TUGAS AKHIR – IS184853

RANCANG BANGUN SISTEM SMART POWER DENGAN KONSEP IOT MENGGUNAKAN ARDUINO BERBASIS WEB (STUDI KASUS: DEPARTEMEN SISTEM INFORMASI ITS)

DESIGN AND DEVELOPMENT OF SMART POWER SYSTEM WITH IOT CONCEPT USING WEB-BASED ARDUINO (CASE STUDY: INFORMATION SYSTEMS ITS DEPARTMENT)

NAUFAL IHZA REVANDHIKA
NRP. 05211640000034

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 SISTEM SMART POWER DENGAN KONSEP IOT MENGGUNAKAN ARDUINO BERBASIS WEB (STUDI KASUS: DEPARTEMEN SISTEM INFORMASI ITS)

NAUFAL IHZA REVANDHIKA
NRP. 05211640000034

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 OF SMART POWER SYSTEM WITH IOT CONCEPT USING WEB-BASED ARDUINO (CASE STUDY: INFORMATION SYSTEMS ITS DEPARTMENT)

NAUFAL IHZA REVANDHIKA
NRP. 05211540000129

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

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

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM SMART POWER DENGAN KONSEP IOT MENGGUNAKAN ARDUINO BERBASIS WEB (STUDI KASUS: DEPARTEMEN SISTEM INFORMASI ITS)

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

Naufal Ihza Revandhika
05211640000034

Surabaya, 14 Agustus 2020

Kepala Departemen Sistem Informasi

Dr. Mudjahidin, ST., MT.
NIP. 197010102003121001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

RANCANG BANGUN SISTEM SMART POWER DENGAN KONSEP IOT MENGGUNAKAN ARDUINO BERBASIS WEB (STUDI KASUS: DEPARTEMEN SISTEM INFORMASI ITS)

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 :

NAUFAL IHZA REVANDHIKA
NRP. 05211640000034

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 SISTEM SMART POWER DENGAN KONSEP IOT MENGGUNAKAN ARDUINO BERBASIS WEB (STUDI KASUS: DEPARTEMEN SISTEM INFORMASI ITS)

Nama	: Naufal Ihza Revandhika
NRP	: 05211640000034
Departemen	: Sistem Informasi ITS
Pembimbing	: Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

ABSTRAK

Banyaknya inovasi di bidang teknologi yang terus berkembang memang dapat memudahkan manusia dalam menjalankan aktivitas sehari-harinya. Berbagai jenis perangkat elektronik yang digunakan tentu membutuhkan energi listrik sebagai sumber tenaganya. Penggunaan perangkat elektronik yang begitu besar ini akan membawa dampak pada penggunaan listrik yang terus meningkat seiring berjalannya waktu. Departemen Sistem Informasi ITS merupakan salah satu departemen di dalam Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC) yang menggunakan listrik sebagai kebutuhan utama dalam menunjang kegiatan riset dan perkuliahan. Oleh sebab besarnya kebutuhan energi listrik yang digunakan, maka perlu diterapkan sebuah sistem yang dapat mengontrol penggunaan listrik sesuai dengan kebutuhan menggunakan konsep Internet of Things (IoT). Teknologi Internet of Things (IoT) dapat diterapkan dengan menggunakan mikrokontroller yang diintegrasikan dengan sensor sebagai pendeteksi inderanya yang akan dihubungkan pada modul elektronik dan internet sebagai media komunikasi dan pengiriman data. Pada penelitian tugas akhir ini, sistem penghematan energi listrik yang dihasilkan dapat bekerja dengan menempatkan sensor Passive Infrared Sensor (PIR) di setiap ruangan untuk memantau keberadaan aktivitas manusia.

Selain itu, pada sistem ini juga diaplikasikan sensor PZEM004T-V3 yang digunakan sebagai pendeteksi arus, tegangan, frekuensi, dan besaran penggunaan energi listrik pada tipe arus bolak-balik (AC). Seluruh data tersebut dikalkulasikan cukup dalam 1 sensor disetiap kelas yang selanjutnya data – data yang didapatkan dari sensor akan dikirimkan ke Arduino *untuk dilakukan pemrosesan lebih lanjut. Untuk menghubungkan antara* Arduino *dengan web, data yang telah masuk ke* Arduino *akan ditransfer ke web dengan menggunakan HTTP Request. Keuntungan dengan penerapan sistem ini adalah selain mendukung upaya penghematan energi dengan mengontrol penggunaan energi listrik secara otomatis juga dapat membantu petugas yang terkait dalam melihat trend penggunaan listrik dan pengontrolan melalui web.*

Kata Kunci : **Arduino*, HTTP Request, Internet of Things, Penghematan energi listrik, Sensor***

DESIGN AND DEVELOPMENT OF SMART POWER SYSTEM WITH IOT CONCEPT USING WEB-BASED ARDUINO (CASE STUDY: INFORMATION SYSTEMS ITS DEPARTMENT)

Name	: Naufal Ihza Revandhika
NRP	: 05211640000034
Department	: Information Systems ITS
Supervisor	: Dr.Eng. Febriliyan Samopa, S.Kom., M.Kom.

ABSTRACT

The number of innovations in the field of technology that continues to develop can indeed facilitate humans in carrying out their daily activities. Various types of electronic devices used certainly require electrical energy as a source of energy. The use of such large electronic devices will have an impact on electricity usage which continues to increase over time. Information System Department is one of the departments in ITS within the Faculty of Electrical and Intelligent Information Technology (FTEIC) that uses electricity as a primary need to support research and lecture activities. Because of the large demand for electrical energy used, it is necessary to implement a system that can control electricity usage in accordance with the needs of using the concept of the Internet of Things (IoT). Internet of Things (IoT) technology can be applied by using a microcontroller that is integrated with the sensor as a sensory detector that will be connected to electronic and internet modules as a medium of communication and data transmission. In this final project research, the resulting electrical energy saving system can work by placing a Passive Infrared Sensor (PIR) sensor in every room to monitor the existence of human activities. In addition, this system also applies the PZEM004T-V3 sensor which is used as a detector of current, voltage, frequency, and the amount of electricity used in alternating

current (AC) types. All data is calculated in one sensor in each class, then the data obtained from the sensor will be sent to Arduino *for further processing. To connect between* Arduino *and the web, data that has entered* Arduino *will be transferred to the web using an HTTP Request. The advantage of implementing this system is that in addition to supporting energy saving efforts by automatically controlling the use of electrical energy, it can also help officials involved in seeing trends in electricity usage and control through the web.*

Kata Kunci : Arduino, HTTP Request, Internet of Things, Electrical Energy Saving, Sensor

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: Naufal Ihza Revandhika
NRP	: 05211640000034
Tempat/Tanggal lahir	: Malang/26 Juli 1998
Fakultas/Departemen	: Fakultas Teknologi Elektro dan Informatika Cerdas/Sistem Informasi
Nomor Telp/Hp/email	: Naufalihza20@gmail.com

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

RANCANG BANGUN SISTEM SMART POWER DENGAN KONSEP IOT MENGGUNAKAN ARDUINO BERBASIS WEB (STUDI KASUS: DEPARTEMEN SISTEM INFORMASI ITS)

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, 6 Agustus 2020

METERAI TEMPEL
4BD01AHF530791957
6000
ENAM RIBU RUPIAH

Naufal Ihza Revandhika
NRP. 05211640000034

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, Tuhan Semesta Alam atas segala karunia dan rahmat yang telah dilimpahkanNya 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.

Keberhasilan penulis dalam menyelesaikan tugas akhir ini tidak terlepas dari bantuan berbagai pihak. Terima kasih penulis sampaikan dengan ketulusan dan kerendahan hati 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 penulis, Ibu Elly Istiyarini dan Ayah Drs. Rory Mashuri, kakak Fahniar Firmansyah dan adik Dhavin Ryansyah Rivaldi, yang senantiasa memberikan doa, motivasi, dan semangat sehingga penulis dapat menjalani masa perkuliahan hingga menyelesaikan tugas akhir 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, Bapak Dr. Bambang Setiawan, S.Kom., M.T. selaku kepala lab IKTI, serta seluruh dosen pengajar beserta staf dan karyawan Departemen Sistem Informasi ITS yang telah membantu dan memberikan pelayanan selama penulis menjalani perkuliahan.

3. Bapak Edwin Reksa Komara S.Kom M.T selaku 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 penulis dengan sabar, memberikan arahan, 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 ilmu, kritik serta saran yang sangat penting dalam proses penyempurnaan tugas akhir ini.
6. Teman-teman Sistem Informasi angkatan 2016 (Artemis) yang senantiasa memberikan *support* dan motivasi bagi penulis selama perkuliahan hingga dapat menyelesaikan tugas akhir ini.
7. Seluruh pihak-pihak lainnya yang tidak dapat disebutkan satu per satu yang telah membantu penulis secara langsung dan tidak langsung selama perkuliahan hingga dapat menyelesaikan tugas akhir ini.

Penulis menyadari masih terdapat banyak kekurangan dalam laporan tugas akhir ini dan masih jauh dari kata sempurna, sehingga penulis menerima adanya kritik maupun saran yang membangun untuk perbaikan di masa mendatang. Semoga buku tugas akhir ini dapat memberikan manfaat baik bagi penulis, pembaca, dan semua pihak.

Surabaya, 26 Juni 2020

Penulis

Naufal Ihza Revandhika

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, inovasi di bidang teknologi terus tumbuh begitu pesat. Tumbuhnya berbagai inovasi di bidang teknologi ini telah membawa manfaat yang begitu besar bagi manusia dalam menunjang aktivitasnya di setiap hari. Disisi lain, pemanfaatan teknologi yang begitu masif oleh manusia ini juga berdampak pada penggunaan listrik yang terus meningkat seiring berjalannya waktu. Energi listrik merupakan salah satu komponen yang memiliki peran penting dalam kehidupan di masa kini hingga masa depan. Kementrian Energi dan Sumber Daya Mineral (ESDM) mencatat bahwa konsumsi listrik rata-rata di Indonesia pada tahun 2018 sebesar 1064 kilo Watt hours (kWh) per kapitanya. Capaian ini meningkat 5,14 persen dari penggunaan listrik per kapita di tahun 2017 sebesar 1012 kilo Watt hours (kWh) [1]. Diperkirakan kebutuhan listrik di Indonesia pada masa depan akan terus meningkat yang akan dipengaruhi oleh berbagai faktor, seperti peningkatan pembangunan infrastruktur, perkembangan jumlah penduduk, serta perkembangan teknologi. Energi listrik saat ini telah dimanfaatkan pada berbagai sektor, seperti pada sektor perindustrian, rumah tangga, penerangan umum, riset dan pendidikan, dan lain sebagainya. Penggunaan listrik yang begitu besar ini tidak diiringi dengan upaya penghematan energi yang dilakukan, sehingga akan menimbulkan masalah baru dengan kebiasaan penggunaan yang kurang efektif dan terkesan boros.

Pada bidang pendidikan dan riset contohnya, Departemen Sistem Informasi ITS merupakan salah satu departemen di dalam Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC) yang menggunakan listrik sebagai kebutuhan utama dalam menunjang kegiatan riset dan perkuliahan. Penggunaan perangkat elektronik di dalam kelas seperti pendingin ruangan (AC), LCD Proyektor, lampu, dan laptop/komputer merupakan kebutuhan yang paling mendasar dalam menunjang kegiatan sehari-hari. Dalam penerapan kesehariannya, seringkali perangkat elektronik dibiarkan menyala tanpa adanya aktivitas yang dilakukan di dalam kelas khususnya di luar jam perkuliahan seperti pada jam pergantian kelas sehingga hal ini menimbulkan penggunaan listrik yang terbuang sia-sia. Terdapat banyak teknologi yang dapat dimanfaatkan untuk menanggulangi penggunaan listrik yang berlebih, salah satunya adalah dengan menerapkan konsep Internet of Things (IoT). Internet of Things (IoT) dapat diterapkan dengan menggunakan mikrokontroller yang diintegrasikan dengan sensor sebagai pendeteksi inderanya yang akan dihubungkan pada modul elektronik dan internet sebagai media komunikasi dan pengiriman data.

Pada penelitian ini, sistem penghematan energi listrik yang diajukan dapat bekerja dengan menempatkan sensor proximity di setiap ruangan untuk memantau keberadaan aktivitas manusia. Selain itu, pada sistem ini juga diaplikasikan sensor pendeteksi arus listrik yang data keluarannya akan dikalkulasikan untuk mengetahui besaran daya listrik yang digunakan. Data-data yang diterima dari sensor akan diterima oleh mikrokontroller untuk dilakukan proses decision making. Apabila di dalam kelas tidak ditemukan aktivitas yang dilakukan dalam *range* waktu yang telah ditentukan maka sistem akan memutus aliran listrik secara otomatis dengan mengirimkan sinyal ke *relay*. Untuk menunjang kegiatan monitoring penggunaan energi listrik, sistem ini menggunakan aplikasi web sebagai media yang dapat digunakan untuk memantau penggunaan listrik secara periodik. Aplikasi web ini akan diintegrasikan dengan database untuk keperluan

pencatatan historis data, sedangkan untuk menghubungkan antara Arduino dengan web, data yang telah masuk ke Arduino akan ditransfer ke web dengan menggunakan HTTP Request. Keuntungan dengan penerapan sistem ini adalah selain mendukung upaya penghematan energi juga dapat membantu user yang terkait dalam melihat trend penggunaan listrik melalui web. Penerapan sistem penghemat energi semacam ini juga mendukung program pemerintah yang tercantum dalam PP No.70/2009 tentang konservasi energi [2].

Dengan demikian, pengerjaan tugas akhir ini diharapkan dapat menghasilkan rancangan berupa alat yang dapat menjadi aspek pendukung dalam upaya penghematan energi di lingkungan Departemen Sistem Informasi ITS serta diharapkan dapat memudahkan petugas yang terkait dalam memantau penggunaan listrik dengan sistem yang terintegrasi.

1.2 Rumusan Masalah

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

1. Bagaimana membuat sistem smart power yang dapat mengatur aliran listrik berdasarkan aktivitas di dalam ruangan menggunakan sensor proximity?
2. Bagaimana menggunakan current sensor / sensor arus untuk mengukur penggunaan daya listrik?
3. Bagaimana menerima hasil pembacaan data dari sensor yang diintegrasikan dengan mikrokontroller Arduino?
4. Bagaimana membuat aplikasi web sebagai media untuk mengontrol sistem kelistrikan dan menampilkan informasi penggunaan secara periodik ?
5. Bagaimana menghubungkan Arduino ke web untuk komunikasi dan transfer data menggunakan HTTP Request?

1.3 Batasan Masalah

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

1. Rancangan sistem yang dibuat berdasarkan studi kasus berupa prototipe maket sebanyak 6 ruangan yang telah disesuaikan dengan kebutuhan penelitian.
2. Sistem kelistrikan yang dapat diatur adalah menggunakan arus alternating current (AC) dengan tegangan sebesar 220V.
3. Mikrokontroler Arduino digunakan sebagai perangkat untuk mengontrol I/O.
4. Antarmuka untuk mengontrol sistem kelistrikan dan menampilkan informasi penggunaan daya listrik dibuat dalam bentuk website menggunakan bahasa pemrograman PHP.

1.4 Tujuan

Berdasarkan latar belakang dan rumusan masalah yang telah disebutkan sebelumnya, adapun tujuan dari penelitian tugas akhir ini adalah mengembangkan sebuah prototipe sistem smart power menggunakan konsep Internet of Things yang mampu membantu dalam penghematan penggunaan energi listrik di suatu ruangan berdasarkan keberadaan aktivitas, serta menyimpan dan menampilkan informasi penggunaan listrik secara periodik melalui aplikasi web.

1.5 Manfaat

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

1. Menjadi panduan dasar bagi Departemen Sistem Informasi ITS maupun pihak lain dalam melakukan pengembangan sistem penghematan penggunaan energi listrik menggunakan konsep Internet of Things.
2. Membantu pihak yang terkait dalam melakukan monitoring penggunaan energi listrik berdasarkan pencatatan data secara historis.

3. Menjadi referensi bagi kalangan praktisi dalam pengembangan penelitian di bidang pemanfaatan Internet of Things untuk menunjang aktivitas sehari-hari.

1.6 Relevansi

Tugas akhir ini disusun untuk memenuhi salah satu syarat kelulusan sebagai Sarjana Komputer di Departemen Sistem Informasi Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC) Institut Teknologi Sepuluh Nopember (ITS). Tugas akhir ini dikerjakan sesuai dengan penerapan mata kuliah dari laboratorium Infrastruktur dan Keamanan Teknologi Informasi (IKTI) Departemen Sistem Informasi FTIK ITS, yaitu Internet Untuk Segala dan Rancang Bangun Perangkat Lunak

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Pada bagian ini akan menjelaskan mengenai studi literatur terhadap penelitian sebelumnya dan dasar teori yang dijadikan acuan atau landasan dalam pengerjaan tugas akhir ini.

2.1 Studi Literatur

Dalam proses pengerjaan dari pengembangan perangkat lunak pada tugas akhir ini, dilakukan pencarian penelitian-penelitian yang sudah dilakukan sebagai referensi dalam pengerjaan dari pengembangan perangkat lunak pada tugas akhir ini, yang ditunjukkan dalam tabel 2.1.

Tabel 2.1 Studi Literatur

1) Development of Web-Based Real-time Energy Monitoring System for Campus University
Penulis; Tahun NY Dahlan, AAM Aris, MA Saidin, MJM Nadzeri, MNM Nawi, WF Abas, AF Abidin, H Mohammad, Z Zakaria, dan P Arshad ; 2016 [3]
Pembahasan Pembahasan Penelitian ini menyajikan pengembangan sistem real-time energy monitoring menggunakan Arduino sebagai papan mikrokontroller, sensor arus (CT Sensor), modul *Wi-Fi* ESP8266, dan Web Server untuk merekam dan menampilkan penggunaan energi listrik secara real-time. Data yang diperoleh dari sensor arus kemudian ditransfer menggunakan modul *Wi-Fi* ESP8266 untuk diolah dalam board Arduino agar dapat ditampilkan hasilnya pada laman web. Sistem ini memungkinkan untuk menangkap data penggunaan energi listrik pada masing-masing perangkat listrik yang sudah terpasang di dalam kelas. Contohnya adalah lampu, power plug, pendingin ruangan (AC), LCD Projector, dan kipas angin. Hasil dari pemrosesan data yang ditampilkan dalam bentuk grafik pada laman web berupa konsumsi daya dalam

setiap waktu, biaya energi yang digunakan, indeks energy, serta emisi gas CO2
Keterkaitan Penelitian ini menyajikan pengembangan sistem real-time energy monitoring menggunakan Arduino sebagai papan mikrokontroller, sensor arus (CT Sensor), modul *Wi-Fi* ESP8266, dan Web Server untuk merekam dan menampilkan penggunaan energi listrik secara real-time. Data yang diperoleh dari sensor arus kemudian ditransfer menggunakan modul *Wi-Fi* ESP8266 untuk diolah dalam board Arduino agar dapat ditampilkan hasilnya pada laman web. Sistem ini memungkinkan untuk menangkap data penggunaan energi listrik pada masing-masing perangkat listrik yang sudah terpasang di dalam kelas. Contohnya adalah lampu, power plug, pendingin ruangan (AC), LCD Projector, dan kipas angin. Hasil dari pemrosesan data yang ditampilkan dalam bentuk grafik pada laman web berupa konsumsi daya dalam setiap waktu, biaya energi yang digunakan, indeks energy, serta emisi gas CO2
Perbedaan dengan penelitian Tugas Akhir ini Pada paper ini dijelaskan bahwa sistem yang diimplementasikan lebih fokus pada pembuatan sistem informasi monitoring penggunaan energi listrik berbasis web dengan menggunakan sensor untuk menerima data dan mikrokontroller Arduino untuk mengolah data menjadi informasi. Tidak ada kemampuan sistem untuk melakukan decision making dengan memutus arus listrik berdasarkan aktivitas yang ada di dalam ruangan. Pada sistem yang dibuat pada penelitian Tugas Akhir ini selain dapat menampilkan informasi penggunaan energi listrik melalui web, juga dapat memutus arus listrik berdasarkan ada atau tidaknya aktivitas di dalam ruangan dengan mengintegrasikan mikrokontroller Arduino, sensor pir dan *relay*.
2) Energy Saving System using a PIR Sensor for Classroom Monitoring
Penulis; Tahun

Cynthia Twumasi, K. A. Dotche*, W. Banuenumah, dan F. Sekyere ; 2017 [4]
Penelitian ini dilakukan bertujuan untuk membuat rancangan sistem penghemat penggunaan energi menggunakan Passive Infrared Radio Sensor (PIR). PIR Sensor adalah sebuah perangkat yang dapat mendeteksi gerakan (motion) dengan menghitung perubahan nilai heat level pada infrared di sekeliling sensor tersebut. Ketika gerakan terdeteksi oleh PIR Sensor, maka akan mengirimkan sinyal output “high” pada pin yang akan diproses oleh mikrokontroller. Logic signal ini nantinya akan dibaca oleh mikrokontroller atau juga dapat diintegrasikan dengan transistor sebagai trigger switch pada perangkat tegangan tinggi yang terpasang di dalam ruangan kelas. Sistem akan memutus aliran listrik secara otomatis apabila nilai pada PIR sensor “low”, dengan kata lain tidak terdeteksinya gerakan atau motion disekitar ruang kelas sehingga diindikasikan tidak adanya kegiatan di dalam ruang kelas. Selain menggunakan sensor PIR, pada penelitian ini juga menggunakan sensor suhu untuk men-trigger ¬kipas menyala sesuai dengan parameter suhu yang sudah ditentukan. Adapun mikrokontroller yang digunakan dalam penelitian ini adalah board Arduino.
Keterkaitan Penelitian tugas akhir yang dilakukan berkaitan dengan penggunaan PIR Sensor sebagai piranti untuk mendeteksi keberadaan aktifitas yang terjadi di dalam kelas yang diintegrasikan dengan mikrokontroller Arduino. Pendeteksian aktifitas yang terjadi di dalam kelas ditentukan dari nilai heat level yang didapatkan dari infrared / PIR Sensor. Selanjutnya nilai ini yang akan digunakan sebagai logika untuk memutus / menghubungkan arus listrik.
Perbedaan dengan penelitian Tugas Akhir ini Pada paper ini sistem yang dibuat tidak ada fitur / dashboard untuk mengetahui penggunaan energi listrik melalui web. Sehingga user tidak dapat mengetahui record historis besaran energi listrik yang digunakan. Sedangkan pada penelitian

Tugas Akhir ini terdapat web yang dapat digunakan untuk memonitor penggunaan energi listrik di setiap ruangan.
3) Intelligent Smart Home Automation and Security System Using Arduino **and Wi-fi**
Penulis; Tahun J.Chandramohan, R.Nagarajan, K.Satheeshkumar, N.Ajithkumar, P.A.Gopinath, dan S.Ranjithkumar [5]
Pembahasan Pada penelitian ini, dilakukan untuk membuat sebuah sistem Smart Home Automation dengan menggunakan Arduino dan *Wi-Fi*. Sistem yang dibuat pada penelitian ini dapat meningkatkan fleksibilitas dalam mengontrol dan memonitor sistem perangkat yang terpasang di rumah. Pengguna dapat melakukan kontrol perangkat elektronik yang ada di dalam rumah secara remote menggunakan ponsel, seperti mematikan dan menyalakan perangkat. Adapun sistemnya menggunakan micro-web server dengan Internet Protocol (IP) yang diintegrasikan melalui *Wi-Fi* sebagai media koneksi antara perangkat dengan sistem. Untuk memutus aliran listrik, sistem ini menggunakan *relay* sebagai switch yang ditentukan sesuai perintah logic pada Arduino.
Keterkaitan Penelitian tugas akhir yang dilakukan berkaitan dengan penerapan sistem Automation dengan menggunakan Arduino yang memungkinkan pengguna untuk dapat mengontrol dan memonitor perangkat elektronik secara remote. Pengguna dapat mematikan dan menyalakan perangkat yang telah terpasang. Selain itu, keterkaitan juga terdapat pada penggunaan *relay* sebagai switch untuk memutus aliran listrik yang terhubung pada perangkat.
Perbedaan dengan penelitian Tugas Akhir ini Sistem yang diajukan pada paper ini tidak mengakomodasi fitur untuk monitoring besaran penggunaan energi listrik. Selain itu, menggunakan aplikasi berbasis android untuk mengontrol perangkat elektroniknya. Sedangkan pada penelitian Tugas Akhir ini sistem yang diajukan memungkinkan user untuk dapat memonitor besaran

penggunaan energi listrik secara periodik melalui aplikasi web. User cukup mengakses laman web tanpa harus meng-install aplikasi sehingga lebih fleksibel dalam penggunaannya.

2.2 Dasar Teori

2.2.1 Konservasi Energi

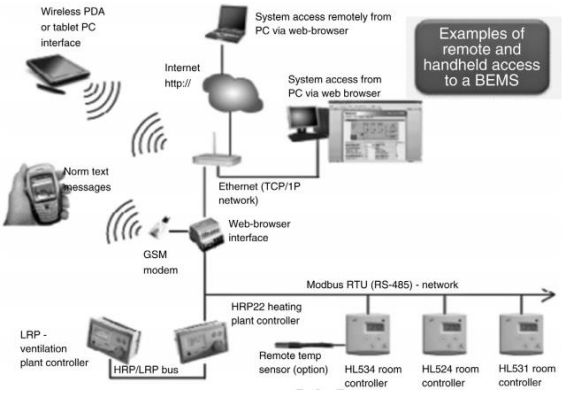
Konservasi Energi adalah upaya yang dilakukan untuk efisiensi pemakaian energi listrik sesuai dengan kebutuhan agar terhindar dari pemborosan listrik. Pemerintah telah mengeluarkan peraturan terkait konservasi energi, yakni di dalam Peraturan Pemerintah No. 70 Tahun 2009 tentang Konservasi Energi. Menurut undang – undang, konservasi energi adalah upaya sistematis, terencana, dan terpadu guna melestarikan sumber daya energi dalam negeri serta meningkatkan efisiensi pemanfaatannya [2]. Contoh paling sederhana terkait dengan konservasi energi adalah dengan mematikan lampu di ruangan apabila tidak sedang digunakan.

2.2.2 Building Energy Management Systems (BEMS)

Building Energy Management Systems (BEMS) adalah sistem berbasis komputer yang digunakan untuk mengontrol dan memantau peralatan mekanik dan listrik di gedung seperti ventilasi, pemanas, penerangan, sistem tenaga, dan sebagainya. BEMS juga terkadang disebut sebagai sistem manajemen gedung / *Building Management Systems* (BMS). Perangkat – perangkat yang penting di dalam gedung akan dihubungkan ke komputer sentral untuk dapat dilakukan kontrol otomatis berdasarkan waktu *on/off*, temperatur, dan lain sebagainya sesuai dengan kebutuhan masing – masing bangunan gedung.

Penerapan *Energy Management Systems* (EMS) dapat membantu dalam penghematan konsumsi energi listrik dan cukup mudah untuk diimplementasikan. *Energy Management Systems* (EMS) memungkinkan untuk mengontrol penggunaan listrik hanya pada saat dibutuhkan saja, sehingga dapat

mengeliminasi penggunaan energi yang terbuang sia-sia [6]. Pada software BEMS, fitur-fitur yang dapat disediakan antara lain meliputi kebutuhan *monitoring, control functions, alarm*, serta memungkinkan operator untuk dapat mengoptimalkan kinerja perangkat pada masing-masing bangunan. Di tiap-tiap ruangan akan dipasang juga perangkat sensing atau juga dapat disebut sebagai sensor yang berfungsi sebaga indera untuk mengirimkan sinyal berupa digital maupun analog. Nilai yang diterima oleh sensor contohnya adalah suhu temperatur, kelembapan ruangan, dan lain sebagainya yang akan dikirimkan ke sistem BEMS. Selanjutnya pada sistem BEMS nilai tersebut akan diolah menjadi sebuah informasi dan juga memungkinkan untuk menentukan pengambilan keputusan dengan mematikan atau menyalakan perangkat listrik secara manual maupun otomatis. Berikut adalah gambaran arsitektur sederhana BEMS yang ditunjukkan pada gambar 2.1.



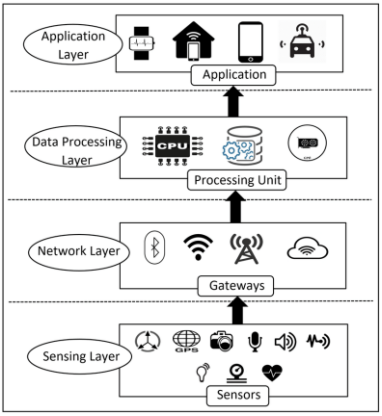
Gambar 2.1 Contoh Arsitektur BEMS [6]

Pada penelitian ini, konsep seperti ini digunakan dengan menghubungkan seluruh perangkat elektronik penting yang ada di setiap ruangan ke mikrokontroller Arduino dan dikontrol berdasarkan kondisi input dari motion sensor (PIR sensor) yang akan mendeteksi adanya aktifitas manusia.

2.2.3 Internet of Things

Internet of Things (IoT) adalah sebuah konsep yang dapat diaplikasikan untuk menghubungkan antar perangkat atau objek fisik yang semula tidak dapat berkomunikasi menjadi dapat saling terkoneksi dan berinteraksi secara otomatis [7]. Masing – masing objek dapat saling terhubung melalui jaringan internet sebagai media untuk transfer data. Bentuk koneksi yang terjadi pada suatu perangkat pada IoT dapat berupa sensor, *smartphone*, perangkat elektronik rumah tangga, dan perangkat fisik lainnya yang masing – masing akan dihubungkan dengan jaringan internet dan mikrokontroller.

Mikrokontroler berfungsi sebagai *processing unit* yang digunakan untuk mengontrol rangkaian elektronik, mengirimkan baris kode perintah pada perangkat yang terhubung dan menerima data. Selanjutnya data yang telah diterima akan dikirimkan ke cloud melalui jaringan internet untuk diproses menjadi sebuah informasi. Sebaliknya, user juga dapat mengontrol perangkat fisik yang telah terhubung dengan ekosistem IoT ini dengan cara mengirimkan perintah melalui jaringan internet. Perintah diteruskan ke sistem mikrokontroller yang selanjutnya akan diproses untuk melakukan sebuah tindakan sesuai perintah. Adapun terkait IoT Architecture Layer beserta komponennya ditunjukkan pada gambar 2.2 berikut.



Gambar 2.2 Arsitektur Iot dan komponennya [7]

Menurut Amit Kumar Sikder, dkk dalam papernya yang berjudul “A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications” [7] yang ditunjukkan pada gambar 2.2 menyebutkan bahwa arsitektur IoT terbagi menjadi 4 *layer*, yakni:

1. *Sensing Layer*
   Sesuai dengan namanya, tujuan utama pada *Sensing Layer* adalah sebagai indera yang digunakan untuk mengidentifikasi fenomena apa pun di perangkat-perangkat dan mendapatkan data dari lingkungan sekitar sistem. Pada layer ini, perangkat yang digunakan biasa disebut dengan sensor. Terdapat beberapa tipe sensor, antara lain *motion sensors*, *environmental sensors*, dan *position sensors*.
2. *Network Layer*
   Network layer bertindak sebagai saluran komunikasi yang digunakan untuk mentransfer data yang telah didapatkan dari *sensing layer* ke perangkat lain yang telah terhubung. Pada perangkat IoT, terdapat beragam teknologi komunikasi yang digunakan seperti *Bluetooth*, *Wi-Fi*, jaringan seluler, dan berbagai jenis lainnya yang memungkinkan untuk saling bertukar data antar perangkat dalam jaringan yang sama.
3. *Data Processing Layer*
   Data processing layer terdiri dari *main data processing unit* dari perangkat IoT. Pada lapisan ini data yang telah dikumpulkan di *Sensing Layer* akan diterima dan dilakukan analisis data untuk mengambil keputusan berdasarkan hasilnya. *Layer* ini dapat membagikan hasil pemrosesan data dengan perangkat lain yang terhubung melalui lapisan *network*.
4. *Application Layer*
   Application Layer berfungsi untuk mengimplementasikan dan menyajikan hasil dari *Data Processing Layer* untuk mencapai tujuan pengoperasian sistem IoT.

