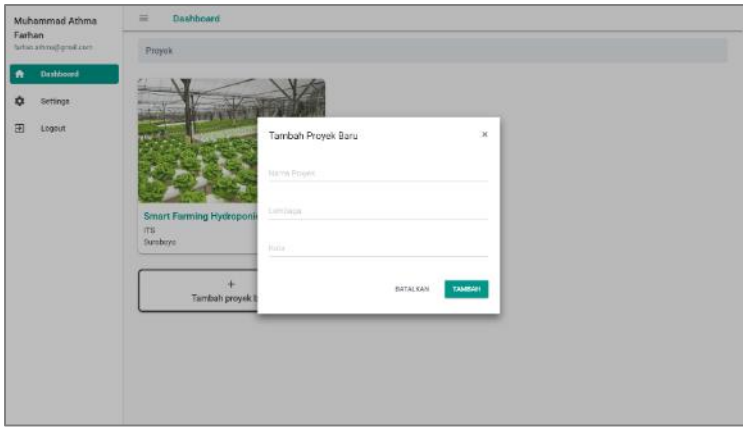
Gambar 5. 7 Tampilan Halaman Sign Up

Jika pengguna belum memiliki akun, maka pengguna akan mengakses halaman *sign up* untuk membuat akun baru yang ditunjukkan seperti Gambar 5.7. Pengguna dapat membuat akun baru dengan mengisikan nama lengkap pengguna, email, *password*, dan konfirmasi *password* kemudian pengguna mengeklik tombol “Sign up”. Jika proses pembuatan akun baru berhasil, aplikasi web *client* akan memberi pemberitahuan bahwa pembuatan akun baru telah berhasil lalu pengguna akan *redirect* ke halaman login secara otomatis.



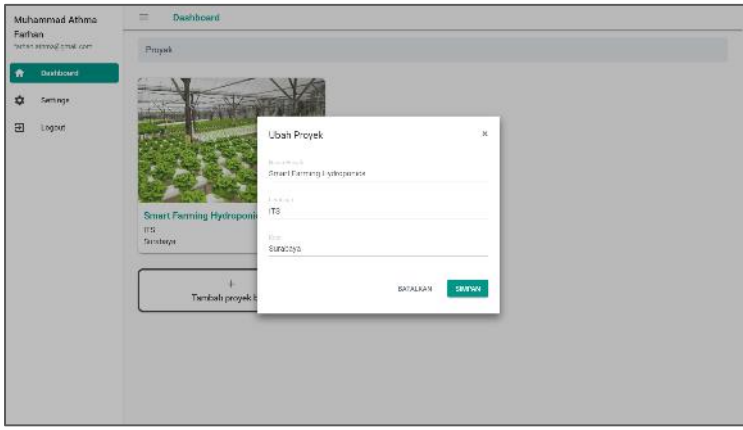
Gambar 5. 8 Tampilan Halaman Dashboard Proyek

Gambar 5.8 meunjukkan bahwa pengguna berhasil masuk ke aplikasi web *client* dan langsung diarahkan ke halaman dashboard proyek. Proyek yang dimaksud dalam halaman *dashboard* ini berdasarkan pengguna dalam memberi kategori untuk memberi batasan masing-masing proyek yang dimilikinya. Jumlah proyek yang diatur bergantung pada kebutuhan masing-masing pengguna. Pada halaman tersebut terdapat *card* proyek yang berisi informasi nama proyek, nama lembaga, dan nama kota proyek. Jika pengguna ingin mengakses halaman dashboard mikrokontroler, maka pengguna akan mengeklik *card* proyek yang berkaitan dengan mikrokontroler sesuai proyek tersebut. Pengguna dapat membuat proyek baru dengan mengeklik tombol "Tambah proyek baru".



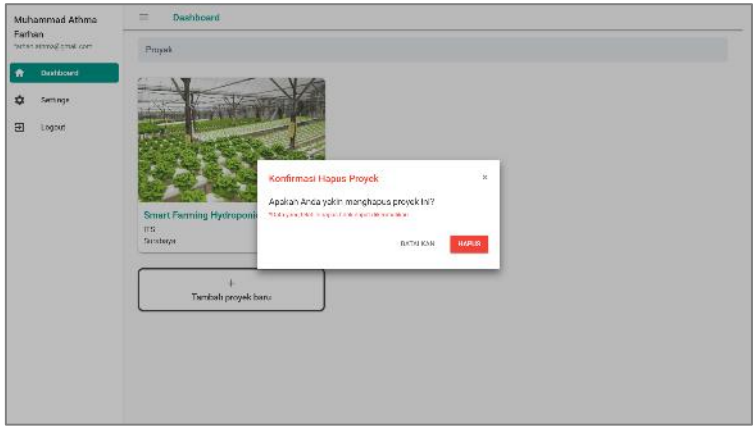
Gambar 5. 9 Halaman Tambah Proyek Baru

Jika tombol "Tambah proyek baru" telah diklik, maka akan muncul *pop-out modal* yang berisi *form* data. Pengguna akan mengisikan seluruh *form* kemudian mengeklik tombol "Tambah" untuk membuat data proyek baru. Pengguna dapat mengubah nama dan deskripsi proyek dengan mengeklik tombol "tiga bulatan kecil" yang terletak pada *card* proyek, kemudian pengguna mengeklik tombol "Ubah".



Gambar 5. 10 Halaman Ubah Proyek

Jika tombol “Ubah” telah diklik, maka akan muncul *pop-out modal* yang berisikan *form* data yang telah berisi deskripsi data proyek seperti yang ditunjukkan pada Gambar 5.10. Jika ingin melakukan perubahan pada data proyek, pengguna dapat mengeklik tombol “Simpan”. Pengguna juga dapat menghapus data proyek dengan mengklik tombol “Hapus” setelah mengklik tombol “tiga bulatan kecil”.

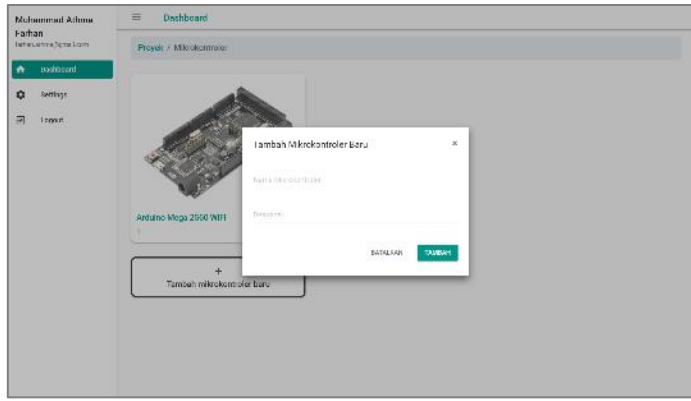


Gambar 5. 11 Halaman Hapus Proyek

Jika tombol “Hapus” berhasil diklik, akan muncul *pop-out modal* konfirmasi apakah pengguna yakin untuk menghapus proyek seperti yang ditunjukkan pada Gambar 5.11. Jika pengguna mengeklik tombol “Hapus” maka data proyek akan terhapus.

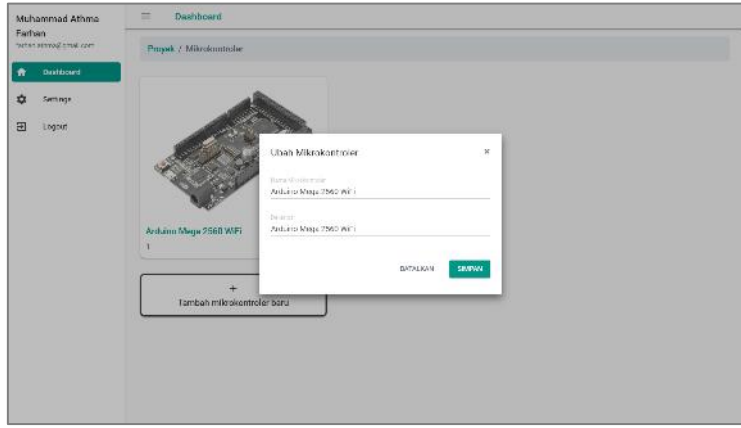
Gambar 5. 12 Tampilan Halaman Dashboard Mikrokontroler

Halaman dashboard mikrokontroler akan muncul seperti yang ditunjukkan pada Gambar 5.12 jika pengguna mengklik *card* mikrokontroler. Mikrokontroler yang dimaksud dalam halaman *dashboard* ini yaitu berdasarkan banyaknya mikrokontroler yang digunakan oleh pengguna dalam proyek terkait, sehingga jumlah mikrokontroler yang digunakan pengguna bergantung pada kebutuhan masing-masing proyek dan pengguna. Pada halaman tersebut terdapat *card* mikrokontroler yang berisi informasi nama mikrokontroler dan id dari mikrokontroler. Jika pengguna ingin mengakses halaman dashboard sensor dan log data, maka pengguna mengeklik *card* mikrokontroler yang berkaitan dengan mikrokontroler sesuai mikrokontroler tersebut. Pengguna dapat membuat mikrokontroler baru dengan mengeklik tombol “Tambah mikrokontroler baru”.



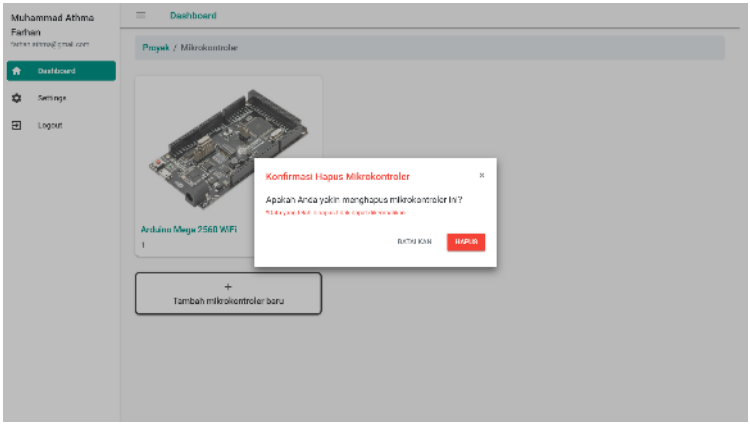
Gambar 5. 13 Halaman Tambah Mikrokontroler Baru

Jika tombol "Tambah mikrokontroler baru" telah diklik, maka akan muncul *pop-out modal* yang berisi *form* data. Pengguna akan mengisikan seluruh *form* kemudian mengeklik tombol "Tambah" untuk membuat data mikrokontroler baru. Pengguna dapat mengubah nama dan deskripsi mikrokontroler dengan mengeklik tombol "tiga bulatan kecil" yang terletak pada *card* mikrokontroler, kemudian pengguna mengeklik tombol "Ubah".



