

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
PALCOMTECH

SKRIPSI

PERANCANGAN KONTROL PERALATAN ELEKTRONIK
RUMAH TANGGA BERBASIS INTERNET OF THINGS DAN
ANDROID



Diajukan Oleh:
CHANDRA EKO SAPUTRA
011180228P

Untuk Memenuhi Sebagian dari Syarat
Mencapai Gelar Sarjana Komputer

PALEMBANG
2022

**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
PALCOMTECH**

HALAMAN PENGESAHAN PEMBIMBING SKRIPSI

NAMA : CHANDRA EKO SAPUTRA
NOMOR POKOK : 011180228P
PROGRAM STUDI : S1 INFORMATIKA
JENJANG PENDIDIKAN : STRATA SATU (S1)
**JUDUL : PERANCANGAN KONTROL
PERALATAN ELEKTRONIK RUMAH
TANGGA BERBASIS INTERNET OF
THINGS DAN ANDROID**

Tanggal : 05 Maret 2022
Pembimbing

Mengetahui,
Ketua

Benedictus Effendi, S.T., M.T.
NIDN: 0221027002

Benedictus Effendi, S.T., M.T.
NIP: 09.PCT.13

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
PALCOMTECH

HALAMAN PENGESAHAN PENGUJI SKRIPSI

NAMA : CHANDRA EKO SAPUTRA
NOMOR POKOK : 011180228P
PROGRAM STUDI : S1 INFORMATIKA
JENJANG PENDIDIKAN : STRATA SATU (S1)
**JUDUL : PERANCANGAN KONTROL PERALATAN
ELEKTRONIK RUMAH TANGGA BERBASIS
INTERNET OF THINGS DAN ANDROID**

Tanggal : 05 Maret 2022
Penguji 2

Tanggal : 05 Maret 2022
Penguji 2

Mahmud, S.Kom., M.Kom.
NIDN: 0229128602

Rezania Agramanisti Azdy, S.Kom., M.Cs.
NIDN: 0215118601

Menyetujui,
Ketua

Benedictus Effendi, S.T., M.T
NIP : 09.PCT.13

MOTTO

*WHAT DOESN'T KILL YOU, MAKES YOU
STRONGER.*

(Friedrich Nietzsche)

Kupersembahkan kepada:

- *Orangtua ku Tercinta*
- *Istriku Febri Romadhona*
- *Saudara-saudaraku tersayang*
- *Para Pendidik yang kuhormati*

KATA PENGANTAR

Puji dan syukur Penulis panjatkan kepada Tuhan Yang Maha Esa atas segala berkat-Nya sehingga Penulis dapat menyelesaikan skripsi ini. Adapun selama penulisan dan penyusunan skripsi ini, Penulis mendapatkan banyak dukungan, semangat serta bimbingan dari berbagai pihak. Oleh karena itu, Penulis ingin mengucapkan terima kasih kepada berbagai pihak tersebut, yaitu kepada Ketua STMIK PalComTech sekaligus pembimbing skripsi saya, Bapak Benedictus Effendi, S.T., M.T., kepada Ketua Program Studi Informatika, Bapak Eka Prasetya Adhy Sugara, S.T., M.Kom., penguji proposal penulis, Bapak Alfred Tenggono, S.Kom., M.Kom. dan Bapak Surahmat S.Kom., M.Kom., kepada kedua orang tua Penulis yang tercinta, kepada istri penulis Ns. Febri Romadhona, S.Kep., dan kepada para sahabat yang selalu memberi semangat serta kepada semua pihak yang telah banyak memberi bantuan yang tidak bisa disebutkan satu per satu.

Demikian kata pengantar dari Penulis, semoga skripsi ini dapat bermanfaat dan berguna bagi para pembaca. Penulis menyadari bahwa penulisan skripsi ini masih memiliki banyak kekurangan dan kelemahan. Maka dari itu, saran dan kritik yang membangun sangat diperlukan untuk menghasilkan sesuatu yang lebih baik lagi. Terima kasih.

Palembang, Maret 2022

Penulis

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN PEMBIMBING SKRIPSI	ii
HALAMAN PENGESAHAN PENGUJI SKRIPSI	iii
HALAMAN MOTTO DAN PERSEMBAHAN	iv
KATA PENGANTAR	v
DAFTAR ISI	vi
DAFTAR GAMBAR	ix
DAFTAR TABEL	xi
DAFTAR LAMPIRAN	xii
ABSTRAK	xiii

BAB I PENDAHULUAN

1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Ruang Lingkup Penelitian	3
1.4 Tujuan	4
1.5 Manfaat	4

BAB II PERANCANGAN PERANGKAT DAN APLIKASI

2.1 Gambaran Umum Perangkat.....	6
----------------------------------	---

BAB III TINJAUAN PUSTAKA

3.1 Tinjauan Pustaka	9
3.2 Internet of Things (IoT)	10
3.3 Flowchart	11
3.4 <i>Unified Modeling Language</i> (UML)	12
3.4.1 Use Case Diagram	12
3.4.2 Activity Diagram	13
3.5 Message Queuing Telemetry Transport (MQTT).....	14
3.6 HiveMQ	16

3.7	NodeMCU ESP8266	16
3.8	Relay	17
3.9	Modul PZEM-004T	18
3.10	Sensor LDR (<i>Light Dependent Resistor</i>)	19
3.11	Energi Listrik dan Perhitungan Biaya	19
3.12	Android	21
3.13	Konektivitas	21
3.14	<i>Firestore Realtime Database</i>	22
3.15	Flutter	22

BAB IV METODE PENELITIAN

4.1	Lokasi dan Waktu Penelitian	25
4.1.1	Lokasi	25
4.1.2	Waktu Penelitian	25
4.2	Alat dan Bahan	26
4.2.1	Perangkat Lunak	26
4.2.2	Perangkat Keras	26
4.3	Metode Penelitian	26
4.3.1	Metode Prototype	27
4.3.2	Studi Pustaka	27
4.4	Diagram Alir Penelitian	29
4.5	Perancangan Alat	31
4.6	Perancangan Aplikasi	32
4.7	Perancangan Database	33

BAB V PEMBAHASAN

5.1	Implementasi dan Uji Coba Sistem	35
5.1.1	Implementasi Sistem	35
5.1.2	Use Case Diagram Sistem	37
5.1.3	Activity Diagram Sistem	38
5.1.4	Pengujian Sistem	44
5.1.5	Hasil Pengujian Sistem	45

BAB VI PENUTUP

6.1	Kesimpulan	53
6.2	Saran	53

DAFTAR PUSTAKA xiv

DAFTAR LAMPIRAN xv

DAFTAR GAMBAR

Gambar 2.1 Skema Rancangan Alat	7
Gambar 2.2 Rancangan Aplikasi Smartphone Android	8
Gambar 3.1 Contoh arsitektur MQTT (MQTT.org, 2020)	15
Gambar 3.2 NodeMCU ESP8266	17
Gambar 3.3 Relay 4 Channel	18
Gambar 3.4 Modul PZEM-004T	18
Gambar 3.5 Sensor LDR	19
Gambar 4.1 Diagram Prototipe	27
Gambar 4.2 Diagram Alir Penelitian.....	30
Gambar 4.3 Skema Rangkaian Alat Kontrol.....	31
Gambar 4.4 Aplikasi Kontrol dan Monitoring Peralatan Elektronik	32
Gambar 4.5 <i>Firestore Realtime Database Key-Value</i>	34
Gambar 5.1 <i>Flowchart</i> Sistem IoT.....	36
Gambar 5.2 Diagram Interaksi <i>User</i> Terhadap Sistem	37
Gambar 5.3 Activity Diagram Kontrol terhadap Sistem.....	38
Gambar 5.4 Activity Diagram Mode Otomatis Sistem	39
Gambar 5.5 <i>Activity Diagram</i> Kontrol Lampu Dalam Ruangan.....	40
Gambar 5.6 <i>Activity Diagram</i> Kontrol Kipas Angin	41
Gambar 5.7 <i>Activity Diagram</i> Kontrol Monitor.....	42
Gambar 5.8 <i>Activity Diagram</i> Kontrol <i>Rice Cooker</i>	43
Gambar 5.9 <i>Activity Diagram</i> Laman <i>Report</i>	44
Gambar 5.10 Menghidupkan Fungsi Otomatis pada Aplikasi	45

Gambar 5.11 Kondisi Lampu Luar Saat Kondisi Cahaya Terang.....	46
Gambar 5.12 Kondisi Lampu Saat Kondisi Cahaya Gelap	47
Gambar 5.13 Kondisi Lampu Luar Saat Mode Manual	48
Gambar 5.14 Kondisi Lampu Dalam Saat Dihidupkan.....	48
Gambar 5.15. Kondisi Kipas Angin Saat Dihidupkan	49
Gambar 5.16. Kondisi Televisi Saat Dihidupkan.....	49
Gambar 5.17. Kondisi Televisi Saat Dihidupkan.....	50
Gambar 5.18. Kondisi Semua Perangkat Saat Dihidupkan.....	51
Gambar 5.19. Hasil Perekaman PZEM-004t.....	52

DAFTAR TABEL

Tabel 3.1 Perbandingan Metode Penelitian.....	9
Tabel 3.2 Simbol Flowchart	11
Tabel 3.3 Tarif dasar Listrik.....	20
Tabel 4.1 Waktu Penelitian	25
Tabel 4.2 Perangkat Keras yang digunakan	26

DAFTAR LAMPIRAN

1. Lampiran 1. *Form* Topik dan Judul (*Fotocopy*).
2. Lampiran 2. *Form* Konsultasi (*Fotocopy*)
3. Lampiran 3. Surat Pernyataan (*Fotocopy*)
4. Lampiran 4. *Form* Revisi Proposal (*Fotocopy*)
5. Lampiran 5. *Form* Revisi Ujian Kompre (Asli)
6. Lampiran 6. *Listing Code*

ABSTRACT

CHANDRA EKO SAPUTRA. *Design Household Electronics Control Based on Internet of Things and Android.*

The need for electrical energy is very large, not comparable to the current availability. Therefore, the Indonesian government urges the use of electrical energy to be more efficient. The behavior of people who often forget and neglect to turn off electronic equipment when leaving the house is also one of the causes of inefficient use of electrical energy. Based on that problem, a system to remotely controlling household electronic equipment and monitoring the use of electrical energy and costs incurred when the owner is not at home is needed. The IoT concept can be applied to overcome this as well as android devices that are commonly owned by the public.

Keyword: *Internet of Things (IoT), Android, Control, Electricity Cost.*

ABSTRAK

CHANDRA EKO SAPUTRA. Perancangan Kontrol Peralatan Elektronik Rumah Tangga Berbasis IoT dan Android.

Kebutuhan listrik yang besar, tidak sebanding dengan ketersediaan energi listrik yang ada saat ini. Oleh sebab itu pemerintah Indonesia menghimbau untuk konsumsi listrik yang lebih efisien. Perilaku masyarakat yang sering lupa dan lalai untuk mematikan peralatan elektronik saat meninggalkan rumah juga menjadi salah satu penyebab pemakaian energi listrik yang tidak efisien ini. Berdasarkan hal tersebut diperlukan suatu sistem untuk memantau dan mengontrol peralatan elektronik rumah tangga tersebut dari jarak jauh saat pemilik sedang tidak berada dirumah dan memantau penggunaan energi listrik serta biaya yang dikeluarkan. Hal ini dapat dilakukan dengan menggunakan konsep Internet of Things (IoT) dan perangkat Android yang sudah umum dimiliki masyarakat.

