

DISERTASI

**MODEL SISTEM SURVEILAN OTOMATIS TERINTEGRASI
DAN DINAMIS UNTUK OPERASIONAL
PERANGKAT-PERANGKAT IOT KELISTRIKAN**

***INTEGRATED AND DYNAMIC AUTOMATIC SURVEILLANCE
SYSTEM MODEL FOR OPERATIONS OF
ELECTRICAL IOT DEVICES***

MULIADI

NIM : D 053171001



**PROGRAM STUDI S3 TEKNIK ELEKTRO
DEPARTEMEN TEKNIK ELEKTRO
UNIVERSITAS HASANUDDIN
MAKASSAR
2022**

HALAMAN PENGESAHAN**MODEL SISTEM SURVEILAN OTOMATIS TERINTEGRASI DAN
DINAMIS UNTUK PENGOPERASIAN PERANGKAT-PERANGKAT IOT
KELISTRIKAN**

Diajukan Oleh:

**MULIADI
D053171001**Telah diperiksa dan dinyatakan memenuhi syarat untuk melaksanakan ujian
Pra-Promosi

Menyetujui

Tim Promotor


Prof. Dr. Ir. H. Andani Achmad, M.T
NIP. 196012311987031002
Dr. Elias Palantel, S.T., M. Eng
NIP. 196012121987031022
Dr. Intan Sari Anehi, S.T., M.T
NIP. 197502032000122002Ketua Program Studi
S3 Teknik Elektro

Prof. Dr. Ir. H. Andani Achmad, M.T
NIP. 196012311987031002

PRAKATA

Segala puji kepada Allah SWT Yang Maha Kuasa yang telah memberikan rahmat dan pertolongan-Nya. Serta salawat dan salam kepada Nabi Muhammad SAW atas perjuangan untuk menyebarkan keindahan ilmu dan akhlak kepada umat manusia.

Ucapan terima kasih penulis haturkan atas bantuan dan motivasi dalam menyelesaikan penelitian dan Disertasi dengan judul “**Model Sistem Surveilan Otomatis Terintegrasi Dan Dinamis Untuk Operasional Perangkat-Perangkat IoT Kelistrikan**” sebagai salah satu syarat penyelesaian Pendidikan pada Program Doktor di Departemen Teknik elektro Fakultas Teknik Universitas Hasanuddin.

1. Prof. Dr. Ir. H. Andani Achmad, MT. selaku Promotor yang telah banyak meluangkan waktunya dalam membimbing dan mengarahkan penulis dari awal penelitian hingga selesainya disertasi ini. Terima kasih atas segala jasa dan keikhlasan yang tak ternilai, semoga Allah SWT memberikan imbalan pahala berupa amal jariah.
2. Dr. Elyas Palantei, ST.,M. Eng. dan Dr. Eng. Intan Sari Areni, ST., MT. selaku Co-Promotor yang telah banyak meluangkan waktunya dalam membimbing dan mengarahkan penulis dalam menyelesaikan kendala yang ada dengan memberikan saran – saran, arahan, petunjuk dari awal penelitian hingga selesainya disertasi ini. Terima kasih atas segala jasa dan keikhlasan yang tak ternilai, semoga Allah SWT memberikan imbalan pahala berupa amal jariah.

3. Prof. Ir. Adit Kurniawan, M.Eng., Ph.D selaku penguji eksternal. Terima kasih atas kesediaannya menjadi Penguji dalam Ujian Disertasi.
4. Bapak Rektor Universitas Hasanuddin, Dekan Fakultas Teknik beserta wakil dekan dan seluruh staf. Ketua program Studi S3 Teknik Elektro Prof. Dr. Ir. H. Andani Achmad, MT. dan seluruh staf. Terima kasih atas segala bantuan waktu dalam memberikan pelayanan terbaik
5. Para sahabat, teman seangkatan maupun sesama mahasiswa S3 Teknik Elektro UNHAS yang tidak tertulis namanya. Terima kasih atas bantuannya selama ini, semoga kebersamaan kita tetap terjalin dan terjaga

Ucapan terkasih yang sebesar-besarnya kepada orang tua, istri dan beserta seluruh keluarga yang telah mendukung dan bersabar dalam perjalanan studi penulis. Begitu banyak hal-hal yang menjadi motivasi dan bantuan yang telah diberikan selama proses pengerjaan disertasi ini.

Penulis menyadari bahwa disertasi ini masih belum sempurna. Penulis memohon maaf jika dalam disertasi ini masih terdapat kekurangan dan mengharapkan saran dan kritik dari para pembaca serta tetap mengharapkan semoga tulisan ini bisa memberikan manfaat kepada seluruh pihak.

Makassar, 1 Oktober 2022

Muliadi

ABSTRAK

MULIADI, *Model Sistem Surveilan Otomatis Terintegrasi dan Dinamis Untuk Operasional Perangkat-Perangkat IoT Kelistrikan*

(dibimbing oleh: Andani Achmad, Intan Sari Areni dan Elyas Palantei)

Penelitian tentang Teknik *Fault detection and diagnostics* (FDD) dapat digunakan untuk memantau sistem kelistrikan bangunan serta mendeteksi dan mendiagnosa kerusakan peralatan kelistrikan. FDD adalah proses secara otomatis mengetahui dan mengisolasi kesalahan dalam bangunan untuk perlindungan sistem dari kerusakan. Beberapa aplikasi FDD dari bangunan dikembangkan dan diteliti berdasarkan hubungan antara suhu, tekanan, dan termodinamika untuk deteksi dan diagnosis kesalahan. Penelitian dalam disertasi ini menyajikan tentang Sistem Pemantauan dan Pengendalian Pengoperasian Perangkat Kelistrikan Berbasis *Internet of Things* (IoT) secara terpusat. Beberapa Perangkat yang dirancang dan dikembangkan dengan pemodelan sedemikian rupa agar memudahkan dalam pengoperasian, bersifat *reconfigurable* dan dinamis. Pemantauan, pengontrolan, pendeteksian kerusakan pada perangkat kelistrikan dapat diakses pada sebuah sistem yang terpusat dan terintegrasi. Pengujian prinsip kerja sistem sebanyak 10 kali pada setiap sistem dilakukan dan sistem dapat bekerja dengan baik sesuai dengan prinsip kerja. Pengujian respon *time* dilakukan sebanyak 25 kali pengiriman data *control* ke server dengan nilai rata-rata respon *time* secara keseluruhan sebesar 5,04 detik. Setelah melakukan pengujian Aplikasi android MACSys menggunakan *Firebase Test Lab* menggunakan *device virtual* dengan spesifikasi pixel 5e dan API level 30, orientasi *portrait* dan dinyatakan lulus/berhasil uji dan dapat berjalan dengan baik pada perangkat Samsung, vivo dan redmi.

Kata kunci: *internet of things, failure detection system, fire base test lab, automation system*

ABSTRACT

Muliadi, An Integrated and Dynamic Automated Surveillance System Model for Operation of IoT-based Electrical Devices

(supervised by: Andani Achmad, Intan Sari Areni and Elyas Palantei)

Research on Fault detection and diagnostics (FDD) techniques can be used to monitor building electrical systems as well as detect and diagnose damage on electrical equipment. FDD is the process of automatically detecting and isolating faults in a building for system protection from damage. Several FDD applications of buildings were developed and researched based on the relationship between temperature, pressure, and thermodynamics for fault detection and diagnosis. The research in this dissertation presents a centralized Internet of Things (IoT)-based Monitoring and Control System for Electrical Device Operations. Some Devices are designed and developed with modeling in such a way as to make them easier to operate, reconfigurable and dynamic. Monitoring, controlling, and detecting damage to electrical devices can be accessed in a centralized and integrated system. Testing the working principle of the system 10 times on each system is carried out and the system can work properly according to the principle. Response time testing was carried out 25 times by sending control data to the server with an overall average response time of 5,04 seconds. After testing the MACSys android application using the Firebase Test Lab using a virtual device with pixel 5e specifications and API level 30, portrait orientation and was declared to have passed/successfully tested and can run well on Samsung, Vivo and Redmi devices.

Keywords: internet of things, Failure Detection system, fire base test lab, automation system

DAFTAR ISI

Halaman

DISERTASI	i
HALAMAN PENGESAHAN	Error! Bookmark not defined.
PRAKATA	iii
ABSTRAK	v
ABSTRACT	vi
DAFTAR ISI	vii
DAFTAR TABEL	x
DAFTAR GAMBAR	xi
DAFTAR LAMPIRAN	xiv
BAB I	1
PENDAHULUAN	1
A. Latar Belakang Masalah	1
B. Rumusan Masalah	6
C. Tujuan Penelitian	6
D. Batasan Masalah	7
E. Manfaat Penelitian	7
F. Ruang Lingkup Penelitian	9
G. Penelitian Terkait	9
BAB II	24
TINJAUAN PUSTAKA	24
A. Surveilans	24
B. Mikrokontroler	24
C. Arduino	26
D. NodeMCU ESP-8266	29
E. Sensor	30

F.	Sensor PZEM 004T	31
G.	Sensor DHT	32
H.	Sensor DS18B20	34
I.	Sensor LDR	36
J.	Led	36
K.	Power Supply	38
L.	Kodular	39
M.	<i>Internet of Things</i>	40
N.	Komunikasi pada IoT	43
O.	<i>Fault Detection and Diagnosting (FDD)</i>	45
P.	Kerangka Pikir	52
BAB III		55
METODOLOGI PENELITIAN		55
A.	Rancangan Penelitian	55
B.	Jenis Penelitian	57
C.	Lokasi dan Waktu Penelitian	58
D.	Teknik Pengumpulan, Validasi, dan Analisis Data	58
E.	Parameter Ukur	59
F.	Alat Penelitian	59
G.	Prosedur Penelitian	60
H.	Arsitektur Sistem	61
I.	<i>Flowchart</i> Sistem	63
J.	Skema Detail, <i>Listing</i> Program, dan Penjelasan Program	64
BAB IV		82
HASIL PENELITIAN DAN PEMBAHASAN		82
A.	Hasil rancangan aritektur Model MACSys	82
B.	Hasil Rancangan Sistem MACSys	83
C.	Hasil Pengujian MACSys	86
D.	Hasil Pengujian <i>Blackbox</i> MACSys	94
E.	Hasil Pengujian Respon <i>Time</i> MACSys	94
BAB V		100

PENUTUP	100
A. Kesimpulan	100
B. Saran.....	102
DAFTAR PUSTAKA	103
LAMPIRAN.....	107

DAFTAR TABEL

Tabel 1.1.	State of the art penelitian	12
Tabel 1.2.	Kebaharuan penelitian	23

DAFTAR GAMBAR

Gambar 2.1	Arduino IDE.....	27
Gambar 2.2	Arduino Web Editor	27
Gambar 2.3	<i>Board</i> NodeMCU.....	29
Gambar 2.4	Rangkaian Signal Conditioning.....	29
Gambar 2.5	Sensor PZEM 004T dan <i>split core</i>	31
Gambar 2.6	Bentuk sensor DHT22.....	32
Gambar 2.7	Pin out sensor DS18B20	33
Gambar 2.8	Mode sensor SD18B20	34
Gambar 2.9	Simbol dan bentuk LDR.....	35
Gambar 2.10	Struktur LED	36
Gambar 2.11	Bentuk dan simbol LED.....	37
Gambar 2.12	<i>Power supply</i>	37
Gambar 2.13	Tampilan website kodular.....	38
Gambar 2.14	Grafik perbandingan perangkat komunikasi.....	44
Gambar 2.15	Aplikasi sederhana dari <i>Fault Detection and Diagnosis</i>	45
Gambar 2.16	Kategorisasi metode FDD untuk bangunan yang diadaptasi dari Kim dan Katipamula.....	47
Gambar 2.17	Klasifikasi teknik FDD untuk ventilasi.....	49
Gambar 2.18	Bagan kerangka pikir penelitian.....	53
Gambar 3.1	Rancangan penelitian.....	55
Gambar 3.2	<i>Flowchart</i> rancangan penelitian	58
Gambar 3.3	Arsitektur sistem.....	59
Gambar 3.4	<i>Flowchart</i> sistem.....	62
Gambar 3.5	Skema detail PSS.....	63

Gambar 3.6	Skema detail ACSS.....	66
Gambar 3.7	Skema detail LSS.....	70
Gambar 3.8	Skema detail SSS.....	74
Gambar 4.1	Desain Model MACSys.....	80
Gambar 4.1.a	Sistem monitoring MACSys.....	81
Gambar 4.1.b	Sistem pengontrolan MASys.....	82
Gambar 4.2	Panel Sub System (PSS).....	83
Gambar 4.3	AC Sub System (ACSS).....	83
Gambar 4.4	Lamp Sub System (LSS).....	83
Gambar 4.5	Socket Sub System (SSS).....	84
Gambar 4.6	Tampilan aplikasi MACSys.....	84
Gambar 4.7	Tampilan aplikasi android MACSys.....	85
Gambar 4.8	Pemasangan PSS pada kwh meter	86
Gambar 4.9	a) Grafik konsumsi energy menggunakan tiga alat ukur dengan nilai rata-rata 16,11 kwh dalam 7 hari, b) Grafik persentasi error rate dengan nilai rata-rata 2,31%	87
Gambar 4.10	Grafik konsumsi energy yang terbuang menggunakan sensor modul PZEM-004T.....	87
Gambar 4.11	Pemasangan ACSS pada Air Conditioner	88
Gambar 4.12	Grafik hasil monitoring ASS mode normal.....	89
Gambar 4.13	Grafik hasil monitoring ACSS mode <i>failure</i>	89
Gambar 4.14	Pemasangan modul LSS	90
Gambar 4.15	Hasil monitoring SSS dalam kondisi normal	91
Gambar 4.16	Pemasangan modul SSS	91
Gambar 4.17	Hasil monitoring SSS dalam kondisi normal, overload dan overheat.....	93
Gambar 4.18	Respon time MACSys dalam kondisi normal.....	94

Gambar 4.19	Respon time MACSys dalam kondisi deteksi kesalahan....	96
Gambar 4.20	Rerata Respon time MACSys dalam kondisi deteksi kesalahan	96
Gambar 4.21	Hasil pengujian sistem.....	97
Gambar 4.22	Hasil pengujian performance Samsung SM-A1015FN.....	98
Gambar 4.23	Hasil pengujian performance Vivo 1610 (AP123).....	98
Gambar 4.24	Hasil pengujian performance Redmi 6A (AP127).....	99

DAFTAR LAMPIRAN

Lampiran A1 Listing dan Penjelasan Program	105
Lampiran A2 Pembuatan Firebase	132
Lampiran A3 Pembuatan Aplikasi Android	139
Lampiran A4 Pembuatan Aplikasi website	156
Lampiran B1 Hasil pengujian Modul ACSS	160
Lampiran B2 Hasil pengujian Modul LSS	166
Lampiran B3 Hasil pengujian Modul SSS	167
Lampiran B4 Hasil pengujian Modul PSS	170
Lampiran B5 Hasil pengujian blackbox MACSys	173
Lampiran B6 Hasil pengujian blackbox aplikasi android MACSys	175
Lampiran B7 Rincian biaya modul-modul MACSys	177

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Pembangunan ekonomi berkelanjutan sangat dipengaruhi oleh konsep ketahanan energi (Harkouss et al., 2018). Badan Energi Internasional memperkirakan permintaan listrik global akan mengalami peningkatan secara signifikan pada tahun 2035 (Hannan et al., 2018).

Peningkatan efisiensi energi dapat mengurangi permintaan kelistrikan Negara, sehingga memberikan penghematan moneter, gas rumah kaca dan pengurangan polutan lainnya, dan peningkatan keamanan energi. Penggunaan energi secara efisien dapat menurunkan permintaan listrik dan mengurangi resiko kekurangan listrik yang berdampak pada pertumbuhan ekonomi. Permintaan energi puncak di Indonesia diperkirakan mencapai tiga kali lipat antara tahun 2020 dan 2030, untuk konsumsi pengkondisian udara (*Air Conditioner*) (McAllister et al., 2017).

PT. PLN (Persero) merupakan satu-satunya Badan Usaha Milik Negara yang bergerak di bidang penyediaan sampai penyaluran jasa tenaga listrik. Seiring berkembangnya teknologi, industri dan meningkatnya kebutuhan akan energi listrik, maka dibutuhkan pasokan energi listrik dan penyaluran yang andal. Keandalan akan pasokan energi listrik adalah merupakan kepuasan pelanggan. PT. PLN (Persero) menjaga keandalan sistem penyaluran tenaga

listrik menggunakan sistem pengoperasian yang mempunyai tingkat keandalan tinggi, dikarenakan PT. PLN (Persero) mempunyai visi diakui sebagai perusahaan kelas dunia yang dinilai dari SAIDI (*System Average Interruption Duration Index*) dan SAIFI (*System Average Interruption Frequency Index*). Untuk itu diperlukan sistem operasi yang mempunyai keandalan baik, sehingga dapat meminimalisir pemadaman untuk menjaga kepuasan pelanggan (Soleh, 2014).

Bangunan modern terdiri dari berbagai peralatan seperti pemanas, ventilasi dan pendingin udara (HAVAC) dan penerangan. Setiap peralatan terdiri dari beberapa komponen seperti pompa, kipas, saluran, sensor, lampu, kabel dan lain-lain. Semua peralatan dapat dipantau dan dikelola oleh system manajemen gedung (BMS). Semua komponen ini dapat mengalami kerusakan yang dapat disebabkan karena aus karena pemakaian dalam waktu yang lama, kerusakan konfigurasi, dan masalah komunikasi. Kerusakan peralatan memengaruhi biaya perawatan dan khususnya efisiensi energi (Mills, 2011) (Kim et al., 2018).

Beban peralatan seperti *Heating, Ventilation, and Air-Conditioning* (HVAC) bervariasi tergantung pada jenis dan lokasi bangunan, tetapi kerusakan pada salah satu peralatan dapat membuat hingga 50% dari total konsumsi energi sehingga dapat menyebabkan kehilangan energi yang besar (Kim et al., 2018)(Pe, 2008). Penelitian menunjukkan bahwa penghematan energi antara 20% hingga 30% dapat dicapai dengan cara *re-commissioning malfungsi* sistem HVAC (Agency, 2005). Sistem HVAC sering disesuaikan

untuk bangunan spesifik sehingga kurangnya integrasi sistem yang berkualitas (Yu, Woradechjumroen and Yu, 2014).

Sektor gedung saat ini merupakan konsumen energi terbesar di dunia (Jain et al., 2016). Sehingga, manajemen energi dalam gedung telah menjadi tujuan internasional teknologi modern (Li et al., 2013). Penelitian menunjukkan bahwa sekitar 40% dari total energi di dunia dikonsumsi oleh gedung, yang merupakan sepertiga dari emisi *Green House Gas* (GHG) (Huang et al., 2017) (Mytafides et al., 2017). Penelitian lain menunjukkan bahwa 49% dari total energi dikonsumsi oleh gedung pada tahun 2014 (Wu et al., 2020) dan 60% dari konsumsi energi gedung karena tujuan pemanasan dan pendinginan (Lizana et al., 2017). Karena itu, konsumsi energi dan pengaruhnya terhadap perubahan iklim paling banyak menjadi masalah yang perlu diselesaikan di sektor pemanfaatan energi pada gedung (Santamouris, 2016) (Sun et al., 2018) (Cao et al., 2016). *Building Energy Manajemen System* (BEMS) dengan menggunakan *Internet of Energy* (IoE) dapat memaksimalkan efisiensi energi dengan meminimalkan kerugian dan dampak lingkungan (Hannan et al., 2018).

Penggunaan peralatan kelistrikan pada gedung umumnya belum memiliki sistem manajemen energi yang pemakaiannya dapat dipantau, begitupun pengendalian peralatan listrik juga masih bersifat konvensional. Kekurangan sistem yang ada saat ini adalah terkadang peralatan-peralatan listrik lupa dinonaktifkan padahal sudah tidak digunakan lagi. Sensor *array* dengan keuntungan secara simultan mendeteksi beberapa parameter yang berbeda digunakan secara luas dalam banyak aplikasi teknik seperti teknik

lingkungan, teknik kimia, teknik biologi, dan teknik *aerospace* (Chen *et al.*, 2016) sebagai sumber perolehan informasi, keakuratan dan keandalan *output array* sensor sangat penting untuk kinerja sistem pengukuran dan kontrol.

Teknik *Fault detection and diagnostics* (FDD) dapat digunakan untuk memantau sistem kelistrikan bangunan serta mendeteksi dan mendiagnosa kerusakan peralatan kelistrikan. (Hannan *et al.*, 2018) FDD adalah proses secara otomatis mengetahui dan mengisolasi kesalahan dalam bangunan untuk perlindungan sistem dari kerusakan. Beberapa aplikasi FDD dari bangunan dikembangkan dan diteliti berdasarkan hubungan antara suhu, tekanan, dan termodinamika untuk deteksi dan diagnosis kesalahan.

Sistem otomasi dengan menerapkan integrasi antara modul dengan lingkungan yang berbeda merupakan sarana yang efisien untuk menyediakan komunikasi otomatis diantara perangkat (Guilherme Mussi, Leonardo Barreto Campos, Carlos Eduardo Cugnasca, 2017).

Hasil verifikasi model menunjukkan bahwa model *manufacture* pintas dari industri konstruksi dapat mewujudkan integrasi masyarakat manusia dan sistem fisik dan mencapai tujuan manajemen waktu nyata dan pengendalian personal, mesin, peralatan dan infrastruktur dalam seluruh jaringan (Lingji Kong dan Biao Ma, 2019).

Tren dalam elektronik terintegrasi adalah rasio kinerja terhadap biaya yang lebih baik, radio berdaya rendah dan sensor mikroeletromekanis dengan pemroses sinyal, komunikasi nirkabel, sumber daya dan sinkronisasi, semua dikemas dalam perangkat miniatur murah (J. Agree, L. Clare, 2000).

Pada suatu sistem tenaga listrik dibutuhkan suatu sistem pengamanan yang handal yang mampu mendeteksi gangguan dan kesalahan yang terjadi pada system tersebut. Hal tersebut diperlukan agar suplai energi listrik dapat dipertahankan dengan baik. Agar sistem yang dimiliki mampu menjaga pelayanan suplai tenaga listrik dengan baik, maka sistem pengamanan harus memenuhi syarat *reliable*, selektif, sensitif, memiliki waktu operasi yang cepat, ekonomis dan sederhana.

Sistem yang dibangun dapat memonitoring energi yang digunakan oleh peralatan kelistrikan seperti AC, peralatan pencahayaan, dan peralatan kelistrikan yang terkoneksi dengan stok kontak yang ada seperti komputer PC atau adaptor laptop. Pengontrolan juga dengan mudah dapat dilakukan karena sistem terkoneksi dengan jaringan internet atau terintegrasi dengan *Internet of Things* (IoT). Pengontrolan dilakukan menggunakan *smartphone* dengan aplikasi *android*. Sistem ini juga dilengkapi dengan peringatan dini terhadap gangguan yang terjadi sehingga pengguna dengan cepat dapat mengetahui jika ada gangguan yang terjadi pada sistem kelistrikan pada suatu gedung.

Penelitian dalam disertasi ini menyajikan tentang Sistem Pemantauan dan Pengendalian Pengoperasian Perangkat Kelistrikan Berbasis *Internet of Things* (IoT).

B. Rumusan Masalah

Berdasarkan identifikasi masalah yang telah dilakukan sehingga dirumuskan permasalahan dalam penelitian ini sebagai berikut:

1. Bagaimana rancangan sistem pemantauan perangkat kelistrikan pada instalasi pemanfaatan gedung?
2. Bagaimana rancangan sistem pengendalian pemakaian peralatan dan pembebanan kelistrikan gedung berbasis *Internet of things* (IoT)?
3. Bagaimana rancangan sistem pemantauan dan pengendalian kerusakan peralatan kelistrikan?
4. Bagaimana rancangan sistem pemantauan, pendeteksi dan pengendalian kerusakan secara otomatis?

C. Tujuan Penelitian

Sesuai dengan latar belakang permasalahan dan rumusan masalah penelitian maka tujuan penelitian ini adalah:

1. Menghasilkan sistem pemantauan perangkat kelistrikan pada instalasi pemanfaatan gedung.
2. Menghasilkan sistem pengendalian pemakaian peralatan dan pembebanan kelistrikan gedung berbasis *Internet of things* (IoT).
3. Menghasilkan sistem pemantauan dan pengendalian kerusakan peralatan kelistrikan.
4. Menghasilkan sistem pemantauan dan pengendalian kerusakan perangkat kelistrikan secara otomatis.

D. Batasan Masalah

1. Pada penelitian ini sistem pemantauan dilakukan pada jumlah konsumsi dan *lost* energi pada instalasi rumah tinggal atau institusi.
2. Pada penelitian ini sistem pengontrolan dilakukan pada pengoperasian perangkat-perangkat kelistrikan seperti AC, pencahayaan, dan stok kontak.
3. Pada penelitian ini sistem pendeteksian kerusakan dilakukan pada perangkat-perangkat kelistrikan seperti pada AC untuk deteksi suhu, lampu untuk deteksi cahaya, stok kontak untuk deteksi kelebihan panas kabel penghantar dan kelebihan beban yang diperbolehkan.
4. Pada penelitian ini sistem otomatis dilakukan pada perangkat-perangkat kelistrikan seperti AC, lampu dan stok kontak.

E. Manfaat Penelitian

Berdasarkan latar belakang, rumusan masalah dan tujuan penelitian tersebut, diharapkan dari hasil penelitian ini dapat memberikan manfaat bagi sistem kelistrikan gedung baik skala rumah tangga maupun skala institusi dan industri. Hasil penelitian ini diharapkan dapat bermanfaat untuk hal-hal sebagai berikut:

1. Teoritis

Diharapkan memberikan kontribusi untuk pengembangan ilmu pengetahuan pada ruang lingkup sistem pemantauan dan monitoring perangkat-perangkat kelistrikan IoT. Pemantauan terhadap

pengoperasian perangkat-perangkat kelistrikan IoT menjadi lebih baik, dalam keadaan normal, gangguan dan *emergency* serta dapat dilakukan secara cepat dan *real-time*.

2. Praktis

- a. Menjadi sumber rujukan bagi peneliti yang tertarik untuk meneliti tentang sistem pemantauan, pengontrolan dan pendeteksian kerusakan pada pengoperasian perangkat-perangkat kelistrikan IoT, baik yang dilakukan secara manual maupun secara otomatis.
- b. Sistem *fault detection* dan diagnosis yang diterapkan pada perangkat-perangkat kelistrikan, tercapainya pemulihan gangguan yang lebih baik dengan pengawasan dan penanganan gangguan yang dilakukan dengan cepat dan tanpa ke lapangan menjadi keunggulan sistem yang dibangun.

3. Implementasi

- a. Memperkenalkan model MACSys yang terdiri sistem monitoring dan pengontrolan perangkat-perangkat IoT kelistrikan yang dilengkapi dengan pendeteksian dan diagnosa kesalahan secara otomatis.
- b. Mewujudkan suatu sistem monitoring perangkat yang lebih mutakhir, terintegrasi, dinamis, *reconfigurable*, dengan pemanfaatan sensor *array* yang dilengkapi notifikasi.

F. Ruang Lingkup Penelitian

Berdasarkan batasan masalah pada penelitian ini maka ruang lingkup penelitian ini meliputi sistem pemantauan, pengontrolan, pendeteksian kerusakan secara otomatis disertai dengan pemberitahuan. Sistem yang dirancang diimplementasikan pada instalasi listrik rumah tinggal dan perangkat-perangkat IoT kelistrikan yaitu panel listrik, AC, lampu dan stok kontak.

G. Penelitian Terkait

Penelitian ini diharapkan dapat memberikan solusi dan berkontribusi dalam pemanfaatan energi dan pengelolaannya. *Building Energy Management System (BEMS)* dengan menggunakan *Internet of Energy (IoE)* sehingga dapat memaksimalkan efisiensi energi dengan meminimalkan kerugian dan dampak lingkungan.

Kontribusi yang sama telah dilaksanakan oleh para peneliti lainnya yaitu:

1. Peneliti (I. A. Zuolkernan et al., 2017), melakukan penelitian mengenai manajemen energi cerdas. Hasil penelitian menunjukkan bahwa pengguna dapat memantau dan mengontrol perangkat dari jarak jauh dan melakukan tagihan secara *online* menggunakan perangkat komputer atau perangkat seluler.
2. Peneliti (D. Despa et al., 2018) menghasilkan prototipe KWh meter digital yang digunakan untuk mengetahui besarnya konsumsi energi di laboratorium berbasis teknologi *Internet of Things (IoT)*.
3. Peneliti (K. Chooruang et al., 2018) mengembangkan sistem pemantauan tegangan, arus, daya aktif, dan konsumsi daya secara keseluruhan.
4. Peneliti (A. D. Pangestu et al., 2019) membangun sistem monitoring yang berfokus pada pembacaan arus dan daya terhadap beban induktif dan

beban resistif dengan Arduino Nodemcu Esp8266. Tingkat akurasi yang diperoleh berada pada kisaran 96% hingga 99%.

5. Peneliti (S. Joshi and V. Kiran, 2020) penelitian tentang sistem pemantauan konsumsi daya listrik dilakukan berfokus pada pemantauan parameter daya seperti tegangan, arus, frekuensi, faktor daya, daya aktif, daya reaktif, dan daya semu dan dapat juga memprediksi kekuatan alat dan mengidentifikasi jenis perangkat yang terhubung.
6. Peneliti (H. T. S. ALRikabi et al., 2020) membangun sistem kontrol perangkat elektronik menggunakan protocol komunikasi GSM. Perintah pengontrolan dikirimkan melalui layanan SMS untuk mengaktifkan dan menonaktifkan perangkat elektronik seperti kulkas, *air conditioner*, *ceiling fan*, *lighting*, dan *water cooler*.
7. Peneliti (D. Indra, 2019) membangun sistem pengontrolan perangkat listrik menggunakan Raspberry Pi 2 melalui komunikasi Wi-Fi. Didapatkan hasil bahwa jarak maksimal untuk mengontrol sistem menggunakan laptop adalah 30 meter dan menggunakan *smartphone* adalah 20 meter.
8. Peneliti (E. Al-Hassan et al., 2018) membangun *smart power socket* yang dikontrol untuk menghubungkan dan memutus arus listrik pada *stock contact* melalui komunikasi *Zigbee*.
9. Peneliti (S. H. M. S. Andrade, et al., 2021) membangun sistem monitoring konsumsi energi listrik melalui aplikasi *mobile* secara *realtime* menggunakan Arduino Uno sebagai *microcontroller* dan LoRa EXP32 SX1278 sebagai komunikasi internal antara *node* yang lain. *Central Unit* mengirimkan data ke Internet menggunakan ESP8266.
10. Peneliti (H. W. Rotib, et al., 2021) membangun teknologi *smart home* pada kendali lampu jarak jauh melalui *website application* berbasis *internet of things*.
11. Peneliti (H. Isyanto, et al., 2020) mengembangkan teknologi *smart home* pada kendali lampu, tv, dan kipas angin berbasis perintah suara menggunakan *google asistant* yang ditujukan untuk memudahkan

penyandang disabilitas dalam mengendalikan perangkat elektronik rumah tangganya.

12. Peneliti (C. Hermanu, et al., 2022) membangun sistem pengendalian lampu dengan dua mode, yaitu manual dan otomatis. Pada manual lampu dikontrol menggunakan saklar fisik, dan saklar pada aplikasi tidak dapat mengontrol lampu sedangkan pada otomatis lampu dikontrol melalui aplikasi dan saklar tidak berfungsi. Pengembangan *smarthome* dibangun menggunakan ESP8266.
13. Peneliti (Hannan et al., 2018) penelitian tentang Teknik *fault detection and diagnostics* (FDD) dapat digunakan untuk memantau sistem kelistrikan bangunan serta mendeteksi dan mendiagnosa kerusakan peralatan kelistrikan.

Tabel 1.1 *State of the Art* Penelitian

No.	Peneliti dan Tahun	Judul Penelitian	Pokok Permasalahan	Metode (solusi Masalah)	Korelasi Usulan Penelitian
1	I. A. Zualkernan, M. Rashid, R. Gupta, and M. Alikarar, 2017	<i>A Smart Home Energy Management System Using IoT and Big Data Analytics Approach</i>	Hasil penelitian menunjukkan bahwa pengguna dapat memantau dan mengontrol perangkat dari jarak jauh dan melakukan tagihan secara <i>online</i> menggunakan perangkat seluler.		Menggunakan IoT untuk Sistem Manajemen Energi Rumah Pintar.
2	D. Despa, 2018	<i>Monitoring and Management of Electrical Energy in Laboratory Building Based on Internet of Things (IoT)</i>	Menghasilkan prototipe KWh meter digital yang digunakan untuk mengetahui besarnya konsumsi energi di laboratorium berbasis teknologi <i>Internet of Things</i> (IoT).		Menerapkan manajemen energi listrik pada gedung laboratorium.

3	(K. Chooruang, 2018)	<i>Design of an IoT Energy Monitoring System</i>	Mengembangkan sistem pemantauan tegangan, arus, daya aktif, dan konsumsi daya secara keseluruhan.	Memantau arus, tegangan dan konsumsi energi.
4	A. Ramschie, J. Makal, V. Ponggawa, and K. Kunci , 2019	<i>IoT Based Electric Power Consumption Monitoring System for Air Conditioner Equipment</i>	Merancang sistem monitoring untuk monitoring konsumsi listrik untuk peralatan <i>Air Conditioning</i> (AC) dimana sistem tersebut dapat memonitor dan menginformasikan besarnya konsumsi energi listrik dan harga yang dibayarkan untuk pengoperasian peralatan AC melalui <i>smartphone</i> atau melalui web pelayan.	merancang sistem monitoring untuk monitoring konsumsi listrik untuk peralatan <i>Air Conditioning</i> (AC).
5	A. D. Pangestu et al., 2019	<i>Electrical Load Monitoring System Based On Arduino NODEMCU ESP8266</i>	Membangun sistem monitoring yang berfokus pada pembacaan arus dan daya terhadap beban induktif dan beban resistif dengan Arduino Nodemcu Esp8266. Tingkat akurasi yang diperoleh berada pada kisaran 96% hingga 99%.	Membangun sistem monitoring arus, tegangan dan konsumsi energi.

6	S. Joshi and V. Kiran, 2020	<i>Design and Development of Power Monitoring System using IoT Technology</i>	penelitian tentang sistem pemantauan konsumsi daya listrik dilakukan oleh Sanjeev Joshi dan Dr. Kiran V berfokus pada pemantauan parameter daya seperti tegangan, arus, frekuensi, faktor daya, daya aktif, daya reaktif, dan daya semu dan dapat juga memprediksi kekuatan alat dan mengidentifikasi jenis perangkat yang terhubung.		Tentang sistem pemantauan konsumsi daya listrik.
7	H. T. S. ALRikabi, A. H. Alaidi, and K. Nasser, 2020	<i>The Application of Wireless Communication in IOT for Saving Electrical Energy</i>	Membangun sistem kontrol perangkat elektronik menggunakan protocol komunikasi GSM. Perintah pengontrolan dikirimkan melalui layanan SMS untuk mengaktifkan dan menonaktifkan perangkat elektronik seperti kulkas, <i>air conditioner</i>, <i>celling fan</i>, <i>lighting</i>, dan <i>water cooler</i>.	Menggunakan protocol komunikasi GSM.	Membangun sistem kontrol perangkat elektronik.

8	D. Indra, 2019	<i>Design Web-Based Electrical Control System Using Raspberry Pi</i>	Membangun sistem pengontrolan perangkat listrik menggunakan Raspberry Pi 2 melalui komunikasi Wi-Fi. Didapatkan hasil bahwa jarak maksimal untuk mengontrol sistem menggunakan laptop adalah 30 meter dan menggunakan <i>smartphone</i> adalah 20 meter.	Menggunakan Raspberry Pi 2 melalui komunikasi Wi-Fi. Menggunakan laptop adalah 30 meter dan menggunakan <i>smartphone</i> adalah 20 meter.	Membangun Sistem pengontrolan perangkat listrik.
9	E. Al-Hassan, H. Shareef, M. M. Islam, A. Wahyudie, and ...,2018	<i>Improved smart power socket for monitoring and controlling electrical home appliances</i>	Membangun <i>smart power socket</i> yang dikontrol untuk menghubungkan dan memutus arus listrik pada stock <i>contact</i> melalui komunikasi <i>Zigbee</i> .		Membangun <i>smart power socket</i> yang dikontrol.
10	T. H. Nasution, M. A. Muchtar, I. Siregar, and ..., 2017	<i>Electrical appliances control prototype by using GSM</i>	Membangun sistem pengontrolan perangkat elektronik melalui relay menggunakan SIM 900 sebagai komunikasi GSM via SMS dan Arduino sebagai <i>microcontroller</i> . Sistem yang	Melalui <i>relay</i> menggunakan SIM 900 sebagai komunikasi GSM via SMS dan Arduino	Membangun sistem pengontrolan perangkat elektronik.

		<i>module and Arduino</i>	dibangun dapat mengaktifkan dan menonaktifkan <i>relay</i> secara paralel.	sebagai <i>microcontroller</i> .	
11	S. H. M. S. Andrade, G. O. Contente, L. B. Rodrigues, L. X. Lima, N. L. Vijaykumar, and C. R. L. Frances, 2021	<i>A Smart Home Architecture for Smart Energy Consumption in a Residence With Multiple Users</i>	Membangun sistem monitoring konsumsi energi listrik melalui aplikasi <i>mobile</i> secara <i>realtime</i> menggunakan Arduino Uno sebagai <i>microcontroller</i> dan LoRa EXP32 SX1278 sebagai komunikasi internal antara <i>node</i> yang lain. <i>Central Unit</i> mengirimkan data ke Internet menggunakan ESP8266.	Menggunakan Arduino Uno sebagai <i>microcontroller</i> dan LoRa EXP32 SX1278 sebagai komunikasi internal antara <i>node</i> yang lain.	Membangun sistem monitoring konsumsi energi listrik.
12	E. Al-Hassan, H. Shareef, Md. M. Islam, A. Wahyudie, and A. A. Abdrabou, 2018	<i>Improved Smart Power Socket for Monitoring and Controlling Electrical Home Appliances</i>	Membangun <i>Smart Power Socket</i> untuk memonitoring konsumsi energi perangkat listrik rumah tangga menggunakan Arduino Nano dengan Xbee sebagai <i>module</i> komunikasi Zigbee.	Menggunakan Arduino Nano dengan Xbee sebagai <i>module</i> komunikasi Zigbee.	Membangun <i>Smart Power Socket</i> untuk memonitoring konsumsi energi perangkat listrik rumah tangga.

13	S. Gan, K. Li, Y. Wang, and C. Cameron,, 2018	<i>“IoT Based Energy Consumption Monitoring Platform for Industrial Processes</i>	Membangun sistem monitoring konsumsi energi berbasis IoT menggunakan ABB B24 <i>digital power</i> meter sebagai pengukur konsumsi energi dan Rapsbery Pi 2 sebagai <i>controller</i> serta LoRa sebagai komunikasi utama pada sistem untuk lingkungan industri.	Membangun sistem monitoring konsumsi energi berbasis IoT.
14	L. Susanti, D. Fatrias, D. Ichwana, H. Kamil, and M. V. Putri, 2018	<i>A Configuration System for Real-Time Monitoring and Controlling Electricity Consumption Behavior</i>	Mengembangkan sistem monitoring konsumsi energi listrik berbasis IoT menggunakan sensor arus SCT013 dan Wemos D1 sebagai <i>central unit</i> yang meneruskan data ke <i>website application</i>	Mengembangkan sistem monitoring konsumsi energi listrik berbasis IoT.
15	S. Wasoontarajaroen, K. Pawasan, and	<i>Development of an IoT device for monitoring</i>	Membangun perangkat IoT untuk memonitoring konsumsi energi listrik menggunakan sensor PZEM-004t yang mengukur tegangan dan arus, Arduino	Membangun perangkat IoT untuk memonitoring konsumsi energi listrik.

	V. Chamnanphrai, 2017	<i>electrical energy consumption</i>	Uno sebagai <i>controller</i> dan ESP8266 sebagai perangkat pengirim data ke internet dan diterima melalui <i>mobile application</i> .	
16	H. W. Rotib, M. B. Nappu, Z. Tahir, A. Arief, and M. Y. A. Shiddiq, 2021	<i>Electric Load Forecasting for Internet of Things Smart Home Using Hybrid PCA and ARIMA Algorithm</i>	Membangun teknologi <i>smart home</i> pada kendali lampu jarak jauh melalui <i>website application</i> berbasis <i>internet of things</i> .	Membangun teknologi <i>smart home</i> pada kendali lampu jarak jauh.
17	H. Isyanto, A. S. Arifin, and M. Suryanegara, 2020	Design and Implementation of IoT-Based Smart Home Voice	mengembangkan teknologi smart home pada kendali lampu, tv, dan kipas angin berbasis perintah suara menggunakan Google Asistant yang ditujukan untuk memudahkan penyandang disabilitas	mengembangkan teknologi smart home pada kendali lampu, tv, dan kipas angin berbasis perintah suara

		Commands for disabled people using Google Assistan	dalam mengendalikan perangkat elektronik rumah tangganya	
18	M. S. Soliman, A. A. Alahmadi, A. A. Maash, and M. O. Elhabib, 2017	Design and Implementation of a Real-Time Smart Home Automation System Based on Arduino Microcontroller Kit and LabVIEW Platform	mengembangkan teknologi smart home pada kendali lampu, kipas dan CCTV secara otomatis berdasarkan nilai dari pembacaan sensor menggunakan microcontroller Arduino dan LabView sebagai platfrom aplikasi untuk menampilkan data monitoring sensor melalui PC.	mengembangkan teknologi smart home pada kendali lampu, kipas dan CCTV secara otomatis berdasarkan nilai dari pembacaan sensor
19	C. Hermanu, H. Magfiroh, H. P. Santoso, Z. Arifin,	Dual Mode System of Smart Home	membangun sistem pengendalian lampu dengan dua mode, yaitu manual dan otomatis. Pada manual lampu di control	membangun sistem pengendalian lampu dengan

	and C. Harsito, 2022	Based on Internet of Things	menggunakan saklar fisik, dan saklar pada aplikasi tidak dapat mengontrol lampu sedangkan pada otomatis lampu di control melalui aplikasi dan saklar tidak berfungsi. Penegmbangan smarthome dibangun menggunakan ESP8266	dua mode, yaitu manual dan otomatis
20	T. S. Gunawan <i>et al.</i> , 2017	Prototype Design of Smart Home System using Internet of Things	membangun prototipe smart home system dengan pemanfaatan teknologi Internet of Things. Sistem yang dibangun dapat memonitoring kadar gas dan nilai temperature ruangan serta status buka dan tutup pintu secara realtime. Sistem juga dapat mengontrol beberapa peralatan elektronik rumah tangga seperti lampu, penguncian pintu dan kipas melalui website application.	membangun prototipe smart home system dengan pemanfaatan teknologi Internet of Things.

21	(Chen et al., 2016)		pada penelitian tentang sensor array dengan keuntungan secara simultan mendeteksi beberapa parameter yang berbeda digunakan secara luas dalam banyak aplikasi teknik seperti teknik lingkungan, teknik kimia, teknik biologi, dan teknik <i>aerospace</i> .		
22	(Hannan et al., 2018).		penelitian tentang Teknik Fault detection and diagnostics (FDD) dapat digunakan untuk memantau sistem kelistrikan bangunan serta mendeteksi dan mendiagnosa kerusakan peralatan kelistrikan.		
23	(Andani Achmad, et al., 2022)	Smart Electrical Devices Control with Intrusion Detection Alert	Penelitian tentang pengontrolan perangkat kelistrikan dengan pendeteksi kerusakan	Menggunakan ESP32 dan Wemos Mini. Web server firebase	Merancang sistem pengontrolan dan pendeteksi kesalahan

24	(Muliadi et al., 2021)	Smart home Energy Consumption Monitoring System Integrated with Internet Connection	Penelitian ini menghasilkan sistem monitoring konsumsi arus peralatan rumah tangga serta melihat akurasi dari sensor	Menggunakan nodeMCU ESP8266 dengan sensor arus PZEM-004T	Merancang sistem monitoring konsumsi energi
25	(Muliadi et al., 2022)	An IoT Based Power Consumption and Losses Monitoring Technique for A mini scale electrical network	Penelitian ini menghasilkan sistem pemantauan energi yang dapat memantau jumlah konsumsi dan <i>lost</i> energi secara <i>realtime</i>	Menggunakan wemos D1 mini dan sensor arus PZEM-004T	Merancang sistem monitoring konsumsi energi

Tabel 1.2 Kebaharuan penelitian yang diajukan

Penelitian yang diajukan	Kebaharuan yang ditawarkan
Model Sistem Surveilans Otomatis Terintegrasi dan Dinamis untuk Operasional Perangkat-Perangkat IoT Kelistrikan	1. Menghasilkan sistem pemantauan (<i>monitoring</i>) dan pengendalian (<i>Controlling</i>) (MACSys) yang dilengkapi dengan pendeteksi dan pendiagnosa parameter-parameter dari sistem kelistrikan yang dapat menyebabkan gangguan fungsi dari suatu peralatan kelistrikan. 2. Menghasilkan aplikasi berbasis android MACSys untuk memantau (<i>monitor</i>) dan mengendalikan (<i>control</i>) sistem MACSys yang telah lulus uji.

BAB II

TINJAUAN PUSTAKA

A. Surveilans

Surveilans merupakan proses pemantauan perilaku, dengan banyak kegiatan, atau informasi dengan tujuan mengumpulkan, menganalisis, mengelola, atau mengarahkan informasi. Hal ini dapat mencakup pengamatan dari jarak jauh melalui peralatan elektronik, seperti televisi sirkuit tertutup (CCTV), atau transmisi secara elektronik seperti lalu lintas Internet yang mencakup metode teknis sederhana, seperti pengumpulan intelijen manusia.

Sistem dalam surveilans dapat berupa sistem pemantauan rumah paling sederhana dan alarm pencuri, hingga kamera definisi tinggi, pendeteksi gerakan, dan solusi keamanan pemindaian retina. Berbagai bentuk konektivitas dengan kabel maupun tanpa kabel untuk kebutuhan komunikasi M2M dari berbagai sumber. Data dari berbagai sumber tersebut dikumpulkan dan dianalisis untuk mengetahui atau memprediksi perilaku atau mencegah kerusakan atau bahkan kejahatan. Data atau informasi tersebut kemudian dapat diakses pada sebuah platform atau perangkat umum.

B. Mikrokontroler

Mikrokontroler merupakan istilah dari *microcontroller* yang bermakna pengendali mikro. Hal tersebut tergambar pada mikrokontroler secara fisik yang merupakan komponen elektronika yang saling terintegrasi satu sama lain untuk menjalankan fungsi pengontrolan dan pengendalian pada sebuah

pekerjaan atau tugas tertentu secara terprogram. Mikrokontroler adalah *single chip* komputer yang dilengkapi dengan mikroprosesor sebagai otak pemrosesan, input dan output (I/O) sebagai pendukung, memori bahkan sampai *Analog to Digital Converter* (ADC). Dengan kelengkapan yang dimiliki oleh mikrokontroler, sering disebut sebagai komputer mikro atau microcomputer karena bekerja layaknya komputer namun dengan ukuran fisik yang lebih kecil. Penggunaan mikrokontroler semakin meningkat dan banyak digunakan dalam berbagai perangkat elektronik rumah tangga, peralatan industri, peralatan medis dan kedokteran, perangkat pendukung otomotif, bahkan sampai pada pengendali robot dan persenjataan militer. Hal ini tentunya tidak lepas dari beberapa keunggulan pada penerapan alat-alat berbasis mikrokontroler, yaitu:

1. Kemudahan dalam integrasi dengan perangkat lain
2. Ukuran dimensi yang relatif lebih kecil
3. Biaya implementasi yang lebih rendah
4. Produksi yang relatif lebih singkat sehingga dapat lebih cepat disebar ke pasar untuk dipergunakan (Ibadillah and Alfita, 2017)

Mikrokontroler digunakan dalam produk dan alat yang dikendalikan secara otomatis, seperti sistem kontrol mesin, *remote control*, mesin kantor, peralatan rumah tangga, alat berat, dan mainan. Dengan mengurangi ukuran, biaya, dan konsumsi tenaga dibandingkan desain menggunakan mikroprosesor memori dan alat *input output* yang terpisah, kehadiran mikrokontroler membuat kontrol elektrik untuk berbagai proses menjadi lebih ekonomis. Dengan penggunaan mikrokontroler ini maka sistem elektronik

akan menjadi lebih ringkas, rancang bangun sistem elektronik dapat dilakukan lebih cepat karena sebagian besar sistem merupakan perangkat lunak yang mudah dimodifikasi, dan juga gangguan yang terjadi lebih mudah ditelusuri karena sistemnya yang kompak.

C. Arduino

Arduino merupakan perangkat *opensource* yang dapat digunakan untuk merancang dan membuat perangkat elektronik semisal perangkat IoT dan disertai dengan perangkat lunak pendukung dalam pengembangan sistem di berbagai bidang. Arduino merupakan salah satu mikrokontroler yang populer dan banyak digunakan karena kemudahan dalam pemakaian dan bahasa pemrograman yang familiar, yaitu menggunakan bahasa pemrograman C/C++.

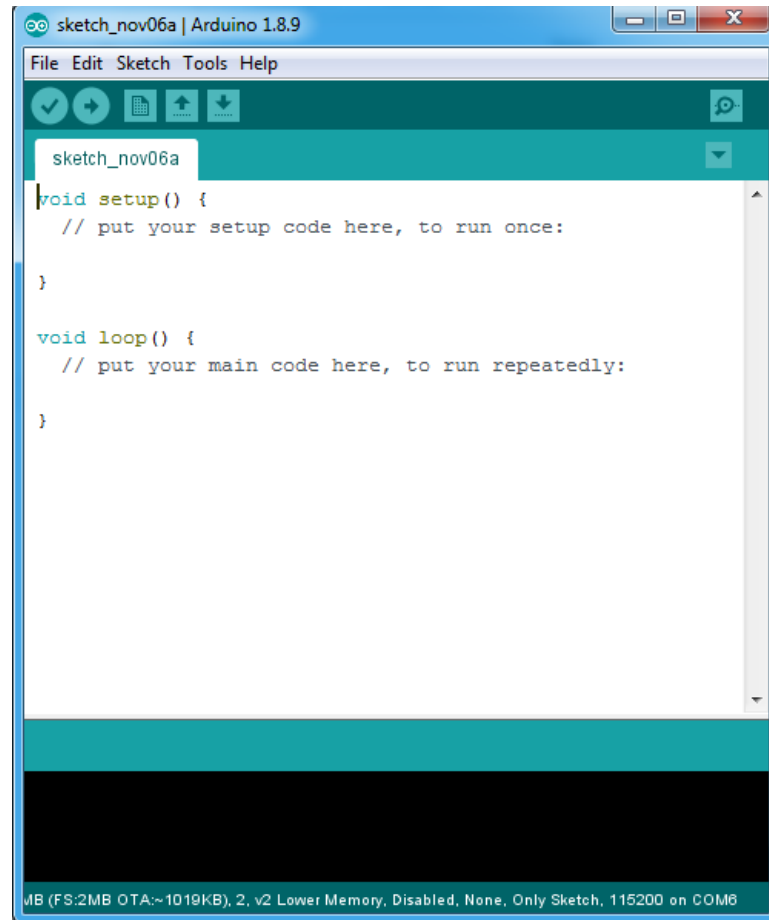
Terdapat banyak varian dari board arduino yang dapat menjadi pilihan, untuk memenuhi standar kebutuhan minimum pada sebuah sistem yang akan dikembangkan. Arduino memiliki banyak varian yang tersedia pada situs resmi [arduino.cc](https://www.arduino.cc) seperti arduino uno, arduino mega, arduino nano, arduino leonardo, 40 Dasar-Dasar Teknologi Internet of Things (IoT) arduino due, arduino micro, arduino zero, dll.