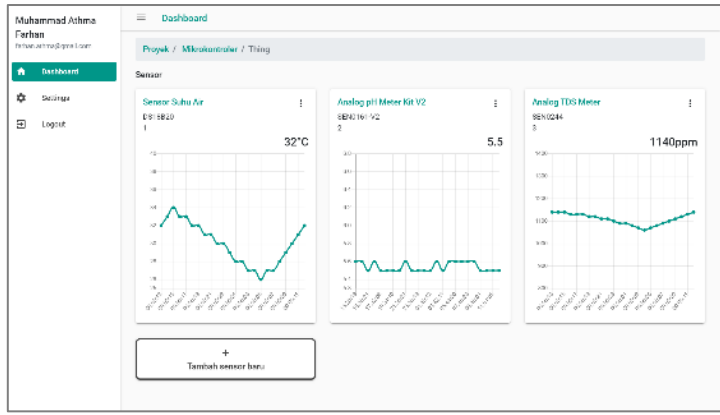
Gambar 5. 14 Halaman Ubah Mikrokontroler

Jika tombol “Ubah” telah diklik, maka akan muncul *pop-out modal* yang berisikan *form* data yang telah berisi deskripsi data mikrokontroler seperti yang ditunjukkan pada Gambar 5.14. Jika ingin melakukan perubahan pada data mikrokontroler, pengguna dapat mengeklik tombol “Simpan”. Pengguna juga dapat menghapus data mikrokontroler dengan mengklik tombol “Hapus” setelah mengklik tombol “tiga bulatan kecil”.



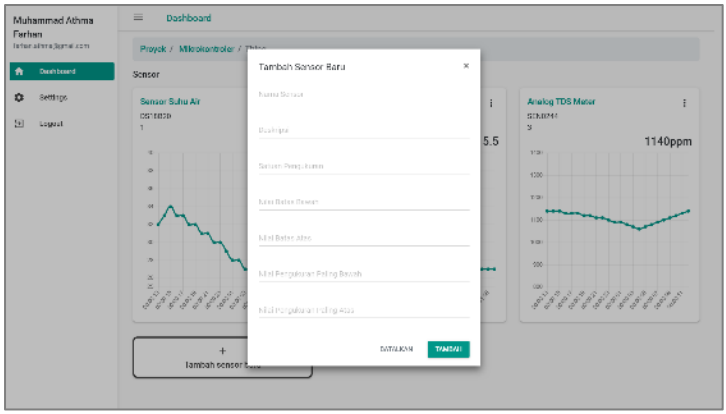
Gambar 5. 15 Halaman Hapus Mikrokontroler

Jika tombol “Hapus” berhasil diklik, akan muncul *pop-out modal* konfirmasi apakah pengguna yakin untuk menghapus mikrokontroler seperti yang ditunjukkan pada Gambar 5.15. Jika pengguna mengeklik tombol “Hapus” maka data mikrokontroler akan terhapus.



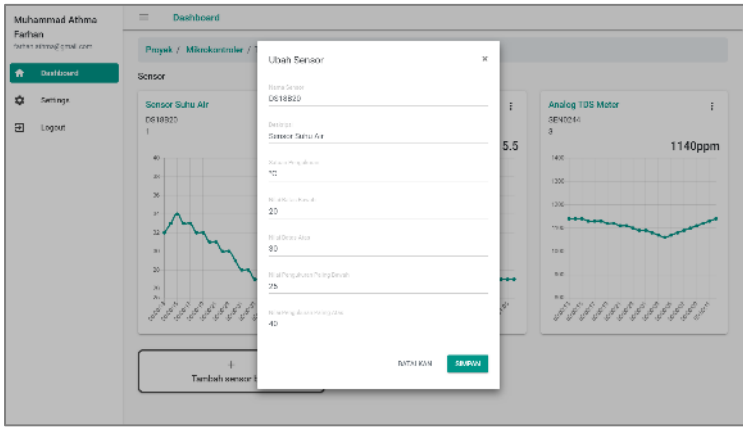
Gambar 5. 16 Tampilan Halaman Dashboard Sensor

Halaman dashboard sensor akan muncul seperti yang ditunjukkan pada Gambar 5.16 jika pengguna mengeklik *card* sensor. Pada halaman tersebut terdapat *card* sensor datalog yang berisi informasi nama sensor, deskripsi sensor, id sensor, dan log tiap masing-masing sensor. Namun kekurangan pada halaman tersebut adalah tidak tersedianya data status aktuator terkini dan data log sensor yang ditampilkan tidak mendekati *real-time* dikarenakan hanya menggunakan protokol HTTP, tidak secepat menggunakan protokol MQTT. Protokol HTTP hanya dapat mengirim satu arah koneksi, sedangkan protokol MQTT dapat mengirimkan koneksi dua arah sehingga dapat meningkatkan performa dari kecepatan data yang didapatkan. Jika aplikasi web menggunakan MQTT maka status aktuator seperti katup solenoida sedang membuka atau menutup dapat ditampilkan, data log yang ditampilkan bersifat *real-time*, dan aplikasi dapat menyimpan data log aktuator ke dalam *database*. Pengguna dapat membuat sensor baru dengan mengeklik tombol “Tambah sensor baru”.



Gambar 5. 17 Halaman Tambah Sensor

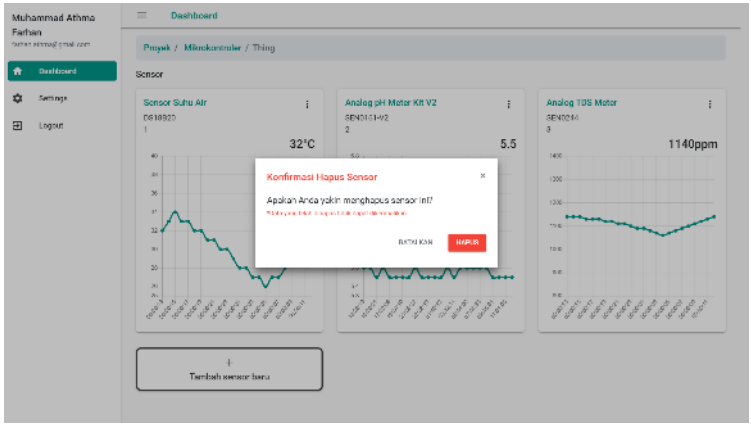
Jika tombol “Tambah sensor baru” telah diklik, maka akan muncul *pop-out modal* yang berisi *form* data. Pengguna akan mengisikan seluruh *form* kemudian mengeklik tombol “Tambah” untuk membuat data sensor baru. Pengguna dapat mengubah data sensor dengan mengeklik tombol “tiga bulatan kecil” yang terletak pada *card* sensor, kemudian pengguna mengeklik tombol “Ubah”.



Gambar 5. 18 Halaman Ubah Sensor

Jika tombol “Ubah” telah diklik, maka akan muncul *pop-out modal* yang berisikan *form* data yang telah berisi deskripsi data

sensor seperti yang ditunjukkan pada Gambar 5.18. Jika ingin melakukan perubahan pada data mikrokontroler, pengguna dapat mengeklik tombol “Simpan”. Pengguna juga dapat menghapus data sensor dengan mengklik tombol “Hapus” setelah mengklik tombol “tiga bulatan kecil”.

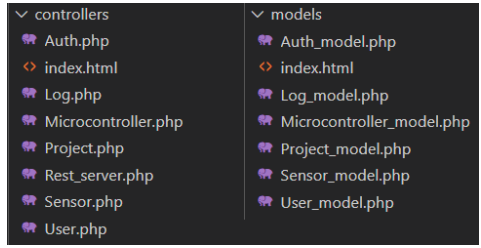


Gambar 5. 19 Halaman Hapus Sensor

Jika tombol “Hapus” berhasil diklik, akan muncul *pop-out modal* konfirmasi apakah pengguna yakin untuk menghapus sensor seperti yang ditunjukkan pada Gambar 5.19. Jika pengguna mengeklik tombol “Hapus” maka data sensor akan terhapus.

5.4 Implementasi Aplikasi Web *API*

Implementasi sistem dilakukan dengan pembuatan aplikasi web *client* sesuai dengan desain yang telah dirancang pada *class diagram* dan bab sebelumnya. Berikut adalah *file* hasil implementasi kode program aplikasi web *client* yang ditunjukkan pada Gambar 5.20.



Gambar 5. 20 Hasil Implementasi Aplikasi Web API

5.4.1 Implementasi *Model* Aplikasi Web *API*

Model pada aplikasi web *API* bertindak sebagai kode program yang melakukan *query* secara langsung ke *database*. Berikut adalah kode untuk masing-masing *Model*.

```
1.  <?php
2.  class Auth_model extends CI_Model {
3.   public function login($user_email, $user_passwor
    d) {
4.    return $this->db->get_where('users', [
5.     'user_email' => $user_email,
6.     'user_password' => $user_password
7.    ])->row_array();
8.   }
9.  }
```

Kode 5. 20 Model Auth_model

Kode 5.20 menunjukkan bahwa *model* Auth_model.php hanya memiliki 1 *function* yaitu login() yang digunakan untuk login menggunakan username dan password.

```
1.  <?php
2.  class User_model extends CI_Model {
3.   public function create_user($data) {
4.    $this->db->insert('users', $data);
5.    return $this->db->affected_rows();
6.   }
7.  }
```

Kode 5. 21 Model User_model

Kode 5.21 menunjukkan *model* User_model.php hanya memiliki 1 *function* yaitu create_user() berfungsi untuk membuat akun baru.

```
1.  <?php
2.  class Project_model extends CI_Model {
3.   public function get_project($id = null) {
4.    if ($id === null) {
5.     return $this->db->get('projects')-
    >result_array();
6.    } else {
7.     return $this->db->get_where('projects', [
8.      'project_id' => $id
9.     ])->result_array();
10.   }
11.  }
12.
13.  public function get_project_by_user_id($id = nul
    l) {
14.   if ($id === null) {
15.    return $this->db->get('projects')-
    >result_array();
16.   } else {
17.
18.    return $this->db->get_where('projects', [
19.     'user_id' => $id
20.    ])->result_array();
21.   }
22.  }
23.  public function create_project($data) {
24.   $this->db->insert('project', $data);
25.   return $this->db->affected_rows();
26.  }
27.
28.  public function update_project($data, $id) {
29.   $this->db->where('project_id', $id);
30.   $this->db->update('projects', $data);
31.   return $this->db->affected_rows();
32.  }
33.
34.  public function delete_project($id) {
35.   $this->db->where('project_id', $id);
36.   $this->db->delete('projects');
37.   return $this->db->affected_rows();
```

```
38.  }
39. }
```