Kata Kunci: *Internet of Things (IoT), Android, Kontrol, Biaya listrik.*

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kebutuhan listrik yang besar, tidak sebanding dengan ketersediaan energi listrik yang ada saat ini. Sehingga penting untuk melakukan hemat energi listrik. Penghematan energi listrik selain bisa mengurangi pengeluaran keuangan ternyata juga bersifat sosial, yaitu membantu mengurangi resiko pemanasan global (Maharmi et al, 2018), (Wijayanti & Sriyeni, 2021), (Aprizal et al., 2019).

Pemerintah Indonesia mencatat konsumsi listrik Indonesia pada 2018 sebesar 1,06 Mega Watt hour (MWh) per kapita. Konsumsi listrik ini selalu meningkat setiap tahunnya menyebabkan pemerintah untuk menghimbau konsumsi listrik yang lebih efisien (Hamdani, Budiarto, dan Hadi 2020). Berdasarkan data dari Badan Pusat Statistik (BPS), konsumsi listrik pada tahun 2020 juga mengalami kenaikan menjadi 1.09 MWH per kapita (Badan Pusat Statistik 2020), (Orlando & Sriyeni, 2021), (Fathullah & Aprizal, 2020).

Pemerintah secara berkelanjutan mengampanyekan hemat energi listrik dalam berbagai kegiatan. Intisari kampanye pemerintah tersebut adalah sosialisasi kebiasaan hemat energi dari lingkungan yang terkecil yakni keluarga. Upaya lain dari pemerintah dalam menjaga kestabilan energi nasional yang dilakukan melalui upaya peningkatan pasokan dan produksi energi listrik. Para pengamat energi menyatakan bahwa kontribusi masyarakat khususnya sektor rumah tangga dalam upaya penghematan

energi listrik adalah cukup besar (Santoso dan Salim 2019). Konservasi energi listrik untuk penggunaan energi yang lebih efisien akan mengurangi konsumsi energi listrik sehingga dengan adanya konservasi energi listrik akan berdampak kepada berkurangnya biaya listrik yang terpakai.

Perilaku masyarakat yang sering lupa untuk mematikan peralatan elektronik saat meninggalkan rumah, menyebabkan pemborosan energi listrik yang berlebihan (Rahman dan Imelda 2020), (Yuliansyah et al., 2023), (Yasermi Syahrul, 2021).

Faktor kesibukan dan kelalaian dari konsumen listrik yang banyak pergi bekerja meninggalkan rumah dari pagi dan terkadang pulang larut malam sehingga penerangan rumah yang ditempatkan diteras atau halaman, ruang tamu, kamar tidur, kamar mandi, dapur, garasi atau gudang dan penerangan lainnya di rumah yang sering terlambat mematikan di pagi hari karena faktor kesibukan atau bahkan sampai lupa untuk mematikan sehingga lampu hidup seharian, akibatnya pemakaian daya listrik setiap bulannya semakin tinggi dan biaya listrik yang harus dikeluarkan karena pemakaian yang tidak efisien ini mahal, padahal pemilik rumah jarang berada dirumah pada saat hari efektif karena faktor pekerjaan. Hal ini tentu saja akan memberatkan konsumen listrik ditambah dengan kenaikan tarif daya listrik per kWh nya semakin naik (Maharmi et al, 2018), (Puspita & Annisa, 2021), (Adelin & Adam Jaya, 2017), (Andrian & Widyanto, 2020).

Berdasarkan hal tersebut diperlukan suatu sistem untuk memantau dan mengontrol peralatan elektronik rumah tangga tersebut dari jarak jauh saat pemilik sedang tidak berada di rumah sehingga dapat lebih mengefisienkan pemakaian energi listrik dan juga dapat memantau peralatan elektronik apa

saja yang lupa dimatikan oleh pemilik rumah. Dalam penelitian ini penulis mengusulkan ”*Perancangan Kontrol Peralatan Elektronik Rumah Tangga Berbasis Internet of Things dan Android*”.

1.2 Rumusan Masalah

Berdasarkan latar belakang penelitian, permasalahan dalam penelitian ini adalah:

1. Bagaimana merancang sistem kontrol peralatan elektronik rumah tangga berbasis IoT dan Android menggunakan NodeMCU sebagai mikrokontroler dengan metode MQTT.
2. Bagaimana merancang sistem untuk mengukur biaya penggunaan listrik dari peralatan elektronik rumah tangga.
3. Bagaimana merancang sebuah aplikasi *smartphone* Android untuk mengontrol dan memantau peralatan elektronik rumah tangga.

1.3 Ruang Lingkup Penelitian

Adapun ruang lingkup dari penelitian ini adalah sebagai berikut:

1. Perangkat yang dibangun berupa prototipe alat kontrol jarak jauh dan aplikasi *smartphone* Android.
2. Kontrol dilakukan dengan aplikasi yang akan dibuat untuk *smartphone* Android.
3. Menggunakan MQTT broker HiveMQ dan Firebase *Realtime Database* sebagai server penyimpanan data pengukuran sensor dan status peralatan elektronik.

4. Mikrokontroler yang digunakan adalah NodeMCU.
5. Metode pengiriman data menggunakan metode MQTT.
6. Menggunakan modul PZEM-004T untuk pengukuran nilai kWh yang digunakan pada peralatan elektronik
7. Menggunakan sensor LDR (*Light Dependant Resistor*) untuk pengukuran intensitas cahaya pada prototipe alat dan relay sebagai saklar *on/off*.

1.4 Tujuan

Berdasarkan permasalahan diatas, Adapun tujuan penelitian ini adalah sebagai berikut:

1. Mengembangkan sistem kontrol jarak jauh berbasis IoT yang dapat membantu pemilik rumah mengontrol peralatan elektronik rumah tangga saat tidak berada dirumah agar pemakaian listrik menjadi lebih efisien.
2. Mengembangkan sistem untuk yang dapat mencatat pemakaian listrik dan biaya yang akan dikeluarkan..
3. Memanfaatkan *smartphone* Android sebagai media untuk mengontrol dan memantau peralatan elektronik rumah tangga serta pemakaian dan biaya listrik yang digunakan.

1.5 Manfaat

Manfaat yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Dapat mengontrol dan memantau peralatan elektronik rumah tangga dari jarak jauh menggunakan jaringan internet.
2. Dapat mengetahui penggunaan dan biaya pemakaian listrik dari peralatan elektronik.
3. Mengurangi penggunaan listrik yang tidak efisien dikarenakan pemilik rumah lupa mematikan peralatan elektronik saat tidak berada dirumah.

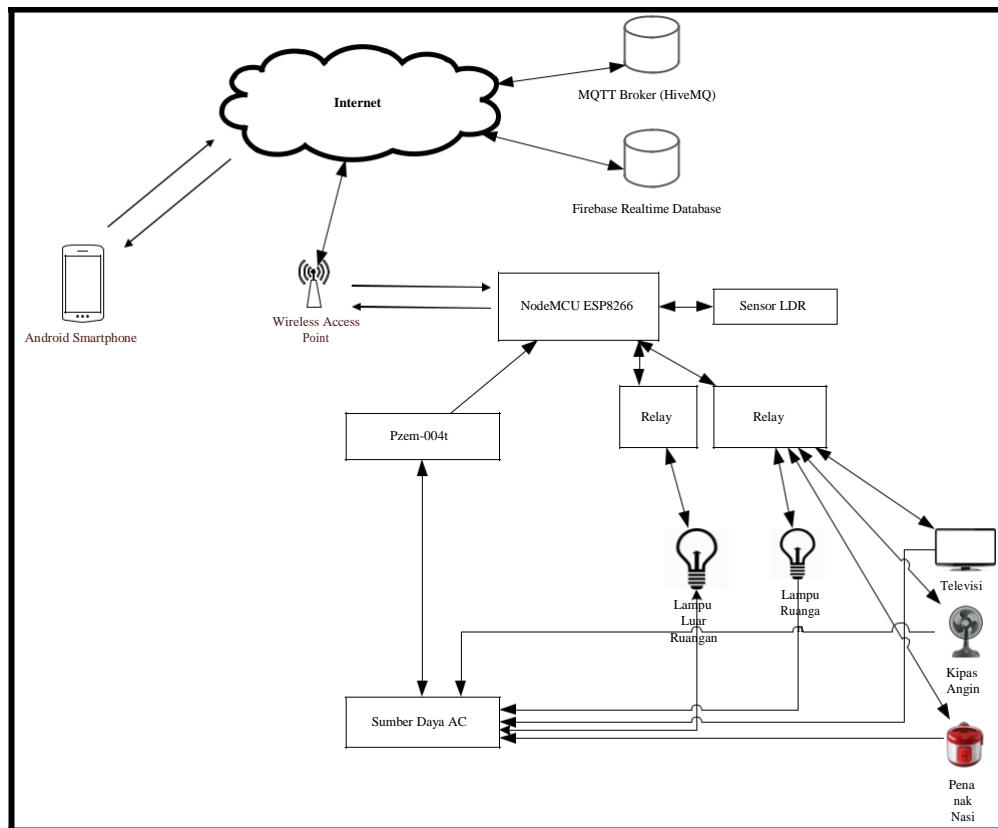
BAB II

PERANCANGAN PERANGKAT DAN APLIKASI

2.1 Gambaran Umum Perangkat

Perangkat yang dikembangkan terdiri dari NodeMCU ESP8266 sebagai mikrokontroler yang sudah terpasang modul wifi, sensor LDR (*Light Dependent Resistor*) untuk mendeteksi intensitas cahaya untuk mode otomatis pada lampu, sensor PZEM-004T untuk mengukur arus serta tegangan yang digunakan, dan relay sebagai saklar digital (Melani & Mahmud, 2020b).

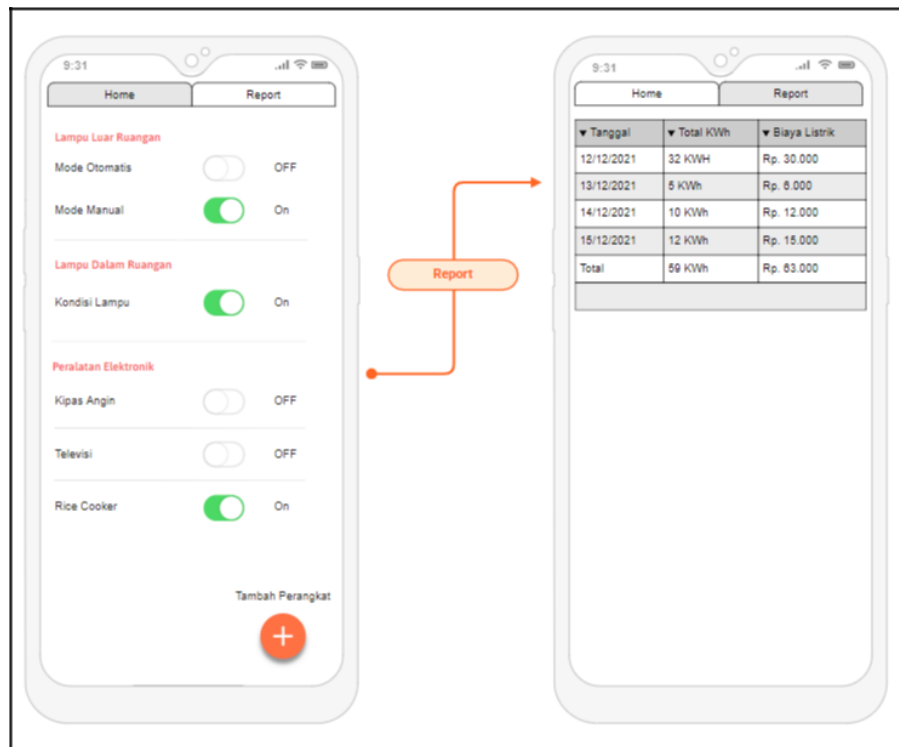
Peralatan elektronik yang menjadi objek kontrol dari penelitian ini adalah 2 buah lampu LED, 1 Televisi, 1 *Rice Cooker* dan 1 kipas angin. Setiap peralatan terhubung dengan relay yang berfungsi sebagai saklar digital untuk *on/off* dari peralatan tersebut. Skema dari peralatan yang dikembangkan dapat dilihat pada Gambar 2.1.