2.2.4 Arduino

Arduino adalah sebuah platform mikrokontroller *open source* yang cukup populer digunakan dalam pengimplementasian IoT. Mikrokontroler ini dapat dengan mudah dalam pemrogramannya dan dapat dengan bebas diubah dan diprogram ulang dalam waktu yang cepat. Diperkenalkan pada tahun 2005, *platform* Arduino didesain untuk hobi, pelajar, maupun professional yang ingin membuat sebuah perangkat yang dapat berinteraksi dengan lingkungan sekitar menggunakan perangkat pendukung seperti sensor dan aktuator [8]. Arduino memiliki kemampuan untuk menerima dan mengirimkan data ke berbagai perangkat pendukung, bahkan dapat juga dihubungkan dengan jaringan internet sebagai media untuk komunikasi dan transfer data. Perangkat ini dapat diprogram menggunakan bahasa *Simplified C++* [9]. Dalam pengembangan kode programnya, pengguna dapat mengakses melalui perangkat lunak bernama Arduino *IDE*. Terdapat banyak jenis dari Arduino yang dibedakan berdasarkan kemampuan pemrosesannya seperti *clock speed*, *memory*, ketersediaan jenis port yang didukung, dan jumlah pin yang I/O yang disediakan. Board Arduino mampu menerima dan mengirimkan sinyal berjenis *digital* maupun *analog*. Beberapa contoh jenis Arduino yang cukup populer digunakan saat ini antara lain: Arduino Nano, Arduino UNO, dan Arduino Mega. Namun masing-masing *board* Arduino tersebut harus dihubungkan dengan sebuah *shield* tambahan seperti ESP 8266 untuk dapat terhubung dengan jaringan internet sebagai media komunikasi dan transfer data dalam pengimplementasian konsep IoT

2.2.4.1 Mega 2560 + ESP8266

Mega 2560+ESP8266 adalah *board* mikrokontroller berbasis Arduino *mega 2560* yang sudah ditambahkan modul *Wi-Fi ESP8266* di dalamnya. Secara spesifikasi, *Wemos Mega R3* sangat identik dengan Arduino Mega 2560. Mikrokontroler ini memiliki *chip* berjenis *Atmega2560* dengan kecepatan *CPU* 16 *MHz*, 54 pin digital *input*/*output*, 16 pin *analog input*, 4 *UARTs*

(*serial*), *USB connector*, *Jack DC*, dan tombol reset [10]. Board Mega 2560 + ESP8266 ditunjukkan pada gambar 2.3 dibawah ini.

Gambar 2.3 Board Mega2560 + ESP8266

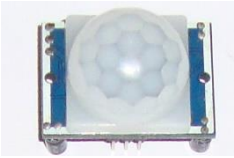
Wemos Mega R3 ini juga diprogram menggunakan Arduino IDE dengan bahasa *Simplified* C++. Tabel 2.2 berikut menunjukkan spesifikasi detil dan perbandingan antara *board* Mega 2560 + ESP8266 dengan Arduino Mega dan Uno.

2.2.5 Sensor

Sensor adalah sebuah alat yang bersifat sebagai indera yang berfungsi untuk mengidentifikasi fenomena yang terjadi pada perangkat dan mengambil data / mendeteksi sinyal – sinyal yang berasal dari perubahan suatu energi. Fenomena perubahan energi yang terdeteksi selanjutnya akan dirubah menjadi sebuah sinyal elektrik yang digunakan sebagai data masukan pada mikrokontroller. Beberapa contoh fenomena fisik yang dapat diterima datanya oleh sensor antara lain suhu, tekanan udara, medan magnet, arus listrik, sinyal infrared, dan lain sebagainya. Setelah sensor mengirim data kepada mikrokontroller Arduino, selanjutnya data dari sensor akan diolah oleh mikrokontroller untuk dilakukan tindakan sesuai dengan aturan logika yang telah ditentukan. Beberapa jenis dari sensor antara lain: *Motion sensors* (sensor pendeteksi gerakan), *Environmental sensors* (sensor pendeteksi perubahan pada lingkungan sekitar), dan *Position sensors* (sensor pendeteksi posisi fisik) [7].

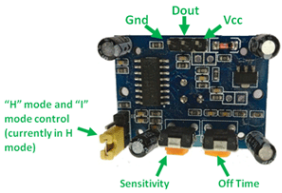
2.2.5.1 Sensor Passive Infrared Receiver (PIR)

Sensor PIR (*Passive Infrared Receiver*) adalah sebuah alat yang dapat mendeteksi sebuah gerakan yang terjadi dengan mengukur perubahan intensitas radiasi inframerah / tingkat panas yang dipancarkan oleh benda disekitarnya. Sensor PIR (*Passive Infrared*) terbuat dari bahan kristal yang akan menimbulkan beban listrik ketika terkena panas dan pancaran sinyal infra merah. Ketika terdapat gerakan yang terdeteksi oleh sensor ini, maka sensor akan mengirimkan sinyal *high* pada pin output [4]. Berikut adalah bentuk sensor pir yang ditunjukkan pada gambar 2.4 dibawah ini.



Gambar 2.4 Bentuk Sensor Pir [17]

Pada penelitian kali ini, nilai output high dari sensor pir kemudian diolah oleh mikrokontroller menjadi sebuah perintah untuk mengaktifkan sistem kelistrikan. Sinyal *high* yang diterima digunakan sebagai *trigger* pada transistor yang selanjutnya memicu *relay* akan menghubungkan jalur aliran listrik. Sebaliknya apabila sensor tidak mendeteksi adanya aktivitas manusia di dalam ruangan, maka sensor akan mengirimkan nilai output *low* yang akan diproses oleh mikrokontroller sebagai sinyal untuk memutus aliran listrik. Berikut adalah pinout dari sensor PIR yang ditunjukkan pada gambar 2.5 dibawah ini.



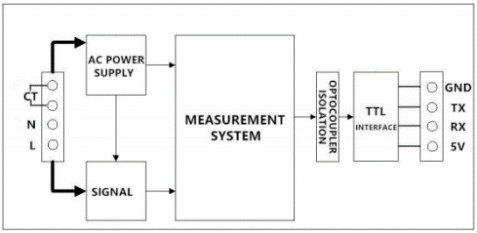
Gambar 2.5 Pinout Sensor PIR [17]

2.2.5.2 Sensor PZEM-004T V3

PZEM-004T V3 adalah sebuah modul sensor yang dapat digunakan untuk mengukur besaran tegangan, arus, daya, frekuensi, *power factor*, dan penggunaan energi dari listrik berjenis arus AC. Sensor ini memiliki jalur TTL untuk komunikasi melalui serial Rx dan TX yang dapat dihubungkan ke Arduino untuk pengolahan data lebih lanjut. Terdapat 2 jenis sensor PZEM-004T yang dibedakan berdasarkan kemampuan pembacaan maksimum arusnya, yakni PZEM-004T 10A dan PZEM-004T 100A. Bentuk sensor PZEM-004T V3 ditunjukkan pada gambar 2.6 dibawah ini.

Gambar 2.6 Bentuk PZEM-004T V3 [11]

Berikut adalah functional block dari modul sensor PZEM004T V3 ditunjukkan pada gambar 2.7 dibawah ini.

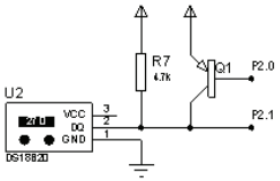


Gambar 2.7 Functional Block PZEM-004T V3 [11]

Pada penelitian kali ini, sensor yang digunakan adalah PZEM-004T 100A dikarenakan besarnya kebutuhan listrik yang digunakan di gedung Departemen Sistem Informasi. Sensor ini memiliki kelebihan pada tingkat akurasinya yang cukup tinggi, dan memiliki ukuran yang kompak dengan berbagai fitur di dalam satu sensor.

2.2.5.3 Sensor Suhu DS18B20

Sensor DS18B20 adalah sebuah sensor suhu yang diciptakan oleh *Dallas Company* di Amerika. Sensor ini bekerja pada tegangan 3 V ~ 5V DC dan memiliki kemampuan mengukur temperatur dari -55°C hingga +125°C dengan tingkat akurasi sebesar 0,00625°C. Sensor DS18B20 memiliki 3 pin, yakni pin VCC (*power supply*), GND (*ground*), dan DATA. Terdapat fitur yang cukup unik pada sensor ini, yakni memiliki ID yang unik sehingga dalam penggunaannya dapat menyambungkan banyak sensor hanya dalam 1 pin saja [12]. Circuit sensor DS18B20 ditunjukkan pada gambar 2.8 berikut ini.



Gambar 2.8 Sensor DS18B20 Circuit

2.2.6 Relay

Relay adalah saklar (*switch*) yang dioperasikan secara elektrik yang terdiri atas coil elektromagnet dan kontak saklar mekanikal. Dalam pengoperasiannya, *relay* menggunakan prinsip elektromagnet untuk membuka atau menutup saklar. Berdasarkan kontak poinnya, *relay* dibedakan menjadi 2 jenis yaitu *Normally Close* (NC) dan *Normally Open* (NO). Relay jenis NC bekerja dengan kondisi awal sebelum dialiri arus listrik berada pada posisi *close* (tertutup), sebaliknya *relay* berjenis NO akan selalu berada pada posisi terbuka pada saat awal sebelum diaktifkan. Pada dasarnya *relay* terdiri dari 4 komponen dasar, yaitu *coil electromagnet, Armature, Switch Contact Point* (saklar), dan *spring* [13]. Contoh tampilan *relay* dapat dilihat pada gambar 2.9.

Gambar 2.9 Bentuk Relay [13]

2.2.7 Database

Database atau dalam bahasa Indonesia disebut sebagai "Basis Data" adalah sebuah kumpulan data yang disimpan secara sistematis di dalam komputer yang dapat diolah atau dimanipulasi menggunakan perangkat lunak untuk dapat menghasilkan informasi. Database saat ini memegang peranan sangat penting dalam sistem informasi dikarenakan berfungsi sebagai gudang data yang akan diolah [14]. Data yang tersimpan berupa tabel kolom dan baris yang dapat saling terhubung sesuai relasinya. Sebagian besar database menggunakan bahasa *Structured Query Language* (*SQL)* untuk menulis dan queri data.

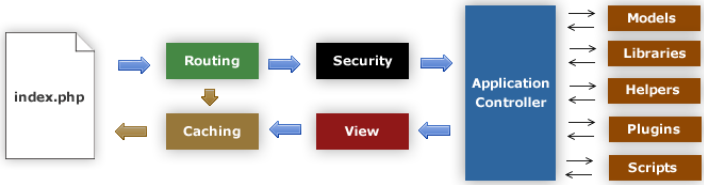
2.2.7.1 Database My SQL

MySQL adalah sistem manajemen basis data (DBMS) relasional *open source* yang berbasis pada bahasa *Structured Query Language* (SQL). MySQL dirancang dan dioptimalkan untuk aplikasi web dan dapat berjalan di berbagai *platform* [14]. Penyimpanan data yang pada MySQL dilakukan dengan cara menyimpan pada tabel-tabel yang terpisah dan dihubungkan menggunakan relasi. Adapun terkait relasi terdapat 3 jenis, yakni *One-to-One*, *One-to-Many*, dan *Many-to-Many*. Dengan menerapkan struktur penyimpanan semacam ini membuat kinerja menjadi lebih cepat dan meningkatkan fleksibilitas dalam mengatur tabel data.

2.2.8 Codeigniter

CodeIgniter adalah sebuah *framework open source* PHP dengan model MVC (*Model*, *View*, *Controller*). Codeigniter pertamakali dikembangkan oleh Rick Ellis pada tahun 2006, dapat digunakan untuk membangun sebuah website dinamis

menggunakan bahasa pemrograman PHP dengan mudah. Codeigniter dilengkapi dengan dokumentasi yang cukup lengkap sehingga dapat membantu user dalam pengembangan website. Selain itu, *framework* ini juga dilengkapi dengan berbagai *libraries* yang bebas digunakan oleh pengembang dalam membangun web, sehingga selain memudahkan juga dapat mempercepat pengembangan sebuah web. Berbeda dengan PHP biasa yang menggabungkan logika pemrograman dengan tampilan, Codeigniter memisahkan logika aplikasi dan tampilan ke dalam *Model*, *View*, dan *Controller*. Model berisi fungsi yang digunakan untuk mengambil, menyisipkan, dan memperbarui data pada *database*. *View* adalah sebuah informasi yang ditampilkan langsung ke pengguna, contohnya adalah laman web. Sedangkan *controller* adalah sebagai perantara antara *model*, *view*, dan *resource* lainnya yang terlibat dalam pembuatan halaman web [15]. Cara kerja codeigniter ditunjukkan pada gambar 2.10 berikut ini.



Gambar 2.10 Cara Kerja Codeigniter

2.2.9 Perhitungan Penghematan Energi Listrik

Perhitungan penghematan penggunaan energi listrik dilakukan untuk mengetahui besaran penghematan / peningkatan efisiensi yang ditimbulkan dengan menerapkan sistem yang diajukan. Dasar perhitungan penghematan dilakukan sesuai dengan yang dilakukan oleh J.L. Taulo dan A.B. Sebitosi pada papernya yang berjudul "*Energy consumption analysis for the Malawian tea industry*" [16] berikut ini.

$$ES = E_{d,el}^{P} - EC_{SP} \quad (1)$$

Keterangan :

ES = Besar penghematan energi listrik (kWh)

$E_{d,el}^{P}$ = Besar energi yang terpakai dalam sehari operasional sebelum diterapkan sistem Smart Power (kWh)

EC_{SP} = Besar energi yang terpakai dalam sehari operasional sesudah diterapkan sistem Smart Power (kWh)

Persamaan (1) di atas digunakan untuk mengetahui selisih penghematan energi (kWh) yang digunakan sebelum dan sesudah penerapan sistem Smart Power. Selanjutnya untuk mengetahui persentase peningkatan efisiensi / penghematan dari kondisi sebelumnya dilakukan perhitungan dengan persamaan (2) berikut ini.

$$\%ES = \frac{ES}{E_{d,el}^{P}} \times 100\% \qquad (2)$$

Keterangan :

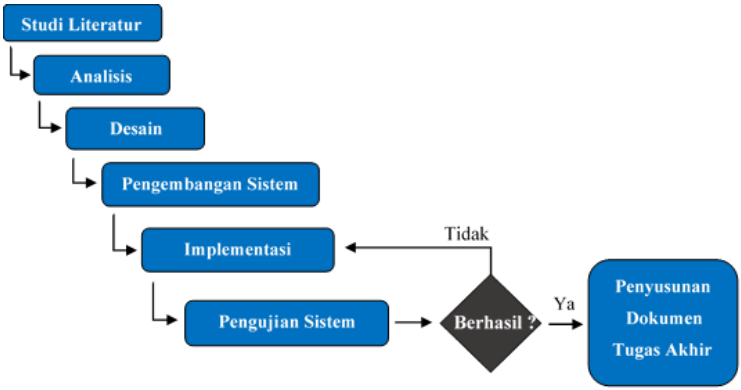
$\%ES$ = Persentase penghematan energi listrik dibandingkan dengan kondisi sebelum diterapkannya sistem Smart Power (%).

ES = Besar penghematan energi listrik (kWh)

$E_{d,el}^{P}$ = Besar energi yang terpakai dalam sehari operasional sebelum diterapkan sistem Smart Power (kWh)

BAB III
METODOLOGI

Pada bagian ini akan menjelaskan mengenai tahapan yang dilakukan dalam pengerjaan tugas akhir menggunakan *software development life cycle* (SDLC) model *waterfall* yang telah disesuaikan. Adapun alur pengerjaan ditunjukkan pada gambar 3.1 di bawah ini.



Gambar 3.1 Metodologi Pengerjaan Tugas Akhir

3.1 Studi Literatur

Pada tahap ini dilakukan pengumpulan literatur yang mendukung dalam penyelesaian tugas akhir ini. Literatur yang digunakan adalah detil penjelasan konsep-konsep atau teori dari penelitian sebelumnya yang memiliki keterkaitan dengan pengerjaan tugas akhir ini. Selain itu, juga dilakukan pembacaan mendalam terkait perbedaan penelitian sebelumnya dengan tugas akhir yang akan dikerjakan. Keluaran dari studi literatur adalah pemahaman metode dan konsep yang akan diterapkan dalam pengerjaan tugas akhir.

3.2 Analisis

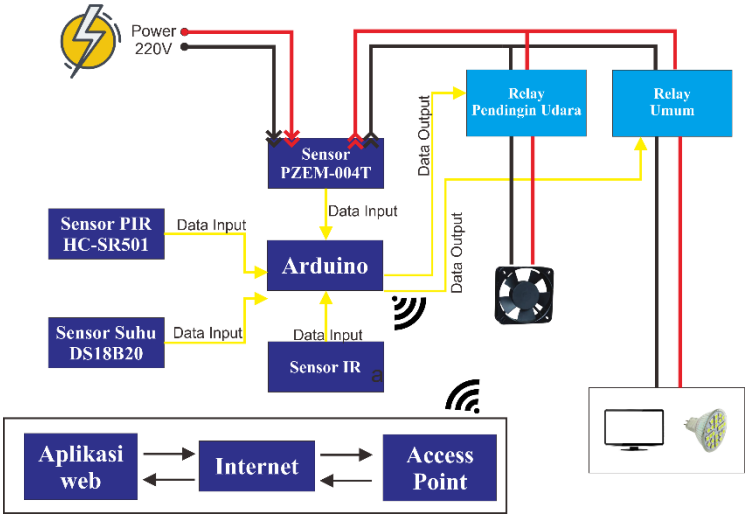
Seluruh pengetahuan yang telah didapat dari hasil studi literatur akan digunakan sebagai panduan dalam tahap analisis. Pada tahap analisis dilakukan penggalian kebutuhan sistem yang

akan dikembangkan. Hasil dari tahap analisis adalah spesifikasi kebutuhan termasuk pembuatan perangkat lunak (*software*) dan perangkat keras (*hardware)* dari sistem yang akan dibuat. Adapun spesifikasi sistem yang akan dibuat adalah sebagai berikut:

1. Mikrokontroler Arduino
   Mikrokontroler Mega2560 + ESP8266 digunakan sebagai *processing unit* dalam sistem yang akan dibuat. Tipe ini digunakan dikarenakan kapasitas pin yang disediakan cukup banyak dan performanya yang baik. Terdapat modul *Wi-fi* ESP8266 di dalamnya yang dapat digunakan untuk menghubungkan Arduino ke jaringan *Wi-fi* sebagai media transfer data.
2. **Sensor**
   Pada sistem yang akan dibuat, dibutuhkan sensor yang digunakan untuk menerima data dari lingkungan. Adapun sensor yang diperlukan pada sistem ini adalah sensor arus *PZEM-004T V3* yang digunakan untuk mendapatkan informasi seputar listrik AC yang digunakan seperti tegangan, arus, daya, hingga perhitungan besaran energi yang digunakan. Selanjutnya terdapat sensor suhu DS18B20 untuk mendeteksi suhu ruangan dan sensor PIR untuk mendeteksi adanya aktivitas di dalam ruangan.
3. **Aplikasi Web**
   Pada sistem ini, web digunakan sebagai media untuk kebutuhan monitoring dan kontrol. Adapun fitur yang diperlukan adalah web mampu menampilkan informasi penggunaan energi listrik dari data sensor dan mengirimkan perintah ke Arduino sesuai kebutuhan.
4. **Model Miniatur Ruang Kelas**
   Model miniatur kelas diperlukan untuk mensimulasikan kinerja dari sistem yang dibuat. Model ini digunakan untuk memastikan apakah sistem dapat bekerja sesuai sebagaimana mestinya.

3.3 Desain

Pada tahap ini dilakukan desain / perancangan dari sistem yang akan dibuat. Adapun desain sistem yang dibuat menjelaskan tentang bagaimana sistem dapat bekerja dan menjelaskan alur yang terjadi di dalam sistem meliputi *flow* kerja sistem, bagaimana data dari sensor dapat ditransfer ke Ardruino hingga ditampilkan ke web, bagaimana sistem kelistrikan dapat dikontrol secara manual dan otomatis melalui web. Keluaran pada tahap ini adalah hasil analisa berupa kebutuhan fungsional dan non fungsional dan desain dari sistem.



Gambar 3.2 Desain Arsitektur Sistem

Gambar 3.2 memperlihatkan desain sistem yang akan dibuat pada penelitian ini. Pada penelitian ini menggunakan 1 mikrokontroller mega2560+Wifi untuk mengontrol 6 ruang kelas yang masing – masing berisi perangkat elektronik *dummy* display, lampu dan fan. Sistem dapat bekerja dimulai dari penerimaan data dari masing-masing sensor yang terhubung pada perangkat elektronik dan lingkungannya baik sensor infrared, sensor suhu, sensor PZEM-004T, dan sensor PIR. Data dari masing – masing sensor akan dikirmkan ke mikrokontroller Arduino untuk dilakukan pengolahan data. Selanjutnya data yang diterima oleh Arduino akan diolah untuk diambil

keputusan sesuai dengan program yang telah ditetapkan. Sensor PZEM-004T terhubung ke jalur listrik menuju perangkat elektronik yang berfungsi untuk mendapatkan besaran arus listrik, tegangan, daya, dan total energi listrik yang digunakan oleh perangkat elektronik di dalam kelas tiap bulannya dalam satuan kWh. Selanjutnya informasi penggunan daya listrik pada masing – masing ruangan ini ditampilkan pada layar lcd dan web dalam bentuk grafik secara periodik. Sumber listrik yang menuju perangkat elektronik akan terhubung ke *relay* terlebih dahulu. Perangkat elektronik akan hidup atau mati berdasarkan adanya deteksi aktifitas melalui sensor PIR yang dikirimkan oleh Arduino menuju *switch relay*. Pengguna juga dapat mengatur mode kehematan melalui web.

3.4 Pengembangan Sistem

Pada tahap ini dilakukan proses pengembangan sistem, yang terbagi menjadi pengembangan perangkat keras (*hardware*), perangkat lunak (*software*), dan komponen pendukung.

3.4.1 Pengembangan Perangkat Keras (Hardware)

Pada proses ini, dilakukan perakitan sensor dan driver *relay* yang akan dihubungkan dengan mikrokontroller Arduino. Pada proses ini juga dilakukan perancangan sistem *wiring* yang tepat, sehingga sistem dapat bekerja sesuai ekspektasi. Proses kalibrasi pada masing-masing sensor juga perlu dilakukan untuk menyesuaikan dengan *environment* sistem dan memastikan *output* yang dihasilkan dari sensor sesuai dengan yang diharapkan.

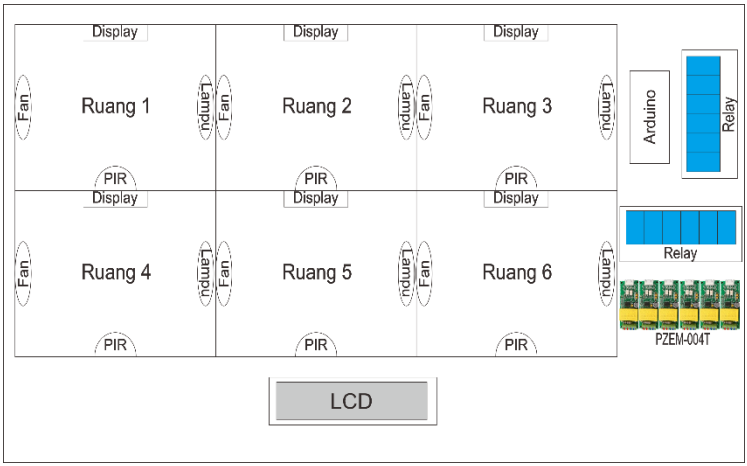
3.4.2 Pengembangan Perangkat Lunak (Software)

Pembuatan program Arduino dilakukan sesuai dengan kebutuhan sistem. Program yang dibuat harus dapat mengakomodasi fitur yang ditawarkan, yakni memungkinkan sistem untuk dapat dikontrol secara manual maupun secara otomatis berdasarkan parameter dan *setting* yang telah ditentukan. Selain itu, program yang dibuat juga harus dapat mengirim data melalui *HTTP Request* guna keperluan sistem

monitoring dan pengaturan kontrol. Selanjutnya dilakukan proses pengembangan web dengan menggunakan *framework CodeIgniter* berbasis bahasa pemrograman PHP dan diintegrasikan dengan *MySQL* sebagai databasenya. Aplikasi web berguna untuk menampilkan status dari sistem, menampilkan informasi penggunaan listrik secara periodik dan mengirimkan perintah ke Arduino.

3.4.3 Pengembangan Komponen Pendukung

Pengembangan komponen pendukung dilakukan dengan membuat model miniatur kelas yang disesuaikan sebagaimana kondisi penggunaan perangkat listrik pada ruangan. Dalam hal ini, jumlah kelas yang dibuat adalah sebanyak 6 kelas. Besaran ukuran maket akan dibuat dengan skala perbandingan sesuai dengan kondisi aslinya. Adapun desain model miniatur kelas yang digunakan sebagai komponen pendukung penelitian Tugas Akhir ini ditunjukkan pada gambar 3.3 dibawah ini.



Gambar 3.3 Desain Model Miniatur Kelas

3.5 Implementasi

Setelah dilakukan tahap pengembangan, tahapan selanjutnya adalah merangkai perangkat Arduino beserta sensornya kedalam model miniatur kelas yang sudah dibuat. Dalam hal ini

juga perlu dilakukan proses kalibrasi ulang dikarenakan kemungkinan perbedaan lingkungan yang dapat menyebabkan perbedaan hasil. Perbedaan terbesar pada data masukan sensor yang perlu disesuaikan kembali terdapat pada sensor PIR.

Proses kalibrasi sensor PIR dilakukan dengan mengukur kemampuan jarak deteksi manusia oleh sensor PIR di ruang kelas. Setelah mendapatkan besaran jarak deteksinya, sensor PIR dipasangkan pada maket dengan penyesuaian. Adapun deteksi manusia pada maket dilakukan dengan menggunakan objek pengganti yang memiliki ukuran yang telah disesuaikan.

Output dari tahap ini adalah sistem dapat menampilkan data yang didapatkan dari Arduino ke display dan dikirim ke *web*. Data yang ditampilkan adalah data pemakaian energi pada perangkat yang digunakan secara periodik. Selain itu, pada tahap ini kemampuan sistem kontrol penggunaan listrik baik secara manual maupun otomatis juga sudah dapat dijalankan.

3.6 Pengujian Sistem

Pada tahap ini akan dilakukan proses pengujian terhadap sistem yang telah dibuat. Pengujian dilukan dengan menjalankan beberapa skenario *test case* dan ekspektasi yang telah ditentukan. Tahapan pengujian dilakukan dengan melakukan *unit testing* yakni melakukan pengujian pada masing-masing fungsi dan system *integration testing* yakni pengujian sistem secara keseluruhan. Proses ini harus memenuhi aspek fungsional dan non-fungsional, apabila ditemukan ketidaksesuaian ataupun *bug* pada sistem maka akan dilakukan penyempurnaan kembali pada tahap implementasi. Setelah sistem berhasil dilakukan pengujian sesuai *test case* dan berfungsi sebagaimana yang diharapkan, maka selanjutnya hasil dari pengujian akan dimasukkan ke dalam dokumen pengujian.

3.7 Penyusunan Dokumen Tugas Akhir

Penyusunan dokumen tugas akhir merupakan tahapan terakhir dalam pengerjaan tugas akhir ini yang berisi output dari keseluruhan proses penelitian yang telah dilakukan sesuai

metodologi. Kesimpulan penelitian dan saran juga diberikan untuk kepentingan 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. Perancangan dilakukan baik pada perangkat keras (*hardware*) maupun pada perangkat lunak (*software*).

4.1 Analisis Kebutuhan

Pada tahap awal dari perancangan dimulai dengan mengumpulkan informasi mengenai kondisi lingkungan yang akan diimplementasikan sistem ini. Pengumpulan informasi dilakukan dengan melakukan wawancara kepada narasumber yang terkait dengan kelistrikan di DSI dan survey langsung ke masing-masing kelas beserta ruang panel kelistrikan untuk dilakukan pencatatan. Narasumber wawancara adalah Nanang Subiantoro yang memiliki kewenangan sebagai Teknisi Sarana dan Prasarana di Departemen Sistem Informasi.

Seiring dengan meningkatnya kebutuhan listrik di DSI untuk memenuhi kebutuhan akademik dan penelitian tentu harus diimbangi dengan upaya pengehamatan energi listrik. Kondisi saat ini yang sering terjadi adalah ketika peralatan elektronik di dalam ruangan dibiarkan menyala ketika jam peralihan dengan tidak adanya kegiatan didalamnya sehingga terkesan pemborosan. Hal inilah yang mendasari pembuatan sistem ini yang memiliki tujuan untuk menekan angka penggunaan listrik agar penggunaannya lebih tepat guna dan tidak terbuang sia-sia. Selain itu, sistem juga dapat memiliki kemampuan untuk *monitoring* dan *controlling* agar petugas terkait dapat meningkatkan pengawasan terhadap kelistrikan untuk mengantisipasi adanya hal-hal yang tidak diinginkan. Untuk memenuhi kebutuhan tersebut dibutuhkan sensor sebagai data masukan yang selanjutnya dikirimkan ke mikrokontroller Arduino untuk dilakukan pemrosesan dan dilakukan pengambilan keputusan. Mikrokontroler Arduino terhubung

dengan relai yang berfungsi untuk memutus aliran arus listrik AC 220V ketika tidak dibutuhkan. Adapun data masukan kondisi lingkungan yang digunakan untuk menentukan apakah listrik perlu dinyalakan atau sebaliknya didapat dari *motion sensor PIR* yang dipasangkan di setiap kelas untuk mendeteksi keberadaan manusia. Selain sensor *PIR*, pada sistem ini juga disematkan sensor suhu *DS18B20* untuk mendapatkan data suhu ruangan yang digunakan sebagai acuan dalam pengoperasian pendingin ruangan ketika mode adaptif dinyalakan. Detil informasi kelistrikan pada sistem ini didapatkan dari sensor *PZEM-004T V3* yang dapat melakukan pencatatan besaran tegangan, arus, daya, hingga total energi yang terpakai. Untuk mensimulasikan fungsional sistem secara keseluruhan pada penelitian ini perlu dibuat sebuah model miniatur kelas berisi 6 ruang kelas sesuai dengan batasan masalah yang telah ditentukan sebelumnya. Ruang kelas yang dipilih sebagai sampel adalah kelas SI 4101, SI 4102, SI 3101, SI 3102, SI 1101, SI 1102. Sistem yang diajukan menyediakan 3 pilihan mode untuk meningkatkan fleksibilitas sesuai kebutuhan. Ketiga mode tersebut ditampilkan pada Tabel 4.1 di bawah ini.

Tabel 4.1 Pilihan Mode Kehematan

Mode	Deskripsi
Adaptif	Mode adaptif merupakan mode kehematan dengan tetap menjaga kestabilan suhu ruangan yang diinginkan. Mode ini akan mematikan seluruh perangkat elektronik di dalam ruangan ketika tidak adanya kegiatan kecuali pendingin ruangan. Pendingin ruangan akan aktif selama suhu ruangan terdeteksi diluar keadaan optimal sesuai *threshold* yang telah ditentukan.
Auto-Cut	Mode *Auto-Cut* adalah mode paling hemat yang dapat dipilih, dengan mematikan seluruh perangkat

	elektronik yang ada di dalam ruangan termasuk pendingin ruangan ketika *motion sensor* tidak mendeteksi keberadaan aktivitas di dalamnya.
Manual	Mode manual adalah mode dimana sistem penghematan tidak aktif. Seluruh perangkat elektronik akan dikontrol secara manual melalui saklar yang telah disediakan.

Dalam penerapannya, mode Adaptif dapat membantu menekan penggunaan energi listrik di kelas secara maksimal dengan asumsi pada saat perpindahan jam perkuliahan waktunya berdekatan. Sistem akan menjaga suhu ruangan tetap stabil sehingga pemakaian daya listrik untuk angkatan daya pertama pada pendingin ruangan dapat ditekan. Sedangkan mode Auto-Cut dapat berfungsi secara maksimal ketika kelas tidak digunakan dalam waktu yang berdekatan sehingga listrik akan mati baik untuk pendingin ruangan maupun perangkat listrik lainnya di dalam kelas tersebut.

Pilihan mode Adaptif ataupun Auto-Cut yang telah dijelaskan pada Tabel 4.1 di atas dapat berfungsi dan diterapkan sesuai dengan kondisi pengambilan keputusan yang telah ditetapkan berdasarkan data masukan sensor. Kondisi-kondisi yang telah ditetapkan ditunjukkan pada Tabel 4.2 di bawah ini.

Tabel 4.2 Kondisi Pengambilan Keputusan

Kondisi	Pengambilan Keputusan
Suhu di bawah *threshold*	Mengirim sinyal ke relai yang terhubung ke stopkontak pendingin ruangan untuk memutus aliran arus listrik.
Suhu di atas *threshold*	Mengirim sinyal ke relai yang terhubung ke stopkontak pendingin ruangan untuk menghubungkan aliran arus listrik.

Terdeteksi keberadaan manusia oleh sensor *PIR*	Mengirim sinyal ke relai yang terhubung ke stopkontak umum untuk menghubungkan aliran arus listrik selama batas waktu yang telah ditentukan.
Tidak terdeteksi keberadaan manusia oleh sensor *PIR*	Mengirim sinyal ke relai yang terhubung ke stopkontak umum untuk memutus aliran arus listrik.

Secara umum sistem terbagi menjadi 2 bagian, yakni aplikasi web dan sistem pada maket yang berisi mikrokontroller Arduino beserta sensor pendukung. Selain diproses oleh Arduino untuk pengambilan keputusan, beberapa data yang diterima dari sensor juga dikirimkan ke aplikasi web untuk ditampilkan dalam bentuk grafik. Berdasarkan hasil analisa di atas maka didapatkan kebutuhan fungsional yang ditampilkan pada Tabel 4.3 berikut.