Kode 5. 22 Model Project_model

Kode 5.22 menunjukkan *model* Project_modal.php memiliki 5 *function* yaitu, get_project() melakukan *query* ke *database* untuk mendapatkan seluruh proyek, get_project_by_user_id() melakukan *query* ke *database* untuk mendapatkan proyek berdasarkan user_id, create_project() melakukan *query* ke *database* untuk membuat data proyek baru, update_project() melakukan *query* ke *database* untuk mengubah data proyek, dan delete_project() melakukan *query* ke *database* untuk menghapus data proyek.

```
1.  <?php
2.  class Microcontroller_model extends CI_Model {
3.   public function get_microcontroller($id = null)
    {
4.    if ($id === null) {
5.     return $this->db->get('microcontrollers')-
    >result_array();
6.    } else {
7.     return $this->db-
    >get_where('microcontrollers', [
8.      'microcontroller_id' => $id
9.     ])->result_array();
10.   }
11.  }
12.  public function get_microcontroller_by_project_i
    d($id = null) {
13.   if ($id === null) {
14.    return $this->db->get('microcontrollers')-
    >result_array();
15.   } else {
16.    return $this->db-
    >get_where('microcontrollers', [
17.     'project_id' => $id
18.    ])->result_array();
19.   }
20.  }
21.  public function create_microcontroller($data) {
22.   $this->db->insert('microcontroller', $data);
```

```
23.   return $this->db->affected_rows();
24.  }
25.  public function update_microcontroller($data, $i
    d) {
26.   $this->db->where('microcontroller_id', $id);
27.   $this->db->update('microcontrollers', $data);
28.   return $this->db->affected_rows();
29.  }
30.  public function delete_microcontroller($id) {
31.   $this->db->where('microcontroller_id', $id);
32.   $this->db->delete('microcontrollers');
33.   return $this->db->affected_rows();
34.  }
35. }
```

Kode 5. 23 Model Microcontroller_model

Kode 5.23 menunjukkan *model* Microcontroller_modal.php memiliki 5 *function* yaitu, get_microcontroller() melakukan *query* ke *database* untuk mendapatkan seluruh mikrokontroler, get_microcontroller_by_project_id() melakukan *query* ke *database* untuk mendapatkan mik berdasarkan project_id, create_microcontroller() melakukan *query* ke *database* untuk membuat data mikrokontroler baru, update_microcontroller() melakukan *query* ke *database* untuk mengubah data mikrokontroler, dan delete_microcontroller() melakukan *query* ke *database* untuk menghapus data mikrokontroler.

```
1.  <?php
2.  class Sensor_model extends CI_Model {
3.   public function get_sensor($id = null) {
4.    if ($id === null) {
5.     return $this->db->get('sensors')-
    >result_array();
6.    } else {
7.     return $this->db->get_where('sensors', [
8.      'sensor_id' => $id
9.     ])->result_array();
10.   }
11.  }
12.  public function get_sensor_by_microcontoller_id(
    $id = null) {
13.   if ($id === null) {
```

```
14.    return $this->db->get('sensors')-
    >result_array();
15.   } else {
16.    return $this->db->get_where('sensors', [
17.     'microcontroller_id' => $id
18.    ])->result_array();
19.   }
20.  }
21.  public function create_sensor($data) {
22.   $this->db->insert('sensors', $data);
23.   return $this->db->affected_rows();
24.  }
25.  public function update_sensor($data, $id) {
26.   $this->db->where('sensor_id', $id);
27.   $this->db->update('sensors', $data);
28.   return $this->db->affected_rows();
29.  }
30.  public function delete_sensor($id) {
31.   $this->db->where('sensor_id', $id);
32.   $this->db->delete('sensors');
33.   return $this->db->affected_rows();
34.  }
35. }
```

Kode 5. 24 Model Sensor_model

Kode 5.24 menunjukkan *model* Sensor_model.php memiliki 5 *function* yaitu, get_sensor() melakukan *query* ke *database* untuk mendapatkan seluruh sensor, get_sensor_by_microcontroller_id() melakukan *query* ke *database* untuk mendapatkan sensor berdasarkan microcontroller_id, create_sensor() melakukan *query* ke *database* untuk membuat data sensor baru, update_sensor() melakukan *query* ke *database* untuk mengubah data sensor, dan delete_sensor() melakukan *query* ke *database* untuk menghapus data sensor.

```
1.  <?php
2.  class Log_model extends CI_Model {
3.   public function get_log($id = null) {
4.    if ($id === null) {
5.     return $this->db->get('logs')-
    >result_array();
```

```
    } else {
     return $this->db->get_where('logs', [
      'log_id' => $id
     ])->result_array();
    }
   }
   public function get_log_by_sensor_id($id = null)
   {
    if ($id === null) {
     return $this->db->get('logs')-
>result_array();
    } else {
     $sql = "SELECT *
      FROM (
       SELECT *
       FROM logs
       WHERE sensor_id=" . $id . "
       ORDER BY log_id
       DESC LIMIT 24
      ) sub
     ORDER BY log_id ASC";
     $query = $this->db->query($sql);
     return $query->result_array();
    }
   }
   public function create_log($data) {
    $this->db->insert('logs', $data);
    return $this->db->affected_rows();
   }
}
```

Kode 5. 25 Model Log_model

Kode 5.25 menunjukkan *model* Log_model.php memiliki 3 *function* yaitu, get_log() melakukan *query* ke *database* untuk mendapatkan seluruh log, get_log_by_sensor_id() melakukan *query* ke *database* untuk mendapatkan log berdasarkan sensor_id, dan create_log() melakukan *query* ke *database* untuk membuat data log baru.

5.4.2 Implementasi *Controller* Aplikasi Web *API*

```
<?php
defined('BASEPATH') or exit('No direct script access allowed');
require APPPATH . '/libraries/REST_Controller.php';
class Auth extends REST_Controller {

 public function __construct() {
  parent::__construct();
  $this->load->model('Auth_model');
 }

 public function login_post() {
  $user_email = $this->post('user_email');
  $user_password = $this->post('user_password');
  $user = $this->Auth_model->login($user_email, $user_password);
  if ($user_email === $user['user_email'] && $user_password === $user['user_password']) {
   unset($user['user_password']);
   $token = AUTHORIZATION::generateToken($user);
   $status = parent::HTTP_OK;
   $response = [
    'status' => $status,
    'token' => $token,
    'user' => $user
   ];
   $this->response($response, $status);
  } else {
   $this->response(['message' => 'Invalid email or password!'], parent::HTTP_NOT_FOUND);
  }
 }
}
```

Kode 5. 26 Controller Auth

Kode 5.26 menunjukkan bahwa Auth.php memiliki 2 *function* yaitu construct() digunakan untuk menginisialisasi *load model*

dan login_post() digunakan dalam menerima *request* POST untuk melakukan login.

```
1.  <?php
2.  header("Access-Control-Allow-
    Methods: DELETE, POST, GET, OPTIONS");
3.  defined('BASEPATH') or exit('No direct script acc
    ess allowed');
4.  require APPPATH . '/libraries/REST_Controller.php
    ';
5.
6.  class User extends REST_Controller {
7.   public function __construct() {
8.    parent::__construct();
9.    $this->load->model('User_model');
10.  }
11.
12.  public function index_post() {
13.   $user_email = $this->post('user_email');
14.   $user_password = $this-
    >post('user_password');
15.   $user_fullname = $this-
    >post('user_fullname');
16.   $data = [
17.    'user_email' => $user_email,
18.    'user_password' => $user_password,
19.    'user_fullname' => $user_fullname,
20.   ];
21.   if ($this->db-
    >get_where('users', ['user_email' => $user_email]
    )->row_array() == NULL) {
22.    if ($this->User_model-
    >create_user($data) > 0) {
23.     $status = parent::HTTP_CREATED;
24.     $response = [
25.      'status' => $status,
26.      'message' => 'Data User berhasil ditambah!'
27.     ];
28.     $this->response($response, $status);
29.    }
30.   } else {
31.    $status = parent::HTTP_CONFLICT;
32.    $response = [
33.     'status' => $status,
```

```
34.     'message' => 'Email telah digunakan!'
35.    ];
36.    $this->response($response, $status);
37.   }
38.  }
39. }
```

Kode 5. 27 Controller User

Kode 5.27 menunjukkan bahwa User.php memiliki 2 *function* yaitu construct() digunakan untuk menginisialisasi *load model* dan index_post() digunakan dalam menerima *request* POST untuk membuat *user* baru.

```
1.  private function verify_request() {
2.   $headers = $this->input->request_headers();
3.   $token = $headers['Authorization'];
4.   try {
5.    $user = AUTHORIZATION::validateToken($token);
6.    if ($user === false) {
7.     $status = parent::HTTP_UNAUTHORIZED;
8.     $response = ['status' => $status, 'msg' => 'Un
    authorized Access!', 'token' => $token];
9.     $this->response($response, $status);
10.    exit();
11.   } else {
12.    return $user;
13.   }
14.  } catch (Exception $e) {
15.   $status = parent::HTTP_UNAUTHORIZED;
16.   $response = ['status' => $status, 'msg' => 'Una
    uthorized Access! ', ''];
17.   $this->response($response, $status);
18.  }
19. }
```