Sumber: diolah sendiri

Gambar 2.1 Skema Rancangan Alat

Perangkat NodeMCU ESP8266 akan terkoneksi ke Wifi Access Point dengan menggunakan protokol MQTT untuk transfer data antar perangkat, dalam hal ini perangkat yang digunakan untuk monitoring dan kontrol peralatan adalah aplikasi android yang akan dikembangkan menggunakan framework flutter dengan minimum SDK adalah Android Oreo (API level 27). Database yang akan digunakan sebagai penyimpanan data penggunaan listrik (kWh) dan biaya yang dikeluarkan dari penggunaan listrik tersebut menggunakan database NoSQL dari Firebase Realtime Database. Rancangan aplikasi dapat dilihat pada Gambar 2.2.



Sumber: diolah sendiri

Gambar 2.2 Rancangan Aplikasi Smartphone Android

BAB III

TINJAUAN PUSTAKA

3.1 Tinjauan Pustaka

Tinjauan Pustaka pada penelitian ini adalah mencari referensi dari beberapa sumber yang berkaitan. Pada Tabel 3.1 berikut adalah referensi yang berkaitan dengan judul penelitian.

Tabel 3.1 Perbandingan Metode Penelitian

No	Penulis	Studi Kasus	Objek	Teknologi	Interface
1	Nugraha et al., 2019	Memantau dan mengendalikan pompa air aquarium, TV CRT dan Dispenser	Pompa air Aquarium, TV CRT, dan Dispenser	Sensor Arus ACS712, Sensor tegangan ZMBT101B, Firebase Realtime Database, Arduino uno dan NodeMCU V3	Android
2	Hamdani et al., 2020	Mengendalikan Lampu secara manual.	Lampu dengan sensor pir	ESP8266 V1, Sensor pir, mqtt	WEB
	Rahman, 2019	Memantau dan mengendalikan lampu, kipas, pompa air, dan TV dengan protokol http	Lampu, Kipas, Pompa Air, dan TV	NodeMCU, Thingspeak	WEB
4	Usulan	Memantau dan mengendalikan Lampu, TV, Kipas, dan Rice Cooker	Lampu, Kipas, TV, dan Rice	Firestore Realtime Database, HiveMQ	Android

		serta mengukur penggunaan kWh dan biaya listrik yang digunakan	Cooker	MQTT, NodeMCU, LDR Sensor, Flutter SDK	
--	--	--	--------	--	--

3.2 Internet of Things (IoT)

Internet of Things (IoT) merupakan bentuk koneksi suatu perangkat yang saling terhubung dan mampu menghasilkan informasi yang dapat diakses dan digunakan oleh manusia atau sistem lainnya. Ide awal IoT pertama kali dimunculkan oleh Kevin Ashton pada tahun 1999 dimana benda-benda disekitar kita dapat berkomunikasi antara satu sama lain melalui sebuah jaringan internet (Tanto dan Darmuji 2019). Konsep IoT digunakan secara luas untuk menghubungkan berbagai perangkat *embedded* karena mendukung teknologi jarak dekat maupun jauh serta banyak protokol dan algoritma. (Iqbal et al. 2018), (Fitria & Adelin, 2021), (Muchlisin & Widyanto, 2021), (Cahyati & Sriyeni, 2021), (Purnama & Aprizal, 2020).


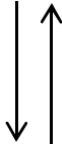

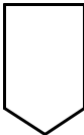


Sebuah sistem kontrol berbasis IoT pada peralatan elektronik rumah tangga membutuhkan sebuah alat untuk menerima data dari sensor dan mengirimkan data ke sistem kontrolnya melalui jaringan internet agar dapat di kontrol dari mana saja dan kapan saja. Terdapat beberapa alat yang dapat memenuhi kebutuhan tersebut, salah satunya adalah menggunakan mikrokontroler NodeMCU yang sudah dilengkapi dengan modul wifi ESP8266. Terdapat beberapa protokol untuk pengiriman data dengan NodeMCU ini salah satunya adalah Message Queuing Telemetry Transport (MQTT).


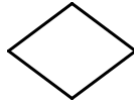
3.3 Flowchart

Flowchart merupakan bagan yang menunjukkan alir di dalam suatu program atau prosedur sistem secara logika. Bagan alir digunakan terutama untuk alat bantu komunikasi dan untuk dokumentasi (Anggiawan, Pandie,

dan Boru 2018), (Syafitri & Annisa, 2022), (Handayani & Adelin, 2020), (Nugraha & Widyanto, 2021), (Saputra & Aprizal, 2021), (Setiawan & Wiza, 2021). Untuk menggambarkan suatu *flowchart*, terdapat banyak simbol-simbol yang harus diketahui antara lain dapat dilihat pada Tabel 3.2.

Tabel 3.2 Simbol Flowchart

Simbol	Nama	Keterangan
	<i>Terminator Symbol</i>	Simbol untuk permulaan (<i>start</i>) atau akhir (<i>stop</i>) dari suatu kegiatan
	<i>Flow Direction Symbol</i>	Simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain.
	Simbol <i>Input- Output</i>	Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya.
	<i>Off- connector Symbol</i>	Simbol untuk keluar-masuk atau penyambungan proses pada lembar/ halaman yang berbeda
	Simbol Dokumen	Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas
	<i>Processing</i>	Simbol yang menunjukkan pengolahan

Simbol	Nama	Keterangan
	<i>Terminator</i> <i>Symbol</i>	Simbol untuk permulaan (<i>start</i>) atau akhir (<i>stop</i>) dari suatu kegiatan
	<i>Symbol</i>	data yang dilakukan komputer
	<i>Decision</i> <i>Symbol</i>	Simbol pemilihan proses berdasarkan kondisi yang ada

3.4 Unified Modeling Language (UML)

Unified Modeling Language (UML) merupakan sebuah metode yang digunakan untuk perancangan sistem aplikasi berbasis *Object Oriented Programming (OOP)*. UML memiliki tujuan untuk menganalisa proses bisnis lalu memodelkannya ke dalam notasi-notasi tertentu. Notasi tersebut dapat menggambarkan setiap *entity* pada OOP dan juga relasi di antaranya. Penggambaran sistem pada UML dapat dilakukan secara dinamis ataupun statis. Pada gambaran dinamis, sistem menggunakan *use case diagram*, *activity diagram*, dan *sequence diagram*. Sedangkan pada gambaran statis, sistem menggunakan *class diagram*.

3.4.1 Use Case Diagram

Diagram *Use Case* pada dasarnya digunakan untuk mengumpulkan persyaratan dan fungsionalitas sistem yang ditangkap dalam *use case* dan juga untuk mengidentifikasi agen internal dan eksternal yang berinteraksi dengan sistem. Agen yang dimaksud disini adalah *user*. *User* atau pengguna ini bisa

perorangan, sebuah organisasi, atau sistem eksternal, asal berinteraksi dengan sistem atau aplikasi.

Sistem menyatakan batasan dalam relasi dengan *user* yang menggunakannya (di luar sistem) dan fitur-fitur yang harus disediakan (dalam sistem). *Use case* sendiri adalah gambaran fungsional dari sebuah sistem. Dengan demikian, antara konsumen dan juga pengguna pada sistem tersebut, akan mengerti atau paham mengenai fungsi sistem yang dibangun (Sri Handayani & Adelin, 2019), (Anggraini & Widyanto, 2021), (Sriyeni & Veronica, 2020), (Setiawan et al., 2019). *Use case* memiliki empat macam relasi yaitu asosiasi (*association*), generalisasi (*generalization*), dipedensi (*dependency*) dan agregasi (*aggregation*).

3.4.2 Activity Diagram

Untuk menggambarkan aspek dinamis dari sistem, diperlukan diagram *activity*. Diagram *activity* dapat menggambarkan aktivitas, objek data yang diproduksi serta urutan eksekusi Tindakan yang berbeda (Febrianty et al., 2020), (Handayani & Adelin, 2019), (Syarifudin & Widyanto, 2022), (Irawan & Sriyeni, 2021), (Setiawan & Hartati, 2022). Pada diagram ini, *edge* berfungsi sebagai pengatur laju pengerjaan dari *node* dalam sebuah aktivitas. Sebuah *node* tidak akan memulai pengerjaannya sampai menerima kontrol atau *input* dari setiap jalur. Apabila sebuah *node* telah menyelesaikan perhitungannya maka kontrol *execution* akan berpindah ke *node* yang ada pada arus keluarannya. Pengerjaan pada

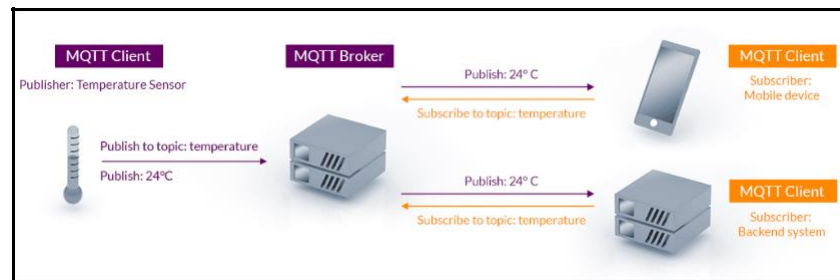
diagram *activity* akan berakhir jika telah sampai pada node terakhir dan atau Kembali ke objek data sebagai sebuah hasil dari komputasi *internal*. Untuk

menentukan suatu keputusan terdapat sebuah *decision node* yang berbentuk berlian dan setidaknya memiliki satu *node* masuk dan dua *node* yang keluar. Setiap *edge* keluar dari *decision node* dikaitkan dengan kondisi *Boolean* sedemikian rupa sehingga *edge* dipilih jika dan hanya jika kondisi terkait berlaku.

3.5 Message Queuing Telemetry Transport (MQTT)

MQTT merupakan sebuah protokol komunikasi data *machine to machine* yang berada pada layer aplikasi. Protokol ini berkomunikasi dengan mengirimkan data pesan dengan ukuran yang relative kecil yaitu hanya sebesar 2 *bytes* untuk setiap jenis data, sehingga protokol ini sangat cocok di implementasikan pada lingkungan yang terbatas sumber dayanya seperti *bandwidth* dan listrik. Protokol ini juga menjamin terkirimnya semua pesan walaupun koneksi terputus sementara. MQTT menerapkan metode *publish/subscribe* sebagai metode komunikasinya (Hamdani et al. 2020), (Melani & Mahmud, 2020a), (Adelin & Fatmariansi, 2012), (Usamah & Setiawan, 2022).

Metode *publish/subscribe* merupakan sebuah pola pertukaran pesan di dalam komunikasi jaringan dimana pengirim data disebut *publisher* dan penerima data disebut dengan *subscriber*. Metode ini memiliki beberapa kelebihan salah satunya yaitu *loose coupling* atau *decouple* dimana berarti antara *publisher* dan *subscriber* tidak saling mengetahui keberadaanya. Gambar 3.1 merupakan contoh arsitektur dari protokol MQTT.