Perangkat lunak pengembang untuk arduino adalah *Arduino Integrated Development Environment* atau Arduino Software IDE. Perangkat lunak ini tersedia di berbagai platform seperti Windows, Linux dan macOS. Arduino IDE mendukung bahasa C dan C++ dengan aturan dan penggunaan kode yang khusus. Arduino IDE dapat didownload pada *website* resmi yaitu

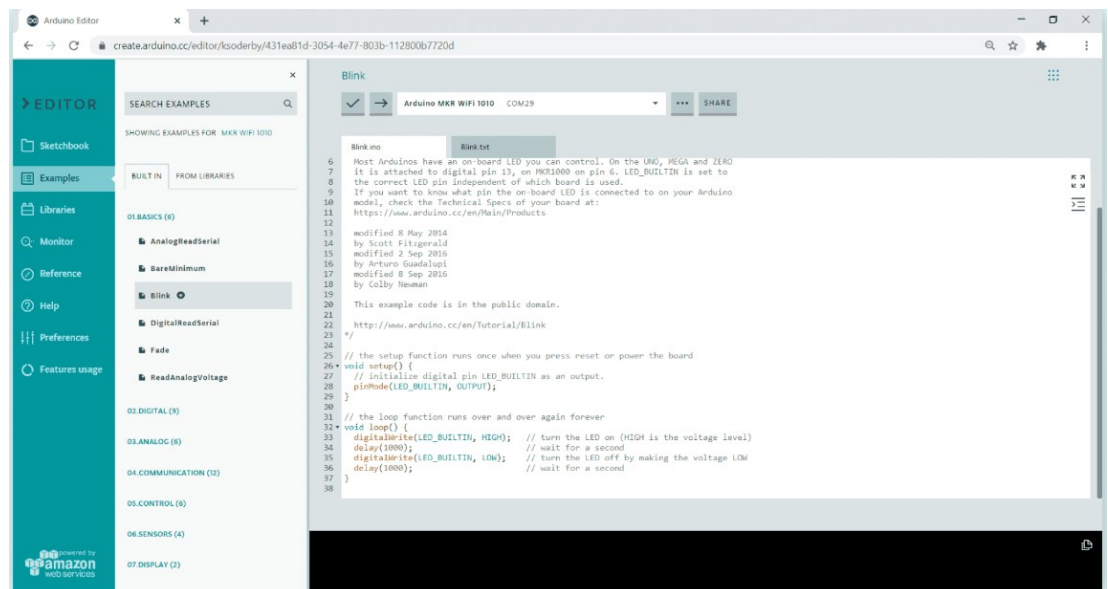
arduino.cc/en/software. Selain itu tersedia pula Arduino Web Editor yang dapat diakses melalui link create.arduino.cc/editor. (Arduino, 2022)

Berikut ini merupakan beberapa spesifikasi dasar yang dimiliki Arduino IDE antara lain:

1. Merupakan IDE (*Integrated Development Environment*) Yang memiliki dukungan library yang lengkap dan mendukung standard bahasa C dan C++.
2. Memiliki fasilitas untuk meng-upload program langsung dari IDE Arduino tanpa menggunakan hardware tambahan dan juga mampu digunakan dengan dukungan *software* pihak kedua seperti processing.
3. Memiliki fasilitas serial monitor tersendiri yang terintegrasi di dalam IDE Arduino sehingga dapat digunakan untuk membantu pengecekan program yang menggunakan fasilitas komunikasi serial.
4. Memiliki kemampuan *interfacing software* dengan Python, Instant Reality (X3D), Flash, Processing, PD (*Pure Data*), MaxMSP, VVVV, Director, Ruby, C, Linux TTY, SuperCollider, Second Life, Squeak, Mathematica, C++



Gambar 2.1. Arduino IDE (Sumber gambar: software ArduinoIDE)



Gambar 2.2. Arduino Web Editor. (Sumber gambar: <https://www.arduino.cc/>)

D. NodeMCU ESP-8266

NodeMCU adalah sebuah platform IoT yang bersifat *opensource*. Terdiri dari perangkat keras berupa System on Chip (SoC) ESP8266 buatan Espressif System, *firmware* dari NodeMCU menggunakan bahasa pemrograman *scripting* Lua, bersifat *open source* dan banyak tersedia salah satunya oleh *user* di GitHub seperti. Istilah NodeMCU secara umum sebenarnya mengacu pada *firmware* yang digunakan dari perangkat keras *development kit*. NodeMCU ESP8266 adalah sebuah komponen *chip* terintegrasi yang didesain untuk keperluan dunia masa kini yang serba tersambung. *Chip* ini menawarkan solusi *networking* Wi-Fi yang lengkap dan menyatu, yang dapat digunakan sebagai penyedia aplikasi atau untuk memisahkan semua fungsi *networking* Wi-Fi ke pemroses aplikasi lainnya.

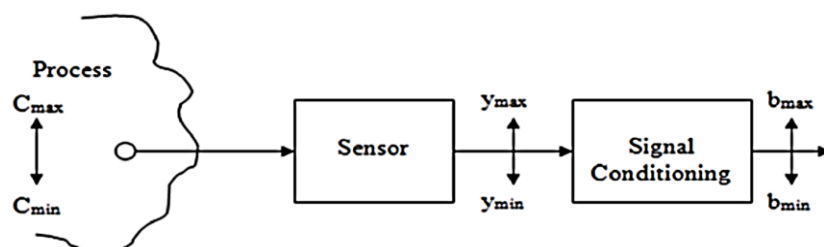
NodeMCU dapat dianalogikan sebagai *board* arduino yang terkoneksi dengan ESP8266. NodeMCU telah terhubung langsung ESP8266 ke dalam sebuah board yang sudah terintegrasi dengan berbagai fitur selengkapnya mikrokontroler dan kapasitas akses terhadap Wi-Fi dan juga chip komunikasi yang berupa USB to *serial*. Sehingga dalam pemrograman hanya membutuhkan kabel data USB. Karena Sumber utama dari NodeMCU adalah ESP8266 khususnya seri ESP-12 yang termasuk ESP-12E. Maka fitur – fitur yang dimiliki oleh NodeMCU akan lebih kurang serupa dengan ESP-12



Gambar 2.3. *Board NodeMCU*

E. Sensor

Sensor merupakan perangkat elektronik yang mengukur atribut fisik seperti suhu, tekanan, jarak, kecepatan, torsi, percepatan, dan lain-lain. Proses pembacaan sensor didasari oleh pendeteksian dari perubahan sinyal-sinyal atau gejala-gejala pada suatu energi tertentu. Sinyal yang dideteksi kemudian diinterpretasikan ke dalam format yang dapat dibaca atau diteruskan ke tahap berikutnya untuk diproses lebih lanjut. Sensor tidak dapat dihubungkan langsung ke perangkat yang direkam, memantau atau memproses sinyal. Ini karena sinyal terlalu lemah, sehingga sinyal dari sensor harus diperbesar.



Gambar 2.4 Rangkaian *Signal Conditioning*

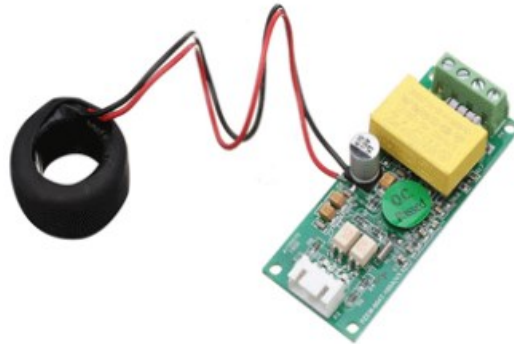
(Sumber gambar: <https://slideplayer.info/slide/12928834/>)

Pada Gambar 2.4 menjelaskan bahwa rangkaian penyesuaian sinyal adalah seperangkat pengkondisian sinyal yang dapat mengubah sinyal menjadi sinyal lain yang diinginkan dan memasang satu set amplifier (*amplifier*) sehingga sinyal yang diperoleh dari sensor (~ 10 mV) dapat diubah menjadi sinyal orde hingga $0 \sim 5$ V.

Pada dasarnya, diklasifikasikan dibagi atas sensor analog dan sensor digital. Sensor analog mengukur nilai pasti dari variabel fisik, sedangkan sensor digital mengukur status. Ada beberapa klasifikasi sensor yang dapat ditemui, seperti klasifikasi berdasarkan sinyal energi yang dibaca, teknologi yang digunakan, fisik sensor, dll

F. Sensor PZEM 004T

PZEM-004T adalah sebuah modul elektronik yang berfungsi untuk mengukur *voltage* / tegangan, arus, daya, frekuensi, energi dan power faktor. Dengan kelengkapan fungsi / fitur ini, maka modul PZEM-004T sangat ideal untuk digunakan sebagai *project* maupun eksperimen alat pengukur daya pada sebuah jaringan listrik seperti rumah atau gedung. Bentuk dari sensor PZEM-004T ditunjukkan pada gambar 2.5.



Gambar 2.5. Sensor PZEM 004T dan *split core*
(Sumber gambar: nn-digital.com)

Spesifikasi PZEM 004T yaitu sebagai berikut,

1. Tegangan kerja: 80 ~ 260VAC
2. Daya maksimum: 100A / 22000W
3. Frekuensi kerja: 45-65Hz
4. Akurasi pengukuran: 1.0

G. Sensor DHT

Sensor DHT dapat mengukur dua parameter sekaligus yaitu suhu dan kelembaban. Sensor ini banyak digunakan lantaran keandalannya dan kepraktisannya. Walaupun mengukur dua parameter sekaligus sensor ini memiliki bentuk yang praktis dan kecil sehingga cocok dipakai di ruangan manapun. Sensor DHT ditunjukkan pada gambar 2.6



Gambar 2.6. Bentuk sensor DHT22

(Sumber gambar: <https://www.communica.co.za/>)

Pin out DHT22

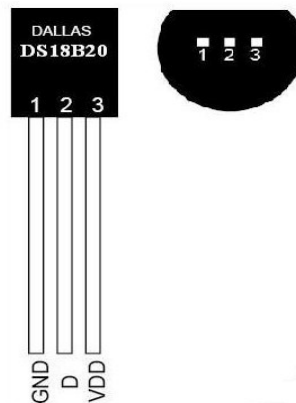
1. Pin Output digital.
2. Pin GND
3. Pin VCC = 5V

Spesifikasi sensor DHT22 yaitu sebagai berikut,

1. Tegangan kerja = 3.3V-5V.
2. Arus maksimum = 2.5mA
3. *Range* pengukuran kelembaban = 0%-100%
4. Akurasi pengukuran kelembaban = 2-5%
5. *Range* pengukuran suhu = -40°C-80°C
6. Akurasi pengukuran suhu = 0.5°C
7. Kecepatan pengambilan sampel tidak lebih dari 0.5 Hz
(pembaruan data setiap 2 detik)
8. Ukuran = 15.1 mm x 25 mm x 7.7 mm

H. Sensor DS18B20

Sensor DS18B20 merupakan sensor digital yang memiliki 12-bit ADC internal. Sangat presisi, sebab jika tegangan referensi sebesar 5Volt, maka akibat perubahan suhu, ia dapat merasakan perubahan terkecil sebesar $5/(2^{12}-1) = 0.0012$ Volt ! Pada rentang suhu -10 sampai +85 derajat Celcius, sensor ini memiliki akurasi +/-0.5 derajat. Sensor ini bekerja menggunakan protokol komunikasi 1-wire (*one-wire*).



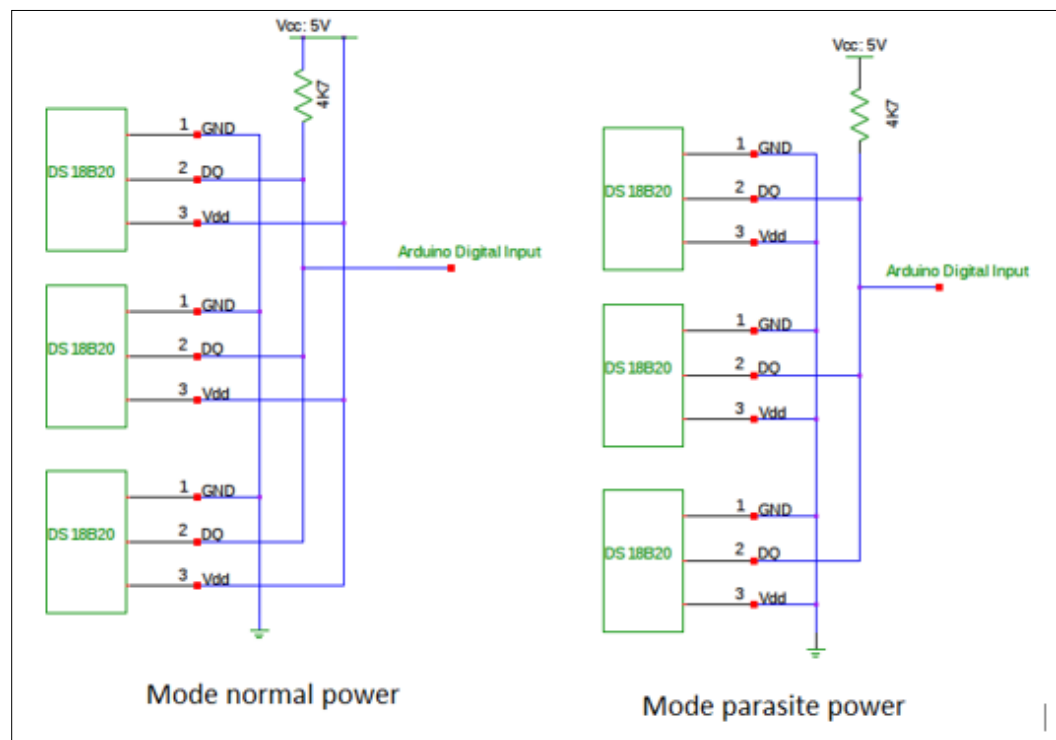
Gambar 2.7. Pin out sensor DS18B20

(Sumber gambar: <https://os.mbed.com>)

DS18B20 terdiri dari tiga kaki, yaitu GND (ground, pin 1), DQ (Data, pin 2), VDD (power, pin 3). Pada Arduino, VDD dikenal sebagai VCC. Dalam hal ini, kita asumsikan VCC sama dengan VDD. Tergantung mode konfigurasi, ketiga kaki IC ini harus dikonfigurasi terlebih dahulu. Sensor dapat bekerja dalam dua mode, yaitu mode normal power dan mode parasite power. Pada mode normal, GND akan terhubung dengan ground, VDD akan terhubung dengan 5V dan DQ akan terhubung dengan pin Arduino, namun

ditambahkan resistor pull-up sebesar 4,7k. Mode ini sangat direkomendasikan pada aplikasi yang melibatkan banyak sensor dan membutuhkan jarak yang panjang.

Pada mode parasite, GND dan VDD disatukan dan terhubung dengan *ground*. DQ akan terhubung dengan pin Arduino melalui resistor pull-up. Pada mode ini, power diperoleh dari power data. Mode ini bisa digunakan untuk aplikasi yang melibatkan sedikit sensor dalam jarak yang pendek. Mode normal power dan mode parasite power ditunjukkan pada gambar 2.8.



Gambar 2.8 Mode sensor SD18B20

(Sumber gambar: <https://learn.openenergymonitor.org/>)

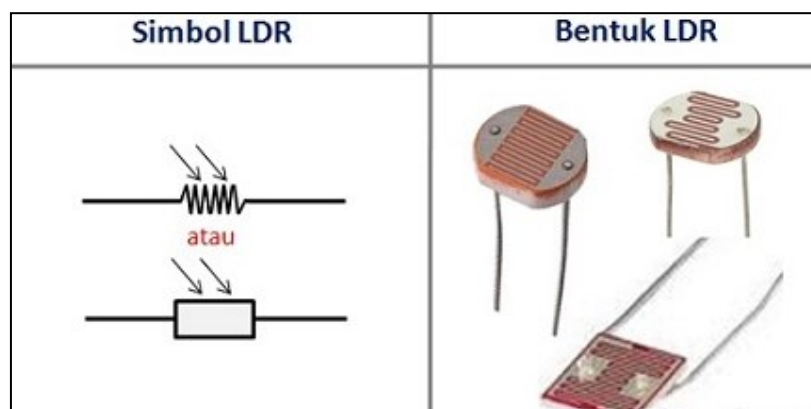
Spesifikasi sensor suhu DS18B20

- Power supply: 3V – 5,5 V
- Konsumsi arus: 1 mA

- c. Range suhu: -55 sampai 125°C
- d. Akurasi: $\pm 0,5\%$
- e. Resolusi: 9 – 12 bit
- f. Waktu konversi: < 750 ms

I. Sensor LDR

Light Dependent Resistor atau disingkat dengan LDR adalah jenis Resistor yang nilai hambatan atau nilai resistansinya tergantung pada intensitas cahaya yang diterimanya. Nilai Hambatan LDR akan menurun pada saat cahaya terang dan nilai Hambatannya akan menjadi tinggi jika dalam kondisi gelap. Dengan kata lain, fungsi LDR (*Light Dependent Resistor*) adalah untuk menghantarkan arus listrik jika menerima sejumlah intensitas cahaya (kondisi terang) dan menghambat arus listrik dalam kondisi gelap. Simbol dan bentuk LDR ditunjukkan pada gambar 2.9.

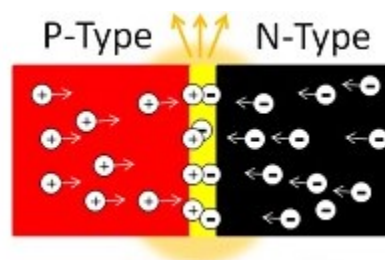


Gambar 2.9. Simbol dan bentuk LDR.
(Sumber gambar: teknikelektronika.com)

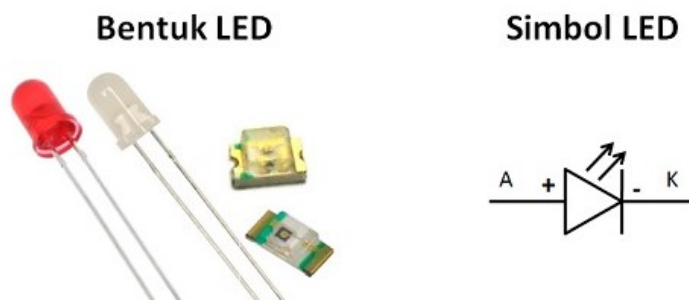
J. Led

Light Emitting Diode atau sering disingkat dengan LED adalah komponen elektronika yang dapat memancarkan cahaya monokromatik

ketika diberikan tegangan maju. LED merupakan keluarga dioda yang terbuat dari bahan semikonduktor. Warna-warna cahaya yang dipancarkan oleh LED tergantung pada jenis bahan semikonduktor yang dipergunakannya. LED juga dapat memancarkan sinar inframerah yang tidak tampak oleh mata seperti yang sering kita jumpai pada *remote control* TV ataupun perangkat elektronik lainnya. Cara kerjanya pun hampir sama dengan Dioda yang memiliki dua kutub yaitu kutub Positif (P) dan Kutub Negatif (N). LED hanya akan memancarkan cahaya apabila dialiri tegangan maju (*bias forward*) dari Anoda menuju ke Katoda. LED terdiri dari sebuah chip semikonduktor yang di doping sehingga menciptakan junction P dan N. Yang dimaksud dengan proses doping dalam semikonduktor adalah proses untuk menambahkan ketidakmurnian (*impurity*) pada semikonduktor yang murni sehingga menghasilkan karakteristik kelistrikan yang diinginkan. Ketika LED dialiri tegangan maju atau *bias forward* yaitu dari Anoda (P) menuju ke Katoda (K), Kelebihan Elektron pada N-Type material akan berpindah ke wilayah yang kelebihan *Hole* (lubang) yaitu wilayah yang bermuatan positif (P-Type material). Saat elektron berjumpa dengan *Hole* akan melepaskan photon dan memancarkan cahaya monokromatik (satu warna). Struktur LED diperlihatkan pada gambar 2.10., serta bentuk dan simbol LED pada gambar 2.11.



Gambar 2.10. Struktur LED
(Sumber gambar: teknikelektronika.com)



Gambar 2.11. Bentuk dan simbol LED.

K. Power Supply

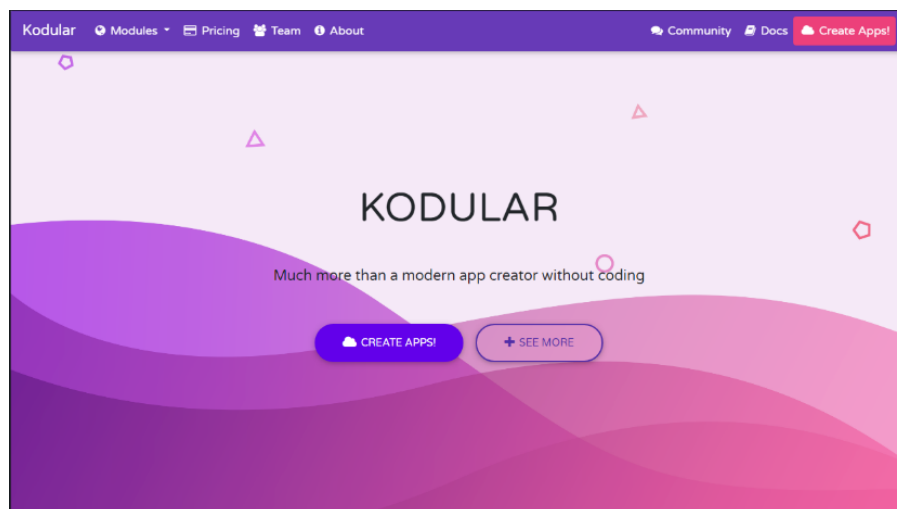
Power supply adalah sebuah rangkaian yang berguna untuk mengubah tegangan AC yang tinggi menjadi DC yang rendah. *Power supply* merupakan sebuah alternatif pengganti dari tegangan DC (seperti; baterai, Aki) karena penggunaan tegangan AC lebih lama dan setiap orang dapat menggunakannya asalkan ada aliran listrik di tempat tersebut. Pada sebuah adaptor terdapat beberapa bagian atau blok yaitu trafo (transformator), *rectifier* (penyearah) dan *filter* (penyaring). *Power supply* diperlihatkan pada gambar 2.12.



Gambar 2.12. *Power supply* (Sumber gambar: <https://ecadio.com>)

L. Kodular

Kodular adalah situs web yang menyediakan *tools* yang menyerupai MIT App Inventor untuk membuat aplikasi android dengan menggunakan *block programming*. Dengan kata lain, pengguna tidak perlu mengetik kode program secara manual untuk membuat aplikasi android. Tampilan kodulator ditunjukkan pada gambar 2.13.



Gambar 2.13. Tampilan website kodular. (Sumber: <https://www.kodular.io>).

Kodular ini menyediakan kelebihan fitur yakni kodular *store* dan kodular extension IDE (sekarang menjadi *AppyBuilder Code Editor*) yang bisa memudahkan developer melakukan unggah (*upload*) aplikasi android ke dalam kodular store, melakukan dalam pembuatan blok program extension IDE sesuai dengan keinginan developer. Kodular dan *AppyBuilder* telah bersatu dalam kontribusi untuk menciptakan orang-orang yang masih awam atau tidak ada kemampuan *coding* bisa membuat aplikasi android sendiri dengan fitur dan layanan hampir mirip dengan android studio secara simpel dan mudah.

Kodular ini bisa melakukan kustom tema (*theme*) sesuai dengan keinginan agar nyaman saat menggunakan situs tersebut dalam membuat/menciptakan aplikasi android sehingga tetap nyaman antara mata dengan layar PC atau laptop. File eksistensi dari Kodular adalah (.aia) dan plugin eksistensinya (.aix). Plugin eksistensi isi berisi beberapa kode perintah dalam bahasa pemrograman Java (.java) yang akan mengkonversi menjadi file plugin eksistensi (.aix), ini berguna bagian extension.

M. Internet of Things

Internet of Things (IoT) adalah suatu konsep mengenai komunikasi benda-benda di sekitar kita dengan suatu jaringan. IoT mengacu pada interkoneksi jaringan benda sehari-hari, yang sering dilengkapi dengan kecerdasan. Konsep IoT bertujuan untuk membuat internet semakin berkembang dan meluas. Selanjutnya, dengan memungkinkan akses dan interaksi yang mudah dengan berbagai perangkat seperti, peralatan rumah tangga, kamera cctv, sensor pemantauan, aktuator, *display*, kendaraan, dan sebagainya. Konsep IoT secara sederhana bekerja dengan mengacu pada 3 elemen utama, yakni: Barang fisik yang dilengkapi modul IoT, perangkat koneksi ke internet seperti modem atau *router wireless*, dan *cloud data center* yang merupakan tempat untuk menyimpan aplikasi beserta data atau informasi. Dasar prinsip kerja perangkat IoT adalah benda di dunia nyata diberikan identitas unik untuk dikenali oleh sistem komputer dan dapat direpresentasikan dalam bentuk data.

Perangkat IoT bisa sekecil bola lampu yang bisa dinyalakan dan dimatikan menggunakan *smartphone*, atau bisa sebesar mobil tanpa pengemudi yang dilengkapi puluhan sensor. Tanpa ragu, saat ini, sebagian besar organisasi telah menjadi berbasis data, dan dengan IoT, data ini diubah menjadi wawasan waktu nyata yang dapat ditindaklanjuti. Selain itu, IoT terkait dengan tiga konsep utama lainnya: otomatis, aman, dan dengan sedikit usaha. Menurut *international data corporation* (yang merupakan penyedia global utama intelijen pasar, layanan konsultasi), pengeluaran global untuk IoT akan melebihi 1 triliun dolar AS pada tahun 2022. Tujuan utama IoT adalah membuat segala sesuatu menjadi cerdas secara *virtual* dengan bantuan teknologi seperti kecerdasan buatan, *big data*, dan lain sebagainya. Analisis data saat ini memiliki beberapa keterbatasan, tetapi dengan IoT dalam gambar, mereka dapat ditingkatkan untuk memberikan gambaran yang akurat. Semua data yang dikumpulkan dari perangkat IoT memerlukan semacam mekanisme konektivitas. Dan mekanisme ini digunakan untuk mengirim data ke semacam hub terpusat seperti infrastruktur cloud. Ini media konektivitas/komunikasi antara lain *bluetooth*, Wi-Fi, WAN, dan sebagainya.

Perangkat IoT bekerja melalui integrasi perangkat lunak dan perangkat keras dalam menyelesaikan tugas tertentu. Perangkat keras datang dalam berbagai bentuk, prosesor dalam pengontrolan, sensor yang mengumpulkan informasi dari dunia fisik, perangkat konektivitas sebagai media perantara komunikasi antar perangkat. Sedangkan perangkat lunak berperan sebagai pemberi instruksi kepada perangkat keras untuk menjalankan suatu fungsi yang telah terprogram. Beberapa tugas perangkat lunak pada IoT seperti

sebagai pengelola dan penyajian data, penghubung antar perangkat pada sebuah sistem dan lain sebagainya. Perangkat Keras IoT mencakup berbagai perangkat yang saling terhubung untuk mencapai sebuah tujuan. Perangkat IoT mengelola tugas dan fungsi utama seperti komunikasi antar sistem, monitoring, deteksi masalah dan penyelesaiannya. Pada perangkat keras IoT terdiri dari sekumpulan perangkat dengan masing-masing fungsi yang berbeda untuk saling mendukung dalam menyelesaikan sebuah masalah. Semisal sensor untuk mengumpulkan data dari dunia fisik, mikroprosesor pengontrol perangkat yang saling terhubung pada sebuah *board* dan memproses data, serta sistem komunikasi yang mengirim informasi ke *user*.

Proses komunikasi antar perangkat terdapat media yang menghubungkan perangkat satu dengan perangkat lainnya. Setidaknya ada dua jenis media penghubung yaitu kabel dan nirkabel yaitu tanpa kabel yang disebut juga *wireless*. Untuk perangkat IoT sendiri dari sisi pengguna hampir seluruhnya menggunakan nirkabel, tentu kita tidak bisa bayangkan jika perangkat IoT seperti *smartwatch* ataupun smart TV diakses oleh user dengan menggunakan kabel. Secara umum terdapat dua jenis model yang digunakan dalam membentuk sebuah jaringan yaitu *peer to peer* dan juga *client server* yang nantinya dalam buku ini akan disimulasikan bagaimana membentuk jaringan IoT dengan menggunakan *client server*.

Komunikasi antar perangkat komputer dengan sebuah objek merupakan fondasi utama dan merupakan awal mula pengembangan konsep dan realisasi dari *Internet of Things* secara lengkap, protokol komunikasi memberikan akses dan kemampuan untuk berkomunikasi antara perangkat,

sensor, mesin dan dunia luar. Protokol komunikasi terdapat pada *layer network* yang bertanggung jawab pada bagian transfer informasi melalui teknologi jaringan seperti ZigBee, GSM, 3G, 4G, UMTS, Wi-Fi, Infrared, Z-Wire, LoRa, 6LoWPAN, IPv6 (Shah and Yaqoob 2016). Berbagai macam teknologi jaringan yang tersedia memiliki karakteristik dan kemampuan yang berbeda-beda dalam hal mentransfer informasi, konsumsi daya, cakupan area dan masalah keamanan data, hal ini diatur pada standar IEEE 802.15.4.

Secara umum arsitektur perangkat keras pada IoT terdiri dari sensor dan aktuator, mikrokontroler, perangkat komunikasi dan perangkat *end user*.

1. Sensor dan aktuator: berhubungan dengan kemampuan untuk merasakan dan berinteraksi dengan dunia fisik.
2. Mikrokontroler: berhubungan dengan perangkat yang menjalankan dan mengontrol perangkat-perangkat lain.
3. Perangkat komunikasi: berhubungan dengan kemampuan perangkat dalam berkomunikasi. Baik dalam bentuk komunikasi kabel maupun komunikasi *wireless* atau *transceiver* radio.
4. Perangkat *end user*: berhubungan dengan perangkat yang digunakan oleh end user dalam berinteraksi. Interaksi dapat berupa kendali dan pemantauan jarak jauh seperti *smartphone*, *tablet* dan *computer*.

N. Komunikasi pada IoT

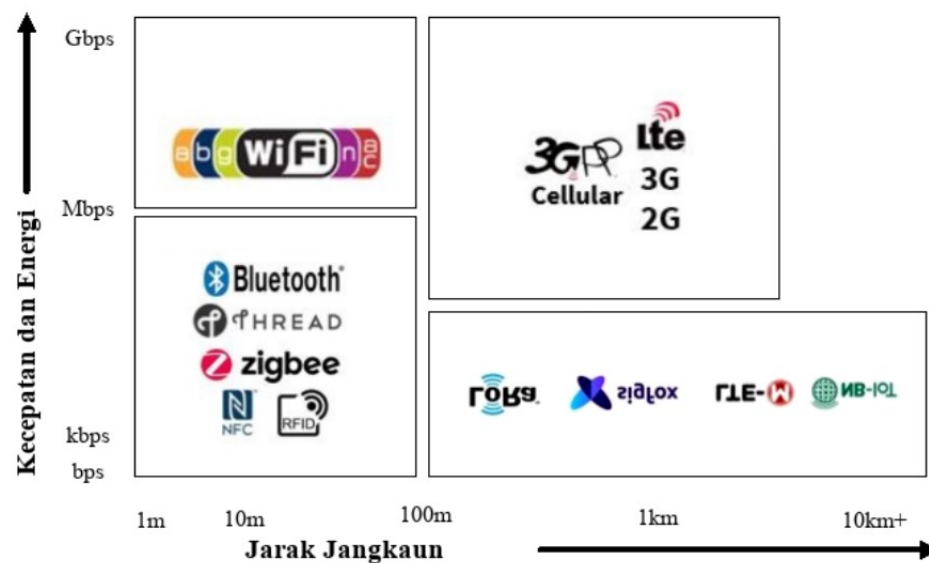
Konektivitas merupakan kunci untuk penerapan IoT. Protokol komunikasi telah banyak dikembangkan dengan berbagai macam model dan kemampuan untuk melayani komunikasi antar perangkat. Setiap teknologi memiliki kelebihan dan kelemahannya masing-masing dalam hal cakupan,

jangkauan, skalabilitas, biaya, dan persyaratan jaringan yang akan menangani kasus penggunaan IoT dan kebutuhan aplikasi tertentu. Pemilihan perangkat komunikasi tentunya perlu mempertimbangkan kemampuan dari perangkat tersebut, seperti *throughput* maksimum, jangkauan jarak, ketersediaan di zona penerapan, tetapi juga konsumsi daya. Terkadang sebuah perangkat dapat mencakup jangkauan yang luas tetapi memerlukan energi yang besar. Ada pula perangkat yang mampu mencakup jangkauan yang luas dan energi yang kecil, akan tetapi *bandwidth* yang terbatas.

Teknologi konektivitas pada IoT secara umum dapat dikategorikan menjadi 2 bagian, yaitu:

1. Teknologi *Wireless Personal and Local Area Network* (WPAN\LAN), jenis teknologi ini biasanya digunakan untuk jarak pendek sampai menengah seperti pada smart home dan sejenisnya. Perangkat yang termasuk pada teknologi ini seperti Bluetooth, ZigBee, 6LowPAN, RFID, NFC, dan Wi-Fi.
2. Teknologi *Wireless Wide Area Network* (WWAN), jenis teknologi ini bagi atas dua yaitu: Seluler (3G/4G dan 5G) dan *Low-Power Wide Area Networks* (LPWANs). LPWANs mencakup LTE-M, NB-IoT EC-GSM, LoRa, Sigfox, dan lain-lain. Teknologi LPWANs sangat cocok digunakan untuk komunikasi jarak jauh seperti pemantau kendaraan, pelacakan hewan, dll. (Anani, Ouda and Hamou, 2019).

Grafik perbandingan perangkat komunikasi ditunjukkan pada gambar 2.14.



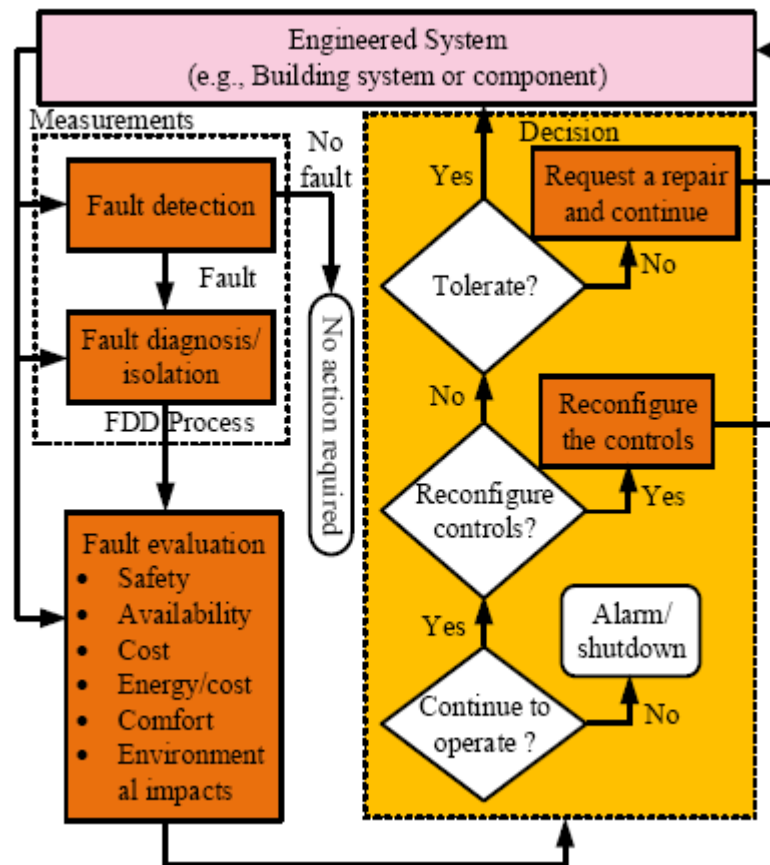
Gambar 2.14. Grafik perbandingan perangkat komunikasi

O. Fault Detection and Diagnosing (FDD)

FDD adalah proses secara otomatis mengetahui dan mengisolasi kesalahan dalam BEMS untuk perlindungan sistem dari kerusakan. Beberapa aplikasi FDD dari BEMS (*Building Energy Management System*) dikembangkan dan diteliti berdasarkan hubungan antara suhu, tekanan, dan termodinamika untuk deteksi dan diagnosis kesalahan. Misalnya, pada pertengahan 1990-an, alat diagnostik pembangunan menyeluruh dikembangkan oleh Departemen Energi Amerika Serikat untuk mendeteksi dan meminimalkan konsumsi energi.

Selanjutnya, peneliti memfokuskan penelitian mereka pada FDD di BEMS. Pada (Hannan *et al.*, 2018), aplikasi sederhana FDD diteliti dengan

mengidentifikasi 4 langkah yaitu pemantauan dan deteksi, diagnostik kesalahan, evaluasi kesalahan, dan tahap pengambilan keputusan seperti ditunjukkan pada Gambar 2.15.



Gambar 2.15. Aplikasi sederhana dari *Fault Detection and Diagnosis* (Sumber gambar: Hannan *et al.*, 2018)

Model yang berbeda, seperti model *fuzzy*, model fisik, dan teknik FDD berbasis jaringan saraf telah dikembangkan untuk FDD. Berdasarkan pada proses pengukuran kesalahan, metode FDD dapat diklasifikasikan sebagai FDD berbasis model, FDD berbasis sinyal, FDD berbasis pengetahuan, FDD aktif, dan FDD hybrid (Hannan *et al.*, 2018). Dalam FDD berbasis model,

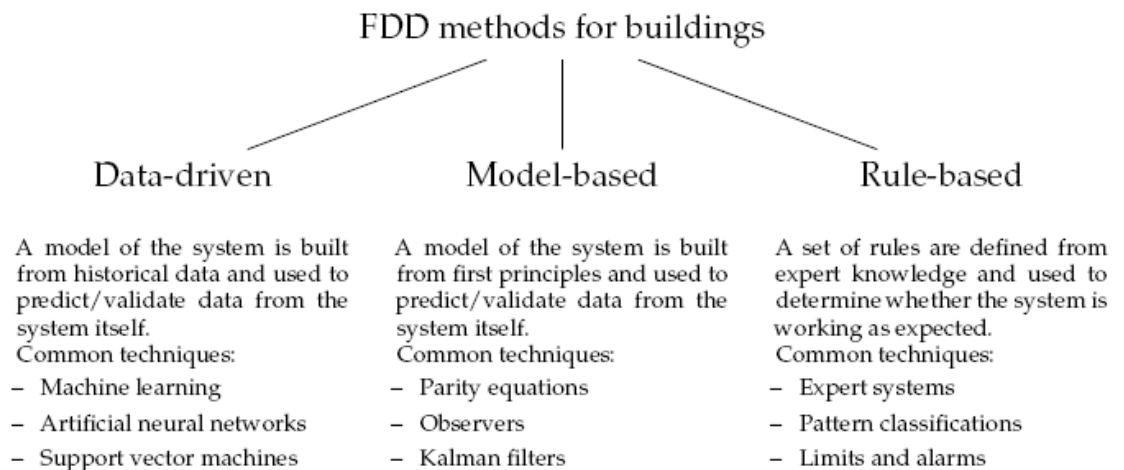
output terus dipantau dan dibandingkan dengan data yang diprediksi deteksi kesalahan.

FDD berbasis model adalah metode yang paling akurat karena bergantung pada metode prinsip pertama dalam bentuk fisik (Hannan *et al.*, 2018) atau mungkin sepenuhnya didorong oleh model *black-cox*. Dalam FDD berbasis sinyal, teknik domain waktu dan frekuensi digunakan. Jika suatu kesalahan terjadi dalam suatu sistem, maka sinyal keluaran yang diukur dari sistem yang rusak berbeda dari keluaran dari sistem yang asli. Metode FDD berbasis pengetahuan menggunakan metode kecerdasan buatan untuk mengevaluasi data waktu nyata dan mengekstrak pengetahuan dari data historis. Dalam FDD aktif, sinyal uji disuntikkan untuk meningkatkan deteksi kesalahan. Sementara itu, FDD hibrid menggabungkan metode model driven dan data-driven. Dalam sebagian besar teknik FDD yang ada, sensor yang digunakan untuk menerapkan keputusan dipilih sesuai dengan keahlian manusia (Hannan *et al.*, 2018).

Namun, manajemen data, biaya, dan skalabilitas adalah keterbatasan utama teknik FDD. Analisis statistik konsumsi energi oleh pengguna akhir yang berbeda diuraikan dan pada Studi sebelumnya metode FDD yang mencakup ukuran, pemeliharaan, dan kalibrasi, dari sistem pemanas, ventilasi, dan pendingin udara (HVAC) bangunan perlu dipertimbangkan untuk mengurangi energi yang terbuang (Won, Kwae and Woo, 2012).

Kim dan Katipamula menyajikan tinjauan komprehensif metode FDD terbaru untuk membangun sistem kategorisasi metode FDD untuk bangunan yang diadaptasi dari Kim dan Katipamula kategorisasi metode FDD untuk

bangunan yang diadaptasi dari Kim dan Katipamula (Phougat *et al.*, 2017). Metode FDD dikategorikan ke dalam tiga kelompok tergantung pada pendekatan: *data driven methodes*, metode berbasis model, dan metode berbasis aturan, seperti yang ditunjukkan pada Gambar 2.16.



Gambar 2.16. Kategorisasi metode FDD untuk bangunan yang diadaptasi dari Kim dan Katipamula (Phougat *et al.*, 2017).

Pada *Data driven methods*, model sistem yang diuji dari data historis dan digunakan untuk memvalidasi data saat ini dari sistem. Ada beberapa teknik, seperti pembelajaran mesin, *artificial neural network* (ANN) atau *support vector machine* (SVM). Pada metode ini tidak diperlukan pengetahuan fisik tentang sistem dan model yang dihasilkan dapat diperlakukan sebagai komponen. Metode ini mudah diterapkan pada beberapa jenis sistem dengan sumber data historis yang diperlukan untuk membuat model, selain sistem yang baru digunakan. Metode ini membutuhkan data yang tepat sehingga model yang dihasilkan tidak akan mengenali kesalahan sebagai perilaku yang

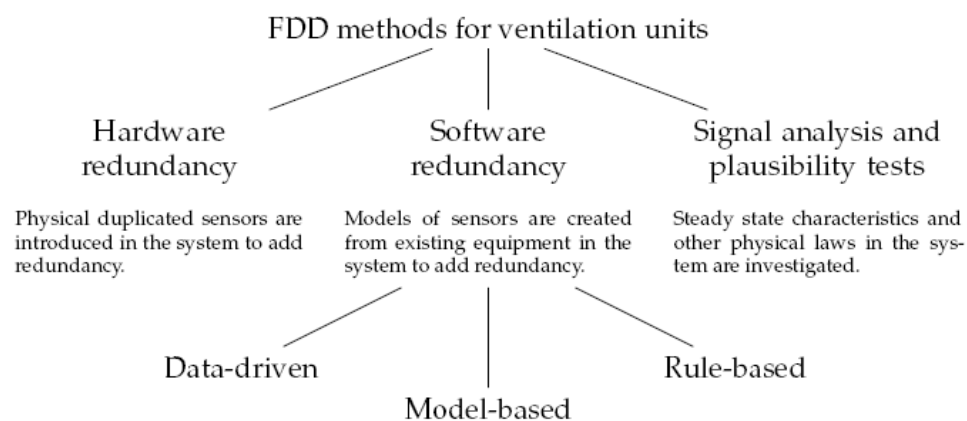
benar. Untuk melakukan diagnosa yang tepat dan mengidentifikasi kesalahan yang tepat biasanya diperlukan data historis yang rusak.

Pada metode berbasis model (*model based*), model fisik sistem diuji dan dibuat dan digunakan untuk memvalidasi data saat ini dari sistem. Pendekatan ini tidak memerlukan data input dan seringkali prediksi lebih akurat daripada *black board model*. Namun, model yang akurat bisa jadi kompleks dan membutuhkan pengetahuan mendalam tentang sistem untuk melakukan estimasi parameter untuk meningkatkan akurasi dan memerlukan data.

Metode berbasis aturan (*rule based*), pengetahuan ahli yang dikumpulkan dari para ahli lapangan digunakan untuk merancang satu set aturan yang menggambarkan perilaku sistem. Tidak ada data historis dan tidak ada pengetahuan fisik rinci tentang sistem diperlukan. Selain itu, beberapa kesalahan memiliki efek yang dapat dijelaskan oleh aturan yang dibuat mungkin untuk mengidentifikasi dan mendiagnosis masalah dengan tepat. Namun, aturan hanya bisa menggambarkan perilaku hingga kompleksitas tertentu dan hanya dapat mencakup kasus-kasus sederhana. Dengan bertambahnya jumlah aturan, kemungkinan peraturan yang saling bertentangan meningkat dan demikian pula upaya untuk mempertahankan seperangkat aturan.

Penelitian Yu et al. menyajikan ulasan teknik FDD untuk unit ventilasi (Yu, Woradechjumroen and Yu, 2014). Dalam hal ini, penulis mengklasifikasikan teknik FDD menjadi empat kelompok: redundansi perangkat keras, redundansi perangkat lunak, analisis sinyal dan uji logika,

seperti yang ditunjukkan pada gambar 2.17 Beberapa sensor dan aktuator identik mengarah ke perangkat keras redundansi yang memungkinkan akurasi dan presisi tinggi, tetapi juga untuk penyebaran dan pemeliharaan dan biaya yang lebih tinggi. Dalam redundansi perangkat lunak, beberapa sensor fisik digantikan oleh model yang diperoleh dari sensor dalam sistem. Dalam analisis sinyal dan metode pengujian logis, karakteristik *steady-state* dan menginvestigasi sistem dalam bentuk fisik lain. Metode redundansi perangkat lunak lebih lanjut diklasifikasikan dalam *model-based*, *data-driven*, and *rule-based*, seperti pada metode FDD umum.



Gambar 2.17. Klasifikasi Teknik FDD untuk ventilasi (Yu, Woradechjumroen and Yu, 2014)

Karakteristik metode FDD yaitu:

- a. *Quick detection and diagnostics*
- b. *Isolability:*
- c. *Robustness*
- d. *Novel identifiability*
- e. *Classification error estimate*

- f. Adaptability*
- g. Explanation facility*
- h. Modeling requirements*
- i. Storage and computational requirements*
- j. Multiple fault identifiability*

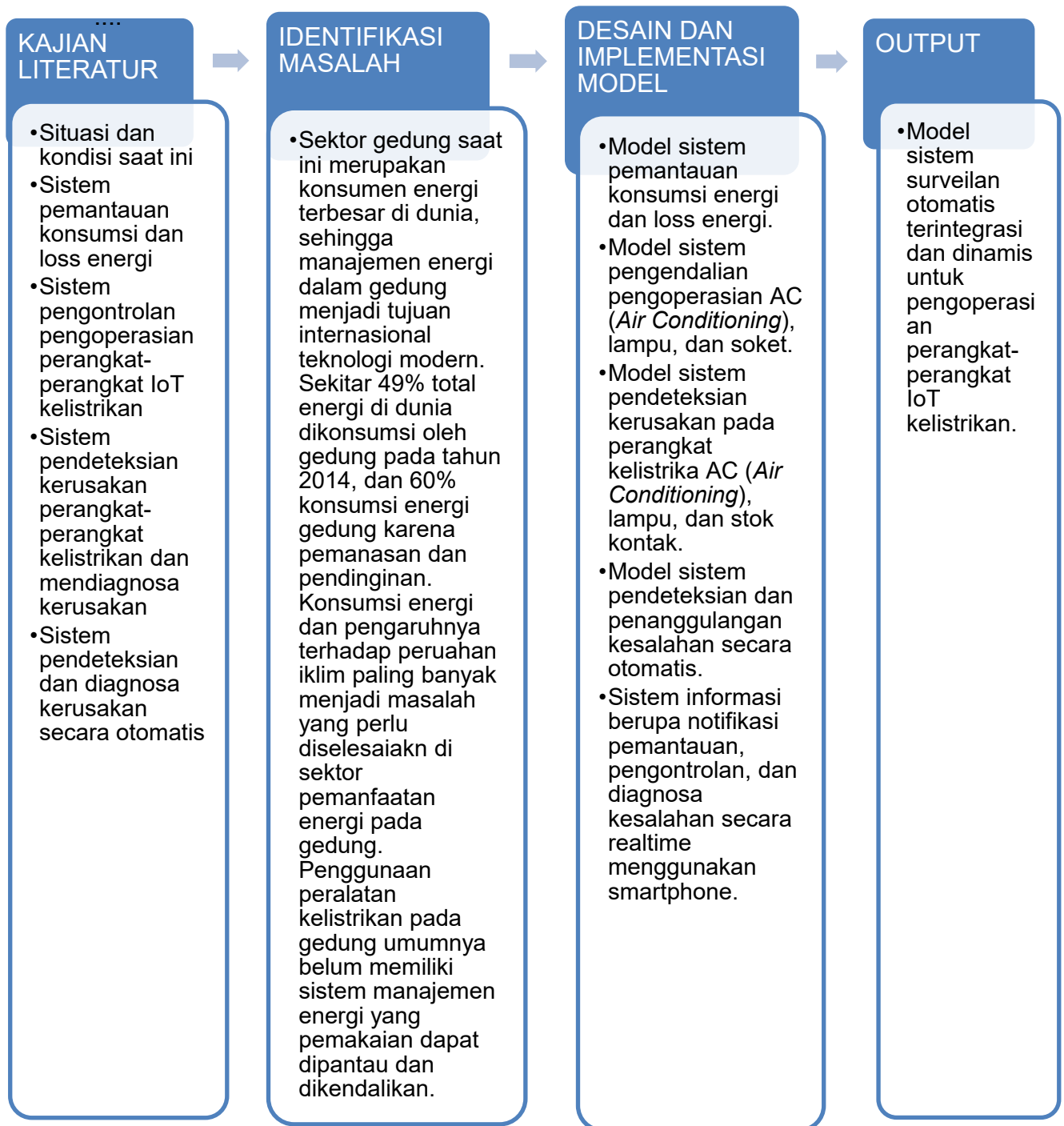
P. Kerangka Pikir

Situasi dan kondisi saat ini sangat dibutuhkan Sistem pemantauan konsumsi dan *loss energi* pada suatu instalasi kelistrikan. Sistem pengontrolan pengoperasian perangkat-perangkat IoT kelistrikan sangat berkembang saat ini dan sangat dibutuhkan oleh pengguna untuk memudahkan pengoperasian perangkat-perangkat kelistrikan yang dimiliki.

Sistem pendeteksian kerusakan perangkat-perangkat kelistrikan dan mendiagnosa kerusakan merupakan salah satu cara untuk penanganan dini pada pengoperasian perangkat kelistrikan untuk menghemat energi yang dikonsumsi. Sistem pendeteksian dan diagnosa kerusakan secara otomatis merupakan sistem yang tidak melibatkan lagi pengguna dalam penanganan terhadap kesalahan atau kerusakan pada perangkat kelistrikan sehingga dapat menghemat energi, memberi kemudahan untuk mengetahui perangkat kelistrikan yang bermasalah dan memberi keamanan terhadap akibat yang dapat ditimbulkan oleh perangkat kelistrikan yang rusak.