Kode 5. 28 Validasi JWT

Di setiap seluruh masing-masing *function* memliki *function* ini yaitu verify_request(), *function* ini berguna untuk melakukan validasi *header Authorization* yang berisikan JWT. Jika JWT salah, maka akan memberikan respons 401 *Unauthorized*.

```
1.  <?php
2.  header("Access-Control-Allow-Origin: *");
3.  header("Access-Control-Allow-
    Methods: DELETE, POST, GET, OPTIONS");
4.  defined('BASEPATH') or exit('No direct script acc
    ess allowed');
5.
6.  require APPPATH . '/libraries/REST_Controller.php
    ';
7.
8.  class Project extends REST_Controller {
9.   public function __construct() {
10.   parent::__construct();
11.   $this->load->model('Project_model');
12.  }
13.
14.  public function index_get() {
15.   $user = $this->verify_request();
16.   $id = $this->get('project_id');
17.   if ($id === null) {
18.    $project = $this->db->get('projects')-
    >result_array();
19.   } elseif ($id <= 0) {
20.    $this->response(null, 400);
21.   } else {
22.
23.    $id = (int) $id;
24.    $project = $this->Project_model-
    >get_project($id);
25.   }
26.   $status = parent::HTTP_OK;
27.   $response = [
28.    'id' => $id,
29.    'status' => $status,
30.    'user' => $user,
31.    'data' => $project
32.   ];
33.   $this->response($response, $status);
34.  }
35.  public function user_id_get() {
36.   $user = $this->verify_request();
37.   $id = $this->get('user_id');
38.   if ($id === null) {
39.    $project = $this->db->get('projects')-
    >result_array();
```

```
40.   } elseif ($id <= 0) {
41.    $this->response(null, 400);
42.   } else {
43.    $id = (int) $id;
44.    $project = $this->Project_model-
    >get_project_by_user_id($id);
45.   }
46.   $status = parent::HTTP_OK;
47.   $response = [
48.    'id' => $id,
49.    'status' => $status,
50.    'user' => $user,
51.    'data' => $project
52.   ];
53.   $this->response($response, $status);
54.  }
55.
56.  public function index_post() {
57.   $user = $this->verify_request();
58.   $data = [
59.    'user_id' => $this->post('user_id'),
60.    'project_name' => $this-
    >post('project_name'),
61.    'project_institution' => $this-
    >post('project_institution'),
62.    'project_city' => $this-
    >post('project_city'),
63.    'project_img' => $this->post('project_img'),
64.   ];
65.   $this->db->insert('projects', $data);
66.   if ($this->db->affected_rows() > 0) {
67.    $status = parent::HTTP_CREATED;
68.    $response = ['status' => $status, 'user' => $u
    ser, 'message' => 'Data Project berhasil ditambah
    !'];
69.    $this->response($response, $status);
70.   }
71.  }
72.
73.  public function index_put() {
74.   $user = $this->verify_request();
75.   $id = $this->put('project_id');
76.   $data = [
77.    'user_id' => $this->put('user_id'),
```

```
78.    'project_name' => $this-
    >put('project_name'),
79.    'project_institution' => $this-
    >put('project_institution'),
80.    'project_city' => $this-
    >put('project_city'),
81.    'project_img' => $this->put('project_img'),
82.   ];
83.   $this->Project_model-
    >update_project($data, $id);
84.   if ($this->db->affected_rows() > 0) {
85.    $status = parent::HTTP_OK;
86.    $response = ['status' => $status, 'user' => $u
    ser, 'message' => 'Data Project berhasil diubah!'
    ];
87.    $this->response($response, $status);
88.   }
89.  }
90.
91.  public function index_delete() {
92.   $user = $this->verify_request();
93.   $id = $this->delete('project_id');
94.   $this->Project_model->delete_project($id);
95.   if ($this->db->affected_rows() > 0) {
96.    $status = parent::HTTP_OK;
97.    $response = ['status' => $status, 'user' => $u
    ser, 'message' => 'Data Project berhasil dihapus!
    '];
98.    $this->response($response, $status);
99.   }
100.        }
101.       }
```

Kode 5. 29 Controller Project

Kode 5.29 menunjukkan bahwa Project.php memiliki 6 function yaitu, construct() digunakan untuk menginisialisasi *load model*, index_get() digunakan dalam menerima *request* GET untuk mendapatkan seluruh data proyek, user_id_get() digunakan dalam menerima *request* GET untuk mendapatkan data proyek berdasarkan user_id pengguna, index_post() digunakan dalam menerima *request* POST untuk membuat data proyek baru, index_put() digunakan dalam menerima *request* PUT untuk

mengubah data proyek, dan index_delete() digunakan dalam menerima *request* DELETE untuk menghapus data proyek.

```
1.  <?php
2.  header("Access-Control-Allow-Origin: *");
3.  header("Access-Control-Allow-
    Methods: DELETE, POST, GET, OPTIONS");
4.  defined('BASEPATH') or exit('No direct script acc
    ess allowed');
5.
6.  require APPPATH . '/libraries/REST_Controller.php
    ';
7.
8.  class Microcontroller extends REST_Controller {
9.   public function __construct() {
10.   parent::__construct();
11.   $this->load->model('Microcontroller_model');
12.  }
13.
14.  public function index_get() {
15.   $user = $this->verify_request();
16.   $id = $this->get('microcontroller_id');
17.   if ($id === null) {
18.    $microcontroller = $this-
    >Microcontroller_model->get_microcontroller();
19.   } elseif ($id <= 0) {
20.    $this->response(null, 400);
21.   } else {
22.
23.    $id = (int) $id;
24.    $microcontroller = $this-
    >Microcontroller_model-
    >get_microcontroller($id);
25.   }
26.   $status = parent::HTTP_OK;
27.   $response = [
28.    'id' => $id,
29.    'status' => $status,
30.    'user' => $user,
31.    'data' => $microcontroller
32.   ];
33.   $this->response($response, $status);
34.  }
35.  public function project_id_get() {
```

```
36.   $user = $this->verify_request();
37.   if ($id === null) {
38.    $microcontroller = $this->db-
    >get('microcontrollers')->result_array();
39.   } elseif ($id <= 0) {
40.    $this->response(null, 400);
41.   } else {
42.    $id = (int) $id;
43.    $microcontroller = $this-
    >Microcontroller_model-
    >get_microcontroller_by_project_id($id);
44.   }
45.   $status = parent::HTTP_OK;
46.   $response = [
47.    'id' => $id,
48.    'status' => $status,
49.    'user' => $user,
50.    'data' => $microcontroller
51.   ];
52.   $this->response($response, $status);
53.  }
54.
55.  public function index_post() {
56.   $user = $this->verify_request();
57.   $data = [
58.    'project_id' => $this->post('project_id'),
59.    'microcontroller_name' => $this-
    >post('microcontroller_name'),
60.    'microcontroller_description' => $this-
    >post('microcontroller_description'),
61.    'microcontroller_img' => $this-
    >post('microcontroller_img'),
62.   ];
63.   $this->Microcontroller_model-
    >create_microcontroller($data);
64.   if ($this->db->affected_rows() > 0) {
65.    $status = parent::HTTP_CREATED;
66.    $response = ['status' => $status, 'user' => $u
    ser, 'message' => 'Data Mikrokontroler berhasil d
    itambah!'];
67.    $this->response($response, $status);
68.   }
69.  }
70.
71.  public function index_put() {
```

```
72.   $user = $this->verify_request();
73.   $id = $this->put('microcontroller_id');
74.   $data = [
75.    'project_id' => $this->put('project_id'),
76.    'microcontroller_name' => $this-
    >put('microcontroller_name'),
77.    'microcontroller_description' => $this-
    >put('microcontroller_description'),
78.    'microcontroller_img' => $this-
    >put('microcontroller_img'),
79.   ];
80.   $this->Microcontroller_model-
    >update_microcontroller($data, $id);
81.   if ($this->db->affected_rows() > 0) {
82.    $status = parent::HTTP_OK;
83.    $response = ['status' => $status, 'user' => $u
    ser, 'message' => 'Data Mikrokontroler berhasil d
    iubah!'];
84.    $this->response($response, $status);
85.   }
86.  }
87.
88.  public function index_delete() {
89.   $user = $this->verify_request();
90.   $id = $this->delete('microcontroller_id');
91.   $this->Microcontroller_model-
    >delete_microcontroller($id);
92.   if ($this->db->affected_rows() > 0) {
93.    $status = parent::HTTP_OK;
94.    $response = ['status' => $status, 'user' => $u
    ser, 'message' => 'Data Mikrokontroler berhasil d
    ihapus!'];
95.    $this->response($response, $status);
96.   }
97.  }
98. }
```

Kode 5. 30 Controller Microcontroller

Kode 5.30 menunjukkan bahwa Microcontroller.php memiliki 6 function yaitu, construct() digunakan untuk menginisialisasi *load model*, index_get() digunakan dalam menerima *request* GET untuk mendapatkan seluruh data mikrokontroler, project_id_get() digunakan dalam menerima *request* GET untuk