Sumber: www.mqtt.org

Gambar 3.1 Contoh arsitektur MQTT (MQTT.org, 2020)

MQTT memiliki 3 level *Quality of Service* (QoS) dalam pengiriman data, yaitu:

1. QoS 0 (*At Most Once*): Pesan dikirimkan dengan upaya terbaik dari TCP/IP. Pesan hanya dikirimkan satu kali atau tidak sama sekali.
2. QoS 1 (*At Least Once*): Menjamin pesan akan dikirimkan setidaknya satu kali.
3. QoS 2 (*Exactly Once*): Pesan pasti tersampaikan dengan tepat dalam satu kali.

Terdapat protokol lain yang dapat mendukung pembuatan sistem IoT seperti HTTP (*HyperText Transfer Protocol*) dan CoAP (*Constrained Application Protocol*). Kelebihan protokol MQTT dengan HTTP adalah dapat digunakan pada lingkungan dengan *bandwidth* internet yang kecil karena lebar data yang dikirimkan pada protokol MQTT lebih kecil. Hal tersebut juga menyebabkan delay yang lebih rendah untuk protokol MQTT ini (Windryani, K, dan Mayasari 2019), (Effendi et al., 2021) dan (Megawati & Setiawan, 2022). Sementara CoAP memiliki keunggulan dalam mengatasi permasalahan delay dalam pengiriman data, protokol ini tidak dapat mengungguli protokol MQTT pada parameter lain

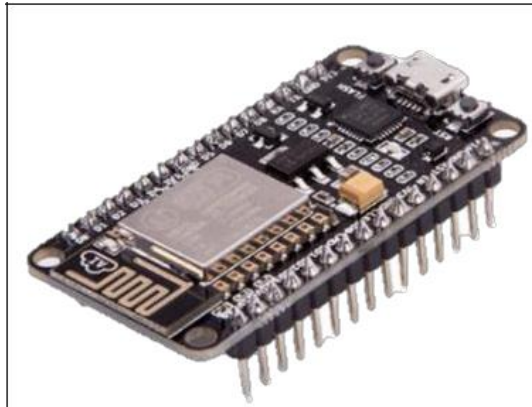
seperti *throughput*, *packet received*, dan *power consumption* (Pratama, Munadi, dan Syauqi 2018), (E. F. Hartati, 2016). Atas pertimbangan tersebut penulis memilih protokol MQTT untuk sistem yang akan diteliti.

3.6 HiveMQ

HiveMQ merupakan salah satu MQTT broker dan platform *client based messaging* yang dirancang untuk kecepatan, efisiensi, serta perpindahan data yang handal untuk menghubungkan ataupun terhubung ke perangkat IoT. HiveMQ menyediakan aplikasi yang dapat kita *download* dan *install* pada *server* sendiri dan fitur Cloud MQTT broker dimana kita dapat menggunakan MQTT broker tersebut secara gratis berbasis cloud. Dalam penelitian ini, penulis menggunakan fitur Cloud HiveMQ.

3.7 NodeMCU ESP8266

NodeMCU merupakan pengembangan dari modul ESP8266 dengan firmware berbasis e-Lua. Pada NodeMCU dilengkapi dengan *micro usb port* yang berfungsi untuk pemrograman maupun *power supply*. Perangkat ini menggunakan Bahasa pemrograman Lua yang merupakan *package* dari modul ESP8266. Bahasa Lua memiliki logika dan susunan pemrograman yang sama dengan C dengan perbedaan pada *syntax* (Siswanto, Nurhadiyan, dan Junaedi 2020). Gambar 3.2 merupakan salah satu dari model NodeMCU ESP8266.



Sumber: (Atmoko 2019)

Gambar 3.2 NodeMCU ESP8266

3.8 Relay

Relay merupakan sebuah komponen elektronika yang dipergunakan untuk menghubungkan atau memutuskan arus listrik yang besar menggunakan arus listrik yang kecil. Prinsip yang digunakan pada relay adalah prinsip electromagnet dimana arus lemah yang mengalir dari kumparan inti besi lunak akan menjadi magnet (E. F. Hartati, 2016), (Ike Melani & Mahmud, 2021), (Fatmariansi & Saputro, 2019). Magnet yang terbentuk dari inti besi tersebut akan menarik jangkar besi yang menyebabkan kontak saklar akan terhubung dan arus listrik dapat mengalir. Pada saat arus lemah yang masuk melalui kumparan di putuskan, maka saklar akan terputus.

Relay terdiri dari *coil* dan *contact*. *Coil* merupakan gulungan kawat yang mendapatkan listrik, sedangkan *contact* adalah sejenis saklar yang dipengaruhi dari ada tidaknya arus listrik pada *coil* (Dewi, Rohmah, dan Zahara 2019). Pada Gambar 3.3 merupakan contoh sebuah relay dengan 4 *channel*.



Sumber: (Bolor 2015)

Gambar 3.3 Relay 4 Channel

3.9 Modul PZEM-004T

Modul PZEM-004T (Mardiana & Hartati, 2019), (Putri & Meilani, 2016), (Febrianty & Fatmariyani, 2018) adalah modul yang terintegrasi dengan sensor arus (CT) dan sensor tegangan. Modul ini dapat mengukur arus, energi, tegangan dan daya yang terdapat pada sebuah rangkaian listrik. Modul PZEM-004T membutuhkan catu daya 5 VDC dan menggunakan komunikasi serial RX TX untuk menerima serta mengirim data ke mikrokontroller (Alfian et al. 2021). Gambar 3.4 merupakan contoh modul PZEM-004T.

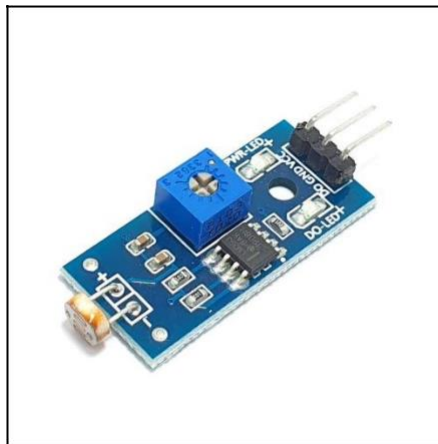


Sumber: <https://mikroavr.com/sensor-pzem-004t-arduino/>

Gambar 3.4 Modul PZEM-004T

3.10 Sensor LDR (Light Dependent Resistor)

Sensor LDR berfungsi untuk menghantarkan arus listrik jika menerima sejumlah intensitas cahaya. Semakin banyak cahaya yang mengenai sensor LDR, maka hambatannya akan menurun, dan jika semakin sedikit cahaya yang mengenai sensor LDR, maka hambatannya akan membesar. Sensor LDR ini berfungsi untuk mendeteksi adanya cahaya atau tidak sebagai indikator untuk *on/off* lampu otomatis. Gambar 3.5 merupakan contoh dari sensor LDR (E. Hartati & Aprizal, 2023).



Sumber: <https://ecadio.com/>

Gambar 3.5 Sensor LDR

3.11 Energi Listrik dan Perhitungan Biaya

Energi listrik merupakan akibat dari muatan listrik yang bergerak, yang disebut arus listrik (I). Energi listrik banyak dimanfaatkan untuk kebutuhan sehari-hari seperti menyalakan lampu, mengisi daya baterai handphone, menghidupkan Komputer, dan lain sebagainya. Energi listrik yang sampai pada rumah mengalami proses Panjang. Sebagian besar,

produksi listrik di Indonesia dilakukan oleh Perusahaan Listrik Negara

(PLN) dengan rumus sebagai berikut:

$$W = P \times \left(\frac{t}{360}\right)$$

$$\text{Biaya Listrik} = \text{TDL} \times \left(\frac{W}{1000}\right)$$

Keterangan:

W = Energi Listrik (*Watt hour*)

P= Beban Listrik (*Watt*)

t= waktu (*s*)

TDL = Tarif dasar Listrik per kWh

TDL ditetapkan oleh Kementerian Energi dan Sumber Daya Mineral (ESDM) dan diterapkan oleh PT PLN Persero (PT PLN Persero 2022) ditunjukkan pada Tabel 3.3:

Tabel 3.3 Tarif dasar Listrik

NO	GOL. TARIF	BATAS DAYA	REGULER (PASCA BAYAR)		PRABAYAR (Rp/kWh)
			BIAYA BEBAN (Rp/kVA/bulan)	BIAYA PEMAKAIAN (Rp/kWh) dan BIAYA kVArh (Rp/kVArh)	
1.	R-1/TR	900 VA-RTM	*)	1.352,00	1.352,00
2.	R-1/TR	1.300 VA	*)	1.444,70	1.444,70
3.	R-1/TR	2.200 VA	*)	1.444,70	1.444,70
4.	R-2/TR	3.500 VA s.d 5.500 VA	*)	1.444,70	1.444,70
5.	R-3/TR	6.600 VA ke atas	*)	1.444,70	1.444,70
6.	B-2/TR	6.600 VA s.d. 200 kVA	*)	1.444,70	1.444,70

Tarif yang digunakan pada penelitian ini adalah Golongan tarif R-1/TR dengan batas daya sebesar 1.300 VA dan biaya per kWh nya sebesar 1.444,70.

3.12 Android

Android merupakan sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android sangat terbuka bagi pengembang untuk membuat aplikasi sendiri melalui platformnya. Android berasal dari Android Inc. yang merupakan pembuat piranti lunak untuk ponsel yang kemudian dibeli oleh Google Inc dan melakukan perilisan perdananya pada tahun 2007 bersama dukungan dari *Open Handset Alliance* (Syifa, Prayoga, dan Amanaf 2020). Pada penelitian ini, penulis membuat aplikasi android dengan dukungan versi terendah adalah Android 8.0 Oreo.

3.13 Konektivitas

Konektivitas *wireless* untuk menghubungkan NodeMCU ESP8266 dalam penelitian ini menggunakan modem Wireless ZTE ZXHN F609 yang didapatkan dari berlangganan internet Indihome dengan lebar *bandwidth* sebesar 10 MBPS. Modem ini memiliki spesifikasi antara lain sebagai berikut:

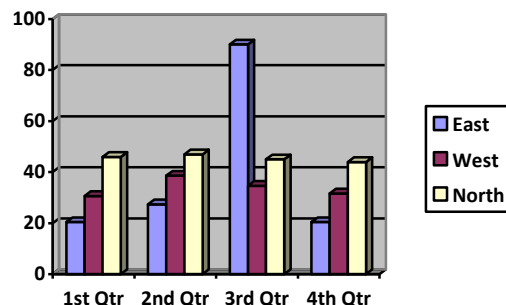
1. Frekuensi: 2.4 GHz
2. *Support* IEEE 802.11b/g/n
3. 128 bits WEP *data encryption*

4. WPA/WPA2 Security

3.14 Firebase Realtime Database

Firebase Realtime Database merupakan *database* berbasis NoSQL yang di-*hosting* di *cloud* dan dapat digunakan untuk menyimpan dan menyinkronkan data antar pengguna secara *realtime*. Data yang tersimpan pada database ini berbentuk JSON file dan memungkinkan pengguna untuk membangun aplikasi kolaboratif dan kaya fitur dengan menyediakan akses yang aman ke *database*, langsung dari kode sisi klien. Data dipertahankan secara lokal, dan meskipun sedang dalam kondisi *offline*, realtime event terus dipicu sehingga *end user* akan merasakan pengalaman yang lebih *responsive*. Ketika koneksi perangkat pulih kembali, *Realtime Database* akan menyinkronkan perubahan data lokal dengan pembaruan jarak jauh yang terjadi selama klien *offline*, sehingga setiap perbedaan akan otomatis dihilangkan.