Model sistem pemantauan konsumsi energi dan loss energi diperlukan pada instalasi kelistrikan pada bagian panel listrik. Model sistem pengendalian pengoperasian AC (*Air Conditioning*), lampu, dan soket, model sistem pendeteksian kerusakan pada perangkat kelistrikan AC (*Air Conditioning*), lampu, dan soket kontak serta model sistem pendeteksian dan penanggulangan kesalahan secara otomatis yang dilengkapi dengan sistem informasi berupa notifikasi pemantauan, pengontrolan, dan diagnosa kesalahan secara *realtime* menggunakan *smartphone* sangat dibutuhkan pada era digital saat ini dengan dukungan teknologi yang sangat memadai.

Desain model sistem pemantauan otomatis terintegrasi dan dinamis yang terdiri dari sistem pemantauan konsumsi energi dan loss energi, sistem pengendalian pengoperasian AC (*Air Conditioning*), lampu, dan soket. Sistem pendeteksian kerusakan pada perangkat kelistrikan AC (*Air Conditioning*), lampu, dan stok kontak. Model sistem pendeteksian dan penanggulangan kesalahan secara otomatis dengan notifikasi pemantauan, pengontrolan, dan diagnosa kesalahan secara *realtime* menggunakan *smartphone*.



Gambar 2.18. Bagan kerangka pikir penelitian

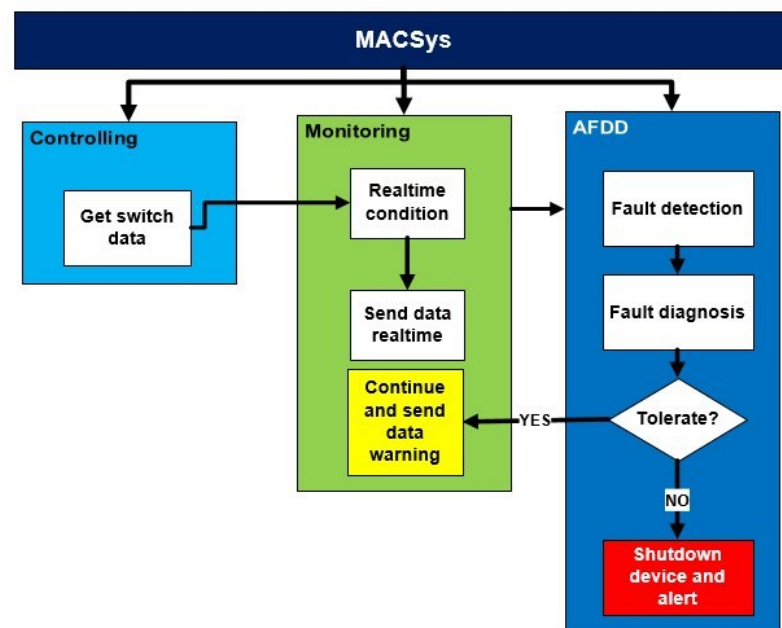
BAB III

METODOLOGI PENELITIAN

A. Rancangan Penelitian

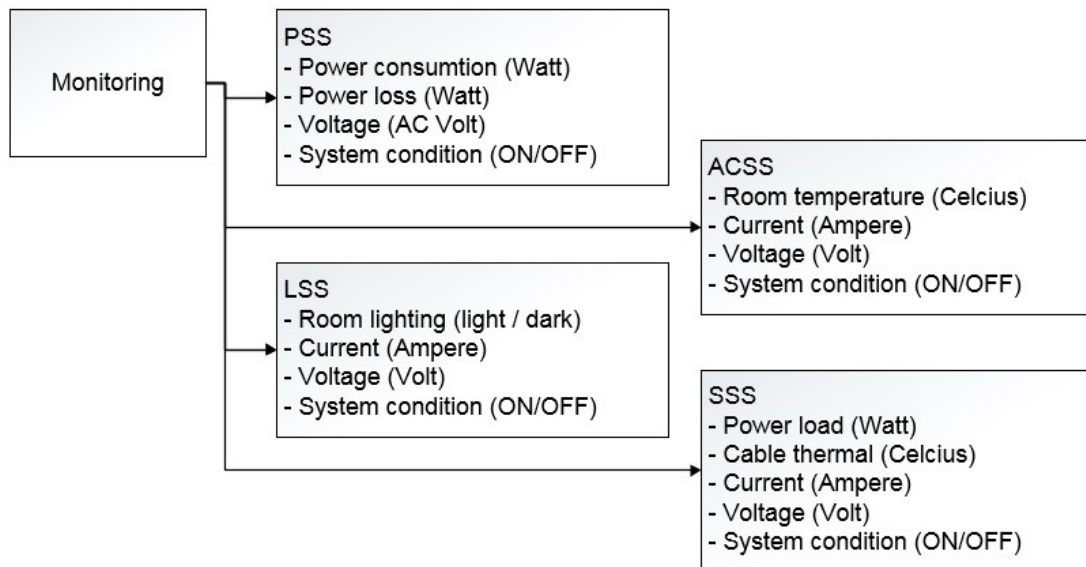
Sebelum membuat rancangan sistem baik perangkat keras maupun perangkat lunak terlebih dahulu dibuat desain modelnya sebagai acuan.

Desain model MACSys ditunjukkan pada gambar 3.1



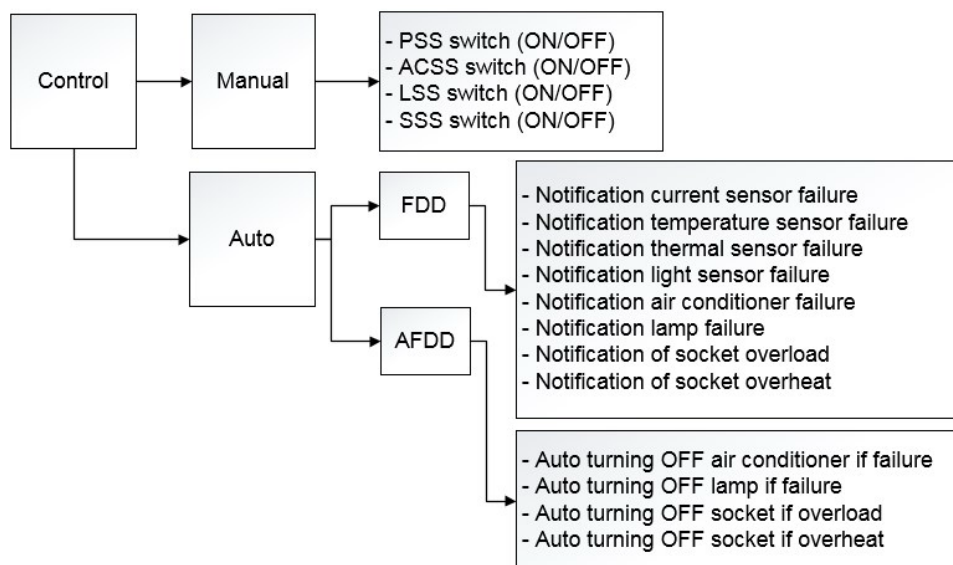
Gambar 3.1 Desain model MACSys

Desain model MACSys terdiri dari sistem *monitoring* dan sistem *controlling*. Sistem monitoring MACSys untuk memonitoring *power consumption*, *power loss*, *voltage* dan *system condition* pada PSS, untuk memonitoring *room lighting*, *current*, *voltage*, dan *system condition* pada LSS, untuk memonitoring *room temperature*, *current*, *voltage* dan *system condition*, untuk momonitoring *power load*, *cable thermal*, *current*, *voltage*, dan *system condition*. System monitoring MACSys ditunjukkan pada gambar 3.2



Gambar 3.2 System Monitoring MACSys

Sistem pengontrolan MACSys terdiri dari sistem pengontrolan *manual* dan *Automatic*. Pengontrolan manual pada *switch* untuk PSS, ACSS, LSS dan SSS. Pengontrolan *automatic* terdiri dari *Fault Detection and Diagnostic* dan *Automatic FDD*. System pengontrolan MACSys ditunjukkan pada gambar 3.3



Gambar 3.3. System Pengontrolan MACSys

Penelitian ini terdiri dari empat tahap yaitu yang pertama studi pendahuluan yang terdiri dari kajian literatur, identifikasi masalah, dan perumusan masalah. Peneliti memulai dengan mempelajari kondisi *existing* serta hasil penelitian terkait sistem pemantauan, pengendalian, pengontrolan, dan diagnosa kerusakan perangkat kelistrikan. Setelah itu identifikasi masalah dengan mengamati konsumsi serta manajemen energi listrik pada instalasi rumah tinggal dan gedung institusi. Kemudian perumusan masalah dengan merumuskan masalah yang terjadi serta tujuan penelitian dan usulan model sistem yang akan digunakan.

Tahap kedua yaitu persiapan penelitian yang terdiri dari desain dan perancangan model, pada tahap ini peneliti menyiapkan alat dan bahan, pembuatan perangkat keras, perangkat lunak, serta aplikasi android.

Tahap ketiga yaitu tahap uji coba, pada tahap ini peneliti melakukan pengujian fungsi *hardware*, fungsi aplikasi android, *respon time sistem*, dan pengujian aplikasi android menggunakan *firebase test lab*.

Tahap keempat yaitu tahap hasil, pada tahap ini telah dihasilkan produk MACSys dan aplikasi android MACSys.app yaitu model surveilan otomatis terintegrasi dan dinamis untuk operasional perangkat-perangkat IoT kelistrikan.

B. Jenis Penelitian

Jenis penelitian yang digunakan pada penelitian ini yaitu *Research and Development*. Penelitian ini bertujuan untuk mengadakan atau melakukan

percobaan dan penyempurnaan dari solusi masalah sebelumnya, sehingga ruang lingkup masalah beserta penyelesaiannya dapat diselesaikan dengan melakukan studi literatur, merancang desain solusi, serta membuat sebuah sistem dari rancangan yang telah didesain. Inti dari pendekatan yang dipilih adalah untuk mengetahui pengaruh pemberian suatu *treatment* atau perlakuan terhadap subjek penelitian dan melihat dampak dari perlakuan tersebut.

C. Lokasi dan Waktu Penelitian

Penelitian mulai dilaksanakan pada bulan Februari 2020. Lokasi penelitian berada di area Laboratorium Departemen Teknik Elektro Fakultas Teknik Universitas Hasanuddin dan rumah kediaman Peneliti untuk implementasi, uji coba dan pengambilan data. Lokasi tersebut dipilih mengingat obyek tersebut dapat merepresentasikan kondisi riil yang sebenarnya sehingga hasil penelitian yang diperoleh nantinya dapat mewakili sistem yang sebenarnya.

D. Teknik Pengumpulan, Validasi, dan Analisis Data

Teknik pengumpulan data yang digunakan pada penelitian ini meliputi pengujian tiap-tiap sub sistem. Tiap sub sistem divalidasi menggunakan alat ukur, untuk *Panel Sub System* (PSS) divalidasi menggunakan KWH Meter dan *tang ampere*, *AC Sub System* (ACSS) divalidasi menggunakan *thermometer digital* dan *tang ampere*, *Lamp Sub System* (LSS) divalidasi menggunakan *Lux meter* dan *tang ampere*, serta *Socket Sub System* (SSS) divalidasi menggunakan *thermometer digital* dan *tang ampere*. Teknik analisis data yang digunakan pada penelitian ini yaitu analisis deskriptif.

E. Parameter Ukur

Pada penelitian ini parameter yang digunakan yaitu konsumsi energi dalam satuan KWH, arus listrik dalam satuan Ampere, tegangan listrik dalam satuan Volt, suhu dalam satuan derajat Celcius, beban listrik dalam satuan Watt, intensitas cahaya dalam satuan Lux.

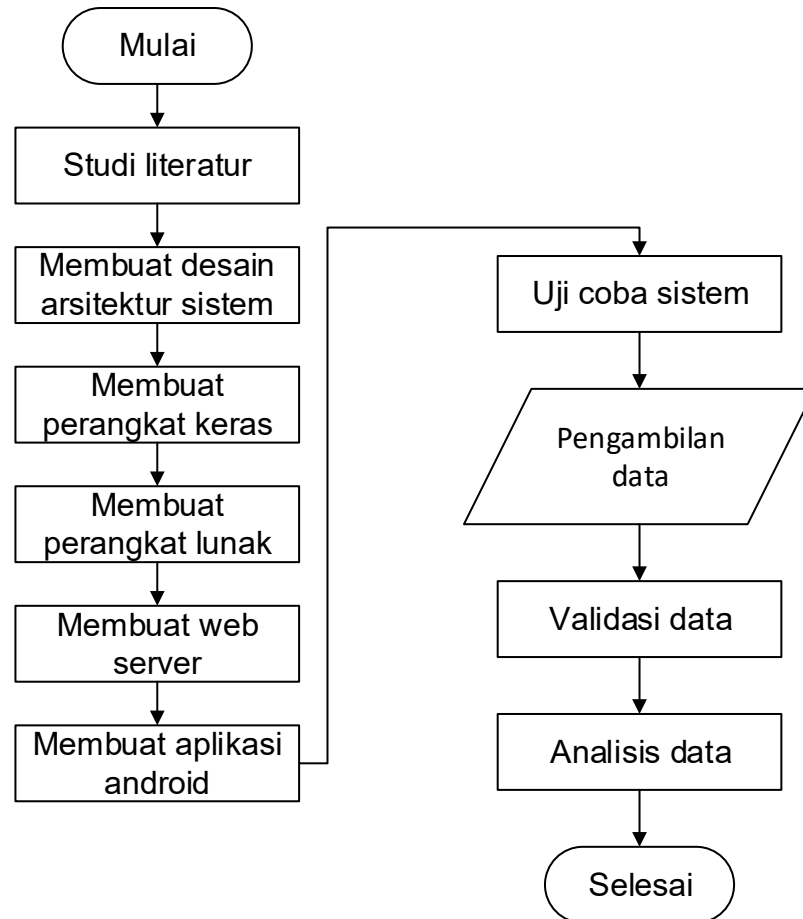
F. Alat Penelitian

Penelitian ini menggunakan perangkat keras dan perangkat lunak untuk pengumpulan data dan analisa data, yaitu sebagai berikut;

1. Perangkat keras, terdiri dari;
 - a. Laptop
 - b. AVO meter
 - c. Tang ampere
 - d. Lux Meter
 - e. KWH meter
 - f. Thermometer digital
 - g. Tool kit
 - h. Laser printer
2. Perangkat lunak, terdiri dari;
 - a. *Microsoft Windows*
 - b. *Microsoft Office*
 - c. *Arduino IDE*
 - d. *Internet browser*

G. Prosedur Penelitian

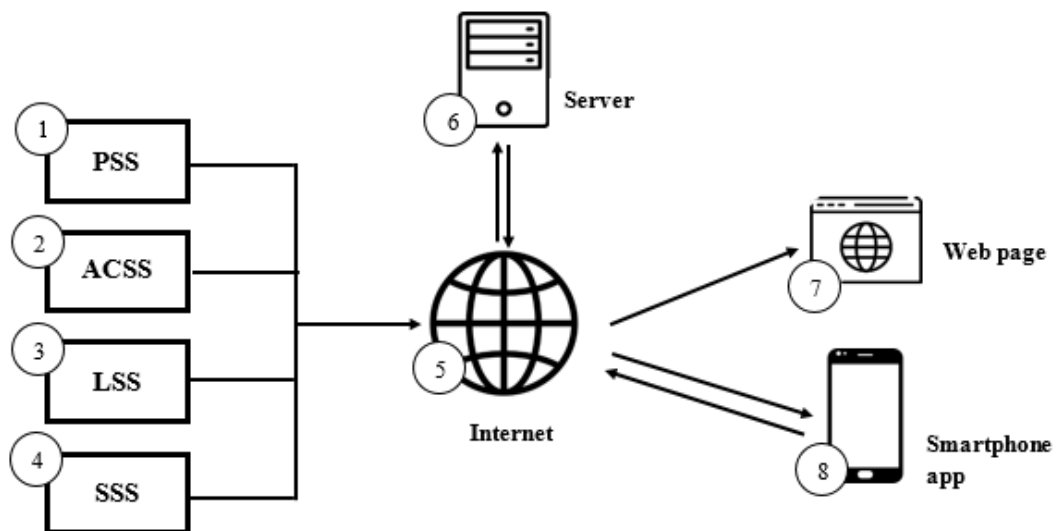
Prosedur penelitian ini dilakukan berdasarkan *flowchart* rancangan penelitian seperti pada gambar 3.2;



Gambar 3.2 *Flowchart* rancangan penelitian

H. Arsitektur Sistem

Sistem yang dibangun menjadi empat bagian yang terpisah yaitu (1) sistem yang memonitoring konsumsi energi dan energi terbuang (*losses energy*) pada panel listrik dinamakan *Panel Sub System* (PSS), (2) sistem yang memonitoring dan mengontrol pengoperasian *Air Conditioner* (AC) dinamakan *Air Conditioner Sub System* (ACSS) dilengkapi dengan deteksi kerusakan dan proteksi, (3) sistem monitoring dan pengontrolan lampu secara manual dan otomatis yang dinamakan *Lamp Sub System* (LSS), dan (4) sistem monitoring dan pengontrolan stok kontak yang dilengkapi dengan pendeteksi kelebihan suhu, kelebihan beban, dan proteksi yang dinamakan *Socket Sub System* (SSS). Arsitektur sistem ditunjukkan pada gambar 3.3.



Gambar 3.3 Arsitektur sistem (Sumber gambar: dokumen pribadi)

Keterangan:

1. PSS = *Panel Sub System*
2. ACSS = *Air Conditioning Sub System*
3. LSS = *Lamp Sub System*

4. *SSS = Socket Sub System*
5. *Internet*
6. *Server*
7. *Web page*
8. *Smartphone app*

I. *Flowchart Sistem*

Tahapan pembuatan suatu sistem diawali dengan membuat algoritma yang menjadi acuan pada pembuatan perangkat keras maupun perangkat lunak, dalam sistem ini terdapat empat algoritma yang bekerja secara bersamaan dan ataupun tidak bersamaan tergantung dari kebutuhan pengguna.

1. *Panel Sub System (PSS)*

PSS berfungsi untuk memonitoring konsumsi energi dan energi terbuang, alat ini bekerja dengan membaca nilai dari sensor arus lalu mengirim data ke *firebase*.

2. *Air Conditioner Sub System (ACSS)*

ACSS berfungsi untuk memonitoring penggunaan *air conditioner* (AC) dengan membaca sensor arus dan sensor suhu kemudian mengirim data ke *firebase*, serta membaca data dari *firebase* untuk mengontrol penggunaan *air conditioner*.

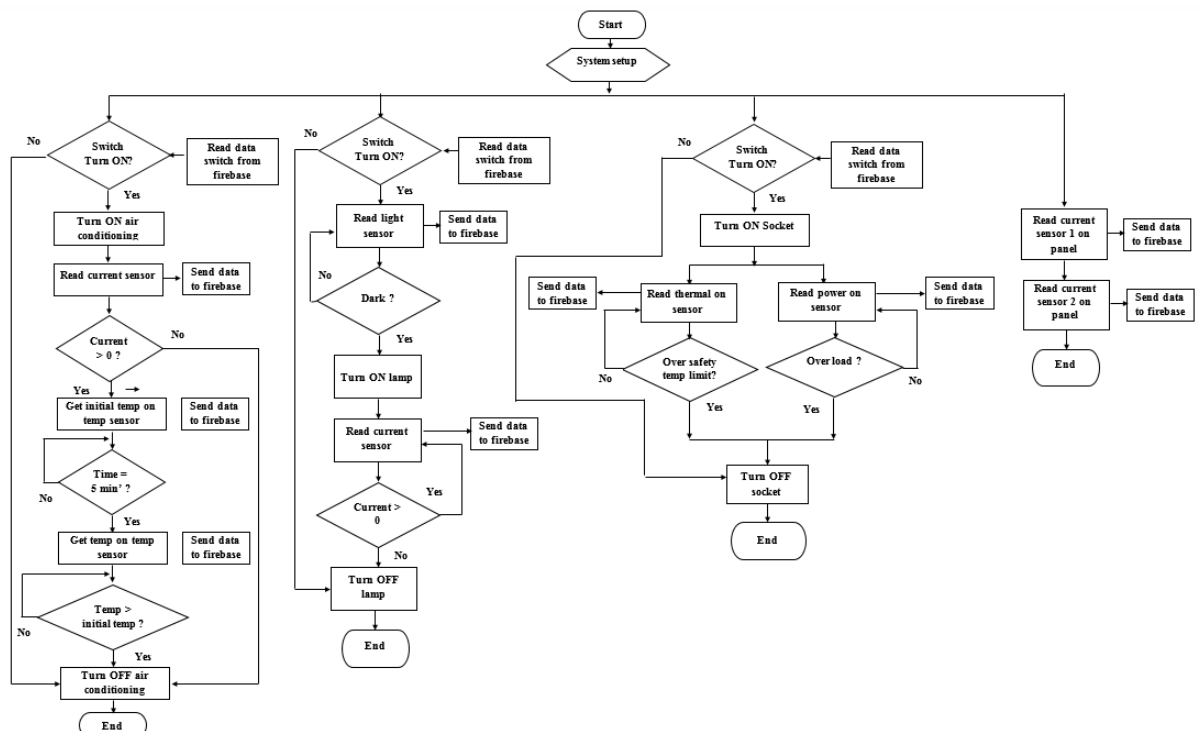
3. *Lamp Sub System (LSS)*

LSS berfungsi untuk memonitoring penggunaan lampu otomatis dengan membaca sensor arus dan sensor cahaya kemudian mengirim data ke *firebase*, serta membaca data dari *firebase* untuk mengontrol penggunaan lampu otomatis maupun manual.

4. *Socket Sub System (SSS)*

SSS berfungsi untuk memonitoring penggunaan stok kontak dengan membaca sensor arus dan sensor suhu kemudian mengirim data ke *firebase*, serta membaca data dari *firebase* untuk mengontrol

penggunaan stok kontak. *Flowchart* atau diagram alir dari sistem ini ditunjukkan pada gambar 3.4.



Gambar 3.4 *Flowchart* sistem. (Sumber gambar: dokumen pribadi)

J. Skema Detail, *Listing* Program, dan Penjelasan Program

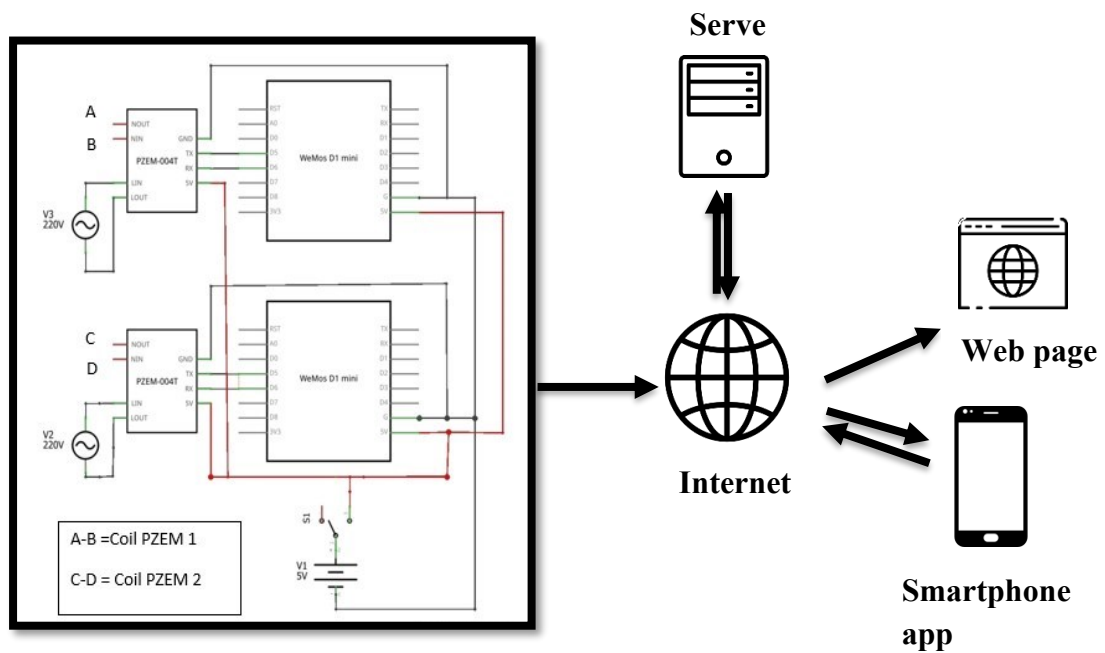
Perangkat keras dari sebuah sistem dibuat berdasarkan skematik yang telah dirancang, di bagian ini akan dijelaskan mengenai komponen utama hingga komponen pendukung yang digunakan. Selain itu juga akan ditunjukkan *listing* program untuk perangkat lunak beserta penjelasannya.

1. Panel Sub Sistem (PSS)

a. Skema detail

Monitoring panel membutuhkan beberapa komponen yaitu dua (2) unit mikrokontroler Node MCU ESP-8266, dua (2) unit sensor PZEM

004T, dan *power supply*. Masing-masing mikrokontroler dan sensor PZEM 004T berfungsi untuk proses monitoring konsumsi energi dan energi terbuang, sedangkan *power supply* berfungsi untuk menyuplai tegangan kedua mikrokontroler dan sensor. Skema rangkaian PSS ditunjukkan pada gambar 3.5.



Gambar 3.5 Skema detail PSS (Sumber gambar: dokumen pribadi)

b. *Listing* dan penjelasan program

```
void loop() {

    statusPZEM_Panel = true;
    currentPanelCons = pzem.current();
    voltagePanelCons = pzem.voltage();
    if (!isnan(currentPanelCons)) {
        Firebase.setFloat(firebaseData, "/Panel/currentCons", currentPanelCons);
        Firebase.setFloat(firebaseData, "/Panel/voltageCons", voltagePanelCons);
        Firebase.setString(firebaseData, "/Panel/warningP", "Normal");
        status = "Normal";
    }
    else {
        Firebase.setString(firebaseData, "/PanelcurrentCons", "-");
        Firebase.setString(firebaseData, "/Panel/voltageCons", "-");
        Firebase.setString(firebaseData, "/Panel/warningP",
"Check_current_sensor_cons");
        statusPZEM_Panel = false;
        status = "Failure";
    }
}
```

```
float powerPanelCons = pzem.power();
float pfPanelCons = pzem.pf();
float energyPanelCons = pzem.energy();
String data = "?x=0" + String(currentPanelCons) + "&current=" +
String(currentPanelCons) + "&voltage=" + String(voltagePanelCons) + "&power=" +
String(powerPanelCons) + "&pf=" + String(pfPanelCons) + "&energy=" +
String(energyPanelCons) + "&status=" + String(status); //Note "?" added at front
host_post += data;
http.begin(host_post); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
httpCode=http.GET();
payload=http.getString(); // get data from webhost continously
Serial.println(payload);
http.end(); //Close connection
```

Saat dinyalakan, sistem membaca nilai sensor arus pada mikrokontroler pertama (sensor *power consumption*) kemudian mengirim ke *database*. Jika terjadi *error* pada pembacaan sensor maka sistem juga akan mengirim data berupa notifikasi.

```

void loop() {
  statusPZEM_Panel = true;
  currentPanelGround = pzem.current();
  voltagePanelGround = pzem.voltage();
  if (!isnan(currentPanelGround)) {
    Firebase.setFloat(firebaseData, "/Panel/currentGround", currentPanelGround);
    //Firebase.setFloat(firebaseData, "/Panel/voltageGround", voltagePanelGround);
    Firebase.setString(firebaseData, "/Panel/warningP", "-");
    status = "Normal";
  }
  else {
    Firebase.setString(firebaseData, "/Panel/currentGround", "-");
    //Firebase.setString(firebaseData, "/Panel/voltageGround", "-");
    Firebase.setString(firebaseData, "/Panel/warningP", "Check current sensor ground");
    statusPZEM_Panel= false;
    status = "Failure";
  }
}

```

```

float powerPanelGround = pzem.power();
float pfPanelGround = pzem.pf();
float energyPanelGround = pzem.energy();
String data = "?current=" + String(currentPanelGround) + "&voltage=" +
String(voltagePanelGround) + "&power=" + String(powerPanelGround) + "&pf=" +
String(pfPanelGround) + "&energy=" + String(energyPanelGround) + "&status=" +
String(status); //Note "?" added at front
host_post += data;
http.begin(host_post); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
httpCode=http.GET();
payload=http.getString(); // get data from webhost continuously
Serial.println(payload);
http.end(); //Close connection

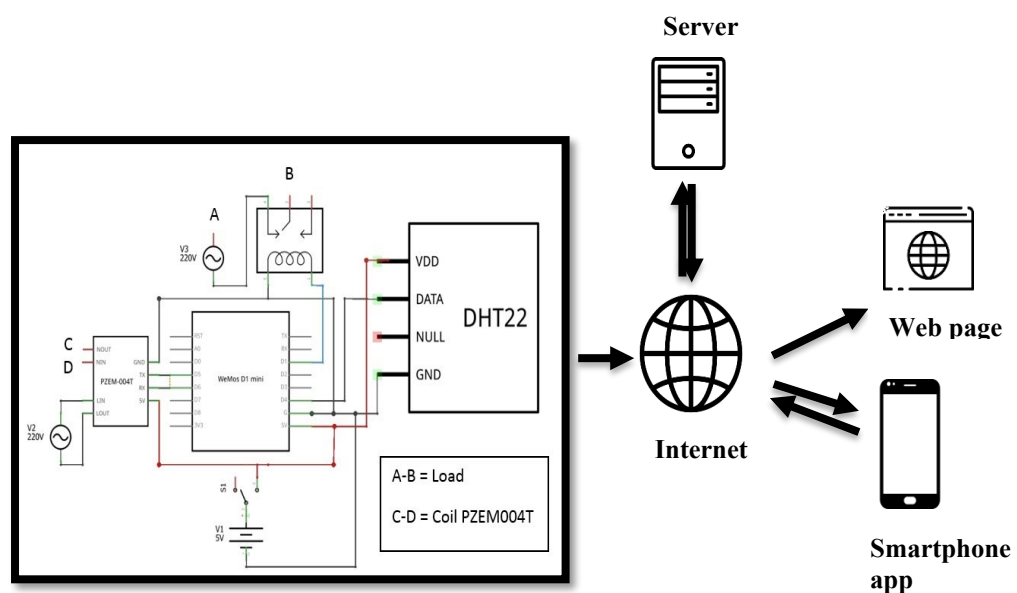
```

Setelah itu, sistem juga membaca nilai sensor arus pada mikrokontroler kedua (sensor *power loss*) kemudian mengirim ke *database*. Jika terjadi *error* pada pembacaan sensor maka sistem juga akan mengirim data berupa notifikasi.

2. AC Sub Sistem (ACSS)

a. Skema detail

Monitoring dan pengontrolan *air conditioner* membutuhkan beberapa komponen yaitu mikrokontroler Node MCU ESP-8266, sensor PZEM 004T, sensor DHT22, *relay*, dan *power supply*. Mikrokontroler, sensor PZEM 004T, *relay*, dan sensor DHT22 berfungsi untuk proses monitoring dan pengontrolan penyejuk ruangan, sedangkan *power supply* berfungsi untuk menyuplai tegangan mikrokontroler, sensor dan *relay*. Skema rangkaian ACSS ditunjukkan pada gambar 3.6.



Gambar 3.6 Skema detail ACSS (Sumber gambar: dokumen pribadi)

b. Listing dan penjelasan program

```

void loop() {
  check_mode:
  if (Firebase.getString(firebaseData, "/AC/switchAc")) {
    if (firebaseData.to<String>() == "1") { goto mode_on;}
    if (firebaseData.to<String>() == "0") {goto mode_off;}
    goto check_mode;
  }

```

Awal dinyalakan, mikrokontroler melakukan proses pengecekan mode (*check mode*) yaitu mengambil data perintah dari *firebase*, jika perintah dari *firebase* (*switch AC*) berlogika 1 (ON), maka mikrokontroler mengaktifkan *air conditioner* dan menjalankan sistem. Dan sebaliknya jika perintah dari *firebase* (*switch AC*) berlogika 0 (OFF), maka mikrokontroler menonaktifkan *air conditioner* dan mematikan sistem.

```

mode_on:
  if (Firebase.getString(firebaseData, "/AC/switchAc")) {
    if (firebaseData.to<String>() == "0") { goto mode_off;}}
  timing2 = millis()-timing1;
  timing = timing2 / 60000;
  Firebase.setInt(firebaseData, "/AC/timeOnAc", timing);
  digitalWrite (relayAC, HIGH);
  temp = dht.readTemperature();
  if (isnan(temp)) { Firebase.setString(firebaseData, "/AC/temperatureAc", "-");
    Firebase.setString(firebaseData, "/AC/warningAc", "Check_temperature_sensor");}
  else { Firebase.setFloat(firebaseData, "/AC/temperatureAc", temp);
    Firebase.setString(firebaseData, "/AC/warningAc", "Temp_Sens_Normal");}

```

```

statusPZEM_AC = true;
currentAC = pzem.current();
voltageAC = pzem.voltage();
if (!isnan(currentAC)) { Firebase.setFloat(firebaseData, "/AC/currentAc", currentAC);
                        Firebase.setFloat(firebaseData, "/AC/voltageAc", voltageAC);
                        Firebase.setString(firebaseData, "/AC/warningAc", "Current_Sens_Normal");}

else { Firebase.setString(firebaseData, "/AC/currentAc", "-");
      Firebase.setString(firebaseData, "/AC/conditionAc", "Not detected");
      Firebase.setString(firebaseData, "/AC/warningAc", "Check_current_sensor");
statusPZEM_AC = false;}

```

```

if (timing >= 1 && currentAC > 0.0)
{ Firebase.setFloat(firebaseData, "/AC/currentAc", currentAC);
  Firebase.setFloat(firebaseData, "/AC/voltageAc", voltageAC);
  Firebase.setString(firebaseData, "/AC/conditionAc", "TURN ON");
  Firebase.setString(firebaseData, "/AC/warningAc", "Normal");}

```

```

if (timing >= 1 && currentAC == 0.0)
{ Firebase.setFloat(firebaseData, "/AC/currentAc", currentAC);
  Firebase.setFloat(firebaseData, "/AC/voltageAc", voltageAC);
  Firebase.setString(firebaseData, "/AC/conditionAc", "Damaged");
  Firebase.setString(firebaseData, "/AC/warningAc", "Fault_detected");}

```

```

float powerPanelCons = pzem.power();
float pfPanelCons = pzem.pf();
float energyPanelCons = pzem.energy();
String data = "?x=" + String(currentAC) + "&current=" + String(currentAC) + "&voltage="
+ String(voltageAC) + "&power=" + String(powerPanelCons) + "&pf=" +
String(pfPanelCons) + "&energy=" + String(energyPanelCons) + "&status=" + String(status)
+ "&toggle=" + String(toggle) + "&temp=" + String(temp); //Note "?" added at front
host_post += data;
http.begin(host_post); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
httpCode=http.GET();
payload=http.getString(); // get data from webhost continously
Serial.println(payload);
http.end(); //Close connection
delay(1000);
}

```

Pada mode ON, sistem membaca nilai sensor suhu dan sensor arus kemudian mengirim ke *database*. Jika terjadi *error* pada pembacaan sensor maka sistem juga akan mengirim data berupa notifikasi.

```

while (timing >= 5 && temp >= roomTemp)
{ digitalWrite (relayAC, LOW);
  Firebase.setFloat(firebaseData, "/AC/currentAc", currentAC);
  Firebase.setFloat(firebaseData, "/AC/voltageAc", voltageAC);
  Firebase.setString(firebaseData, "/AC/conditionAc", "TURN OFF");
  Firebase.setString(firebaseData, "/AC/warningAc", "Fault_detected");}
goto mode_on;

```

Jika sistem *air conditioner* telah menyala selama lebih dari 5 menit namun tidak ada penurunan suhu pada ruangan, maka sistem akan mengirim notifikasi *fault* dan mematikan saklar *air conditioner*.

```

mode_off:
digitalWrite (relayAC, LOW);
if (Firebase.getString(firebaseData, "/AC/switchAc")) {
  if (firebaseData.to<String>() == "1") {
    timing1 = millis();
    timing = timing1 - millis();
    Firebase.setInt(firebaseData, "/AC/timeOnAc", timing);
    goto mode_on;
  }
  Firebase.setString(firebaseData, "/AC/timeOnAc", "-");
  Firebase.setString(firebaseData, "/AC/currentAc", "-");
  Firebase.setString(firebaseData, "/AC/conditionAc", "-");
  Firebase.setString(firebaseData, "/AC/warningAc", "OFF");
  Firebase.setString(firebaseData, "/AC/voltageAc", "-");}
goto mode_off;}

```

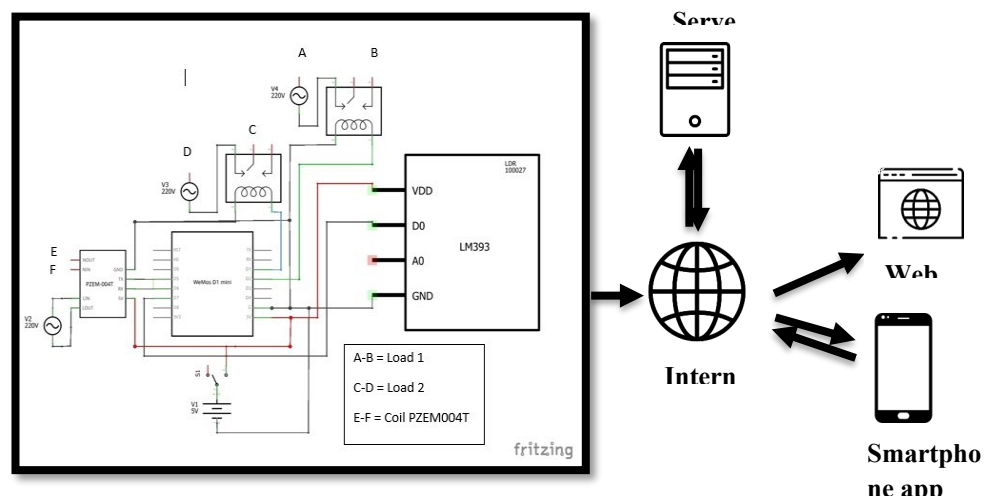
Jika perintah dari *firebase* (*switch AC*) berlogika 0 (OFF), maka mikrokontroler menonaktifkan *air conditioner* dan mematikan sistem, kemudian mengirim data berupa informasi ke *database* bahwa sistem dalam kondisi tidak aktif.

3. Lamp Sub Sistem (LSS)

a. Skema detail

Monitoring dan pengontrolan lampu membutuhkan beberapa komponen yaitu mikrokontroler Node MCU ESP-8266, sensor PZEM 004T, *relay*, sensor LDR, dan *power supply*. Mikrokontroler, sensor

PZEM 004T, *relay*, dan sensor LDR berfungsi untuk proses monitoring dan pengontrolan pencahayaan ruangan dan teras, sedangkan *power supply* berfungsi untuk menyuplai tegangan mikrokontroler, sensor dan *relay*. Skema rangkaian LSS ditunjukkan pada gambar 3.7.



Gambar 3.7 Skema detail LSS (Sumber gambar: dokumen pribadi)

b. Listing dan penjelasan program

```
void loop() {
  check_mode:
  if (Firebase.getString(firebaseData, "/Lamp/switchL")) {
    if (firebaseData.to<String>() == "1") {goto mode_on;}
    if (firebaseData.to<String>() == "0") {goto mode_off;}}
  goto check_mode;
```

Awal dinyalakan, mikrokontroler melakukan proses pengecekan *mode* (*check mode*) yaitu mengambil data perintah dari *firebase*, jika perintah dari *firebase* (*switch Lamp*) berlogika 1 (ON), maka mikrokontroler mengaktifkan lampu dan menjalankan sistem. Dan

sebaliknya jika perintah dari firebase (switch Lamp) berlogika 0 (OFF), maka mikrokontroler menonaktifkan lampu dan mematikan sistem.

```

mode_on:
  if (Firebase.getString(firebaseData, "/Lamp/switchL")) {
    if (firebaseData.to<String>() == "0") {goto mode_off;}}
  if (Firebase.getString(firebaseData, "/Lamp/switchL2")) {
    if (firebaseData.to<String>() == "1") {
      digitalWrite (relayLamp2, LOW) ;}
    if (firebaseData.to<String>() == "0") {
      digitalWrite (relayLamp2, HIGH) ;}}
  timing2 = millis() - timing1;
  timing = timing2 / 60000;
  Firebase.setInt(firebaseData, "/Lamp/timeOnL", timing);
  int value_ldr = digitalRead(lightLDR);
  if (value_ldr == HIGH) {
    digitalWrite (relayLamp, LOW) ;
    Firebase.setString(firebaseData, "/Lamp/sensCond", "DARK");
    Firebase.setString(firebaseData, "/Lamp/lampCond", "TURN ON");
    Firebase.setString(firebaseData, "/Lamp/warningL", "Normal");}
  else {
    digitalWrite (relayLamp, HIGH) ;
    Firebase.setString(firebaseData, "/Lamp/sensCond", "LIGHT");
    Firebase.setString(firebaseData, "/Lamp/lampCond", "TURN OFF");
    Firebase.setString(firebaseData, "/Lamp/warningL", "Normal");}

```

```

statusPZEM_Lamp = true;
currentLamp = pzem.current();
voltageLamp = pzem.voltage();
if (!isnan(currentLamp)) {
  Firebase.setFloat(firebaseData, "/Lamp/currentL", currentLamp);
  Firebase.setFloat(firebaseData, "/Lamp/voltageL", voltageLamp);}
else {
  Firebase.setString(firebaseData, "/Lamp/currentL", "Not detected");
  Firebase.setString(firebaseData, "/Lamp/voltageL", "Not detected");
  statusPZEM_Lamp = false;}

```

```

while ( value_ldr == HIGH && currentLamp == 0) {
  Firebase.setString(firebaseData, "/Lamp/sensCond", "DARK");
  Firebase.setString(firebaseData, "/Lamp/lampCond", "Damaged");
  Firebase.setString(firebaseData, "/Lamp/warningL", "Fault detected");
  statusPZEM_Lamp = true; currentLamp = pzem.current();}
goto mode_on;}

```

```

float powerPanelCons = pzem.power();
float pfPanelCons = pzem.pf();
float energyPanelCons = pzem.energy();
String data = "?x=" + String(currentLamp) + "&current=" + String(currentLamp) +
"&voltage=" + String(voltageLamp) + "&power=" + String(powerPanelCons) + "&pf=" +
String(pfPanelCons) + "&energy=" + String(energyPanelCons) + "&status=" + String(status)
+ "&toggle=" + String(toggle) + "&light=" + String(light); //Note "?" added at front
host_post += data;
http.begin(host_post); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
httpCode=http.GET();
payload=http.getString(); // get data from webhost continously
Serial.println(payload);
http.end(); //Close connection

```

Pada mode ON, sistem membaca nilai sensor cahaya dan sensor arus kemudian mengirim ke *database*. Jika terjadi error pada pembacaan sensor maka sistem juga akan mengirim data berupa notifikasi. Jika kondisi ruangan dalam keadaan gelap maka lampu otomatis dinyalakan, namun jika kondisi ruangan terang maka lampu otomatis dipadamkan.

Untuk lampu manual, jika *switch* lampu manual pada *firebase* berlogika 1 (*switch* lampu manual dalam kondisi ON), maka lampu manual akan dinyalakan. Namun jika *switch* lampu manual berlogika 0, maka lampu manual akan dipadamkan.

```

mode_off:
digitalWrite (relayLamp, HIGH);
if (Firebase.getString(firebaseData, "/Lamp/switchL")) {
  if (firebaseData.to<String>() == "1") { timing1 = millis();
    timing = timing1 - millis();
    Firebase.setInt(firebaseData, "/Lamp/timeOnL", timing);
    goto mode_on;}
  Firebase.setString(firebaseData, "/Lamp/timeOnL", "-");
  Firebase.setString(firebaseData, "/Lamp/sensCond", "-");
  Firebase.setString(firebaseData, "/Lamp/lampCond", "-");
  Firebase.setString(firebaseData, "/Lamp/warningL", "OFF");
  Firebase.setString(firebaseData, "/Lamp/currentL", "-");
  Firebase.setString(firebaseData, "/Lamp/voltageL", "-");}
if (Firebase.getString(firebaseData, "/Lamp/switchL2")) {
  if (firebaseData.to<String>() == "1") {digitalWrite (relayLamp2, LOW) ;}
  if (firebaseData.to<String>() == "0") {digitalWrite (relayLamp2, HIGH) ;}
  goto mode_off;
}

```

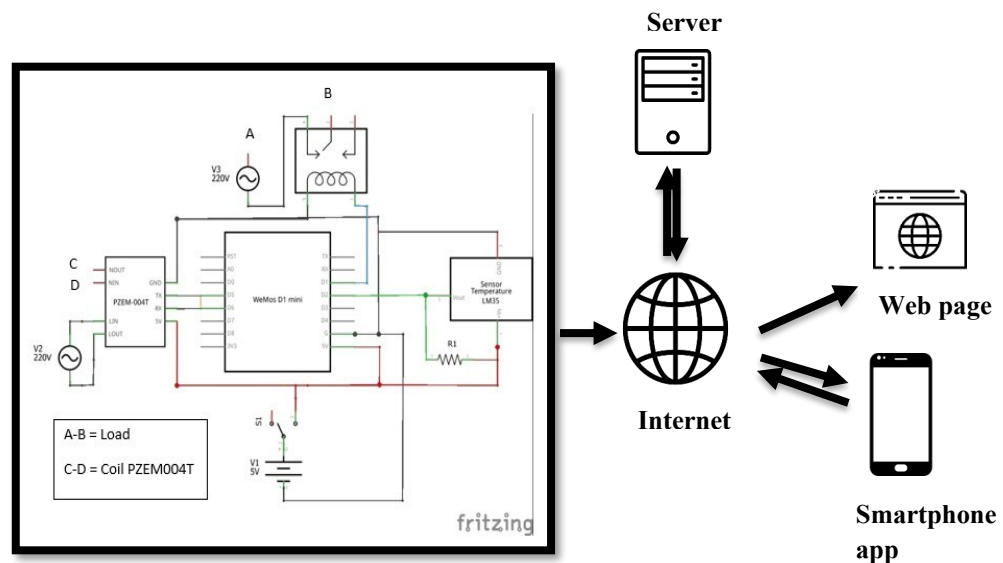
Jika perintah dari *firebase* (*switch lamp*) berlogika 0 (OFF), maka mikrokontroler menonaktifkan lampu dan mematikan sistem, kemudian mengirim data berupa informasi ke *database* bahwa sistem dalam kondisi tidak aktif.

4. Socket Sub Sistem (SSS)

a. Skema detail

Monitoring dan pengontrolan stok kontak membutuhkan beberapa komponen yaitu mikrokontroler Node MCU ESP-8266, sensor PZEM 004T, *relay*, sensor DS18B20, dan *power supply*. Mikrokontroler, sensor PZEM 004T, *relay*, dan sensor DS18B20 berfungsi untuk proses monitoring dan pengontrolan stok kontak, sedangkan *power supply* berfungsi untuk menyuplai tegangan mikrokontroler, sensor dan *relay*.

Skema rangkaian SSS ditunjukkan pada gambar 3.8.