mendapatkan data mikrokontroler berdasarkan project_id pengguna, index_post() digunakan dalam menerima *request* POST untuk membuat data mikrokontroler baru, index_put() digunakan dalam menerima *request* PUT untuk mengubah data mikrokontroler, dan index_delete() digunakan dalam menerima *request* DELETE untuk menghapus data mikrokontroler.

```
1.  <?php
2.  header("Access-Control-Allow-Origin: *");
3.  header("Access-Control-Allow-
    Methods: DELETE, POST, GET, OPTIONS");
4.  defined('BASEPATH') or exit('No direct script acc
    ess allowed');
5.
6.  require APPPATH . '/libraries/REST_Controller.php
    ';
7.
8.  class Sensor extends REST_Controller {
9.   public function __construct() {
10.   parent::__construct();
11.   $this->load->model('Sensor_model');
12.  }
13.
14.  public function index_get() {
15.   $user = $this->verify_request();
16.   $id = $this->get('sensor_id');
17.   if ($id === null) {
18.    $sensor = $this->db->get('sensors')-
    >result_array();
19.   } elseif ($id <= 0) {
20.    $this->response(null, 400);
21.   } else {
22.    $id = (int) $id;
23.    $sensor = $this->Sensor_model-
    >get_sensor($id);
24.   }
25.   $status = parent::HTTP_OK;
26.   $response = [
27.    'id' => $id,
28.    'status' => $status,
29.    'user' => $user,
30.    'data' => $sensor
31.   ];
32.   $this->response($response, $status);
```

```
33.  }
34.  public function microcontroller_id_get() {
35.   $user = $this->verify_request();
36.   $id = $this->get('microcontroller_id');
37.   if ($id === null) {
38.    $sensor = $this->db->get('sensors')-
    >result_array();
39.   } elseif ($id <= 0) {
40.    $this->response(null, 400);
41.   } else {
42.    $id = (int) $id;
43.    $sensor = $this->Sensor_model-
    >get_sensor_by_microcontoller_id($id);
44.   }
45.   $status = parent::HTTP_OK;
46.   $response = [
47.    'id' => $id,
48.    'status' => $status,
49.    'user' => $user,
50.    'data' => $sensor
51.   ];
52.   $this->response($response, $status);
53.  }
54.
55.  public function index_post() {
56.   date_default_timezone_set('Asia/Jakarta');
57.   $now = date('Y-m-d H:i:s');
58.   $user = $this->verify_request();
59.   $data = [
60.    'microcontroller_id' => $this-
    >post('microcontroller_id'),
61.    'sensor_name' => $this->post('sensor_name'),
62.    'sensor_description' => $this-
    >post('sensor_description'),
63.    'sensor_unit' => $this->post('sensor_unit'),
64.    'sensor_value' => $this-
    >post('sensor_value'),
65.    'sensor_min_limit' => $this-
    >post('sensor_min_limit'),
66.    'sensor_max_limit' => $this-
    >post('sensor_max_limit'),
67.    'sensor_min_value' => $this-
    >post('sensor_min_value'),
68.    'sensor_max_value' => $this-
    >post('sensor_max_value'),
```

```
69.     'sensor_datetime' => $now,
70.   ];
71.   $this->db->insert('sensors', $data);
72.
73.   if ($this->db->affected_rows() > 0) {
74.     $status = parent::HTTP_CREATED;
75.     $response = ['status' => $status, 'user' => $u
    ser, 'message' => 'Data Sensor berhasil ditambah!
    '];
76.     $this->response($response, $status);
77.   }
78.  }
79.
80.  public function index_put() {
81.   date_default_timezone_set('Asia/Jakarta'); # ad
    d your city to set local time zone
82.   $now = date('Y-m-d H:i:s');
83.   $user = $this->verify_request();
84.   $id = $this->put('sensor_id');
85.   $data = [
86.     'microcontroller_id' => $this-
    >put('microcontroller_id'),
87.     'sensor_name' => $this->put('sensor_name'),
88.     'sensor_description' => $this-
    >put('sensor_description'),
89.     'sensor_unit' => $this->put('sensor_unit'),
90.     'sensor_value' => $this-
    >put('sensor_value'),
91.     'sensor_min_limit' => $this-
    >put('sensor_min_limit'),
92.     'sensor_max_limit' => $this-
    >put('sensor_max_limit'),
93.     'sensor_min_value' => $this-
    >put('sensor_min_value'),
94.     'sensor_max_value' => $this-
    >put('sensor_max_value'),
95.     'sensor_datetime' => $now,
96.   ];
97.   $this->Sensor_model-
    >update_sensor($data, $id);
98.   if ($this->db->affected_rows() > 0) {
99.     $status = parent::HTTP_OK;
100.          $response = [
101.            'status' => $status,
102.            'user' => $user,
```

```
103.             'message' => 'Data Sensor berhasil di
    ubah!'
104.           ];
105.           $this->response($response, $status);
106.          }
107.         }
108.
109.         public function index_delete() {
110.          $user = $this->verify_request();
111.          $id = $this->delete('sensor_id');
112.          $this->Sensor_model-
    >delete_sensor($id);
113.          if ($this->db->affected_rows() > 0) {
114.           $status = parent::HTTP_OK;
115.           $response = ['status' => $status, 'use
    r' => $user, 'message' => 'Data Sensor berhasil d
    ihapus!'];
116.           $this->response($response, $status);
117.          }
118.         }
119.
120.         public function min_max_limit_get() {
121.          $id = $this-
    >get('microcontroller_id');
122.          $min_max = ($this->Sensor_model-
    >get_sensor_min_max($id));
123.          for ($i = 0; $i <= sizeof($min_max) - 1
    ; $i++) {
124.           unset($min_max[$i]['sensor_name']);
125.           unset($min_max[$i]['sensor_description
    ']);
126.           unset($min_max[$i]['sensor_unit']);
127.           unset($min_max[$i]['sensor_value']);
128.           unset($min_max[$i]['sensor_min_value']
    );
129.           unset($min_max[$i]['sensor_max_value']
    );
130.           unset($min_max[$i]['sensor_datetime'])
    ;
131.          }
132.          $status = parent::HTTP_OK;
133.          $response = [
134.           'status' => $status,
135.           'data' => $min_max,
136.           'user' => $user,
```

```
        ];
        $this->response($response, $status);
      }
    }
```

Kode 5. 31 Controller Sensor

Kode 5.31 menunjukkan bahwa Sensor.php memiliki 6 function yaitu, construct() digunakan untuk menginisialisasi *load model*, index_get() digunakan dalam menerima *request* GET untuk mendapatkan seluruh data sensor, microcontroller_id_get() digunakan dalam menerima *request* GET untuk mendapatkan data sensor berdasarkan microcontroller_id pengguna, index_post() digunakan dalam menerima *request* POST untuk membuat data sensor baru, index_put() digunakan dalam menerima *request* PUT untuk mengubah data mikrokontroler, dan index_delete() digunakan dalam menerima *request* DELETE untuk menghapus data mikrokontroler.

```
<?php
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-
Methods: DELETE, POST, GET, OPTIONS");
defined('BASEPATH') or exit('No direct script acc
ess allowed');
require APPPATH . '/libraries/REST_Controller.php
';

class Log extends REST_Controller {
 public function __construct() {
  parent::__construct();
  $this->load->model('Log_model');
  $this->load->model('Sensor_model');
 }

 public function index_get() {
  $user = $this->verify_request();
  $id = $this->get('log_id');
  if ($id === null) {
   $log = $this->db->get('logs')-
>result_array();
  } elseif ($id <= 0) {
   $this->response(null, 400);
```

```
21.   } else {
22.    $id = (int) $id;
23.    $log = $this->Log_model->get_log($id);
24.   }
25.   $status = parent::HTTP_OK;
26.   $response = [
27.    'id' => $id,
28.    'status' => $status,
29.    'user' => $user,
30.    'data' => $log
31.   ];
32.   $this->response($response, $status);
33.  }
34.
35.  public function microcontroller_id_get() {
36.   $user = $this->verify_request();
37.   $id = $this->get('microcontroller_id');
38.   if ($id === null) {
39.    $this->response(null, 400);
40.   } elseif ($id <= 0) {
41.    $this->response(null, 400);
42.   } else {
43.    $id = (int) $id;
44.    $list_sensor = $this->Sensor_model-
    >get_sensor_by_microcontoller_id($id);
45.    $list_log = [];
46.    for ($i = 0; $i < sizeof($list_sensor); $i++)
    {
47.     if ($this->Log_model-
    >get_log_by_sensor_id($list_sensor[$i]['sensor_id
    ']) != NULL) {
48.      $temp_list_log['sensor_id'] = $list_sensor[$
    i]['sensor_id'];
49.      $temp_list_log['microcontroller_id'] = $list
    _sensor[$i]['microcontroller_id'];
50.      $temp_list_log['sensor_name'] = $list_sensor
    [$i]['sensor_name'];
51.      $temp_list_log['sensor_description'] = $list
    _sensor[$i]['sensor_description'];
52.      $temp_list_log['sensor_unit'] = $list_sensor
    [$i]['sensor_unit'];
53.      $temp_list_log['sensor_min_limit'] = $list_s
    ensor[$i]['sensor_min_limit'];
54.      $temp_list_log['sensor_max_limit'] = $list_s
    ensor[$i]['sensor_max_limit'];
```

```
55.      $temp_list_log['sensor_min_value'] = $list_s
    ensor[$i]['sensor_min_value'];
56.      $temp_list_log['sensor_max_value'] = $list_s
    ensor[$i]['sensor_max_value'];
57.      $temp_list_log['log'] = $this->Log_model-
    >get_log_by_sensor_id($list_sensor[$i]['sensor_id
    ']);
58.      array_push($list_log, $temp_list_log);
59.     }
60.    }
61.    $status = parent::HTTP_OK;
62.    $response = [
63.     'id' => $id,
64.     'status' => $status,
65.     'user' => $user,
66.     'data' => $list_log
67.    ];
68.    $this->response($response, $status);
69.   }
70.  }
71.
72.  public function index_post() {
73.   $user = $this->verify_request();
74.   $data = [
75.    'sensor_id' => $this->post('sensor_id'),
76.    'sensor_value' => $this-
    >post('sensor_value'),
77.    'log_date' => date("Y-m-d"),
78.    'log_time' => date("H:i:s"),
79.   ];
80.   if ($this->Log_model-
    >create_log($data) > 0) {
81.    $status = parent::HTTP_CREATED;
82.    $response = [
83.     'status' => $status,
84.     'user' => $user,
85.     'message' => 'Data Log berhasil ditambah!'
86.    ];
87.    $this->response($response, $status);
88.   }
89.  }
90. }
```

Kode 5. 32 Controller Log

Kode 5.32 menunjukkan bahwa Sensor.php memiliki 3 function yaitu, construct() digunakan untuk menginisialisasi *load model*, index_get() digunakan dalam menerima *request* GET untuk mendapatkan seluruh data sensor, microcontroller_id_get() digunakan dalam menerima *request* GET untuk mendapatkan data log berdasarkan microcontroller_id pengguna, dan index_post() digunakan dalam menerima *request* POST untuk membuat data log baru.