3.15 Flutter



Flutter

merupakan

sebuah SDK yang dikembangkan oleh google untuk membangun aplikasi yang memiliki kinerja yang tinggi dan tampilan (UI) *User Interface* aplikasi yang dihasilkan menarik serta dapat dipublikasi ke *platform Android* dan iOS dari *codebase* tunggal. Kinerja dari *flutter* sama halnya dengan *native*

framework. Flutter menarik untuk digunakan atau dipelajari karena menggunakan bahasa pemrograman Dart.

Flutter secara teknis mempunyai dua fungsi yaitu *Flutter Framework* dan *Flutter SDK*. *Flutter Framework* merupakan sebuah Framework dari bahasa pemrograman Dart yang menyediakan fungsi dan elemen UI atau disebut *Widget* di dalam *Flutter*. *Flutter SDK* merupakan sekumpulan alat yang digunakan untuk mengembangkan/membangun aplikasi iOS ataupun Android. Pada *framework* ini, semua kodenya di compile dalam bentuk *native* (Android NDK, LLVM, AOT-compiled) tanpa ada *interpreter* pada prosesnya sehingga proses *compile*-nya menjadi lebih cepat.

Komponen utama Flutter yaitu:

a. *Flutter engine*

Flutter engine ditulis dengan bahasa pemrograman C++, memberikan dukungan tingkat rendah menggunakan library grafik Skia milik Google. Selain itu, flutter engine juga berinteraksi dengan pengembang perangkat lunak (SDK) *platform-specific* seperti yang disediakan oleh android dan IOS.

b. *Foundation library*

Foundation library, ditulis dengan bahasa pemrograman Dart, menyediakan fungsi dan *class-class* dasar yang digunakan untuk membangun aplikasi menggunakan flutter. Seperti API untuk berkomunikasi dengan engine.

c. *Widget*

Framework Flutter berisi dua set *widget* yaitu Material Design dan Cupertino. *Widget Material Design* menerapkan bahasa desain Google

dengan nama yang sama, sedangkan widget 'Cupertino' meniru desain dari IOS milik Apple.

BAB IV

METODE PENELITIAN

4.1 Lokasi dan Waktu Penelitian

4.1.1 Lokasi

Tempat penelitian ini dilakukan di rumah dengan alamat Komplek TPI Blok B1 No 4 Indralaya, Ogan Ilir, Sumatera Selatan. Lokasi ini dipilih sebagai tempat penelitian, karena merupakan rumah tempat tinggal penulis sehingga apabila terjadi sesuatu selama penelitian akan menjadi tanggung jawab sepenuhnya dari penulis.

4.1.2 Waktu Penelitian

Waktu yang dibutuhkan oleh penulis dalam Perancangan kontrol lampu dan peralatan elektronik rumah tangga berbasis IoT dan Android dapat dilihat pada Tabel 4.1.

Tabel 4.1 Waktu Penelitian

No	Kegiatan	2021								2022							
		November				Desember				Januari				Februari			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Studi literatur																
2	Perancangan Prototipe																
3	Penentuan alat dan bahan																
4	Pembuatan prototipe dan aplikasi																
5	Pengujian sistem																
6	Pengumpulan dan analisa data																
7	pembuatan laporan																

4.2 Alat dan Bahan

Peralatan dan bahan yang digunakan dalam penelitian ini terbagi menjadi 2 yaitu perangkat lunak dan perangkat keras.

4.2.1 Perangkat Lunak

Perangkat lunak yang digunakan pada penelitian ini adalah Arduino IDE untuk memprogram NodeMCU ESP8266 dan VSCode untuk membuat aplikasi *android* dengan menggunakan *framework flutter*.

4.2.2 Perangkat Keras

Perangkat keras yang digunakan dalam penelitian ini antara lain ditunjukkan pada Tabel 4.2.

Tabel 4.2 Perangkat Keras yang digunakan

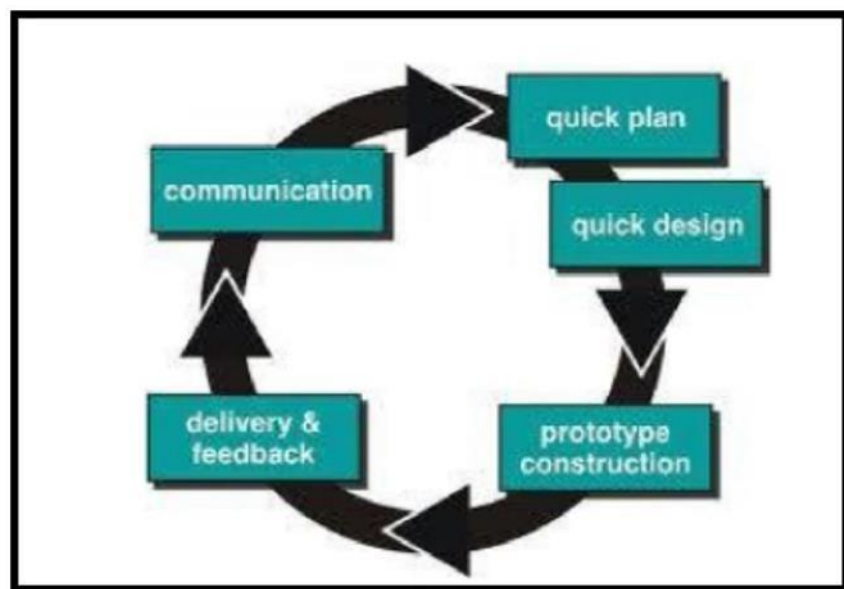
No	Nama	Jumlah
1	NodeMCU ESP8266	1
2	Relay 4 Channel	2
3	Sensor PZEM-004t	1
4	Modul Sensor LDR	1
5	TV LCD	1
6	Rice Cooker	1
7	Lampu LED	2
8	Kipas Angin	1
9	PCB Breadboard	2
10	Kabel Jumper	Secukupnya

4.3 Metode Penelitian

Metode yang digunakan penulis untuk penelitian ini adalah metode pengembangan *prototype* dan Studi Pustaka.

4.3.1 Metode Prototype

Metode *prototype* (E. Hartati et al., 2019) merupakan interaksi berulang dalam pengembangan kerangka dimana kebutuhan diubah menjadi kerangka kerja yang berfungsi yang terus menerus dikerjakan melalui kerja sama antara klien dan penguji. Metode ini juga dapat bekerja melalui beberapa instrument perbaikan untuk bekerja pada interaksi (Kurnia dan Chusyairi 2021). Diagram metode ini ditunjukkan pada Gambar 4.1.



Sumber: (Prabowo 2020)

Gambar 4.1 Diagram Prototipe

4.3.2 Studi Pustaka

Studi Pustaka (E. Hartati & Mardiana, 2018), (Muna & Annisa, 2021) merupakan metode untuk menyelesaikan persoalan dengan cara menelusuri sumber-sumber tulisan yang pernah dibuat sebelumnya. Sumber-sumber yang didapat, dijadikan sebagai bahan studi dan disusun menurut kaidah penulisan ilmiah. Terdapat beberapa

metode yang dapat dilakukan untuk melakukan studi pustaka, seperti mengupas (*criticize*), membandingkan (*compare*), meringkas (*summarize*), dan mengumpulkan (*synthesize*) suatu bahan pustaka (Nuryana, Pawito, dan Utari 2019), (Damara & Annisa, 2021).

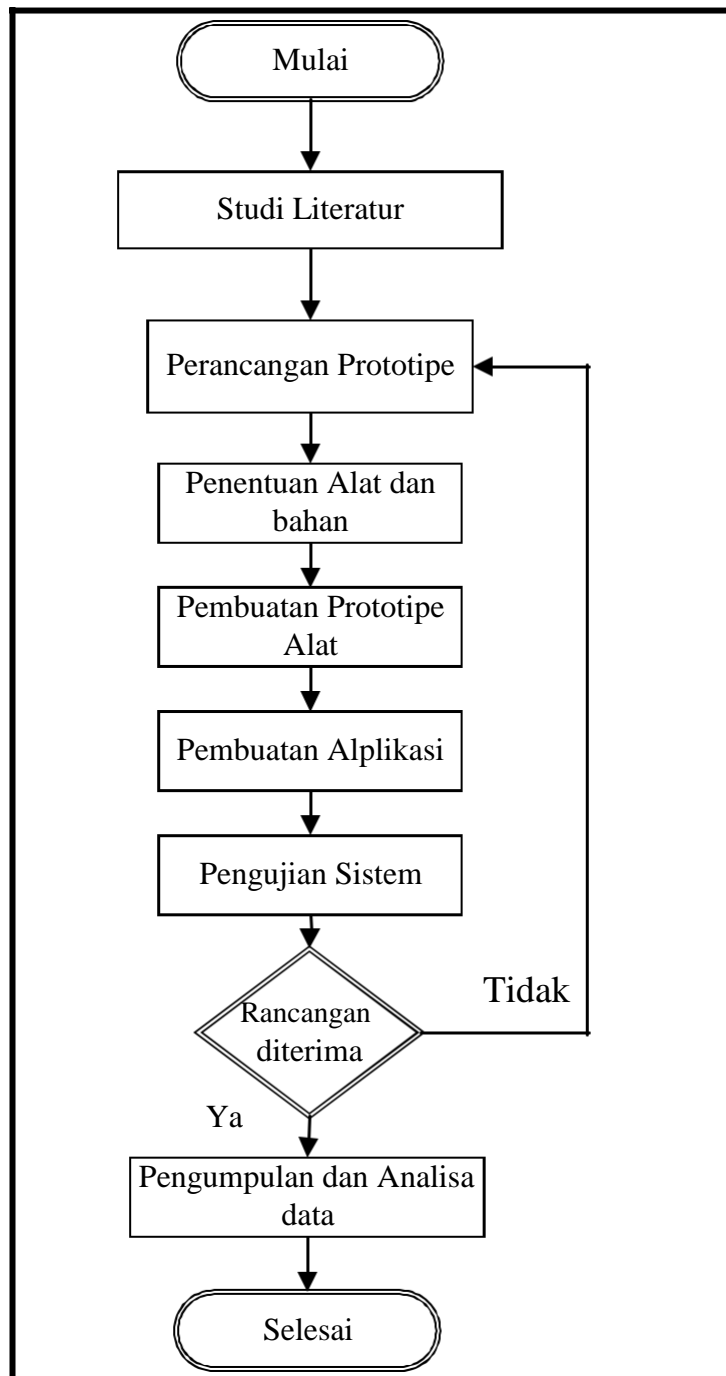
Penulis melakukan studi Pustaka dengan menelaah penelitian yang mempunyai keterkaitan dengan penelitian yang dilakukan oleh penulis dalam menentukan data yang dibutuhkan untuk menunjang perancangan kontrol peralatan elektronik rumah tangga berbasis IoT dan Android ini.

Pada penelitian yang dilakukan oleh Nugraha, pengukuran arus dan tegangan dilakukan menggunakan 2 sensor yang berbeda yaitu sensor ACS712 untuk pengukuran arus listrik dan sensor ZMPT101B untuk pengukuran tegangan (Nugraha et al., 2019). Penelitian ini tidak menggunakan sensor LDR untuk pengukuran intensitas cahaya agar dapat mengotomatisasikan penggunaan lampu.