Gambar 3.8 Skema detail SSS (Sumber gambar: dokumen pribadi)

b. *Listing* dan penjelasan program

```
void loop() {
  check_mode:
  if (Firebase.getString(firebaseData, "/SC/switchSC")) {
    if (firebaseData.to<String>() == "1") {goto mode_on;}
    if (firebaseData.to<String>() == "0") {goto mode_off;}}
  goto check_mode;
```

Awal dinyalakan, mikrokontroler melakukan proses pengecekan *mode* (*check mode*) yaitu mengambil data perintah dari *firebase*, jika perintah dari *firebase* (*switch Socket*) berlogika 1 (ON), maka mikrokontroler mengaktifkan *socket* dan menjalankan sistem. Dan sebaliknya jika perintah dari *firebase* (*switch Socket*) berlogika 0 (OFF), maka mikrokontroler menonaktifkan *socket* dan mematikan sistem.

```

mode_on:
  if (Firebase.getString(firebaseData, "/SC/switchSC")) {
    if (firebaseData.to<String>() == "0") {goto mode_off;}}
  timing2 = millis() - timing1;
  timing = timing2 / 60000;
  Firebase.setInt(firebaseData, "/SC/timeOnSC", timing);
  digitalWrite (relaySC, HIGH);
  sensors.requestTemperatures();
  float temperatureC = sensors.getTempCByIndex(0);
  Firebase.setFloat (firebaseData, "/SC/thermalSC", temperatureC);
  Firebase.setString(firebaseData, "/SC/warningSC", "-");

```

```

//Read Current Sensor
statusPZEM_SC = true;
currentSC = pzem.current();
voltageSC = pzem.voltage();
if (!isnan(currentSC)) {
  Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC);
  Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC);}
else {
  Firebase.setString(firebaseData, "/SC/currentSC", "-");
  Firebase.setString(firebaseData, "/SC/voltageSC", "-");
  Firebase.setString(firebaseData, "/SC/conditionSC", "Not detected");
  Firebase.setString(firebaseData, "/SC/warningSC", "Check current sensor");
  statusPZEM_SC = false;}

```

```

powerSC = pzem.power();
if ( !isnan(powerSC) ) {
  Firebase.setFloat(firebaseData, "/SC/powerSC", powerSC);}
else {Firebase.setString(firebaseData, "/SC/powerSC", "-");}
if (powerSC == 0.0)
{ Firebase.setString(firebaseData, "/SC/conditionSC", "No device installed");
  Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC );
  Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC );
  Firebase.setString(firebaseData, "/SC/warningSC", "-");
  Firebase.setFloat(firebaseData, "/SC/powerSC", powerSC );
  jeda = 0;}

```

```

float pfPanelCons = pzem.pf();
float energyPanelCons = pzem.energy();
String data = "?x=" + String(currentSC) + "&current=" + String(currentSC) + "&voltage="
+ String(voltageSC) + "&power=" + String(powerSC) + "&pf=" + String(pfPanelCons) +
"&energy=" + String(energyPanelCons) + "&status=" + String(status) + "&toggle=" +
String(toggle) + "&temp=" + String(temperatureC); //Note "?" added at front
host_post += data;
http.begin(host_post); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
httpCode = http.GET();
payload = http.getString(); // get data from webhost continuously
Serial.println(payload);
http.end(); //Close connection

```

Pada mode ON, sistem membaca nilai sensor panas dan sensor arus kemudian mengirim ke *database*. Jika terjadi *error* pada pembacaan sensor maka sistem juga akan mengirim data berupa notifikasi.

```

if (temperatureC >= 71) {
  while (1) {
    digitalWrite (relaySC, LOW);
    digitalWrite (buzzer, HIGH); delay (100);
    digitalWrite (buzzer, LOW); delay (100);
    digitalWrite (buzzer, HIGH); delay (100);
    digitalWrite (buzzer, LOW); delay (100);
    Firebase.setFloat (firebaseData, "/SC/thermalSC", temperatureC);
    Firebase.setString(firebaseData, "/SC/warningSC", "Overheat");
    Firebase.setString(firebaseData, "/SC/conditionSC", "OFF");
    if (temperatureC < 71) {
      digitalWrite (relaySC, HIGH);break;}}}

```

Jika suhu *socket* mencapai 71 derajat celcius ke atas, maka *relay socket* akan di OFF kan dan *buzzer* akan mengeluarkan bunyi *beep*. Sedangkan jika suhu socket dibawah 71 derajat celcius maka *relay socket* akan di ON kan kembali dan *buzzer* berhenti mengeluarkan bunyi *beep*.

```

if (powerSC >= 1.0 && powerSC <= 400)
{
  Firebase.setString(firebaseData, "/SC/conditionSC", "ON");
  Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC);
  Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC);
  Firebase.setString(firebaseData, "/SC/warningSC", "-");
  Firebase.setFloat(firebaseData, "/SC/powerSC", powerSC);
  jeda = 0;}

```

Jika daya pada *socket* berkisar antara 1 – 400 Watt, maka mikrokontroler akan mengirim nilai daya yang terukur ke *firebase* serta tidak memberikan peringatan, ini menandakan bahwa *socket* dalam kondisi aman (tidak kelebihan beban).

```

if (powerSC > 400 && powerSC <= 420)
{
  Firebase.setString(firebaseData, "/SC/conditionSC", "ON");
  Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC);
  Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC);
  Firebase.setString(firebaseData, "/SC/warningSC", "Almost overload");
  Firebase.setFloat(firebaseData, "/SC/powerSC", powerSC);
  jeda = 0;}

```

Jika daya pada *socket* berkisar antara 401 – 420 Watt, maka mikrokontroler akan mengirim nilai daya yang terukur ke *firebase* serta memberikan peringatan "*Almost overload*", ini menandakan bahwa *socket* dalam kondisi kurang aman (menghampiri kelebihan beban).

```

if (powerSC > 420) {
  digitalWrite (buzzer, HIGH); delay(10);
  digitalWrite (buzzer, LOW); delay(10);
  digitalWrite (buzzer, HIGH); delay(10);
  digitalWrite (buzzer, LOW); delay(10);
  currentmillis = millis ();
  if (currentmillis - startmillis >= 1000) { startmillis = currentmillis; jeda++;
    if (jeda >= 30) {
      while (1) {
        digitalWrite (relaySC, LOW);
        statusPZEM_SC = false;
        currentSC = pzem.current();
        powerSC = pzem.power();
        jeda = 0; } } }
  Firebase.setString(firebaseData, "/SC/conditionSC", "ON");
  Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC);
  Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC);
  Firebase.setString(firebaseData, "/SC/warningSC", "Unplug any load and press reset button");
  Firebase.setFloat(firebaseData, "/SC/powerSC", powerSC); } }

```

Jika daya pada *socket* melebihi 420 Watt, maka *buzzer* akan mengeluarkan bunyi *beep* pendek secara terus menerus hingga 30 detik. Jika beban dilepas dan daya kembali berada di bawah 420 Watt maka bunyi *beep* akan berhenti, namun jika bunyi *beep* pendek diabaikan hingga 31 detik, maka *relay socket* akan di OFF kan dan mikrokontroller akan mengirim data ke *firebase* berupa informasi “*Unplug and press reset button*”. Untuk menyalakan kembali *socket*, maka beberapa beban harus dilepas terlebih dahulu kemudian menekan tombol *reset* supaya sistem *set up* ulang.

```

mode_off:
digitalWrite (relaySC, LOW);
if (Firebase.getString(firebaseData, "/SC/switchSC")) {
  if (firebaseData.to<String>() == "1") {timing1 = millis();
    timing = timing1 - millis();
    Firebase.setInt(firebaseData, "/SC/timeOnSC", timing);
    goto mode_on;}
  //timing = 0;
  Firebase.setString(firebaseData, "/SC/timeOnSC", "-");
  Firebase.setString(firebaseData, "/SC/currentSC", "-");
  Firebase.setString(firebaseData, "/SC/voltageSC", "-");
  Firebase.setString(firebaseData, "/SC/conditionSC", "-");
  Firebase.setString(firebaseData, "/SC/warningSC", "OFF");
  Firebase.setString(firebaseData, "/SC/powerSC", "-");}
goto mode_off;

```

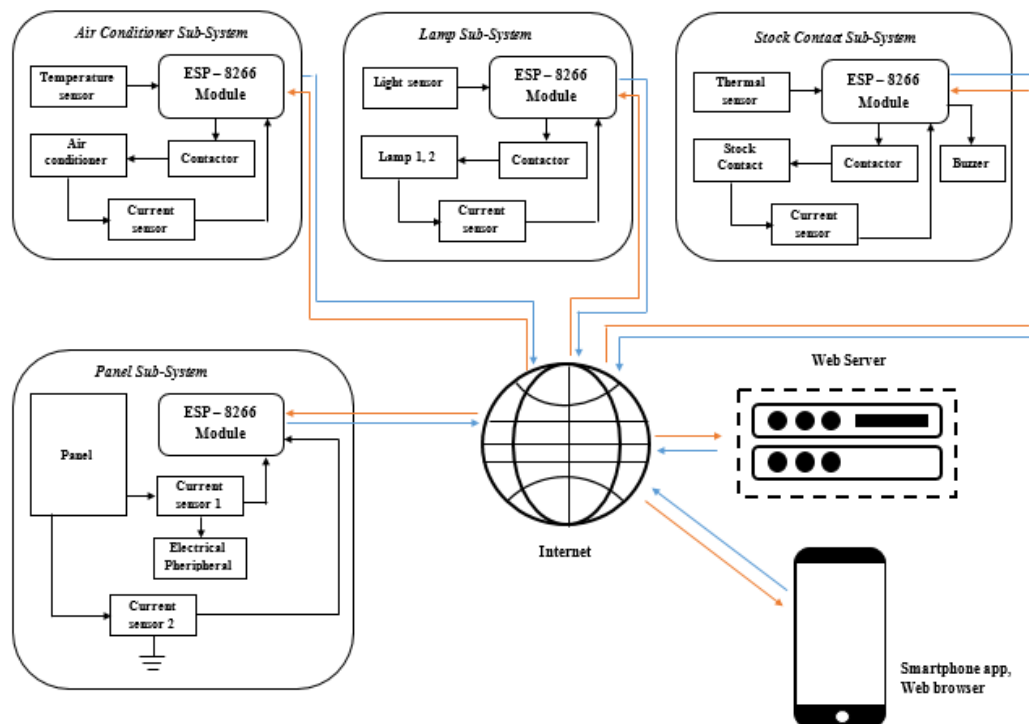
Jika perintah dari *firebase* (*switch Socket*) berlogika 0 (OFF), maka mikrokontroler menonaktifkan *socket* dan mematikan sistem, kemudian mengirim data berupa informasi ke *database* bahwa sistem dalam kondisi tidak aktif.

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

A. Hasil rancangan arsitektur Model MACSys

Hasil rancangan arsitektur model MACSys yang terintegrasi, dinamis, dan *reconfigurable* yang terdiri dari sistem *monitoring*, *controlling*, pendeteksian pengdiagnosaan kesalahan (gangguan) ditunjukkan pada gambar 4.1



Gambar 4.1 Rancangan arsitektur model MACSys

Hasil rancangan model MACSys terdiri dari sistem monitoring dan sistem controlling. Sistem monitoring MACSys untuk memonitoring *power consumption, power loss, voltage* dan *system condition* pada PSS, untuk memonitoring *room lighting, current, voltage*, dan *system condition* pada LSS, untuk memonitoring *room temperature, current, voltage* dan *system condition*, untuk memonitoring *power load, cable thermal, current, voltage*, dan *system condition*.

Sistem pengontrolan MACSys terdiri dari sistem pengontrolan *manual* dan *Automatic*. Pengontrolan manual pada *switch* untuk PSS, ACSS, LSS dan SSS. Pengontrolan *automatic* terdiri dari *Fault Detection and Diagnostic* dan *Automatic FDD*.

B. Hasil Rancangan Sistem MACSys

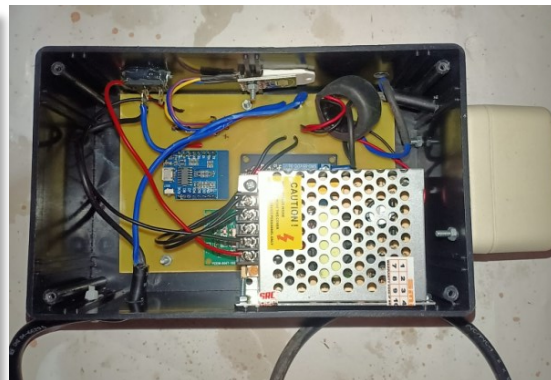
Tahapan-tahapan hasil penelitian dari model yang dikembangkan untuk memonitoring dan pengontrolan perangkat-perangkat listrik.

1. Perangkat Keras

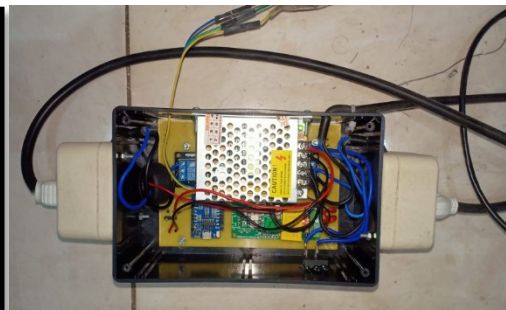
Perangkat keras yang telah dihasilkan terdiri dari empat (4) bagian yaitu PSS, ACSS, LSS, dan SSS. Gambar 4.2 menunjukkan PSS, gambar 4.3 menunjukkan ACSS, gambar 4.4 menunjukkan LSS, dan gambar 4.5 menunjukkan SSS.



Gambar 4.2. Panel Sub System (PSS)
(Sumber gambar: dokumen pribadi)



Gambar 4.3 AC Sub System (ACSS)
(Sumber gambar: dokumen pribadi)



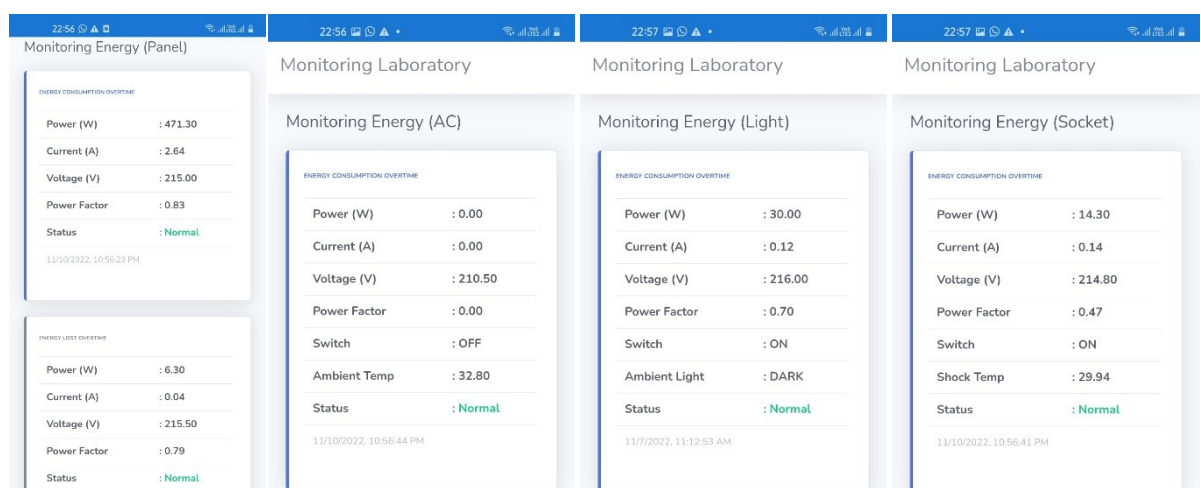
Gambar 4.4 Lamp Sub System (LSS)
(Sumber gambar: dokumen pribadi)



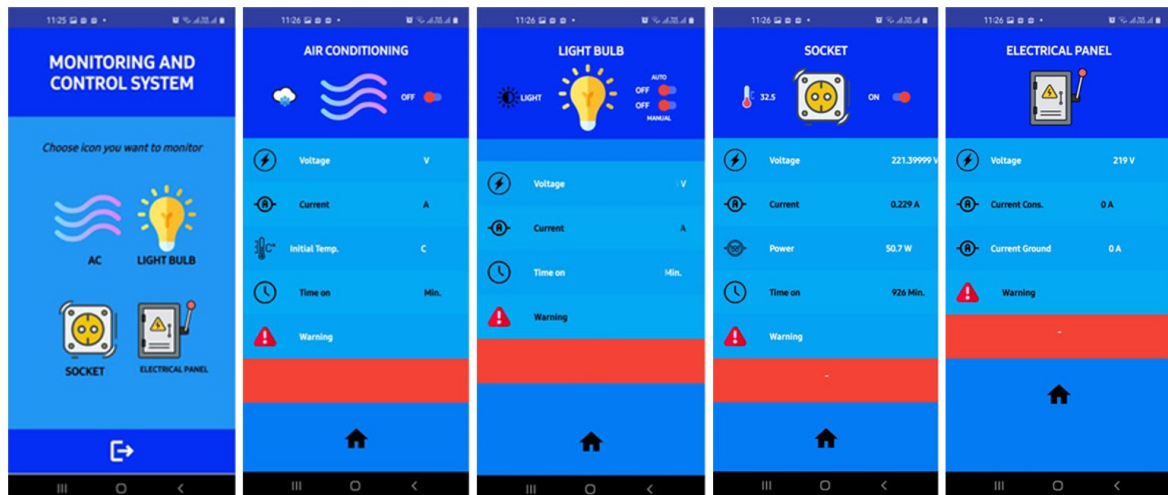
Gambar 4.5 Socket Sub System (SSS)
(Sumber gambar: dokumen pribadi)

2. Perangkat Lunak

Perangkat lunak yang telah dihasilkan terdiri dari dua (2) macam yaitu aplikasi web dan aplikasi android. Aplikasi web ditunjukkan pada gambar 4.6 , dan aplikasi android ditunjukkan pada gambar 4.7.



Gambar 4.6 Tampilan aplikasi MACSys web



Gambar 4.7 Tampilan aplikasi android MACSys

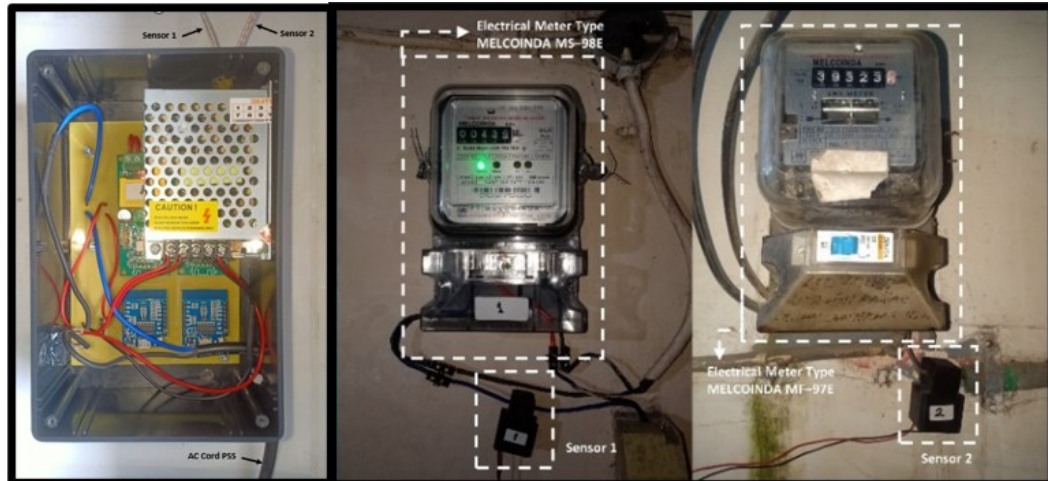
C. Hasil Pengujian MACSys

1. Panel Sub-System

Panel sub system menggunakan dua (2) unit sensor yang masing-masing diletakkan pada jalur MCB untuk menghitung konsumsi energi dan pada jalur *ground* untuk menghitung *lost energy* (energi terbuang). Data dari sensor akan diteruskan ke *Central Unit* untuk dilakukan proses pengolahan data yang selanjutnya akan dikirim ke web server sebagai pusat penyimpanan data sensor. Pengguna dapat mengakses data konsumsi dan *lost* energi melalui website dan *android application* yang dapat diakses melalui internet.

Power sensor yang digunakan yaitu PZEM-004T untuk mengukur tegangan, arus dan *power*. Wemos D1 Mini yang sudah dilengkapi dengan module ESP8266 digunakan sebagai perangkat central unit yang berfungsi untuk menerima dan memproses data dari sensor kemudian meneruskannya ke web server menggunakan komunikasi Wi-Fi karena lebih murah dan

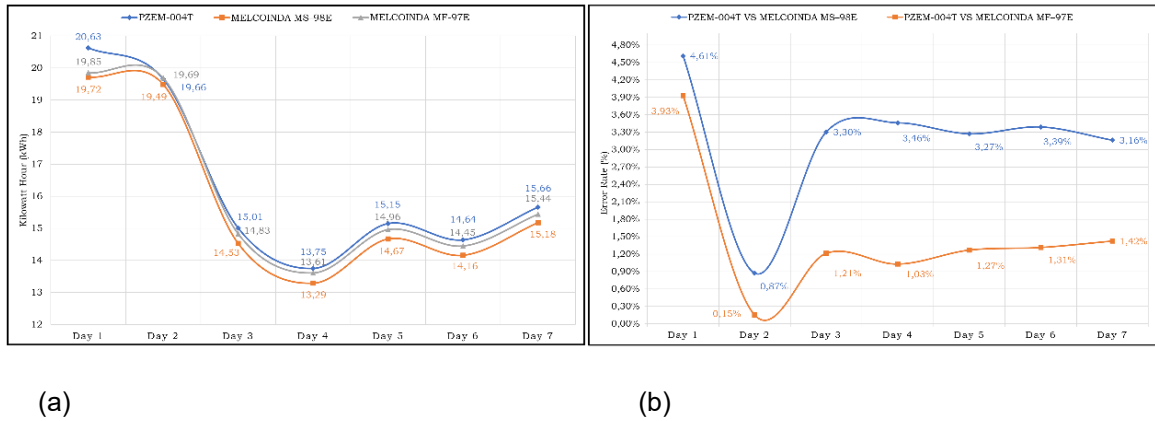
compatible dibandingkan dengan ESP32. Implementasi dari *panel sub sistem* dapat dilihat pada gambar 4.8.



Gambar 4.8 Pemasangan PSS pada KWH Meter

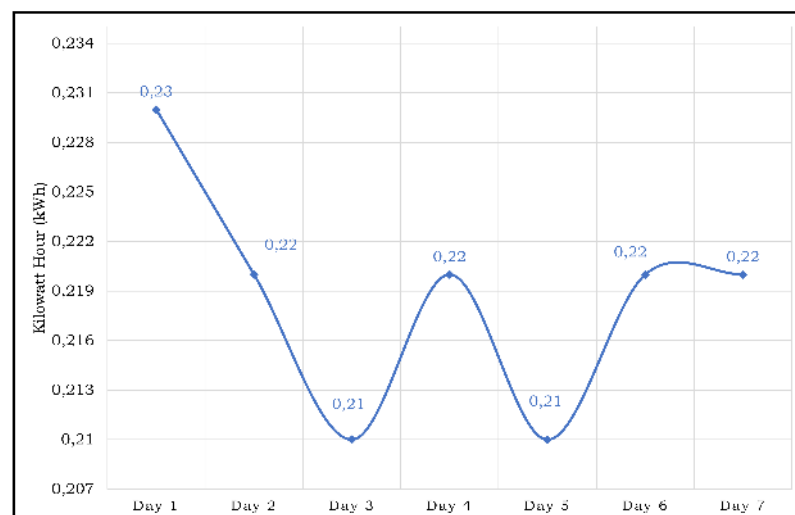
Hasil pengujian monitoring konsumsi yang dilakukan selama 7 hari dapat dilihat pada gambar 4.9. Hasil pengujian menunjukan bahwa jumlah rata-rata konsumsi energi yang digunakan melalui pengukuran PZEM-004T module sensor sebesar 16.39 kWh dan rata-rata konsumsi energi melalui pengukuran *electrical meter* tipe MELCOINDA MS-98E sebesar 15.86 kWh dan tipe MELCOINDA MF-97E sebesar 16.12 kWh. Nilai rata-rata akurasi berdasarkan persentase *error rate* PZEM-004T module sensor terhadap *electrical meter* tipe MELCOINDA MS-98E sebesar 3.15% dan untuk tipe MELCOINDA MF-97E sebagai acuan validasi terhadap PZEM-004T module sensor mendapatkan nilai rata-rata *error rate* lebih kecil yaitu sebesar 1.48%. Untuk lebih jelasnya dapat dilihat pada Gambar 4.9. pada bagian (a) yang menampilkan nilai konsumsi energi selama tujuh hari dari tiga alat pengukuran yang digunakan dan bagian (b) terkait nilai presentase error rate terhadap

pengukuran PZEM-004T module sensor dengan MELCOINDA MS-98E dan PZEM-004T module sensor dengan MELCOINDA MF-97E.



Gambar 4.9. (a) Grafik Konsumsi Energi menggunakan Tiga Alat Ukur dengan nilai rata-rata 16,11 kWh dalam tujuh hari (b) Grafik Persentase *Error Rate* dengan nilai rata-rata 2,31%.

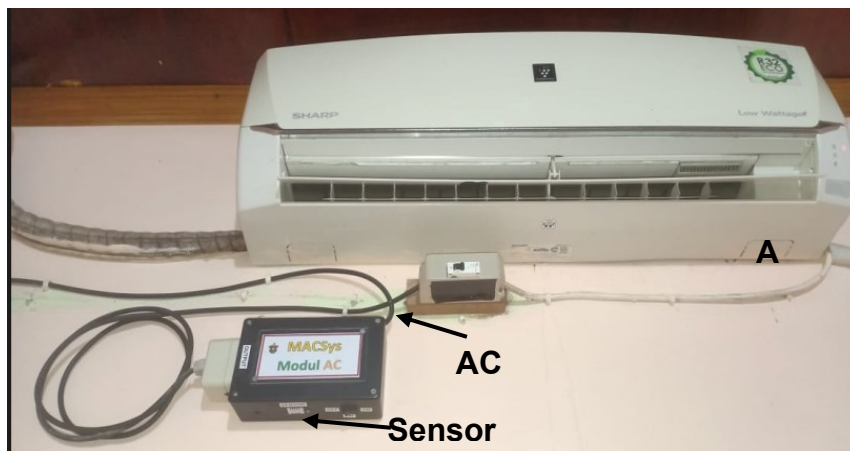
Hasil pengujian dari pengukuran nilai energi listrik yang terbuang selama tujuh hari dengan menggunakan PZEM-004T module sensor, didapatkan nilai rata-rata sebesar 0.22 kWh. Gambar 4.10 menampilkan nilai *lost energy* pada setiap harinya.



Gambar 4.10. Grafik konsumsi energi yang terbuang menggunakan sensor modul PZEM-004T

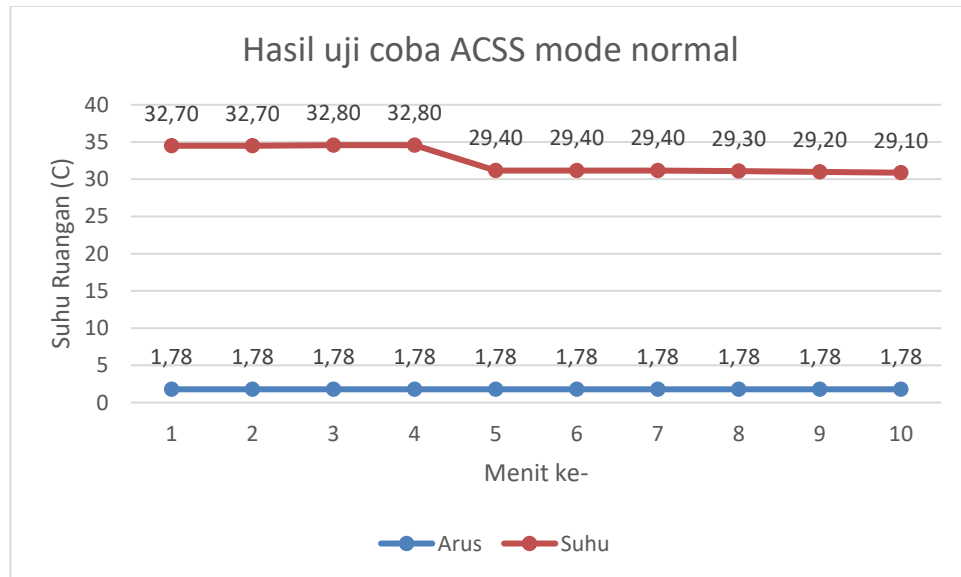
2. Air Conditioner Sub-System

Perangkat keras yang digunakan pada *Air Conditioner Sub-System* yaitu mikrokontroler Wemos D1 Mini, sensor PZEM-004T, sensor suhu DHT22, *power supply* 5V/2A dilengkapi dengan kabel AC, relay 1 *channel* opto 5V, *switch on/off* dan stop kontak. Pemasangan modul ACSS ditunjukkan pada gambar 4.11.



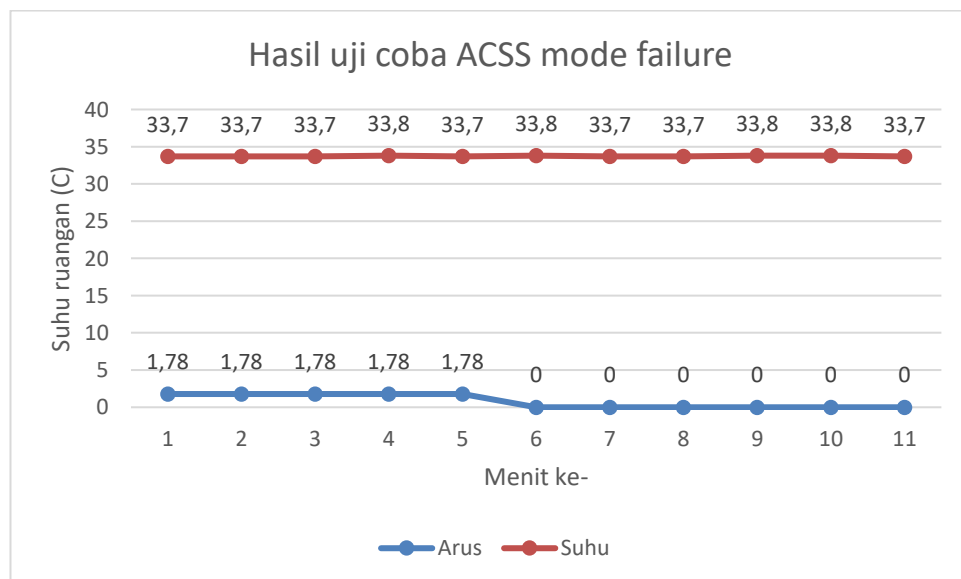
Gambar 4.11. Pemasangan ACSS pada *Air Conditioner*

Hasil monitoring ACSS mode normal ditunjukkan pada gambar 4.12. Pada saat ACSS dalam kondisi ON dan sensor DHT22 membaca suhu ruangan sekitar $32,70^{\circ}\text{C}$ dan Arus AC sebesar 1,78A, setelah menit ke-5 suhu ruangan terbaca $29,40^{\circ}\text{C}$ sehingga arus AC masih tetap terbaca dan AC dalam kondisi ON.



Gambar 4.12. Grafik hasil monitoring ACSS mode normal

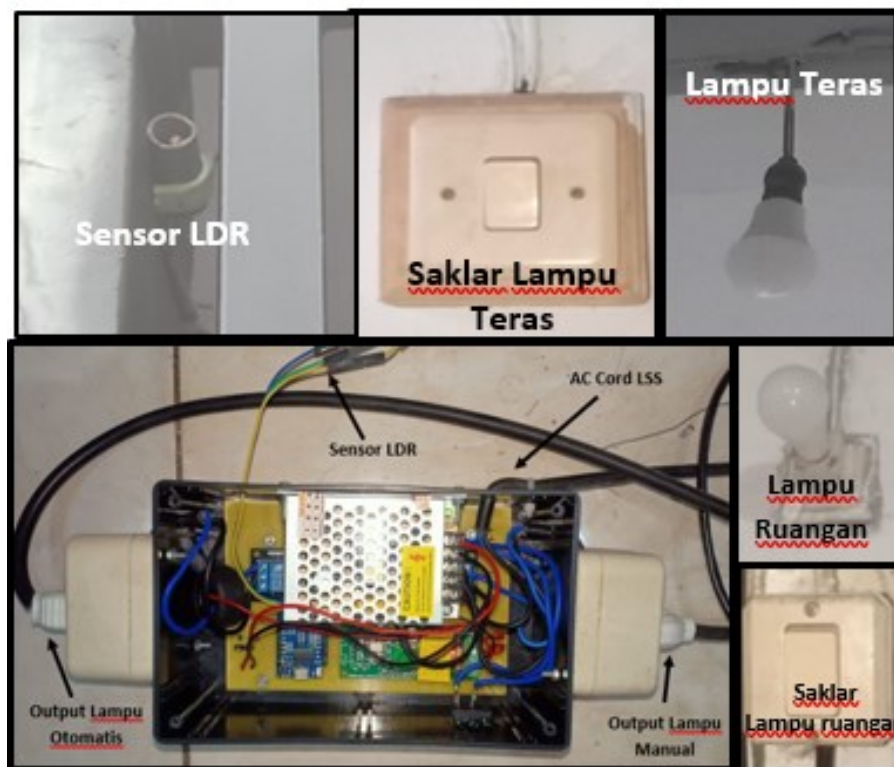
Hasil monitoring ACSS mode failure ditunjukkan pada gambar 4.13. Pada saat ACSS dalam kondisi ON dan sensor DHT22 membaca suhu ruangan sekitar $33,70^{\circ}\text{C}$ dan Arus AC sebesar 1,78A, setelah menit ke-5 suhu ruangan terbaca $33,70^{\circ}\text{C}$ sehingga arus AC terbaca 0A dan AC dalam kondisi OFF.



Gambar 4.13. Grafik Hasil monitoring ACSS mode failure

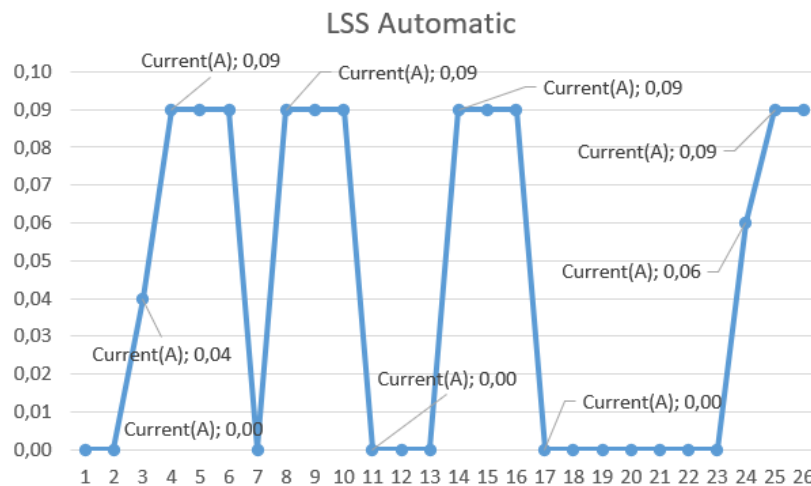
3. Lamp Sub System

Perangkat keras yang digunakan pada LSS yaitu Wemos D1 Mini, sensor PZEM-004T, sensor cahaya LDR, power supply 5V/2A dilengkapi dengan kabel AC, Relay 1 channel opto 5V dua buah, *switch on/off* dan stop kontak dua buah. Pemasangan modul LSS ditunjukkan pada gambar 4.14



Gambar 4.14. Pemasangan modul LSS

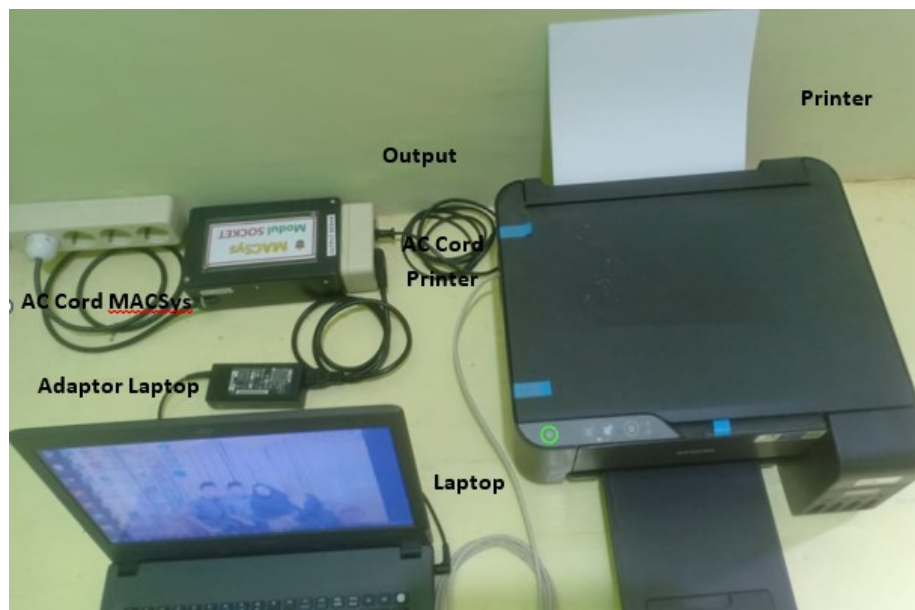
Hasil monitoring LSS pada saat sensor cahaya dalam kondisi terang menunjukkan lampu akan padam sehingga arus yang mengalir ke lampu menjadi 0 A. Hasil monitoring LSS pada saat sensor cahaya dalam kondisi gelap menyebabkan lampu akan menyala sehingga ada arus yang mengalir ke lampu. Hasil monitoring LSS pada kondisi sensor gelap atau terang ditunjukkan pada gambar 4.15



Gambar 4.15. Hasil monitoring LSS pada kondisi sensor gelap atau terang

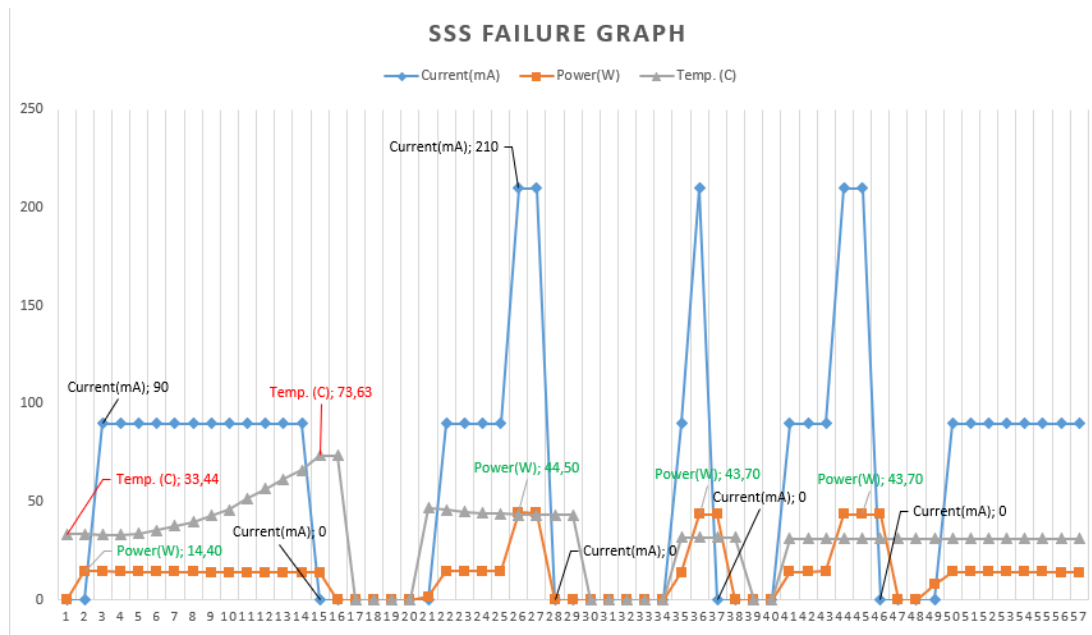
4. Socket Sub System

Perangkat keras yang digunakan pada SSS yaitu Node MCU ESP-8266, sensor PZEM-004T, *relay* sensor DS18B20 dan *power supply* 5V/2A dilengkapi dengan kabel AC, Relay 1 channel opto 5V dua buah, *switch on/off* dan stop kontak satu buah. Pemasangan modul SSS ditunjukkan pada gambar 4.16.



Gambar 4.16. Pemasangan modul SSS

Pada saat modul SSS dalam kondisi ON dan sudah terhubung ke beban maka sensor PZEM-004T akan mendeteksi besar arus, tegangan dan daya listrik yang ada pada stok kontak dan sensor DS18B20 akan mendeteksi suhu kabel. Jika ada arus, tegangan dan daya tidak melebihi 44W maka SSS tetap dalam kondisi ON, begitu juga jika suhu kabel tidak melebihi 70°C maka SSS tetap dalam kondisi ON. Salah satu kesalahan yang terdeteksi antara beban berlebihan atau kelebihan panas maka SSS akan berubah menjadi kondisi OFF. Notifikasi selalu ditampilkan untuk menginformasikan kondisi SSS. Hasil monitoring SSS dalam kondisi beban dan suhu normal, kelebihan beban (*overload*) dan kelebihan panas (*overheat*) ditunjukkan pada gambar 4.17. Sistem diujicoba dengan memanaskan sensor suhu menggunakan solder dan pengaturan beban lebih pada daya 440 watt. Suhu awal pada saat SSS diujicoba adalah 33,4°C, beban sebesar 14,40 watt dan arus sebesar 90 mA dan sistem bekerja dalam kondisi normal. Pada saat sensor mendeteksi suhu sebesar 73°C maka arus berubah menjadi 0 *ampere* dan daya menjadi 0 watt, hal ini disebabkan karena sistem memutuskan arus yang mengalir ke output stok kontak SSS dan sistem terdeteksi *overheat*. Pada saat sensor PZEM-004T mendeteksi daya 44,50 dan arus sebesar 210 mA, maka sistem akan memutus arus yang mengalir ke output stok kontak SSS sehingga arus berubah menjadi 0 ampere dan sistem terdeteksi *overload*.



Gambar 4.17. Hasil monitoring SSS dalam kondisi normal, *overload* dan *overheat*

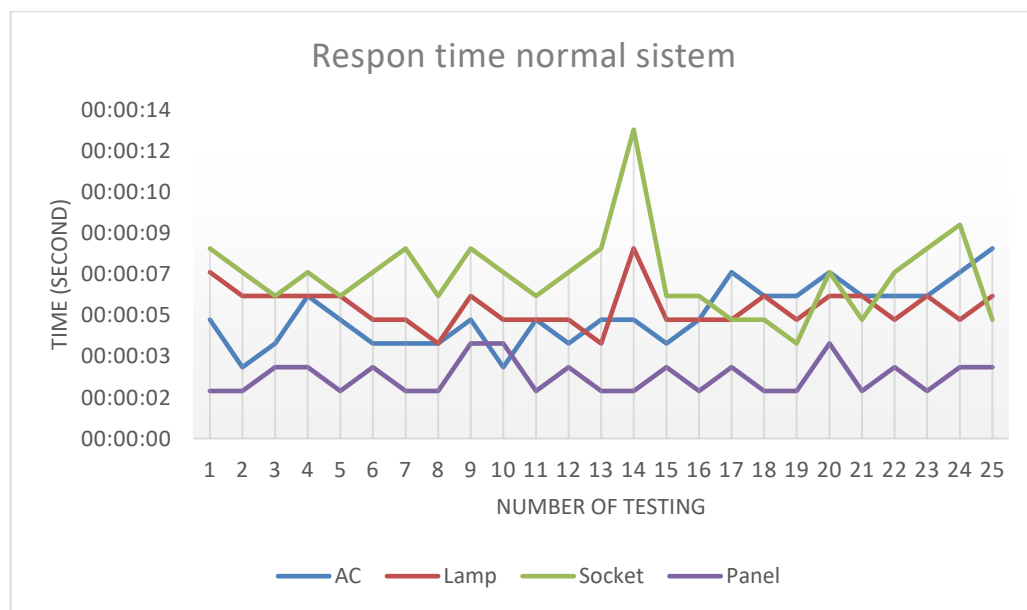
D. Hasil Pengujian *Blackbox MACSys*

Prinsip kerja MACSys yang terdiri dari 4 modul yaitu *panel sub system*, *air conditioner sub system*, *lamp sub system*, dan *socket sub system* yang telah diuji sebanyak 20 kali, hasil pengujian ditunjukkan pada lampiran B.5. Pengujian dilakukan dengan mengetes fungsi dari masing-masing modul yang digunakan dengan Teknik pengujian blackbox. Pengujian blackbox juga dilakukan untuk aplikasi MACSys dengan menguji fungsi aplikasi sebanyak 20 kali pengujian. Hasil pengujian blackbox untuk aplikasi MACSys ditunjukkan pada lampiran B.6.

E. Hasil Pengujian Respon *Time MACSys*

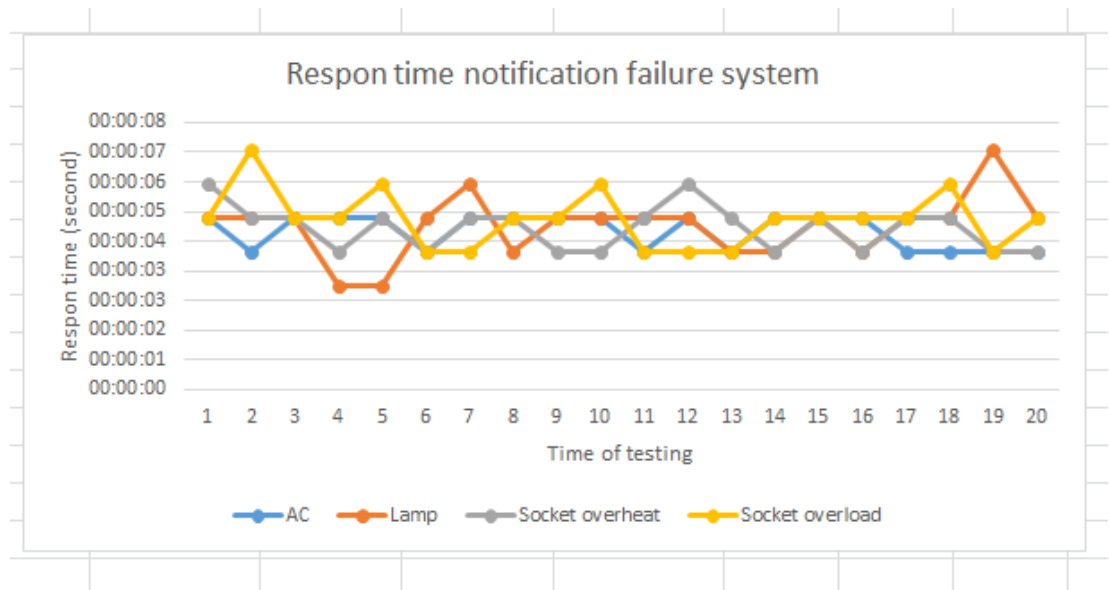
Pengujian respon *time* telah dilakukan dan diperoleh hasil rata-rata respon *time* tercepat pada modul *panel sub system* sebesar 2,6 detik, *air conditioner sub system* sebesar 5,2 detik, *lamp sub system* sebesar 5,52 detik

dan modul *electrichal socket sub-system* sebesar 6,84 detik. Respon *time* sangat dipengaruhi oleh *traffic data* dan kekuatan sinyal seluler yang digunakan pada saat pengujian. Telah dilakukan pengujian sebanyak 25 kali pengiriman instruksi control secara manual dan mendapatkan hasil *packet loss* adalah 0% yang berarti semua data berhasil dikirim dan untuk hasil pengujian respon *time* sistem dalam kondisi normal. Grafik respon *time* saat sistem normal ditunjukkan pada gambar 4.18.

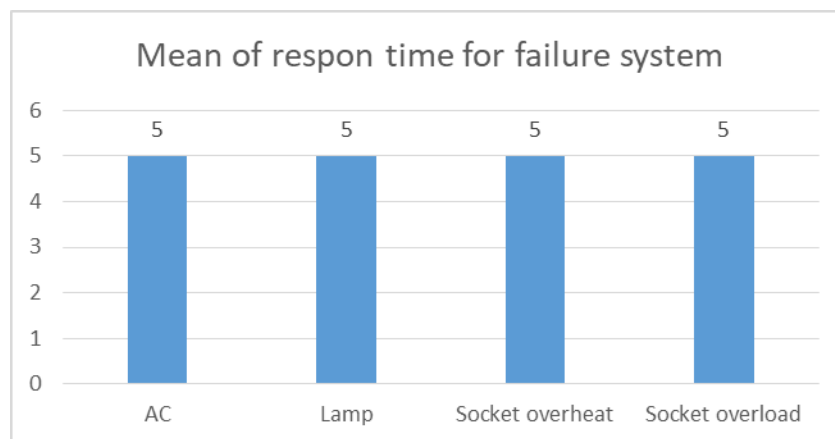


Gambar 4.18. Respon *time* MACSys dalam kondisi normal

Pendeteksi kesalahan secara otomatis juga dapat bekerja pada modul *air conditioner sub-system*, *lamp sub-system* dan *electrichal socket*. Pemberitahuan hasil pendeteksian kesalahan dapat dilihat pada aplikasi android MACSys. Respon *time* pada saat mendeteksi kesalahan yang dilakukan sebanyak 20 kali dan ditunjukkan pada gambar 4.19. Rerata respon *time* pada saat mendeteksi kesalahan (gangguan) sebesar 5 detik, nilai respon *time* ini masih berada dalam kategori cepat. Respon *time* MACSys dalam kondisi deteksi kesalahan (gangguan) ditunjukkan pada gambar 4.20.



Gambar 4.19. Respon *time* MACSys dalam kondisi deteksi kesalahan



Gambar 4.20. Respon *time* MACSys dalam kondisi deteksi kesalahan

Selain pengujian respon *time*, pengujian menggunakan *firebase test lab* dilakukan pada aplikasi Android MACSys. Pengujian dilakukan dengan cara: Aplikasi MACSys diuji melalui *test lab firebase* menggunakan *device virtual* dengan spesifikasi **pixel 5e** dan **API level 30**, orientasi **portrait** dan dinyatakan **lulus/berhasil** uji. Hasil *test* menunjukkan bahwa *time to initial display* adalah 614ms, *graphics stats* dengan *missed Vsync* 11%, *high input*

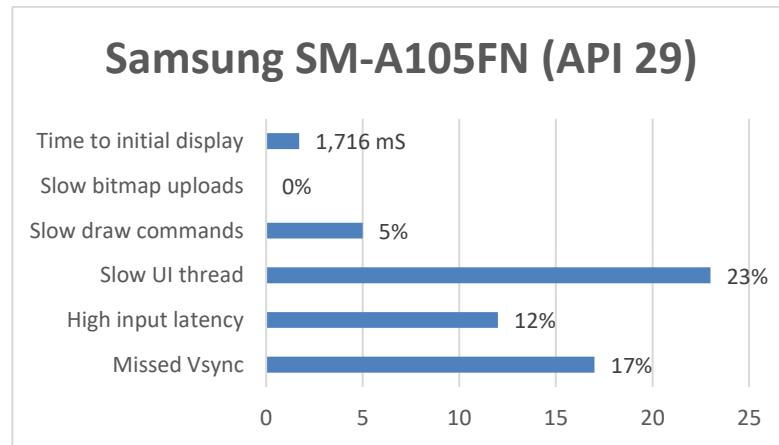
latency 8%, slow UI thread 21%, slow draw commands 4%, slow bitmap uploads 0%.

Hasil pengujian sistem dengan google pixel 5e dan level API 30 dengan waktu pengujian selama 5 menit ditunjukkan pada gambar 4.21. dengan pengujian CPU, pengujian *memory* dan pengujian *network*

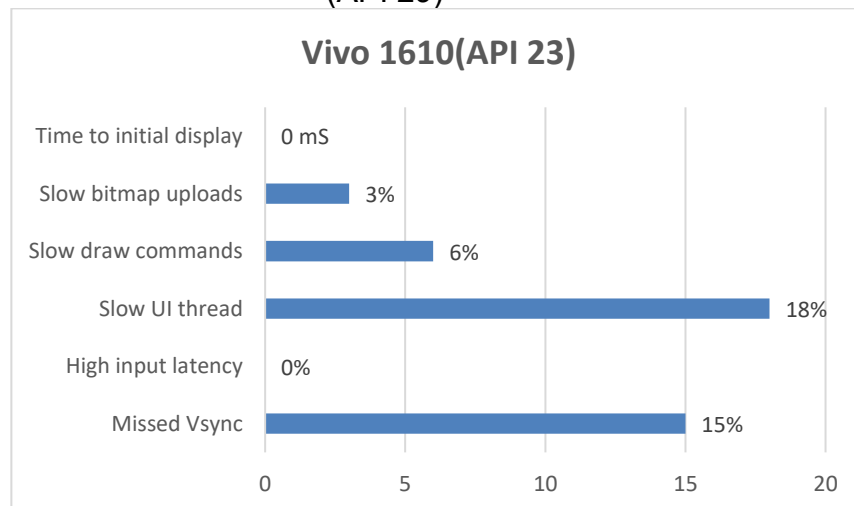


Gambar 4.21. Hasil pengujian *system*

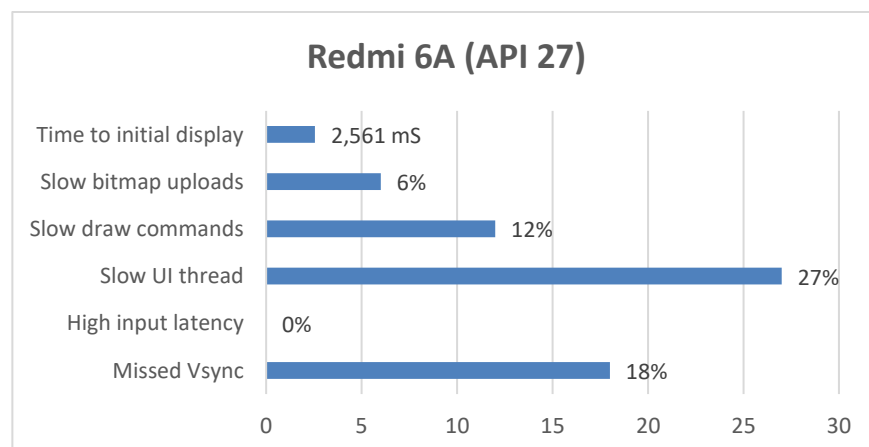
Pengujian *performance* aplikasi android MACSys untuk Samsung SM-A105FN (API 29) dijabarkan pada gambar 4.22, Vivo 1610 (API 23) dijabarkan pada gambar 4.23, dan Redmi 6A (API27) dijabarkan pada gambar 4.24. Pengujian *performance* aplikasi terkait dengan *Missed Vsync*, *high input latency*, *slow UI thread*, *slow draw commands*, *slow bitmap uploads* dan *time to initial display*.



Gambar 4.22. Hasil pengujian *performance* Samsung SM-A105FN (API 29)



Gambar 4.23. Hasil pengujian *performance* Vivo 1610 (API 23)



Gambar 4.24. Hasil pengujian *performance* Redmi 6A (API 27)

Hasil pengujian performance dari tiga perangkat yang berbeda menunjukkan bahwa waktu inisialisasi tampilan antara 0 ms s.d 2,51, *slow bitmap uploads* terhitung antara 0% s.d. 6%, *slow draw command* terhitung antara 5% s.d. 12%, *slow UI (User Interface) Tread* terhitung antara 18% s.d. 27%, *high input latency* tidak ditemukan pada 2 perangkat dan satu perangkat menunjukkan 12%, *missed vertical synchronization* antara 15% s.d. 18%.