Tabel 4.3 Kebutuhan Fungsional Sistem

No.	Kategori	Kebutuhan
1	Aplikasi Web	Mengambil dan menampilkan data monitoring penggunaan energi dari basis data web server secara periodik
2	Aplikasi Web	Memiliki akses untuk dapat mengontrol sistem berdasarkan input yang diberikan oleh pengguna melalui dashboard
3	Aplikasi Web	Membedakan akses pengaturan sistem dan fitur berdasarkan *role*
4	Aplikasi Web	Melakukan penyimpanan data yang telah diterima dari Arduino ke dalam database server
5	Aplikasi Web	Menghubungkan web client dengan Arduino untuk mengontrol kelistrikan

6	Sistem Maket (Arduino)	Menghubungkan Arduino ke jaringan internet melalui Wi-fi
7	Sistem Maket (Arduino)	Mengirimkan data dari Arduino ke database web server menggunakan metode HTTP Request secara aman
8	Sistem Maket (Arduino)	Mengambil data dari sensor dan menampilkan di lcd display
9	Sistem Maket (Arduino)	Mengendalikan perangkat listrik untuk pengambilan keputusan berdasarkan data yang diambil dari sensor
10	Sistem Maket (Arduino)	Mengirimkan perintah ke *relay* untuk memutus aliran listrik
11	Sistem Maket (Kelistrikan)	Mensimulasikan kondisi ruang kelas sesuai dengan kondisi yang telah diatur oleh sistem

Selain kebutuhan fungsional, juga didapatkan hasil analisis kebutuhan non fungsional yang ditampilkan pada Tabel 4.4 berikut.

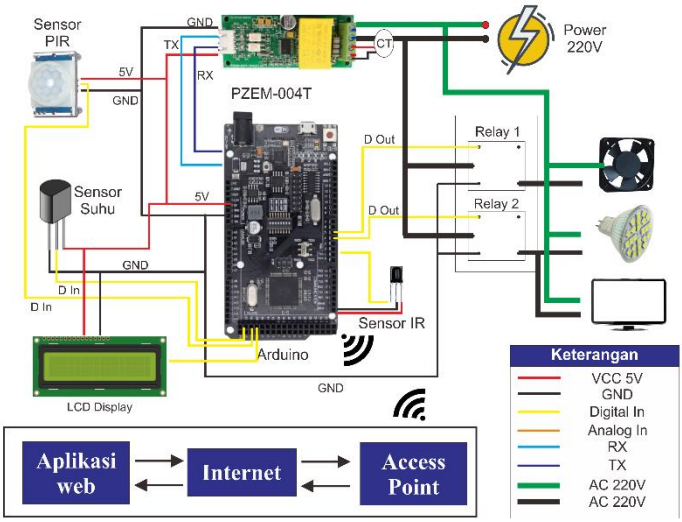
Tabel 4.4 Kebutuhan Non Fungsional

No.	Kategori	Kebutuhan
1	Aplikasi Web	Dapat menampilkan grafik dengan jelas dan *smooth*
2	Aplikasi Web	Sistem memiliki akses yang relatif cepat dan stabil serta tampilan yang mudah dipahami
3	Aplikasi Web	Menerapkan sistem otorisasi sebagai keamanan dalam transfer data.
4	Sistem Maket (Arduino)	Berjalan dengan baik dan stabil
5	Sistem Maket (Kelistrikan)	Berjalan dengan baik dan stabil

4.2 Desain Sistem

Desain sistem berikut ini akan digunakan sebagai panduan dalam melakukan pengembangan sistem pada tahap implementasi.

Gambar 4.1 berikut ini menjelaskan rancangan desain sistem secara keseluruhan. Mikrokontroler Arduino berada di tengah sebagai *processing unit* berfungsi untuk menerima dan mengolah data dari sensor. Sensor suhu berfungsi untuk mendapatkan data suhu pada ruang kelas, sensor PIR berfungsi untuk mendeteksi keberadaan manusia, sensor PZEM-004T berfungsi untuk mendapatkan informasi terkait listrik, sedangkan sensor *infrared* digunakan untuk input pengoperasian remote LCD. Sebagian data dikirim ke web server melalui modul ESP8266 *Wi-Fi* untuk keperluan *monitoring* dan juga sebagian data lainnya dikirim ke *relay* untuk mengirimkan sinyal *on/off* pada masing-masing perangkat elektronik sesuai dengan keadaan lingkungan dan setelan mode yang telah ditentukan.



Gambar 4.1 Detil Desain Arsitektur Sistem

Mikrokontroler Arduino juga mendapatkan data setelan mode dan *threshold* secara periodik dari *database web* server melalui modul ESP8266 dengan metode HTTP *Request*.

4.3 Desain Perangkat Keras

Desain perangkat keras dibuat dengan tujuan untuk menjadi acuan dalam membuat perangkat Arduino yang sesuai dengan kebutuhan yang diinginkan. Desain yang dibuat akan mampu melakukan pengambilan keputusan dalam memutus / menyambungkan arus listrik berdasarkan kondisi lingkungannya. Selain itu desain ini juga mampu menerima data energi listrik yang terpakai dan suhu ruangan agar selanjutnya dapat ditampilkan menjadi informasi yang berguna. Desain perangkat keras secara umum terbagi menjadi 2 bagian, yakni desain pada perangkat *low voltage* meliputi mikrokontroller Arduino beserta sensor pendukung, *relay* dan *LCD display*. Bagian selanjutnya terdapat pada rangkaian pada arus *AC* yang meliputi sambungan kelistrikan dari sumber tegangan PLN menuju model miniatur kelas.

4.3.1 Desain Perangkat Mikrokontroler

Terdapat beberapa komponen yang terhubung dengan mikrokontroller sebagai data masukan. Penjelasan detil komponen yang digunakan terdapat pada Tabel 4.5 berikut.

Tabel 4.5 Daftar Komponen Mikrokontroler

No	Nama Komponen	Fungsi
1	Arduino Mega2560 + Wifi	Digunakan sebagai *processing unit* untuk mengolah data masukan dari masing-masing sensor yang selanjutnya dapat memberikan tindakan sesuai kondisi lingkungan. Perangkat ini juga berfungsi untuk

		mengirim dan menerima data dengan web.
2	Sensor PZEM-004T V3	Modul sensor ini digunakan untuk menghimpun detil data kelistrikan yang mengalir menuju masing-masing ruangan. Sensor ini memiliki kemampuan untuk mendapatkan data tegangan, arus listrik, daya, frekuensi, *power factor*, dan total energi listrik yang digunakan.
3	Sensor Suhu DS18B20	Sensor ini berfungsi untuk mendapatkan data suhu dari masing-masing ruangan sebagai input untuk menentukan pengoperasian pendingin ruangan pada mode adaptif. Data suhu ini juga dikirimkan ke web untuk ditampilkan menjadi grafik.
4	Sensor PIR HC-SR501	Sensor PIR digunakan untuk menentukan keberadaan manusia di dalam kelas dengan mendeteksi adanya gerakan. Data dari sensor ini akan dikirimkan ke Arduino sebagai sinyal untuk men-*trigger relay* aktif atau tidak.
5	Sensor I*R*	Sensor IR (*Infrared*) digunakan untuk menerima gelombang sinyal inframerah dari *remote* dalam pengoperasian layar LCD.
6	LCD Display	Layar LCD digunakan untuk menampilkan data langsung dari Arduino sebelum dikirimkan ke web.

7	Modul RTC	Modul RTC digunakan untuk mendapatkan waktu. Dalam hal ini digunakan ketika terdeteksi pegantian bulan, maka Arduino akan me-*reset* data penggunaan energi listrik.

Komponen pendukung mikrokontroller pada Tabel 4.5 di atas terhubung langsung pada pin digital Arduino dan serial (TX, RX). Tabel 4.6 berikut adalah daftar hubungan pin pada sensor menuju Arduino.

Tabel 4.6 Daftar Pin Sensor

No	Komponen Sensor	Pin Arduino
1	PIR kelas SI 4101	32, 33, Vcc, Gnd
2	PIR kelas SI 4102	30, 31, Vcc, Gnd
3	PIR kelas SI 3101	28, 29, Vcc, Gnd
4	PIR kelas SI 3102	26, 27, Vcc, Gnd
5	PIR kelas SI 1101	24, 25, Vcc, Gnd
6	PIR kelas SI 1102	22, 23, Vcc, Gnd
7	PZEM-004T kelas SI 4101	Serial 2, Vcc, Gnd
8	PZEM-004T kelas SI 4102	10, 11, Vcc, Gnd
9	PZEM-004T kelas SI 3101	12, 13, Vcc, Gnd
10	PZEM-004T kelas SI 3102	50, 51, Vcc, Gnd
11	PZEM-004T kelas SI 1101	52, 53, Vcc, Gnd
12	PZEM-004T kelas SI 1102	62, 63, Vcc, Gnd
13	DS18B20 (semua)	2, Vcc, Gnd
14	Infrared	9, Vcc, Gnd

Sensor PZEM-004T membutuhkan jalur komunikasi serial untuk transfer datanya. Namun, karena jumlah jalur serial pada board mikrokontroller terbatas maka dalam hal ini digunakan fasilitas *library Software Serial* yang memungkinkan menggunakan pin digital sebagai *serial*. Selain sensor,

komponen yang digunakan terdapat modul RTC, LCD, dan *relay*. Tabel 4.7 berikut adalah daftar pin yang digunakan.

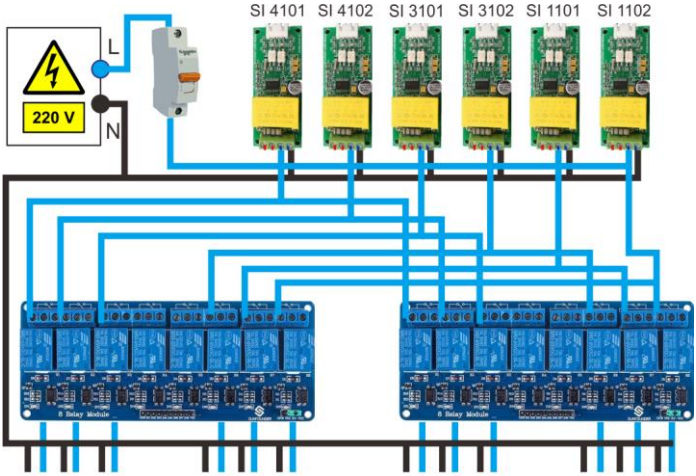
Tabel 4.7 Daftar Pin Relay, RTC, dan LCD

No	Komponen	Pin
1	Relay kelas SI 4101	47, 3, Vcc, Gnd
2	Relay kelas SI 4102	46, 4, Vcc, Gnd
3	Relay kelas SI 3101	45, 5, Vcc, Gnd
4	Relay kelas SI 3102	44, 6, Vcc, Gnd
5	Relay kelas SI 1101	43, 7, Vcc, Gnd
6	Relay kelas SI 1102	42, 8, Vcc, Gnd
7	LCD	Scl, Sda, Vcc, Gnd
8	Modul RTC	Scl, Sda, Vcc, Gnd

Relay yang digunakan pada sistem ini menganut aturan *Normally Open* (NO), yakni arus listrik akan terhubung ketika pin data *relay* mendapatkan sinyal *HIGH*.

4.3.2 Desain Rangkaian Listrik AC

Pada rangkaian listrik AC, listrik dari PLN akan masuk terlebih dahulu ke MCB yang bertujuan untuk keamanan. Terdapat 2



Gambar 4.2 Rangkaian Listrik AC

kabel utama pada arus listrik AC, yakni *Neutral* (N) dan *Line* (L). MCB akan memutus arus pada kabel L secara otomatis ketika terjadi konsleting listrik pada modul miniatur kelas. Selanjutnya kabel listrik AC akan dicabangkan dan masuk ke sensor PZEM-004T pada masing-masing kelas. Kabel listrik dari masing-masing sensor PZEM-004T akan dihubungkan ke *relay* umum dan *relay* pendingin ruangan masing-masing kelas. Detil rangkaian listrik AC ditunjukkan pada Gambar 4.2 berikut ini.

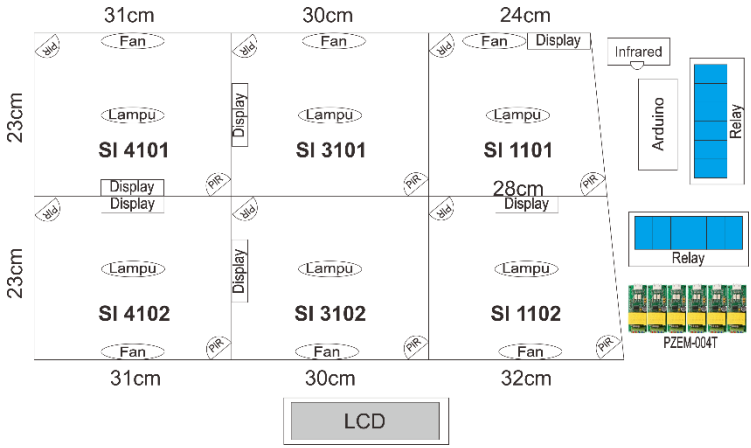
4.3.3 Desain Modul Miniatur Kelas

Modul miniatur kelas dibuat menggunakan bahan kayu yang mudah didapatkan. Ukuran masing-masing kelas pada maket adalah ukuran asli kelas dengan skala 1:30. Berikut adalah ukuran kelas sebelum dan sesudah dilakukan skala perbandingan.

Tabel 4.8 Ukuran Kelas DSI

Kelas	Ukuran Real	Ukuran Skala
SI 4101	7m x 9,3m	23cm x 31cm
SI 4102	7m x 9,3m	23cm x 31cm
SI 3101	7m x 9m	23cm x 30cm
SI 3102	7m x 9m	23cm x 30cm
SI 1101	7m x 8,4m	23cm x 28cm
SI 1102	7m x 9,6m	23cm x 32cm

Berdasarkan ukuran pada Tabel 4.8, selanjutnya dibuat desain modul miniatur kelas. Berikut adalah desain modul miniatur kelas yang ditunjukkan pada Gambar 4.3.



Gambar 4.3 Desain Modul Miniatur Kelas

4.3.4 Desain Program Mikrokontroler Arduino

Pengontrolan seluruh sensor dan komponen pendukung mikrokontroller pada sistem perlu dilakukan pemrograman agar dapat menjalankan fungsi sebagaimana mestinya. Bahasa pemrograman yang digunakan adalah *simplified c++* menggunakan Arduino IDE. Terdapat 2 *method* utama pada pemrograman Arduino, yakni *method setup()* yang hanya dijalankan sekali saja pada saat awal board Arduino dinyalakan dan *method loop()* yang dijalankan secara berulang-ulang selama Arduino aktif. Selain kedua *method* tersebut, kita bisa menambahkan *method* lain bila diperlukan, namun tetap berjalan di dalam *method* utama. Pendefinisian variabel global dan library dilakukan sebelum *void setup()*.

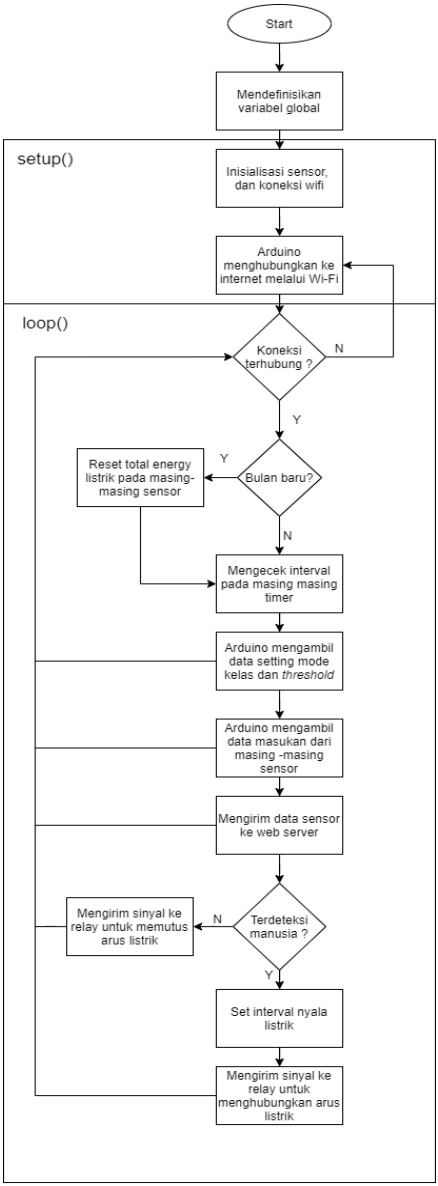
Pada sistem ini, variabel global yang dibutuhkan meliputi variabel yang menyimpan konfigurasi *Wi-Fi* seperti SSID dan *password*, variabel pin pada masing-masing komponen pendukung (*relay*, sensor, lcd, modul RTC), variabel untuk menyimpan data masukan dari sensor, dan variabel yang

digunakan untuk lama *timer* / durasi pada masing-masing fungsi pemrograman.

Pada *method setup()* dilakukan inisiasi *baud rate* yang digunakan pada masing-masing jalur serial. Kecepatan yang digunakan pada sistem ini adala 115200bps. Setelah inisiasi *baud rate*, dilakukan inisiasi jalur komunikasi *Wi-Fi* serial, inisiasi LCD, *Infrared*, sensor suhu DS18B20, dan modul RTC. Pengaturan mode pin digital pada *relay* dan sensor pir juga dilakukan di dalam *method setup()* ini, untuk sensor PIR digunakan mode INPUT sedangkan untuk *relay* digunakan mode OUTPUT. Pada saat ini, Arduino juga melakukan koneksi awal ke *Wi-Fi* tujuan serta memanggil beberapa *method* lain yang perlu dijalankan diawal. Setelah semua inisiasi dan konfigurasi pada *method setup()* selesai dilakukan, selanjutnya program akan berjalan di dalam *method loop()*.

Pada *method loop()* berisi perintah program berulang yang diawali dengan pengecekan interval milis() untuk timer pada masing-masing *method* yang telah ditentukan. Pengecekan tanggal oleh modul RTC juga dilakukan, apabila modul RTC sudah mendeteksi terjadinya pergantian bulan maka akan menjalankan method reset energy pada sensor PZEM-004T. Saat *method loop* berjalan, terdapat beberapa *method* lain yang dipanggil secara berulang, seperti method readDataListrik() untuk mendapatka data tentang kelistrikan, method getSuhu() untuk mendapatkan data suhu masing-masing kelas dan *method* yang digunakan untuk pengiriman data dan penerimaan data. Untuk pengiriman data dilakukan dengan memanggil *method postData()*. Pengiriman data sensor ke *web* dilakukan dengan menggunakan metode POST yang dilengkapi dengan key sebagai syarat untuk pengiriman data. Sedangkan untuk penerimaan data *setting* Arduino dilakukan dengan memanggil *method getData()* menggunakan metode GET. *Method postData()* dan *getData()* bertipe *bool*, bertujuan agar tidak

terjadi bentrok saat mengirim atau menerima data. Pengiriman dan penerimaan data dilakukan sesuai *interval* waktu yang telah ditentukan. Penerimaan data sensor dari sensor pir digunakan sebagai masukan untuk pengambilan keputusan dalam menghubungkan aliran listrik ke masing-masing kelas sesuai dengan mode yang dipilih. Ketika mode adaptif dan mode auto-cut diaktifkan maka *timer* listrik juga ikut aktif. Listrik akan mati otomatis saat tidak terdeteksi adanya aktifitas di dalam kelas dan saat waktu *interval* sudah terpenuhi. Selama method loop ini berjalan juga dilakukan pengecekan koneksi *Wi-Fi*, apabila koneksi *Wi-Fi* putus maka Arduino akan kembali melakukan sambungan ke *access point* yang telah ditentukan. *Flow chart* program Arduino ditunjukkan pada Gambar 4.4 berikut.



Gambar 4.4 Flow Chart Program Arduino

4.4 Desain Perangkat Lunak

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

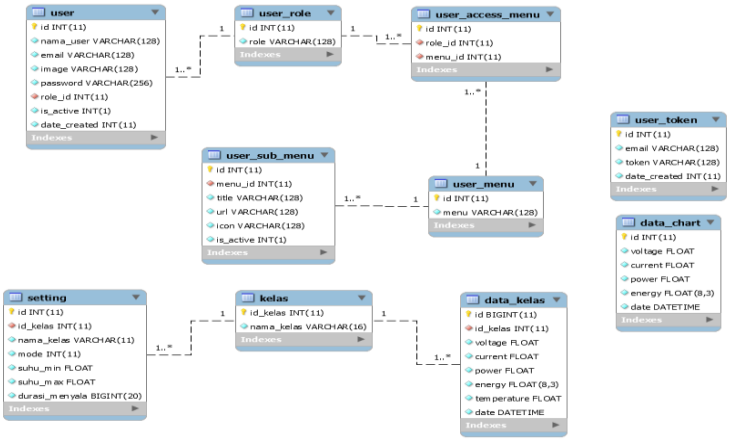
4.4.1 Desain Database

Aplikasi web yang akan dibuat harus memiliki kemampuan untuk dapat melakukan penyimpanan data. Untuk memenuhi kebutuhan tersebut maka dibutuhkan DBMS. Jenis *Database Management System* (*DBMS)* yang digunakan pada sistem ini adalah mySQL. Terdapat 10 tabel pada database yang digunakan yang dijelaskan pada tabel berikut ini

Tabel 4.9 Deskripsi Tabel Keseluruhan

No	Nama Tabel	Penjelasan
1	data_kelas	Menyimpan data sensor yang diterima dari Arduino meliputi data *voltage*, *current*, *power*, *energy*, *temperature*, dan waktu dari masing-masing kelas.
2	data_chart	Merupakan tabel yang berisi row dari hasil query pada tabel data_kelas. Berisi data akumulasi penggunaan listrik dari seluruh kelas.
3	kelas	Berisi informasi kelas, yang terdiri dari id_kelas dan nama_kelas
4	setting	Merupakan tabel yang menyimpan data setting kontrol dan *threshold* untuk Arduino. Meliputi id_kelas, nama_kelas, mode, suhu_min, suhu_max, dan durasi_menyala.
5	user	Berisi data *user* yang terdaftar pada aplikasi web. Meliputi

		nama_user, email, image, password, role_id, is_active, dan date_created
6	user_access_menu	Tabel yang berisi data aturan akses menu berdasarkan role_id.
7	user_menu	Tabel yang berisi data jenis menu yang tersimpan pada aplikasi web
8	user_role	Tabel yang berisi id dan role pengguna web
9	user_sub_menu	Tabel yang berisi data submenu yang tersimpan dan ditampilkan pada sidebar masing-masing role pada aplikasi web. Meliputi menu_id, title, url, icon, dan is_active
10	user_token	Tabel yang berisi token untuk masing-masing email yang digunakan dan berlaku pada saat pendaftaran.



Gambar 4.5 Desain Database

Gambar 4.5 menunjukkan desain database untuk aplikasi web. Detil deskripsi kolom pada masing-masing tabel di database dijelaskan pada Tabel 4.10 hingga Tabel 4.19 berikut ini.

Tabel 4.10 Deskripsi Tabel data_kelas

Tabel : data_kelas	
Nama Kolom	**Deskripsi**
id (PK)	*Primary key* tabel data kelas
id_kelas (FK)	id masing-masing kelas untuk data yang tersimpan
voltage	Angka besar tegangan listrik pada kelas
current	Angka total arus yang terpakai pada kelas
power	Angka total daya yang terpakai pada kelas
energy	Angka akumulasi penggunaan energi listrik di kelas
date	Tanggal waktu data ditambahkan

Tabel 4.11 Deskripsi Tabel data_chart

Tabel : data_chart	
Nama Kolom	**Deskripsi**
id (PK)	Primary key tabel data chart
voltage	Rata-rata angka voltage terbaru dari seluruh kelas
current	Angka total arus yang terpakai pada seluruh kelas
power	Angka total daya yang terpakai pada seluruh kelas
energy	Angka akumulasi penggunaan energi listrik di seluruh kelas
date	Tanggal waktu data ditambahkan

Tabel 4.12 Deskripsi Tabel kelas

Tabel : kelas	
Nama Kolom	**Deskripsi**
id_kelas (PK)	Kode *primary key* masing-masing kelas
nama_kelas	Nama kelas masing-masing id_kelas

Tabel 4.13 Deskripsi Tabel setting

Tabel : setting	
Nama Kolom	**Deskripsi**
id (PK)	Kode *primary key* masing-masing kelas
id_kelas (FK)	Id masing-masing kelas
nama_kelas	Nama kelas masing-masing id_kelas
mode	Mode kehematan yang digunakan
suhu_min	*Threshold* untuk suhu minimal
suhu_max	*Threshold* untuk suhu maksimal
durasi_menyala	Lama durasi timer untuk listrik yang menyala di setiap terdeteksi aktifitas di dalam kelas

Tabel 4.14 Deskripsi Tabel user

Tabel : user	
Nama Kolom	**Deskripsi**
id (PK)	*Primary key* untuk id tabel user
nama_user	Nama user yang terdaftar pada aplikasi web
email	Email user yang terdaftar
image	Foto yang ditampilkan pada aplikasi
password	*Password* akun user
role_id (FK)	Id *role* untuk mengidentifikasi jenis user
is_active	*Boolean* untuk set aktif/non-aktif akun
date_created	Tanggal *user* terdaftar

Tabel 4.15 Deskripsi Tabel user_access_menu

Tabel : user_access_menu	
Nama Kolom	**Deskripsi**
id (PK)	Id *Primary key* tabel user_access_menu
role_id (FK)	Kode *role* yang diatur aksesnya
menu_id (FK)	Kode menu yang diperbolehkan diakses oleh masing-masing jenis *role*

Tabel 4.16 Deskripsi Tabel user_menu

Tabel : user_menu	
Nama Kolom	**Deskripsi**
id (PK)	Id *Primary key* tabel user_menu
menu	Nama menu yang terdaftar

Tabel 4.17 Deskripsi Tabel user_role

Tabel : user_role	
Nama Kolom	**Deskripsi**
id (PK)	Id *Primary key* tabel user_menu
role	Nama *role* untuk menentukan akses akun

Tabel 4.18 Deskripsi Tabel user_sub_menu

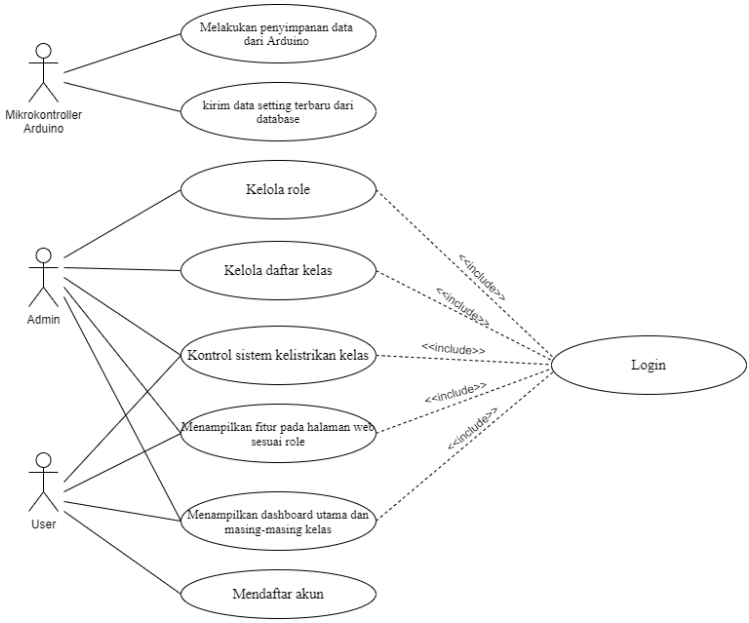
Tabel : user_sub_menu	
Nama Kolom	**Deskripsi**
id (PK)	Id *Primary key* tabel user_sub_menu
menu_id (FK)	Id menu untuk masing-masing submenu
title	Judul submenu
url	Alamat link akses submenu
icon	Jenis icon
is_active	*Boolean* untuk set aktif/non-aktif sub menu

Tabel 4.19 Deskripsi Tabel user_token

Tabel : user_token	
Nama Kolom	**Deskripsi**
id (PK)	Id *Primary key* token
email	Email masing-masing token
token	Kode token
date_created	Waktu token dibuat

4.4.2 Use Case Model Aplikasi Web

Perancangan *use case model* aplikasi web dibuat untuk menunjukkan fungsi-fungsi utama pada sistem sesuai dengan masing-masing pengguna. Gambar 4.6 di bawah ini menunjukkan *use case model* aplikasi web.



Gambar 4.6 Use Case Model Aplikasi Web

Dari *use case* model di atas akan dibuat *use case scenario* yang ditunjukkan pada Tabel 4.20 hingga Tabel 4.27 berikut ini.

Tabel 4.20 Use Case Scenario Melakukan Penyimpanan Data dari Arduino

***Use Case* : Melakukan Penyimpanan Data dari** Arduino	
Actor	Mikrokontroler Arduino
Description	Aktor mengirimkan data kelistrikan dan suhu masing-masing kelas yang selanjutnya untuk disimpan pada *database*
Pre-Condition	Aktor telah terhubung ke internet serta k*ey*, data kelistrikan dan suhu sudah ada
Basic Course	1. Aktor mengirimkan *request* HTTP POST ke web yang berisi data dan *key*

	2. Sistem melakukan validasi *key* dan kelengkapan data yang sudah lengkap 3. Sistem melakukan penyimpanan data ke dalam database 4. Sistem mengirimkan *response 200 OK*
Alternate Course	2a. Validasi key gagal dikarenakan tidak sesuai 2b. Data gagal dimasukkan karena *field* data kurang lengkap

Tabel 4.21 Use Case Scenario Kirim Data Setting Terbaru dari database

***Use Case* : Kirim Data Setting Terbaru dari Database**	
Actor	Mikrokontroler Arduino
Description	Aplikasi web mengirimkan data setting dari database ke Arduino
Pre-Condition	Aktor sudah terhubung ke internet
Basic Course	1. Aktor mengirimkan request HTTP Get ke web untuk mendapatkan data setting 2. Sistem mengirimkan data *setting* terbaru ke Arduino 3. Sistem mengirimkan *response 200 OK*
Alternate Course	2a. Sistem gagal mengirimkan data karena request salah

Tabel 4.22 Use Case Scenario Kelola Role

***Use Case* : Kelola Role**	
Actor	*Admin*
Description	Admin dapat mengelola role pada aplikasi web, seperti tambah, edit nama role, ubah hak akses, dan hapus
Pre-Condition	Aktor sudah login dengan menggunakan akun admin

Basic Course	1. Aktor mengakses halaman Kelola Role 2. Aktor melakukan klik tombol fitur pada halaman kelola *role* 3. Aktor melakukan perubahan pada role sesuai fungsi fiturnya
Alternate Course	-

Tabel 4.23 Use Case Scenario Kelola Daftar Kelas

***Use Case* : Kelola Daftar Kelas**	
Actor	*Admin*
Description	Admin dapat mengelola daftar kelas pada aplikasi web, seperti tambah, edit, dan hapus kelas
Pre-Condition	Aktor sudah login dengan menggunakan akun admin
Basic Course	1. Aktor mengakses halaman Kelola Kelas 2. Aktor melakukan klik tombol fitur pada halaman Kelola Kelas 3. Aktor melakukan perubahan pada daftar kelas sesuai fungsi fiturnya
Alternate Course	-

Tabel 4.24 Use Case Scenario Kontrol Sistem Kelistrikan Kelas

***Use Case* : Kontrol Sistem Kelistrikan Kelas**	
Actor	*Admin, User*
Description	Aktor dapat melakukan kontrol kelistrikan pada kelas seperti mengubah mode, *threshold* suhu, dan durasi menyala.
Pre-Condition	Aktor sudah login
Basic Course	1. Aktor mengakses halaman Kontrol Kelas.

	2. Aktor melakukan klik tombol edit pada kelas yang dipilih. 3. Aktor mengubah setting kelas pada form di halaman web. 4. Aktor melakukan klik tombol Edit. 5. Sistem memperbarui setting dan memberi notifikasi berhasil.
Alternate Course	5a. Sistem gagal memperbarui *setting* karena *User* salah melakukan perubahan dan memberi notifikasi gagal sesuai kesalahan input.

Tabel 4.25 Use Case Scenario Menampilkan Fitur pada Halaman Web Sesuai Role

***Use Case* : Menampilkan Fitur pada Halaman Web Sesuai Role**	
Actor	*Admin, User*
Description	Aktor dapat mengakses halaman web sesuai dengan *role*nya
Pre-Condition	Aktor sudah login
Basic Course	1. Sistem menampilkan fitur daftar halaman sesuai *role* pada *sidebar*. 2. Aktor mengakses halaman yang dipilih. 3. Sistem menampilkan halaman yang telah dipilih oleh Aktor.
Alternate Course	2a. Aktor mencoba mengakses halaman yang tidak ada pada daftar dari alamat langsung. 3a. Sistem memblokir akses dengan menampilkan halaman *403 forbidden.*

Tabel 4.26 Use Case Scenario Menampilkan Dashboard Utama dan Masing-Masing Kelas

***Use Case* : Menampilkan dashboard utama dan masing-masing kelas**	
Actor	*Admin, User*
Description	Aktor dapat melihat halaman dashboard utama dan halaman detil pada masing-masing kelas
Pre-Condition	Aktor sudah login
Basic Course	1. Aktor mengakses halaman dashboard utama / masing-masing kelas 2. Sistem menampilkan informasi dalam bentuk grafik dan tabel pada halaman
Alternate Course	-

Tabel 4.27 Use Case Scenario Mendaftar Akun

***Use Case* : Mendaftar Akun**	
Actor	*User*
Description	User dapat mendaftar akun dengan mengakses halaman registrasi
Pre-Condition	-
Basic Course	1. Aktor mengakses halaman registrasi 2. Aktor mengisi form pendaftaran 3. Aktor melakukan klik tombol “Register Account” 4. Admin melakukan aktivasi akun 5. Aktor sudah dapat menggunakan akun dengan melakukan login pada halaman login
Alternate Course	3a. Muncul notifikasi error karena kesalahan input

	4a. Admin belum melakukan aktivasi akun user, sehingga akun belum bisa digunakan

4.4.3 Desain Aplikasi Web

Aplikasi web dibangun dengan bahasa pemrograman PHP menggunakan *framework Codeigniter*. Penggunaan *framework* ini bertujuan untuk memudahkan pada saat pembuatan web dan juga dapat memaksimalkan fitur yang ada dengan fasilitas yang telah disediakan. *Codeigniter* memisahkan logika aplikasi dan tampilan ke dalam *Model*, *View*, dan *Controller*. Model berisi fungsi yang digunakan untuk mengambil, menyisipkan, dan memperbarui data pada *database*. *View* adalah sebuah informasi yang ditampilkan langsung ke pengguna, contohnya adalah laman web. Sedangkan *controller* adalah sebagai perantara antara *model*, *view*, dan *resource* lainnya yang terlibat dalam pembuatan halaman web

4.4.3.1 Detil Desain MVC Aplikasi Web

Berikut adalah detil dari masing-masing logika aplikasi *model*, *view* dan *controller* yang ditunjukkan dalam Tabel 4.28 hingga Tabel 4.30 di bawah ini.