5.5 Implementasi *Database*

Database dibutuhkan untuk memfasilitasi kebutuhan penyimpanan data pada aplikasi web *API*. *Database MySQL* adalah jenis *dataabase* yang digunakan pada penelitian ini. Untuk memudahkan dalam pengelolaan *database*, digunakan *tools* seperti *phpMyAdmin*. Berikut adalah hasil implementasi yang telah dilakukan dalam bentuk *data definition language* (DDL) untuk melakukan pembuatan struktur *database* yang ditunjukkan pada Kode 5.33.

```
1.  SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
2.  SET AUTOCOMMIT = 0;
3.  START TRANSACTION;
4.  SET time_zone = "+07:00";
5.
6.  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTE
    R_SET_CLIENT */;
7.  /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACT
    ER_SET_RESULTS */;
8.  /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATIO
    N_CONNECTION */;
9.  /*!40101 SET NAMES utf8mb4 */;
10.
11. CREATE TABLE `logs` (
12.   `log_id` int(11) NOT NULL,
13.   `sensor_id` int(11) NOT NULL,
14.   `sensor_value` float NOT NULL,
15.   `log_date` date NOT NULL,
16.   `log_time` time NOT NULL
17. ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
18.
```

```
19. CREATE TABLE `microcontrollers` (
20.   `microcontroller_id` int(11) NOT NULL,
21.   `project_id` int(16) NOT NULL,
22.   `microcontroller_name` varchar(64) NOT NULL,
23.   `microcontroller_description` varchar(128) NOT
    NULL,
24.   `microcontroller_img` varchar(64) NOT NULL
25. ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
26.
27. CREATE TABLE `projects` (
28.   `project_id` int(11) NOT NULL,
29.   `user_id` int(16) NOT NULL,
30.   `project_name` varchar(64) NOT NULL,
31.   `project_institution` varchar(128) NOT NULL,
32.   `project_city` varchar(64) NOT NULL,
33.   `project_img` varchar(128) NOT NULL
34. ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
35.
36. CREATE TABLE `sensors` (
37.   `sensor_id` int(11) NOT NULL,
38.   `microcontroller_id` int(16) NOT NULL,
39.   `sensor_name` varchar(32) NOT NULL,
40.   `sensor_description` varchar(64) NOT NULL,
41.   `sensor_unit` varchar(16) NOT NULL,
42.   `sensor_value` float NOT NULL,
43.   `sensor_min_limit` float NOT NULL,
44.   `sensor_max_limit` float NOT NULL,
45.   `sensor_min_value` float NOT NULL,
46.   `sensor_max_value` float NOT NULL,
47.   `sensor_datetime` datetime NOT NULL
48. ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
49.
50. CREATE TABLE `users` (
51.   `user_id` int(11) NOT NULL,
52.   `user_google_id` text NOT NULL,
53.   `user_email` varchar(64) NOT NULL,
54.   `user_password` varchar(128) NOT NULL,
55.   `user_fullname` varchar(128) NOT NULL
56. ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
57.
58. ALTER TABLE `logs`
59.   ADD PRIMARY KEY (`log_id`);
60.
61. ALTER TABLE `microcontrollers`
62.   ADD PRIMARY KEY (`microcontroller_id`);
```

```
63.
64. ALTER TABLE `projects`
65.   ADD PRIMARY KEY (`project_id`);
66.
67. ALTER TABLE `sensors`
68.   ADD PRIMARY KEY (`sensor_id`);
69.
70. ALTER TABLE `users`
71.   ADD PRIMARY KEY (`user_id`);
72.
73. ALTER TABLE `logs`
74.   MODIFY `log_id` int(11) NOT NULL AUTO_INCREMENT
    , AUTO_INCREMENT=2139;
75.
76. ALTER TABLE `microcontrollers`
77.   MODIFY `microcontroller_id` int(11) NOT NULL AU
    TO_INCREMENT, AUTO_INCREMENT=48;
78.
79. ALTER TABLE `projects`
80.   MODIFY `project_id` int(11) NOT NULL AUTO_INCRE
    MENT, AUTO_INCREMENT=4;
81.
82. ALTER TABLE `sensors`
83.   MODIFY `sensor_id` int(11) NOT NULL AUTO_INCREM
    ENT, AUTO_INCREMENT=10;
84.
85. ALTER TABLE `users`
86.   MODIFY `user_id` int(11) NOT NULL AUTO_INCREMEN
    T, AUTO_INCREMENT=13;
87. COMMIT;
88.
89. /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_
    SET_CLIENT */;
90. /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER
    _SET_RESULTS */;
91. /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_
    CONNECTION */;
```

Kode 5. 33 Implementasi DDL Database

5.6 *Testing* Sistem

Pengujian pada *hardware* dilakukan sesuai desain yang telah dibuat pada bagian sebelumnya. Sistem dikatakan layak apabila

telah menjalani seluruh rangkaian *test case* dan menghasilkan performa yang baik.

5.6.1 *Testing* Perangkat Keras

Testing pada perangkat keras mengacu pada *test case* yang telah telah dibuat pada tahap sebelumnya. Setiap *test case* memiliki kriteria ekspektasi yang harus dipenuhi.

Tabel 5. 5 Hasil Testing Perangkat Keras

Test case	Skenario	*Ekspektasi*	Hasil
Arduino terhubung dengan WiFi	Menghubungkan mikrokontroller Arduino ke WiFi, ssid dan password benar	Arduino dapat terhubung ke jaringan WiFi dan mendapatkan *IP Address*	OK
Pembacaan sensor pH SEN0161-V2	Sensor SEN0161-V2 terhubung dengan arduino dan dicelupkan ke larutan hidroponik	Data pH dapat terbaca oleh perangkat Arduino dari sensor SEN0161-V2 dan ditampilkan pada LCD Display	OK
Pembacaan sensor *total dissolved solids* SEN0244	Sensor SEN0244 terhubung dengan Arduino dan dicelupkan ke larutan hidroponik	Data *total dissolved solids* dapat terbaca oleh perangkat Arduino dari sensor SEN0244 dan ditampilkan	OK

		pada LCD Display	
Pembacaan sensor temperatur DS18B20	Sensor SEN0244 terhubung dengan Arduino dan dicelupkan ke larutan hidroponik	Data temperatur dapat terbaca oleh perangkat Arduino dari sensor DS18B20 dan ditampilkan pada LCD Display	OK
Pembacaan sensor *infrared*	Memancarkan gelombang sinar inframerah melalui *remote control* untuk diterima oleh sensor pembacaan datanya	Arduino dapat menerima data gelombang yang dipancarkan oleh remote dengan berubahnya tampilan layar LCD	OK
Pengiriman data sensor ke aplikasi web *API* melalui HTTP *request* POST	Mengirimkan data hasil pembacaan sensor SEN0161-V2, SEN0244, DS18B20, dan HC-SR04 ke aplikasi web *API* menggunakan HTTP POST	Data berhasil terkirim, HTTP *response code* 201 CREATED	OK
Penerimaan data nilai batas bawah dan batas atas	Menerima data nilai batas bawah dan batas atas dari aplikasi web *API*	HTTP *response code* 200 dan mendapatkan	OK

dari aplikasi web *API* melalui HTTP *request* GET	menggunakan HTTP GET	data respons dari aplikasi web *API*	
Respon *relay* untuk membuka/menutup katup solenoiddalam menerima *trigger* dari sensor	Relay terhubung dengan Arduino dan masing-masing sensor	Katup solenoid dapat membuka dan menutup sesuai perintah dari Arduino	OK
Tampilkan data ke LCD	Arduino mengirimkan data pH, *total dissolved solids*, temperatur, dan ketinggian air untuk ditampilkan ke LCD	LCD dapat menampilkan data kelistrikan, suhu, dan mode	OK

Kode 5. 34 Hasil Output Pada Layar LCD

5.6.2 *Testing* Perangkat Lunak

Pengujian pada *software* dilakukan sesuai rancangan desain pengujian *software* sebelumnya. Sistem dikatakan layak apabila telah menjalani seluruh rangkaian *test case* dan menghasilkan performa yang baik.

Tabel 5. 6 Hasil Testing Perangkat Lunak

Test Case	Skenario	Ekspektasi	Hasil
Login	Pengguna mengetik *username* dan *password* dengan benar	Pengguna masuk ke halaman dashboard proyek	OK
	Pengguna mengetik *username* dan atau *password* yang salah	Aplikasi web *client redirect* ke halaman login kembali	OK
Membuat akun	Pengguna memasukkan data yang sesuai pada form dengan benar	Terdapat pemberitahuan akun berhasil dibuat dan akan *redirect* ke halaman login	OK
	Pengguna tidak memasukkan data yang sesuai	Aplikasi web *client* akan *redirect* ke halaman *signup* kembali	OK
Melihat dashboard Proyek	Membuka halaman dashboard proyek dan telah login	Aplikasi web *client* menampilkan daftar proyek	OK
	Belum melakukan login	*Redirect* ke halaman login	OK
Mengelola Proyek	Mengklik tombol buat,	*User* dapat membuat,	OK

	ubah, atau hapus proyek pada halaman dashboard proyek	mengubah, atau menghapus proyek	
Melihat dashboard Mikrokontroler	Membuka halaman dashboard mikrokontroler dan telah login	Aplikasi web *client* menampilkan daftar mikrokontroler	OK
	Belum melakukan login	*Redirect* ke halaman login	OK
Mengelola Mikrokontroler	Mengklik tombol buat, ubah, atau hapus mikrokontroler pada halaman dashboard mikrokontroler	*User* dapat membuat, mengubah, atau menghapus mikrokontroler	OK
Melihat dashboard Sensor	Membuka halaman dashboard sensor dan telah login	Aplikasi web *client* menampilkan daftar sensor dan menampilkan data log dalam bentuk grafik	OK
	Belum melakukan login	*Redirect* ke halaman login	OK
Mengelola Sensor	Mengklik tombol buat, ubah, atau hapus sensor pada halaman dashboard sensor	*User* dapat membuat, mengubah, atau menghapus sensor	OK

Tabel 5.4 menjelaskan hasil *test case* perangkat Arduino. Setiap *test case* diuji coba menggunakan skenario yang telah

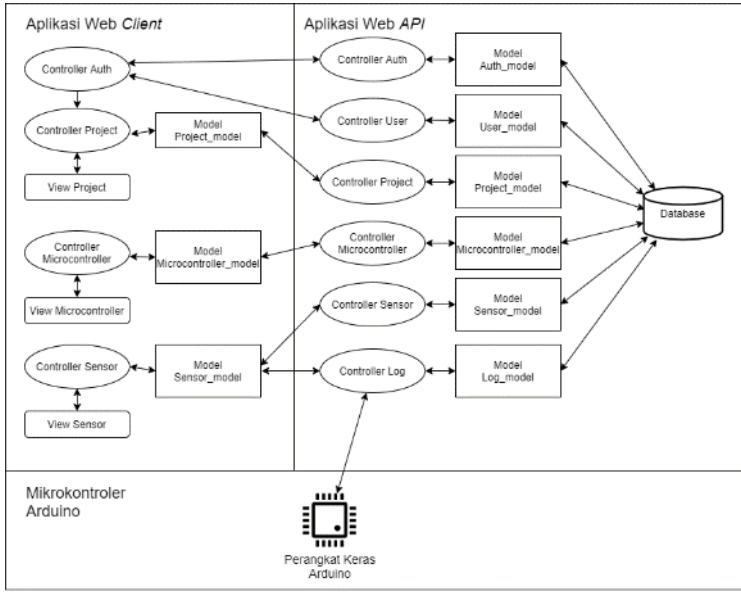
ditetapkan dan diperiksa apakah hasil dari skenario tersebut telah memenuhi ekspektasi yang ditetapkan. Hasil *testing* menunjukkan bahwa semua *test case* yang dijalankan memenuhi ekspektasi.

BAB VI
HASIL DAN PEMBAHASAN

Pada bagian ini akan menjelaskan mengenai hasil dan pembahasan dari pengembangan sistem yang telah dibuat pada pengerjaan tugas akhir ini.