BAB V

PENUTUP

A. Kesimpulan

Sistem monitoring konsumsi *energy* dan *lost energy* menggunakan sensor PZEM-004T berbasis *Internet of Things* berhasil dibangun. Didapatkan hasil pengujian akurasi sensor dengan rata-rata *error rate* sebesar 2.31% dari hasil validasi pembacaan dua unit *electrical meter* tipe MELCOINDA MF-97E dan MELCOINDA MS-98E. Hal ini menandakan bahwa sistem monitoring ini memiliki akurasi kebenaran dalam mendeteksi konsumsi energi sebesar 97,69%. Untuk energi listrik yang terbuang, EMS mendeteksi nilai kWh rata-rata selama tujuh hari sebesar 0.22 kWh. Sistem ini sudah dapat diimplementasikan secara nyata sebagai alat monitoring konsumsi dan *lost* energi listrik yang fleksibel.

MACSys memiliki sistem pendeteksi kesalahan yang dapat terjadi pada perangkat kelistrikan yang terpasang seperti *air conditioner*, lampu dan *electrical socket*. Pendeteksi kesalahan pada MACSys berhasil dibangun dan hasil pengujian prinsip kerja sebanyak 10 kali dan berhasil beroperasi sesuai dengan prinsip kerja yang telah dirancang sehingga pengguna dapat mengetahui kesalahan yang terjadi pada perangkat kelistrikan dan bagian apa yang perlu ditangani.

Aplikasi android MACSys dapat memonitoring dan mengontrol modul-modul yang dibangun dan diimplementasikan dimana saja posisinya dan kapanpun akan digunakan dengan syarat terkoneksi jaringan internet. Aplikasi android MACSys sangat tepat digunakan untuk memonitoring dan mengontrol peralatan kelistrikan yang berbeda-beda dan jumlah banyak

Uji *quality of service* dari sisi respon *time* telah dilakukan sebanyak 25 kali di masing-masing sistem dan mendapatkan rata-rata *respon time* terbesar pada modul *socket sub system* 6,84 detik, modul *lamp sub-system* 5,52 detik *air conditioner sub-system* 5,2 detik dan modul panel *sub-system* 2,6 detik dengan rata-rata respon *time* keseluruhan adalah 5,04 detik. Aplikasi MACSys diuji melalui *test lab firebase* menggunakan *device virtual* dengan spesifikasi

pixel 5e dan **API level 30**, orientasi **portrait** dan dinyatakan **lulus/berhasil** uji.

B. Saran

Model surveillance kelistrikan berbasis IoT bekerja secara otomatis dan terkonfigurasi secara mudah dalam jaringan dan terintegrasi telah memperlihatkan prospek pengaplikasian untuk skala menengah hingga skala industri (skala besar). Karena realibiliti, efisiensi, akurasi dalam monitoring konsumsi energy dan los energy. Respon time (waktu yang digunakan untuk pengiriman data ke server) pada saat sistem bekerja dengan normal dan pada saat terjadi gangguana pada sistem. Kondisi animo dan kebutuhan masyarakat dan industry akan sangat besar sehingga keberlanjutan kegiatan R & D pada kedua aspek Perangkat keras:

Pemutakhiran kinerja operasi 4 modul IoT terkait dengan komponen mikrokontroler dan sensor yang digunakan

Perangkat lunak

Aplikasi android yang telah didesain dan diimplementasikan memiliki kekurangan diantaranya hanya sebagian simbol yang bersifat aktif dan symbol yang lain hanya bersifat informasi.

Peningkatan penggunaan TKDN (dan TRL (Technology Readiness Level) tingkat kesiapterapan teknologi saat ini masih berada pada level 7 hingga 8. Sedangkan TKDN dari produk sistem surveillance perangkat-perangkat kelistrikan IoT sekitar 70%. Oleh karena itu upaya riset berkelanjutan harus diimplementasikan untuk meningkatkan TKDL dan TRL.

DAFTAR PUSTAKA

- Abid, A. J. (2017). *Internet of Energy : A Design to Manage Energy Consumption for Limited Resources Building*. June, 19–25.
- Agdas, D., Srinivasan, R. S., Frost, K., & Masters, F. J. (2015). Energy use assessment of educational buildings : Toward a campus-wide sustainable energy policy. *Sustainable Cities and Society*, 17, 15–21. <https://doi.org/10.1016/j.scs.2015.03.001>
- Akbar, Y. M. (2014). *Sistem Pemantau Jaringan Wireless dan Estimasi Arah Wireless Access Point berbasis Raspberry-Pi*. 16(1), 1–9.
- Alanne, K. (2015). Ac ce pt cr t. *Energy & Buildings*. <https://doi.org/10.1016/j.enbuild.2015.07.060>
- Ascione, F., Bianco, N., Francesca, R., Masi, D., Maria, G., & Peter, G. (2017). Energy retrofit of educational buildings : Transient energy simulations , model calibration and multi-objective optimization towards nearly zero-energy performance. *Energy & Buildings*, 144, 303–319. <https://doi.org/10.1016/j.enbuild.2017.03.056>
- Attia, S., Eleftheriou, P., Xeni, F., Morlot, R., Ménézo, C., Kostopoulos, V., Betsi, M., Kalaitzoglou, I., Pagliano, L., Cellura, M., Almeida, M., Ferreira, M., Baracu, T., Badescu, V., Crutescu, R., & Hidalgo-betanzos, J. M. (2019). Overview and future challenges of nearly zero energy buildings (nZEB) design in Southern Europe. *Energy & Buildings*, 155(2017), 439–458. <https://doi.org/10.1016/j.enbuild.2017.09.043>
- Azmi, R. (2013). *Analisis Migrasi Radio Trunking Analog ke Radio Trunking Digital di Indonesia*. 247–264.
- Cao, X., Dai, X., & Liu, J. (2016). Building energy-consumption status worldwide and the state-of-the-art technologies for zero-energy buildings during the past decade. *Energy & Buildings*, 128, 198–213. <https://doi.org/10.1016/j.enbuild.2016.06.089>
- Darmawanputra, S. F., Rachmat, H., & Sjafrizal, T. (2015). Perancangan Sistem Otomasi Terintegrasi dan Supervisory Control And Data Acqusion (SCADA) Pada Stasiun Kerja Ex-Turning, Drilling-Chamfering dan Tthreading di PT. ABC Menggunakan Jaringan Komunikasi Wireless. *E-Proceeding of Engineering*, 4502–4528.
- Ferrari, S. (2017). Ac ce pt cr t. *Sustainable Cities and Society*. <https://doi.org/10.1016/j.scs.2017.03.010>
- Ferreira, M., Almeida, M., & Rodrigues, A. (2017). Impact of co-benefits on

the assessment of energy related building renovation with a nearly-zero energy target. *Energy & Buildings*.

<https://doi.org/10.1016/j.enbuild.2017.07.066>

Firman, B., & Firmansyah, E. (2012). Implementasi Komunikasi Data Berbasis Zigbee Pada SCADA (Supervisory Control and Data Acquisition) PLTMH. *Jurnal Teknologi*, 5, 149–155.

Guilherme Mussi., Tochi., L. B. Campos., C. E. Cugnasca. (2017). Home Automation Networks. A Survey.

Hannan, M. A., Faisal, M., Ker, P. J., Mun, L. H., Parvin, K., Mahlia, T. M. I., & Blaabjerg, F. (2018). *A Review of Internet of Energy Based Building Energy Management Systems : Issues and Recommendations*. 3536(c). <https://doi.org/10.1109/ACCESS.2018.2852811>

Harkouss, F., Fardoun, F., & Biwole, P. H. (2018). Multi-objective optimization methodology for net zero energy buildings. *Journal of Building Engineering*, 16(December 2017), 57–71. <https://doi.org/10.1016/j.jobbe.2017.12.003>

Huang, Z., Lu, Y., Wei, M., & Liu, J. (2017). Performance Analysis of Optimal Designed Hybrid Energy Systems for Grid-connected Nearly/Net Zero Energy Buildings. *Energy*. <https://doi.org/10.1016/j.energy.2017.11.093>

Jain, M., Hoppe, T., & Bressers, H. (2016). Analyzing sectoral niche formation : The case of net-zero energy buildings in India. *Environmental Innovation and Societal Transitions*, 1–17. <https://doi.org/10.1016/j.eist.2016.11.004>

Khajenasiri, I., Virgone, J., & Gielen, G. (2015). *A Presence-Based Control Strategy Solution for HVAC Systems*. 620–622.

Kim, J., Cai, J., Braun, J. E., Technical, N., Stephen, M., Kim, J., Cai, J., & Braun, J. E. (2018). *Common Faults and Their Prioritization in Small Commercial Buildings Common Faults and Their Prioritization in Small Commercial Buildings*. February 2017.

Li, D. H. W., Yang, L., & Lam, J. C. (2013). Zero energy buildings and sustainable development implications e A review. *Energy*. <https://doi.org/10.1016/j.energy.2013.01.070>

Linji Kong & Biao Ma (2019). Intelligent Manufacturing Model of Construction Indusry based on Internet of Things Technology

Lizana, J., Chacartegui, R., Barrios-padura, A., & Manuel, J. (2017). *Advances in thermal energy storage materials and their applications towards zero energy buildings : A critical review*. 203, 219–239. <https://doi.org/10.1016/j.apenergy.2017.06.008>

- Lou, S., Tsang, E. K. W., Li, D. H. W., Lee, E. W. M., & Lam, J. C. (2017). Towards zero energy school building designs in Hong Kong. *Energy Procedia*, 105, 182–187. <https://doi.org/10.1016/j.egypro.2017.03.299>
- McAllister, T. D., El-Tawab, S., & Heydari, M. H. (2017). Localization of Health Center Assets Through an IoT Environment (LoCATE). *2017 Systems and Information Engineering Design Symposium, SIEDS 2017*, 132–137. <https://doi.org/10.1109/SIEDS.2017.7937703>
- Mills, E. (2011). Building commissioning: A golden opportunity for reducing energy costs and greenhouse gas emissions in the United States. In *Energy Efficiency* (Vol. 4, Issue 2, pp. 145–173). <https://doi.org/10.1007/s12053-011-9116-8>
- Molina-huelva, M., Chacartegui, R., & Molina-huelva, M. (2017). *Accepted Manuscript*. <https://doi.org/10.1016/j.energy.2017.05.188>
- Morales, P., Letelier, V., & Mora, D. (2017). Implications of Life Cycle Energy Assessment of a new school building, regarding the nearly Zero Energy Buildings targets in EU: A case of Study. *Sustainable Cities and Society*. <https://doi.org/10.1016/j.scs.2017.03.016>
- Mytafides, A. C. K., Dimoudia, A., Mytafides, C. K., Dimoudi, A., & Zoras, S. (2017). Title : Transformation of a university building into a zero energy building in Mediterranean climate Transformation of a University Building into a Zero Energy Building in Mediterranean Climate. *Energy & Buildings*. <https://doi.org/10.1016/j.enbuild.2017.07.083>
- Rahman, M. A., Jakaria, A. H. M., & Al-Shaer, E. (2016). Formal Analysis for Dependable Supervisory Control and Data Acquisition in Smart Grids. *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 263–274. <https://doi.org/10.1109/DSN.2016.32>
- Richman, E. E. (n.d.). *“Measurement and Verification of Energy Savings and Performance from Advanced Lighting Controls.”*
- Santamouris, M. (2016). ScienceDirect Innovating to zero the building sector in Europe : Minimising the energy consumption , eradication of the energy poverty and mitigating the local climate change. *SOLAR ENERGY*. <https://doi.org/10.1016/j.solener.2016.01.021>
- Soleh, M. (2014). Desain Sistem SCADA Untuk Peningkatan Pelayanan Pelanggan Dan Efisiensi Operasional Sistem Tenaga Listrik di APJ Cirebon. *Jurnal Telekomunikasi Dan Komputer*, 5(1), 31–32.
- Stouffer, K., Joe, F., & Keut, K. (2006). Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control System Security. In *Nist Special Publication*.

https://scadahacker.com/library/Documents/ICS_Basics/SCADA Basics - NCS TIB 04-1.pdf

- Sun, Y., Huang, G., Xu, X., & Lai, A. C. (2018). Building-group-level performance evaluations of net zero energy buildings with non-collaborative controls. *Applied Energy*, 212(July 2017), 565–576. <https://doi.org/10.1016/j.apenergy.2017.11.076>
- Susanto, H., & Lysbetti, N. (2015). *ANALISA PENERAPAN SISTEM SCADA PADA PENGENDALIAN JARINGAN TEGANGAN MENENGAH 20 kV PT . PLN AREA*. 1–9.
- Won, J., Kwae, Y., & Woo, I. (2012). *Energy Procedia Automatic sensor arrangement system for building energy and environmental management*. <https://doi.org/10.1016/j.egypro.2011.12.887>
- Wu, W., Guo, J., Li, J., Hou, H., Meng, Q., & Wang, W. (2020). A multi-objective optimization design method in zero energy building study : A case study concerning small mass buildings in cold district of China. *Energy & Buildings*, 158(2018), 1613–1624. <https://doi.org/10.1016/j.enbuild.2017.10.102>

LAMPIRAN

Lampiran A

1. Listing dan Penjelasan Program

Program mikrokontroler dibuat menggunakan software Arduino IDE. Berikut listing beserta penjelasan program mikrokontroler yang telah dibuat.

a. Air Conditioner Sub System (ACSS)

Berikut listing untuk *air conditioner* dijelaskan secara bertahap.

1) *Library* (pustaka)

Library merupakan kumpulan kode program arduino dasar yang dikemas untuk memberikan perintah terhadap suatu komponen agar bekerja sesuai dengan fungsinya. Sebelum memasuki program inti, di awal penulisan program pada Arduino IDE harus dimasukkan atau diinkludkan beberapa library yaitu;

- a) `#include "FirebaseESP8266.h"`, yaitu library firebase untuk ESP8266.
- b) `#include <ESP8266Wifi.h>`, yaitu library wifi ESP8266.
- c) `#include <PZEM004Tv30.h>`, yaitu library sensor arus PZEM004T.
- d) `#include "DHT.h"`, yaitu library sensor temperatur DHT22.
- e) `#include "ESP8266HTTPClient"`, yaitu library untuk ESP8266 sebagai client yang melakukan request pada server.

```
#include "FirebaseESP8266.h"
#include <ESP8266WiFi.h>
#include <PZEM004Tv30.h>
#include "DHT.h"
#include <ESP8266HTTPClient.h>
```

2) *Define* (defenisi)

Pendefinisian dalam hal ini yang dimaksud adalah penentuan perangkat keras ataupun perangkat lunak yang akan digunakan sebagai pendukung bekerjanya mikrokontroler.

Beberapa pendefinisian yang harus dituliskan yaitu;

- a) `#define FIREBASE_HOST`, yang diisi dengan nama hosting firebase yang telah dibuat.
- b) `#define FIREBASE_AUTH`, yang diisi dengan firebase autentikasi firebasae yang akan digunakan.
- c) `#define WIFI_SSID`, yang diisi dengan nama wifi direct yang akan digunakan untuk koneksi ke internet.

- d) `#define WIFI_PASSWORD`, yang diisi dengan password wifi yang akan digunakan.

```
//Wifi-FireBase
#define FIREBASE_HOST "fds-android-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "KywiAMupDG4wOoL3E3kVeULboWQMACAdLf9V7sv"
#define WIFI_SSID "Muliadi"
#define WIFI_PASSWORD "qwerty1234"
```

Selanjutnya pendefinisian sensor-sensor input dan aktuator output yang akan digunakan seperti pada listing program berikut.

```
//PZEMAC
PZEM004Tv30 pzem(D6, D5);
//Tempsensor-RelayAC
#define DHTPIN D4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
#define relayAC D1
```

- a) PZEM004Tv30 pzem (D6,D5), yaitu penentuan pin D6 sebagai RX dan D5 sebagai TX untuk sensor arus PZEM004T.
- b) `#define DHTPIN D4`, yaitu penentuan pin D4 sebagai input dari sensor DHT.
- c) `#define DHTTYPE DHT22`, penentuan tipe sensor DHT yang digunakan yaitu DHT22.
- d) `#define relayAC D1`, yaitu penentuan D1 sebagai output relay untuk air conditioner.

3) *Variable* (variabel)

Variabel merupakan tempat penyimpanan data yang akan diproses, pembuatan variabel diawali dengan penulisan tipe data yang akan digunakan dan diikuti dengan nama variabel seperti pada listing program berikut.

```
FirestoreData firebaseData;
unsigned long timing = 0;
unsigned long timing2 = 0;
unsigned long timing1 = millis();
float temp;
float roomTemp;
float currentAC;
float voltageAC;
boolean statusPZEM_AC = true;
int counter;
```

Ada beberapa tipe data yang digunakan yaitu;

- a) `firebaseData`, untuk mengolah data dari firebase.

- b) unsigned long timing, untuk mengolah nilai millis.
- c) float temp, untuk mengolah data temperatur dalam satuan desimal (floating).
- d) float currentAC, untuk mengolah data arus pada air conditioner dalam satuan desimal (floating).
- e) float voltageAC, untuk mengolah data tegangan pada air conditioner dalam satuan desimal (floating).
- f) boolean statusPZEM_AC untuk mengolah status dari sensor arus PZEM004T.
- g) int counter, untuk mengolah data counter (penghitung) yang dibuat manual menggunakan data integer.

Selanjutnya variabel yang digunakan untuk pengirim data pada web server sebagai berikut :

```

HTTPClient http;
int httpCode;
String payload;
String host_post= "http://project3.monlab.site/index.php/post/ac";
String status = "Normal";
String toggle = "OFF";

```

Ada beberapa tipe data yang digunakan yaitu;

- a) HTTPClient, untuk protokol pengiriman data dari HTTP request.
- b) String payload, untuk mengolah data feedback dari request
- c) String host_post, untuk menginisialisasi domain host server
- d) String status, sebagai data yang dikirim mewakili kondisi sistem
- e) String toggle, sebagai data yang dikirim mewakili kondisi relay.

4) *Void Setup*

Void setup merupakan bagian program yang pertama dan hanya sekali diproses oleh mikrokontroler. Pada bagian ini mikrokontroler melakukan beberapa proses seperti yang terlihat pada listing program berikut ini.

```

void setup() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    counter++;
    if (counter >= 40) {break;} delay(500);}

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  dht.begin();
  pinMode(relayAC, OUTPUT);
  digitalWrite (relayAC, HIGH);
  roomTemp = dht.readTemperature();
  delay(8000);
  Firebase.setFloat(firebaseData, "/AC/initTemp", roomTemp);
}

```

- a) `Wifi.begin(WIFI_SSID, WIFI_PASSWORD)`, berfungsi untuk mengaktifkan wifi dengan menyesuaikan nama SSID dengan password.
- b) `counter++` , berfungsi untuk menghitung timer sampai 40 detik, jika telah melewati 40 detik namun tidak dapat terhubung dengan wifi tersebut maka mikrokontroler beralih pada mode offline.
- c) `Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH)`, berfungsi untuk menghubungkan mikrokontroler dengan firebase dengan menyesuaikan firebase host dengan firebase auth. Proses ini akan berhasil jika mikrokontroler dalam kondisi online. Namun jika dalam kondisi offline maka bagian ini akan diabaikan atau dilewatkan, dengan demikian mikrokontroler tidak akan mengirim data ke firebase.
- d) `dht.begin()` , berfungsi untuk mengaktifkan sensor DHT (sensor temperatur).
- e) `pinMode (relayAC,OUTPUT)`, untuk menyatakan bahwa relayAC merupakan OUTPUT.
- f) `digitalWrite(relayAC, HIGH)`, memberikan logika awal berupa HIGH pada relay.
- g) `roomTemp=dht.readTemperature()` , berfungsi untuk membaca suhu awal ruangan.
- h) `Firebase.setFloat(firebaseData,"/AC/initTemp",roomTemp)`, berfungsi untuk mengirim data ke firebase yang berisikan nilai suhu awal ruangan.

5) *Void Loop*

Void loop merupakan bagian program yang akan diproses oleh mikrokontroler secara terus menerus selama mikrokontroler bekerja. Pada bagian ini mikrokontroler melakukan beberapa proses seperti yang terlihat pada listing program berikut ini.

```

void loop() {
  check_mode:
  if (Firebase.getString(firebaseData, "/AC/switchAc")) {
    if (firebaseData.to<String>() == "1") { goto mode_on;}
    if (firebaseData.to<String>() == "0") {goto mode_off;}
    goto check_mode;
  }
}

```

Pada awal dinyalakan, mikrokontroler melakukan proses pengecekan mode (check mode) yaitu mengambil data perintah dari firebase, jika perintah dari firebase (switch AC) berlogika 1 (ON), maka mikrokontroler mengaktifkan air conditioner dan menjalankan sistem. Mode on ditunjukkan pada listing berikut ini.

```

mode_on:
  if (Firebase.getString(firebaseData, "/AC/switchAc")) {
    if (firebaseData.to<String>() == "0") { goto mode_off;}}
  timing2 = millis()-timing1;
  timing = timing2 / 60000;
  Firebase.setInt(firebaseData, "/AC/timeOnAc", timing);
  digitalWrite (relayAC, HIGH);
  temp = dht.readTemperature();
  if (isnan(temp)) { Firebase.setString(firebaseData, "/AC/temperatureAc", "-");
    Firebase.setString(firebaseData, "/AC/warningAc", "Check_temperature_sensor");}
  else { Firebase.setFloat(firebaseData, "/AC/temperatureAc", temp);
    Firebase.setString(firebaseData, "/AC/warningAc", "Temp_Sens_Normal");}
}

```

Mode on;

- a) `Firebase.setInt(firebaseData, "/AC/timeOnAc", timing)`, yaitu mikrokontroler mengirim data waktu ON air conditioner (timing) ke firebase.
- b) `digitalWrite (relayAC, HIGH)`, yaitu mikrokontroler menyalakan atau mengaktifkan air conditioner.
- c) `temp = dht.readTemperature()`, yaitu mikrokontroler membaca nilai temperatur ruangan.
- d) `Firebase.setFloat(firebaseData, "/AC/temperatureAc", temp)`, yaitu mikrokontroler mengirim data nilai temperatur ruangan ke firebase.

Selanjutnya pembacaan sensor arus PZEM004T, diperlihatkan pada listing berikut.

```

statusPZEM_AC = true;
currentAC = pzem.current();
voltageAC = pzem.voltage();
if (!isnan(currentAC)) { Firebase.setFloat(firebaseData, "/AC/currentAc", currentAC);
                        Firebase.setFloat(firebaseData, "/AC/voltageAc", voltageAC);
                        Firebase.setString(firebaseData, "/AC/warningAc", "Current_Sens_Normal");}

else { Firebase.setString(firebaseData, "/AC/currentAc", "-");
      Firebase.setString(firebaseData, "/AC/conditionAc", "Not detected");
      Firebase.setString(firebaseData, "/AC/warningAc", "Check_current_sensor");
statusPZEM_AC = false;}

```

- a) statusPZEM_AC = true, yaitu perintah mengaktifkan sensor arus air conditioner.
- b) currentAC = pzem.current(), yaitu perintah membaca arus air conditioner.
- c) voltageAC = pzem.voltage(), yaitu perintah membaca tegangan air conditioner.
- d) Firebase.setFloat(firebaseData, "/AC/currentAc", currentAC), yaitu mengirim nilai arus air conditioner ke firebase.
- e) Firebase.setFloat(firebaseData, "/AC/voltageAc", voltageAC), yaitu mengirim nilai tegangan air conditioner ke firebase.

Selanjutnya program bersyarat pada sistem ditunjukkan pada listing berikut.

```

if (timing >= 1 && currentAC > 0.0)
{ Firebase.setFloat(firebaseData, "/AC/currentAc", currentAC);
  Firebase.setFloat(firebaseData, "/AC/voltageAc", voltageAC);
  Firebase.setString(firebaseData, "/AC/conditionAc", "TURN ON");
  Firebase.setString(firebaseData, "/AC/warningAc", "Normal");}

```

Penjelasan:

Jika waktu menyala (ON) air conditioner telah melebihi 1 menit dan arus lebih besar dari 0A, maka mikrokontroler mengirim data ke firebase berupa nilai arus, nilai tegangan, kondisi air conditioner, dan catatan peringatan "Normal".

```

if (timing >= 1 && currentAC == 0.0)
{ Firebase.setFloat(firebaseData, "/AC/currentAc", currentAC);
  Firebase.setFloat(firebaseData, "/AC/voltageAc", voltageAC);
  Firebase.setString(firebaseData, "/AC/conditionAc", "Damaged");
  Firebase.setString(firebaseData, "/AC/warningAc", "Fault_detected");}

```

Penjelasan:

Jika waktu menyala (ON) air conditioner telah melebihi 1 menit dan arus 0A, maka mikrokontroler mengirim data ke firebase berupa nilai arus,

nilai tegangan, kondisi air conditioner, dan catatan peringatan “Fault detected”.

```
while (timing >= 5 && temp >= roomTemp)
{ digitalWrite (relayAC, LOW);
  Firebase.setFloat(firebaseData, "/AC/currentAc", currentAC);
  Firebase.setFloat(firebaseData, "/AC/voltageAc", voltageAC);
  Firebase.setString(firebaseData, "/AC/conditionAc", "TURN OFF");
  Firebase.setString(firebaseData, "/AC/warningAc", "Fault_detected");}
goto mode_on;
```

Penjelasan:

Jika air conditioner telah ON selama 5 menit ke atas dan temperatur ruangan masih sama atau lebih besar dari temperatur awal ruangan, maka relay air conditioner di OFF kan (air conditioner dinonaktifkan). Namun jika temperatur ruangan menjadi lebih rendah dari temperatur awal ruangan, maka air conditioner akan terus dinyalakan.

Namun jika perintah dari firebase (switch AC) berlogika 0 (OFF), maka mikrokontroler menonaktifkan air conditioner sistem dalam kondisi standby. Seperti yang diperlihatkan pada listing berikut.

```
mode_off:
digitalWrite (relayAC, LOW);
if (Firebase.getString(firebaseData, "/AC/switchAc")) {
  if (firebaseData.to<String>() == "1") {
    timing1 = millis();
    timing = timing1 - millis();
    Firebase.setInt(firebaseData, "/AC/timeOnAc", timing);
    goto mode_on;
  }
  Firebase.setString(firebaseData, "/AC/timeOnAc", "-");
  Firebase.setString(firebaseData, "/AC/currentAc", "-");
  Firebase.setString(firebaseData, "/AC/conditionAc", "-");
  Firebase.setString(firebaseData, "/AC/warningAc", "OFF");
  Firebase.setString(firebaseData, "/AC/voltageAc", "-");
}
goto mode_off;}
```

- a) digitalWrite (relayAC, LOW), yaitu mikrokontroler akan memberikan logika LOW pada relay dan air conditioner dan sistem akan dinonaktifkan.
- b) Mikrokontroler mengirim data ke firebase berupa nilai arus, nilai tegangan, kondisi air conditioner, dan catatan peringatan “OFF”.

Selanjutnya pengiriman seluruh data ke server, diperlihatkan pada listing berikut.

```
String data = "?current=" + String(currentAC) + "&voltage=" + String(voltageAC)
host_post += data;
http.begin(host_post); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
httpCode=http.GET();
payload=http.getString(); // get data from webhost continously
Serial.println(payload);
http.end(); //Close connection
```

Penjelasan:

Pada baris pertama data digabung dalam sebuah baris String data kemudian digabung/dimasukkan bersama dengan data dari host_post yang selanjutnya memulai melakukan melakukan request pada host_post dengan menggunakan header dan request type GET. Hasil dari request dimasukkan dalam variabel payload dan diakhiri dengan pemutusan koneksi pada server.

b. Lampu Sub System (LSS)

Berikut listing untuk lampu dijelaskan secara bertahap.

1) *Library* (pustaka)

Sebelum memasuki program inti, di awal penulisan program pada Arduino IDE harus dimasukkan atau diinkludkan beberapa library yaitu;

- a) #include "FirebaseESP8266.h", yaitu library firebase untuk ESP8266.
- b) #include <ESP8266Wifi.h>, yaitu library wifi ESP8266.
- c) # <PZEM004Tv30.h, yaitu library sensor arus PZEM004T.
- d) #include "ESP8266HTTPClient, yaitu library untuk ESP8266 sebagai client yang melakukan request pada server.

```
#include <FirebaseESP8266.h>
#include <ESP8266WiFi.h>
#include <PZEM004Tv30.h>
#include <ESP8266HTTPClient.h>
```

2) *Define* (defenisi)

Beberapa pendefinisian yang harus dituliskan yaitu;

- a) #define FIREBASE_HOST, yang diisi dengan nama hosting firebase yang telah dibuat.
- b) #define FIREBASE_AUTH, yang diisi dengan firebase autentikasi firebase yang akan digunakan.

- c) `#define WIFI_SSID`, yang diisi dengan nama wifi direct yang akan digunakan untuk koneksi ke internet.
- d) `#define WIFI_PASSWORD`, yang diisi dengan password wifi yang akan digunakan.

```
//Wifi-Firebase
#define FIREBASE_HOST "fds-android-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "KywiAMupDG4wOoL3E3kVeULboWQMACAdLfy9V7sv"
#define WIFI_SSID "Muliadi"
#define WIFI_PASSWORD "qwerty1234"
```

Selanjutnya pendefinisian sensor-sensor input dan aktuator output yang akan digunakan seperti pada listing program berikut.

```
//PZEMLamp
PZEM004Tv30 pzem(D6, D5);
//Lightsensor-RelayLamp
#define lightLDR D7
#define relayLamp D1
#define relayLamp2 D2
```

- a) PZEM004Tv30 pzem (D6,D5), yaitu penentuan pin D6 sebagai RX dan D5 sebagai TX untuk sensor arus PZEM004T.
- b) `#define lightLDR D7`, yaitu penentuan pin D7 sebagai input dari sensor LDR.
- c) `#define relayLamp D1`, penentuan pin D1 sebagai output relay untuk lampu 1.
- d) `#define relayLamp D2`, penentuan pin D2 sebagai output relay untuk lampu 2.

3) *Variable* (variabel)

Berikut variabel yang digunakan.

```
FirestoreData firestoreData;
unsigned long timing = 0;
float currentLamp;
float voltageLamp;
boolean statusPZEM_Lamp = true;
int counter;
```

Ada beberapa tipe data yang digunakan yaitu;

- a) `FirestoreData`, untuk mengolah data dari firebase.
- b) `unsigned long timing`, untuk mengolah nilai millis.

- c) float currentLamp, untuk mengolah data arus pada lampu dalam satuan desimal (floating).
- d) float voltageLamp, untuk mengolah data tegangan pada lampu dalam satuan desimal (floating).
- e) boolean statusPZEM_AC untuk mengolah status dari sensor arus PZEM004T.
- f) int counter, untuk mengolah data counter (penghitung) yang dibuat manual menggunakan data integer.

Selanjutnya variabel yang digunakan untuk pengirim data pada web server sebagai berikut :

```
HTTPClient http;
int httpCode;
String payload;
String host_post= "http://project3.monlab.site/index.php/post/lamp";
String status = "Normal";
String toggle = "OFF";
String light = "Null";
```

Ada beberapa tipe data yang digunakan yaitu;

- f) HTTPClient, untuk protokol pengiriman data dari HTTP request.
- g) String payload, untuk mengolah data feedback dari request
- h) String host_post, untuk menginisialisasi domain host server
- i) String status, sebagai data yang dikirim mewakili kondisi sistem
- j) Sting light, sebagai data yang dikirim mewakili pembacaan sensor
- k) String toggle, sebagai data yang dikirim mewakili kondisi relay.

4) *Void Setup*

Berikut program yang pertama dan hanya sekali dibaca oleh mikrokontroler.

```

void setup() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    counter++;
    if (counter >= 40) {break;}
    delay(500);}
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.setString(firebaseData, "/Lamp/cond", "Standby");
  pinMode(relayLamp, OUTPUT);
  pinMode(relayLamp2, OUTPUT);
  digitalWrite (relayLamp, HIGH);
  digitalWrite (relayLamp2, HIGH);}

```

- a) `Wifi.begin(WIFI_SSID, WIFI_PASSWORD)`, berfungsi untuk mengaktifkan wifi dengan menyesuaikan nama SSID dengan password.
- b) `counter++`, berfungsi untuk menghitung timer sampai 40 detik, jika telah melewati 40 detik namun tidak dapat terhubung dengan wifi tersebut maka mikrokontroler beralih pada mode offline.
- c) `Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH)`, berfungsi untuk menghubungkan mikrokontroler dengan firebase dengan menyesuaikan firebase host dengan firebase auth. Proses ini akan berhasil jika mikrokontroler dalam kondisi online. Namun jika dalam kondisi offline maka bagian ini akan diabaikan atau dilewatkan, dengan demikian mikrokontroler tidak akan mengirim data ke firebase.
- d) `Firebase.setString(firebaseData, "/Lamp/cond", "Standby")`, berfungsi untuk mengirim data ke firebase yang berisikan tulisan "Standby".
- e) `pinMode (relayLamp,OUTPUT)`, untuk menyatakan bahwa `relayLamp` merupakan OUTPUT.
- f) `pinMode (relayLamp2,OUTPUT)`, untuk menyatakan bahwa `relayLamp2` merupakan OUTPUT.
- g) `digitalWrite(relayLamp, HIGH)`, memberikan logika awal berupa HIGH pada relay 1.
- h) `digitalWrite(relayLamp2, HIGH)`, memberikan logika awal berupa HIGH pada relay 2.

5) *Void Loop*

Pada bagian void loop mikrokontroler melakukan beberapa proses seperti yang terlihat pada listing program berikut ini.

```

void loop() {
  check_mode:
  if (Firebase.getString(firebaseData, "/Lamp/switchL")) {
    if (firebaseData.to<String>() == "1") {goto mode_on;}
    if (firebaseData.to<String>() == "0") {goto mode_off;}}
  goto check_mode;

```

Pada awal dinyalakan, mikrokontroler melakukan proses pengecekan mode (check mode) yaitu mengambil data perintah dari firebase, jika perintah dari firebase (switch Lamp) berlogika 1 (ON), maka mikrokontroler mengaktifkan lampu dan menjalankan sistem. Mode on ditunjukkan pada listing berikut ini.

```

mode_on:
if (Firebase.getString(firebaseData, "/Lamp/switchL")) {
  if (firebaseData.to<String>() == "0") {goto mode_off;}}
if (Firebase.getString(firebaseData, "/Lamp/switchL2")) {
  if (firebaseData.to<String>() == "1") {
    digitalWrite (relayLamp2, LOW) ;}
  if (firebaseData.to<String>() == "0") {
    digitalWrite (relayLamp2, HIGH) ;}}
timing2 = millis() - timing1;
timing = timing2 / 60000;
Firebase.setInt(firebaseData, "/Lamp/timeOnL", timing);
int value_ldr = digitalRead(lightLDR);
if (value_ldr == HIGH) {
  digitalWrite (relayLamp, LOW) ;
  Firebase.setString(firebaseData, "/Lamp/sensCond", "DARK");
  Firebase.setString(firebaseData, "/Lamp/lampCond", "TURN ON");
  Firebase.setString(firebaseData, "/Lamp/warningL", "Normal");}
else {
  digitalWrite (relayLamp, HIGH) ;
  Firebase.setString(firebaseData, "/Lamp/sensCond", "LIGHT");
  Firebase.setString(firebaseData, "/Lamp/lampCond", "TURN OFF");
  Firebase.setString(firebaseData, "/Lamp/warningL", "Normal");}

```

Mode on;

- Firebase.setInt(firebaseData, "/Lamp/timeOnL", timing), yaitu mikrokontroler mengirim data waktu ON lampu (timing) ke firebase.
- int value_ldr(digitalRead(lightLDR), yaitu mikrokontroler membaca data digital dari sensor LDR, dan tipe datanya adalah integer.
- if(value_ldr==HIGH){digitalWrite(relayLamp,LOW);
else{digitalWrite(relayLamp,HIGH)}; yaitu jika LDR berlogika HIGH (gelap), maka relay lampu diberikan logika LOW (lampu dinyalakan), dan sebaliknya.

Selanjutnya pembacaan sensor arus PZEM004T, diperlihatkan pada listing berikut.

```

statusPZEM_Lamp = true;
currentLamp = pzem.current();
voltageLamp = pzem.voltage();
if (!isnan(currentLamp)) {
  Firebase.setFloat(firebaseData, "/Lamp/currentL", currentLamp);
  Firebase.setFloat(firebaseData, "/Lamp/voltageL", voltageLamp);}
else {
  Firebase.setString(firebaseData, "/Lamp/currentL", "Not detected");
  Firebase.setString(firebaseData, "/Lamp/voltageL", "Not detected");
  statusPZEM_Lamp = false;}

```

- a) statusPZEM_Lamp = true, yaitu perintah mengaktifkan sensor arus lampu.
- b) currentLamp = pzem.current(), yaitu perintah membaca arus lampu.
- c) voltageLamp = pzem.voltage(), yaitu perintah membaca tegangan lampu.
- d) Firebase.setFloat(firebaseData, "/Lamp/currentL", currentL), yaitu mengirim nilai arus lampu ke firebase.
- e) Firebase.setFloat(firebaseData, "/Lamp/voltageL", voltageL), yaitu mengirim nilai tegangan lampu ke firebase.

Selanjutnya program bersyarat pada sistem ditunjukkan pada listing berikut.

```

while ( value_1dr == HIGH && currentLamp == 0) {
  Firebase.setString(firebaseData, "/Lamp/sensCond", "DARK");
  Firebase.setString(firebaseData, "/Lamp/lampCond", "Damaged");
  Firebase.setString(firebaseData, "/Lamp/warningL", "Fault detected");
  statusPZEM_Lamp = true; currentLamp = pzem.current();}
goto mode_on;}

```

Penjelasan:

Jika waktu menyala (ON) arus lampu tetap 0A (tidak ada arus), maka mikrokontroler mengirim data ke firebase berupa informasi “Fault detected”.

Namun jika perintah dari firebase (switch Lamp) berlogika 0 (OFF), maka mikrokontroler menonaktifkan lampu dan sistem dalam kondisi standby. Seperti yang diperlihatkan pada listing berikut.

```

mode_off:
digitalWrite (relayLamp, HIGH);
if (Firebase.getString(firebaseData, "/Lamp/switchL")) {
  if (firebaseData.to<String>() == "1") { timing1 = millis();
    timing = timing1 - millis();
    Firebase.setInt(firebaseData, "/Lamp/timeOnL", timing);
    goto mode_on;}
  Firebase.setString(firebaseData, "/Lamp/timeOnL", "-");
  Firebase.setString(firebaseData, "/Lamp/sensCond", "-");
  Firebase.setString(firebaseData, "/Lamp/lampCond", "-");
  Firebase.setString(firebaseData, "/Lamp/warningL", "OFF");
  Firebase.setString(firebaseData, "/Lamp/currentL", "-");
  Firebase.setString(firebaseData, "/Lamp/voltageL", "-");}
if (Firebase.getString(firebaseData, "/Lamp/switchL2")) {
  if (firebaseData.to<String>() == "1") {digitalWrite (relayLamp2, LOW) ;}
  if (firebaseData.to<String>() == "0") {digitalWrite (relayLamp2, HIGH) ;}
  goto mode_off;
}

```

Penjelasan :

- a) digitalWrite (relayLamp, HIGH);
 if (Firebase.getString(firebaseData, "/Lamp/switchL"))
 {if (firebaseData.to<String>() == "1") { timing1 = millis();timing =
 timing1 - millis(); Firebase.setInt(firebaseData, "/Lamp/timeOnL",
 timing); goto mode_on;}}

Pada mode off, relayLamp akan diberikan logika HIGH (lampu dimatikan) dan waktu ON akan direset. Setelah itu sistem akan kembali standby, jika data switchL dari firebase berubah menjadi ON (logika 1), maka sistem akan dialihkan ke mode ON.

- b) if (Firebase.getString(firebaseData, "/Lamp/switchL2")) {
 if (firebaseData.to<String>() == "1") {digitalWrite (relayLamp2, LOW)
 ;}
 if (firebaseData.to<String>() == "0") {digitalWrite (relayLamp2,
 HIGH) ;}

Pada mode ON, jika data switchL2 dari firebase berlogika 1, maka relay lampu diberikan kondisi LOW (lampu dinyalakan). Dan jika data switchL2 dari firebase berlogika 0, maka relay lampu diberikan kondisi HIGH (lampu dimatikan). Lampu 2 tidak dipengaruhi oleh sensor LDR, namun dapat dikendalikan secara manual.

Selanjutnya pengiriman seluruh data ke server, diperlihatkan pada listing berikut.

```
String data = "?current=" + String(currentAC) + "&voltage=" + String(voltageAC)
host_post += data;
http.begin(host_post); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
httpCode=http.GET();
payload=http.getString(); // get data from webhost continously
Serial.println(payload);
http.end(); //Close connection
```

Penjelasan:

Pada baris pertama data digabung dalam sebuah baris String data kemudian digabung/dimasukkan bersama dengan data dari host_post yang selanjutnya memulai melakukan request pada host_post dengan menggunakan header dan request type GET. Hasil dari request dimasukkan dalam variabel payload dan diakhiri dengan pemutusan koneksi pada server.

c. Socket Sub System (SSS)

Berikut listing untuk lampu dijelaskan secara bertahap.

1) Library (pustaka)

Sebelum memasuki program inti, di awal penulisan program pada Arduino IDE harus dimasukkan atau diinkludkan beberapa library yaitu;

- a) #include "FirebaseESP8266.h", yaitu library firebase untuk ESP8266.
- b) #include <ESP8266Wifi.h>, yaitu library wifi ESP8266.
- c) #include <PZEM004Tv30.h>, yaitu library sensor arus PZEM004T.
- d) #include <OneWire.h>, yaitu library untuk sensor suhu DS18B20
- e) #include <DallasTemperature.h>, yaitu library untuk sensor suhu DS18B20.
- f) #include "ESP8266HTTPClient", yaitu library untuk ESP8266 sebagai client yang melakukan request pada server.

```
#include <FirebaseESP8266.h>
#include <ESP8266WiFi.h>
#include <PZEM004Tv30.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <ESP8266HTTPClient.h>
```

2) Define (defenisi)

Beberapa pendefinisian yang harus dituliskan yaitu;

- a) #define FIREBASE_HOST, yang diisi dengan nama hosting firebase yang telah dibuat.
- b) #define FIREBASE_AUTH, yang diisi dengan firebase autentikasi firebase yang akan digunakan.

- c) `#define WIFI_SSID`, yang diisi dengan nama wifi direct yang akan digunakan untuk koneksi ke internet.
- d) `#define WIFI_PASSWORD`, yang diisi dengan password wifi yang akan digunakan.

```
//Wifi-Firebase
#define FIREBASE_HOST "fds-android-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "KywiAMupDG4wOoL3E3kVeULboWQMACAdLfY9V7sv"
#define WIFI_SSID "Muliadi"
#define WIFI_PASSWORD "qwerty1234"
```

Selanjutnya pendefinisian sensor-sensor input dan aktuator output yang akan digunakan seperti pada listing program berikut.

```
//PZEMSC
PZEM004Tv30 pzem(D6, D5);
//RelaySC
#define buzzer D8
#define relaySC D1
const int oneWireBus = D2;
```

- a) PZEM004Tv30 pzem (D6,D5), yaitu penentuan pin D6 sebagai RX dan D5 sebagai TX untuk sensor arus PZEM004T.
- b) `#define buzzer D8`, yaitu penentuan pin D8 sebagai output ke buzzer.
- c) `#define relaySC D1`, penentuan pin D1 sebagai output relay untuk stok kontak.
- d) `const int oneWire = D2` , penentuan D2 sebagai input dari sensor suhu DS18B20.

3) *Variable* (variabel)

Berikut variabel yang digunakan.

```
FirebaseData firebaseData;
unsigned long timing = 0;
float currentSC;
float powerSC;
float voltageSC;
boolean statusPZEM_SC = true;
unsigned long startmillis, currentmillis;
int counter;
int jeda = 0;
OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);
```

Ada beberapa tipe data yang digunakan yaitu;

- a) firebaseData, untuk mengolah data dari firebase.
- b) unsigned long timing, untuk mengolah nilai millis.
- c) unsigned long startmillis, currentmillis, untuk mengolah nilai millis.
- d) float currentSC, untuk mengolah data arus pada stok kontak dalam satuan desimal (floating).
- e) float voltageSC, untuk mengolah data tegangan pada stok kontak dalam satuan desimal (floating).
- f) float powerSC, untuk mengolah data daya pada stok kontak dalam satuan desimal (floating).
- g) boolean statusPZEM_AC untuk mengolah status dari sensor arus PZEM004T.
- h) int counter, untuk mengolah data counter (penghitung) yang dibuat manual menggunakan data integer.
- i) int jeda, untuk mengolah data jeda counter.
- j) OneWire oneWire(oneWireBus), untuk mengolah data sensor suhu DS18B20.
- k) DallasTemperature sensor (&oneWire), untuk mengolah data sensor DS18B20.

Selanjutnya variabel yang digunakan untuk pengiriman data pada web server sebagai berikut :

```
HTTPClient http;
int httpCode;
String payload;
String host_post= "http://project3.monlab.site/index.php/post/shock";
String status = "Normal";
String toggle = "OFF";
```

Ada beberapa tipe data yang digunakan yaitu;

- l) HTTPClient, untuk protokol pengiriman data dari HTTP request.
- m) String payload, untuk mengolah data feedback dari request
- n) String host_post, untuk menginisialisasi domain host server
- o) String status, sebagai data yang dikirim mewakili kondisi sistem
- p) String toggle, sebagai data yang dikirim mewakili kondisi relay.

4) *Void Setup*

Berikut program yang pertama dan hanya sekali dibaca oleh mikrokontroler.

```

void setup() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    counter++;
    if (counter >= 40) {break;}
    delay(500);}
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  //Firebase.reconnectWiFi(true);
  Firebase.setString(firebaseData, "/SC/cond", "Standby");
  sensors.begin();
  pinMode(relaySC, OUTPUT);
  pinMode (buzzer, OUTPUT);
  digitalWrite (relaySC, LOW);}

```

- a) `Wifi.begin(WIFI_SSID, WIFI_PASSWORD)`, berfungsi untuk mengaktifkan wifi dengan menyesuaikan nama SSID dengan password.
- b) `counter++`, berfungsi untuk menghitung timer sampai 40 detik, jika telah melewati 40 detik namun tidak dapat terhubung dengan wifi tersebut maka mikrokontroler beralih pada mode offline.
- c) `Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH)`, berfungsi untuk menghubungkan mikrokontroler dengan firebase dengan menyesuaikan firebase host dengan firebase auth. Proses ini akan berhasil jika mikrokontroler dalam kondisi online. Namun jika dalam kondisi offline maka bagian ini akan diabaikan atau dilewatkan, dengan demikian mikrokontroler tidak akan mengirim data ke firebase.
- d) `Firebase.setString(firebaseData, "/SC/cond", "Standby")`, berfungsi untuk mengirim data ke firebase yang berisikan tulisan "Standby".
- e) `sensors.begin()`, untuk memulai menjalankan sensor DS18B20.
- f) `pinMode (relaySC, OUTPUT)`, untuk menyatakan bahwa relaySC merupakan OUTPUT.
- g) `pinMode (buzzer, OUTPUT)`, untuk menyatakan bahwa buzzer merupakan OUTPUT.
- h) `digitalWrite(relaySC, LOW)`, memberikan logika awal berupa LOW (ON) pada relay stok kontak.

5) *Void Loop*

Pada bagian void loop mikrokontroler melakukan beberapa proses seperti yang terlihat pada listing program berikut ini.

```

void loop() {
  check_mode:
  if (Firebase.getString(firebaseData, "/SC/switchSC")) {
    if (firebaseData.to<String>() == "1") {goto mode_on;}
    if (firebaseData.to<String>() == "0") {goto mode_off;}}
  goto check_mode;
}

```

Pada awal dinyalakan, mikrokontroler melakukan proses pengecekan mode (check mode) yaitu mengambil data perintah dari firebase, jika perintah dari firebase (switch SC) berlogika 1 (ON), maka mikrokontroler mengaktifkan lampu dan menjalankan sistem. Mode on ditunjukkan pada listing berikut ini.

```

mode_on:
  if (Firebase.getString(firebaseData, "/SC/switchSC")) {
    if (firebaseData.to<String>() == "0") {goto mode_off;}}
  timing2 = millis() - timing1;
  timing = timing2 / 60000;
  Firebase.setInt(firebaseData, "/SC/timeOnSC", timing);
  digitalWrite (relaySC, HIGH);
  sensors.requestTemperatures();
  float temperatureC = sensors.getTempCByIndex(0);
  Firebase.setFloat (firebaseData, "/SC/thermalSC", temperatureC);
  Firebase.setString(firebaseData, "/SC/warningSC", "-");
}

```

Mode on;

- d) `Firebase.setInt(firebaseData, "/SC/timeOnSC", timing)`, yaitu mikrokontroler mengirim data waktu ON stok kontak (timing) ke firebase.
- e) `digitalWrite(relaySC, HIGH)`, yaitu mikrokontroler memberikan kondisi HIGH (ON) pada relay stok kontak.
- f) `sensors.requestTemperatures()`, yaitu mikrokontroler meminta data temperature sensor DS18B20.
- g) `float temperatureC = sensors.getTempCByIndex(0)`, yaitu mikrokontroler mengambil data temperature sensor DS18B20 kemudian dikonversi menjadi celsius dengan bentuk desimal (floating).
- h) `Firebase.setFloat(firebaseData, "/SC/thermalSC", temperatureC)`, yaitu firebase mengirim data suhu sensor DS18B20 ke firebase.

Selanjutnya pembacaan sensor arus PZEM004T, diperlihatkan pada listing berikut.

```
//Read Current Sensor
statusPZEM_SC = true;
currentSC = pzem.current();
voltageSC = pzem.voltage();
if (!isnan(currentSC)) {
  Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC);
  Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC);}
else {
  Firebase.setString(firebaseData, "/SC/currentSC", "-");
  Firebase.setString(firebaseData, "/SC/voltageSC", "-");
  Firebase.setString(firebaseData, "/SC/conditionSC", "Not detected");
  Firebase.setString(firebaseData, "/SC/warningSC", "Check current sensor");
  statusPZEM_SC = false;}
```

- a) statusPZEM_SC = true, yaitu perintah mengaktifkan sensor arus stok kontak.
- b) currentSC = pzem.current(), yaitu perintah membaca arus stok kontak.
- c) voltageSC = pzem.voltage(), yaitu perintah membaca tegangan stok kontak.
- d) Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC), yaitu mengirim nilai arus stok kontak ke firebase.
- e) Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC), yaitu mengirim nilai tegangan stok kontak ke firebase.