Tabel 4.28 Detil Model Aplikasi Web

Nama Model	Deskripsi
M_data	Model yang digunakan untuk penanganan insert data dari Arduino ke dalam *database*
M_graph	Model yang digunakan untuk mengolah data dengan menjalankan query-query sesuai kebutuhan
M_userAdmin	Model yang digunakan untuk mengolah data dengan menjalankan query-query yang berhubungan dengan fitur masing-masing *user*

Tabel 4.29 Detil View Aplikasi Web

Nama *View*	Deskripsi
role	*View* yang digunakan untuk menampilkan halaman kelola *role*
role-access	*View* yang ditampilkan pada halaman *edit access role*
blocked	*View* yang digunakan untuk menampilkan halaman yang terblokir saat diakses oleh user
changepassword	*View* digunakan untuk menampilkan halaman ganti *password*
forgotpassword	*View* yang digunakan untuk menampilkan halaman lupa *password*
login	*View* yang digunakan untuk menampilkan halaman *login*
registration	*View* yang digunakan untuk menampilkan halaman registrasi akun
home	*View* yang digunakan untuk menampilkan halaman *dashboard* utama
kelas	*View* yang digunakan untuk menampilkan halaman *dashboard* masing-masing kelas
menu	*View* yang digunakan untuk menampilkan halaman kontrol kelas untuk mengubah *setting* pada Arduino
kelolakelas	*View* yang digunakan untuk menampilkan halaman kelola kelas
user	*View* yang digunakan untuk menampilkan halaman detil *user*
edit	*View* yang digunakan untuk menampilkan halaman *edit* data *user*

Tabel 4.30 Detil Controller Aplikasi Web

Nama Controller	Deskripsi
Admin	Berfungsi sebagai *controller* untuk menampilkan dan mengelola halaman web yang memiliki fitur untuk admin
Auth	Berfungsi sebagai *controller* untuk menampilkan dan mengelola halaman web yang berkaitan dengan autentikasi seperti login, registrasi, fitur reset password, dan pembatasan halaman sesuai role
Data	Berfungsi sebagai *controller* untuk menampilkan dan mengelola data pada aplikasi web.
Home	Berfungsi sebagai controller untuk menampilkan dan mengelola halaman dashboard utama pada aplikasi web
Kelas	Berfungsi sebagai *controller* untuk menampilkan dan mengelola halaman masing-masing kelas
Menu	Berfungsi sebagai *controller* untuk menampilkan dan mengelola halaman kontrol kelas
User	Berfungsi sebagai *controller* untuk menampilkan dan mengelola halaman yang berhubungan dengan user, seperti menampilkan halaman detil user, edit informasi user, dan ganti password

Dalam menjalankan masing-masing perannya pada sistem, baik *controller* maupun *model* memiliki *function*. Berikut adalah *function* yang ditambahkan pada tiap-tiap *controller* yang ditunjukkan pada Tabel 4.31 hingga Tabel 4.37.

Tabel 4.31 Function pada Controller Admin

***Controller :* Admin**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
kelolaKelas()	*Function* yang berfungsi untuk memanggil *view* halaman kelola kelas dan menampilkannya
tambahKelas()	*Function* yang berfungsi untuk menangani fitur tambah kelas
hapusKelas()	*Function* yang berfungsi untuk menangani fitur hapus kelas
editKelas()	*Function* yang berfungsi untuk menangani fitur edit kelas
kelolaRole()	*Function* yang berfugsi untuk memanggil *view* halaman kelola role dan menampilkannya
roleAccess()	*Function* yang berfungsi untuk memanggil *view role-access* dan menampilkan untuk spesifik id
changeAccess()	*Function* yang berfungsi untuk menangani fitur ubah akses menu oleh masing-masing *role*
hapusRole()	*Function* yang berfungsi untuk menangani fitur hapus *role*

editRole()	*Function* yang berfungsi untuk menangani fitur edit nama *role*
tambahRole()	*Function* yang berfungsi untuk menangani fitur tambah *role*

Tabel 4.32 Function pada Controller Auth

***Controller :* Auth**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
index()	*Function* yang berfungsi untuk memanggil *view login*
_login()	*Function* yang berfungsi untuk menangani fitur *login*
registration()	*Function* yang berfungsi untuk menangani fitur registrasi akun dan menampilkan halaman registrasi
verify()	*Function* yang berfungsi untuk menangani fitur verifikasi akun
_sendEmail()	*Function* yang berfungsi untuk menangani fitur mengirimkan email ke user untuk *reset password* dan ke admin untuk verifikasi
logout()	*Function* yang berfungsi untuk menangani *logout* akun
forgotPassword()	*Function* yang berfungsi untuk menangani fitur lupa

	password dan menampilkan halamannya
resetPassword()	*Function* yang berfungsi untuk melakukan *reset password* pada *form* yang telah diisi melalui *function changePassword()*
changePassword()	*Function* yang berfungsi untuk menampilkan halaman ganti *password*

Tabel 4.33 Function pada Controller Data

***Controller :* Data**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
getDataKelas()	*Function* yang berfungsi untuk mengambil data spesifik kelas dengan jumlah terbatas untuk ditampilkan pada *chart dashboard* masing-masing kelas
getDataKelasAll()	*Function* yang berfungsi untuk mengambil data spesifik kelas dengan jumlah keseluruhan untuk ditampilkan pada tabel kelas
getChartData()	*Function* yang berfungsi untuk mengambil data rangkuman penggunaan listrik untuk ditampilkan pada *chart dashboard* utama

getChartDataAll()	*Function* yang berfungsi untuk mengambil data rangkuman penggunaan listrik keseluruhan pada *dashboard* utama
getEnergy()	*Function* yang berfungsi untuk mengambil data total penggunaan energi listrik oleh masing-masing kelas
getEnergyByMonth()	*Function* yang berfungsi untuk mengambil data total penggunaan energi listrik keseluruhan tiap bulan
getLastDataChart()	*Function* yang berfungsi untuk mengambil data listrik terbaru yang ditampilkan pada *card* di halaman *dashboard* utama
getSetting()	*Function* yang berfungsi untuk mendapatkan data setting Arduino
insertData()	*Function* yang berfungsi untuk *insert* data penerimaan dari Arduino ke dalam tabel *database*

Tabel 4.34 Function pada Controller Home

***Controller* : Home**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
index()	*Function* yang berfungsi untuk menampilkan halaman dashboard utama

Tabel 4.35 Function pada Controller Kelas

***Controller :* Kelas**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
index()	*Function* yang berfungsi untuk menampilkan halaman masing-masing kelas

Tabel 4.36 Function pada Controller Menu

***Controller :* Menu**	
Nama *Function*	**Deskripsi**
__construct()	Berfungsi untuk memanggil *function default* yang harus dipenuhi oleh masing-masing *function* di dalam *controller* tersebut
index()	*Function* yang berfungsi untuk menampilkan halaman kontrol kelas
editSetting()	*Function* yang berfungsi untuk menangani fitur *edit setting* Arduino

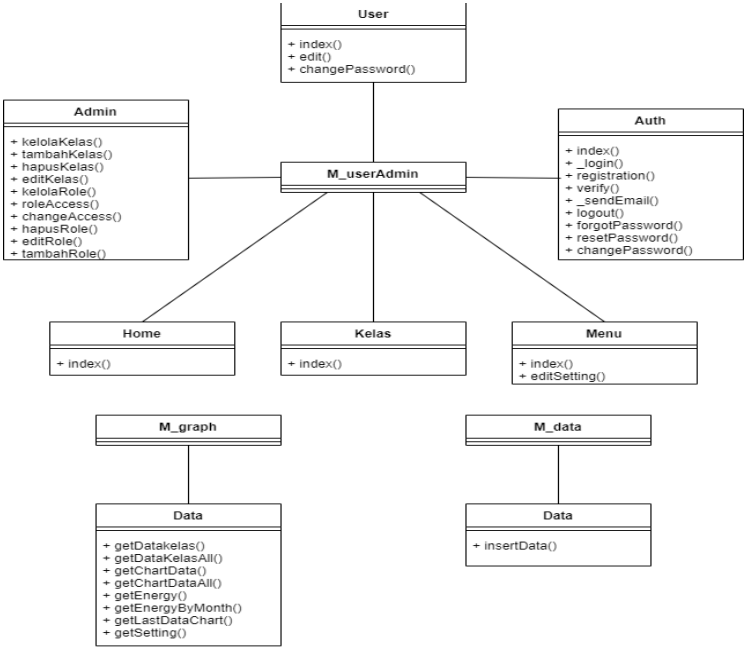
Tabel 4.37 Function pada Controller User

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

index()	*Function* yang berfungsi untuk menampilkan halaman detil user
edit()	*Function* yang berfungsi untuk menangani fitur edit data user
changePassword()	*Function* yang berfungsi untuk menangani fitur ubah password saat sudah login

4.4.3.2 *Class Diagram* Aplikasi Web

Setelah mendapatkan desain MVC untuk aplikasi web, berikutnya adalah menerjemahkan rancangan tersebut ke dalam bentuk *class diagram* untuk memudahkan proses pengembangan aplikasi. Berikut adalah *class diagram* aplikasi web yang ditunjukkan pada Gambar 4.7 di bawah ini.



Gambar 4.7 Class Diagram Aplikasi Web

4.4.4 Desain Pengujian Sistem

Pengujian sistem merupakan salah satu tahapan yang penting dalam pengembangan sistem. Hal ini dilakukan untuk mengetahui performa sistem yang dikembangkan. Sistem dilakukan pengujian dengan metode *black box test* dengan menyiapkan *test case*, masing-masing *test case* dilengkapi dengan hasil yang diharapkan (ekspektasi) sebagai syarat lulus uji. Pengujian sistem terbagi menjadi 2, yakni pengujian pada *hardware* dan *software*.

4.4.4.1 Desain Pengujian *Hardware*

Pengujian pada *hardware* dilakukan pada sistem mikrokontroler Arduino beserta komponen pendukungnya dan pengujian pada sistem kelistrikan. Berikut adalah daftar *test case* yang ditunjukkan pada Tabel 4.38.

Tabel 4.38 Test Case Hardware

Test case	Skenario	*Ekspektasi*
Koneksi WiFi Arduino	Menghubungkan mikrokontroller Arduino ke WiFi	Arduino dapat terhubung ke jaringan WiFi dan mendapatkan *IP Address*
Pembacaan sensor PIR	Terdapat objek yang bergerak di dalam kelas	Arduino dapat mendeteksi adanya gerakan dan memberikan sinyal ke *relay* sesuai setting
Pembacaan sensor PZEM-004T	Sensor PZEM-004T terhubung ke sambungan arus listrik AC masing-masing kelas	Hasil pembacaan sensor dapat ditampilkan pada layar LCD
Pembacaan sensor suhu DS18B20	Sensor suhu terpasang di	Hasil suhu masing-masing kelas dapat

	masing-masing kelas	ditampilkan di layar LCD
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 ke *web*	Mengirimkan data hasil pembacaan sensor PZEM-004T dan DS18B20 ke web menggunakan HTTP POST	Data berhasil terkirim, *respons code* 200 OK
Penerimaan data setting dari *web*	Menerima data setting dan *threshold* dari web menggunakan HTTP GET	Data berhasil diterima oleh Arduino, *response code* 200 OK, setting pada Arduino berubah sesuai data yang diterima
Respon *relay* dalam menerima *trigger* dari sensor PIR	Relay terhubung dengan Arduino dan sensor PIR, serta menggunakan mode Auto-cut / Adaptif	Relay dapat aktif dan non-aktif sesuai perintah dari Arduino
Tampilkan data ke LCD	Arduino mengirimkan data kelistrikan, suhu, dan mode untuk ditampilkan ke LCD	LCD dapat menampilkan data kelistrikan, suhu, dan mode

4.4.4.2 Desain Pengujian *Software*

Pengujian *software* dilakukan untuk mengetahui performa *aplikasi web* dari berbagai kemungkinan yang dapat terjadi pada fitur yang disediakan. Metode *black box testing* kembali

dipilih untuk menguji aplikasi *web* ini. Berikut adalah daftar *test case* yang ditunjukkan pada Tabel 4.39.

Tabel 4.39 Test Case Software

Test case	**Skenario**	***Ekspektasi***
Login	Pengguna memasukkan *email* dan *password* akun yang sudah aktif dengan benar	Pengguna dapat masuk ke halaman utama dan mengakses fitur sesuai *role*
	Pengguna memasukkan *email* dan *password* akun yang sudah aktif dengan salah	Pengguna tidak dapat masuk ke halaman utama, terdapat pemberitahuan salah *input email / password*
	Pengguna memasukkan *email* dan *password* akun yang belum diaktivasi	Pengguna tidak dapat masuk ke halaman utama, terdapat pemberitahuan akun belum diaktivasi
Registrasi akun	Pengguna memasukkan data yang diminta pada form dengan benar	Terdapat pemberitahuan akun berhasil dibuat
	Pengguna memasukkan data yang diminta pada form salah / kurang lengkap	Terdapat pemberitahuan *warning* pada form yang bermasalah
Penerimaan data dari Arduino	Arduino telah mengirimkan *HTTP request* untuk mengirimkan data dengan *key* dan parameter yang sesuai	Data tersimpan pada *database* web dan mengirimkan respon ke Arduino

	Arduino telah mengirimkan *HTTP request* untuk mengirimkan data dengan *key* atau parameter yang tidak sesuai	Data gagal tersimpan di *database* dan mengirimkan respon ke Arduino
Pengiriman data setting ke Arduino	Arduino telah mengirimkan *HTTP request* untuk meminta data *setting* dengan *parameter* yang sesuai	Web merespons dengan mengirimkan data *setting* ke Arduino
	Arduino telah mengirimkan *HTTP request* untuk meminta data setting dengan parameter yang tidak sesuai	Data gagal dikirim ke Arduino, web mengirimkan respons kesalahan
Kelola *role*	*Admin* telah *login* menggunakan akun *admin* dan mengakses fitur kelola *role*	*Admin* dapat melakukan modifikasi pada *role* yang telah terdaftar
Kelola daftar kelas	Admin telah *login* menggunakan akun *admin* dan mengakses fitur kelola daftar kelas	*Admin* dapat melakukan modifikasi pada daftar kelas
Kontrol sistem kelistrikan kelas	Admin / *user* telah *login* dan mengakses fitur kontrol kelas dengan memberikan *input*	Admin / user dapat melakukan modifikasi pada *setting* kelistrikan masing-masing kelas

	yang benar pada *form*	
	Admin / *user* telah login dan mengakses fitur kontrol kelas dengan memberikan input yang salah pada *form*	Web gagal memperbarui *setting* dan memberikan notifikasi kesalahan *input*
Tampilkan halaman fitur sesuai role	*Admin* telah login menggunakan akun *admin* dan mengakses fitur *admin*	Web menampilkan halaman fitur yang dituju dan memperbolehkan *admin* untuk memodifikasi
	User login menggunakan akun user untuk mengakses fitur user	Web menampilkan halaman fitur yang dituju dan memperbolehkan *user* untuk memodifikasi
	User selain *admin* mengakses halaman fitur yang dimiliki oleh admin langsung melalui alamat *link*	Web menampilkan halaman 403 *forbidden*, user dilarang mengakses halaman yang dituju.
Tampilkan halaman *dashboard* utama dan masing-masing kelas	*Admin* / *user* telah *login* dan mengakses halaman *dashboard* utama / *dashboard* masing-masing kelas	Web menambilkan halaman dashboard sesuai yang dituju
User mengakses	*User* mengakses halaman web yang	Web menampilkan halaman login

langsung halaman web tanpa *login*	telah diketahui sebelumnya langsung tanpa *login*	
Akses halaman *login* ketika sudah *login*	*User / admin* mengakses halaman login ketika sudah login	Web menampilkan halaman utama

BAB V
IMPLEMENTASI

Pada bagian ini akan menjelaskan mengenai proses implementasi terhadap sistem sesuai dengan desain yang sudah dirancang pada tahap sebelumnya.

5.1 Lingkungan Implementasi Sistem

Pengembangan program pada sistem *hardware* dan *software* menggunakan komputer dengan spesifikasi yang tertera pada Tabel 5.1.

Tabel 5.1 Spesifikasi Komputer Implementasi

Processor	Intel® Core™ i5-7200U CPU @2.5GHz
Memory (RAM)	8 GB
Sistem Operasi	Windows 10 Education 64Bit
Graphic Card	Nvidia GeForce 930MX

Pengembangan sistem pada sisi *hardware* Arduino dilakukan dengan menggunakan berbagai *tools* dan *library*. Tabel 5.2 berikut ini menjelaskan teknologi yang digunakan.

IDE	Arduino IDE v1.8.12
Bahasa Pemrograman	C/C++
Library Arduino	1. WifiEsp v2.2.2 2. ArduinoJson v6.15.2 3. SoftwareSerial 4. PZEM-004T v3.0 5. LiquidCrystal v1.0.7 6. IRremote 7. Wire 8. OneWire v2.3.5 9. DallasTemperature v3.8.0 10. DS3231 v1.0.1

Sedangkan pengembangan *software* aplikasi web juga dilakukan dengan menggunakan berbagai teknologi yang tersedia. Teknologi yang digunakan diantaranya adalah *web server*, *framework*, bahasa pemrograman, *database*, IDE, dan *library*. Penjabaran teknologi yang digunakan ditunjukkan pada Tabel 5.3.

Web server	Apache
Bahasa Pemrograman	1. PHP v7.4.6 2. HTML 3. SQL 4. Javascript
Framework	Codeigniter v3.1.11
Database	MySQL
IDE / Tools	1. Visual Studio Code v1.46.1 2. MySQL Workbench v8.0 3. phpMyAdmin 5.0.2
Library	AdminLTE3

5.2 Implementasi *Hardware*

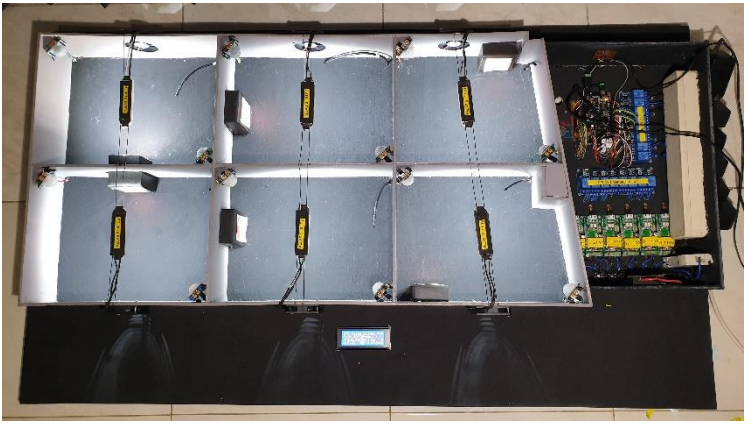
Implementasi *hardware* dilakukan sesuai dengan rancangan yang telah dibuat pada tahap desain. Adapun tahapannya dimulai dengan merangkai mikrokontroller Arduino beserta komponen pendukung seperti sensor PZEM-004T, sensor DS18B20, sensor *infrared*, sensor *PIR*, modul *RTC*, *LCD Display*, dan *relay*.

Arduino Mega 2560 dilengkapi dengan *pin output* 5v yang sangat terbatas, sehingga perlu ditambahkan *board extender* untuk mengakomodir kebutuhan *supply* listrik pada masing-masing sensor, sedangkan *supply* listrik Arduino didapatkan dari adaptor 12v yang ditancapkan pada *jack DC* yang sudah tersedia agar

lebih stabil. Listrik *DC* yang masuk ke Arduino akan langsung di *step down* oleh *board* untuk memenuhi syarat pengoperasian mikrokontroller.

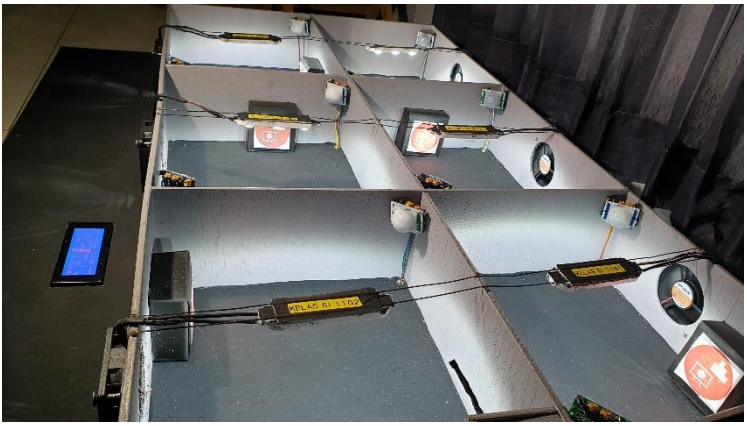
Setelah merangkai komponen untuk perangkat *low voltage*, selanjutnya dilakukan perangkaian jalur kabel kelistrikan AC ke *relay*, sensor PZEM-004T, dan diteruskan ke masing-masing kelas. Proses perangkaian *wiring* pada jalur kabel listrik AC harus dilakukan dengan ketelitian yang sangat tinggi karena potensi risiko yang cukup besar apabila terjadi kesalahan pada proses pemasangan kabel. *Wiring* kabel listrik AC dilakukan sesuai desain yang telah dibuat sebelumnya pada Gambar 4.2. Kabel listrik dari PLN akan masuk ke MCB terlebih dahulu sebagai pengaman apabila terjadinya korsleting listrik. Setelah itu, kabel netral dan fasa akan dibagi menjadi 6 bagian untuk dihubungkan ke sensor kelas masing-masing. Kabel fasa akan dihubungkan ke CT sensor yang sudah dalam 1 paket sensor PZEM-004T yang digunakan untuk pembacaan kelistrikan. Kabel dari masing-masing sensor akan dicabangkan menjadi 2 jalur. Jalur pertama terhubung ke *relay* umum yang digunakan untuk mengontrol *supply* listrik lampu dan *display*, sedangkan jalur kedua masuk ke *relay* pendingin ruangan untuk mengontrol kipas. Dibutuhkan masing-masing adaptor 12v 1A yang dihubungkan ke stopkontak untuk mengakomodir *supply* listrik pada kipas.

Setelah semua rangkaian pada Arduino dan rangkaian listrik AC sudah terpasang dan berjalan dengan baik, selanjutnya adalah melakukan pengembangan modul miniatur kelas sesuai desain yang telah dibuat sebelumnya. Seluruh perangkat akan dipasangkan pada modul miniatur kelas apabila sudah selesai. Berikut adalah hasil dari modul miniatur kelas yang sudah dibuat termasuk mikrokontroller Arduino dan komponen pendukungnya yang ditampilkan pada Gambar 5.1 hingga Gambar 5.3.

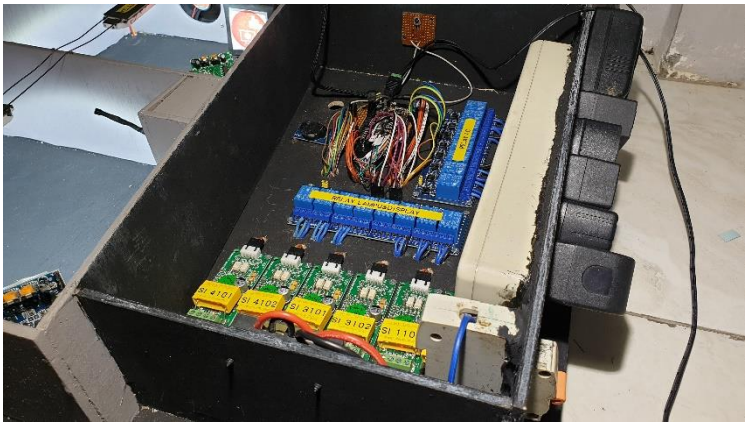


Gambar 5.1 Hasil Modul Miniatur Kelas

Peletakan Mikrokontroler Arduino beserta komponen pendukungnya diletakkan pada box di sebelah kanan secara terpisah dengan tujuan untuk mengantisipasi dari korsleting listrik.



Gambar 5.2 Detil Kelas pada Modul Miniatur Kelas



Gambar 5.3 Detil Box Hardware pada Modul Kelistrikan Kelas

5.3 Implementasi Program Mikrokontroler Arduino

Agar mikrokontroller Arduino dapat berfungsi menjalankan fiturnya, maka harus dilakukan pemrograman untuk melakukan pebacaan data dari masing-masing sensor dan melakukan pengambilan keputusan terhadap data yang diterima. Pembuatan kode program dan *upload* dilakukan menggunakan Arduino IDE. Berikut adalah kode program pada *mikrokontroller* Arduino.

```
#include "WiFiEsp.h"
#include <ArduinoJson.h>
#include <SoftwareSerial.h>
#include <PZEM004Tv30.h>
#include <LiquidCrystal_I2C.h>
#include <IRremote.h>
#include <Wire.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <DS3231.h>

#define si1101 2640478335
#define si1102 2640462015
#define si3101 2640494655
#define si3102 2640453855
#define si4101 2640486495
#define si4102 2640470175
```

```
18. #define ONE_WIRE_BUS 2
```

Kode 5.1 Include Library dan Define Sensor

Kode 5.1 di atas merupakan awal dari program yang dibuat, dimulai dengan melakukan *include library* yang akan digunakan dan melakukan definisi untuk kode *remote* serta pin sensor suhu.

```
19. char ssid[] = "SSID";
20. char pass[] = "password-Wi-Fi";
21. char server[] = "host-web";
22. String text = "";
23. String Respon = "";
24. bool responDariServer = false;
25. bool statusRequest = false;
26. bool debug = false;
27. long waktuMulai_get;
28. long waktuGet = 20000;
29. long waktuMulai_post;
30. long waktuPost = 120000;
31. long waktuMulai_Suhu;
32. long waktuUpdate_Suhu = 1000;
33.
34. //pin IR
35. int RECV_PIN = 9;
36.
37. // pin Relay
38. int R_SI4101 = 47; int RF_SI4101 = 3;
39. int R_SI4102 = 46; int RF_SI4102 = 4;
40. int R_SI3101 = 45; int RF_SI3101 = 5;
41. int R_SI3102 = 44; int RF_SI3102 = 6;
42. int R_SI1101 = 43; int RF_SI1101 = 7;
43. int R_SI1102 = 42; int RF_SI1102 = 8;
44.
45. //pin pir sensor
46. int P_SI4101_1 = 32; int P_SI4101_2 = 33;
47. int P_SI4102_1 = 30; int P_SI4102_2 = 31;
48. int P_SI3101_1 = 28; int P_SI3101_2 = 29;
49. int P_SI3102_1 = 26; int P_SI3102_2 = 27;
50. int P_SI1101_1 = 24; int P_SI1101_2 = 25;
51. int P_SI1102_1 = 22; int P_SI1102_2 = 23;
```

Kode 5.2 Deklarasi Variabe Global 1

Kode 5.2 merupakan potongan kode dalam melakukan deklarasi variabel global untuk menyimpan konfigurasi koneksi WiFi,

waktu timer GET dan POST, pin sensor *infrared*, pin *relay*, dan pin sensor *PIR*.

```
52. // variabel global
53. float power1; float power2; float power3;
54. float power4; float power5; float power6;
55. float energy1; float energy2; float energy3;
56. float energy4; float energy5; float energy6;
57. float current1; float current2; float current3;
58. float current4; float current5; float current6;
59. float voltage1; float voltage2; float voltage3;
60. float voltage4; float voltage5; float voltage6;
61. float temperature1; float temperature2;
62. float temperature3; float temperature4;
63. float temperature5; float temperature6;
64. int tempIdKelas; float tempVoltage;
65. float tempCurrent; float tempPower;
66. float tempEnergy; float tempTemperature;
67. boolean mulai = false;
68. boolean masih_panas = false;
69.
70. //variabel setting
71. int mode_si1101; int mode_si1102;
72. int mode_si3102; int mode_si3101;
73. int mode_si4101; int mode_si4102;
74. float suhuMin_si1101; float suhuMin_si1102;
75. float suhuMin_si3101; float suhuMin_si3102;
76. float suhuMin_si4101; float suhuMin_si4102;
77. float suhuMax_si1101; float suhuMax_si1102;
78. float suhuMax_si3101; float suhuMax_si3102;
79. float suhuMax_si4101; float suhuMax_si4102;
80. unsigned long durasi_si1101;
81. unsigned long durasi_si1102;
82. unsigned long durasi_si3101;
83. unsigned long durasi_si3102;
84. unsigned long durasi_si4101;
85. unsigned long durasi_si4102;
86. unsigned long waktuAwal_sil101=-durasi_si1101;
87. unsigned long waktuAwal_sil102 = -
      durasi_si1102;
88. unsigned long waktuAwal_si3101 = -
      durasi_si3101;
89. unsigned long waktuAwal_si3102 = -
      durasi_si3102;
```

```
90. unsigned long waktuAwal_si4101 = -
      durasi_si4101;
91. unsigned long waktuAwal_si4102 = -
      durasi_si4102;
```

Kode 5.3 Deklarasi Variabel Global 2

Kode 5.3 di atas merupakan potongan kode dalam deklarasi variabel global untuk keperluan pencatatan data listrik, variabel penampung untuk pengiriman data , variabel untuk menampung data setting yang diterima meliputi mode, *threshold* suhu, dan timer durasi.

```
92. //sensor pzem004t
93. PZEM004Tv30 pzem_SI4101(&Serial2);
94. PZEM004Tv30 pzem_SI4102(10, 11); //RX, TX
95. PZEM004Tv30 pzem_SI3101(12, 13);
96. PZEM004Tv30 pzem_SI3102(50, 51);
97. PZEM004Tv30 pzem_SI1101(52, 53);
98. PZEM004Tv30 pzem_SI1102(62, 63);
99.
100. //DS18b20
101. OneWire oneWire(ONE_WIRE_BUS);
102. DallasTemperature sensorSuhu(&oneWire);
103. DeviceAddress sensor1 = { 0x28, 0xF6, 0x51, 0x1
     6, 0xA8, 0x01, 0x3C, 0xBD };
104. DeviceAddress sensor2 = { 0x28, 0x70, 0x6F, 0x16, 0
     xA8, 0x01, 0x3C, 0xC3};
105. DeviceAddress sensor3 = { 0x28, 0xA2, 0x0D, 0x16, 0
     xA8, 0x01, 0x3C, 0xFB };
106. DeviceAddress sensor4 = { 0x28, 0x76, 0xF9, 0x16, 0
     xA8, 0x01, 0x3C, 0x5A };
107. DeviceAddress sensor5 = { 0x28, 0xA0, 0x49, 0x16, 0
     xA8, 0x01, 0x3C, 0x63 };
108. DeviceAddress sensor6 = { 0x28, 0xB0, 0x1E, 0x16, 0
     xA8, 0x01, 0x3C, 0xFC };
109.
110. //clock
111. DS3231  rtc(20, 21);
112. String lastMonth;
113.
114. //Wi-Fi client
115. WiFiEspClient client;
116. int status = WL_IDLE_STATUS;
```

```
117. int modeLcd;
```