Pada penelitian lain (Hamdani et al., 2020), menggunakan sensor PIR (Passive Infrared Sensor) dalam pendekatan mode otomatis untuk menghidupkan lampu saat ada seseorang yang melewati sensor tersebut. Dalam penelitian ini juga tidak terdapat sistem untuk merekam penggunaan daya listrik yang terpakai.

4.4 Diagram Alir Penelitian

Diagram alir dari penelitian ini secara garis besar terdiri dari studi literatur, perancangan dan pembuatan alat serta aplikasi, serta pengumpulan dan Analisa data. Detail dari diagram alir penelitian ini ditunjukkan pada Gambar 4.2`.

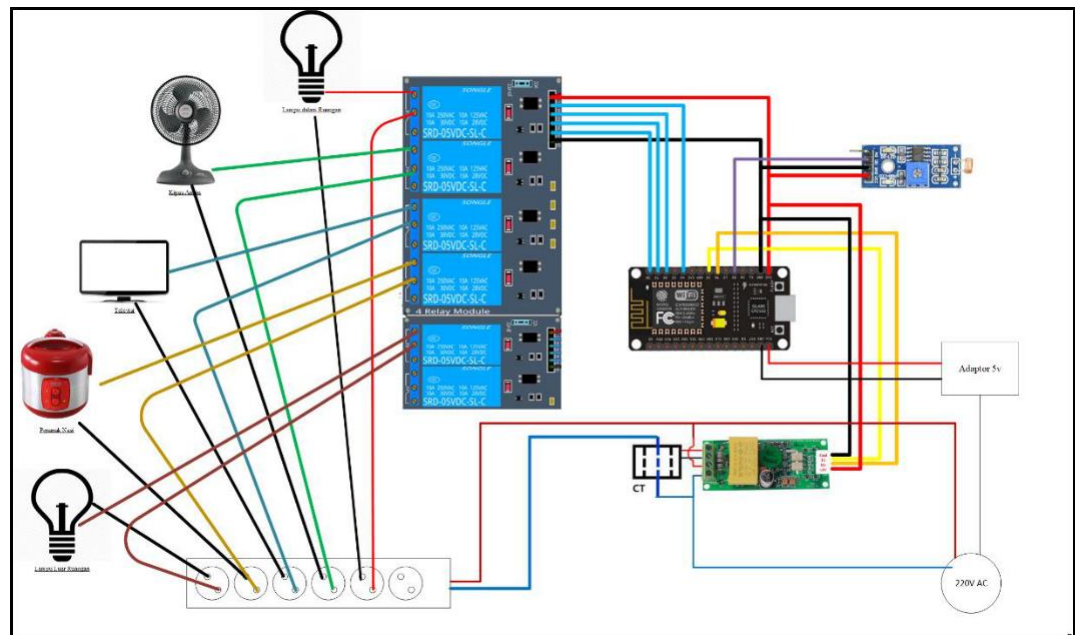


Sumber: diolah sendiri

Gambar 4.2 Diagram Alir Penelitian

4.5 Perancangan Alat

Alat kontrol peralatan elektronik yang telah dirancang seperti pada Gambar 2.1, kemudian dibuat menjadi sebuah rangkaian alat yang ditunjukkan pada Gambar 4.3 dimana masing-masing peralatan elektronik terhubung ke sumber daya listrik melalui relay yang terhubung ke NodeMCU ESP8266. Sumber catu daya juga sebelum tersambung ke peralatan, terlebih dahulu melewati sensor PZEM-004t agar daya, arus, tegangan, dan energi yang digunakan dapat terbaca oleh sensor.



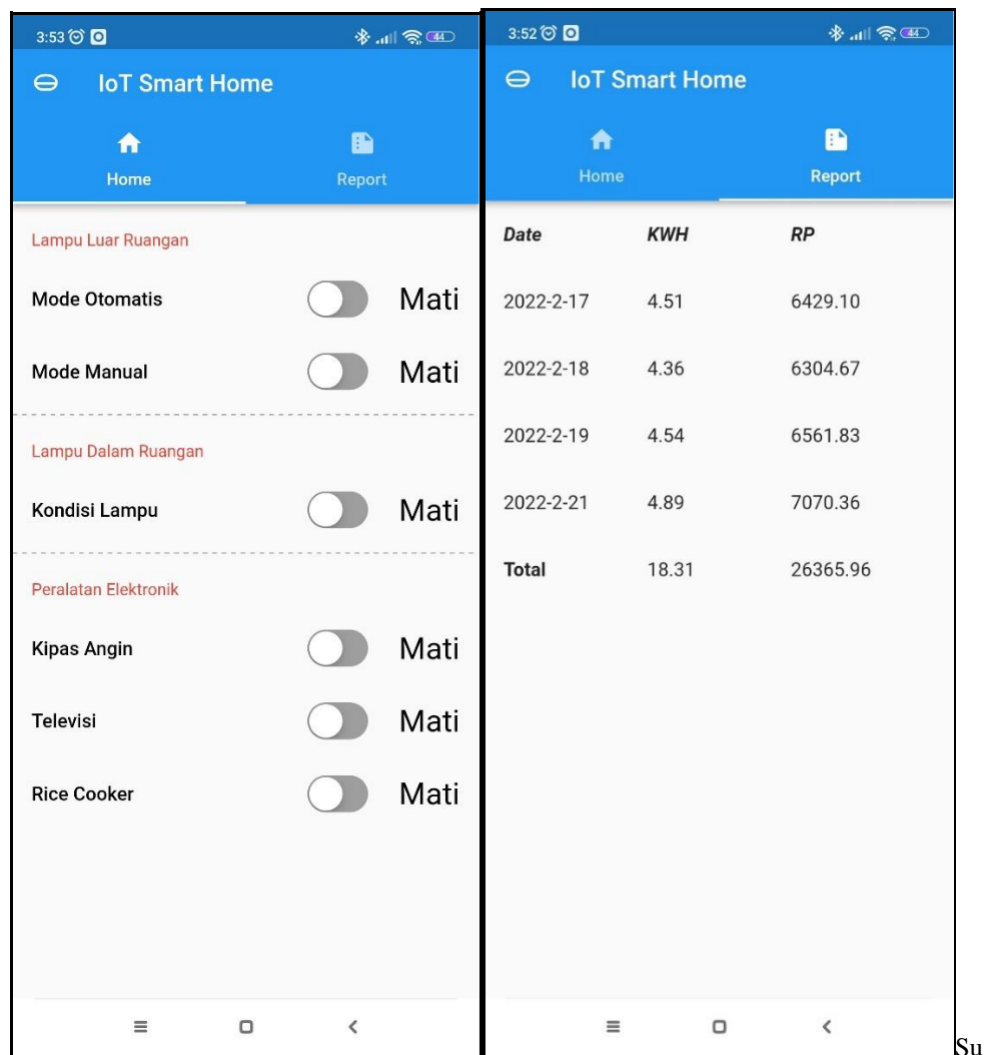
Sumber: diolah sendiri

Gambar 4.3 Skema Rangkaian Alat Kontrol

Untuk pemrograman NodeMCU, penulis menggunakan Arduino IDE yang *support* untuk bahasa pemrogramannya.

4.6 Perancangan Aplikasi

Rangkaian pada Gambar 4.3, dikendalikan oleh aplikasi yang dibuat untuk *smartphone android* menggunakan *framework flutter* yang dapat menjalankan fungsi saklar serta melihat penggunaan kWh yang terpakai serta berapa biaya yang akan dikeluarkan. Aplikasi yang telah dibuat pada penelitian ini dapat dilihat pada Gambar 4.4.



mber: diolah sendiri

Gambar 4.4 Aplikasi Kontrol dan Monitoring Peralatan Elektronik

Framework flutter ini menggunakan bahasa pemrograman *dart* sebagai *codebase* yang selanjutnya akan di-*compile* menjadi aplikasi android dan iOS. Akan tetapi, pada penelitian ini, penulis hanya akan focus pada aplikasi androidnya saja.

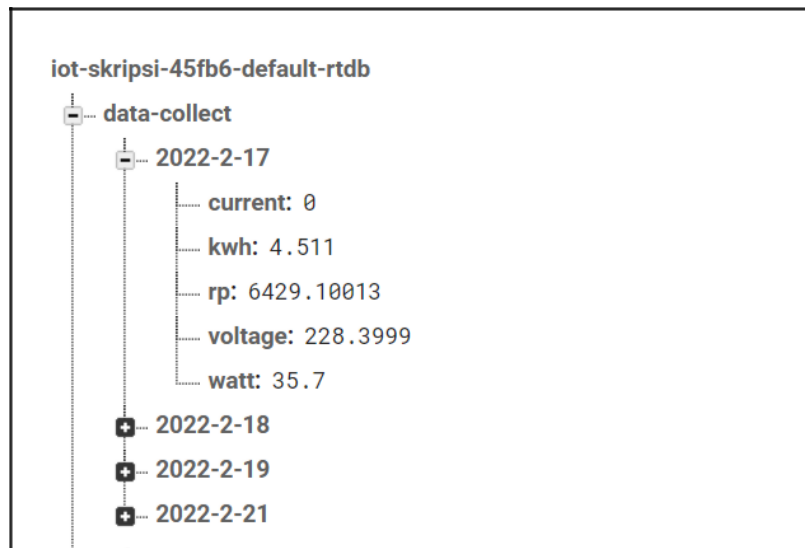
Terdapat 2 laman yang dibuat pada aplikasi ini yaitu laman *Home*, dan laman *Report*. Laman *Home* merupakan laman untuk mengatur *relay* yang terdapat pada rangkaian dengan mengirimkan data dari aplikasi android ke *MQTT broker* yang selanjutnya akan diteruskan oleh *MQTT broker* ke perangkat yang telah dirancang. Agar perangkat android yang dirancang tersebut dapat terhubung ke *MQTT broker*, diperlukan sebuah *package flutter* yang harus di *install* terlebih dahulu yaitu *package mqtt_client*. Pada laman ini, penulis menggunakan QoS 2 (Exactly Once) yang berarti pesan akan tersampaikan dengan tepat dalam satu kali.

Pada Laman *Report*, diperlukan *package firebase_core* agar flutter dapat menggunakan *library* dari firebase dan terkoneksi ke *firebase realtime database* yang sebelumnya sudah penulis buat.

4.7 Perancangan Database

Database yang digunakan untuk menyimpan data pemakaian kWh listrik serta biaya yang dikeluarkan akibat pemakaian listrik tersebut adalah *Firebase Realtime database* yang merupakan NoSQL dengan format data berupa JSON. *Database* ini merupakan tipe *key-value database* dimana merupakan *database* sederhana yang setiap *item*-nya merupakan pasangan kunci (*key*) dan nilai (*value*) dalam bentuk tabel *hash*. Pada Gambar 4.5,

merupakan database untuk menyimpan hasil pembacaan dari sensor PZEM-004t yang berikutnya akan di tampilkan berupa report seperti pada Gambar 4.4.



Sumber: diolah sendiri

Gambar 4.5 *Firebase Realtime Database Key-Value*

Pada saat tanggal berganti, mikrokontroller NodeMCU akan otomatis membuat *child* pada *data-collect* baru sesuai dengan tanggal yang akan disinkronisasikan dengan NTP *Server*.