6.1 Arsitektur Sistem

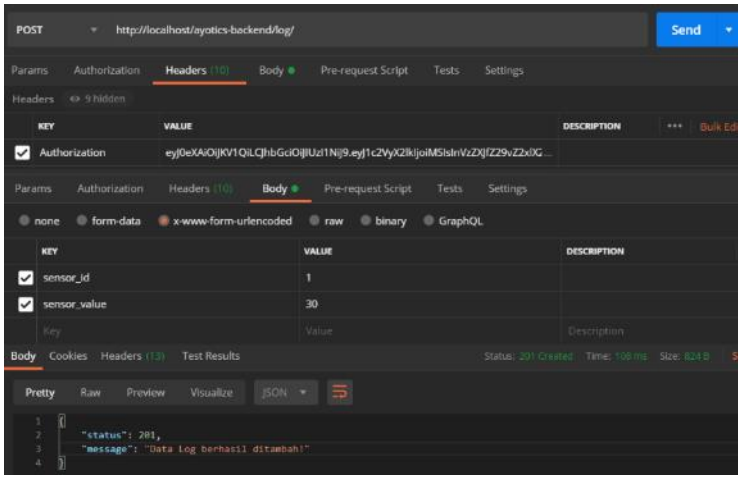
Arsitektur dari sistem yang dikembangkan disesuaikan dengan kebutuhan yang telah disebutkan pada tahap perancangan. Arsitektur dikembangkan dengan menggunakan *design pattern* Model-View-Controller. *Design pattern* tersebut digunakan sesuai dengan *framework* yang digunakan pada aplikasi web yaitu CodeIgniter. Selain MVC, *design pattern microservice* juga diterapkan dalam pembuatan arsitektur sistem. *Design pattern* ini juga memudahkan dari sisi *maintenance* karena masing-masing aplikasi web lebih *modular*.



Gambar 6. 1 Arsitektur Sistem

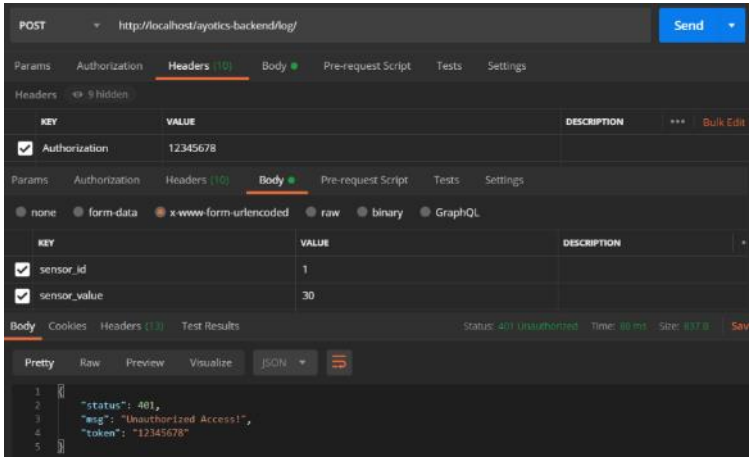
6.2 Komunikasi Aplikasi Web *API* Validasi JWT

Seluruh aplikasi web menggunakan JWT sebagai otorisasi untuk mencegah pengunaan API diluar pengguna. Berikut adalah hasil jika mengirim *request* POST bila menggunakan JWT yang benar.



Gambar 6. 2 *Post* Data menggunakan JWT yang benar

Gambar 6.2 menunjukkan bahwa data log yang dikirimkan ke aplikasi web *API* telah berhasil dilakukan menggunakan *tools* Postman. Status 201 yang berarti *created* (data telah dibuat). Kolom sensor_id menunjukkan id sensor yang mana dan kolom sensor_value menunjukkan nilai dari pembacaan sensor.

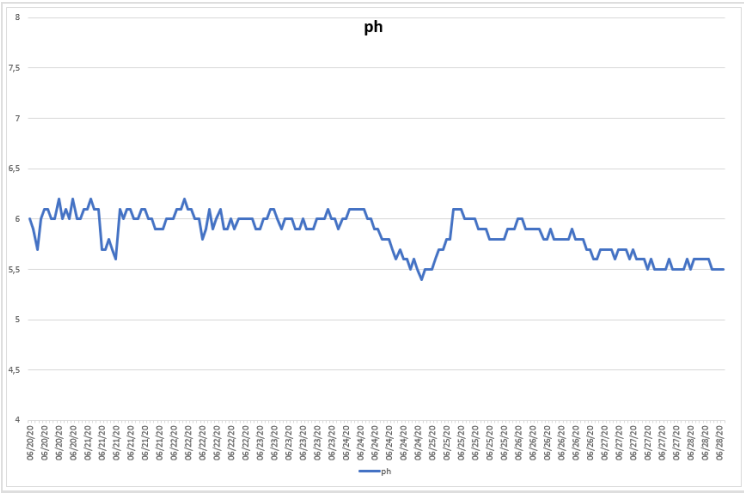


Gambar 6. 3 *Post* JWT menggunakan JWT yang salah

Gambar 6.3 menunjukkan bahwa data log yang dikirimkan ke aplikasi web *API* gagal dilakukan dan sistem memberikan *response* statu 401 yang artinya *unauthorized* (tidak memiliki otorisasi yang benar). Data yang dimasukkan sama dengan data yang dimasukkan pada Gambar 6.2 namun Gambar 6.3 menunjukkan kegagalan dalam mengirimkan *request* dikarenakan *token* yang tidak valid pada *header* "*Authorization*".

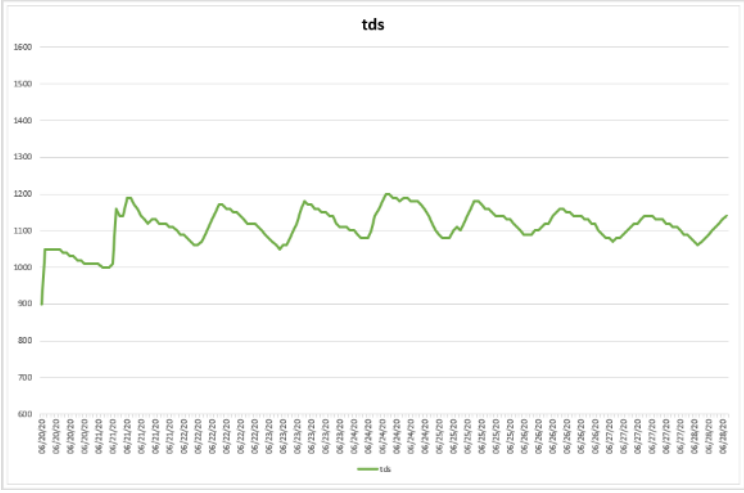
6.3 Hasil Pembacaan Grafik Tanaman Hidroponik

Hasil dari arsitektur sistem, perangkat keras, instalasi hidroponik, dan sistem kontrol telah dikembangkan. Dilakukan proses penanaman tanaman sawi selama 7 hari. Proses yang dilakukan terhadap tanaman hidroponik tersebut melakukan pemantauan kualitas seperti nilai pH, *total dissolved solids*, temperatur, dan ketinggian air. Berikut adalah penjelasan masing-masing grafik pada Gambar 6.4 hingga 6.6.



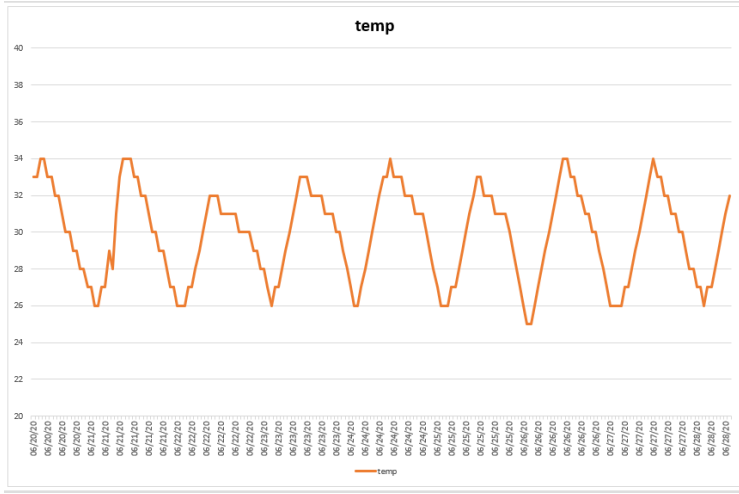
Gambar 6. 4 Grafik pH selama 7 hari

Kondisi awal instalasi hidroponik ini menggunakan air PDAM dengan pH 7,7. Setelah penambahan nutrisi A dan nutrisi B hingga 1080ppm, nilai pH turun menjadi 7,1. Kemudian untuk melakukan pengendalian hingga mendekati nilai batas bawah dan batas atas ditambahkan asam hingga nilai pH menjadi 6,5.



Gambar 6. 5 Grafik *Total Dissolveds Solids* selama 7 hari

Kondisi awal instalasi hidroponik ini menggunakan air PDAM dengan *total dissolved solids* sebesar 150ppm. Kemudian ditingkatkan secara manual menggunakan nutrisi A dan nutrisi B hingga sekitar 900ppm seperti yang ditunjukkan pada Gambar 6.5. Nilai batas bawah *total dissolved solids* yaitu 1000ppm, pada awal grafik mengalami peningkatan dikarenakan katup solenoida yang berisi larutan nutrisi A dan B bekerja dan meningkatkan *total dissolved solids* hingga batas bawah dan batas atas. Adanya kenaikan dan penurunan pembacaan *total dissolved solids* dikarenakan adanya perbedaan temperatur.



Gambar 6. 6 Grafik Temperatur selama 7 hari

Grafik Gambar 6.6 menunjukkan bahwa adanya kenaikan dan penurunan temperatur. Temperatur tertinggi yang dibaca oleh sensor 34°C sekitar pukul 13.00 hingga 14.00. Temperatur terendah yang dibaca oleh sensor yaitu 25°C.

6.4 Hasil Pertumbuhan Tanaman

Sistem instalasi hidroponik yang digunakan sebanyak 2 buah. Yaitu sistem hidroponik tanaman otomatis dan manual. Pada

awal tanaman masih dalam satu instalasi, kemudian pada proses budidaya tanaman dipisah berdasarkan otomatis dan manual. Gambar 6.7 menunjukkan kondisi tanaman sebagai berikut.

Gambar 6. 7 Tanaman Sebelum Proses Budidaya

Tanaman yang dibudidayakan berusia sekitar 3 minggu. Tanaman yang digunakan adalah sayuran sawi pak choi, *Brassica rapa*. Setelah itu tanaman dipisah dan diberi label agar dapat dibandingkan tingginya sebelum dan sesudah proses budidaya. Berikut adalah kondisi tanaman otomatis yang sudah dilakukan proses budidaya pada Gambar 6.8

Gambar 6. 8 Tanaman Otomatis Sesudah Budidaya

Tanaman otomatis menggunakan sistem yang telah didesain dan diimplementasikan sebelumnya. Perbedaan tanaman manual yaitu proses pemantauan dan pengendalian dilakukan secara manual, tidak menggunakan sistem. Berikut adalah kondisi tanaman manual sesudah dilakukan proses budidaya.