Kode 5.4 Deklarasi Variabel Global 3

Kode 5.4 merupakan lanjutan pada deklarasi variabel global, meliputi pin sensor PZEM-004T, pin sensor suhu beserta alamat masing-masing sensor, pin modul RTC, *Client WiFi* dan mode layar LCD.

```
118. void setup(){
119. Serial.begin(115200);
120. Serial3.begin(115200);
121. WiFi.init(&Serial3);
122. lcd.begin();
123. irrecv.enableIRIn();
124. sensorSuhu.begin();
125.
126. //clock setting
127. rtc.begin();
128. //  rtc.setDate(19, 6, 2020);
129. //  rtc.setTime(11, 00, 00);
130. //  rtc.setDOW(5);     //set hari “Sabtu”
131. lastMonth = rtc.getMonthStr();
132.
133. if (WiFi.status() == WL_NO_SHIELD) {
134.    Serial.println(“WiFi shield not present”);
135.    // don’t continue
136.    while (true);
137. }
138. while ( status != WL_CONNECTED) {
139.   Serial.print(“
140.   Attempting to connect to WPA SSID: “);
141.   Serial.println(ssid);
142.   // Connect to WPA/WPA2 network
143.   status = WiFi.begin(ssid, pass);
144. }
145. Serial.println(“You’re connected to the network”);

146.
147. //pinmode relay
148. pinMode(R_SI4101, OUTPUT); pinMode(RF_SI4101, OUTPU
     T);
149. pinMode(R_SI4102, OUTPUT); pinMode(RF_SI4102, OUTPU
     T);
```

```
150. pinMode(R_SI3101, OUTPUT); pinMode(RF_SI3101, OUTPU
     T);
151. pinMode(R_SI3102, OUTPUT); pinMode(RF_SI3102, OUTPU
     T);
152. pinMode(R_SI1101, OUTPUT); pinMode(RF_SI1101, OUTPU
     T);
153. pinMode(R_SI1102, OUTPUT); pinMode(RF_SI1102, OUTPU
     T);
154. //pinmode pir sensor
155. pinMode(P_SI4101_1, INPUT); pinMode(P_SI4101_2, INP
     UT);
156. pinMode(P_SI4102_1, INPUT); pinMode(P_SI4102_2, INP
     UT);
157. pinMode(P_SI3101_1, INPUT); pinMode(P_SI3101_2, INP
     UT);
158. pinMode(P_SI3102_1, INPUT); pinMode(P_SI3102_2, INP
     UT);
159. pinMode(P_SI1101_1, INPUT); pinMode(P_SI1101_2, INP
     UT);
160. pinMode(P_SI1102_1, INPUT); pinMode(P_SI1102_2, INP
     UT);
161.
162. modeLcd = 0;
163. printWifiStatus();
164. waktuMulai_get = millis();
165. getSuhu();
166. }
```

Kode 5.5 Method setup() Arduino

Kode 5.5 di atas merupakan kode yang terdapat pada method void setup(). Method ini hanya dijalankan sekali saat Arduino dinyalakan. Pada kode ini yang dilakukan oleh program adalah dimulai dari inisiasi *baud rate* pada masing-masing serial yang digunakan dan mengaktifkan sensor. Pengaturan waktu pada modul *RTC* hanya perlu dilakukan sekali saja karena data akan tersimpan pada *chip* yang memiliki baterai CMOS. Selanjutnya menyambungkan Arduino ke WiFi, pengaturan pinmode untuk sensor *PIR* dan *relay*. Saat Arduino telah berhasil terhubung ke WiFi dilakukan set mode lcd ke 0, memulai timer dan mengambil data suhu diawal waktu.

```
167. void loop()
168. {
169. //reset energy saat ganti bulan
170. if (lastMonth != rtc.getMonthStr()) {
171.   pzem_SI4101.resetEnergy();
172.   pzem_SI4102.resetEnergy();
173.   pzem_SI3101.resetEnergy();
174.   pzem_SI3102.resetEnergy();
175.   pzem_SI1101.resetEnergy();
176.   pzem_SI1102.resetEnergy();
177.   lastMonth = rtc.getMonthStr();
178. }
179. if (irrecv.decode(&results)) {
180.  Serial.println(results.value);
181.   if (results.value == si1101) {
182.     modeLcd = 1;
183.   } else if (results.value == si1102) {
184.     modeLcd = 2;
185.   } else if (results.value == si3101) {
186.     modeLcd = 3;
187.   } else if (results.value == si3102) {
188.     modeLcd = 4;
189.   } else if (results.value == si4101) {
190.     modeLcd = 5;
191.   } else if (results.value == si4102) {
192.     modeLcd = 6;
193.   } else {
194.     Serial.println("Tombol Salah");
195.   }
196.   irrecv.resume();
197. }
198. printLcd();
199. readDataListrik();
```

Kode 5.6 Method loop() Awal Arduino

Kode 5.6 merupakan bagian program di dalam method *loop()* yang akan dibaca terus-menerus oleh Arduino. Diawali dengan pengecekan bulan, apabila terdeteksi pergantian bulan maka Arduino akan melakukan *reset* pada data *energy* yang tersimpan. Selanjutnya adalah pengecekan untuk penerimaan kode *remote* oleh sensor *infrared*, melakukan *print* pada *LCD* sesuai modenya dengan memanggil *method printLcd()* dan melakukan

pembacaan data listrik dengan memanggil *method readDataListrik()*.

```
200. if (millis() >= 30000) {
201.   mulai = true;
202. }
203.
204. if (waktuUpdate_Suhu < millis() - waktuMulai_Suhu)

205. {
206.   getSuhu();
207.   waktuMulai_Suhu = millis();
208. }
209. if (waktuPost < millis() - waktuMulai_post){
210. tempIdKelas = 1; tempVoltage = voltage1; tempCurren
     t = current1; tempPower = power1; tempEnergy = ener
     gy1; tempTemperature = temperature1;
211. statusRequest = postData(tempIdKelas, tempVoltage,
     tempCurrent, tempPower, tempEnergy, tempTemperature
     );
212. delay(1000);
213.
214. tempIdKelas = 2; tempVoltage = voltage2; tempCurren
     t = current2; tempPower = power2; tempEnergy = ener
     gy2; tempTemperature = temperature2;
215. statusRequest = postData(tempIdKelas, tempVoltage,
     tempCurrent, tempPower, tempEnergy, tempTemperature
     );
216. delay(1000);
217.
218. tempIdKelas = 3; tempVoltage = voltage3; tempCurren
     t = current3; tempPower = power3; tempEnergy = ener
     gy3; tempTemperature = temperature3;
219. statusRequest = postData(tempIdKelas, tempVoltage,
     tempCurrent, tempPower, tempEnergy, tempTemperature
     );
220. delay(1000);
221.
222. tempIdKelas = 4; tempVoltage = voltage4; tempCurren
     t = current4; tempPower = power4; tempEnergy = ener
     gy4; tempTemperature = temperature4;
223. statusRequest = postData(tempIdKelas, tempVoltage,
     tempCurrent, tempPower, tempEnergy, tempTemperature
     );
```

```
224. delay(1000);
225. tempIdKelas = 5; tempVoltage = voltage5; tempCurren
     t = current5; tempPower = power5; tempEnergy = ener
     gy5; tempTemperature = temperature5;
226.
227. statusRequest = postData(tempIdKelas, tempVoltage,
     tempCurrent, tempPower, tempEnergy, tempTemperature
     );
228. delay(1000);
229. tempIdKelas = 6; tempVoltage = voltage6; tempCurren
     t = current6; tempPower = power6; tempEnergy = ener
     gy6; tempTemperature = temperature6;
230.
231. statusRequest = postData(tempIdKelas, tempVoltage,
     tempCurrent, tempPower, tempEnergy, tempTemperature
     );
232. waktuMulai_post = millis();
233. }
234.
235. if (waktuGet < millis() - waktuMulai_get)
236. {
237.   statusRequest = getData();
238.   waktuMulai_get = millis();
239. }
```

Kode 5.7 Timer pada Method loop() Arduino

Kode 5.7 merupakan bagian kode program di dalam *method loop()* yang berfungsi untuk *timer* dalam menjalankan *method* lain. Diawali dengan *timer* untuk memulai sistem pada modul miniatur kelas, *timer* untuk *update* pembacaan data suhu, *timer* untuk pengiriman data ke *web*, dan *timer* untuk penerimaan data dari *web*.

```
240. if (statusRequest)
241. {
242.   while (client.available())
243.   {
244.       char c = client.read();
245.       Respon += c;
246.   }
247.
248. // if the server's disconnected, stop the client
```

```
249.  if (!client.connected()) {
250.   Serial.println("Disconnecting from server...");

251.    client.stop();
252.    statusRequest = false;
253.    responDariServer = true;
254.    }
255.  }
256.
257.  // penanganan data yang diretima dari server
258.  if (responDariServer)
259.  {
260.  responDariServer = false;
261.  int posisiData = Respon.indexOf("\r\n\r\n");
262.  String Data = Respon.substring(posisiData + 4);

263.  Data.trim();
264.
265.  if (Data.length() > 0) {
266.  const size_t capacity = JSON_ARRAY_SIZE(6) + * JS
      ON_OBJECT_SIZE(7) + 590;
267.  DynamicJsonDocument doc(capacity);
268.  deserializeJson(doc, Data);
269.
270.  JsonObject si_1101 = doc[0];
271.  mode_si1101 = si_1101["mode"];
272.  suhuMin_si1101 = si_1101["suhu_min"];
273.  suhuMax_si1101 = si_1101["suhu_max"];
274.  durasi_si1101 = si_1101["durasi_menyala"];
275.
276.  JsonObject si_1102 = doc[1];
277.  mode_si1102 = si_1102["mode"];
278.  suhuMin_si1102 = si_1102["suhu_min"];
279.  suhuMax_si1102 = si_1102["suhu_max"];
280.  durasi_si1102 = si_1102["durasi_menyala"];
281.
282.  JsonObject si_3101 = doc[2];
283.  mode_si3101 = si_3101["mode"];
284.  suhuMin_si3101 = si_3101["suhu_min"];
285.  suhuMax_si3101 = si_3101["suhu_max"];
286.  durasi_si3101 = si_3101["durasi_menyala"];
287.
288.  JsonObject si_3102 = doc[3];
289.  mode_si3102 = si_3102["mode"];
290.  suhuMin_si3102 = si_3102["suhu_min"];
```

```
291.   suhuMax_si3102 = si_3102["suhu_max"];
292.   durasi_si3102 = si_3102["durasi_menyala"];
293.
294.   JsonObject si_4101 = doc[4];
295.   mode_si4101 = si_4101["mode"];
296.   suhuMin_si4101 = si_4101["suhu_min"];
297.   suhuMax_si4101 = si_4101["suhu_max"];
298.   durasi_si4101 = si_4101["durasi_menyala"];
299.
300.   JsonObject si_4102 = doc[5];
301.   mode_si4102 = si_4102["mode"];
302.   suhuMin_si4102 = si_4102["suhu_min"];
303.   suhuMax_si4102 = si_4102["suhu_max"];
304.   durasi_si4102 = si_4102["durasi_menyala"];
305.   }
306.   Respon = "";
307. }
```

Kode 5.8 Penanganan Data pada Method loop() Arduino

Kode 5.8 memperlihatkan penanganan respon yang diterima dari web. Ketika *statusRequest* bernilai *true* maka dilakukan pemutusan koneksi dengan method *client.stop()*. Setelah itu dilakukan pembacaan datanya dengan mengidentifikasi apakah *request* yang dilakukan merupakan respon pengiriman data dari web atau respon balik atas suksesnya pengiriman data dari Arduino. Ketika respon berisi data *setting*, maka selanjutnya dilakukan pemilihan data tanpa *header*. Data tersebut akan diekstrak menggunakan *library* Arduino*Json* untuk memperbarui nilai variabel yang berhubungan dengan *setting* di masing-masing kelas.

```
308. if (mulai)
309. {
310.   // mode si 1101
311.   if (mode_si1101 == 0) {
312.     if ((millis() - waktuAwal_sil101) < durasi_si11
     01) {
313.       digitalWrite(R_SI1101, HIGH);
314.     } else {
315.       digitalWrite(R_SI1101, LOW);
316.     }
```

```
317.
318.     if (temperature1 > suhuMax_si1101) {
319.       digitalWrite(RF_SI1101, HIGH);
320.       masih_panas = true;
321.     } else if (temperature1 >= suhuMin_si1101
322.           && temperature1 <= suhuMax_si1101) {
323.       if (masih_panas) {
324.         digitalWrite(RF_SI1101, HIGH);
325.       } else
326.         digitalWrite(RF_SI1101, LOW);
327.     } else {
328.       digitalWrite(RF_SI1101, LOW);
329.       masih_panas = false;
330.     }
331.    if (digitalRead(P_SI1101_1) == HIGH ||
332.        digitalRead(P_SI1101_2) == HIGH) {
333.       waktuAwal_sil101 = millis();
334.     }
335.   } else if (mode_si1101 == 1) {
336.     if ((millis() - waktuAwal_sil101) <
337.           durasi_si1101) {
338.       digitalWrite(R_SI1101, HIGH);
339.       digitalWrite(RF_SI1101, HIGH);
340.       //Serial.println("Lampu 1101 nyala");
341.     } else {
342.       digitalWrite(R_SI1101, LOW);
343.       digitalWrite(RF_SI1101, LOW);
344.       //Serial.println("Lampu 1101 Mati");
345.     }
346.
347.     if (digitalRead(P_SI1101_1) == HIGH || digitalR
    ead(P_SI1101_2) == HIGH) {
348.      waktuAwal_sil101 = millis();
349.     }
350.   } else {
351.     digitalWrite(R_SI1101, HIGH);
352.     digitalWrite(RF_SI1101, HIGH);
353.   }
354.
355.  (...)
580.  }
581. }
```

Kode 5.9 Pengaturan Relay Sesuai Mode Listrik Kelas Arduino

Kode 5.9 memperlihatkan potongan program di dalam *method loop()* yang menangani logika menyalakan/mematikan *relay* sesuai dengan mode listrik yang digunakan pada masing-masing kelas. Pada nilai mode 0 berlaku untuk mode *adaptif*, nilai mode 1 berlaku untuk mode *auto-cut*, dan nilai mode 2 berlaku untuk mode manual. Mode ini akan berjalan setelah *boolean* mulai bernilai *true*. Penambahan *boolean* mulai ini dilakukan untuk memberikan waktu komponen sensor melakukan kalibrasi. Selanjutnya *relay* akan memutus arus listrik secara otomatis ketika mode *adaptif* atau *auto-cut* dinyalakan. Adapun *trigger* yang memicu *relay* nyala/mati didapat dari penerimaan data sensor *PIR* dalam pendeteksian keberadaan aktivitas di dalam kelas dan data suhu masing-masing kelas. Logika pemrograman yang tertera pada kode 5.9 berlaku untuk semua kelas, hanya saja terdapat perbedaan pada nama variabel yang digunakan. Kode tersebut merupakan akhir dari *method void loop()*. Berikut adalah *method-method* yang ditambahkan untuk mempermudah pembacaan logika kode program.

```
582. bool getData()
583. {
584.  Serial.println();
585.  Serial.println("Starting connection to server");
586.  if (client.connect(server, 80)) {
587.   Serial.println("Connected to server");
588.   // Make a HTTP request
589.   client.println(F("GET [alamathost]/data/getsetti
     ng HTTP/1.1"));
590.   client.println(F("Host: [alamathost]"));
591.   client.println("Connection: close");
592.   client.println();
593.
594. long _startMillis = millis();
595. while (!client.available() and (millis() - _startMi
     llis < 2000));
596.    return true;
597.   }
598.   return false;
599. }
```

Kode 5.10 Method getData() Arduino

Kode 5.10 di atas memperlihatkan kode program pada *method getData()*. *Method* ini bertipe *bool* yang digunakan untuk mengirimkan *request* GET ke *web* dengan respon balik yang berisi data *setting* Arduino terbaru dari *web*.

```
600. bool postData(int tempIdKelas, float tempVoltage, f
     loat tempCurrent, float tempPower, float tempEnerg
     y, float tempTemperature)
601. {
602.   String key = "12345678";
603.   String data;
604.   data = "key=" + key;
605.   data += "&id_kelas=";
606.   data += tempIdKelas;
607.   data += "&voltage=";
608.   data += tempVoltage;
609.   data += "&current=";
610.   data += tempCurrent;
611.   data += "&power=";
612.   data += tempPower;
613.   data += "&energy=";
614.   data += String(tempEnergy, DEC);
615.   data += "&temperature=";
616.   data += tempTemperature;
617.
618.   Serial.println();
619.   Serial.println("Starting connection to server..."
      );
620.   // if you get a connection, report back via seria
      l
621.   if (client.connect(server, 80)) {
622.     Serial.println("Connected to server");
623.     // Make a HTTP request
624.     client.println("POST [alamat-
      host]/Data/insertData HTTP/1.1");
625.
626.     // EDIT: 'Host' to match your domain
627.     client.println("Host: [alamat-host]");
628.     client.println("User-Agent: Arduino/1.0");
629.     client.println("Connection: Keep-Alive");
630.     client.println("Content-Type: application/x-
      www-form-urlencoded; charset=UTF-8");
```

```
631.     client.print("Content-Length: ");
632.     client.println(data.length());
633.     client.println();
634.     client.println(data);
635.     Serial.println("Sukses kirim data !!");
636.     client.stop();
637.
638.     return true;
639.   }
640.   return false;
641. }
```

Kode 5.11 Method postData() Arduino

Kode 5.11 merupakan penggalan kode program yang digunakan untuk melakukan *request* pengiriman data sensor ke *web*. Pengiriman dilakukan menggunakan HTTP POST agar lebih aman. Pengamanan juga ditingkatkan dengan adanya *key* sebagai syarat data dapat diterima oleh aplikasi *web*. Bila kode yang dikirimkan di Arduino dan di *web* cocok maka data akan berhasil masuk ke *database*, sedangkan bila kode tidak cocok maka data akan ditolak dan tidak dimasukkan ke *database*.

```
642. void readDataListrik() {
643. //pzem 1101
644.   voltage1 = pzem_SI1101.voltage();
645.   current1 = pzem_SI1101.current();
646.   power1 = pzem_SI1101.power();
647.   energy1 = pzem_SI1101.energy();
648. //pzem 1102
649.   voltage2 = pzem_SI1102.voltage();
650.   current2 = pzem_SI1102.current();
651.   power2 = pzem_SI1102.power();
652.   energy2 = pzem_SI1102.energy();
653. //pzem 3101
654.   voltage3 = pzem_SI3101.voltage();
655.   current3 = pzem_SI3101.current();
656.   power3 = pzem_SI3101.power();
657.   energy3 = pzem_SI3101.energy();
658. //pzem 3102
659.   voltage4 = pzem_SI3102.voltage();
660.   current4 = pzem_SI3102.current();
661.   power4 = pzem_SI3102.power();
```

```
662.   energy4 = pzem_SI3102.energy();
663. //pzem 4101
664.   voltage5 = pzem_SI4101.voltage();
665.   current5 = pzem_SI4101.current();
666.   power5 = pzem_SI4101.power();
667.   energy5 = pzem_SI4101.energy();
668. //pzem 4102
669.   voltage6 = pzem_SI4102.voltage();
670.   current6 = pzem_SI4102.current();
671.   power6 = pzem_SI4102.power();
672.   energy6 = pzem_SI4102.energy();
673. }
```

Kode 5.12 Method readDataListrik() Arduino

Kode 5.12 di atas merupakan kode program yang digunakan untuk melakukan pembacaan data kelistrikan dan disimpan ke dalam variabel masing-masing kelas.

```
674. void printLcd() {
675.   switch (modeLcd) {
676.     case 0:
677.       lcd.clear();
678.       lcd.setCursor(0, 0);
679.       lcd.print("     SMART POWER    ");
680.       lcd.setCursor(1, 2);
681.       lcd.print(rtc.getDOWStr());
682.       lcd.setCursor(7, 2);
683.       lcd.print(",");
684.       lcd.setCursor(9, 2);
685.       lcd.print(rtc.getDateStr());
686.       lcd.setCursor(6, 3);
687.       lcd.print(rtc.getTimeStr());
688.       break;
689.
690.     case 1:
691.       lcd.clear();
692.       lcd.setCursor(0, 0);
693.       lcd.print("=== Kelas SI1101 ===");
694.       lcd.setCursor(0, 1);
695.       lcd.print("T:");
696.       lcd.setCursor(2, 1);
697.       lcd.print(temperature1);
698.       lcd.setCursor(7, 1);
699.       lcd.print((char) 223);
```

```
700.       lcd.setCursor(8, 1);
701.       lcd.print("C");
702.       lcd.setCursor(9, 1);
703.       lcd.print((char) 255);
704.       lcd.setCursor(10, 1);
705.       lcd.print("Mode:");
706.       lcd.setCursor(16, 1);
707.       lcd.print(mode_si1101);
708.       lcd.setCursor(0, 2);
709.       lcd.print("V:");
710.       lcd.setCursor(2, 2);
711.       lcd.print(voltage1);
712.       lcd.setCursor(8, 2);
713.       lcd.print("V");
714.       lcd.setCursor(9, 2);
715.       lcd.print((char) 255);
716.       lcd.setCursor(10, 2);
717.       lcd.print("W:");
718.       lcd.setCursor(12, 2);
719.       lcd.print(power1);
720.       lcd.setCursor(19, 2);
721.       lcd.print("W");
722.       lcd.setCursor(0, 3);
723.       lcd.print("A:");
724.       lcd.setCursor(2, 3);
725.       lcd.print(current1);
726.       lcd.setCursor(8, 3);
727.       lcd.print("A");
728.       lcd.setCursor(9, 3);
729.       lcd.print((char) 255);
730.       lcd.setCursor(10, 3);
731.       lcd.print("E:");
732.       lcd.setCursor(12, 3);
733.       lcd.print(energy1, 3);
734.       lcd.setCursor(17, 3);
735.       lcd.print("kWh");
736.       break;
737.
738. (...)
739.
930.     case 6:
931.       lcd.clear();
932.       lcd.setCursor(0, 0);
933.       lcd.print("=== Kelas SI4102 ===");
934.       lcd.setCursor(0, 1);
```

```
935.        lcd.print("T:");
936.        lcd.setCursor(2, 1);
937.        lcd.print(temperature6);
938.        lcd.setCursor(7, 1);
939.        lcd.print((char) 223);
940.        lcd.setCursor(8, 1);
941.        lcd.print("C");
942.        lcd.setCursor(9, 1);
943.        lcd.print((char) 255);
944.        lcd.setCursor(10, 1);
945.        lcd.print("Mode:");
946.        lcd.setCursor(16, 1);
947.        lcd.print(mode_si4102);
948.        lcd.setCursor(0, 2);
949.        lcd.print("V:");
950.        lcd.setCursor(2, 2);
951.        lcd.print(voltage6);
952.        lcd.setCursor(8, 2);
953.        lcd.print("V");
954.        lcd.setCursor(9, 2);
955.        lcd.print((char) 255);
956.        lcd.setCursor(10, 2);
957.        lcd.print("W:");
958.        lcd.setCursor(12, 2);
959.        lcd.print(power6);
960.        lcd.setCursor(19, 2);
961.        lcd.print("W");
962.        lcd.setCursor(0, 3);
963.        lcd.print("A:");
964.        lcd.setCursor(2, 3);
965.        lcd.print(current6);
966.        lcd.setCursor(8, 3);
967.        lcd.print("A");
968.        lcd.setCursor(9, 3);
969.        lcd.print((char) 255);
970.        lcd.setCursor(10, 3);
971.        lcd.print("E:");
972.        lcd.setCursor(12, 3);
973.        lcd.print(energy6, 3);
974.        lcd.setCursor(17, 3);
975.        lcd.print("kWh");
976.        break;
977.
978.      default:
979.        //lcd.clear();
```

```
980.        break;
981.    }
982. }
```

Kode 5.13 Method printLcd() Arduino

Kode 5.13 merupakan kode program untuk menampilkan data bacaan sensor dan mode ke LCD. Terdapat 7 *case* pada *method* ini dengan logika pemrograman yang sama, yakni *case* 1 – 6 yang membedakan hanyalah nama variabel data pada masing-masing kelas yang akan ditampilkan.

```
983.  void getSuhu() {
984.    sensorSuhu.requestTemperatures();
985.    temperature1 = sensorSuhu.getTempC(sensor1);
986.    temperature2 = sensorSuhu.getTempC(sensor2);
987.    temperature3 = sensorSuhu.getTempC(sensor3);
988.    temperature4 = sensorSuhu.getTempC(sensor4);
989.    temperature5 = sensorSuhu.getTempC(sensor5);
990.    temperature6 = sensorSuhu.getTempC(sensor6);
991.  }
```

Kode 5.14 Method getSuhu() Arduino

Kode 5.14 memperlihatkan *method getSuhu()* yang digunakan untuk mengambil data suhu terbaru pada masing-masing kelas. Adapun alamat masing-masing sensor sudah dideklarasikan di awal.

```
992. void printWifiStatus()
993. {
994.   // print the SSID of the network you're attached
       to
995.   Serial.print("SSID: ");
996.   Serial.println(WiFi.SSID());
997.
998.   // print your WiFi shield's IP address
999.   IPAddress ip = WiFi.localIP();
1000.    Serial.print("IP Address: ");
1001.    Serial.println(ip);
1002.
1003.    // print the received signal strength
1004.    long rssi = WiFi.RSSI();
```

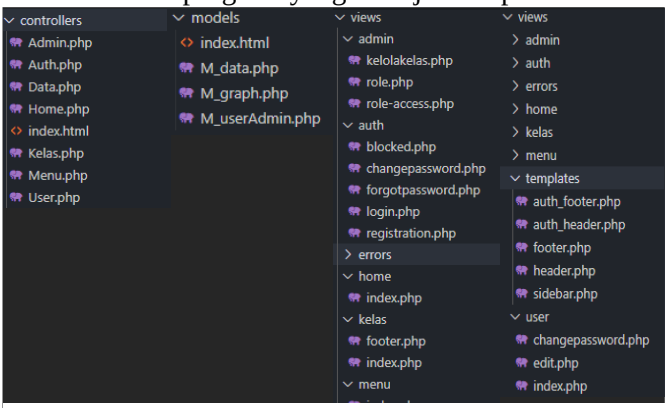
```
1005.   Serial.print("Signal strength (RSSI):");
1006.   Serial.print(rssi);
1007.   Serial.println(" dBm");
1008.
1009.   IPAddress gateway = WiFi.gatewayIP();
1010.   Serial.print("gateway:");
1011.   Serial.print(gateway);
1012.   Serial.println(" ");
1013. }
```

Kode 5.15 Method printWifiStatus() Arduino

Kode 5.15 merupakan isi program pada *method printWifiStatus()* yang digunakan untuk menampilkan informasi koneksi *WiFi* pada *serial monitor*. Informasi yang ditampilkan meliputi nama *SSID*, *IP Address*, *RSSI* (kekuatan sinyal), dan *gateway*.

5.4 Implementasi *Software*

Implementasi sistem pada sisi *software* dilakukan dengan pembuatan aplikasi *web* sesuai dengan desain yang telah dirancang pada bab sebelumnya. Berikut adalah *file* hasil implementasi kode program yang ditunjukkan pada Gambar 5.4.



Gambar 5.4 Hasil Implementasi Aplikasi Web

5.4.1 Implementasi Model Aplikasi *Web*

Model merupakan bagian kode program yang digunakan untuk melakukan pengolahan data. Terdapat 3 model yang digunakan

pada sistem ini. Kode 5.16 berikut adalah potongan kode program untuk masing-masing model.

```
1. <?php
2. class M_data extends CI_Model
3. {
4.   public function insert($tabel, $data)
5.   {
6.     $this->db->insert($tabel, $data);
7.     return TRUE;
8.   }
9.
10.   public function insert_chart()
11.   {
12.    $sql = "SELECT  SUBSTRING(AVG(voltage), 1, 3)
13.            voltage,SUBSTRING(SUM(current), 1, 5)
14.            current, SUBSTRING(SUM(power), 1, 5)
15.            power, SUBSTRING(SUM(energy), 1, 5)
16.            energy,MAX(date) date
17.            FROM (SELECT m.*, ROW_NUMBER()
18.            OVER (PARTITION BY id_kelas
19.            ORDER BY id DESC) AS rn
20.            FROM data_kelas AS m)
21.            SUB WHERE rn=1";
22.    $query = $this->db->query($sql);
23.    return $query->row();
24.   }
25. }
```

Kode 5.16 Model M_data

Kode 5.16 memperlihatkan kode program yang terdapat di dalam *model* M_data. Terdapat 2 *function*, yakni *function insert()* yang digunakan untuk menangani *insert* data dari *controller* data dan *function insert_chart()* yang digunakan untuk melakukan *insert* data akumulasi dari Arduino ke tabel *data_chart*.

```
1. <?php
2. class M_graph extends CI_Model
3. {
4.  public function __construct()
5.  {
```

```
6.    $this->load->database();
7.   }
8.
9.   public function selectDailyKelasData($id_kelas)
10.  {
11.    $sql = "SELECT * FROM (SELECT * FROM data_kelas
12.            WHERE id_kelas=$id_kelas ORDER BY id
13.            DESC LIMIT 300)
14.            SUB ORDER BY id ASC";
15.    $query = $this->db->query($sql);
16.    return $query->result();
17.   }
18.   public function selectKelasData($id_kelas)
19.   {
20.    $sql = "SELECT * FROM data_kelas
21.            WHERE id_kelas=$id_kelas
22.            ORDER BY id DESC";
23.    $query = $this->db->query($sql);
24.    return $query->result();
25.   }
26.
27.   public function selectDailyChartData()
28.   {
29.    $sql = "SELECT * FROM (SELECT * FROM
30.           `data_chart` ORDER BY id DESC LIMIT 300)
31.            SUB ORDER BY id ASC";
32.    $query = $this->db->query($sql);
33.    return $query->result();
34.   }
35.
36.   public function selectChartDataAll()
37.   {
38.    $sql = "SELECT * FROM `data_chart`
39.            ORDER BY id DESC";
40.    $query = $this->db->query($sql);
41.    return $query->result();
42.   }
43.
44.   public function selectTotalEnergy()
45.   {
46.    $sql = "SELECT data_kelas.energy,
47.            kelas.nama_kelas AS nama_kelas
48.            FROM `data_kelas` INNER JOIN kelas
49.            ON kelas.id_kelas=data_kelas.id_kelas
50.            ORDER BY id DESC LIMIT 6";
```

```
51.    $query = $this->db->query($sql);
52.    return $query->result();
53.   }
54.
55.   public function selectLastData()
56.   {
57.    $sql = "SELECT * FROM (SELECT * FROM
58.            `data_chart` ORDER BY id DESC LIMIT 1)
59.            SUB ORDER BY id ASC";
60.    $query = $this->db->query($sql);
61.    return $query->result();
62.   }
63.
64.   public function selectSetting()
65.   {
66.    $sql = "SELECT * FROM setting
67.            ORDER BY id_kelas";
68.    $query = $this->db->query($sql);
69.    return $query->result();
70.   }
71.
72.   public function selectEnergyByMonth()
73.   {
74.    $sql = "SELECT id, MONTHNAME(date) as 'month',
75.            energy FROM data_chart
76.            WHERE id IN (SELECT MAX(id) FROM
77.            data_chart WHERE
78.            year(date) = year(now())
79.            GROUP BY MONTHNAME(date))";
80.    $query = $this->db->query($sql);
81.    return $query->result();
82.   }
83. }
```

Kode 5.17 Model M_graph

Kode 5.17 merupakan hasil implementasi dari *model M_graph*. *Model* ini secara umum berisi 9 *function* yang digunakan untuk menjalankan *query SQL*. *Function contstruct* berisi baris kode yang digunakan untuk memanggil *library database*. 8 *function* lain yang ditambahkan digunakan untuk mengambil data suhu dan kelistrikan kelas sesuai kebutuhan. Program di dalam *function* dieksekusi dimulai dari memasukkan *query SQL* sesuai

kebutuhan ke dalam variabel *$query*. Selanjutnya *query* tersebut akan dieksekusi dan menghasilkan data dalam bentuk *array*.

```
1.  <?php
2.  class M_userAdmin extends CI_Model
3.  {
4.   function get_where($get, $where)
5.   {
6.    $query = $this->db->get_where($get, $where);
7.    return $query->row_array();
8.   }
9.
10.  function get($get)
11.  {
12.   $query = $this->db->get($get);
13.   return $query->result_array();
14.  }
15.
16.  function insert($tabel, $data)
17.  {
18.   $this->db->insert($tabel, $data);
19.   return TRUE;
20.  }
21.
22.  function delete($tabel, $data)
23.  {
24.   $this->db->delete($tabel, $data);
25.   return TRUE;
26.  }
27.
28.  function update($tabel, $kolom, $id)
29.  {
30.   $this->db->update($tabel, $kolom, $id);
31.   return TRUE;
32.  }
33. }
```

Kode 5.18 Model M_userAdmin

Kode 5.18 adalah hasil implementasi *model M_userAdmin*. *Model* ini memiliki 5 *function* yang digunakan untuk memanipulasi data-data yang berhubungan dengan fitur pada *User* dan *Admin*, meliputi pengambilan data, menambahkan,

menghapus, dan melakukan *update* data. Fungsi akan memberi nilai balik dalam bentuk *array* maupun *boolean*.