Selanjutnya program bersyarat pada sistem ditunjukkan pada listing berikut.

```
if (temperatureC >= 71) {
  while (1) {
    digitalWrite (relaySC, LOW);
    digitalWrite (buzzer, HIGH); delay (100);
    digitalWrite (buzzer, LOW); delay (100);
    digitalWrite (buzzer, HIGH); delay (100);
    digitalWrite (buzzer, LOW); delay (100);
    Firebase.setFloat (firebaseData, "/SC/thermalSC", temperatureC);
    Firebase.setString(firebaseData, "/SC/warningSC", "Overheat");
    Firebase.setString(firebaseData, "/SC/conditionSC", "OFF");
    if (temperatureC < 71) {
      digitalWrite (relaySC, HIGH);break;}}}
```

Penjelasan:

Jika suhu stok kontak (sensor DS18B20) mencapai 71 derajat celsius ke atas, maka relay stok kontak akan di OFF kan dan buzzer akan mengeluarkan bunyi beep. Sedangkan jika suhu stok kontak dibawah 71 derajat celsius maka relay stok kontak akan di ON kan kembali dan buzzer berhenti mengeluarkan bunyi beep.

```

powerSC = pzem.power();
if ( !isnan(powerSC) ) {
  Firebase.setFloat(firebaseData, "/SC/powerSC", powerSC);
} else {Firebase.setString(firebaseData, "/SC/powerSC", "-");}
if (powerSC == 0.0)
{ Firebase.setString(firebaseData, "/SC/conditionSC", "No device installed");
  Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC);
  Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC);
  Firebase.setString(firebaseData, "/SC/warningSC", "-");
  Firebase.setFloat(firebaseData, "/SC/powerSC", powerSC);
  jeda = 0;}

```

Penjelasan:

Mikrokontroler mengambil data daya (power) dari sensor PZEM004T, lalu mengirim ke firebase. Jika nilai daya adalah 0 Watt, maka mikrokontroler akan mengirim data kondisi stok kontak yaitu “Not device installed”.

```

if (powerSC >= 1.0 && powerSC <= 400)
{ Firebase.setString(firebaseData, "/SC/conditionSC", "ON");
  Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC);
  Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC);
  Firebase.setString(firebaseData, "/SC/warningSC", "-");
  Firebase.setFloat(firebaseData, "/SC/powerSC", powerSC);
  jeda = 0;}

```

Penjelasan :

Jika daya pada stok kontak berkisar antara 1 – 400 Watt, maka mikrokontroler akan mengirim nilai daya yang terukur ke firebase serta tidak memberikan peringatan dalam hal ini stok kontak dalam kondisi aman (tidak kelebihan beban).

```

if (powerSC > 400 && powerSC <= 420)
{ Firebase.setString(firebaseData, "/SC/conditionSC", "ON");
  Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC);
  Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC);
  Firebase.setString(firebaseData, "/SC/warningSC", "Almost overload");
  Firebase.setFloat(firebaseData, "/SC/powerSC", powerSC);
  jeda = 0;}

```

Penjelasan :

Jika daya pada stok kontak berkisar antara 401 – 420 Watt, maka mikrokontroler akan mengirim nilai daya yang terukur ke firebase serta

memberikan peringatan “Almost overload”, dalam hal ini stok kontak dalam kondisi kurang aman (menghampiri kelebihan beban).

```

if (powerSC > 420) {
  digitalWrite (buzzer, HIGH); delay(10);
  digitalWrite (buzzer, LOW); delay(10);
  digitalWrite (buzzer, HIGH); delay(10);
  digitalWrite (buzzer, LOW); delay(10);
  currentmillis = millis ();
  if (currentmillis - startmillis >= 1000) { startmillis = currentmillis; jeda++;
    if (jeda >= 30) {
      while (1) {
        digitalWrite (relaySC, LOW);
        statusPZEM_SC = false;
        currentSC = pzem.current();
        powerSC = pzem.power();
        jeda = 0; } } }
  Firebase.setString(firebaseData, "/SC/conditionSC", "ON");
  Firebase.setFloat(firebaseData, "/SC/currentSC", currentSC);
  Firebase.setFloat(firebaseData, "/SC/voltageSC", voltageSC );
  Firebase.setString(firebaseData, "/SC/warningSC", "Unplug any load and press reset button");
  Firebase.setFloat(firebaseData, "/SC/powerSC", powerSC ); } }

```

Penjelasan :

Jika daya pada stok kontak melebihi 420 Watt, maka buzzer akan mengeluarkan bunyi beep pendek secara terus menerus hingga 30 detik. Jika beban dilepas dan daya kembali berada di bawah 420 Watt maka bunyi beep akan berhenti, namun jika bunyi beep pendek diabaikan hingga 31 detik, maka relay stok kontak akan di OFF kan dan mikrokontroler akan mengirim data ke firebase berupa informasi “Unplug and press reset button”. Untuk menyalakan kembali stok kontak, maka beberapa beban harus dilepas terlebih dahulu kemudian menekan tombol reset supaya sistem set up ulang.

Namun jika perintah dari firebase (switch SC) berlogika 0 (OFF), maka mikrokontroler menonaktifkan lampu dan sistem dalam kondisi standby. Seperti yang diperlihatkan pada listing berikut.

```

mode_off:
digitalWrite (relaySC, LOW);
if (Firebase.getString(firebaseData, "/SC/switchSC")) {
  if (firebaseData.to<String>() == "1") {timing1 = millis();
    timing = timing1 - millis();
    Firebase.setInt(firebaseData, "/SC/timeOnSC", timing);
    goto mode_on;}
  //timing = 0;
  Firebase.setString(firebaseData, "/SC/timeOnSC", "-");
  Firebase.setString(firebaseData, "/SC/currentSC", "-");
  Firebase.setString(firebaseData, "/SC/voltageSC", "-");
  Firebase.setString(firebaseData, "/SC/conditionSC", "-");
  Firebase.setString(firebaseData, "/SC/warningSC", "OFF");
  Firebase.setString(firebaseData, "/SC/powerSC", "-");}
goto mode_off;

```

Penjelasan:

Pada mode OFF relay stok kontak akan di OFF kan, mikrokontroler akan mengirim beberapa data ke firebase dan sistem berada dalam kondisi standby.

Selanjutnya pengiriman seluruh data ke server, diperlihatkan pada listing berikut.

```

String data = "?current=" + String(currentAC) + "&voltage=" + String(voltageAC)
host_post += data;
http.begin(host_post); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
httpCode=http.GET();
payload=http.getString(); // get data from webhost continously
Serial.println(payload);
http.end(); //Close connection

```

Penjelasan:

Pada baris pertama data digabung dalam sebuah baris String data kemudian digabung/dimasukkan bersama dengan data dari host_post yang selanjutnya memulai melakukan melakukan request pada host_post dengan menggunakan header dan request type GET. Hasil dari request dimasukkan dalam variabel payload dan diakhiri dengan pemutusan koneksi pada server.

d. Panel Sub System (PSS)

Berikut listing untuk lampu dijelaskan secara bertahap.

1) Library (pustaka)

Sebelum memasuki program inti, di awal penulisan program pada Arduino IDE harus dimasukkan atau diinkludkan beberapa library yaitu;

- a) `#include "FirebaseESP8266.h"`, yaitu library firebase untuk ESP8266.
- b) `#include <ESP8266Wifi.h>`, yaitu library wifi ESP8266.
- c) `# <PZEM004Tv30.h`, yaitu library sensor arus PZEM004T.
- d) `#include "ESP8266HTTPClient"`, yaitu library untuk ESP8266 sebagai client yang melakukan request pada server.

```
#include <FirebaseESP8266.h>
#include <ESP8266WiFi.h>
#include <PZEM004Tv30.h>
#include <ESP8266HTTPClient.h>
```

2) *Define* (defenisi)

Beberapa pendefinisian yang harus dituliskan yaitu;

- a) `#define FIREBASE_HOST`, yang diisi dengan nama hosting firebase yang telah dibuat.
- b) `#define FIREBASE_AUTH`, yang diisi dengan firebase autentikasi firebasae yang akan digunakan.
- c) `#define WIFI_SSID`, yang diisi dengan nama wifi direct yang akan digunakan untuk koneksi ke internet.
- d) `#define WIFI_PASSWORD`, yang diisi dengan password wifi yang akan digunakan.

```
//Wifi-FireBase
#define FIREBASE_HOST "fds-android-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "KywiAMupDG4wOoL3E3kVeULboWQMACAdLf9V7sv"
#define WIFI_SSID "Muliadi"
#define WIFI_PASSWORD "qwerty1234"
```

Selanjutnya pendefenisian sensor-sensor input dan aktuator output yang akan digunakan seperti pada listing program berikut.

```
//PZEMSC
PZEM004Tv30 pzem(D6, D5);
```

- a) `PZEM004Tv30 pzem (D6,D5)`, yaitu penentuan pin D6 sebagai RX dan D5

6) *Variable* (variabel)

Berikut variabel yang digunakan.

```

FirebaseData firebaseData;
float currentPanelCons;
float voltagePanelCons;
boolean statusPZEM_Panel = true;
int counter;

```

Ada beberapa tipe data yang digunakan yaitu;

- g) firebaseData, untuk mengolah data dari firebase.
- h) float currentPanel, untuk mengolah data arus pada panel dalam satuan desimal (floating).
- i) float voltagePanel, untuk mengolah data tegangan pada panel dalam satuan desimal (floating).
- j) boolean statusPZEM_Panel untuk mengolah status dari sensor arus PZEM004T.
- k) int counter, untuk mengolah data counter (penghitung) yang dibuat manual menggunakan data integer.

Selanjutnya variabel yang digunakan untuk pengirim data pada web server sebagai berikut :

```

HTTPClient http;
int httpCode;
String payload;
String host_post= "http://project3.monlab.site/index.php/post/panel_cons";
String status;

```

Ada beberapa tipe data yang digunakan yaitu;

- a) HTTPClient, untuk protokol pengiriman data dari HTTP request.
 - b) String payload, untuk mengolah data feedback dari request
 - c) String host_post, untuk menginisialisasi domain host server
 - d) String status, sebagai data yang dikirim mewakili kondisi sistem
- 3) *Void Setup*

Berikut program yang pertama dan hanya sekali dibaca oleh mikrokontroler.

```

void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    counter++;
    if (counter >= 40) {
      break;
    }
    delay(500);
  }
  Serial.println("WiFi connected");
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}

```

- i) `Wifi.begin(WIFI_SSID, WIFI_PASSWORD)`, berfungsi untuk mengaktifkan wifi dengan menyesuaikan nama SSID dengan password.
- j) `counter++`, berfungsi untuk menghitung timer sampai 40 detik, jika telah melewati 40 detik namun tidak dapat terhubung dengan wifi tersebut maka mikrokontroler beralih pada mode offline.
- k) `Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH)`, berfungsi untuk menghubungkan mikrokontroler dengan firebase dengan menyesuaikan firebase host dengan firebase auth. Proses ini akan berhasil jika mikrokontroler dalam kondisi online. Namun jika dalam kondisi offline maka bagian ini akan diabaikan atau dilewatkan, dengan demikian mikrokontroler tidak akan mengirim data ke firebase.

4) *Void Loop*

Pada bagian void loop mikrokontroler melakukan beberapa proses seperti yang terlihat pada listing program berikut ini.

```

statusPZEM_Panel = true;
currentPanelCons = pzem.current();
voltagePanelCons = pzem.voltage();
if (!isnan(currentPanelCons)) {
  Firebase.setFloat(firebaseData, "/Panel/currentCons", currentPanelCons);
  Firebase.setFloat(firebaseData, "/Panel/voltageCons", voltagePanelCons);
  Firebase.setString(firebaseData, "/Panel/warningP", "Normal");
  status = "Normal";
}
else {
  Firebase.setString(firebaseData, "/Panel/currentCons", "-");
  Firebase.setString(firebaseData, "/Panel/voltageCons", "-");
  Firebase.setString(firebaseData, "/Panel/warningP", "Check_current_sensor_cons");
  statusPZEM_Panel = false;
  status = "Failure";
}

```

Pada awal dinyalakan, mikrokontroler melakukan proses pembacaan sensor, kemudian jika pembacaan berhasil mikrokontroler akan melakukan pengiriman data ke firebase antara lain :

- a) `Firestore.setString(firebaseData, "/Panel/CurrentCons", currentPanelCons)`, yaitu mikrokontroler mengirim data arus panel (`currentPanelCons`) ke firebase.
- b) `Firestore.setString(firebaseData, "/Panel/voltageCons", voltagePanelCons)`, yaitu mikrokontroler mengirim data tegangan panel (`voltagePanelCons`) ke firebase.
- c) `Firestore.setString(firebaseData, "/Panel/warningP", "Normal")`, yaitu mikrokontroler mengirim data string Normal ke firebase.

Selanjutnya pengiriman seluruh data ke server, diperlihatkan pada listing berikut.

```
String data = "?current=" + String(currentAC) + "&voltage=" + String(voltageAC)
host_post += data;
http.begin(host_post); //Specify request destination
http.addHeader("Content-Type", "application/x-www-form-urlencoded");
httpCode=http.GET();
payload=http.getString(); // get data from webhost continously
Serial.println(payload);
http.end(); //Close connection
```

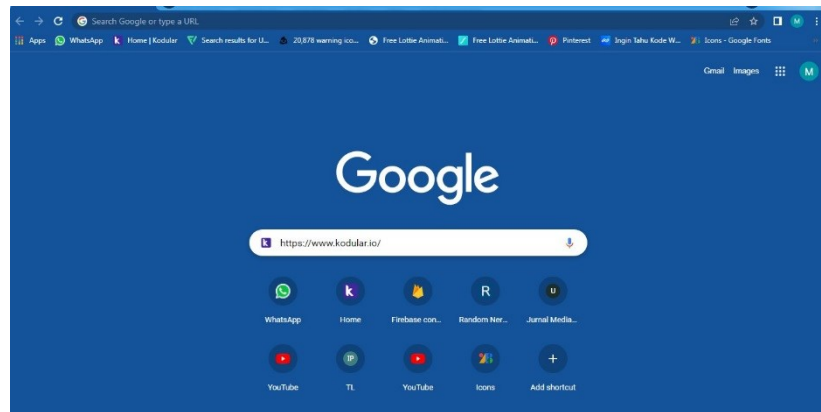
Penjelasan:

Pada baris pertama data digabung dalam sebuah baris String data kemudian digabung/dimasukkan bersama dengan data dari `host_post` yang selanjutnya memulai melakukan melakukan request pada `host_post` dengan menggunakan header dan request type GET. Hasil dari request dimasukkan dalam variabel `payload` dan diakhiri dengan pemutusan koneksi pada server.

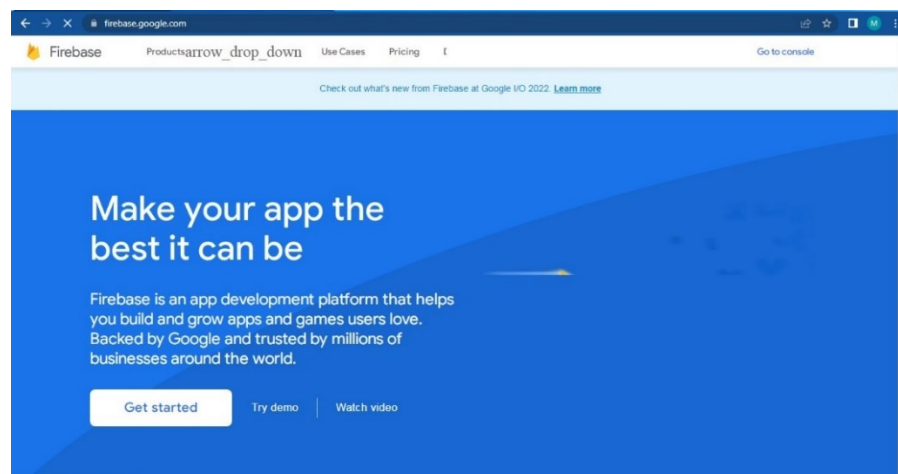
2. Pembuatan Firebase MACSys

Firebase merupakan tools database dari Google yang dapat kita gunakan untuk mengembangkan aplikasi. Langkah-langkah pembuatan database pada firebase yaitu sebagai berikut;

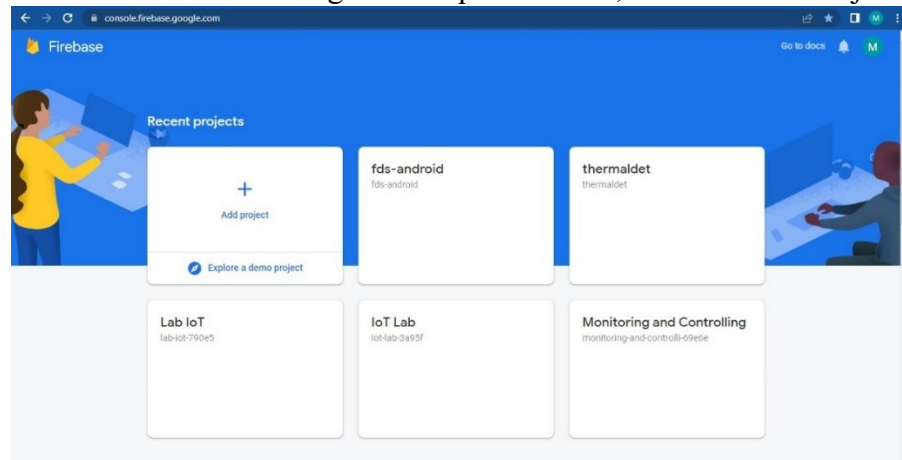
- a. Buka aplikasi google chrome (atau sejenisnya), pastikan komputer terkoneksi dengan internet (online).



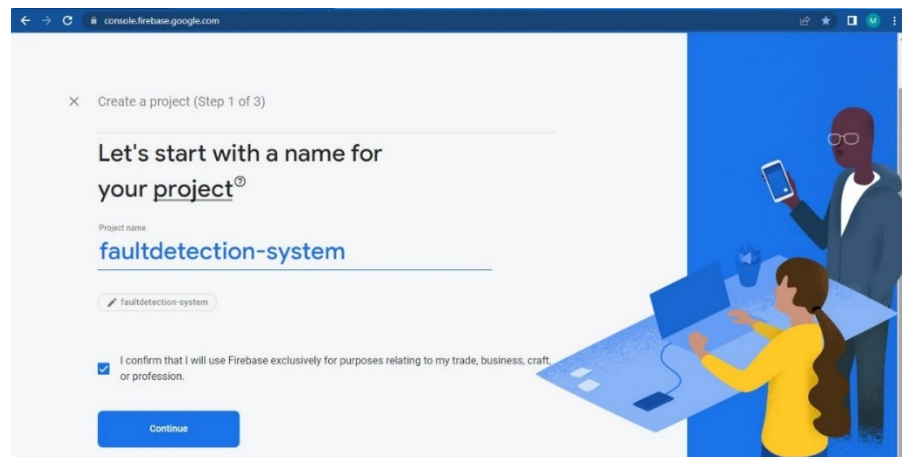
- a. Buka laman <https://firebase.google.com/> , seperti gambar berikut ini. Lalu klik Go to Console.



- b. Setelah itu akan muncul gambar seperti berikut, lalu klik Add Project.

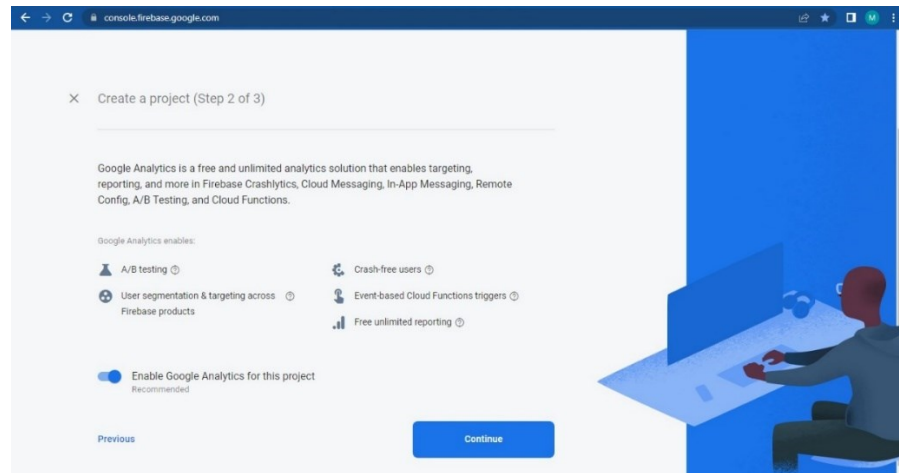


- c. Setelah mengklik Add Project, maka akan muncul gambar seperti di bawah ini.



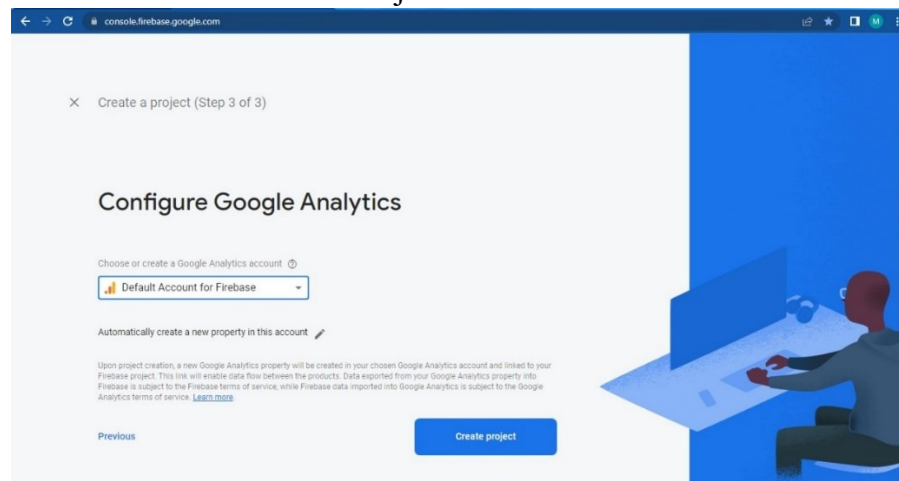
Pada bagian ini, masukkan nama project yang akan dibuat pada bagian Project Name.

- d. Setelah mengisi nama project, klik Continue maka akan muncul tampilan sebagai berikut.

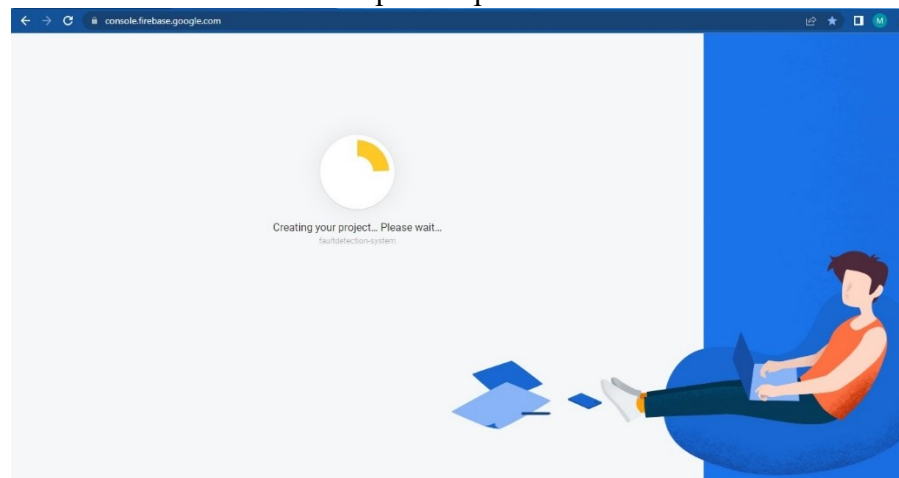


Klik Continue.

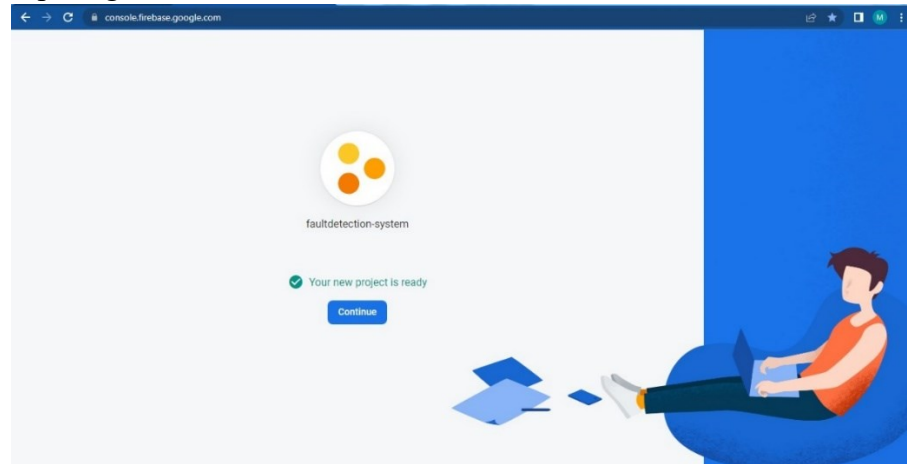
- e. Setelah itu akan muncul tampilan seperti berikut ini. Set bagian Choose and Create a Google Analytics Account menjadi Default Account for Firebase. Lalu klik Create Project.



- f. Setelah itu akan muncul tampilan seperti di bawah ini.

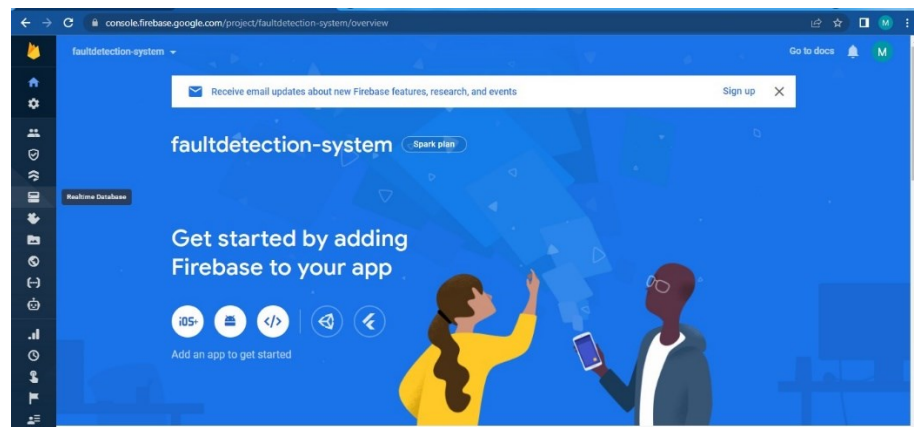


- g. Silahkan tunggu hingga sukses 100%. Lalu akan muncul tampilan seperti gambar berikut.

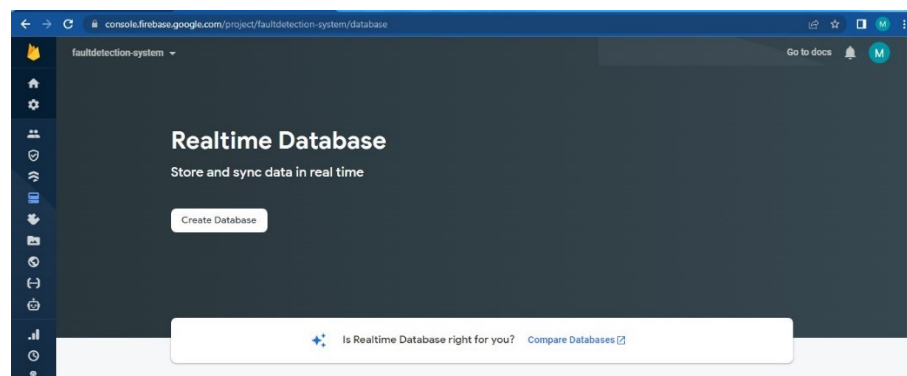


Bagian tersebut menandakan project telah berhasil dibuat. Silahkan klik Continue.

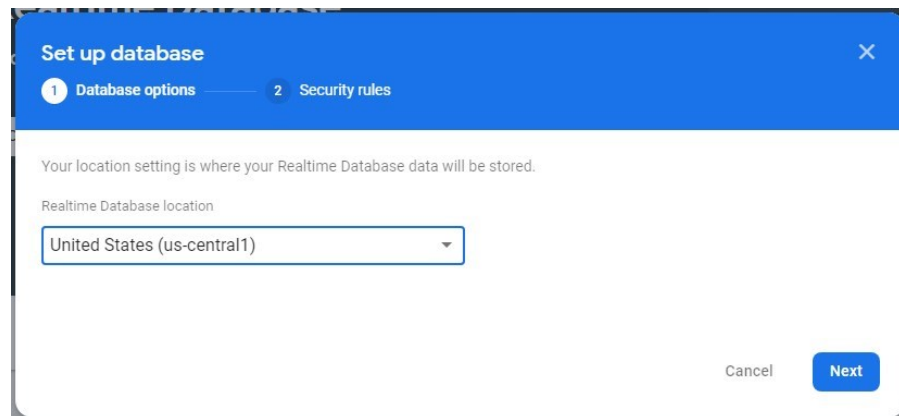
- h. Setelah mengklik Continue maka akan muncul gambar seperti berikut ini.



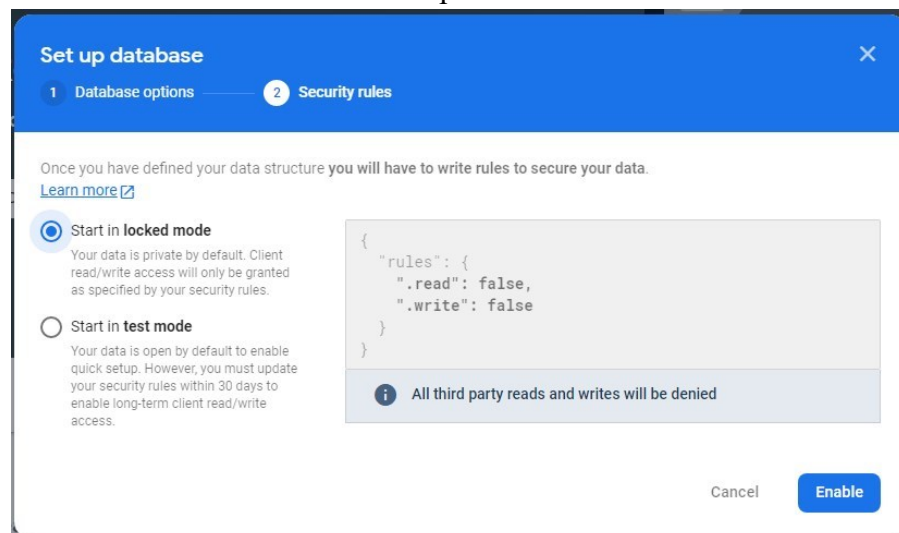
Pada bagian ini silahkan memilih dan klik icon Realtime Database. Setelah itu akan muncul gambar seperti gambar berikut.



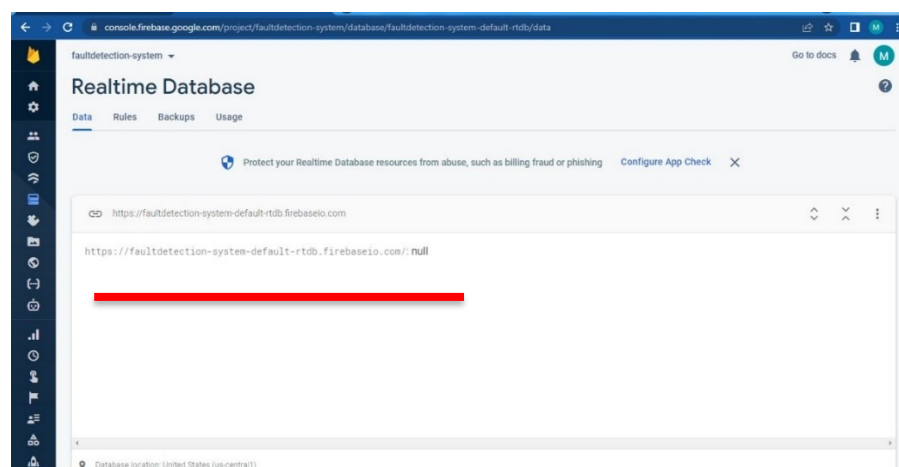
Setelah itu klik Create Database, lalu akan muncul lagi seperti berikut.



Pilih dan klik United States (us-central1), lalu klik Next. Setelah itu akan dimunculkan Step Database.

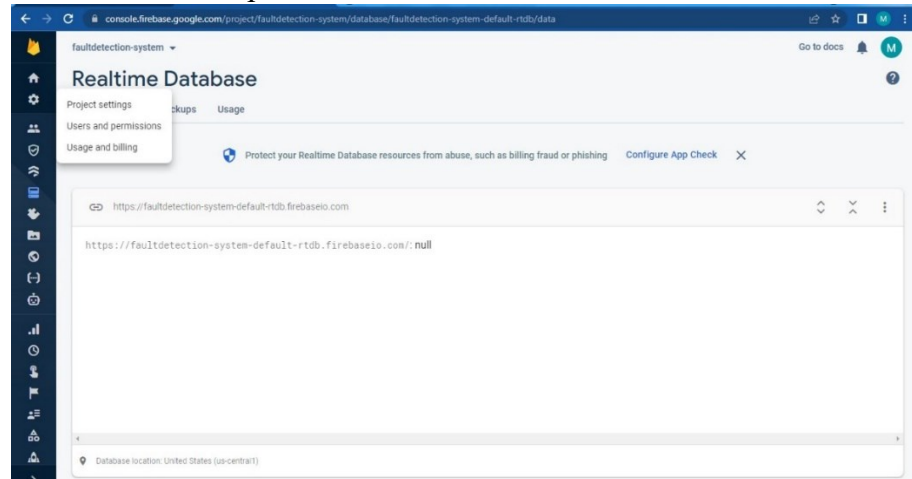


Pada bagian pilih Start in locked mode lalu klik Enable. Setelah itu akan muncul database baru yang masih kosong dan siap untuk digunakan. Terlihat pada gambar berikut ini.

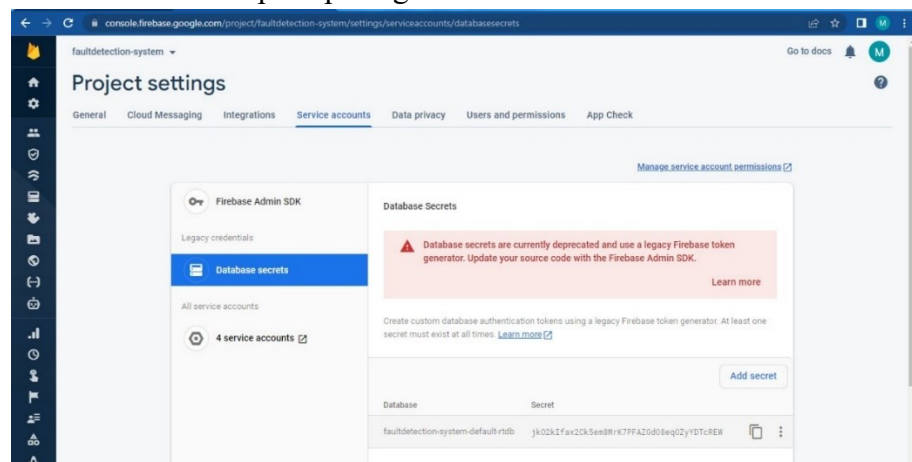


Bagian yang ditandai warna merah merupakan Firebase Host yang akan diinputkan pada program arduino untuk proses posting data pada firebase.

- i. Sedangkan untuk mendapatkan nomor unik atau disebut Firebase Authentication, dapat dilakukan melalui cara berikut ini.

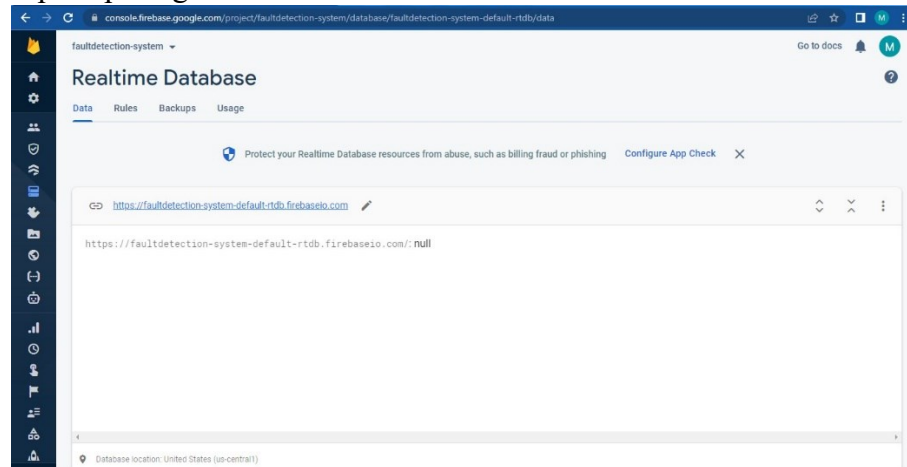


Pada bagian Project klik Project Setting, lalu pilih dan klik menu Service Account seperti pada gambar berikut ini.



Pada bagian Service Account klik Database ~~Secret~~ maka ~~kode Auth~~ akan dimunculkan seperti yang telah ditandai warna merah. Kode tersebut akan diinputkan pada bagian Firebase Auth di aplikasi mikrokontroler.

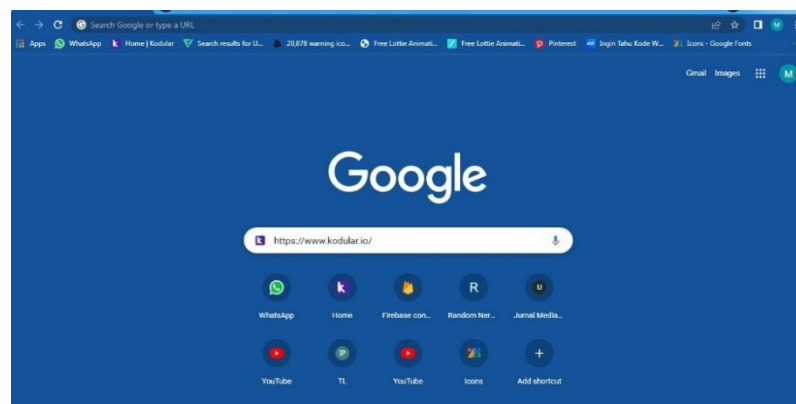
- j. Setelah melakukan langkah-langkah tersebut, maka akan kembali pada layar utama database yang baru saja dibuat dan siap untuk digunakan seperti pada gambar berikut ini.



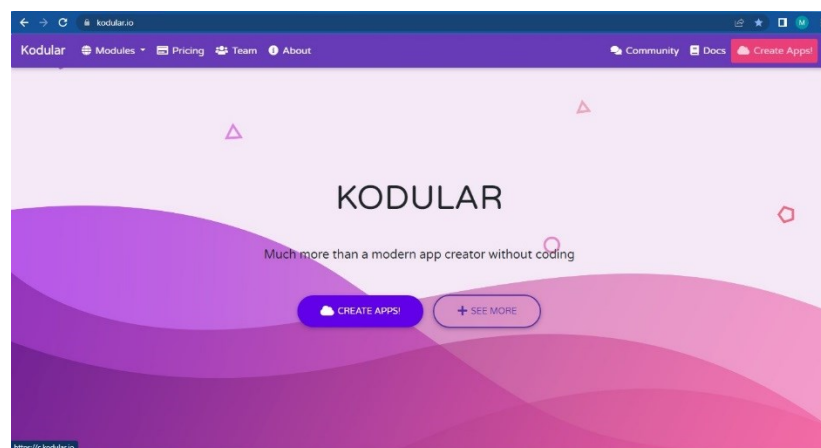
3. Pembuatan Aplikasi Android

Aplikasi yang digunakan pada penelitian ini dibuat menggunakan salah satu website online yang disebut Kodular. Kodular merupakan situs web yang menyediakan tools yang menyerupai MIT App Inventor untuk membuat aplikasi android dengan menggunakan block programming. Adapun software yang digunakan dalam membuat aplikasi yaitu google chrome, website kodular di “<https://www.kodular.io/>“, dan kodular companion untuk testing aplikasi di handphone android. Berikut langkah-langkahnya.

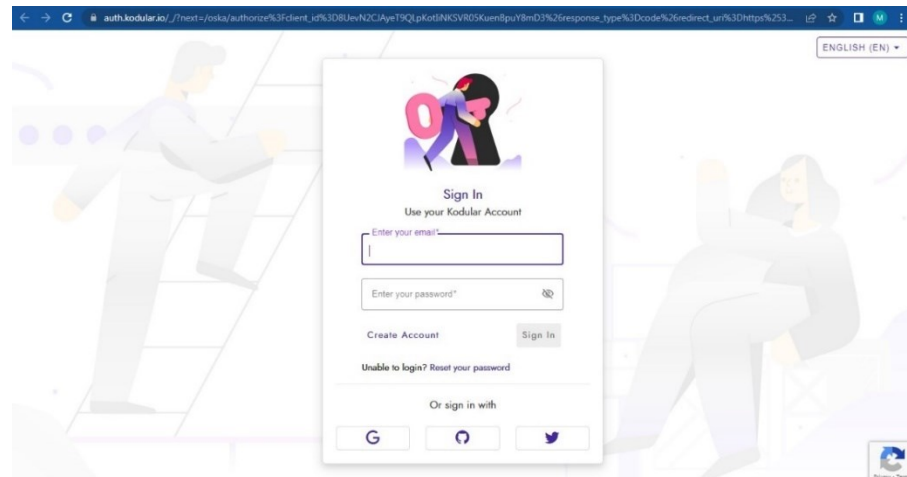
- k. Buka aplikasi google chrome (atau sejenisnya), pastikan komputer terkoneksi dengan internet (online).



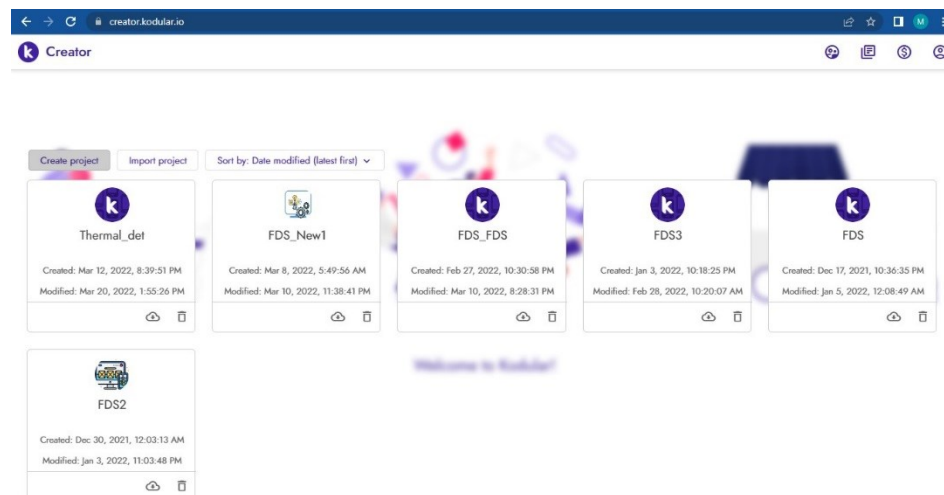
- l. Buka laman <https://www.kodular.io/>, seperti gambar berikut ini. Lalu klik Create Apps.



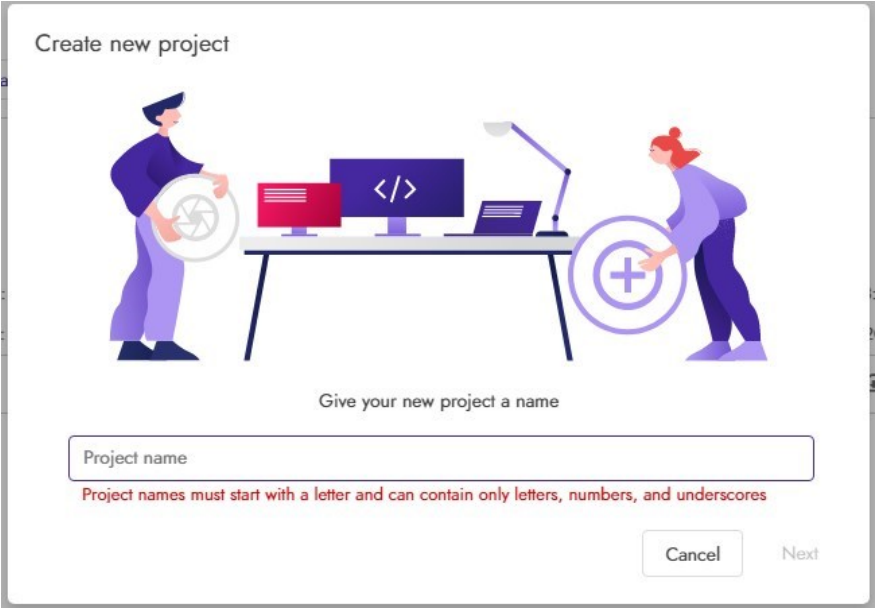
- m. Setelah mengklik Create Apps, maka akan muncul tampilan seperti berikut ini. Pada bagian ini terlebih dahulu harus membuat akun dengan mengklik Create Account lalu Sign in menggunakan akun yang telah dibuat.



n. Jika berhasil Sign in maka akan muncul tampilan seperti berikut ini.



Untuk membuat project baru, klik “Create project” maka akan muncul tampilan seperti berikut ini.



Create new project

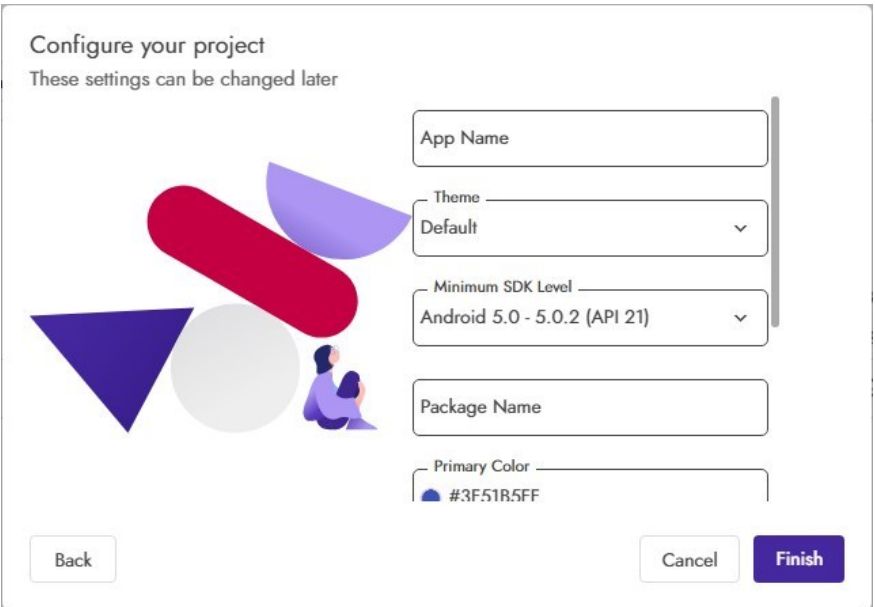
Give your new project a name

Project name

Project names must start with a letter and can contain only letters, numbers, and underscores

Cancel Next

Ketikkan nama project, kemudian klik “Next”. Setelah itu akan muncul lagi tampilan seperti gambar berikut ini.



Configure your project

These settings can be changed later

App Name

Theme
Default

Minimum SDK Level
Android 5.0 - 5.0.2 (API 21)

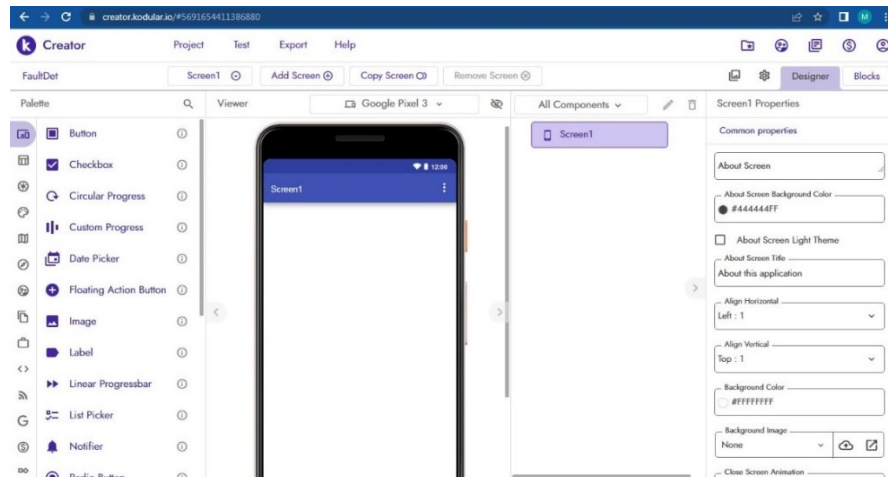
Package Name

Primary Color
#3F51B5FF

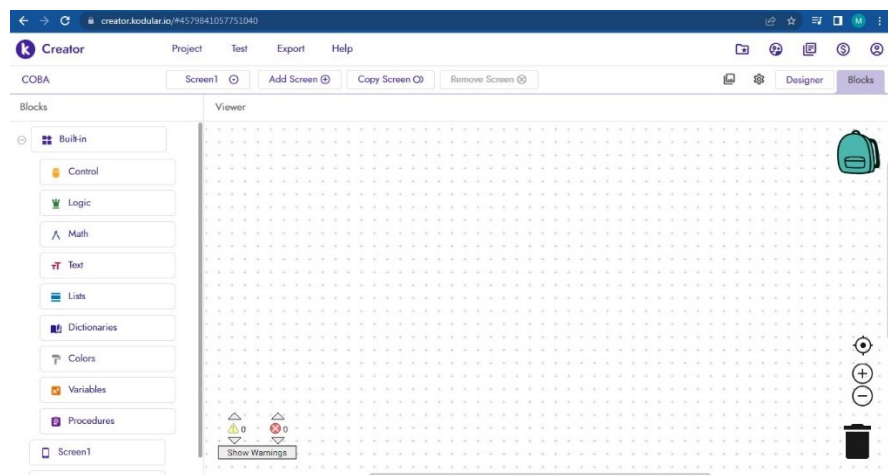
Back Cancel Finish

Pada bagian SDK level, silahkan tentukan minimal versi android yang dapat menggunakan aplikasi tersebut nantinya. Setelah itu klik “Finish”.

Setelah mengklik “Finish” maka akan terbuka lembar kerja seperti gambar berikut ini.

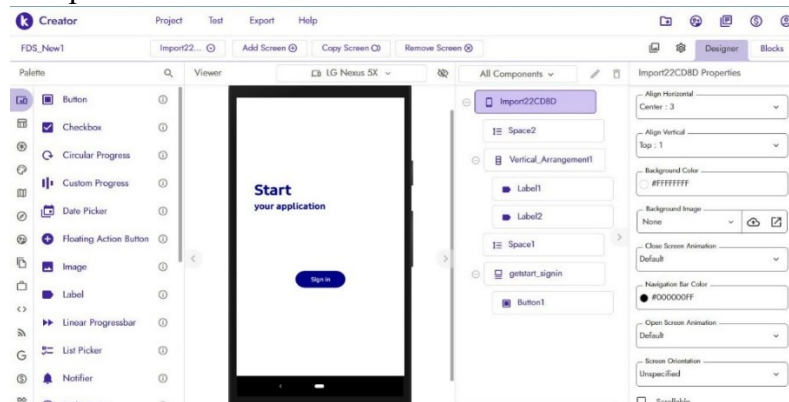


Pada lembar kerja project tersebut akan didesain tampilan aplikasi maupun coding. Gambar di atas menunjukkan lembar kerja bagian “Designer”, selain itu juga terdapat lembar kerja untuk codingan program atau “Blocks” seperti yang terlihat pada gambar berikut ini.

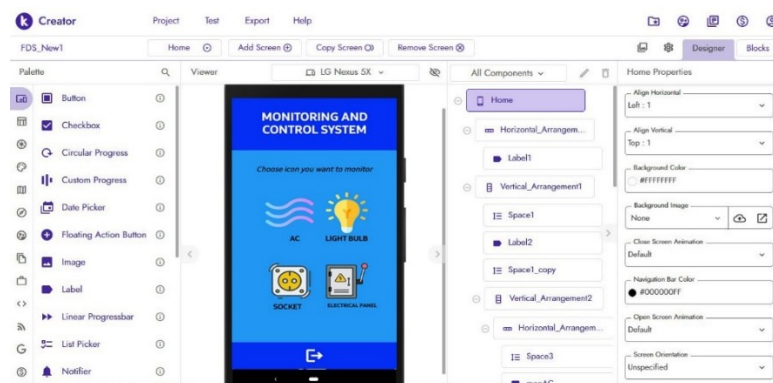


Berikut ini akan diperlihatkan dan dijelaskan mengenai desain dan codingan blok dari aplikasi android yang telah dibuat.