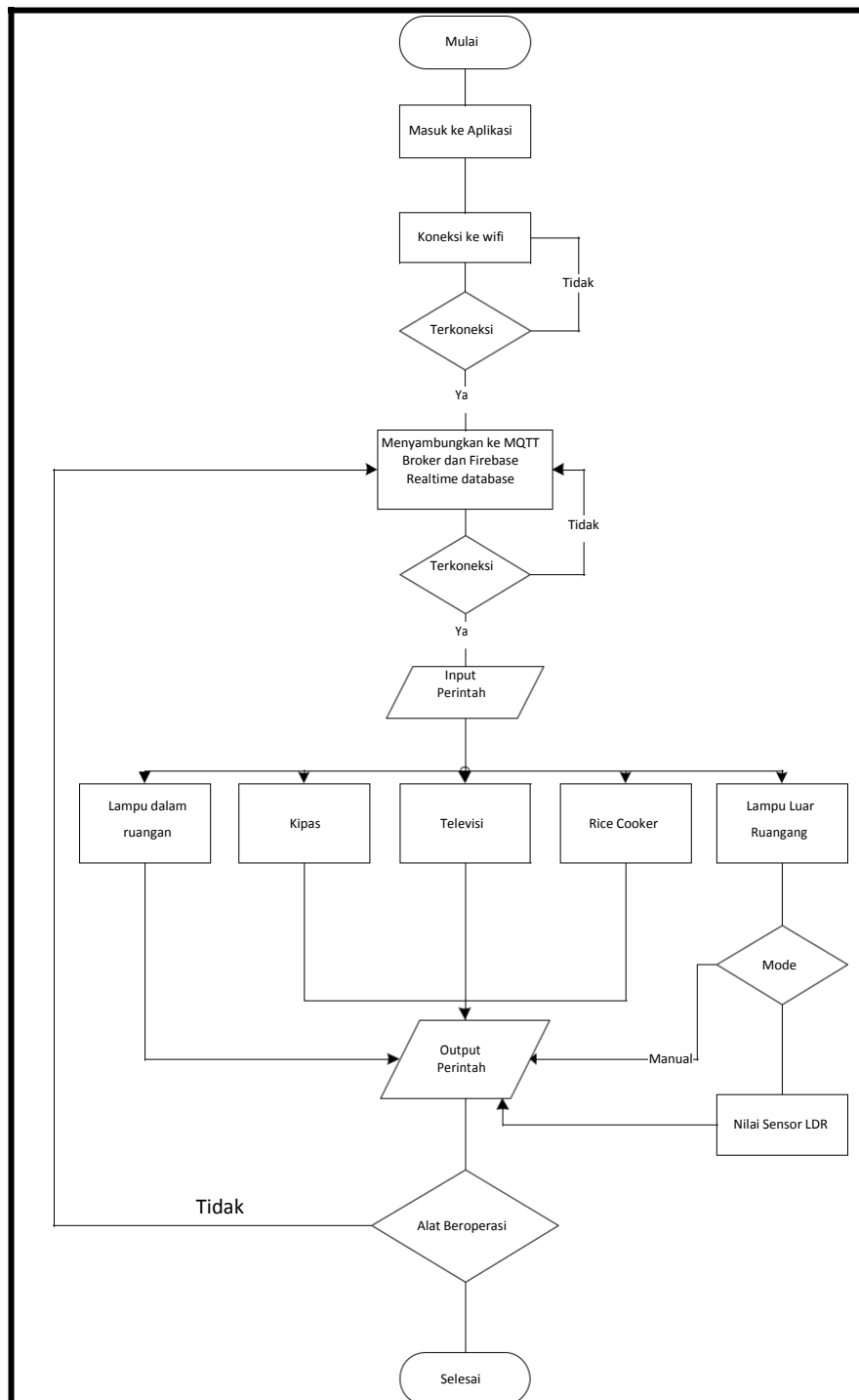
BAB V

PEMBAHASAN

5.1 Implementasi dan Uji Coba Sistem

5.1.1 Implementasi Sistem

Implementasi dari sistem yang dibuat ditunjukkan pada *flowchart* pada Gambar 5.1 dengan mekanisme awal adalah pengguna yang telah menginstal aplikasi yang telah dibuat, lalu membuka aplikasi tersebut yang selanjutnya akan ditampilkan laman utama seperti pada Gambar 4.4. Terdapat 2 laman yang ada pada aplikasi ini yaitu laman *Home*, digunakan untuk mengatur kondisi peralatan elektronik dari jarak jauh, lalu laman yang kedua adalah laman *Report*, dimana menunjukkan jumlah pemakaian kWh listrik serta biaya yang dikeluarkan akibat dari pemakaian listrik tersebut.

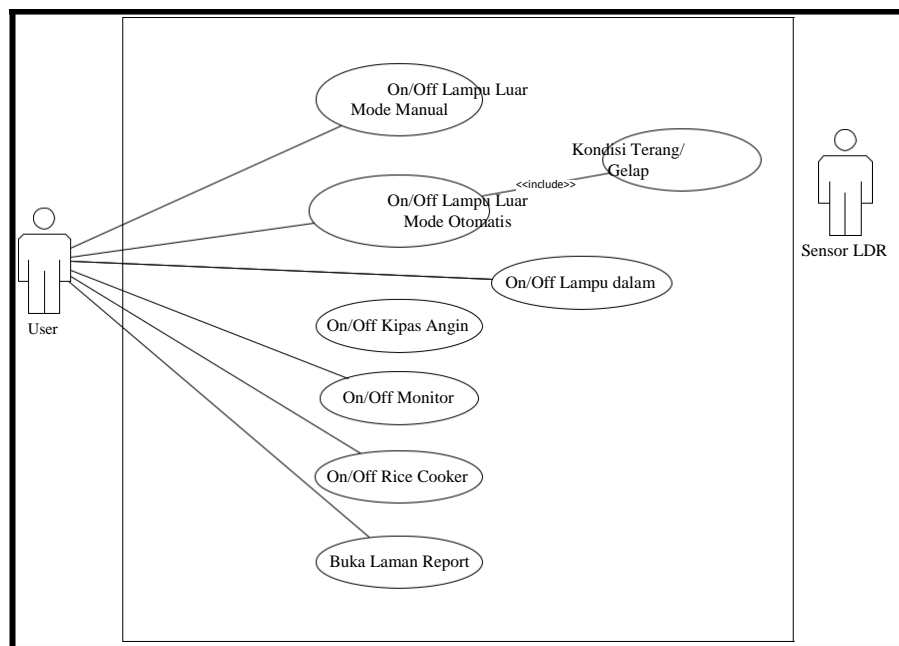


Sumber: diolah sendiri

Gambar 5.1 Flowchart Sistem IoT

5.1.2 Use Case Diagram Sistem

Interaksi ataupun tindakan dari *user* terhadap system yang berupa banyak jenis kegiatan dapat dimodelkan dengan mudah menggunakan *use case diagram*. Pada Gambar 5.2 ditunjukkan interaksi antara *user* dan sistem.



Sumber: diolah sendiri

Gambar 5.2 Diagram Interaksi *User* Terhadap Sistem

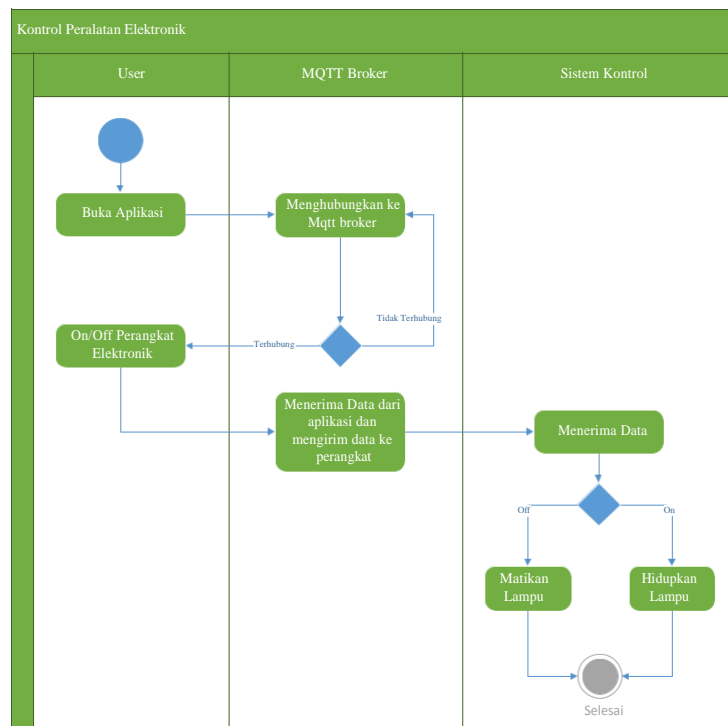
Berdasarkan Gambar 5.2, fungsi yang berjalan pada sistem control adalah melakukan *on/off* pada *mode* otomatis untuk mendapatkan nilai pada sensor LDR, *on/off* peralatan elektronik, dan menampilkan laman laporan yang menunjukkan jumlah kWh yang digunakan serta biaya yang dikeluarkan.

5.1.3 Activity Diagram Sistem

Fungsi dari suatu sistem dapat dijalankan dengan adanya hubungan antara *user* dan sistem. Penjelasan tersebut dapat dijelaskan dengan menggunakan *Activity Diagram* yang merupakan salah satu tools dari diagram UML.

5.1.3.1 Activity Diagram Lampu Luar Mode Manual

Pada aplikasi kontrol perangkat terdapat beberapa fungsi yang dapat digunakan oleh user yaitu kontrol *on/off* peralatan elektronik dan mode otomatis lampu luar dengan menggunakan sensor LDR. Untuk *Activity Diagram* kontrol *on/off* peralatan elektronik ditunjukkan pada Gambar 5.3.



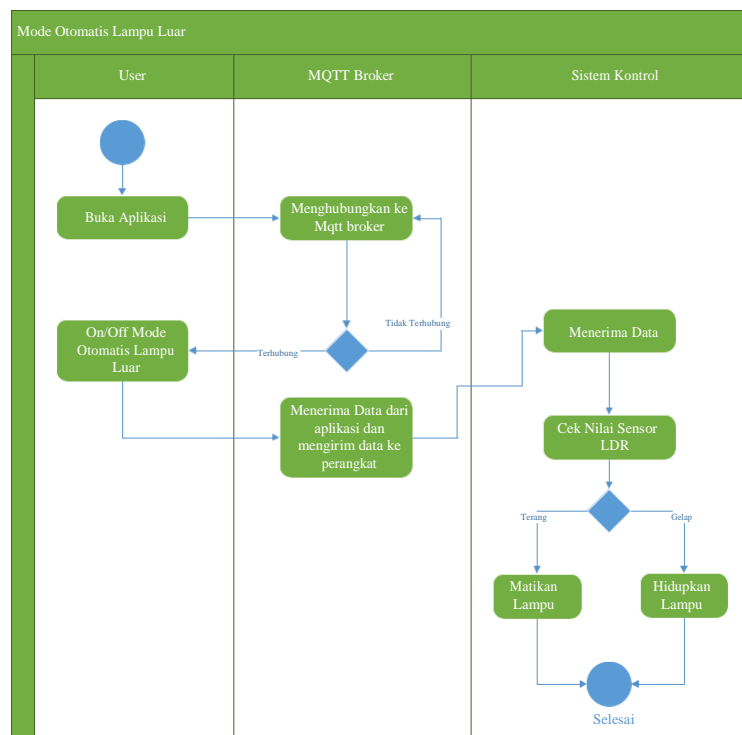
Sumber: diolah sendiri

Gambar 5.3 Activity Diagram Kontrol terhadap Sistem

Berdasarkan Gambar 5.3 dapat dilihat bahwa setelah user membuka aplikasi, maka aplikasi langsung menghubungkan ke MQTT broker, lalu saat sudah terhubung maka user akan dapat mengontrol on/off peralatan elektronik yang ada, lalu aplikasi akan menerima dan meneruskan data tersebut ke sistem kontrol untuk menjalankan perintah sesuai yang dikirimkan dari aplikasi user.