5.4.2 Implementasi *Controller*

Controller merupakan komponen kode program aplikasi *web* yang digunakan untuk menghubungkan antara *view* dengan *model*. Pembuatan masing-masing *controller* dilakukan sesuai dengan desain yang telah dijelaskan pada bab sebelumnya. Aplikasi *web* pada sistem ini total memiliki 7 *controller* yang memiliki fungsinya masing-masing. Kode 5.19 berikut adalah kode program *controller* beserta penjelasannya masing-masing.

```
1. <?php
2. defined('BASEPATH') or exit('No direct script access
   allowed');
3.
4. class Admin extends CI_Controller
5. {
6.  public function __construct()
7.  {
8.  parent::__construct();
9.  is_logged_in();
10.  $this->load->model('M_userAdmin');
11.  }
12.
13.  public function kelolaKelas()
14.  {
15.   $data['title'] = 'Kelola Kelas';
16.   $data['user'] = $this->M_userAdmin->
17.     get_where('user', ['email' =>
18.     $this->session->userdata('email')]);
19.   $data['kelas'] = $this->M_userAdmin->
20.     get('kelas');
21.   $this->load->view('templates/header', $data);
22.   $this->load->view('templates/sidebar', $data);
23.   $this->load->view('admin/kelolakelas', $data);
24.   $this->load->view('templates/footer');
25.   }
26.
27.   public function tambahKelas()
28.   {
29.     $this->form_validation->
```

```
30.   set_rules('kelas', 'Kelas', 'required');
31.   if ($this->form_validation->run()) {
32.    $this->M_userAdmin->insert('kelas',
33.      ['nama_kelas' => $this->input->
34.      post('kelas')]);
35.    $this->session->set_flashdata('message',
36.      '<div class="alert alert-success" role=
37.      "alert">Kelas baru berhasil ditambahkan!
38.      </div>');
39.    redirect('admin/kelolaKelas');
40.   }
41.   $this->session->set_flashdata('message',
42.     '<div class="alert alert-warning" role=
43.     "alert">Masukkan nama kelas!</div>');
44.   redirect('admin/kelolaKelas');
45.   }
46.
47.   public function hapusKelas($id_kelas)
48.   {
49.    $data['user'] = $this->M_userAdmin->get_where
50.      ('user', ['email' => $this->session->
51.      userdata('email')]);
52.    $this->M_userAdmin->delete('kelas',
53.      ['id_kelas' => $id_kelas]);
54.    $this->session->set_flashdata('message',
55.      '<div class="alert alert-success" role=
56.      "alert">Kelas berhasil dihapus</div>');
57.    redirect('admin/kelolaKelas');
58.   }
59.   public function editKelas()
60.   {
61.    $this->form_validation->set_rules('kelas',
62.      'Kelas', 'required', ['required' =>
63.      'Nama kelas tidak boleh kosong !']);
64.    if ($this->form_validation->run() == false){
65.     $data['title'] = 'Kelola Kelas';
66.     $data['user'] = $this->M_userAdmin->
67.       get_where('user', ['email' => $this->
68.       session->userdata('email')]);
69.     $data['kelas'] = $this->M_userAdmin->
70.       get('kelas');
71.
72.     $this->load->view('templates/header'
73.       , $data);
74.     $this->load->view(
```

```
75.       'templates/sidebar', $data);
76.     $this->load->view(
77.       'admin/kelolakelas', $data);
78.     $this->load->view('templates/footer');
79.     } else {
80.     $this->M_userAdmin->update('kelas',
81.       ['nama_kelas' => $this->input->post('kelas')
82.       ], ['id_kelas' => $this->input->post(
83.       'id_kelas')]);
84.     $this->session->set_flashdata('message',
85.       '<div class="alert alert-success" role=
86.       "alert">Data kelas berhasil diubah!</div>');
87.     redirect('admin/kelolaKelas');
88.     }
89.   }
90.
91.   public function kelolaRole()
92.   {
93.     $data['title'] = 'Kelola Role';
94.     $data['user'] = $this->M_userAdmin->
95.       get_where('user', ['email' => $this->
96.       session->userdata('email')]);
97.     $data['kelas'] = $this->M_userAdmin->
98.       get('kelas');
99.     $data['role'] = $this->M_userAdmin->
100.      get('user_role');
101.
102.    $this->load->view('templates/header', $data);
103.    $this->load->view('templates/sidebar', $data);
104.    $this->load->view('admin/role', $data);
105.    $this->load->view('templates/footer');
106.  }
107.
108.
109.  public function roleAccess($role_id)
110.  {
111.    $data['title'] = 'Kelola Role';
112.    $data['user'] = $this->M_userAdmin->
113.      get_where('user', ['email' =>
114.      $this->session->userdata('email')]);
115.    $data['kelas'] = $this->M_userAdmin->
116.      get('kelas');
117.    $data['role'] = $this->M_userAdmin->get_where(
118.      'user_role', ['id' => $role_id]);
119.    $this->db->where('id !=', 1);
```

```
120.   $data['menu'] = $this->M_userAdmin->
121.     get('user_menu');
122.
123.   $this->load->view('templates/header', $data);
124.   $this->load->view('templates/sidebar', $data);
125.   $this->load->view('admin/role-access', $data);
126.   $this->load->view('templates/footer');
127.  }
128.
129.  public function changeAccess()
130.  {
131.   $menu_id = $this->input->post('menuId');
132.   $role_id = $this->input->post('roleId');
133.   $data = [
134.     'role_id' => $role_id,
135.     'menu_id' => $menu_id
136.   ];
137.
138.   $result = $this->db->get_where(
139.     'user_access_menu', $data);
140.
141.   if ($result->num_rows() < 1) {
142.    $this->M_userAdmin->insert(
143.      'user_access_menu', $data);
144.   } else {
145.    $this->M_userAdmin->delete('user_access_menu'
146.      , $data);
147.        }
148.    $this->session->set_flashdata('message',
149.      '<div class="alert alert-success" role=
150.      "alert">Role access berhasil diubah!
151.      </div>');
152.  }
153.
154.  public function hapusRole($role_id)
155.  {
156.   $data['role'] = $this->M_userAdmin->
157.     get_where('user_role', ['id' => $role_id]);
158.   $data['user'] = $this->M_userAdmin->
159.     get_where('user', ['email' =>
160.     $this->session->userdata('email')]);
161.   $this->M_userAdmin->delete('user_role',
162.     ['id' => $role_id]);
163.   $this->M_userAdmin->delete('user_access_menu',
164.     ['role_id' => $role_id]);
```

```
165.   $this->session->set_flashdata('message',
166.     '<div class="alert alert-   success" role=
167.     "alert">Role berhasil dihapus</div>');
168.   redirect('admin/kelolarole');
169.  }
170.
171. public function editRole()
172.  {
173.   $this->form_validation->set_rules('role', 'Role'
174.     , 'required', ['required' =>
175.     'Nama role tidak boleh kosong !']);
176.   if ($this->form_validation->run() == false) {
177.    $data['title'] = 'Kelola Role';
178.    $data['user'] = $this->M_userAdmin->
179.      get_where('user', ['email' =>
180.      $this->session->userdata('email')]);
181.    $data['kelas'] = $this->M_userAdmin->
182.      get('kelas');
183.    $data['role'] = $this->M_userAdmin->
184.      get('user_role');
185.
186.    $this->load->view('templates/header', $data);
187.    $this->load->view('templates/sidebar',
188.      $data);
189.    $this->load->view('admin/role', $data);
190.    $this->load->view('templates/footer');
191.   } else {
192.    $this->M_userAdmin->update('user_role',
193.      ['role' => $this->input->post('role')],
194.      ['id' => $this->input->post('id')]);
195.    $this->session->set_flashdata('message',
196.      '<div class="alert alert-success" role=
197.      "alert">Data Role berhasil diubah!</div>');
198.    redirect('admin/kelolarole');
199.   }
200.  }
201.
202.  public function tambahRole()
203.  {
204.   $data['user'] = $this->M_userAdmin->
205.     get_where('user', ['email' => $this->session->
206.     userdata('email')]);
207.   $data['kelas'] = $this->M_userAdmin->
208.     get('kelas');
209.   $this->form_validation->set_rules('role', 'Role'
```

```
210.     , 'required');
211.
212.   if ($this->form_validation->run() == false) {
213.    $this->load->view('templates/header', $data);
214.    $this->load->view(
215.      'templates/sidebar', $data);
216.    $this->load->view(
217.      'admin/kelolakelas', $data);
218.    $this->load->view('templates/footer');
219.   } else {
220.    $this->M_userAdmin->insert('user_role',
221.      ['role' => $this->input->post('role')]);
222.    $this->session->set_flashdata('message',
223.      '<div class="alert alert-success" role=
224.      "alert">Role baru berhasil ditambahkan!
225.      </div>');
226.    redirect('admin/kelolarole');
227.   }
228.  }
229. }
```

Kode 5.19 Controller Admin

Kode 5.19 merupakan hasil implementasi dari *controller Admin*. *Controller* ini memiliki fungsi untuk menangani fitur yang berhubungan dengan *Admin*. Terdapat 11 *function* di dalam *controller* Admin ini. *Function construct* merupakan fungsi dasar yang dijalankan disetiap pemanggilan fungsi lain di dalam *controller* yang sama. Isi dari *function construct* adalah *method is_logged_in()* yang digunakan untuk mengecek *login user* dan pemanggilan *model M_userAdmin* untuk penanganan data. Apabila terdapat akses ke fungsi lain, dilakukan pengecekan *session login* dengan mengecek kepemilikan *session email* yang mengakses. *Function kelolaKelas(), editKelas(), kelolaRole(), roleAccess(), editRole()*, dan *tambahRole()* berisi kode program untuk menampilkan *view* masing-masing. Ketika controller ini diakses, maka *controller* akan menjalankan logika pemrograman dan menampilkan halaman *view* yang terdaftar. *Function tambahKelas(), hapusKelas(), changeAccess()*, dan *hapusRole()* tidak memiliki kemampuan untuk mengakses *view*, sehingga ketika fungsi ini diakses hanya akan menjalankan logika program yang telah ditulis tanpa menampilkan *view*. Setelah fungsi selesai

dijalankan, akan dikembalikan ke halaman masing-masing dengan menampilkan flash data.

```
1.  <?php
2.  defined('BASEPATH') or exit('No direct script acces
    s allowed');
3.
4.  class Auth extends CI_Controller
5.  {
6.   public function __construct()
7.   {
8.    parent::__construct();
9.    $this->load->library('form_validation');
10.   $this->load->model('M_userAdmin');
11.  }
12.
13.  public function index()
14.  {
15.   if ($this->session->userdata('email')) {
16.    redirect('user');
17.   }
18.   $this->form_validation->set_rules('email',
19.     'Email', 'required|trim|valid_email');
20.   $this->form_validation->set_rules('password',
21.     'Password', 'required|trim');
22.   if ($this->form_validation->run() == false) {
23.    $data['title'] = "Smart Power - Login";
24.    $this->load->view('templates/auth_header'
25.     , $data);
26.    $this->load->view('auth/login');
27.    $this->load->view('templates/auth_footer');
28.   } else {
29.    $this->_login();
30.   }
31.  }
32.
33.  private function _login()
34.   {
35.    $email = $this->input->post('email');
36.    $password = $this->input->post('password');
37.
38.   $user = $this->M_userAdmin->get_where('user',
39.     ['email' => $email]);
40.
```

```
41.    //jika user ada
42.    if ($user) {
43.     if ($user['is_active'] == 1) {
44.      //cek password
45.      if (password_verify($password,
46.        $user['passwo    rd'])) {
47.       $data = [
48.        'email' => $user['email'],
49.        'role_id' => $user['role_id']
50.       ];
51.       $this->session->set_userdata($data);
52.       if ($user['role_id'] == 1) {
53.        redirect('home');
54.       } else {
55.        redirect('user');
56.       }
57.      } else {
58.       $this->session->set_flashdata('message',
59.         '<div class="alert alert-danger" role=
60.         "alert">Password salah !
61.         Silakan periksa kembali. </div>');
62.       redirect('auth');
63.      }
64.     } else {
65.      $this->session->set_flashdata('message',
66.        '<div class="alert alert-danger" role=
67.        "alert">Mohon maaf, email ini
68.        belum diaktivasi. Silakan hubungi
69.        tim admin untuk aktivasi
70.        akun anda. </div>');
71.      redirect('auth');
72.     }
73.    } else {
74.     $this->session->set_flashdata('message',
75.       '<div class="alert alert-danger" role=
76.       "alert">Email ini belum terdaftar !
77.       </div>');
78.     redirect('auth');
79.    }
80.   }
81.
82.   public function registration()
83.   {
84.    if ($this->session->userdata('email')) {
85.     redirect('user');
```

```
86.     }
87.     $this->form_validation->set_rules(‘name’,
88.       ‘Name’, ‘required|trim’);
89.     $this->form_validation->set_rules(‘email’,
90.       ‘Email’, ‘required|trim|valid_email|
91.       is_unique[user.email]’, [
92.       ‘is_unique’ => ‘Email ini sudah terdaftar !’
93.       ]);
94.      $this->form_validation->set_rules(
95.       ‘password1’,
96.       ‘Password’,
97.       ‘required|trim|min_length[6]|matches[
98.         password2]’,
99.       [
100.       ‘matches’ => ‘password tidak sama !’,
101.       ‘min_length’ => ‘Password terlalu singkat
102.         (min 6 karakter)’
103.       ]
104.      );
105.      $this->form_validation->set_rules
106.        (‘password2’, ‘Password’, ‘required|
107.        trim|matches[password1]’);
108.      if ($this->form_validation->run()==false){
109.       $data[‘title’]=”Smart Power – Registration”;
110.       $this->load->view(‘templates/auth_header’
111.         , $data);
112.       $this->load->view(‘auth/registration’);
113.       $this->load->view(‘templates/auth_footer’);
114.      } else {
115.       $email = $this->input->post(‘email’);
116.       $data = [
117.        ‘nama_user’ => htmlspecialchars($this->input
118.          ->post(‘name’), true),
119.        ‘email’ => htmlspecialchars($this->input->
120.          post(‘email’), true),
121.        ‘image’ => ‘default.jpg’,
122.        ‘password’ => password_hash($this->input->
123.          post(‘password1’), PASSWORD_DEFAULT),
124.          ‘role_id’ => 2,
125.          ‘is_active’ => 0,
126.          ‘date_created’ => time()
127.       ];
128.
129.       $token = base64_encode(random_bytes(32));
130.       $user_token = [
```

```
131.        'email' => $email,
132.        'token' => $token,
133.        'date_created' => time()
134.      ];
135.      $this->M_userAdmin->insert('user', $data);
136.      $this->M_userAdmin->
137.        insert('user_token', $user_token);
138.      $this->_sendEmail($token, 'verify');
139.
140.      $this->session->set_flashdata('message',
141.        '<div class="alert alert-success"
142.        role="alert">Akun anda berhasil dibuat.
143.        Silakan hubungi tim admin untuk aktivasi
144.        akun anda. </div>');
145.      redirect('auth');
146.    }
147.  }
148.
149.  public function verify()
150.  {
151.    $email = $this->input->get('email');
152.    $token = $this->input->get('token');
153.
154.    $user = $this->M_userAdmin->
155.      get_where('user', ['email' => $email]);
156.
157.    if ($user) {
158.      $user_token = $this->M_userAdmin->get_where(
159.        'user_token', ['token' => $token]);
160.
161.      if ($user_token) {
162.        $this->M_userAdmin->update('user',
163.          ['is_active' => 1],['email' => $email]);
164.        $this->M_userAdmin->delete('user_token',
165.          ['email' => $email]);
166.
167.        $this->session->set_flashdata('message',
168.          '<div class="alert alert-success" role=
169.          "alert">' . $email . ' berhasil
170.          diaktivasi!.</div>');
171.        redirect('auth');
172.      } else {
173.        $this->session->set_flashdata('message',
174.          '<div class="alert alert-danger" role=
175.          "alert">Aktivasi akun gagal !
```

```
176.         Token salah .</div>');
177.       redirect('auth');
178.      }
179.     } else {
180.      $this->session->set_flashdata('message',
181.        '<div class="alert alert-danger" role=
182.        "alert">Aktivasi akun gagal !
183.        email salah.</div>');
184.      redirect('auth');
185.     }
186.    }
187.    private function _sendEmail($token, $type)
188.    {
189.     $this->load->library('email');
190.     $config = array();
191.     $config['protocol'] = 'smtp';
192.     $config['smtp_host'] =
193.       'ssl://smtp.googlemail.com';
194.     $config['smtp_user'] =
195.       'smartpower.auth@gmail.com';
196.     $config['smtp_pass'] = 'smartpower123';
197.     $config['smtp_port'] = 465;
198.     $config['mailtype'] = 'html';
199.     $config['charset'] = 'utf-8';
200.     $this->email->initialize($config);
201.     $this->email->set_newline("\r\n");
202.
203.     $this->email->from(
204.       'smartpower.auth@gmail.com',
205.       'Smart Power Account Verification');
206.
207.     if ($type == 'verify') {
208.      $this->email->to(
209.        'smartpower.verify@gmail.com');
210.      $this->email->subject(
211.        'Smart Power Account Verification');
212.      $this->email->message('Klik link berikut
213.        untuk aktivasi akun : <a href="'
214.        .base_url().'auth/verify?email='
215.        . $this->input->post('email').'&token='
216.        .urlencode($token).'">Activate</a>');
217.     } else if ($type == 'forgot') {
218.      $this->email->to(
219.        $this->input->post('email'));
220.      $this->email->subject('Reset Password');
```

```
221.      $this->email->message('Klik link berikut
222.        untuk reset password : <a href="'
223.        .base_url().'auth/resetpassword?email='
224.        .$this->input->post('email') . '&token='
225.        . urlencode($token) . '">
226.        Reset Password</a>');
227.     }
228.
229.     if ($this->email->send()) {
230.      return true;
231.     } else {
232.      echo $this->email->print_debugger();
233.      die;
234.     }
235.    }
236.
237.    public function logout()
238.    {
239.     $this->session->unset_userdata('email');
240.     $this->session->unset_userdata('role_id');
241.
242.     $this->session->set_flashdata('message',
243.       '<div class="alert alert-success" role=
244.       "alert">Anda telah logout </div>');
245.     redirect('auth');
246.    }
247.
248.    public function blocked()
249.    {
250.     $this->load->view('auth/blocked');
251.    }
252.
253.    public function forgotPassword()
254.    {
255.     $this->form_validation->set_rules('email',
256.       'Email', 'trim|required|valid_email');
257.
258.     if ($this->form_validation->run()==false){
259.      $data['title'] = "Lupa Password";
260.      $this->load->view(
261.        'templates/auth_header',$data);
262.      $this->load->view('auth/forgotpassword');
263.      $this->load->view(
264.        'templates/auth_footer');
265.     } else {
```

```
266.      $email = $this->input->post('email');
267.      $user = $this->M_userAdmin->get_where
268.        ('user', ['email' => $email,
269.        'is_active' => 1]);
270.
271.      if ($user) {
272.       $token = base64_encode(random_bytes(32));
273.       $user_token = [
274.        'email' => $email,
275.        'token' => $token,
276.        'date_created' => time()
277.       ];
278.
279.       $this->M_userAdmin->insert('user_token',
280.         $user_token);
281.       $this->_sendEmail($token, 'forgot');
282.
283.       $this->session->set_flashdata('message',
284.         '<div class="alert alert-success" role=
285.         "alert">Please check your email to reset
286.         your password!</div>');
287.       redirect('auth/forgotpassword');
288.      } else {
289.       $this->session->set_flashdata('message',
290.         '<div class="alert alert-danger" role=
291.         "alert">Email belum teraktivasi atau
292.         tidak terdaftar!</div>');
293.       redirect('auth/forgotpassword');
294.      }
295.     }
296.    }
297.
298.    public function resetPassword()
299.    {
300.     $email = $this->input->get('email');
301.     $token = $this->input->get('token');
302.
303.     $user = $this->M_userAdmin->get_where
304.       ('user', ['email' => $email]);
305.
306.     if ($user) {
307.      $user_token = $this->M_userAdmin->get_where
308.        ('user_token', ['token' => $token]);
309.
310.      if ($user_token) {
```

```
311.        $this->session->set_userdata('reset_email'
312.          , $email);
313.        $this->changePassword();
314.      } else {
315.        $this->session->set_flashdata('message',
316.          '<div class="alert alert-danger" role=
317.          "alert">Reset password gagal!
318.          Token salah.</div>');
319.        redirect('auth');
320.      }
321.    } else {
322.      $this->session->set_flashdata('message',
323.        '<div class="alert alert-danger" role=
324.        "alert">Reset password gagal!
325.        Email salah.</div>');
326.      redirect('auth');
327.    }
328.  }
329.
330.  public function changePassword()
331.  {
332.    if (!$this->session->userdata(
333.      'reset_email')) {
334.      redirect('auth');
335.    }
336.
337.    $this->form_validation->set_rules(
338.      'password1','Password','trim|required|
339.      min_length[3]|matches[password2]');
340.    $this->form_validation->set_rules(
341.      'password2','Repeat Password',
342.      'trim|required|min_length[3]
343.      |matches[password1]');
344.
345.    if ($this->form_validation->run()==false){
346.      $data['title'] = 'Ganti Password';
347.      $this->load->view('templates/auth_header'
348.        , $data);
349.      $this->load->view('auth/changepassword');
350.      $this->load->view(
351.        'templates/auth_footer');
352.    } else {
353.      $password = password_hash(
354.        $this->input->post('password1'),
355.        PASSWORD_DEFAULT);
```

```
356.      $email = $this->session->
357.        userdata('reset_email');
358.      $this->M_userAdmin->update('user',
359.        ['password' => $password],
360.        ['email' => $email]);
361.      $this->session->
362.        unset_userdata('reset_email');
363.      $this->M_userAdmin->delete('user_token',
364.        ['email' => $email]);
365.      $this->session->set_flashdata('message',
366.        '<div class="alert alert-success" role=
367.        "alert">Password berhasil diganti!
368.        Silakan login.</div>');
369.      redirect('auth');
370.    }
371.  }
372. }
```

Kode 5.20 Controller Auth

Kode 5.20 di atas merupakan hasil implementasi kode pada *controller Auth*. Controller ini memiliki peran penting pada sistem, yakni melakukan penanganan fitur terkait otentifikasi pada sistem. Terdapat 11 *function* di dalam *controller* ini, sama seperti controller lain *function construct()* digunakan untuk menjalankan method default yang akan dijalankan pada seluruh *function* pada *controller*.

Function index() berfungsi untuk menampilkan halaman default saat web diakses, yakni menampilkan halaman login dan memanggil *function _login()* saat melakukan otentifikasi akun. Apabila akun benar terdaftar dan sudah aktif, maka *user* dapat masuk ke halaman utama

Function registration() digunakan untuk menangani pendaftaran akun. *User* akan diminta mengisi form pendaftaran yang masing-masing form terdapat ketentuan khusus. Saat pendaftaran berhasil, selanjutnya sistem akan mengirim pesan verifikasi akun ke tim admin menggunakan *function _sendEmail()*. Akun yang belum diverifikasi oleh tim admin masih belum dapat digunakan. Proses ini diciptakan untuk memastikan pengguna sistem ini

merupakan *stakeholder* yang terkait di DSI. Saat *admin* telah mengaktivasi akun dengan klik tautan yang masuk melalui *email*, sistem akan melakukan cek kode token dengan *function verify()*.

Beberapa *function* lain di dalam controller ini memiliki peran dalam menangani fitur pengelolaan *password* yakni *forgotPassword()*, *resetPassword()*, dan *changePassword()*. Selain pada penanganan password, terdapat 2 *function* lain pada *controller* ini, yakni *logout()* untuk menangani *logout* akun dan *blocked()* yang berfungsi untuk menampilkan halaman "*403 Forbidden*".

```
1.  class Data extends CI_Controller
2.  {
3.
4.   public function __construct()
5.   {
6.    parent::__construct();
7.    $this->load->model('M_graph');
8.   }
9.
10.  public function getDataKelas($id_kelas)
11.  {
12.   $data['sensor'] = $this->M_graph->
13.     selectDailyKelasData($id_kelas);
14.   echo json_encode($data['sensor']);
15.  }
16.  public function getDataKelasAll($id_kelas)
17.  {
18.   $data['all'] = $this->M_graph->
19.     selectKelasData($id_kelas);
20.   echo json_encode($data['all']);
21.  }
22.
23.  public function getChartData()
24.  {
25.   $data = $this->M_graph->
26.     selectDailyChartData();
27.   echo json_encode($data);
28.  }
```

```
29.
30.   public function getChartDataAll()
31.   {
32.    $data = $this->M_graph->selectChartDataAll();
33.    echo json_encode($data);
34.   }
35.
36.   public function getEnergy()
37.   {
38.    $data['energy'] = $this->M_graph->
39.      selectTotalEnergy();
40.    echo json_encode($data['energy']);
41.   }
42.
43.   public function getEnergyByMonth()
44.   {
45.    $data['total_energy'] = $this->M_graph->
46.      selectEnergyByMonth();
47.    echo json_encode($data['total_energy']);
48.   }
49.
50.   public function getLastDataChart()
51.   {
52.    $data['last_chart_data'] = $this->M_graph->
53.      selectLastData();
54.    echo json_encode($data['last_chart_data']);
55.   }
56.
57.   public function getSetting()
58.   {
59.    $data['setting'] = $this->M_graph->
60.      selectSetting();
61.    echo json_encode($data['setting']);
62.   }
63.
64.   //data masukan dari Arduino
65.   public function insertData()
66.   {
67.    date_default_timezone_set('Asia/Jakarta');
68.    $now = date('Y-m-d H:i:s');
69.    $this->load->model('M_data');
70.
71.    if ($_POST['key'] == "12345678") {
72.     $id_kelas = $this->input->post('id_kelas');
73.     $voltage = $this->input->post('voltage');
```

```
74.     $current = $this->input->post('current');
75.     $power = $this->input->post('power');
76.     $energy = $this->input->post('energy');
77.     $temperature =
78.       $this->input->post('temperature');
79.     $datasensor = array(
80.      'id_kelas' => $id_kelas,
81.      'voltage' => $voltage,
82.      'current' => $current,
83.      'power' => $power,
84.      'energy' => $energy,
85.      'temperature' => $temperature,
86.      'date' => $now
87.     );
88.     $this->M_data->insert(
89.       'data_kelas', $datasensor);
90.     $data_chart = $this->M_data->insert_chart();
91.     $this->M_data->insert(
92.       'data_chart', $data_chart);
93.    } else {
94.     echo "kode salah";
95.    }
96.   }
97. }
```

Kode 5.21 Controller Data

Kode 5.21 di atas merupakan hasil implementasi pada *controller Data*. *Controller* ini berfungsi dalam penanganan data pada sistem dengan menjalankan *query SQL* melalui *model M_graph* dan *M_data*. Function yang berfungsi untuk mengambil data dari *database* yakni *getDataKelas(), getDataKelasAll(), getChartData(), getChartDataAll(), getEnergy(), getEnergyByMonth(), getLastDataChart()*, dan *getSetting()*. 1 *function* lain yang digunakan untuk memasukkan data yang diterima dari Arduino ke *database* adalah *insertData()*. Fungsi ini melakukan *insert* data ke tabel *data_kelas* dan tabel *data_chart*.

```
1.  class Home extends CI_Controller {
2.   public function __construct() {
3.    parent::__construct();
```

```
4.     $this->load->model('M_userAdmin');
5.     is_logged_in();
6.    }
7.
8.    public function index() {
9.     $data['title'] = 'Dashboard Utama';
10.    $data['user'] = $this->M_userAdmin->
11.      get_where('user', ['email' => $this->session
12.      ->userdata('email')]);
13.    $data['kelas'] = $this->M_userAdmin->
14.      get('kelas');
15.
16.    $this->load->view('templates/header', $data);
17.    $this->load->view('templates/sidebar', $data);
18.    $this->load->view('home/index', $data);
19.    $this->load->view('templates/footer', $data);
20.   }
21.  }
```

Kode 5.22 Controller Home

Kode 5.22 di atas merupakan hasil implementasi pada *controller Home*. Hanya terdapat 2 function di dalam controller ini, yakni *__construct()* sebagai *method default* dan *index()*. *Function __construct* berfungsi untuk memanggil *method is_logged_in()* dan *load model M_userAdmin*. *Function index()* berfungsi untuk menampilkan halaman *dashboard* utama.

```
1.   class Kelas extends CI_Controller {
2.
3.    public function __construct() {
4.     parent::__construct();
5.     $this->load->model('M_userAdmin');
6.     is_logged_in();
7.    }
8.
9.    public function index($id_kelas) {
10.
11.    $data['kelas'] = $this->M_userAdmin->
12.      get('kelas');
13.    $data['current_kelas'] = $this->M_userAdmin->
14.      get_where('kelas',
15.      ['id_kelas' => $id_kelas]);
16.    $data['title'] =
```

```
17.      $data[‘current_kelas’][‘nama_kelas’];
18.    $data[‘user’] = $this->M_userAdmin->get_where
19.      (‘user’, [‘email’ => $this->session->
20.      userdata(‘email’)]);
21.    $data[‘id_kelas’] = $id_kelas;
22.    $this->load->view(‘templates/header’, $data);
23.    $this->load->view(‘templates/sidebar’, $data);
24.    $this->load->view(‘kelas/index’, $data);
25.    $this->load->view(‘kelas/footer’, $data);
26.   }
27. }
```

Kode 5.23 Controller Kelas

Kode 5.23 di atas adalah hasil implementasi dari *controller* Kelas. *Controller* ini hanya memiliki fungsi untuk menampilkan view masing-masing kelas sesuai id_kelas.

```
1.   class Menu extends CI_Controller {
2.    public function __construct() {
3.     parent::__construct();
4.     $this->load->model('M_userAdmin');
5.     is_logged_in();
6.    }
7.
8.    public function index() {
9.     $data['title'] = 'Kontrol Kelas';
10.    $data['user'] = $this->M_userAdmin-
    >get_where('user', ['email' => $this->session-
    >userdata('email')]);
11.    $data['kelas'] = $this->M_userAdmin->
12.      get('kelas');
13.    $data['setting'] = $this->M_userAdmin->
14.      get('setting');
15.
16.    $this->load->view('templates/header', $data);
17.    $this->load->view('templates/sidebar', $data);
18.    $this->load->view('menu/index', $data);
19.    $this->load->view('templates/footer');
20.   }
21.
22.   public function editSetting() {
23.    $this->form_validation->set_rules('suhumin',
24.      'Suhumin', 'required|less_than
25.      ['.$this->input->post('suhumax').']', [
```

```
26.       'required' => 'Suhu minimal tidak boleh kosong
27.       !', 'less_than' => 'Nilai harus kurang dari
28.       suhu maximal !']);
29.     $this->form_validation->set_rules('suhumax',
30.       'Suhumax','required|greater_than['.$this-
    >input->post('suhumin') . ']', [
31.       'required' => 'Suhu maximal tidak boleh kosong
32.       !', 'greater_than' => 'Nilai harus lebih dari
33.       suhu minimal !']);
34.     $this->form_validation->set_rules('durasi',
35.       'Durasi', 'required', [
36.       'required' => 'Durasi tidak boleh kosong !']);
37.     if ($this->form_validation->run() == false) {
38.      $data['title'] = 'Kontrol Kelas';
39.      $data['user'] = $this->M_userAdmin->get_where
40.        ('user', ['email' => $this->session->
41.        userdata('email')]);
42.      $data['kelas'] = $this->M_userAdmin->
43.        get('kelas');
44.      $data['setting'] = $this->M_userAdmin->
45.        get('setting');
46.
47.      $this->load->view('templates/header', $data);
48.      $this->load->view(
49.        'templates/sidebar', $data);
50.      $this->load->view('menu/index', $data);
51.      $this->load->view('templates/footer');
52.     } else {
53.      $data = [
54.       'nama_kelas' => $this->input->
55.         post('nama_kelas'),
56.       'mode' => $this->input->post('mode'),
57.       'suhu_min' => $this->input->post('suhumin'),
58.       'suhu_max' => $this->input->post('suhumax'),
59.       'durasi_menyala' => $this->input->
60.         post('durasi')
61.      ];
62.      $this->M_userAdmin->update('setting', $data
63.        ,['id_kelas' => $this->input->post(
64.        'id_kelas')]);
65.      $this->session->set_flashdata('message',
66.        '<div class="alert alert-success" role=
67.        "alert">Setting berhasil diubah ! </div>');
68.      redirect('menu/index');
69.     }
```

```
70.   }
71. }
```

Kode 5.24 Controller Menu

Kode 5.24 menunjukkan kode program pada *controller Menu* yang digunakan untuk menangani fitur kontrol kelas. Terdapat 3 *function* di dalam *controller* ini, yakni *__construct(), index(),* dan *editSetting(). Function index()* berfungsi untuk menampilkan halaman kontrol kelas, sedangkan *function editSetting()* digunakan untuk menangani logika pemrograman dalam perubahan setting yang dilakukan. User dapat melakukan perubahan dengan klik tombol edit pada kelas yang dituju. Perubahan parameter dapat diisi melalui form yang ada pada *modal*, masing-masing *form* memiliki ketentuan yang harus dipenuhi. Apabila ketentuan sudah dipenuhi, sistem akan melakukan *update* data pada *database* dan muncul notifikasi berhasil.

```
1.  class User extends CI_Controller {
2.   public function __construct() {
3.    parent::__construct();
4.    is_logged_in();
5.    $this->load->model('M_userAdmin');
6.   }
7.
8.   public function index() {
9.    $data['title'] = 'Profil Saya';
10.   $data['user'] = $this->M_userAdmin->get_where(
11.     'user', ['email' => $this->session->
12.     userdata('email')]);
13.   $data['kelas'] = $this->M_userAdmin->
14.     get('kelas');
15.   $this->load->view('templates/header', $data);
16.   $this->load->view('templates/sidebar', $data);
17.   $this->load->view('user/index', $data);
18.   $this->load->view('templates/footer');
```

```
19.   }
20.
21.   public function edit() {
22.    $data['title'] = 'Ubah Profil Saya';
23.    $data['user'] = $this->M_userAdmin->get_where(
24.      'user', ['email' => $this->session-      >
25.      userdata('email')]);
26.    $data['kelas'] = $this->M_userAdmin->
27.      get('kelas');
28.    $this->form_validation->set_rules('nama_user',
29.      'Full Name', 'required|trim');
30.
31.    if ($this->form_validation->run() == false) {
32.     $this->load->view('templates/header', $data);
33.     $this->load->view(
34.       'templates/sidebar', $data);
35.     $this->load->view('user/edit', $data);
36.     $this->load->view('templates/footer');
37.    } else {
38.     $name = $this->input->post('nama_user');
39.     $email = $this->input->post('email');
40.
41.     // cek jika ada gambar yang akan diupload
42.     $upload_image = $_FILES['image']['name'];
43.     if ($upload_image) {
44.      $config['allowed_types'] = 'gif|jpg|png';
45.      $config['max_size']      = '2048';
46.      $config['upload_path'] =
47.        './assets/dist/img/profile/';
48.
49.      $this->load->library('upload', $config);
50.
51.      if ($this->upload->do_upload('image')) {
52.       $old_image = $data['user']['image'];
53.       if ($old_image != 'default.jpg') {
54.        unlink(FCPATH . 'assets/dist/img/profile/'
55.          .$old_image);
56.       }
57.       $new_image = $this->upload->
58.         data('file_name');
59.       $this->M_userAdmin->update('user',
60.         ['image' => $new_image],
61.         ['email' => $email]);
62.      } else {
63.       echo $this->upload->dispay_errors();
```

```
64.       }
65.     }
66.     $this->M_userAdmin->update('user',
67.       ['nama_user' => $name],
68.       ['email' => $email]);
69.
70.     $this->session->set_flashdata('message',
71.       '<div class="alert alert-success" role=
72.       "alert">Profil anda telah
73.       berhasil diperbarui!</div>');
74.     redirect('user');
75.   }
76.  }
77.
78.  public function changePassword() {
79.   $data['title'] = 'Ganti Password';
80.   $data['user'] = $this->M_userAdmin->
81.     get_where('user', ['email' => $this->session->
82.     userdata('email')]);
83.   $data['kelas'] = $this->M_userAdmin->
84.     get('kelas');
85.
86.   $this->form_validation->set_rules(
87.     'current_password','Password Saat ini',
88.     'required|trim');
89.   $this->form_validation->set_rules
90.     ('new_password1','Password Baru',
91.      'required|trim|min_length[6]|
92.       matches[new_password2]');
93.   $this->form_validation->set_rules
94.     ('new_password2','Ulangi Password Baru',
95.      'required|trim|min_length[6]|
96.      matches[new_password1]');
97.
98.   if ($this->form_validation->run() == false) {
99.    $this->load->view('templates/header', $data);
100.   $this->load->view(
101.     'templates/sidebar', $data);
102.   $this->load->view(
103.     'user/changepassword', $data);
104.   $this->load->view('templates/footer');
105.  } else {
106.   $current_password = $this->input->post(
107.     'current_password');
108.   $new_password = $this->input->post(
```

```
109.      'new_password1');
110.    if (!password_verify($current_password,
111.      $data['user']['password'])) {
112.     $this->session->set_flashdata('message',
113.       '<div class="alert alert-danger" role=
114.       "alert">Password saat ini salah!</div>');
115.     redirect('user/changepassword');
116.    } else {
117.     if ($current_password == $new_password) {
118.      $this->session->set_flashdata('message',
119.       '<div class="alert alert-danger" role=
120.       "alert">Password baru tidak boleh
121.       sama dengan password saat ini!</div>');
122.      redirect('user/changepassword');
123.     } else {
124.      // password sudah ok
125.      $password_hash = password_hash
126.        ($new_password, PASSWORD_DEFAULT);
127.      $this->M_userAdmin->update('user',
128.        ['password' => $password_hash],
129.        ['email' => $this->session->
130.        userdata('email')]);
131.
132.      $this->session->set_flashdata('message',
133.        '<div class="alert alert-success" role=
134.        "alert">Password berhasil diubah!
135.        </div>');
136.      redirect('user/changepassword');
137.     }
138.    }
139.   }
140.  }
141. }
```

Kode 5.25 Controller User

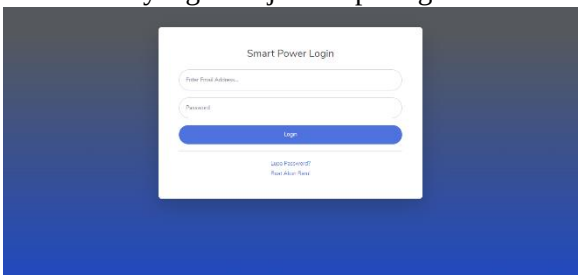
Kode 5.25 di atas merupakan hasil implementasi kode program pada *controller User*. *Controller* ini memiliki fungsi dalam penanganan fitur yang dimiliki oleh *user*. Terdapat 4 function di dalam controller ini, yakni *__construct(), index(), edit()*, dan *changePassword()*. *Function index()* digunakan untuk menampilkan halaman profil saya. *Function edit()* digunakan untuk menangani fitur edit profil user, meliputi ganti nama profil dan foto profil.