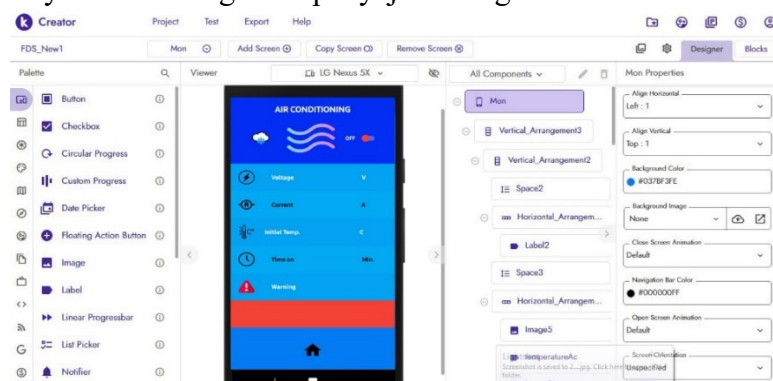
1) Tampilan awal.



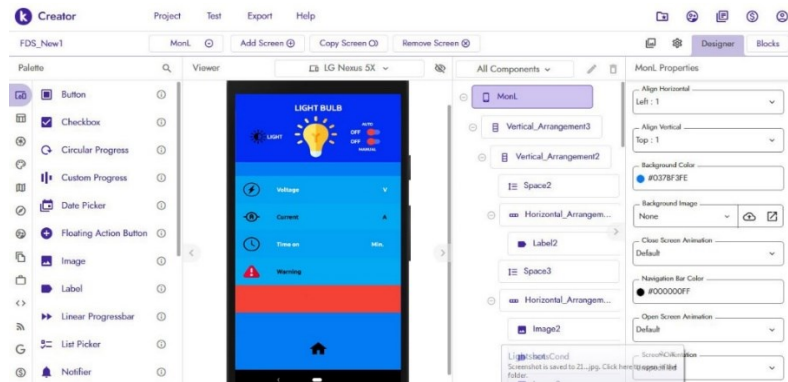
2) Layar utama / Home



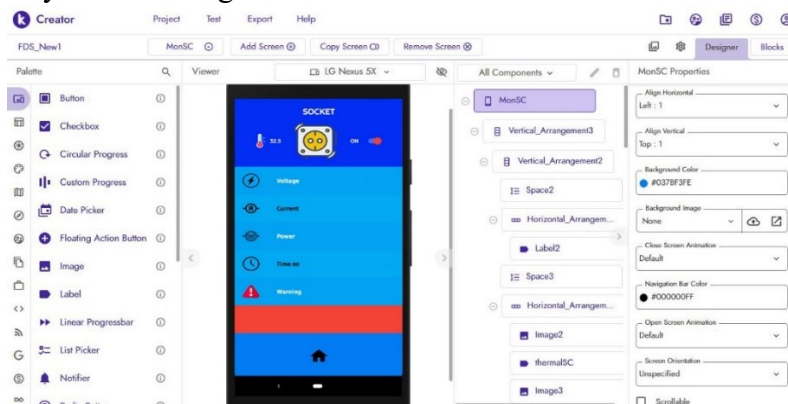
3) Layar monitoring AC / penyejuk ruangan.



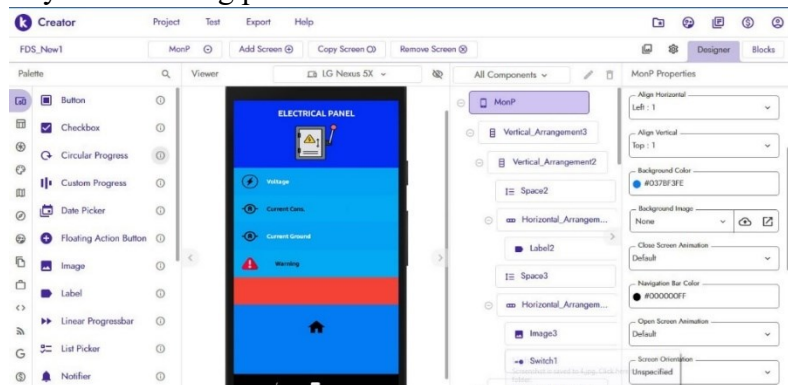
4) Layar monitoring lampu



5) Layer monitoring stok kontak.

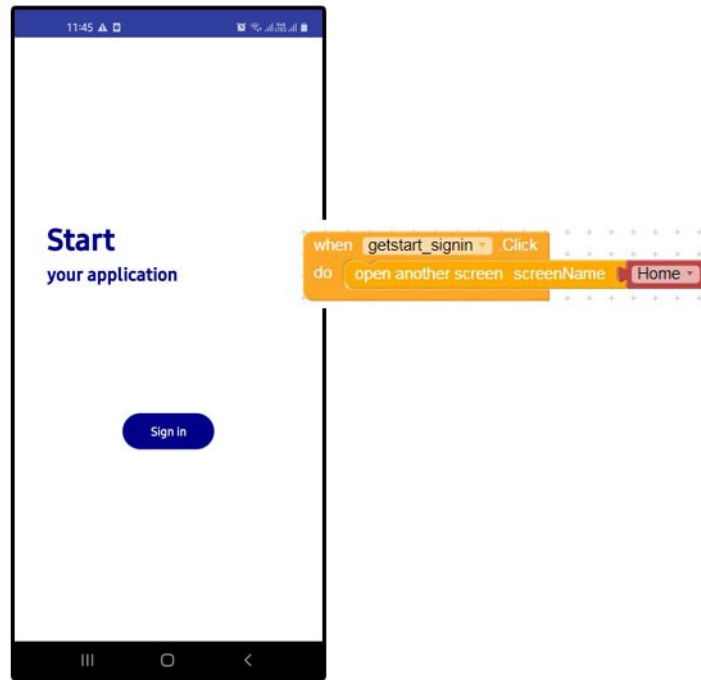


6) Layer monitoring panel.



Berikut ini akan diperlihatkan codingan blok dari aplikasi android yang telah dibuat.

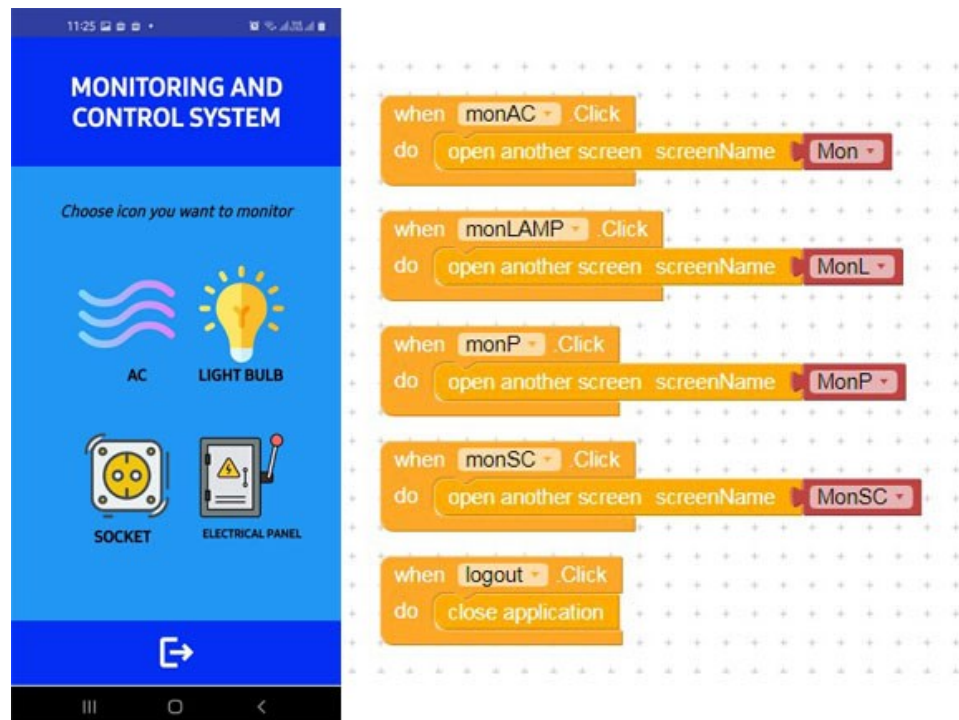
1) Tampilan awal



Penjelasan:

Saat tombol “Sign in” ditekan maka akan diarahkan ke screen “Home”.

2) Layar utama / home.



Penjelasan:

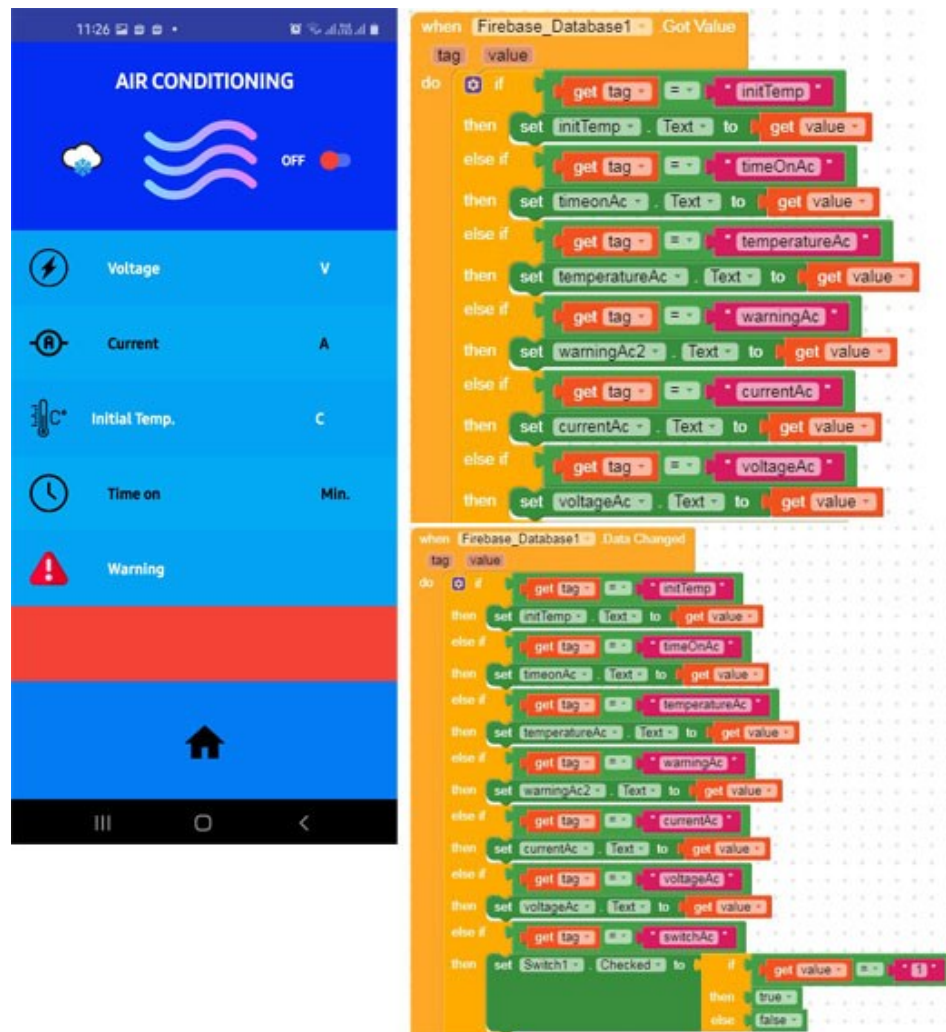
- Jika menekan icon AC maka akan diarahkan ke screen AC
- Jika menekan icon Ligh bulb maka akan diarahkan ke screen Ligh Bulb
- Jika menekan icon Socket maka akan diarahkan ke screen Socket
- Jika menekan icon Electrical Panel maka akan diarahkan ke screen Electrical Panel
- Jika menekan icon Close maka aplikasi akan ditutup.

3) Layar monitoring AC



Penjelasan:

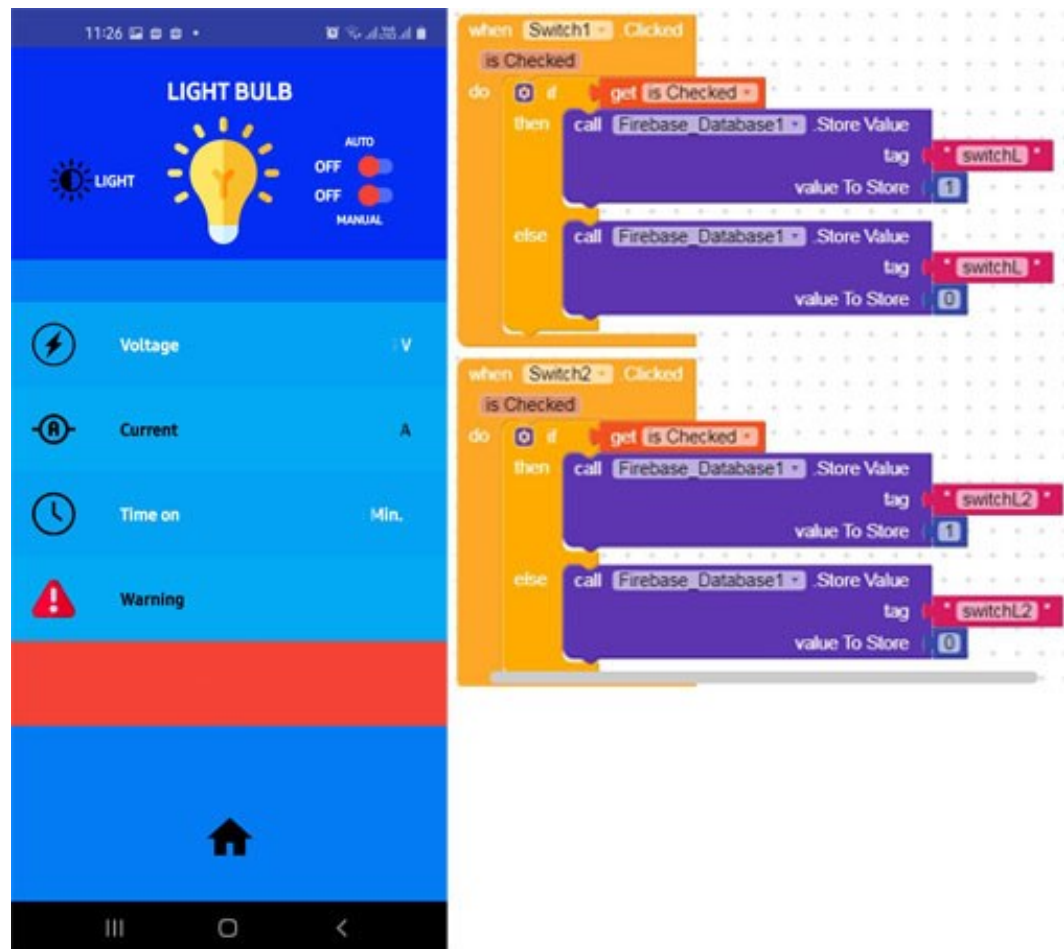
- Jika switch AC ditekan ke ON, maka sistem AC pada hardware akan diaktifkan.
- Jika switch AC ditekan ke OFF, maka sistem AC pada hardware akan dinonaktifkan.
- Jika icon Home ditekan, maka tampilan akan diarahkan ke layar utama / home.
- Pada saat inisialisasi screen, aplikasi akan mengambil data nilai (get value) dari firebase, yang berupa voltage, current, initial temp, time on, warning, dan data switch.



Penjelasan:

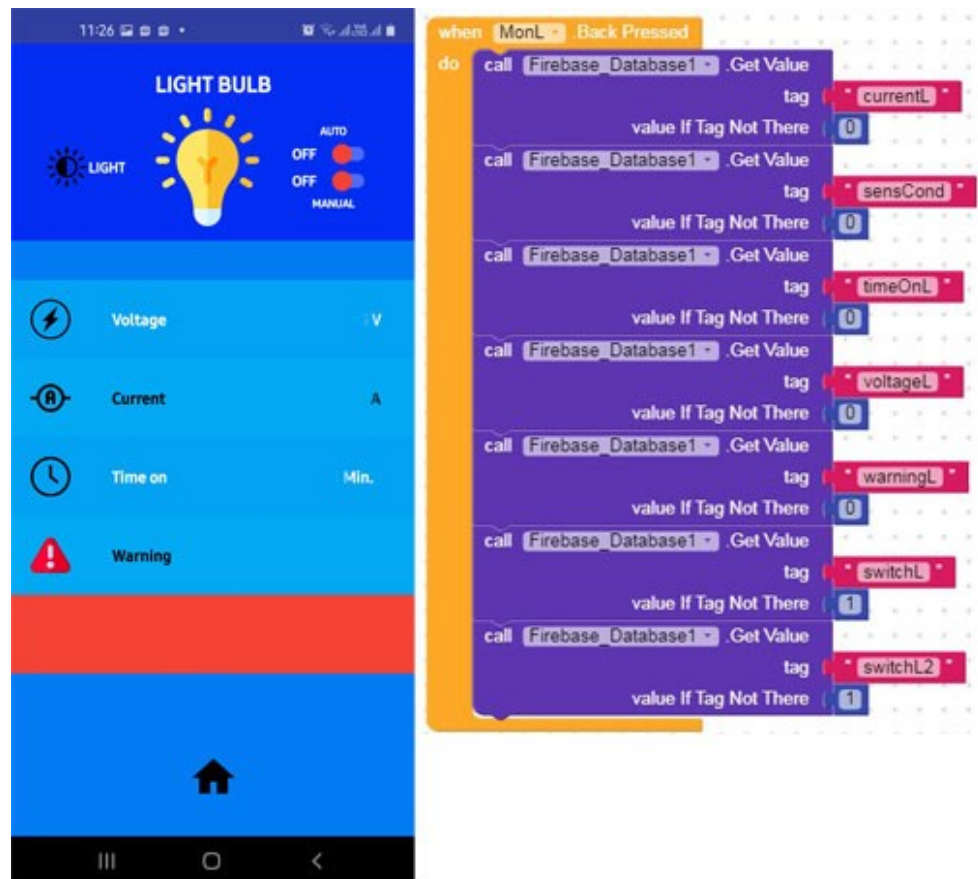
Setelah semua data telah didapat (got value) dari firebase maka aplikasi akan mengganti data (data change) pada tampilan monitoring AC.

4) Layar monitoring lampu.



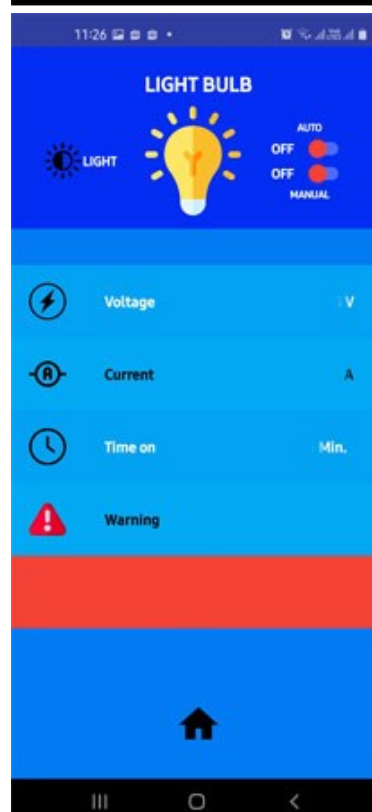
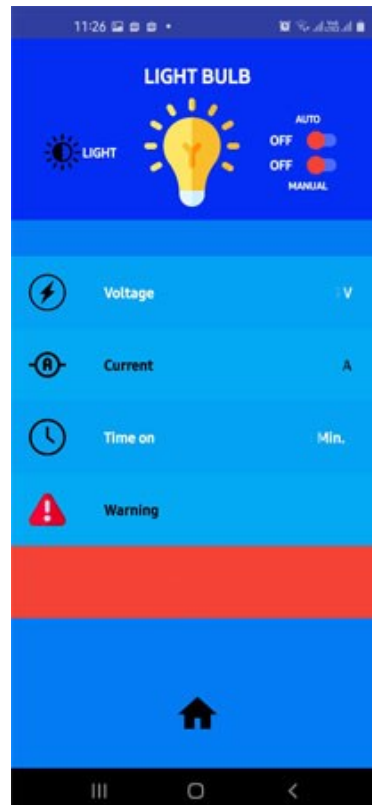
Penjelasan:

- Jika switch AUTO ditekan ke ON, maka sistem lampu pada hardware akan diaktifkan.
- Jika switch AUTO ditekan ke OFF, maka sistem lampu pada hardware akan dinonaktifkan.
- Jika switch MANUAL ditekan ke ON, maka lampu manual pada hardware akan diaktifkan.
- Jika switch MANUAL ditekan ke OFF, maka sistem lampu manual pada hardware akan dinonaktifkan.
- Jika icon Home ditekan, maka tampilan akan diarahkan ke layar utama / home.



Penjelasan:

Pada saat inisialisasi screen, aplikasi akan mengambil data nilai (get value) dari firebase, yang berupa voltage, current, time on, warning, dan data switch.



```

when Firebase_Database1 Got Value
  tag value
do
  if
    get tag == currentL
  then
    set currentL . Text to get value
  else if
    get tag == sensCond
  then
    set sensCond . Text to get value
  else if
    get tag == timeOnL
  then
    set timeOnL . Text to get value
  else if
    get tag == voltageL
  then
    set voltageL . Text to get value
  else if
    get tag == warningL
  then
    set warningL2 . Text to get value

```

```

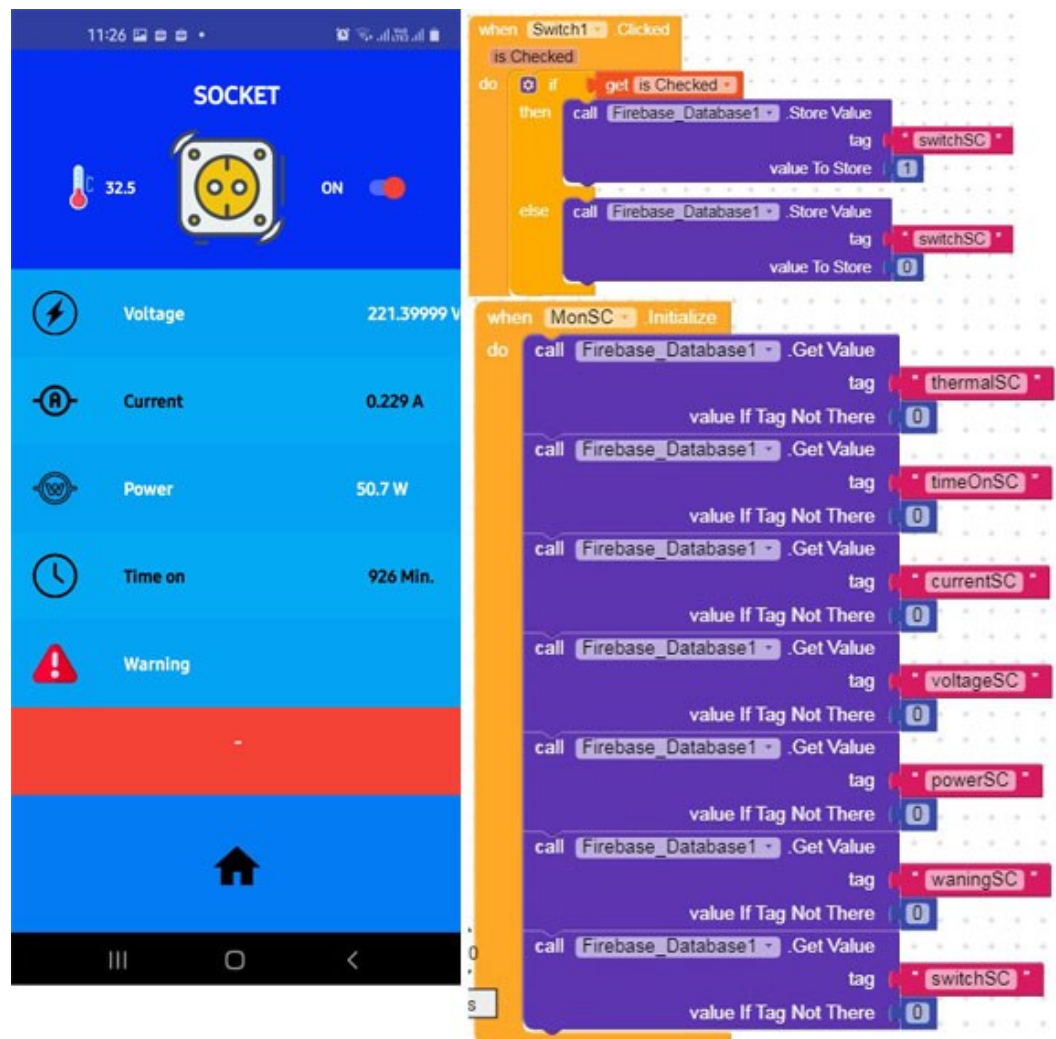
when Firebase_Database1 Data Changed
  tag value
do
  if
    get tag == currentL
  then
    set currentL . Text to get value
  else if
    get tag == sensCond
  then
    set sensCond . Text to get value
  else if
    get tag == timeOnL
  then
    set timeOnL . Text to get value
  else if
    get tag == voltageL
  then
    set voltageL . Text to get value
  else if
    get tag == warningL
  then
    set warningL2 . Text to get value
  else if
    get tag == switchL
  then
    set Switch1 . Checked to if
      get value == 1
    then true
    else false
  else if
    get tag == switchL2
  then
    set Switch2 . Checked to if
      get value == 1
    then true
    else false

```

Penjelasan:

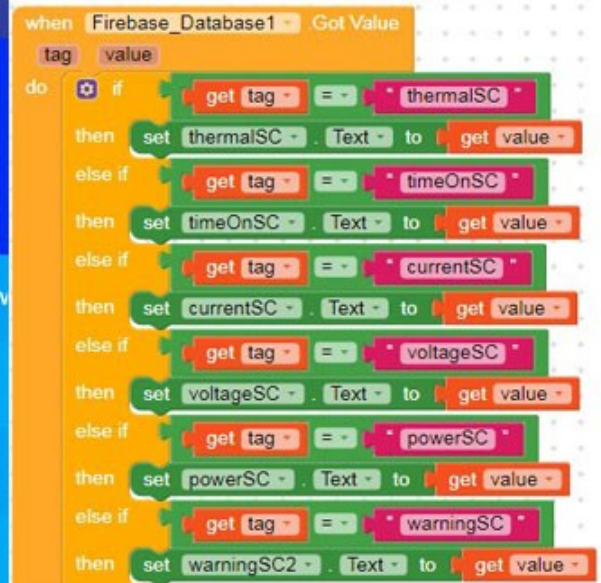
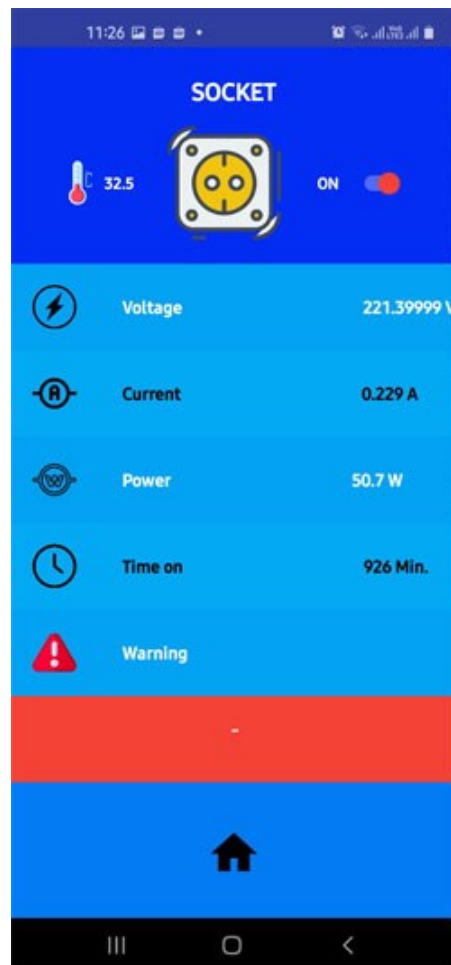
Setelah semua data telah didapat (got value) dari firebase maka aplikasi akan mengganti data (data change) pada tampilan monitoring lampu.

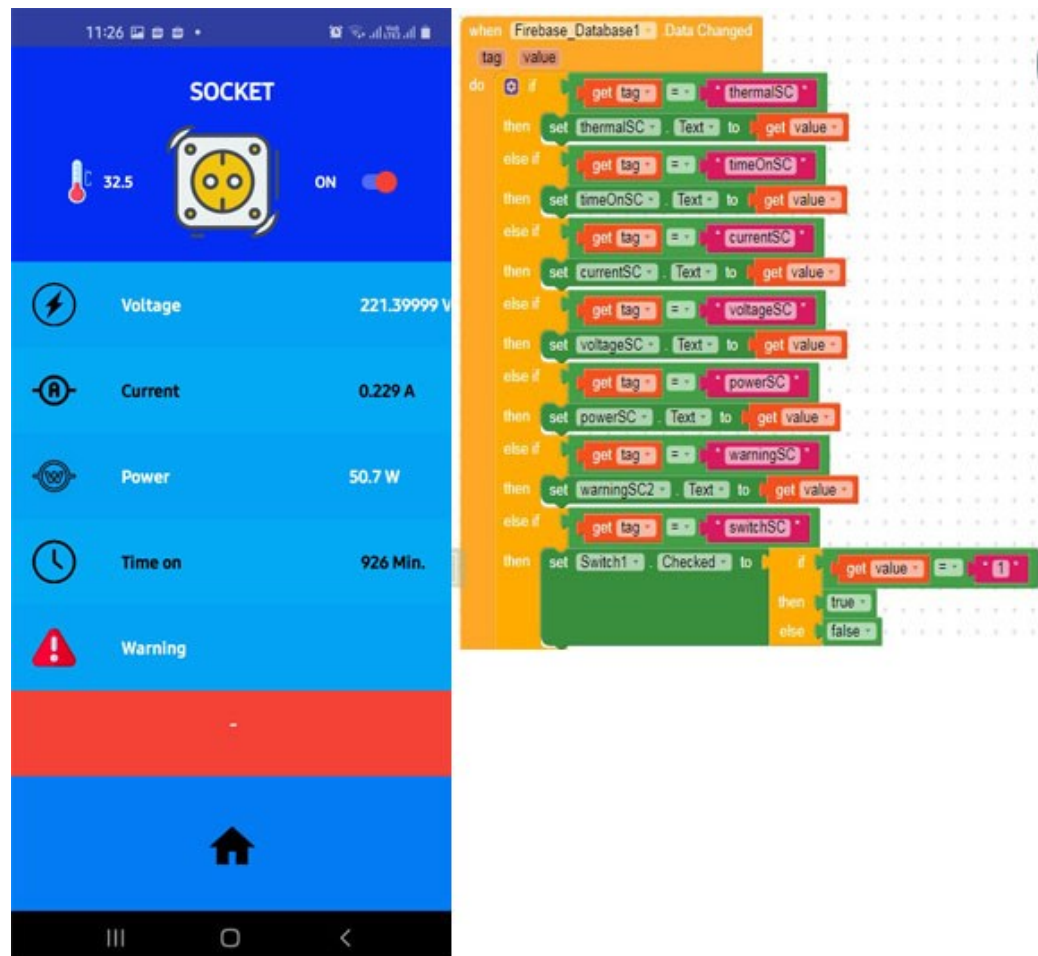
5) Layar monitor stok kontak



Penjelasan:

- Jika switch Socket ditekan ke ON, maka sistem Socket pada hardware akan diaktifkan.
- Jika switch Socket ditekan ke OFF, maka sistem Socket pada hardware akan dinonaktifkan.
- Jika icon Home ditekan, maka tampilan akan diarahkan ke layar utama / home.
- Pada saat inisialisasi screen, aplikasi akan mengambil data nilai (get value) dari firebase, yang berupa voltage, current, power, time on, warning, dan data switch.

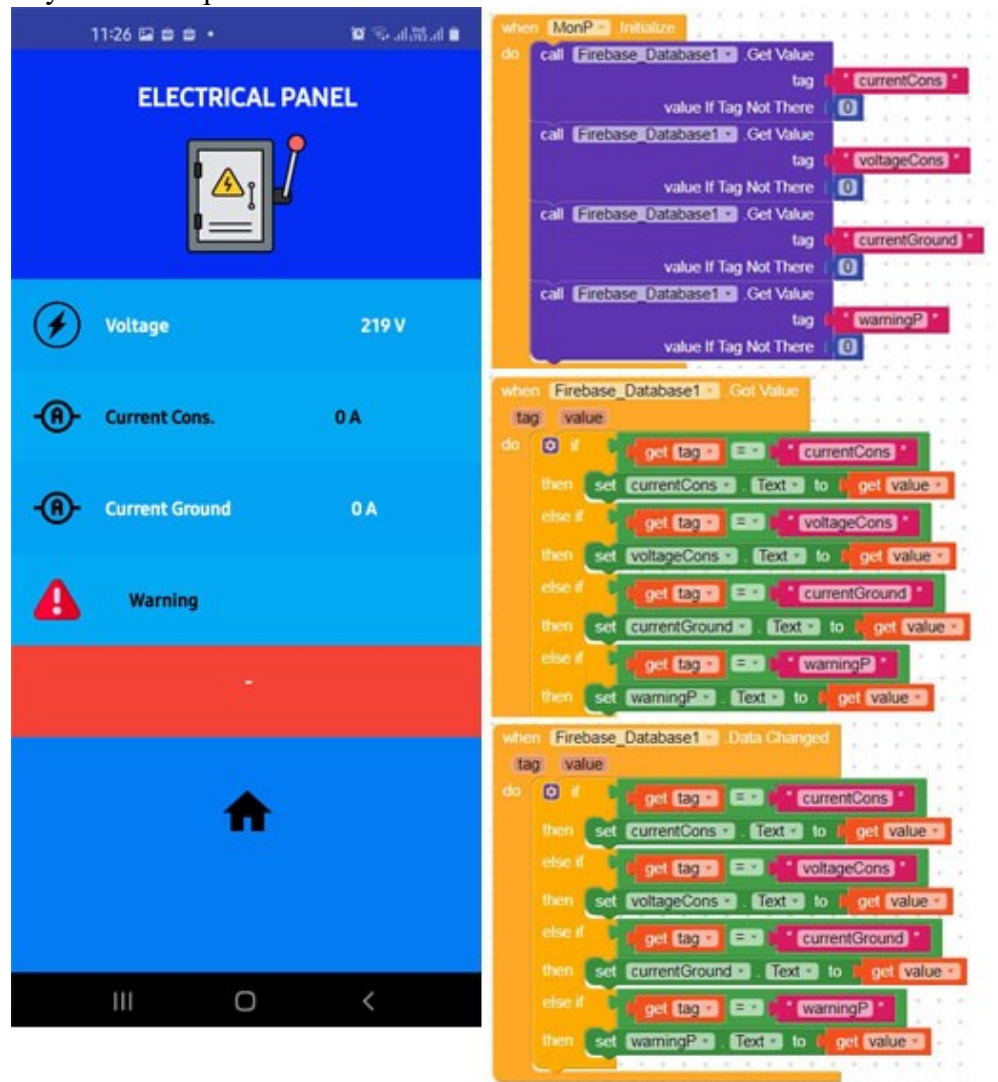




Pernjelasan:

Setelah semua data telah didapat (got value) dari firebase maka aplikasi akan mengganti data (data change) pada tampilan monitoring stok kontak.

6) Layar monitor panel.



Penjelasan:

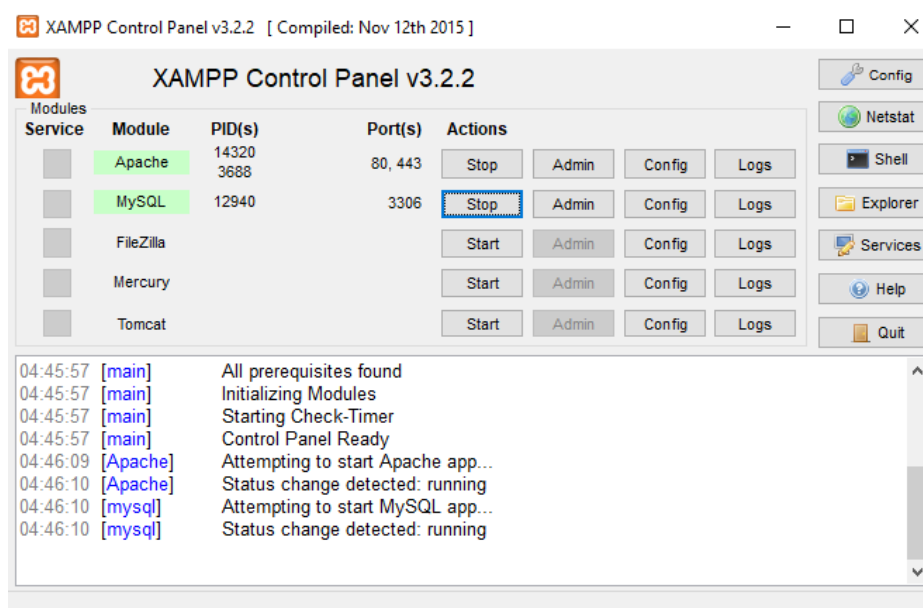
- Jika icon Home ditekan, maka tampilan akan diarahkan ke layar utama / home.
- Pada saat inisialisasi screen, aplikasi akan mengambil data nilai (get value) dari firebase, yang berupa voltage, current consumption, current grounding, dan warning.
- Setelah semua data telah didapat (got value) dari firebase maka aplikasi akan mengganti data (data change) pada tampilan monitoring panel.

4. Pembuatan Aplikasi Website

Aplikasi website yang digunakan pada penelitian ini dibuat dengan menggunakan bahasa pemrograman PHP dengan framework Codeigniter 3.1.4. CodeIgniter merupakan aplikasi sumber terbuka yang berupa kerangka kerja PHP dengan model MVC untuk membangun situs web dinamis dengan menggunakan PHP. CodeIgniter memudahkan pengembang web untuk membuat aplikasi web dengan cepat dan mudah dibandingkan dengan membuatnya dari awal. Berikut langkah langkah pembuatan aplikasi website

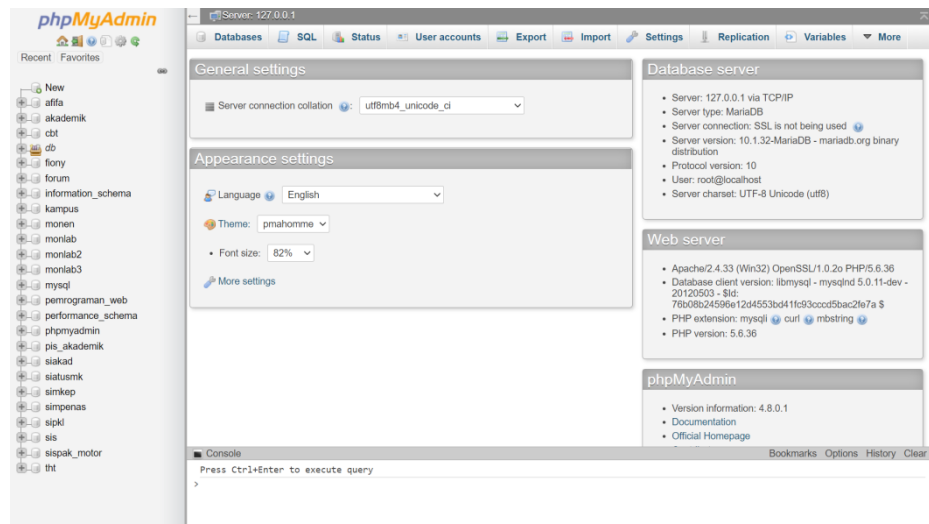
a. Menginstal *Tools* yang Diperlukan

hal ini mencakup sever web Apache, SQL, Phpmyadmin, web browser, text editor dan Codeingiter. Server web Apache, SQL dan phpmyadmin yang dibutuhkan bisa didapatkan dengan menginstal XAMPP, dalam penelitian ini digunakan XAMPP 3.2.2.



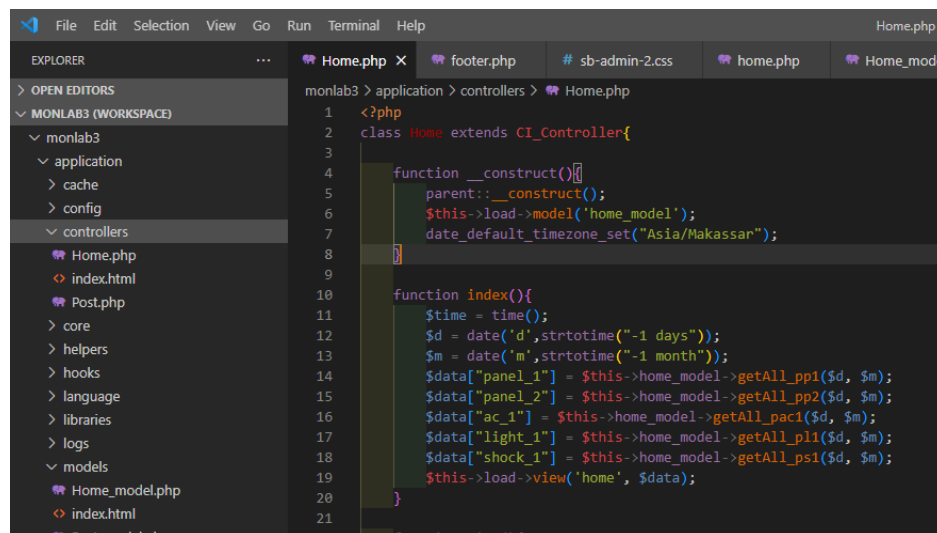
b. Perancang dan Pembuatan Database

pembuatan dan pengelolaan database dapat dilakukan dengan terlebih dahulu menjalankan Apache dan MySQL pada control panel XAMPP kemudian mengakses phpmyadmin dengan mengakses url pada browser <http://localhost/phpmyadmin>



c. Pembuatan *Controller* dan *Routing* Website

Controller merupakan bagian yang menjembatani *model* dan *view*. *Controller* berisi perintah-perintah yang berfungsi untuk memproses suatu data dan mengirimkannya ke halaman web. Pembuatan *controller* dapat dilakukan pada folder `folder_projek/application/controller`, pada folder ini dilakukan pembuatan file *controller*. Aplikasi web ini menggunakan dua *controller* yaitu `Home.php` sebagai *controller view* aplikasi dan `Post.php` sebagai *controller* pengiriman data dari ESP8266



Router pada Codeigniter bertugas untuk menentukan *controller* dan *method/fungsi* yang akan dieksekusi, pembuatan *route* dapat dilakukan pada file di dalam folder `folder_projek/application/config/routes.php`

```

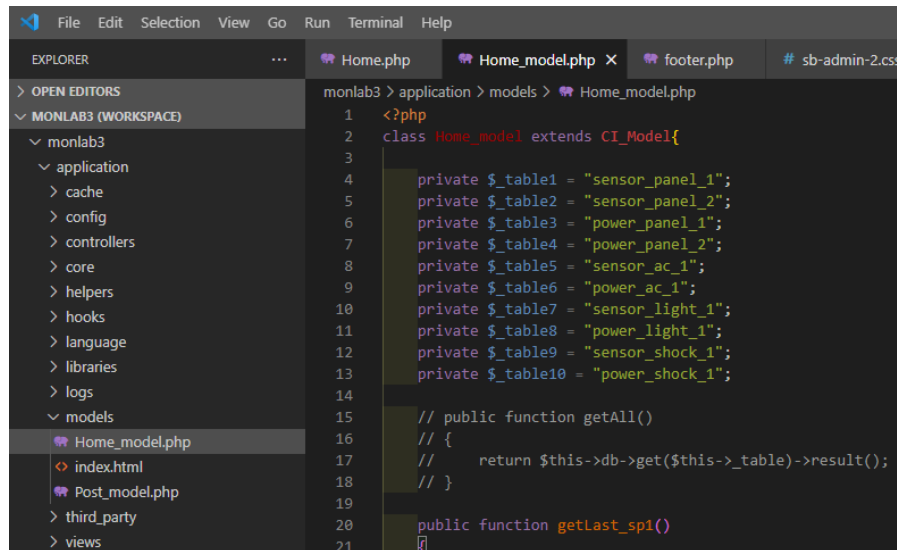
monlab3 > application > config > routes.php
48 |
49 | Examples: my-controller/index -> my_control
50 |           my-controller/my-method -> my_control
51 | */
52 | $route['default_controller'] = 'home';
53 | $route['404_override'] = '';
54 | $route['translate_uri_dashes'] = FALSE;
55 |
56 | $route['post'] = 'post';
57 | $route['home'] = 'home';
58 |

```

d. Pembuatan *Model* (CRUD)

Model mewakili struktur data, *model* merupakan bagian yang bertugas untuk mengatur, menyiapkan, memanipulasi, dan mengorganisir data (biasanya dari basis data). Tugas yang ia lakukan meliputi memasukkan data ke basis data, pembaruan data, menghapus data, dan lain-lain. *Model* menjalankan tugasnya berdasarkan instruksi dari *controller*. Pembuatan model dapat dilakukan di dalam folder folder_projek/application/model. Aplikasi web ini menggunakan dua *model*, *Home_model.php* menangani pengelolaan data untuk tampilan aplikasi dan *Post_model.php* menangani pengelolaan data dari

ESP8266



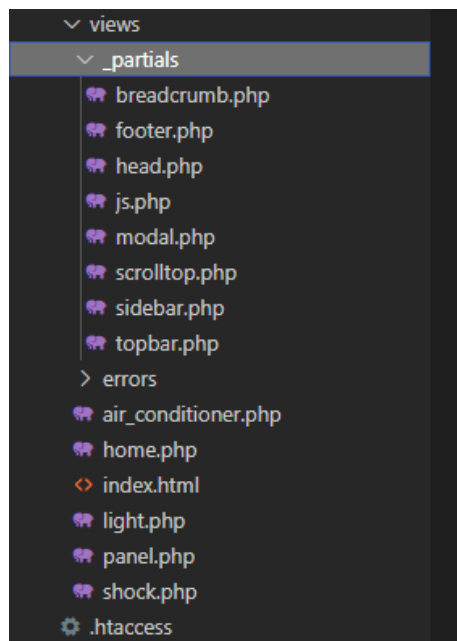
```

1  <?php
2  class Home_model extends CI_Model{
3
4      private $_table1 = "sensor_panel_1";
5      private $_table2 = "sensor_panel_2";
6      private $_table3 = "power_panel_1";
7      private $_table4 = "power_panel_2";
8      private $_table5 = "sensor_ac_1";
9      private $_table6 = "power_ac_1";
10     private $_table7 = "sensor_light_1";
11     private $_table8 = "power_light_1";
12     private $_table9 = "sensor_shock_1";
13     private $_table10 = "power_shock_1";
14
15     // public function getAll()
16     // {
17     //     return $this->db->get($this->_table)->result();
18     // }
19
20     public function getLast_sp1()
21

```

e. **Pembuatan View**

View merupakan bagian yang mengatur tampilan ke pengguna. Bisa dikatakan berupa halaman web. Pembuatan *view* dapat dilakukan di dalam folder `folder_projek/application/view`



f. **Hosting Aplikasi Web**

Setelah pembuatan aplikasi web selesai dan dapat berjalan di dalam server lokal yang dibuat, selanjutnya adalah melakukan *hosting* secara online pada aplikasi web yang dilakukan dengan mengupload seluruh folder/file project pada hoster/ server online, sehingga aplikasi web dapat diankses secara online

B. Hasil Pengujian MACSys

Berikut diperlihatkan tabel hasil penerimaan web server dari mikrokontroller sistem.

1. Modul ACSS

Modul AC ini menggunakan sensor arus PZEM-004T dan sensor temperatur DHT22 yang menjadi alat untuk monitoring sekaligus membandingkan data untuk dijadikan acuan tindakan otomatisasi pada sistem. Berikut diperlihatkan tabel hasil uji coba modul AC dengan kondisi pendingin ruangan normal dan tidak normal. Uji coba ini merupakan uji coba terbatas menggunakan beban 20 Watt dan es batu sebagai pendingin.

Tabel 1. Percobaan dengan kondisi pendingin ruangan normal.