Gambar 6. 9 Tanaman Manual Sesudah Budidaya

Pertumbuhan tanaman yang diukur untuk dilakukan perbandingan sistem otomatis dan sistem manual adalah tinggi masing-masing tanaman. Pada Gambar 6.10 ditunjukkan dua tanaman berbeda. Tanaman pada kiri gambar merupakan tanaman yang dibudidayakan dengan sistem otomatis, tanaman pada kanan gambar merupakan tanaman yang dibudidayakan dengan sistem manual.

Gambar 6. 10 Perbandingan Pertumbuhan Tanaman

Tanaman yang dibudidayakan pada Tabel 6.1 merupakan hasil budidaya tanaman secara otomatis yang memiliki rata-rata pertumbuhan tinggi tanaman sebesar 2,3 cm.

Tabel 6. 1 Tabel Tinggi Tanaman menggunakan Sistem Otomatis

No.	Sebelum (cm)	Sesudah (cm)	Selisih (cm)
1	13,4	15,1	1,7
2	12,1	14,0	1,9
3	15,0	17,5	2,5
4	12,2	15,0	2,8
5	14,1	16,0	1,9
6	11,6	14,0	2,4
7	12,4	13,5	1,1
8	16,0	18,1	2,1

9	15,8	17,9	2,1
10	17,0	21,0	4,0
11	17,0	19,0	2,0
12	14,5	16,5	2,0
13	14,2	16,9	2,7
14	15,8	17,8	2,0
15	15,0	18,2	3,2
		Rata-rata	2,3 cm

Tanaman yang dibudidayakan pada Tabel 6.2 merupakan hasil budidaya tanaman secara manual yang memiliki rata-rata pertumbuhan tinggi tanaman sebesar 1,2 cm.

Tabel 6. 2 Tabel Tinggi Tanaman menggunakan Sistem Manual

No.	Sebelum (cm)	Sesudah (cm)	Selisih (cm)
1	18,0	19,0	1,0
2	12,2	13,4	1,2
3	15,0	16,5	1,5
4	14,2	15,5	1,3
5	15,5	16,8	1,3
6	15,0	16,5	1,5
7	16,1	17,2	1,1
8	17,0	18,1	1,1
9	15,2	16,4	1,2
10	13,3	14,5	1,2
11	12,5	13,7	1,2
12	12,0	13,0	1,0
13	15,4	16,5	1,1
14	14,0	15,0	1,0
15	15,5	16,9	1,4
		Rata-rata	1,2 cm

Dari data di atas terlihat bahwa kondisi tanaman yang dilakukan proses budidaya sistem otomatis sebesar 2,3 cm dibandingkan secara manual sebesar 1,2 cm.

BAB VII
KESIMPULAN DAN SARAN

Pada bagian ini akan menjelaskan mengenai kesimpulan dari pengerjaan tugas akhir ini dan saran untuk pengembangan sistem selanjutnya yang relevan.

7.1 Kesimpulan

Berikut adalah kesimpulan dari pengembangan sistem yang telah dilakukan pada pengerjaan tugas akhir ini.

1. Perangkat keras Arduino dapat digunakan sebagai *processing unit* dalam pembuatan *smart farming* menggunakan metode hidroponik.
2. Perangkat keras Arduino yang dikembangkan dapat mengetahui informasi pH, *total dissolved solids*, temperatur, dan ketinggian air mengenai nilai larutan hidroponik menggunakan sensor SEN0161-V2, SEN0244, DS18B20, dan HC-SR04.
3. Aplikasi web *API* dapat melakukan pengelolaan data pada aplikasi web *client* dan Arduino untuk menyimpan informasi ke dalam *database*.
4. Tanaman yang dibudidayakan menggunakan sistem *smart farming* otomatis mengalami pertumbuhan lebih tinggi 1,1 cm dibandingkan tanaman yang dibudidayakan secara manual.

7.2 Saran

Berikut adalah beberapa saran untuk pengembangan lebih lanjut untuk menyempurnakan pengembangan sistem.

1. Menggunakan protokol MQTT dalam hal melakukan komunikasi antara perangkat keras dengan perangkat lunak untuk menampilkan data pada aplikasi web.

2. Pengembangan aplikasi *mobile* untuk melakukan pemantauan sistem hidroponik dan pengendalian pada perangkat genggam.
3. Sangat banyak faktor yang mempengaruhi tumbuh dan kembangnya tanaman, dibutuhkan lebih banyak sensor dan pengondisian dalam proses budidaya tanaman.

DAFTAR PUSTAKA

[1] T. Wahyuni, "BPS Sebut Luas Lahan Pertanian Kian Menurun," *CNN Indonesia*, 2018. [Online]. Available: https://www.cnnindonesia.com/ekonomi/20181025153705-92-341433/bps-sebut-luas-lahan-pertanian-kian-menurun.

[2] B. A. Sheikh, "Hydroponics: Key to sustain agriculture in water stressed and urban environment," *Pakistan J. Agric. Agric. Eng. Vet. Sci.*, vol. 22, no. 2, pp. 53–57, 2006.

[3] N. Zhang, M. Wang, and N. Wang, "Precision agriculture - A worldwide overview," in *Computers and Electronics in Agriculture*, 2002, vol. 36, no. 2–3, pp. 113–132.

[4] R. Khosla, "Precision agriculture : challenges and opportunities in a flat world," *Soil Solut. a Chang. World. 19th World Congr. Soil Sci. 2010.*, no. August, pp. 26–28, 2010.

[5] K. Kularbphettong, U. Ampant, and N. Kongrodj, "An Automated Hydroponics System Based on Mobile Application," *Int. J. Inf. Educ. Technol.*, vol. 9, no. 8, pp. 548–552, 2019.

[6] U. Nurhasan, A. Prasetyo, G. Lazuardi, E. Rohadi, and H. Pradibta, "Implementation IoT in System Monitoring Hydroponic Plant Water Circulation and Control," *Int. J. Eng. Technol.*, vol. 7, no. 4.44, p. 122, 2018.

[7] H. A. Jabbar, "Rancang Bangun Sistem Pendeteksi Kesuburan Tanah menggunakan Arduino Dengan Sensor Electrical Conductivity, Temperatur, Kelembapan Tanah, dan pH," 2019.

[8] Y. A. Badamasi, "The working principle of an Arduino," *Proc. 11th Int. Conf. Electron. Comput. Comput. ICECCO 2014*, 2014.

[9] J. D. dos Santos *et al.*, “Development of a vinasse nutritive solution for hydroponics,” *J. Environ. Manage.*, vol. 114, pp. 8–12, 2013.

[10] R. Tandil, J. Yapson, W. Atmadja, S. Liawatimena, and R. Susanto, “Hydroponic nutrient mixing system based on STM32,” *IOP Conf. Ser. Earth Environ. Sci.*, vol. 195, no. 1, 2018.

[11] World Health Organization, *Guidelines for Drinking-Water Quality: Fourth Edition Incorporating the First Addendum*. 2014.

[12] J. Kovalevsky and T. J. Quinn, “The international system of units (SI),” *Comptes Rendus Phys.*, vol. 5, no. 8 SPEC.ISS., pp. 799–811, 2004.

[13] R. G. Bates, “The Modern Meaning of pH,” *C R C Crit. Rev. Anal. Chem.*, vol. 10, no. 3, pp. 247–278, 1981.

[14] B. W. Mangum *et al.*, “The Kelvin and temperature measurements,” *J. Res. Natl. Inst. Stand. Technol.*, vol. 106, no. 1, pp. 105–149, 2001.

[15] CodeIgniter, “CodeIgniter at a Glance,” *British Columbia Institute of Technology*, 2019. [Online]. Available: https://codeigniter.com/user_guide/overview/at_a_glance.html.

[16] CodeIgniter, “Application Flow Chart,” *British Columbia Institute of Technology*, 2019. [Online]. Available: https://codeigniter.com/user_guide/overview/appflow.html

[17] "MySQL 8.0 Reference Manual: What is MySQL?," Oracle, [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html [Diakses 11 November 2019]

[18] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, Irvine: University of California, 2000.

[19] R. T. Fielding and J. F. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," *Internet Eng. Task Force*, pp. 1–101, 2014.

[20] "DHT22 - SparkFun Electronics." [Online]. Available: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf.

[21] "DS18B20 Programmable Resolution 1-Wire Digital Thermometer." [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf.

[22] "Gravity: Analog TDS Sensor / Meter For Arduino SKU: SEN0244." [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0244_Web.pdf.

[23] "PH meter(SKU: SEN0161)." [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0161_SEN0169_Web.pdf.

[24] "Ultrasonic Ranging Module HC - SR04." [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf.

[25] Q. Liu, H. Bo, and B. Qin, "Experimental study and numerical analysis on electromagnetic force of direct action solenoid valve," *Nucl. Eng. Des.*, vol. 240, no. 12, pp. 4031–4036, 2010.

[26] N. Agrawal and S. Singhal, "Smart Drip Irrigation System using Raspberry Pi and Arduino," *Int. Conf. Comput. Commun. Autom. ICCCA 2015*, pp. 928–932, 2015.

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis bernama Muhammad Athma Farhan, lahir di Surabaya pada tanggal 9 Oktober 1998, merupakana anak pertama dari tiga bersaudara. Penulis telah menempuh jenjang pendiikan formal di beberapa sekolah, yaitu: SD Al Hikmah Surabaya (2004-2010), SMP Al Hikmah Surabaya (2010-2013), dan SMA Al Hikmah Surabaya (2013-2016). Penulis melanjutkan pendidikan sarjana di Departemen Sistem Informasi Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC) Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2016 yang terdaftar sebagai mahasiswa dengan NRP 05211640000051.

Selama menjadi mahasiswa, penulis aktif mengikuti kemahasiswaan Himpunan Mahasiswa Sistem Informasi ITS pada tahun kepengurusan 2017-2018 serta acara kepanitiaan lain seperti Information Systems Expo 2017 dan 2018.

Pada Oktober hingga Desember 2019, penulis berpengalaman kerja lepas membangun aplikasi web e-kinerja Kabupaten Berau, Provinsi Kalimantan Timur sebagai Back End Developer. Penulis dapat dihubungi melalui email farhan.athma@gmail.com