5.4.3 Implementasi *View*

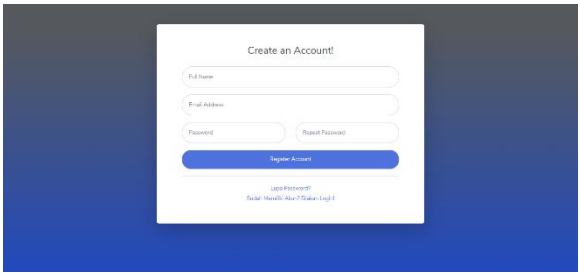
Komponen *view* pada *Codeigniter* merupakan bagian yang akan dilihat langsung oleh pengguna aplikasi web. Masing-masing *view* akan dikontrol oleh *controller* yang telah dijabarkan pada poin sebelumya. Berikut adalah view yang telah diimplementasikan pada aplikasi web yang ditunjukkan berdasarkan direktori penyimpanannya.

5.4.3.1 Implementasi *View* pada Direktori *Auth*

Terdapat 5 view pada direktori Auth. Berikut ini merupakan hasil implementasi view yang ditunjukkan pada gambar 5.5 – 5.8.



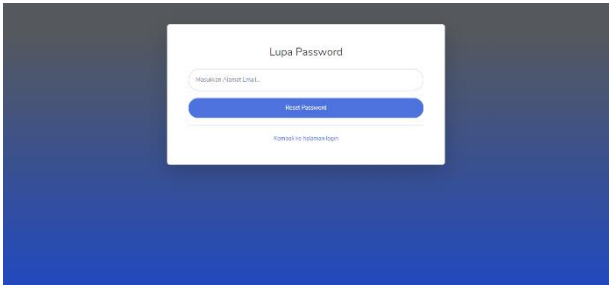
Gambar 5.5 Tampilan Halaman Login



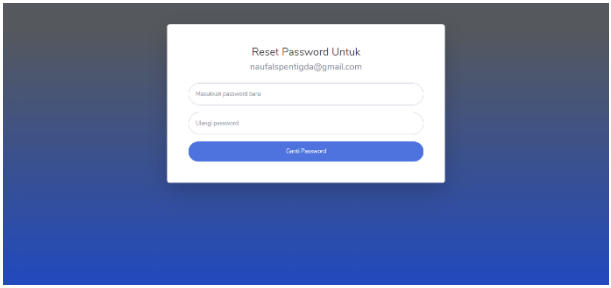
Gambar 5.6 Tampilan Halaman Daftar Akun

Gambar 5.5 di atas merupakan implementasi *view* halaman *login*. *User* diminta untuk mengisikan *email* dan *password* yang sudah terdaftar pada *form* yang disediakan. Gambar 5.6 merupakan

hasil implementasi view halaman daftar akun. *User* diminta untuk mengisikan *email* saat pendaftaran pada *form*.



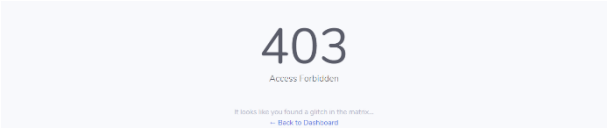
Gambar 5.7 Tampilan Halaman Lupa Password



Gambar 5.8 Tampilan Halaman Reset Password

Gambar 5.7 merupakan hasil implementasi *view* pada halaman lupa *password*. *User* diminta untuk memasukkan alamat *email* akun yang terdaftar, sistem akan mengirimkan alamat *link reset password* melalui *email*. Setelah *user* mengakses *link* yang telah dikirimkan maka akan diarahkan ke halaman *reset password* yang ditunjukkan pada gambar 5.8.

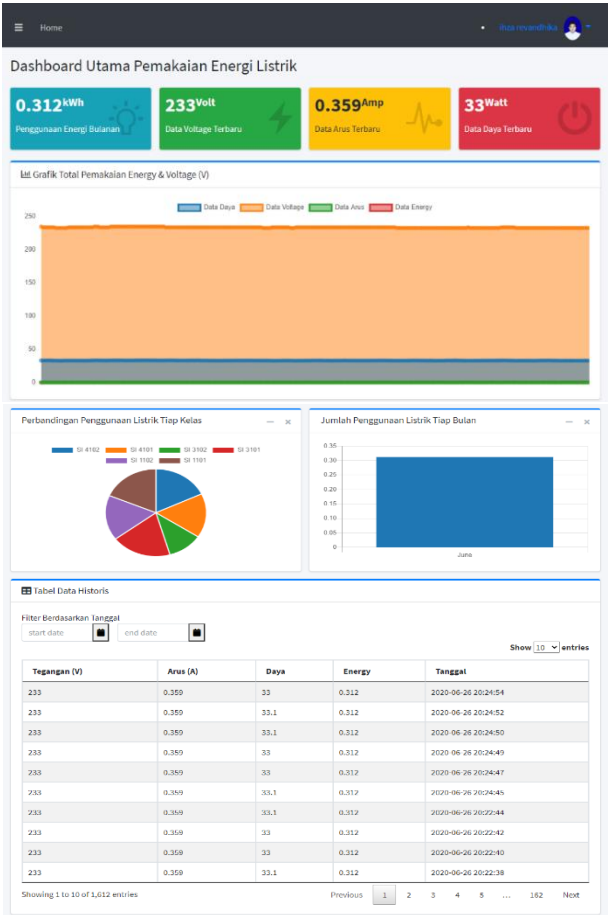
Selanjutnya terdapat hasil implementasi *view* pada halaman *blocked*. Halaman ini akan ditampilkan ketika *user* yang tidak memiliki otoritas mencoba mengakses halaman yang dilarang. Berikut adalah tampilan halaman blocked yang ditunjukkan pada gambar 5.9.



Gambar 5.9 Halaman Blocked

5.4.3.2 Implementasi *View* pada Direktori *Home*

Terdapat 1 view saja pada direktori *Home*, yakni tampilan pada halaman *dashboard* utama. Berikut adalah tampilan halaman *dashboard* utama yang ditunjukkan pada gambar 5.10.



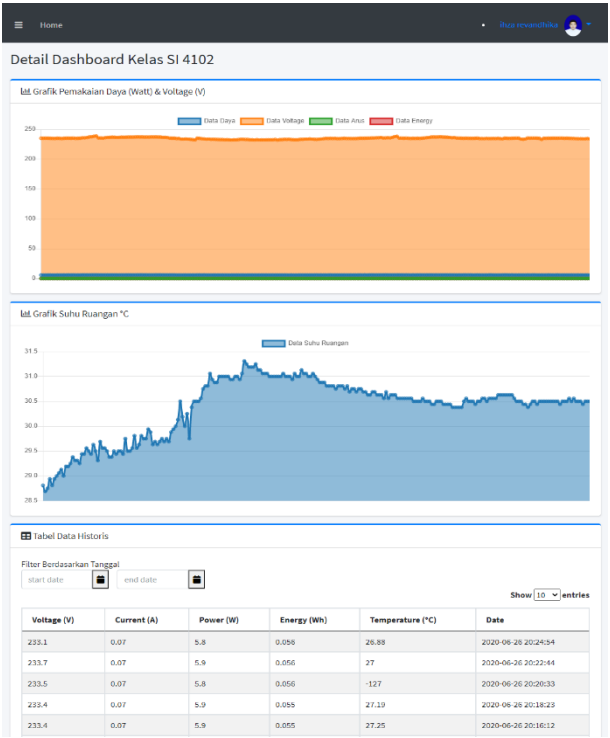
Gambar 5.10 Tampilan Halaman Dashboard Utama

Pada sisi atas halaman, terdapat 4 informasi yang ditampilkan dalam bentuk text, yakni informasi total akumulasi penggunaan energi listrik bulanan, rata-rata tegangan listrik dari seluruh

kelas, besar arus listrik yang mengalir ke seluruh kelas, dan besar daya yang digunakan di seluruh kelas. Selanjutnya terdapat grafik yang menampilkan data kelistrikan secara historis dalam bentuk diagram garis. Pada sisi bawah terdapat informasi perbandingan total penggunaan energi listrik tiap kelas yang ditampilkan dalam bentuk diagram lingkaran, informasi jumlah penggunaan energi listrik tiap bulan yang ditampilkan dalam bentuk diagram batang, dan pada sisi bawah terdapat tabel yang menampilkan data historis.

5.4.3.3 Implementasi *View* pada Direktori Kelas

Berikut adalah tampilan halaman *dashboard* masing-masing kelas yang ditunjukkan pada gambar 5.11.

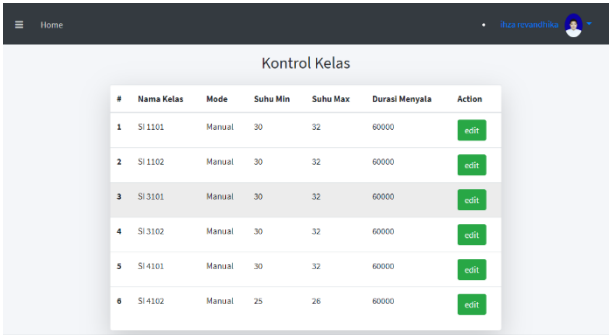


Gambar 5.11 Tampilan Halaman Kelas

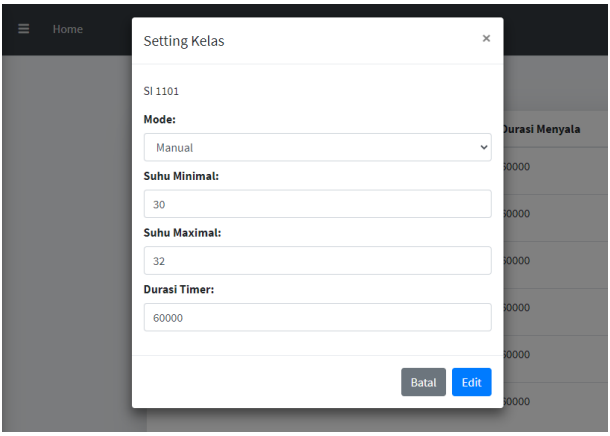
Terdapat 1 *view* pada direktori kelas, berisi tampilan pada dashboard masing-masing kelas. Terdapat 2 grafik dan 1 tabel dalam halaman ini. Di sisi atas terdapat informasi kelistrikan untuk spesifik kelas dalam bentuk grafik garis. Selanjutnya terdapat informasi suhu ruang kelas yang ditunjukkan dalam bentuk diagram garis. Seluruh data historis ditampilkan pada tabel yang terdapat di sisi bawah halaman web.

5.4.3.4 Implementasi *View* pada Direktori *Menu*

Pada direktori *Menu* terdapat 1 view yang dibuat, yakni halaman kontrol kelas. Pada halaman tersebut ditampilkan seluruh pengaturan untuk Arduino meliputi mode kehematan, *threshold* suhu, dan *timer* nyala listrik. Jika ingin melakukan perubahan setting, pengguna dapat mengakses tombol edit pada kelas yang ingin dirubah. Selanjutnya akan muncul tampilan *form* modal untuk mengubah setting, apabila pengguna memasukkan pengaturan yang tidak sesuai maka akan muncul notifikasi kesalahan berwarna merah pada halaman tersebut. Bila *update* setting berhasil, akan muncul notifikasi sukses berwarna hijau. Halaman kontrol kelas dan modal setting kelas ditampilkan secara berturu-turut pada gambar 5.12 dan 5.13 di bawah ini



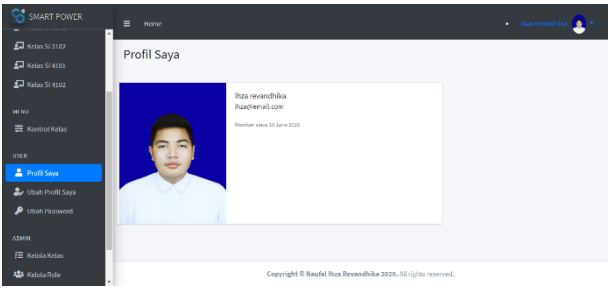
Gambar 5.12 Tampilan Halaman Kontrol Kelas



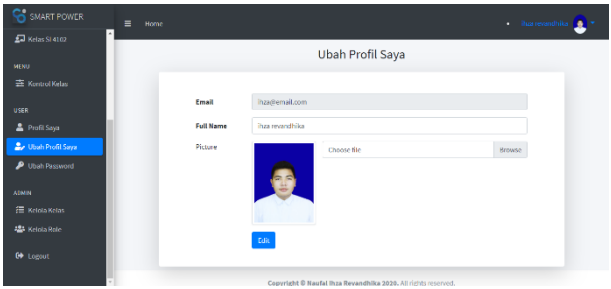
Gambar 5.13 Tampilan Modal Setting Kelas

5.4.3.5 Implementasi *View* pada Direktori *User*

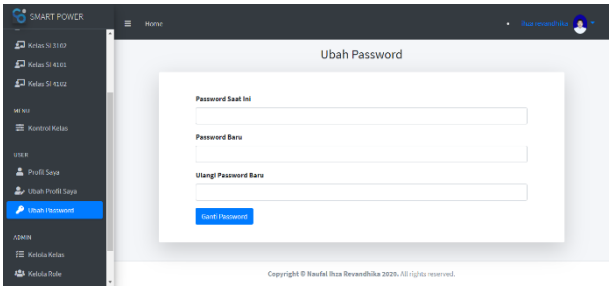
Pada direktori *User* terdapat 3 *view* yang dibuat, yakni view halaman profil saya sebagai halaman utama, halaman ubah profil saya, dan halaman ubah password. Tampilan masing-masing halaman berturut-turut ditunjukkan pada gambar 5.14 – 5.16 berikut ini.



Gambar 5.14 Tampilan Halaman Profil Saya



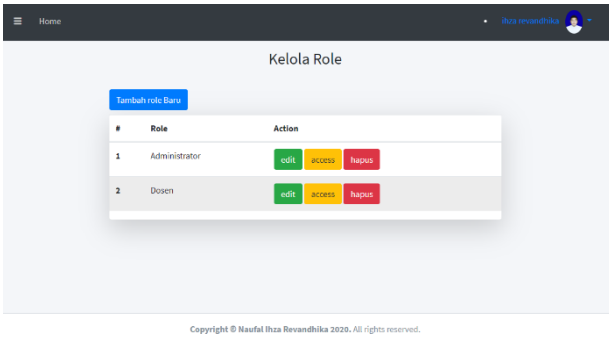
Gambar 5.15 Tampilan Halaman Ubah Profil Saya



Gambar 5.16 Tampilan Halaman Ubah Password

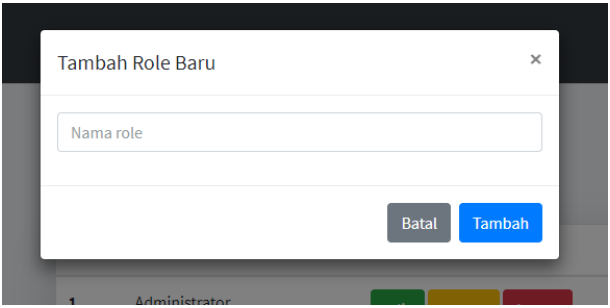
5.4.3.6 Implementasi *View* pada Direktori *Admin*

Pada direktori *Admin* terdapat 3 view yang dibuat, yakni view halaman kelola role, halaman kelola akses role, dan kelola kelas.

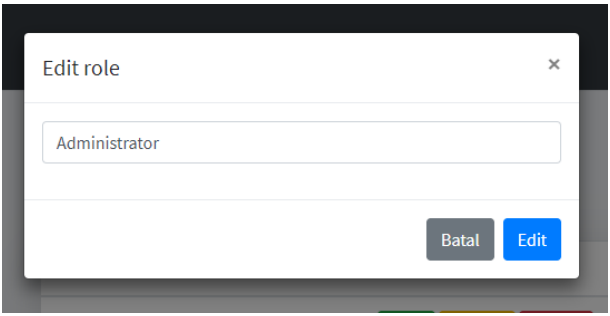


Gambar 5.17 Tampilan Halaman Kelola Role

Gambar 5.17 merupakan hasil implementasi *view* pada halaman kelola role. Pada halaman ini ditampilkan jenis role yang telah terdaftar beserta fitur untuk melakukan modifikasi pada role seperti edit nama role, tambah role, hapus role, dan pengubahan akses masing-masing role ke menu. Tampilan penambahan role baru dan edit role ditunjukkan berturut-turut pada gambar 5.18 dan 5.19 berikut.

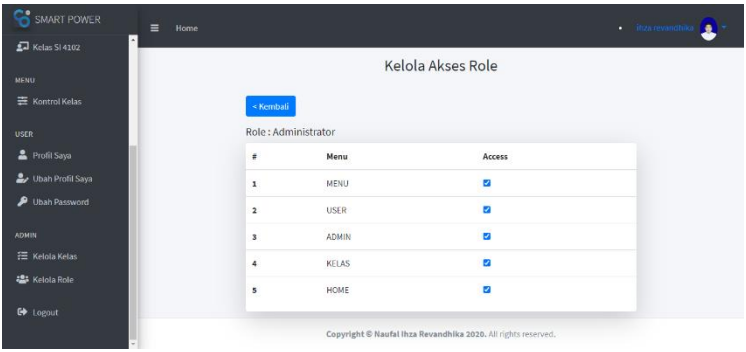


Gambar 5.18 Tampilan Modal Tambah Role



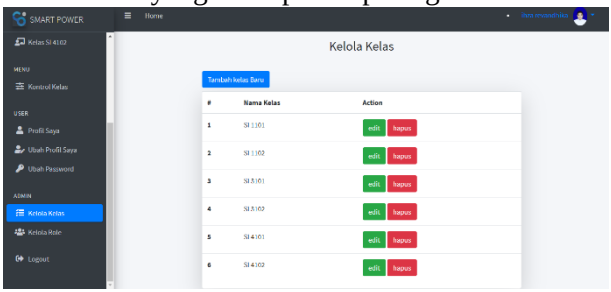
Gambar 5.19 Tampilan Modal Edit Role

Tampilan halaman kelola akses role ditunjukkan pada gambar 5.20 berikut ini. Pada halaman ini *Admin* dapat melakukan perubahan akses menu masing-masing role.



Gambar 5.20 Tampilan Halaman Kelola Akses Role

Selanjutnya terdapat halaman kelola kelas. Pada halaman ini *Admin* dapat mengelola halaman masing-masing kelas. *Admin* dapat menambah kelas baru sesuai kebutuhan maupun menghapus kelas tertentu. Berikut ini adalah tampilan halaman kelola kelas yang ditampilkan pada gambar 5.21.