No	Date	Time	I (A)	V (Volt)	P (W)	PF	Sw	Temp. (C)	Status
1	22/05/2022	19:05:57	0.00	0.00	0.00	0.00	OFF	0.00	Setup system
2	22/05/2022	19:06:03	0.00	0.00	0.00	0.00	OFF	0.00	Setup system
3	22/05/2022	19:06:38	0.00	216.20	0.00	0.00	OFF	32.70	System ON
4	22/05/2022	19:06:42	0.00	216.10	0.00	0.00	OFF	32.70	System ON
5	22/05/2022	19:06:48	0.09	216.10	14.10	0.71	ON	32.70	AC turn ON
6	22/05/2022	19:06:53	0.09	216.60	14.20	0.70	ON	32.70	AC turn ON
7	22/05/2022	19:06:57	0.09	216.60	14.30	0.71	ON	32.70	AC turn ON
8	22/05/2022	19:07:01	0.09	216.80	14.40	0.71	ON	32.70	AC turn ON
9	22/05/2022	19:07:05	0.09	216.60	14.30	0.71	ON	32.70	AC turn ON
10	22/05/2022	19:07:10	0.09	216.60	14.30	0.71	ON	32.70	AC turn ON
11	22/05/2022	19:07:13	0.09	216.60	14.20	0.71	ON	32.70	AC turn ON
12	22/05/2022	19:07:18	0.09	216.60	14.20	0.71	ON	32.70	AC turn ON
13	22/05/2022	19:07:22	0.09	216.60	14.10	0.71	ON	32.70	AC turn ON
14	22/05/2022	19:07:27	0.09	216.50	14.00	0.71	ON	32.70	AC turn ON
15	22/05/2022	19:07:32	0.09	216.60	13.90	0.70	ON	32.70	AC turn ON
16	22/05/2022	19:07:36	0.09	216.50	13.90	0.71	ON	32.70	AC turn ON
17	22/05/2022	19:07:41	0.09	216.70	13.80	0.71	ON	32.70	AC turn ON
18	22/05/2022	19:07:48	0.09	216.40	13.70	0.70	ON	32.80	AC turn ON
19	22/05/2022	19:07:54	0.09	216.40	13.60	0.71	ON	32.80	AC turn ON
20	22/05/2022	19:08:00	0.09	217.00	13.60	0.70	ON	32.80	AC turn ON
21	22/05/2022	19:08:07	0.09	217.00	13.50	0.71	ON	32.70	AC turn ON
22	22/05/2022	19:08:13	0.09	217.00	13.50	0.71	ON	32.70	AC turn ON
23	22/05/2022	19:08:19	0.09	217.10	13.50	0.71	ON	32.70	AC turn ON
24	22/05/2022	19:08:25	0.09	217.20	13.40	0.70	ON	32.80	AC turn ON
25	22/05/2022	19:08:32	0.09	217.20	13.30	0.70	ON	32.80	AC turn ON
26	22/05/2022	19:08:40	0.09	217.20	13.40	0.70	ON	32.80	AC turn ON
27	22/05/2022	19:08:46	0.09	217.10	13.30	0.70	ON	32.70	AC turn ON

28	22/05/2022	19:08:51	0.09	217.30	13.30	0.70	ON	32.70	AC turn ON
29	22/05/2022	19:08:56	0.09	217.30	13.30	0.70	ON	32.80	AC turn ON
30	22/05/2022	19:09:03	0.09	217.20	13.30	0.70	ON	32.80	AC turn ON
31	22/05/2022	19:09:08	0.09	217.40	13.20	0.70	ON	32.80	AC turn ON
32	22/05/2022	19:09:14	0.09	217.40	13.20	0.70	ON	32.80	AC turn ON
33	22/05/2022	19:09:20	0.09	217.50	13.20	0.70	ON	32.70	AC turn ON
34	22/05/2022	19:09:26	0.09	217.80	13.20	0.70	ON	32.80	AC turn ON
35	22/05/2022	19:09:33	0.09	217.60	13.20	0.70	ON	32.70	AC turn ON
36	22/05/2022	19:09:40	0.09	217.90	13.30	0.70	ON	32.80	AC turn ON
37	22/05/2022	19:09:46	0.09	218.00	13.20	0.70	ON	31.70	AC turn ON
38	22/05/2022	19:09:52	0.09	218.00	13.20	0.70	ON	31.70	AC turn ON
39	22/05/2022	19:09:59	0.09	218.10	13.20	0.70	ON	31.70	AC turn ON
40	22/05/2022	19:10:06	0.09	217.90	13.20	0.70	ON	30.90	AC turn ON
41	22/05/2022	19:10:12	0.09	217.50	13.20	0.70	ON	30.80	AC turn ON
42	22/05/2022	19:17:20	0.06	218.10	13.00	0.69	ON	29.40	AC turn ON
43	22/05/2022	19:17:24	0.09	218.10	13.00	0.69	ON	29.50	AC turn ON
44	22/05/2022	19:17:29	0.09	218.20	13.10	0.70	ON	29.50	AC turn ON
45	22/05/2022	19:17:34	0.09	218.10	13.10	0.70	ON	29.50	AC turn ON
46	22/05/2022	19:17:38	0.09	218.00	13.00	0.69	ON	29.40	AC turn ON
47	22/05/2022	19:17:42	0.09	218.00	12.90	0.69	ON	29.40	AC turn ON
48	22/05/2022	19:17:47	0.09	217.80	13.00	0.69	ON	29.40	AC turn ON
49	22/05/2022	19:17:52	0.09	218.10	12.90	0.70	ON	29.40	AC turn ON
50	22/05/2022	19:17:56	0.09	218.00	12.90	0.70	ON	29.40	AC turn ON
51	22/05/2022	19:18:01	0.09	218.10	12.90	0.70	ON	29.40	AC turn ON
52	22/05/2022	19:18:05	0.09	218.10	12.90	0.69	ON	29.30	AC turn ON
53	22/05/2022	19:18:12	0.09	217.50	13.00	0.70	ON	29.30	AC turn ON
54	22/05/2022	19:18:18	0.09	217.40	12.90	0.70	ON	29.30	AC turn ON
55	22/05/2022	19:18:26	0.09	217.60	12.90	0.69	ON	29.30	AC turn ON
56	22/05/2022	19:18:32	0.09	217.60	12.90	0.69	ON	29.30	AC turn ON
57	22/05/2022	19:18:38	0.09	217.70	12.90	0.69	ON	29.30	AC turn ON
58	22/05/2022	19:18:44	0.09	217.70	12.90	0.69	ON	29.30	AC turn ON
59	22/05/2022	19:18:50	0.09	217.60	12.90	0.69	ON	29.30	AC turn ON
60	22/05/2022	19:18:58	0.09	217.80	12.90	0.69	ON	29.30	AC turn ON
61	22/05/2022	19:19:06	0.09	217.60	12.80	0.69	ON	29.30	AC turn ON
62	22/05/2022	19:19:13	0.09	217.10	12.90	0.69	ON	29.30	AC turn ON
63	22/05/2022	19:19:22	0.09	217.30	12.90	0.69	ON	29.30	AC turn ON
64	22/05/2022	19:19:31	0.09	217.80	12.90	0.70	ON	29.30	AC turn ON
65	22/05/2022	19:19:37	0.09	217.90	12.80	0.69	ON	29.30	AC turn ON
66	22/05/2022	19:19:45	0.09	217.80	12.90	0.70	ON	29.20	AC turn ON
67	22/05/2022	19:19:52	0.09	217.80	12.80	0.69	ON	29.20	AC turn ON
68	22/05/2022	19:20:00	0.09	218.30	12.90	0.69	ON	29.10	AC turn ON
69	22/05/2022	19:20:06	0.09	218.60	12.90	0.69	ON	29.10	AC turn ON
70	22/05/2022	19:20:15	0.09	218.40	12.80	0.69	ON	29.10	AC turn ON

71	22/05/2022	19:20:21	0.09	218.30	12.90	0.69	ON	29.20	AC turn ON
72	22/05/2022	19:20:28	0.09	218.70	12.90	0.69	ON	29.20	AC turn ON
73	22/05/2022	19:20:38	0.09	218.70	12.90	0.69	ON	29.10	AC turn ON
74	22/05/2022	19:20:44	0.09	218.40	12.90	0.69	ON	29.20	AC turn ON
75	22/05/2022	19:20:51	0.09	218.20	12.90	0.69	ON	29.20	AC turn ON
76	22/05/2022	19:20:59	0.09	218.50	12.90	0.69	ON	29.20	AC turn ON
77	22/05/2022	19:21:06	0.09	218.30	12.90	0.70	ON	29.30	AC turn ON
78	22/05/2022	19:21:12	0.09	218.20	12.90	0.70	ON	29.30	AC turn ON
79	22/05/2022	19:21:18	0.09	218.20	12.80	0.69	ON	29.20	AC turn ON
80	22/05/2022	19:21:27	0.09	218.40	12.90	0.69	ON	29.20	AC turn ON

Berdasarkan tabel hasil uji, sesaat setelah sistem ON (baris ke 5) menunjukkan pendingin ruangan telah dinyalakan oleh sistem, pada pukul 19:06:42 dengan kondisi temperatur ruangan 32.70

4	22/05/2022	19:06:42	0.00	216.10	0.00	0.00	OFF	32.70	System ON
5	22/05/2022	19:06:48	0.09	216.10	14.10	0.71	ON	32.70	AC turn ON

Gambar 1. Kondisi pendingin ruangan dinyalakan.

Setelah sekitar 3 menit yaitu pukul 19:09:46, temperatur ruangan berubah menjadi 31.70 derajat celcius, hal ini dianggap pendingin ruangan dalam keadaan normal karena temperatur ruangan berubah menjadi lebih rendah sebelum 5 menit. Oleh karena itu penyalan pendingin ruangan dilanjutkan.

36	22/05/2022	19:09:40	0.09	217.90	13.30	0.70	ON	32.80	AC turn ON
37	22/05/2022	19:09:46	0.09	218.00	13.20	0.70	ON	31.70	AC turn ON
38	22/05/2022	19:09:52	0.09	218.00	13.20	0.70	ON	31.70	AC turn ON

Gambar 2. Kondisi pendingin ruangan menurunkan temperatur setelah 3 menit.

Tabel 2. Percobaan dengan kondisi pendingin ruangan tidak normal.

No	Date	Time	I (A)	V (Volt)	P (W)	PF	Sw	Temp. (C)	Status
1	22/05/2022	17:36:05	0.00	218.80	0.00	0.00	OFF	33.70	System ON
2	22/05/2022	17:36:09	0.00	218.70	0.00	0.00	OFF	33.70	System ON
3	22/05/2022	17:36:16	0.00	218.60	0.00	0.00	ON	33.70	AC turn ON
4	22/05/2022	17:36:24	0.00	218.50	0.00	0.00	ON	33.80	AC turn ON
5	22/05/2022	17:36:28	0.00	218.80	4.10	0.49	ON	33.80	AC turn ON
6	22/05/2022	17:36:32	0.09	218.80	14.00	0.69	ON	33.70	AC turn ON
7	22/05/2022	17:36:36	0.09	218.60	14.30	0.70	ON	33.70	AC turn ON
8	22/05/2022	17:36:40	0.09	218.70	14.40	0.69	ON	33.70	AC turn ON

9	22/05/2022	17:36:44	0.09	218.90	14.50	0.70	ON	33.80	AC turn ON
10	22/05/2022	17:36:48	0.10	218.80	14.60	0.70	ON	33.70	AC turn ON
11	22/05/2022	17:36:52	0.10	218.90	14.60	0.69	ON	33.80	AC turn ON
12	22/05/2022	17:36:56	0.10	218.80	14.60	0.70	ON	33.70	AC turn ON
13	22/05/2022	17:37:01	0.10	218.70	14.60	0.70	ON	33.80	AC turn ON
14	22/05/2022	17:37:06	0.09	218.50	14.50	0.70	ON	33.70	AC turn ON
15	22/05/2022	17:37:13	0.09	218.30	14.40	0.69	ON	33.80	AC turn ON
16	22/05/2022	17:37:18	0.09	218.20	14.30	0.69	ON	33.70	AC turn ON
17	22/05/2022	17:37:23	0.09	218.30	14.20	0.69	ON	33.80	AC turn ON
18	22/05/2022	17:37:29	0.09	218.60	14.20	0.69	ON	33.70	AC turn ON
19	22/05/2022	17:37:35	0.09	218.80	14.10	0.69	ON	33.70	AC turn ON
20	22/05/2022	17:37:40	0.09	218.60	14.10	0.69	ON	33.80	AC turn ON
21	22/05/2022	17:37:46	0.09	218.70	14.00	0.69	ON	33.70	AC turn ON
22	22/05/2022	17:37:51	0.09	218.30	14.00	0.69	ON	33.70	AC turn ON
23	22/05/2022	17:37:58	0.09	218.40	13.90	0.69	ON	33.70	AC turn ON
24	22/05/2022	17:38:05	0.09	218.40	13.90	0.69	ON	33.70	AC turn ON
25	22/05/2022	17:38:10	0.09	218.30	13.80	0.69	ON	33.80	AC turn ON
26	22/05/2022	17:38:16	0.09	218.40	13.80	0.69	ON	33.80	AC turn ON
27	22/05/2022	17:38:21	0.09	218.40	13.70	0.69	ON	33.70	AC turn ON
28	22/05/2022	17:38:28	0.09	218.60	13.80	0.69	ON	33.80	AC turn ON
29	22/05/2022	17:38:33	0.09	218.70	13.70	0.69	ON	33.70	AC turn ON
30	22/05/2022	17:38:39	0.09	218.70	13.80	0.69	ON	33.80	AC turn ON
31	22/05/2022	17:38:44	0.09	219.00	13.80	0.69	ON	33.70	AC turn ON
32	22/05/2022	17:38:50	0.09	219.00	13.80	0.69	ON	33.80	AC turn ON
33	22/05/2022	17:38:57	0.09	219.10	13.90	0.69	ON	33.70	AC turn ON
34	22/05/2022	17:39:04	0.09	219.10	13.80	0.68	ON	33.80	AC turn ON
35	22/05/2022	17:39:10	0.09	219.30	13.80	0.68	ON	33.70	AC turn ON
36	22/05/2022	17:39:15	0.09	219.20	13.80	0.68	ON	33.70	AC turn ON
37	22/05/2022	17:39:21	0.09	219.10	13.80	0.68	ON	33.70	AC turn ON
38	22/05/2022	17:39:27	0.09	219.30	13.80	0.68	ON	33.70	AC turn ON
39	22/05/2022	17:39:34	0.09	219.50	13.80	0.69	ON	33.80	AC turn ON
40	22/05/2022	17:39:41	0.09	219.40	13.80	0.69	ON	33.70	AC turn ON
41	22/05/2022	17:40:57	0.09	219.30	13.40	0.69	ON	33.70	AC turn ON
42	22/05/2022	17:41:01	0.09	219.30	13.40	0.69	ON	33.70	AC turn ON
43	22/05/2022	17:41:06	0.09	219.30	13.30	0.68	ON	33.70	AC turn ON
44	22/05/2022	17:41:10	0.09	219.10	13.30	0.68	ON	33.70	AC turn ON
45	22/05/2022	17:41:14	0.09	218.80	13.30	0.68	ON	33.70	5'th minutes
46	22/05/2022	17:41:19	0.00	219.90	0.00	0.00	OFF	33.80	AC turn OFF
47	22/05/2022	17:41:23	0.00	219.70	0.00	0.00	OFF	33.70	AC turn OFF
48	22/05/2022	17:41:27	0.00	219.50	0.00	0.00	OFF	33.70	AC turn OFF
49	22/05/2022	17:41:31	0.00	219.50	0.00	0.00	OFF	33.80	AC turn OFF
50	22/05/2022	17:41:35	0.00	219.60	0.00	0.00	OFF	33.80	AC turn OFF
51	22/05/2022	17:41:38	0.00	219.70	0.00	0.00	OFF	33.70	AC turn OFF

52	22/05/2022	17:41:42	0.00	219.70	0.00	0.00	OFF	33.70	AC turn OFF
53	22/05/2022	17:41:46	0.00	219.40	0.00	0.00	OFF	33.80	AC turn OFF
54	22/05/2022	17:41:50	0.00	219.10	0.00	0.00	OFF	33.70	AC turn OFF
55	22/05/2022	17:41:57	0.00	219.00	0.00	0.00	OFF	33.80	AC turn OFF
56	22/05/2022	17:42:02	0.00	218.90	0.00	0.00	OFF	33.80	AC turn OFF
57	22/05/2022	17:42:08	0.00	218.90	0.00	0.00	OFF	33.80	AC turn OFF
58	22/05/2022	17:42:13	0.00	218.90	0.00	0.00	OFF	33.70	AC turn OFF
59	22/05/2022	17:42:19	0.00	219.00	0.00	0.00	OFF	33.70	AC turn OFF
60	22/05/2022	17:42:24	0.00	218.90	0.00	0.00	OFF	33.70	AC turn OFF
61	22/05/2022	17:42:31	0.00	219.10	0.00	0.00	OFF	33.70	AC turn OFF
62	22/05/2022	17:42:37	0.00	219.10	0.00	0.00	OFF	33.70	AC turn OFF
63	22/05/2022	17:42:43	0.00	219.30	0.00	0.00	OFF	33.70	AC turn OFF
64	22/05/2022	17:42:50	0.00	219.50	0.00	0.00	OFF	33.70	AC turn OFF
65	22/05/2022	17:42:55	0.00	219.40	0.00	0.00	OFF	33.70	AC turn OFF
66	22/05/2022	17:43:01	0.00	219.50	0.00	0.00	OFF	33.70	AC turn OFF
67	22/05/2022	17:43:08	0.00	219.20	0.00	0.00	OFF	33.70	AC turn OFF
68	22/05/2022	17:43:16	0.00	219.20	0.00	0.00	OFF	33.70	AC turn OFF
69	22/05/2022	17:43:21	0.00	219.10	0.00	0.00	OFF	33.70	AC turn OFF
70	22/05/2022	17:43:32	0.00	219.10	0.00	0.00	OFF	33.70	AC turn OFF
71	22/05/2022	17:43:39	0.00	219.20	0.00	0.00	OFF	33.70	AC turn OFF
72	22/05/2022	17:43:46	0.00	218.90	0.00	0.00	OFF	33.70	AC turn OFF
73	22/05/2022	17:43:53	0.00	218.90	0.00	0.00	OFF	33.70	AC turn OFF
74	22/05/2022	17:43:59	0.00	218.80	0.00	0.00	OFF	33.70	AC turn OFF
75	22/05/2022	17:44:06	0.00	218.90	0.00	0.00	OFF	33.70	AC turn OFF
76	22/05/2022	17:44:11	0.00	218.90	0.00	0.00	OFF	33.70	AC turn OFF
77	22/05/2022	17:44:17	0.00	219.00	0.00	0.00	OFF	33.70	AC turn OFF
78	22/05/2022	17:44:22	0.00	219.00	0.00	0.00	OFF	33.70	AC turn OFF
79	22/05/2022	17:45:32	0.00	219.00	0.00	0.00	OFF	33.60	AC turn OFF
80	22/05/2022	17:45:36	0.00	218.80	0.00	0.00	OFF	33.70	AC turn OFF
81	22/05/2022	17:45:40	0.00	218.70	0.00	0.00	OFF	33.70	AC turn OFF
82	22/05/2022	17:45:44	0.00	218.80	0.00	0.00	OFF	33.70	AC turn OFF
83	22/05/2022	17:45:49	0.00	218.70	0.00	0.00	OFF	33.70	AC turn OFF
84	22/05/2022	17:45:52	0.00	218.80	0.00	0.00	OFF	33.70	AC turn OFF

Berdasarkan tabel hasil uji, sesaat setelah sistem ON (baris ke 3) menunjukkan pendingin ruangan telah dinyalakan oleh sistem pada pukul 17: 36:16 dengan kondisi temperatur ruangan 33.70.

2	22/05/2022	17:36:09	0.00	218.70	0.00	0.00	OFF	33.70	System ON
3	22/05/2022	17:36:16	0.00	218.60	0.00	0.00	ON	33.70	AC turn ON

Gambar 3. Kondisi pendingin ruangan dinyalakan.

Setelah 5 menit yaitu pukul 17:41:19, namun temperatur ruangan tidak berubah menjadi lebih rendah dari 33.70 derajat celcius akan tetapi berubah lebih tinggi menjadi 33.80 derajat celcius , hal ini dianggap pendingin ruangan dalam keadaan tidak normal. Oleh karena itu pendingin ruangan dimatikan.

44	22/05/2022	17:41:10	0.09	219.10	13.30	0.68	ON	33.70	AC turn ON
45	22/05/2022	17:41:14	0.09	218.80	13.30	0.68	ON	33.70	5'th minutes
46	22/05/2022	17:41:19	0.00	219.90	0.00	0.00	OFF	33.80	AC turn OFF

Gambar 4. Kondisi pendingin ruangan dimatikan.

2. Modul LSS

Modul lampu ini menggunakan sensor arus PZEM-004T dan sensor LDR yang menjadi alat untuk monitoring sekaligus membandingkan data untuk dijadikan acuan tindakan otomatisasi pada sistem. Berikut diperlihatkan tabel hasil uji coba modul lampu dengan kondisi ruangan gelap dan terang. Uji coba ini merupakan uji coba menggunakan beban lampu 20 Watt.

Tabel 3. Percobaan dengan kondisi ruangan gelap dan terang.

No	Date	Time	I (A)	V (Volt)	P (W)	PF	Sw	Room Cond.	Status
1	22/05/2022	20:01:44	0.00	217.10	0.00	0.00	OFF	Null	System ON
2	22/05/2022	20:01:51	0.00	217.40	0.00	0.00	OFF	Null	System ON
3	22/05/2022	20:01:57	0.04	217.10	13.80	0.71	ON	DARK	Lamp turn ON
4	22/05/2022	20:02:03	0.09	217.00	14.10	0.71	ON	DARK	Lamp turn ON
5	22/05/2022	20:02:09	0.09	216.80	14.30	0.71	ON	DARK	Lamp turn ON
6	22/05/2022	20:02:15	0.00	217.00	0.00	0.00	OFF	BRIGHT	Lamp turn OFF
7	22/05/2022	20:02:20	0.00	217.10	0.00	0.00	OFF	BRIGHT	Lamp turn OFF
8	22/05/2022	20:02:25	0.09	216.80	14.40	0.71	ON	DARK	Lamp turn ON
9	22/05/2022	20:02:29	0.09	217.10	14.50	0.71	ON	DARK	Lamp turn ON
10	22/05/2022	20:02:35	0.09	216.90	14.40	0.71	ON	DARK	Lamp turn ON
11	22/05/2022	20:02:40	0.00	217.10	0.70	1.00	OFF	BRIGHT	Lamp turn OFF
12	22/05/2022	20:02:45	0.00	217.00	0.00	0.00	OFF	BRIGHT	Lamp turn OFF
13	22/05/2022	20:02:50	0.00	217.00	0.00	0.00	OFF	BRIGHT	Lamp turn OFF
14	22/05/2022	20:02:54	0.09	216.70	13.80	0.67	ON	DARK	Lamp turn ON
15	22/05/2022	20:03:02	0.09	217.30	14.60	0.71	ON	DARK	Lamp turn ON
16	22/05/2022	20:03:07	0.09	217.40	0.00	0.00	OFF	Null	Resetted system
17	22/05/2022	20:03:12	0.00	217.40	0.00	0.00	OFF	Null	Resetted system
18	22/05/2022	20:03:17	0.00	217.50	0.00	0.00	OFF	Null	Resetted system
19	22/05/2022	20:03:23	0.00	217.50	0.00	0.00	OFF	Null	Resetted system
20	22/05/2022	20:03:28	0.00	216.90	0.00	0.00	OFF	Null	Resetted system
21	22/05/2022	20:03:34	0.00	217.00	0.00	0.00	OFF	Null	Resetted system
22	22/05/2022	20:03:40	0.00	216.80	0.00	0.00	OFF	Null	Resetted system
23	22/05/2022	20:03:45	0.00	216.60	0.00	0.00	OFF	Null	Resetted system
24	22/05/2022	20:03:51	0.06	216.70	14.50	0.71	ON	DARK	Lamp turn ON
25	22/05/2022	20:03:56	0.09	216.50	14.50	0.71	ON	DARK	Lamp turn ON
26	22/05/2022	20:04:02	0.09	216.70	14.40	0.71	ON	DARK	Lamp turn ON

Berdasarkan tabel hasil uji, sesaat setelah sistem ON (baris ke 3) menunjukkan ruangan dalam kondisi gelap, maka lampu dinyalakan.

2	22/05/2022	20:01:51	0.00	217.40	0.00	0.00	OFF	Null	System ON
3	22/05/2022	20:01:57	0.04	217.10	13.80	0.71	ON	DARK	Lamp turn ON
4	22/05/2022	20:02:03	0.09	217.00	14.10	0.71	ON	DARK	Lamp turn ON

Gambar 5. Kondisi ruangan gelap.

Pada data baris ke 6 menunjukkan kondisi ruangan terang, maka lampu dimatikan.

5	22/05/2022	20:02:09	0.09	216.80	14.30	0.71	ON	DARK	Lamp turn ON
6	22/05/2022	20:02:15	0.00	217.00	0.00	0.00	OFF	BRIGHT	Lamp turn OFF

Gambar 6. Kondisi ruangan terang.

3. Modul SSS

Modul stok kontak ini menggunakan sensor arus PZEM-004T dan sensor suhu DS18B20 yang menjadi alat untuk monitoring sekaligus membandingkan data untuk dijadikan acuan tindakan otomatisasi pada sistem. Berikut diperlihatkan tabel hasil uji coba modul stok kontak dengan kondisi panas berlebih (overheat) dan beban berlebih (overload). Uji coba ini merupakan uji coba terbatas menggunakan beban lampu 20 Watt, cas laptop 45 Watt, serta solder untuk memanaskan sensor suhu.

Tabel 4. Percobaan dengan kondisi kabel panas berlebih.

ID	Date	Time	I (A)	V (Volt)	P (W)	PF	Sw	Cable therm.	Status
1	22/05/2022	20:52:25	0.00	218.10	0.00	0.00	OFF	33.44	System ON
2	22/05/2022	20:52:33	0.0	217.90	14.40	0.70	ON	33.38	Socket turn ON
3	22/05/2022	20:52:40	0.09	218.00	14.40	0.70	ON	33.25	Socket turn ON
4	22/05/2022	20:52:46	0.09	218.00	14.30	0.71	ON	33.25	Socket turn ON
5	22/05/2022	20:52:53	0.09	218.10	14.30	0.70	ON	33.69	Socket turn ON
6	22/05/2022	20:52:59	0.09	217.90	14.20	0.71	ON	35.50	Socket turn ON
7	22/05/2022	20:53:06	0.09	218.20	14.20	0.70	ON	37.63	Socket turn ON
8	22/05/2022	20:53:14	0.09	218.30	14.10	0.70	ON	39.81	Socket turn ON
9	22/05/2022	20:53:20	0.09	218.20	13.90	0.70	ON	42.75	Socket turn ON
10	22/05/2022	20:53:28	0.09	217.90	14.00	0.70	ON	45.94	Socket turn ON
11	22/05/2022	20:53:35	0.09	218.00	14.00	0.70	ON	51.81	Socket turn ON
12	22/05/2022	20:53:41	0.09	218.10	14.00	0.70	ON	56.75	Socket turn ON
13	22/05/2022	20:53:48	0.09	218.00	14.00	0.71	ON	61.50	Socket turn ON
14	22/05/2022	20:53:56	0.09	217.90	14.00	0.71	ON	66.19	Socket turn ON
15	22/05/2022	20:54:09	0.00	218.00	0.00	0.00	OFF	73.63	Overheat
16	22/05/2022	20:54:15	0.00	218.00	0.00	0.00	OFF	73.63	Socket turn OFF
17	22/05/2022	20:54:21	0.00	218.10	0.00	0.00	OFF	73.63	Socket turn OFF
18	22/05/2022	20:54:26	0.00	217.90	0.00	0.00	OFF	73.63	Socket turn OFF
19	22/05/2022	20:54:31	0.00	218.30	0.00	0.00	OFF	73.63	Socket turn OFF

20	22/05/2022	20:54:35	0.00	218.40	0.00	0.00	OFF	73.63	Socket turn OFF
21	22/05/2022	20:54:42	0.00	218.40	0.00	0.00	OFF	73.63	Socket turn OFF
22	22/05/2022	20:54:47	0.00	218.30	0.00	0.00	OFF	73.63	Socket turn OFF
23	22/05/2022	20:54:54	0.00	218.10	0.00	0.00	OFF	73.63	Socket turn OFF
24	22/05/2022	20:55:02	0.00	217.60	0.00	0.00	OFF	73.63	Socket turn OFF
25	22/05/2022	20:55:11	0.00	218.20	0.00	0.00	OFF	73.63	Socket turn OFF
26	22/05/2022	20:55:16	0.00	218.30	0.00	0.00	OFF	73.63	Socket turn OFF
27	22/05/2022	20:55:23	0.00	218.10	0.00	0.00	OFF	73.63	Socket turn OFF
28	22/05/2022	20:55:28	0.00	217.90	0.00	0.00	OFF	73.63	Socket turn OFF
29	22/05/2022	20:56:38	0.00	0.00	0.00	0.00	OFF	0.00	Resetted system
30	22/05/2022	20:56:42	0.00	218.70	0.00	0.00	OFF	0.00	System ON
31	22/05/2022	20:56:47	0.00	218.80	0.00	0.00	OFF	0.00	System ON
32	22/05/2022	20:56:52	0.00	218.50	0.00	0.00	OFF	0.00	System ON
33	22/05/2022	20:56:58	0.00	218.30	0.00	0.00	OFF	0.00	System ON
34	22/05/2022	20:57:02	0.00	218.20	0.00	0.00	OFF	0.00	System ON
35	22/05/2022	20:57:07	0.00	218.60	0.00	0.00	OFF	0.00	System ON
36	22/05/2022	20:57:12	0.00	218.10	0.00	0.00	OFF	0.00	System ON
37	22/05/2022	20:57:17	0.00	218.00	0.00	0.00	OFF	0.00	System ON
38	22/05/2022	20:57:21	0.00	217.90	0.00	0.00	OFF	0.00	System ON
39	22/05/2022	20:57:26	0.00	218.00	0.00	0.00	OFF	0.00	System ON
40	22/05/2022	20:57:32	0.00	218.10	0.00	0.00	OFF	0.00	System ON
41	22/05/2022	20:57:40	0.00	218.20	0.00	0.00	OFF	0.00	System ON
42	22/05/2022	20:57:47	0.00	218.10	1.60	0.71	ON	46.75	Socket turn ON
43	22/05/2022	20:57:56	0.09	218.00	14.40	0.71	ON	46.00	Socket turn ON
44	22/05/2022	20:58:08	0.09	218.40	14.50	0.71	ON	45.00	Socket turn ON
45	22/05/2022	20:58:17	0.09	218.40	14.50	0.71	ON	44.31	Socket turn ON
46	22/05/2022	20:58:23	0.09	218.90	14.40	0.71	ON	43.75	Socket turn ON

Berdasarkan tabel hasil uji, sesaat setelah sistem ON (baris ke 2) menunjukkan stok kontak dinyalakan.

1	22/05/2022	20:52:25	0.00	218.10	0.00	0.00	OFF	33.44	System ON
2	22/05/2022	20:52:33	0.0	217.90	14.40	0.70	ON	33.38	Socket turn ON

Gambar 7. Kondisi stok kontak dinyalakan.

Pada data baris ke 15 menunjukkan kondisi kabel stok kontak mencapai 73,63 derajat celcius, telah melebihi batas maksimal suhu kabel yaitu 70 derajat celcius (overheat). Oleh karena itu stok kontak dinonaktifkan.

14	22/05/2022	20:53:56	0.09	217.90	14.00	0.71	ON	66.19	Socket turn ON
15	22/05/2022	20:54:09	0.00	218.00	0.00	0.00	OFF	73.63	Overheat
16	22/05/2022	20:54:15	0.00	218.00	0.00	0.00	OFF	73.63	Socket turn OFF

Gambar 8. Kondisi kabel overheat.

Tabel 5. Percobaan dengan kondisi beban berlebih.

No	Date	Time	I (A)	V (Volt)	P (W)	PF	Sw	Cable temp.	Status
1	22/05/2022	21:56:20	0.00	219.80	0.00	0.00	OFF	0.00	System ON
2	22/05/2022	21:56:24	0.00	220.10	0.00	0.00	OFF	0.00	Socket turn ON
3	22/05/2022	21:56:31	0.09	220.50	14.00	0.71	ON	31.94	Socket turn ON
4	22/05/2022	21:57:22	0.21	220.20	43.70	0.95	ON	31.94	Overload
5	22/05/2022	21:57:27	0.00	220.20	0.00	0.00	OFF	31.94	Soket turn OFF
6	22/05/2022	21:57:34	0.00	220.00	0.00	0.00	OFF	31.94	Soket turn OFF
7	22/05/2022	21:57:39	0.00	220.10	0.00	0.00	OFF	31.94	Soket turn OFF
8	22/05/2022	21:57:48	0.00	220.40	0.00	0.00	OFF	31.94	Soket turn OFF
9	22/05/2022	21:57:54	0.00	220.60	0.00	0.00	OFF	31.94	Soket turn OFF
10	22/05/2022	21:58:04	0.00	221.10	0.00	0.00	OFF	31.94	Soket turn OFF
11	22/05/2022	22:10:48	0.00	0.00	0.00	0.00	OFF	0.00	Resetted system
12	22/05/2022	22:10:55	0.00	0.00	0.00	0.00	OFF	0.00	Resetted system
13	22/05/2022	22:11:02	0.09	218.00	14.30	0.71	ON	31.31	Socket turn ON
14	22/05/2022	22:11:09	0.09	218.00	14.30	0.71	ON	31.31	Socket turn ON
15	22/05/2022	22:11:18	0.09	217.90	14.60	0.95	ON	31.31	Socket turn ON
16	22/05/2022	22:12:26	0.21	217.60	43.70	0.95	ON	31.31	Overload
17	22/05/2022	22:12:33	0.00	0.00	0.00	0.00	OFF	0.00	Resetted system
18	22/05/2022	22:12:37	0.00	0.00	0.00	0.00	OFF	0.00	Resetted system
19	22/05/2022	22:12:43	0.09	217.80	14.30	0.00	ON	31.25	Socket turn ON
20	22/05/2022	22:12:47	0.09	217.40	14.30	0.71	ON	31.25	Socket turn ON
21	22/05/2022	22:12:56	0.00	218.20	7.80	0.72	ON	31.25	Socket turn ON
22	22/05/2022	22:13:03	0.09	217.80	14.30	0.00	ON	31.25	Socket turn ON
23	22/05/2022	22:13:09	0.09	217.40	14.30	0.71	ON	31.25	Socket turn ON
24	22/05/2022	22:13:16	0.09	217.40	14.20	0.71	ON	31.25	Socket turn ON
25	22/05/2022	22:13:24	0.09	217.60	14.20	0.70	ON	31.25	Socket turn ON
26	22/05/2022	22:13:32	0.09	217.70	14.10	0.71	ON	31.25	Socket turn ON
27	22/05/2022	22:13:39	0.09	217.50	14.10	0.71	ON	31.25	Socket turn ON
28	22/05/2022	22:13:46	0.09	217.60	14.00	0.71	ON	31.25	Socket turn ON
29	22/05/2022	22:13:53	0.09	217.70	14.00	0.71	ON	31.25	Socket turn ON
30	22/05/2022	22:14:19	0.21	217.50	13.90	0.71	ON	31.25	Socket turn ON

31	22/05/2022	22:14:25	0.09	218.40	13.90	0.71	ON	31.25	Socket turn ON
32	22/05/2022	22:14:39	0.09	218.80	13.90	0.95	ON	31.25	Socket turn ON
33	22/05/2022	22:15:06	0.21	218.10	13.70	0.71	ON	31.25	Socket turn ON
34	22/05/2022	22:15:13	0.09	217.90	13.80	0.70	ON	31.25	Socket turn ON
35	22/05/2022	22:15:20	0.09	218.10	13.70	0.71	ON	31.25	Socket turn ON
36	22/05/2022	22:15:27	0.09	218.10	13.70	0.95	ON	31.25	Socket turn ON
37	22/05/2022	22:16:36	0.21	217.70	43.30	0.95	ON	31.25	Overload
38	22/05/2022	22:16:41	0.06	218.60	11.70	0.00	OFF	31.25	Socket turn ON
39	22/05/2022	22:16:45	0.00	218.70	0.00	0.00	OFF	31.25	Socket turn ON
40	22/05/2022	22:16:51	0.00	218.60	0.00	0.00	OFF	31.25	Socket turn ON
41	22/05/2022	22:16:55	0.00	218.50	0.00	0.00	OFF	31.25	Socket turn ON
42	22/05/2022	22:17:00	0.00	218.50	0.00	0.00	OFF	31.25	Socket turn ON
43	22/05/2022	22:17:05	0.00	218.50	0.00	0.00	OFF	31.25	Socket turn ON
44	22/05/2022	22:17:10	0.00	218.30	0.00	0.00	OFF	31.25	Socket turn ON

Berdasarkan tabel hasil uji, sesaat setelah sistem ON (baris ke 2) menunjukkan stok kontak dinyalakan.

1	22/05/2022	21:56:20	0.00	219.80	0.00	0.00	OFF	0.00	System ON
2	22/05/2022	21:56:24	0.00	220.10	0.00	0.00	OFF	0.00	Socket turn ON

Gambar 9. Kondisi stok kontak dinyalakan.

Pada data baris ke 4 menunjukkan kondisi daya beban stok kontak mencapai 43,70 watt, telah melebihi batas maksimal daya yang telah ditentukan yaitu 40 watt (overload). Oleh karena itu stok kontak dinonaktifkan.

3	22/05/2022	21:56:31	0.09	220.50	14.00	0.71	ON	31.94	Socket turn ON
4	22/05/2022	21:57:22	0.21	220.20	43.70	0.95	ON	31.94	Overload

Gambar 10. Kondisi stok kontak overload.

4. Modul PSS

Modul panel ini hanya menggunakan sensor arus PZEM-004T yang menjadi alat untuk monitoring penggunaan arus. Berikut diperlihatkan tabel hasil uji coba modul panel.

Tabel 6. Percobaan monitoring penggunaan arus.

No	Date	Time	I (A)	V (Volt)	P (W)	PF	Status
1	22/05/2022	16:42:54	0.16	219.90	23.80	0.72	Normal
2	22/05/2022	16:42:56	0.16	220.20	24.00	0.72	Normal
3	22/05/2022	16:42:58	0.16	220.20	24.30	0.72	Normal

4	22/05/2022	16:43:01	0.16	220.30	23.80	0.72	Normal
5	22/05/2022	16:43:04	0.16	220.10	23.80	0.72	Normal
6	22/05/2022	16:43:06	0.16	220.20	25.50	0.71	Normal
7	22/05/2022	16:43:09	0.16	220.10	24.50	0.72	Normal
8	22/05/2022	16:43:11	0.16	220.20	24.60	0.72	Normal
9	22/05/2022	16:43:13	0.16	220.20	24.00	0.72	Normal
10	22/05/2022	16:43:17	0.16	220.30	24.00	0.72	Normal
11	22/05/2022	16:43:21	0.16	220.20	24.40	0.71	Normal
12	22/05/2022	16:43:23	0.16	219.50	24.40	0.72	Normal
13	22/05/2022	16:43:26	0.16	219.60	24.60	0.71	Normal
14	22/05/2022	16:43:28	0.16	219.80	24.40	0.71	Normal
15	22/05/2022	16:43:30	0.16	219.90	24.10	0.71	Normal
16	22/05/2022	16:43:33	0.16	220.00	23.70	0.71	Normal
17	22/05/2022	16:43:35	0.16	220.10	23.70	0.71	Normal
18	22/05/2022	16:43:38	0.16	220.00	24.00	0.72	Normal
19	22/05/2022	16:43:40	0.16	220.10	23.90	0.71	Normal
20	22/05/2022	16:43:42	0.16	220.10	0.00	0.71	Normal
21	22/05/2022	16:43:46	0.16	220.20	24.60	0.71	Normal
22	22/05/2022	16:43:48	0.16	220.20	24.30	0.71	Normal
23	22/05/2022	16:43:51	0.16	220.20	24.30	0.71	Normal
24	22/05/2022	16:43:53	0.16	220.00	23.70	0.71	Normal
25	22/05/2022	16:43:56	0.16	220.10	24.10	0.72	Normal
26	22/05/2022	16:43:59	0.16	220.10	24.10	0.71	Normal
27	22/05/2022	16:44:01	0.16	219.90	23.80	0.72	Normal
28	22/05/2022	16:44:03	0.16	220.00	24.50	0.71	Normal
29	22/05/2022	16:44:06	0.17	220.00	23.70	0.71	Normal
30	22/05/2022	16:44:08	0.16	219.90	23.90	0.71	Normal
31	22/05/2022	16:44:11	0.16	220.70	26.30	0.71	Normal
32	22/05/2022	16:44:13	0.16	220.90	24.00	0.72	Normal
33	22/05/2022	16:44:15	0.16	220.90	24.30	0.72	Normal
34	22/05/2022	16:44:20	0.16	220.80	23.80	0.71	Normal
35	22/05/2022	16:44:22	0.16	220.90	24.30	0.72	Normal
36	22/05/2022	16:44:24	0.16	220.60	23.60	0.72	Normal
37	22/05/2022	16:44:27	0.15	220.60	23.90	0.72	Normal
38	22/05/2022	16:44:29	0.15	220.90	10.60	0.50	Normal
39	22/05/2022	16:44:32	0.10	221.00	10.90	0.50	Normal
40	22/05/2022	16:44:34	0.10	220.80	10.50	0.50	Normal
41	22/05/2022	16:44:38	0.10	220.80	10.60	0.50	Normal
42	22/05/2022	16:44:41	0.10	220.80	10.50	0.50	Normal
43	22/05/2022	16:44:43	0.10	220.90	10.40	0.49	Normal
44	22/05/2022	16:44:46	0.10	220.90	11.20	0.50	Normal
45	22/05/2022	16:44:48	0.10	220.80	10.70	0.50	Normal
46	22/05/2022	16:44:51	0.11	220.90	11.30	0.51	Normal

47	22/05/2022	16:44:53	0.10	220.80	10.70	0.50	Normal
48	22/05/2022	16:45:50	0.16	221.30	24.20	0.72	Normal
49	22/05/2022	16:45:53	0.16	221.20	23.90	0.72	Normal
50	22/05/2022	16:45:55	0.16	221.30	23.90	0.72	Normal
51	22/05/2022	16:45:58	0.16	221.10	24.00	0.72	Normal
52	22/05/2022	16:46:01	0.16	221.50	24.00	0.72	Normal
53	22/05/2022	16:46:04	0.16	221.60	24.20	0.72	Normal
54	22/05/2022	16:46:10	0.15	221.00	24.00	0.72	Normal
55	22/05/2022	16:46:14	0.16	221.10	24.40	0.72	Normal
56	22/05/2022	16:46:18	0.16	221.20	23.90	0.72	Normal
57	22/05/2022	16:46:22	0.16	221.20	24.00	0.72	Normal
58	22/05/2022	16:46:25	0.15	221.20	24.00	0.72	Normal
59	22/05/2022	16:46:28	0.15	221.10	23.90	0.72	Normal

5. Hasil Pengujian Blackbox MACSys

Tabel 1. Hasil pengujian blackbox untuk hardware

No.	Perangkat sistem	Input tes / kondisi	Hasil yang diharapkan	Output tes	Hasil tes
1	Sistem AC	Sistem AC di on kan	Sistem standby	Sistem standby	Sukses
		Saklar on AC pada aplikasi android ditekan	Sistem menyalakan AC	AC menyala	Sukses
		Suhu berubah menjadi lebih rendah dari suhu awal setelah 5 menit	Perangkat AC terus dinyalakan	AC menyala secara terus menerus	Sukses
		Suhu tidak berubah menjadi lebih rendah dari suhu awal setelah 5 menit	Perangkat dipadamkan, notifikasi dikirim ke aplikasi	AC dipadamkan dan notifikasi diterima di aplikasi	Sukses
		Saklar off AC pada aplikasi android ditekan	Sistem memadamkan AC	AC padam	Sukses
2	Sistem lampu	Sistem lampu di on kan	Sistem standby	Sistem standby	Sukses
		Saklar on auto lampu pada aplikasi ditekan	Sistem lampu otomatis dinyalakan	Sistem lampu otomatis menyala	Sukses
		Teras gedung gelap	Lampu teras dinyalakan	Lampu teras menyala	Sukses
		Teras gedung terang	Lampu teras dipadamkan	Lampu teras padam	Sukses
		Saklar off auto lampu pada aplikasi ditekan	Sistem lampu otomatis dipadamkan, sensor cahaya tidak merespon	Sistem lampu otomatis padam, dan sensor cahaya tidak merespon	Sukses
		Tombol on lampu manual pada aplikasi ditekan	Lampu manual / ruangan dinyalakan	Lampu manual / ruangan menyala	Sukses
		Tombol off lampu manual pada aplikasi ditekan	Lampu manual / ruangan dipadamkan	Lampu manual / ruangan padam	Sukses
3	Sistem soket	Sistem soket di on kan	Sistem soket standby	Sistem soket standby	Sukses

		Saklar on soket pada aplikasi ditekan	Sistem soket dinyalakan	Sistem soket menyala	Sukses
		Beban lebih rendah dari 40 watt	Soket dinyalakan terus	Soket menyala terus	Sukses
		Beban sebesar 40 watt keatas	Soket dipadamkan, notifikasi dikirim ke aplikasi	Soket padam, notifikasi diterima di aplikasi	Sukses
		Suhu kabel di bawah 70 derajat celcius	Soket dinyalakan terus	Soket menyala terus	Sukses
		Suhu kabel sebesar 70 derajat ke atas	Soket dipadamkan, notifikasi dikirim ke aplikasi	Soket padam, notifikasi dikirim ke aplikasi	Sukses
		Saklar off soket pada aplikasi ditekan	Sistem soket dipadamkan	Sistem soket padam	Sukses
4	Panel	Sistem panel di on kan	Sistem panel dinyalakan	Sistem panel menyala	Sukses
		Sistem on	Selama sistem di on kan, data arus konsumsi dan grounding dikirim ke aplikasi	Data arus konsumsi dan grounding dikirim ke aplikasi secara terus menerus	Sukses

6. Hasil Pengujian Blackbox Aplikasi MACSys

Tabel 2. Hasil pengujian aplikasi android MACSys

No.	Input tes	Hasil yang diharapkan	Output tes	Hasil tes
1	Icon aplikasi MACsys	Aplikasi terbuka	Aplikasi terbuka	Sukses
2	Tombol Sign In ditekan	Screen Home terbuka	Screen home ditampilkan	Sukses
3	Icon AC ditekan	Monitoring dan Kontrol sistem AC terbuka, dan memunculkan data dari mikrokontroler	Monitoring dan Kontrol sistem AC terbuka, dan memunculkan data dari mikrokontroler	Sukses
4	Saklar off AC ditekan	Saklar off berubah menjadi on, dan mengaktifkan sistem AC	Saklar off berubah menjadi on, dan mengaktifkan sistem AC	Sukses
5	Saklar on AC ditekan	Saklar on berubah menjadi off, dan menonaktifkan sistem AC	Saklar on berubah menjadi off, dan menonaktifkan sistem AC	Sukses
6	Icon home pada screen AC ditekan	Screen dialihkan ke menu home	Screen dialihkan ke menu home	Sukses
7	Icon lampu ditekan	Monitoring dan kontrol sistem lampu terbuka, dan memunculkan data dari mikrokontroler	Monitoring dan kontrol sistem lampu terbuka, dan memunculkan data dari mikrokontroler	Sukses
8	Saklar off lampu auto ditekan	Saklar off berubah menjadi on, dan mengaktifkan sistem lampu auto	Saklar off berubah menjadi on, dan mengaktifkan sistem lampu auto	Sukses
9	Saklar on lampu auto ditekan	Saklar on berubah menjadi off, dan menonaktifkan sistem lampu auto	Saklar on berubah menjadi off, dan menonaktifkan sistem lampu auto	Sukses

10	Saklar off lampu manual ditekan	Saklar off berubah menjadi on, dan mengaktifkan sistem lampu manual	Saklar off berubah menjadi on, dan mengaktifkan sistem lampu manual	Sukses
11	Saklar on lampu auto ditekan	Saklar on berubah menjadi off, dan menonaktifkan sistem lampu manual	Saklar on berubah menjadi off, dan menonaktifkan sistem lampu manual	Sukses
12	Icon home pada screen lampu ditekan	Screen dialihkan ke menu home	Screen dialihkan ke menu home	Sukses
13	Icon soket ditekan	Monitoring dan Kontrol sistem soket terbuka, dan memunculkan data dari mikrokontroler	Monitoring dan Kontrol sistem soket terbuka, dan memunculkan data dari mikrokontroler	Sukses
14	Saklar off soket ditekan	Saklar off berubah menjadi on, dan mengaktifkan sistem soket	Saklar off berubah menjadi on, dan mengaktifkan sistem soket	Sukses
15	Saklar on AC ditekan	Saklar on berubah menjadi off, dan menonaktifkan sistem soket	Saklar on berubah menjadi off, dan menonaktifkan sistem soket	Sukses
16	Icon home pada screen soket ditekan	Screen dialihkan ke menu home	Screen dialihkan ke menu home	Sukses
17	Icon panel ditekan	Monitoring sistem panel terbuka, dan memunculkan data dari mikrokontroler	Monitoring sistem panel terbuka, dan memunculkan data dari mikrokontroler	Sukses
18	Icon home pada screen panel ditekan	Screen dialihkan ke menu home	Screen dialihkan ke menu home	Sukses

7. Rincian Biaya Modul MACSys

Tabel 1. Rincian biaya modul ACSS (*AC sub system*)

No.	Nama Part	Model / Type	Harga	Jumlah (pcs)	Sub total
1	Power supply	Switching 5V/5A	Rp 36.000	1	Rp 36.000
2	Mikrokontroller	Wemos mini D1	Rp 34.000	1	Rp 34.000
3	Sensor arus	PZEM-004T	Rp 102.000	1	Rp 102.000
4	Sensor suhu	DHT-22	Rp 45.000	1	Rp 45.000
5	Relay	1 Channel Opto 5V	Rp 12.000	1	Rp 12.000
6	Switch	On/Off	Rp 1.000	1	Rp 1.000
7	PCB polos	Fiber glass single layer	Rp 20.000	1	Rp 20.000
8	Stok kontak	2 hole	Rp 12.000	1	Rp 12.000
9	Box	X6 180x110x60mm	Rp 9.500	1	Rp 9.500
10	Stekker	Broco	Rp 12.500	1	Rp 12.500
11	Kabel serabut	Eterna	Rp 8.000	1	Rp 8.000
12	Pelarut PCB	Fery Chloride	Rp 7.500	1	Rp 7.500
Total :				12	Rp 299.500

Tabel 2. Rincian biaya Modul LSS (*Lamp sub system*)

No.	Nama Part	Model / Type	Harga	Jumlah (pcs)	Sub total
1	Power supply	Switching 5V/5A	Rp 36.000	1	Rp 36.000
2	Mikrokontroller	Wemos mini D1	Rp 34.000	1	Rp 34.000
3	Sensor arus	PZEM-004T	Rp 102.000	1	Rp 102.000
4	Sensor cahaya	LDR	Rp 8.000	1	Rp 8.000
5	Relay	1 Channel Opto 5V	Rp 12.000	2	Rp 24.000
6	Switch	On/Off	Rp 1.000	1	Rp 1.000
7	PCB polos	Fiber glass single layer	Rp 20.000	1	Rp 20.000
8	Stok kontak	1 hole	Rp 8.000	2	Rp 16.000
9	Box	X6 180x110x60mm	Rp 9.500	1	Rp 9.500
10	Stekker	Broco	Rp 12.500	1	Rp 12.500
11	Kabel serabut	Eterna	Rp 8.000	1	Rp 8.000
12	Pelarut PCB	Fery Chloride	Rp 7.500	1	Rp 7.500
Total :				14	Rp 278.500

Tabel 3. Rincian biaya modul SSS (*socket sub system*)

No.	Nama Part	Model / Type	Harga	Jumlah (pcs)	Sub total
1	Power supply	Switching 5V/5A	Rp 36.000	1	Rp 36.000
2	Mikrokontroller	Wemos mini D1	Rp 34.000	1	Rp 34.000
3	Sensor arus	PZEM-004T	Rp 102.000	1	Rp 102.000
4	Sensor panas	DS18B20	Rp 12.000	1	Rp 12.000
5	Relay	1 Channel Opto 5V	Rp 12.000	1	Rp 12.000
6	Switch	On/Off	Rp 1.000	1	Rp 1.000
7	PCB polos	Fiber glass single layer	Rp 20.000	1	Rp 20.000
8	Stok kontak	2 hole	Rp 12.000	1	Rp 12.000
9	Box	X6 180x110x60mm	Rp 9.500	1	Rp 9.500
10	Stekker	Broco	Rp 12.500	1	Rp 12.500
11	Kabel serabut	Eterna	Rp 8.000	1	Rp 8.000
12	Pelaut PCB	Fery Chloride	Rp 7.500	1	Rp 7.500
Total :				12	Rp 266.500

Tabel 4. Rincian biaya modul PSS (*panel sub system*)

No.	Nama Part	Model / Type	Harga	Jumlah (pcs)	Sub total
1	Power supply	Switching 5V/5A	Rp 36.000	1	Rp 36.000
2	Mikrokontroller	Wemos mini D1	Rp 34.000	2	Rp 68.000
3	Sensor arus	PZEM-004T	Rp 102.000	2	Rp 204.000
4	Switch	On/Off	Rp 1.000	1	Rp 1.000
5	PCB polos	Fiber glass single layer	Rp 20.000	1	Rp 20.000
6	Box	X6 180x110x60mm	Rp 9.500	1	Rp 9.500
7	Stekker	Broco	Rp 12.500	1	Rp 12.500
8	Kabel serabut	Eterna	Rp 8.000	1	Rp 8.000
9	Pelaut PCB	Fery Chloride	Rp 7.500	1	Rp 7.500
Total :				11	Rp 366.500

Rekapitulasi Biaya Modul MACSys

1.	Modul ACSS	Rp. 299.500,-
2.	Modul LSSS	Rp. 278.500,-
3.	Modul SSS	Rp. 266.500,-
4.	Modul PSS	Rp. 366.500,-
Jumlah		Rp. 1.211.000,-