Gambar 5.21 Tampilan Halaman Kelola Kelas

5.4.4 Implementasi *Database*

Untuk mengakomodir kebutuhan penyimpanan data pada *web* maka diperlukan sebuah *database*. Pada sistem ini database yang diimplementasikan sesuai dengan desain yang telah dijabarkan pada bab sebelumnya, yakni menggunakan database *MySQL*. Dalam pengelolaan database ini digunakan 2 *tools*, yakni *phpMyAdmin* dan *MyQL Workbench* untuk membuat kode pembuatan *database* secara otomatis. Berikut adalah hasil implementasi *data definition language* (DDL) pembuatan *database* yang ditunjukkan pada kode 5.26.

```sql
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-----------------------------------------------
-- Schema smartpower  --
CREATE SCHEMA IF NOT EXISTS `smartpower` DEFAULT CHARACTER SET utf8 ;
USE `smartpower` ;


-- Table `smartpower`.`data_chart`  --
CREATE TABLE IF NOT EXISTS `smartpower`.`data_chart` (
  `id` INT(11) NOT NULL,
  `voltage` FLOAT NOT NULL,
  `current` FLOAT NOT NULL,
  `power` FLOAT NOT NULL,
  `energy` FLOAT(8,3) NOT NULL,
  `date` DATETIME NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4;
-- ------------------------------------------------
-- Table `smartpower`.`kelas` --
CREATE TABLE IF NOT EXISTS `smartpower`.`kelas` (
  `id_kelas` INT(11) NOT NULL,
  `nama_kelas` VARCHAR(16) NOT NULL,
  PRIMARY KEY (`id_kelas`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-------------------------------------------------
-- Table `smartpower`.`data_kelas`--
CREATE TABLE IF NOT EXISTS `smartpower`.`data_kelas` (
  `id` BIGINT(11) NOT NULL,
  `id_kelas` INT(11) NOT NULL,
  `voltage` FLOAT NOT NULL,
```

```
38.   `current` FLOAT NOT NULL,
39.   `power` FLOAT NOT NULL,
40.   `energy` FLOAT(8,3) NOT NULL,
41.   `temperature` FLOAT NOT NULL,
42.   `date` DATETIME NOT NULL,
43.   PRIMARY KEY (`id`),
44.   INDEX `id_kelas` (`id_kelas` ASC) VISIBLE,
45.   CONSTRAINT `data_kelas_ibfk_1`
46.     FOREIGN KEY (`id_kelas`)
47.     REFERENCES `smartpower`.`kelas` (`id_kelas`))
48. ENGINE = InnoDB
49. DEFAULT CHARACTER SET = latin1;
50. -- -----------------------------------------------
51. -- Table `smartpower`.`setting` --
52. CREATE TABLE IF NOT EXISTS `smartpower`.`setting` (

53.   `id` INT(11) NOT NULL,
54.   `id_kelas` INT(11) NOT NULL,
55.   `nama_kelas` VARCHAR(11) NOT NULL,
56.   `mode` INT(11) NOT NULL,
57.   `suhu_min` FLOAT NOT NULL,
58.   `suhu_max` FLOAT NOT NULL,
59.   `durasi_menyala` BIGINT(20) NOT NULL,
60.   PRIMARY KEY (`id`),
61.   INDEX `id_kelas` (`id_kelas` ASC) VISIBLE,
62.   CONSTRAINT `setting_ibfk_1`
63.     FOREIGN KEY (`id_kelas`)
64.     REFERENCES `smartpower`.`kelas` (`id_kelas`))
65. ENGINE = InnoDB
66. DEFAULT CHARACTER SET = latin1;
67. -- -----------------------------------------------
68. -- Table `smartpower`.`user_role` --
69. CREATE TABLE IF NOT EXISTS `smartpower`.`user_role`
    (
70.   `id` INT(11) NOT NULL,
71.   `role` VARCHAR(128) NOT NULL,
72.   PRIMARY KEY (`id`))
73. ENGINE = InnoDB
74. DEFAULT CHARACTER SET = latin1;
75. -- -----------------------------------------------
76. -- Table `smartpower`.`user` --
77. CREATE TABLE IF NOT EXISTS `smartpower`.`user` (
78.   `id` INT(11) NOT NULL,
79.   `nama_user` VARCHAR(128) NOT NULL,
80.   `email` VARCHAR(128) NOT NULL,
```

```
81.    `image` VARCHAR(128) NOT NULL,
82.    `password` VARCHAR(256) NOT NULL,
83.    `role_id` INT(11) NOT NULL,
84.    `is_active` INT(1) NOT NULL,
85.    `date_created` INT(11) NOT NULL,
86.    PRIMARY KEY (`id`),
87.    INDEX `role_id` (`role_id` ASC) VISIBLE,
88.    CONSTRAINT `user_ibfk_1`
89.      FOREIGN KEY (`role_id`)
90.      REFERENCES `smartpower`.`user_role` (`id`))
91.  ENGINE = InnoDB
92.  DEFAULT CHARACTER SET = latin1;
93.  -- -----------------------------------------------
94.  -- Table `smartpower`.`user_menu` --
95.  CREATE TABLE IF NOT EXISTS `smartpower`.`user_menu`
      (
96.    `id` INT(11) NOT NULL,
97.    `menu` VARCHAR(128) NOT NULL,
98.    PRIMARY KEY (`id`))
99.  ENGINE = InnoDB
100. DEFAULT CHARACTER SET = latin1;
101.
102.
103. -- -----------------------------------------------
104. -- Table `smartpower`.`user_access_menu`--
105. CREATE TABLE IF NOT EXISTS `smartpower`.`user_acces
     s_menu` (
106.   `id` INT(11) NOT NULL,
107.   `role_id` INT(11) NOT NULL,
108.   `menu_id` INT(11) NOT NULL,
109.   PRIMARY KEY (`id`),
110.   INDEX `role_id` (`role_id` ASC) VISIBLE,
111.   INDEX `menu_id` (`menu_id` ASC) VISIBLE,
112.   CONSTRAINT `user_access_menu_ibfk_1`
113.     FOREIGN KEY (`role_id`)
114.     REFERENCES `smartpower`.`user_role` (`id`),
115.   CONSTRAINT `user_access_menu_ibfk_2`
116.     FOREIGN KEY (`menu_id`)
117.     REFERENCES `smartpower`.`user_menu` (`id`))
118. ENGINE = InnoDB
119. DEFAULT CHARACTER SET = latin1;
120. -- -----------------------------------------------
121. -- Table `smartpower`.`user_sub_menu` --
122. CREATE TABLE IF NOT EXISTS `smartpower`.`user_sub_m
     enu` (
```

```
123.   `id` INT(11) NOT NULL,
124.   `menu_id` INT(11) NOT NULL,
125.   `title` VARCHAR(128) NOT NULL,
126.   `url` VARCHAR(128) NOT NULL,
127.   `icon` VARCHAR(128) NOT NULL,
128.   `is_active` INT(1) NOT NULL,
129.   PRIMARY KEY (`id`),
130.   INDEX `menu_id` (`menu_id` ASC) VISIBLE,
131.   CONSTRAINT `user_sub_menu_ibfk_1`
132.     FOREIGN KEY (`menu_id`)
133.     REFERENCES `smartpower`.`user_menu` (`id`))
134. ENGINE = InnoDB
135. DEFAULT CHARACTER SET = latin1;
136. -- -----------------------------------------------
137. -- Table `smartpower`.`user_token` --
138. CREATE TABLE IF NOT EXISTS `smartpower`.`user_token
     ` (
139.   `id` INT(11) NOT NULL,
140.   `email` VARCHAR(128) NOT NULL,
141.   `token` VARCHAR(128) NOT NULL,
142.   `date_created` INT(11) NOT NULL,
143.   PRIMARY KEY (`id`))
144. ENGINE = InnoDB
145. DEFAULT CHARACTER SET = utf8mb4;
146.
147.
148. SET SQL_MODE=@OLD_SQL_MODE;
149. SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
150. SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Kode 5.26 Implementasi DDL Database

5.5 Hasil Pengujian Sistem

Pada proses pengujian sistem dilakukan sesuai rancangan yang telah dijabarkan pada bab 4 sebelumnya, yakni menggunakan metode *black box testing*. Proses pengujian sistem terbagi menjadi 2 bagian, yakni pengujian *hardware* dan *software*.

5.5.1 Hasil Pengujian *Hardware*

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.

Tabel 5.2 Hasil Pengujian Hardware Sesuai Test Case

Test case	**Skenario**	***Ekspektasi***	**Hasil**
Koneksi WiFi Arduino	Menghubungkan mikrokontroller Arduino ke Wi-Fi	Arduino dapat terhubung ke jaringan Wi-Fi dan mendapatkan *IP Address*	OK
Pembacaan sensor PIR	Terdapat objek yang bergerak di dalam kelas	Arduino dapat mendeteksi adanya gerakan dan memberikan sinyal ke *relay* sesuai setting	OK
Pembacaan sensor PZEM-004T	Sensor PZEM-004T terhubung ke sambungan arus listrik AC masing-masing kelas	Hasil pembacaan sensor dapat ditampilkan pada layar LCD	OK
Pembacaan sensor suhu DS18B20	Sensor suhu terpasang di masing-masing kelas	Hasil suhu masing-masing kelas dapat ditampilkan di layar LCD	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 ke *web*	Mengirimkan data hasil pembacaan sensor PZEM-004T dan DS18B20 ke web menggunakan HTTP POST	Data berhasil terkirim, *respons code* 200 OK	OK
Penerimaan data setting dari *web*	Menerima data setting dan *threshold* dari web menggunakan HTTP GET	Data berhasil diterima oleh Arduino, *response code* 200 OK, setting pada Arduino berubah sesuai data yang diterima	OK
Respon *relay* dalam menerima *trigger* dari sensor PIR	Relay terhubung dengan Arduino dan sensor PIR, serta menggunakan mode Auto-cut / Adaptif	Relay dapat aktif dan non-aktif sesuai perintah dari Arduino	OK
Tampilkan data ke LCD	Arduino mengirimkan data kelistrikan, suhu, dan mode untuk ditampilkan ke LCD	LCD dapat menampilkan data kelistrikan, suhu, dan mode	OK

Tabel 5.2 di atas merupakan hasil dari rangkaian pengujian *hardware* yang dilakukan mengacu pada *test case*. Secara keseluruhan, sistem *hardware* dapat memenuhi kriteria ekspektasi. Mikrokontroler Arduino dapat tersambung pada koneksi *Wi-Fi* dengan baik dan mendapatkan *IP address*.

Pengujian terhadap sensor PIR juga dilakukan dan menghasilkan respon yang baik ketika diuji terdapat objek yang bergerak di dalam kelas. Hasil data yang didapatkan dari masing-masing sensor PZEM-004T dan sensor suhu DS18B20 juga sesuai ekspektasi. Kedua sensor dapat menghasilkan data yang akurat dan berhasil ditampilkan pada layar LCD. Sensor infrared dapat menerima masukan setelah dilakukan uji coba dengan memberikan masukan melalui remote, hasilnya layar LCD dapat berubah sesuai masukan yang diberikan. Masing-masing data pada sensor dikirimkan ke web menggunakan HTTP POST, data tersebut berhasil diterima pada web dan tersimpan di dalam database. Penerimaan data setting Arduino berhasil diterima setelah mengirimkan request HTTP GET ke aplikasi web, Arduino dapat berjalan sesuai dengan setting yang diterima dari web. Performa *relay* dalam menerima input dari Arduino juga baik, ditandai dengan aktif/non-aktif ketika mendapat masukan dari Arduino. Seluruh data baik dari sensor suhu, sensor kelistrikan dan pengaturan mode berhasil ditampilkan pada layar LCD. Berikut adalah contoh hasil output pada layar LCD yang ditunjukkan pada gambar 5.22.

Gambar 5.22 Hasil Output Pada Layar LCD

5.5.1.1 Uji Sensitivitas Sensor

Proses pengujian dan kalibrasi juga dilakukan pada sensor untuk mengetahui tingkat akurasi pada sensor suhu DS18B20, PZEM-004T, dan tingkat sensitifitas terhadap objek pada sensor PIR.

Proses kalibrasi sensor suhu dilakukan dengan membandingkan hasil bacaan sensor DS18B20 dengan alat *non-contact infrared thermometer*. Berikut adalah hasil bacaan suhu pada alat *non-contact infrared thermometer* yang dilakukan di 2 tempat berbeda.

Tabel 5.3 Hasil Bacaan Suhu pada Thermometer

Suhu Titik Awal	Suhu pada Jarak 6m
30,4°C	30,6°C

Berikut adalah hasil pengujian sensor DS18B20 yang ditampilkan pada tabel 5.4, 5.5, dan 5.6.

Tabel 5.4 Hasil Pengujian Akurasi Sensor Suhu

Sensor	Suhu Titik Awal	Suhu pada Jarak 6m	Perbedaan Suhu
Sensor 1	30,25°C	30,31°C	0,06°C
Sensor 2	30,56°C	30,69°C	0,13°C
Sensor 3	30,50°C	30,62°C	0,12°C
Sensor 4	30,44°C	30,56°C	0,12°C
Sensor 5	30,12°C	30,31°C	0,19°C
Sensor 6	30,56°C	30,69°C	0,13°C
Rata-rata perbedaan suhu			0,125°C

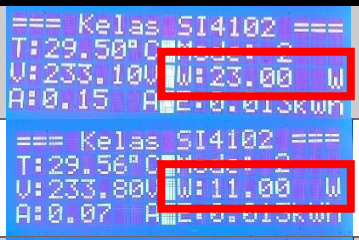
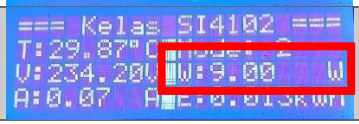
Tabel 5.3 di atas merupakan hasil pengujian akurasi sensor suhu *DS18B20.* Pengujian sensor ini dilakukan dengan kondisi lingkungan di dalam ruangan tanpa pendingin. Teknis pengujian dilakukan selama 2 kali percobaan. Pertama masing-masing sensor diukur nilai bacaan suhunya, selanjutnya dilakukan pembacaan nilai suhu pada jarak 6m dari titik sebelumnya. Hasilnya adalah dengan jarak 6m di dalam ruangan, dihasilkan rata-rata perbedaan suhu seluruh sensor sebesar 0,125°C. Perbedaan bacaan antar sensor yang terbesar berada pada angka 0,44°C, hasil ini masih berada di bawah klaim pabrik yang berada pada angka ±0,5°C.

Tabel 5.5 Hasil Pengujian Sensitivitas Sensor PIR

Objek	Jarak	Respon
Manusia	1m	Aktif – stabil
	2m	Aktif – stabil
	3m	Aktif – stabil
	4m	Aktif – stabil
	5m	Aktif – stabil
	6m	Aktif – kurang stabil
	7m	Aktif – kurang stabil
	>7m	Tidak aktif
Bola	1m	Tidak Aktif
	>1m	Tidak Aktif

Tabel 5.5 merupakan hasil dari pengujian sensitivitas sensor PIR. Setelah dilakukan pengujian terhadap sensor PIR, didapatkan hasil bahwa sensor PIR efektif dapat menerima rangsangan gerakan hanya terhadap manusia / objek yang memiliki suhu yang berbeda dibandingkan dengan lingkungan sekitar. Jarak optimal pembacaan sensor PIR berada pada radius 1 meter - 5 meter. Implementasi sensor PIR pada modul miniatur kelas perlu dilakukan sedikit modifikasi agar dapat menerima rangsangan, yakni dengan menutup sebagian *fresnel lens.* Penutupan dilakukan dengan tujuan untuk mengurangi masukan rangsangan yang diterima dari sisi atas sensor. Jumlah sensor yang digunakan menyesuaikan kebutuhan pada kelas sesungguhnya, yakni menggunakan 2 sensor di tiap kelas.

Tabel 5.6 Hasil Pengujian Sensitivitas Sensor PZEM-004T

No	Jenis Beban	Daya klaim	Daya terbaca
1	Lampu dop	23 Watt	
2	Lampu LED in-lite	11 Watt	
3	Lampu LED holz	9 Watt	

Tabel 5.6 merupakan hasil uji sensitivitas / akurasi sensor PZEM-004T. Pengujian dilakukan dengan memanipulasi pada variabel daya beban yang terhubung pada sensor. Dari ketiga percobaan yang telah dilakukan, didapatkan hasil bahwa sensor PZEM-004T dapat membaca daya yang digunakan masing-masing beban dengan akurat sesuai dengan klaim pabrik pada masing-masing produk. Hasil ini sesuai dengan spesifikasi sensor dengan tingkat error pembacaan di bawah 0,5%.

5.5.2 Hasil Pengujian *Software*

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 5.7 Hasil Pengujian Software Sesuai Test Case

Test case	Skenario	*Ekspektasi*	Hasil
Login	Pengguna memasukkan *email* dan *password* akun yang sudah	Pengguna dapat masuk ke ke halaman utama dan mengakses fitur sesuai *role*	OK

	aktif dengan benar		
	Pengguna memasukkan *email* dan *password* akun yang sudah aktif dengan salah	Pengguna tidak dapat masuk ke halaman utama, terdapat pemberitahuan salah *input email / password*	OK
	Pengguna memasukkan *email* dan *password* akun yang belum diaktivasi	Pengguna tidak dapat masuk ke halaman utama, terdapat pemberitahuan akun belum diaktivasi	OK
Registrasi akun	Pengguna memasukkan data yang diminta pada form dengan benar	Terdapat pemberitahuan akun berhasil dibuat	OK
	Pengguna memasukkan data yang diminta pada form salah / kurang lengkap	Terdapat pemberitahuan *warning* pada form yang bermasalah	OK
Penerimaan data dari Arduino	Arduino telah mengirimkan *HTTP request* untuk mengirimkan data dengan key dan parameter yang sesuai	Data tersimpan pada *database* web dan mengirimkan respon ke Arduino	OK

	Arduino telah mengirimkan *HTTP request* untuk mengirimkan data dengan key atau parameter yang tidak sesuai	Data gagal tersimpan di *database* dan mengirimkan respon ke Arduino	OK
Pengiriman data setting ke Arduino	Arduino telah mengirimkan *HTTP request* untuk meminta data setting dengan *parameter* yang sesuai	Web merespons dengan mengirimkan data setting ke Arduino	OK
	Arduino telah mengirimkan *HTTP request* untuk meminta data setting dengan *parameter* yang tidak sesuai	Data gagal dikirim ke Arduino, web mengirimkan respons kesalahan	OK
Kelola *role*	*Admin* telah *login* menggunakan akun *admin* dan mengakses fitur kelola *role*	*Admin* dapat melakukan modifikasi pada *role* yang telah terdaftar	OK
Kelola daftar kelas	Admin telah *login* menggunakan akun *admin* dan mengakses	*Admin* dapat melakukan modifikasi pada daftar kelas	OK

	fitur kelola daftar kelas		
Kontrol sistem kelistrikan kelas	Admin / *user* telah *login* dan mengakses fitur kontrol kelas dengan memberikan *input* yang benar pada *form*	Admin / user dapat melakukan modifikasi pada setting kelistrikan masing-masing kelas	OK
	Admin / *user* telah login dan mengakses fitur kontrol kelas dengan memberikan input yang salah pada *form*	Web gagal memperbarui setting dan memberikan notifikasi kesalahan *input*	OK
Tampilkan halaman fitur sesuai role	*Admin* telah login menggunakan akun *admin* dan mengakses fitur *admin*	web menampilkan halaman fitur yang dituju dan memperbolehkan *admin* untuk memodifikasi	OK
	User login menggunakan akun user untuk mengakses fitur user	web menampilkan halaman fitur yang dituju dan memperbolehkan *user* untuk memodifikasi	OK
	User selain *admin* mengakses halaman fitur yang dimiliki oleh admin	Web menampilkan halaman 403 forbidden, user dilarang mengakses	OK

	langsung melalui alamat *link*	halaman yang dituju.	
Tampilkan halaman *dashboard* utama dan masing-masing kelas	*Admin / user* telah *login* dan mengakses halaman dashboard utama / dashboard masing-masing kelas	Web menambilkan halaman dashboard sesuai yang dituju	OK
User mengakses langsung halaman web tanpa *login*	*User* mengakses halaman web yang telah diketahui sebelumnya langsung tanpa *login*	Web menampilkan halaman login	OK
Akses halaman *login* ketika sudah *login*	*User / admin* mengakses halaman login ketika sudah login	Web menampilkan halaman utama	OK

Tabel 5.7 di atas merupakan hasil dari pengujian sistem aplikasi web yang dilakukan dengan menjalankan seluruh test case. Hasilnya aplikasi web yang dikembangkan dapat menjalankan seluruh fitur dengan baik dan telah memenuhi ekspektasi.

Halaman ini sengaja dikosongkan

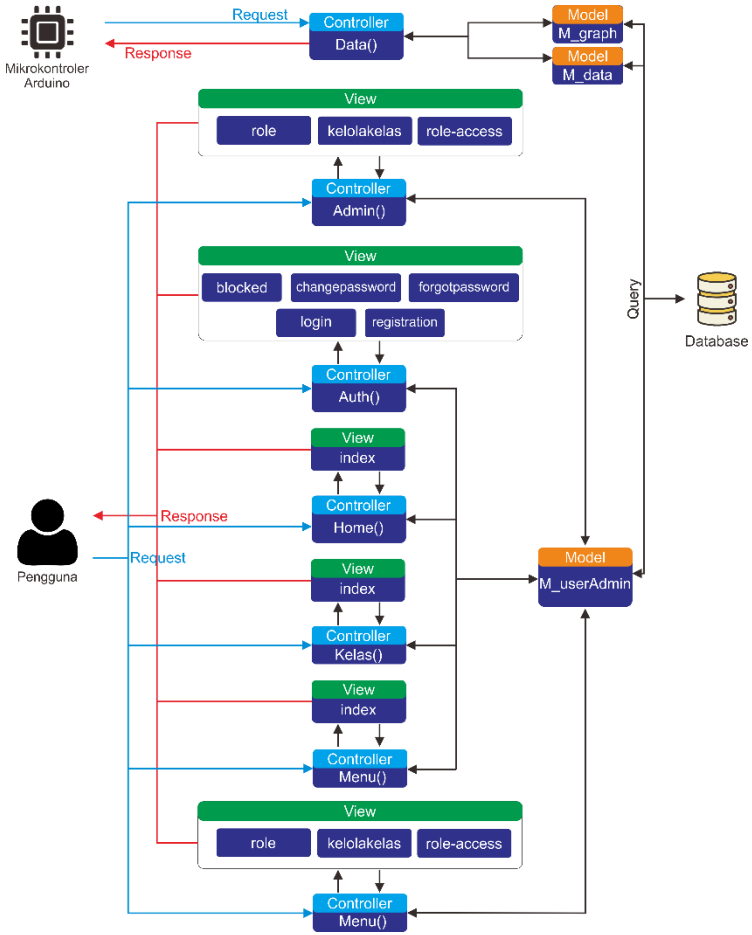
BAB VI
HASIL DAN PEMBAHASAN

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

6.1 Arsitektur Sistem

Arsitektur sistem merupakan sebuah gambaran untuk mendefinisikan sebuah sistem dapat bekerja oleh seluruh komponen yang terkait secara terstruktur. Sesuai pada tahap perencanaan, sistem ini terbagi menjadi 2 bagian yakni *software* dan *hardware*. Kedua bagian tersebut harus saling terhubung agar sistem secara keseluruhan dapat bekerja. Secara garis besar, sistem ini dikembangkan dengan menggunakan *design pattern Model-View-Controller* (MVC) sesuai *framework* Codeigniter yang digunakan.

Total terdapat 7 controller pada sistem ini, masing-masing *controller* memiliki fungsi yang digunakan untuk menjalankan fitur yang dimiliki sistem Smart Power ini. *Controller* berfungsi untuk menerima request, memproses, dan mengirimkan respon ke *client*. Masing-masing controller terhubung langsung ke *view* untuk tampilan ke pengguna dan *model* yang digunakan untuk keperluan pengolahan query data pada database. *Client* sistem ini terbagi menjadi 2 macam, yakni *user*/*admin* sebagai pengguna yang menjalankan sistem melalui *view* dan Mikrokontroler Arduino yang terhubung ke web untuk melakukan pertukaran data. Ketika *user*/*admin* mengirimkan request ke controller, respon balik yang diberikan oleh controller berupa view hasil pemrosesan. Sedangkan untuk Mikrokontroler Arduino ketika mengakses *controller* Data(), respon balik yang diberikan oleh *controller* berupa data berbentuk JSON. Berikut adalah gambar arsitektur sistem yang ditunjukkan oleh gambar 6.1.



Gambar 6.1 Arsitektur Sistem

6.2 Hasil Pengujian Modul Miniatur Kelas

Modul miniatur kelas merupakan media yang digunakan untuk mensimulasikan sistem yang diajukan sesuai dengan kondisi aslinya yang telah disesuaikan. Modul ini diatur oleh Mikrokontroler Arduino dalam pengoperasiannya. Proses pengujian akhir pada modul miniatur kelas ini terbagi menjadi beberapa proses. Berikut adalah hasil pengujian yang telah dilakukan.

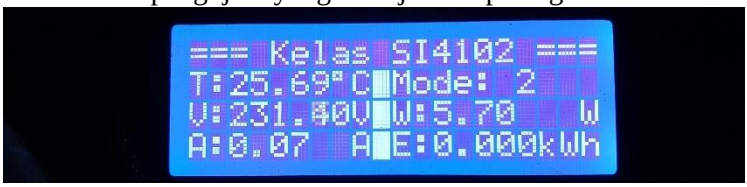
6.2.1 Pengujian Mode

Terdapat 3 mode pada sistem ini, yakni mode Adaptif (0), mode Auto-Cut (1), dan mode Manual. Mode Adaptif merupakan mode yang dapat digunakan untuk menghemat penggunaan energi dengan mematikan perangkat listrik ketika tidak terdeteksi aktivitas di dalam kelas namun dengan tetap menjaga suhu ruangan sesuai yang diinginkan. Mode Auto Cut merupakan mode paling hemat, dengan mematikan seluruh perangkat listrik ketika tidak ada aktivitas. Mode Manual merupakan mode dimana otomasi kontrol pada kelistrikan kelas dimatikan.

6	SI 4102	Adaptif	26	28	60000	edit
6	SI 4102	Auto Cut	26	28	60000	edit
6	SI 4102	Manual	26	28	60000	edit

Gambar 6.2 Tampilan Setting pada Web

Gambar 6.2 memperlihatkan setting mode Arduino pada halaman web. Untuk memastikan pertukaran data dapat berjalan sempurna, dilakukan pengujian dengan mengganti mode pada kelas yang sama. Bila mode pada Arduino dapat ter-*update* maka proses pertukaran data dapat berjalan dengan baik. Berikut adalah hasil pengujian yang ditunjukkan pada gambar 6.3 – 6.5.

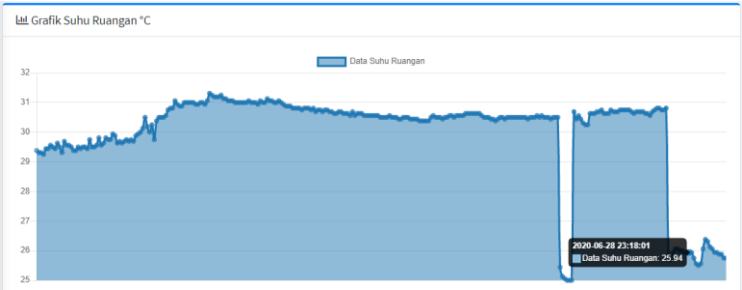


Gambar 6.3 Hasil Tampilan Mode Manual pada LCD

Gambar 6.4 Hasil Tampilan Mode Auto Cut pada LCD

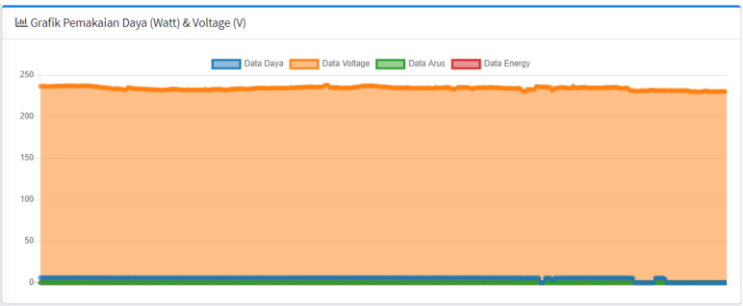
Gambar 6.5 Hasil Tampilan Mode Adaptif pada LCD

Hasil pengujian menunjukkan setting mode pada Arduino dapat ter-*update* dan berhasil mengoperasikan mode kelas sesuai settingnya. Arduino melakukan request ke web untuk menerima setting setiap 15 detik, sedangkan untuk pengiriman data dilakukan setiap 2 menit.



Gambar 6.6 Hasil Grafik Data Suhu

Gambar 6.6 di atas merupakan hasil tampilan grafik data suhu yang diupload dan ditampilkan pada laman web sesuai data pada gambar 6.5 yang ditampilkan di layar LCD. Informasi kelistrikan juga berhasil ditampilkan pada grafik yang ditunjukkan pada gambar 6.7 berikut ini.



Gambar 6.7 Hasil Grafik Kelistrikan Kelas

6.2.2 Pengujian Respon Sensor

Untuk menguji keberadaan aktivitas di dalam modul miniatur kelas ini dibutuhkan sebuah objek pengganti. Objek pengganti yang digunakan adalah dengan menggunakan lampu yang telah dimodifikasi. Penggunaan lampu sebagai objek pengganti adalah dikarenakan lampu akan memancarkan suhu yang hangat ketika aktif sehingga dapat terdeteksi oleh sensor PIR saat bergerak. Pengujian dilakukan dengan setting mode kelistrikan pada mode Adaptif / Auto Cut. Berikut adalah hasil pengujiannya yang ditunjukkan pada gambar 6.8 dan 6.9

Gambar 6.8 Kondisi Sebelum Objek Dimasukkan

Gambar 6.9 Kondisi Sesudah Objek Dimasukkan

Dari hasil pengujian yang ditampilkan pada gambar 6.8 dan 6.9 didapat hasil bahwa sensor PIR dapat menangkap respon gerakan dengan sempurna.

6.3 Pengujian Kehematan Sistem

Pengujian kehematan dilakukan dengan tujuan untuk mengetahui tingkat efisiensi yang dihasilkan dari sistem untuk membantu menekan penggunaan energi listrik di lingkungan Departemen Sistem Informasi ITS.

6.3.1 Kondisi dan Asumsi Pengujian

Proses uji kehematan dilakukan dengan menjalankan modul miniatur kelas sistem Smart Power pada mode *Auto-Cut* dan Manual sesuai dengan jam operasional kelistrikan di lingkungan DSI (pukul 07.00 WIB – pukul 16.00 WIB). Pengujian dilakukan dengan menjalankan 2 mode, yakni mode Auto-cut dan Adaptif. Mode Auto-cut merupakan mode paling hemat pada sistem ini, dengan memutus aliran listrik pada kelas secara keseluruhan apabila tidak terdeteksinya aktivitas. Pada pengujian dengan mode Auto-cut, manipulasi pengujian dilakukan dengan

mengubah durasi timer menyala listrik sesuai dengan lama sesi perkuliahan. Saat objek terdeteksi oleh sensor pir, sistem akan mengatur listrik tetap nyala ketika jam perkuliahan berlangsung dan akan mati bila diluar jam perkuliahan. Mode Manual merupakan mode yang paling boros, yakni dengan menggunakan listrik seperti pada sistem konvensional tanpa adanya otomasi untuk kehematan berdasarkan sensor. Mode Adaptif merupakan mode yang akan memutus arus listrik pada kelas apabila tidak terdeteksinya aktivitas namun dengan tetap menjaga suhu ruangan. Tingkat kehematan pada mode Adaptif sangat bergantung dengan *threshold* suhu yang diatur untuk memicu pendingin ruangan aktif dan aktifitas di dalam kelas. Pada mode Adaptif tidak dilakukan pengujian tingkat kehematan dikarenakan hasilnya dapat berubah-ubah tergantung setelan yang diterapkan. Berikut adalah detil kondisi pengujian yang diterapkan pada masing-masing kelas maket yang ditunjukkan pada tabel 6.1.

Tabel 6.1 Kondisi Pengujian Maket

No	Perangkat Elektronik	Jumlah	Daya
1	Lampu LED	1	3 Watt
2	Display LED	1	1-3 Watt
3	Fan (Pendingin Ruangan)	1	1 Watt
Total			5-7 watt

Berikut adalah kondisi umum asumsi kelas pada Departemen Sistem Informasi ITS yang ditunjukkan pada tabel 6.2.

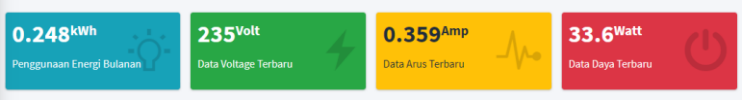
Tabel 6.2 Kondisi Kelas DSI

No	Perangkat Elektronik	Jumlah	Daya
1	Lampu TL Neon	10	360 Watt
2	Display LCD Proyektor	1	300-500 Watt
3	AC 2PK	2	3000 Watt
Total			± 3500 – 4000 Watt

Pengujian ini hanya untuk mengetahui besaran penghematan minimum yang dapat dicapai oleh sistem bila diterapkan pada kondisi riil. Pada maket ini simulasi dilakukan dengan mengabaikan banyaknya jumlah manusia di dalam kelas yang berpotensi mempengaruhi suhu ruangan dan kehematan yang dicapai. Keberadaan manusia di dalam kelas digantikan dengan objek pengganti, sehingga sensor PIR masih dapat mendeteksi sesuai yang telah dijelaskan pada poin 6.2.2 sebelumnya. Bila dilihat dari kedua tabel perbandingan kondisi simulasi & riil (tabel 6.1 dan tabel 6.2), terdapat perbedaan konsumsi daya listrik yang cukup besar. Hasil besaran persentase penghematan energi listrik pada sistem yang didapatkan dapat berpotensi lebih besar bila diterapkan pada kondisi riil dikarenakan perbedaan penggunaan daya dan faktor lain seperti penggunaan AC yang konsumsi daya listriknya cukup besar pada saat awal pengoperasiannya.

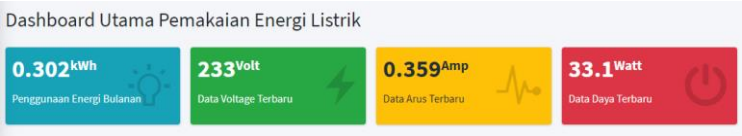
6.3.2 Hasil Uji Kehematan

Pada mode *Auto-Cut* dalam menjalankan sistem kelistrikannya mencatatkan hasil penggunaan energi listrik sebesar 0,248 kWh. Berikut adalah hasil pengujian kehematan pada mode *Auto-Cut* yang ditunjukkan pada gambar 6.10 di bawah ini.



Gambar 6.10 Hasil Penggunaan Mode Auto-Cut

Pada mode Manual juga dilakukan pengujian dengan metode yang sama, yakni menjalankan modul miniatur kelas Smart Power pada jam operasional kelistrikan DSI dengan mode Manual. Hasil penggunaan energi pada mode ini mencatatkan sebesar 0,302 kWh. Berikut adalah hasil pengujian kehematan pada mode Manual yang ditunjukkan pada gambar 6.11 di bawah ini.



Gambar 6.11 Hasil Penggunaan Mode Manual

Dari hasil pengujian kedua mode ini selanjutnya dilakukan penghitungan penghematan energi dan persentasenya. Berikut adalah perhitungan penghematan energi yang dilakukan dengan menggunakan persamaan (1).

$$\begin{aligned} ES &= E_{d,el}^{P} - EC_{SP} \\ &= 0{,}302 - 0{,}248 \\ &= 0{,}054 \text{ kWh} \end{aligned}$$

Dari hasil perhitungan nilai penghematan energi yang telah didapatkan, selanjutnya dilakukan perhitungan persentase penghematan energi listrik dengan menggunakan persamaan (2).

$$\begin{aligned} \%ES &= \frac{ES}{E_{d,el}^{P}} \times 100\% \\ &= \frac{0{,}054}{0{,}302} \times 100\% \\ &= 17{,}88\% \end{aligned}$$

Dengan demikian terbukti bahwa penggunaan otomasi pada sistem kelistrikan dapat meningkatkan efisiensi penggunaan energi listrik. Besar peningkatan efisiensi penggunaan mode *Auto-Cut* dibandingkan dengan Manual berada pada angka 17,8%. Capaian ini merupakan angka minimal yang dapat diraih oleh sistem.

Halaman ini sengaja dikosongkan

BAB VII
KESIMPULAN DAN SARAN

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

7.1 Kesimpulan

Berikut merupakan kesimpulan yang dapat diambil dari pengembangan perangkat lunak yang telah dilakukan.

1. Sistem yang dibuat terbukti dapat menekan penggunaan energi listrik yang terbuang sebesar 17,8% pada maket. Angka ini merupakan penghematan minimum yang bisa dicapai dengan mode Auto-Cut.
2. Besarnya penghematan mode Adaptif sangat dipengaruhi oleh *threshold* suhu ruangan yang ditetapkan dan adanya aktivitas, sedangkan pada mode Auto-cut besarnya penghematan hanya dipengaruhi oleh deteksi aktivitas saja.
3. Besar penghematan energi yang dihasilkan oleh sistem dapat berpotensi lebih besar bila diterapkan pada kondisi riil dikarenakan konsumsi daya elektronik yang digunakan pada kondisi riil jauh lebih besar bila dibandingkan dengan maket.

7.2 Saran

Berikut merupakan saran untuk pengembangan lebih lanjut yang dapat dilakukan dalam menyempurnakan pengembangan perangkat lunak.

1. Pengembangan sistem dengan menggunakan *multiple* mikrokontroler, yakni menempatkan 1 Arduino pada masing-masing kelas yang terhubung ke Arduino utama.
2. Peningkatan keamanan pada transfer data antara Arduino dengan web menggunakan *Json Web Token* (JWT).
3. Pengembangan aplikasi *mobile* Smart Power untuk menampilkan informasi dan pengontrolan sistem kelistrikan melalui perangkat genggam.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] P. E. Wicaksono, “Konsumsi Listrik Terus Meningkat, RI Menuju Negara Maju?,” Liputan6, 05 Januari 2019. [Online]. Available: https://www.liputan6.com/bisnis/read/3863789/konsumsi-listrik-terus-meningkat-ri-menuju-negara-maju. [Diakses 20 Oktober 2019].

[2] Direktorat Konservasi Energi Kementrian Energi dan Sumber Daya Mineral RI, EDISI III DATA & INFORMASI KONSERVASI ENERGI 2018, Jakarta, 2018.

[3] N. Y. Dahlan, A. Aris, M. Saidin, M. Nadzeri, M. Nawi, W. F. Abas, A. F. Abidin, H. Mohammad, Z. Zakaria dan P. Arshad, “Development of Web-Based Real-time Energy Monitoring System for Campus University,” *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 8, no. 10, 2016.

[4] C. Twumasi, K. A. Dotche, W. Banuenumah dan F. Sekyere, “Energy Saving System using a PIR Sensor for Classroom Monitoring,” *IEEE PES-IAS PowerAfrica*, 2017.

[5] J. Chandramohan, R. Nagarajan, K. Satheeshkumar, N. Ajithkumar, P. A. Gopinathh dan S. Ranjithkumar, “Intelligent Smart Home Automation and Security System Using Arduino and Wi-fi,” *International Journal Of Engineering And Computer Science*, vol. 6, no. 3, 2017.

[6] H. A. Gabbar dan K. Sayed, “Building Energy Management Systems (BEMS),” dalam *Energy Conservation in Residential, Commercial, and Industrial Facilities*, John Wiley & Sons, Inc, 2018, pp. 15-81.

[7] A. K. Sikder, G. Petracca, H. Aksu, T. Jaeger dan A. S. Uluagac, “A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications”.

[8] L. Louis, “WORKING PRINCIPLE OF ARDUINO AND USING IT AS A TOOL FOR STUDY AND RESEARCH,” *International Journal of Control, Automation, Communication and Systems (IJCACS),* vol. 1, no. 2, 2016.

[9] Y. A. Badamasi, “The Working Principle Of An Arduino,” *2014 11th International Conference on Electronics, Computer and Computation (ICECCO),* pp. 1-4, 2014.

[10] F. Koyanagi, “Arduino MEGA 2560 With WiFi Built-in - ESP8266: Instructables,” [Online]. Available: https://www.instructables.com/id/Arduino-MEGA-2560-With-WiFi-Built-in-ESP8266/. [Diakses 6 November 2019].

[11] InnovatorsGuru, “PZEM-004T V3,” [Online]. Available: https://innovatorsguru.com/pzem-004t-v3/. [Diakses 15 Mei 2020].

[12] F. Xiong, “Wireless temperature sensor network based on DS18B20, CC2420, MCU AT89S52,” *2015 IEEE International Conference on Communication Software and Networks (ICCSN),* pp. 294-298, 2015.

[13] D. A. O. Turang, “PENGEMBANGAN SISTEM RELAY PENGENDALIAN DAN PENGHEMATAN PEMAKAIAN LAMPU BERBASIS MOBILE,” *Seminar Nasional Informatika 2015 (semnasIF 2015),* 2015.

[14] Oracle, “What Is a Database?,” Oracle, 2019. [Online]. Available: https://www.oracle.com/database/what-is-database.html. [Diakses 29 September 2019].

[15] CodeIgniter, “CodeIgniter Overview,” 19 September 2019. [Online]. Available:

https://codeigniter.com/user_guide/overview. [Diakses 27 Oktober 2019].

[16] J. L. Taulo dan A. B. Sebitosi, “Energy consumption analysis for the Malawian tea industry,” *2015 International Conference on the Industrial and Commercial Use of Energy (ICUE),* 2015.

[17] Components101, “HC-SR501 PIR Sensor,” 18 September 2017. [Online]. Available: https://components101.com/hc-sr501-pir-sensor. [Diakses 10 November 2019].

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis bernama Naufal Ihza Revandhika, lahir di Malang pada tanggal 26 Juli 1998, merupakan anak kedua dari tiga bersaudara. Penulis telah menempuh jenjang pendidikan formal di beberapa sekolah, yaitu : SD Muhammadiyah 1 Sidoarjo (2004 – 2010), SMP Negeri 3 Sidoarjo (2010 - 2013), dan SMA Negeri 1 Sidoarjo (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 05211640000034.

Selama menjadi mahasiswa, penulis aktif mengikuti organisasi kemahasiswaan, seperti UKM Robotika pada tahun 2016-2018, Tim robot *soccer humanoid* ICHIRO pada tahun 2016-2018 serta kegiatan kepanitiaan, seperti Information Systems Expo 2018. Penulis juga pernah meraih penghargaan pada kompetisi nasional dan internasional selama bergabung dengan tim ICHIRO, seperti juara 1 Kontes Robot Indonesia (Regional IV), juara 2 Kontes Robot Indonesia (Nasional), dan juara umum pada kejuaraan robot FIRA RoboWorldCup 2017 di Taiwan. Selain penghargaan pada perlombaan, penulis juga menjadi peraih beasiswa SPARK Scholarship Schneider pada tahun 2018.

Pada Juni - September 2019, penulis melaksanakan kegiatan *internship* di PT. Schneider Electric Manufacturing Batam sebagai *Project Leader* pada Unit Digital Transformation. Penulis dapat dihubungi melalui email Naufalihza20@gmail.com.