5.1.3.2 Activity Diagram Lampu Luar Mode Otomatis

Mode otomatis lampu luar menggunakan nilai dari sensor LDR sebagai acuan untuk on/off lampu. Hal ini ditunjukkan pada Gambar 5.4



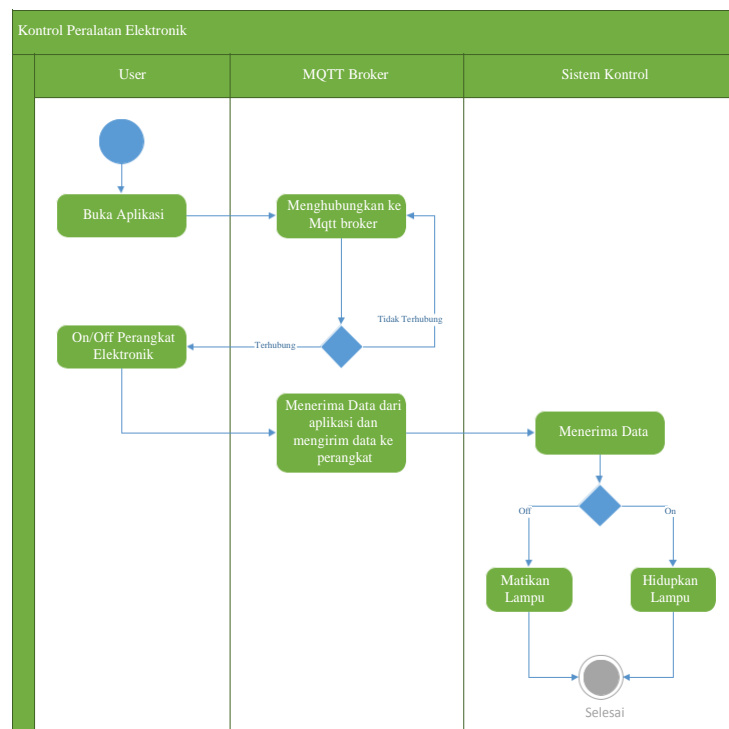
Sumber: diolah sendiri

Gambar 5.4 Activity Diagram Mode Otomatis Sistem

Perbedaan mode otomatis Gambar 5.4 dengan Activity diagram sebelumnya terdapat pada saat sistem kontrol menerima data dari MQTT Broker yang selanjutnya akan mengaktifkan sensor LDR untuk mendapatkan nilai gelap/terang cahaya dimana saat kondisi gelap, lampu akan dihidupkan dan saat kondisi terang, maka lampu akan dimatikan.

5.1.3.3 Activity Diagram On/Off Lampu Dalam Ruangan

Activity diagram on/off lampu dalam ruangan ditunjukkan pada Gambar 5.5.

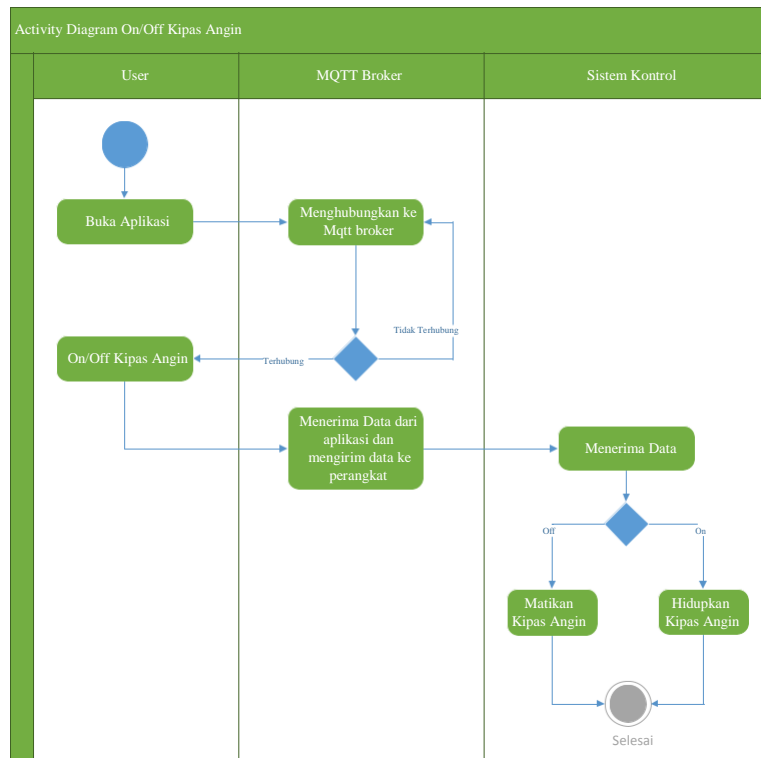


Sumber: diolah sendiri

Gambar 5.5 Activity Diagram Kontrol Lampu Dalam Ruangan

5.1.3.4 Activity Diagram On/Off Kipas Angin

Activity diagram on/off kipas angin ditunjukkan pada Gambar 5.6.

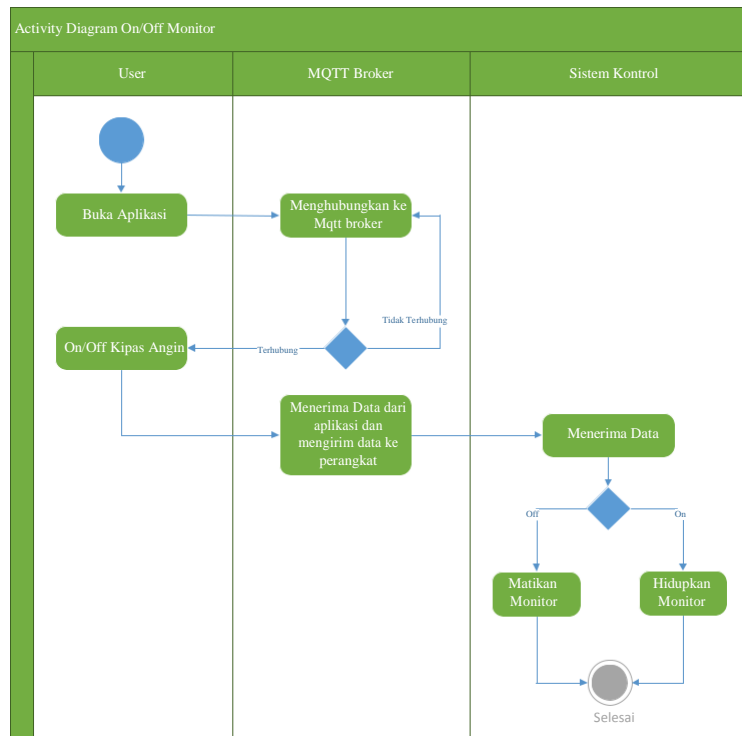


Sumber: diolah sendiri

Gambar 5.6 Activity Diagram Kontrol Kipas Angin

5.1.3.5 Activity Diagram On/Off Monitor

Activity diagram on/off Monitor ditunjukkan pada Gambar 5.7.

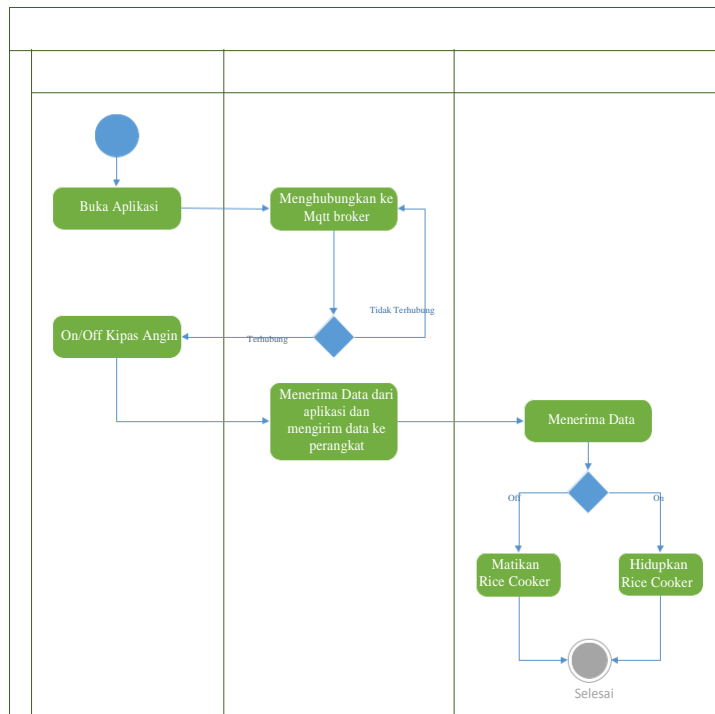


Sumber: diolah sendiri

Gambar 5.7 Activity Diagram Kontrol Monitor

5.1.3.6 Activity Diagram On/Off Rice Cooker

Activity diagram on/off Rice Cooker ditunjukkan pada Gambar 5.8.

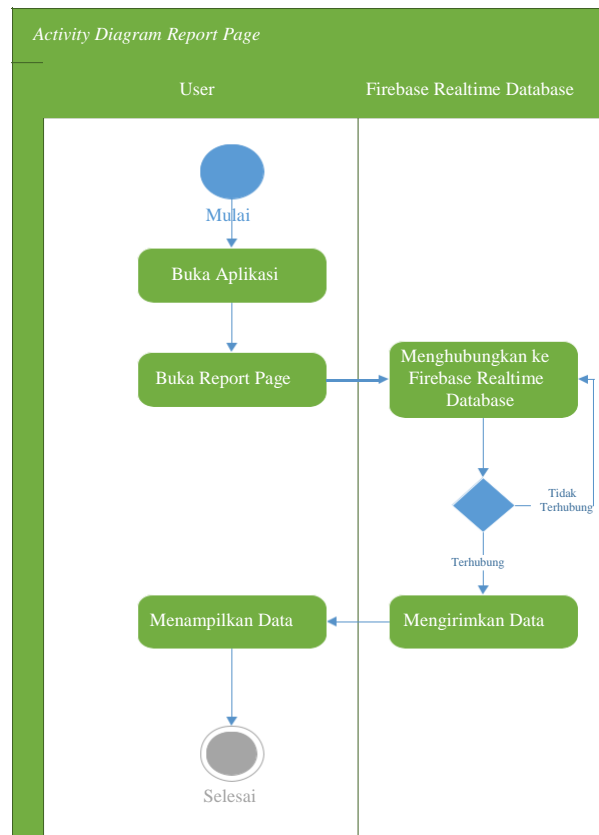


Sumber: diolah sendiri

Gambar 5.8 Activity Diagram Kontrol Rice Cooker

5.1.3.7 Activity Diagram Laman Report

Activity diagram laman report ditunjukkan pada Gambar 5.9. Activity diagram ini dimulai dengan membuka aplikasi lalu user membuka laman Report yang pada saat bersamaan, aplikasi menghubungkan ke *Firestore Realtime Database* untuk mengambil data yang telah tersimpan, lalu data tersebut ditampilkan pada aplikasi.



Sumber: diolah sendiri

Gambar 5.9 Activity Diagram Laman Report

5.1.4 Pengujian Sistem

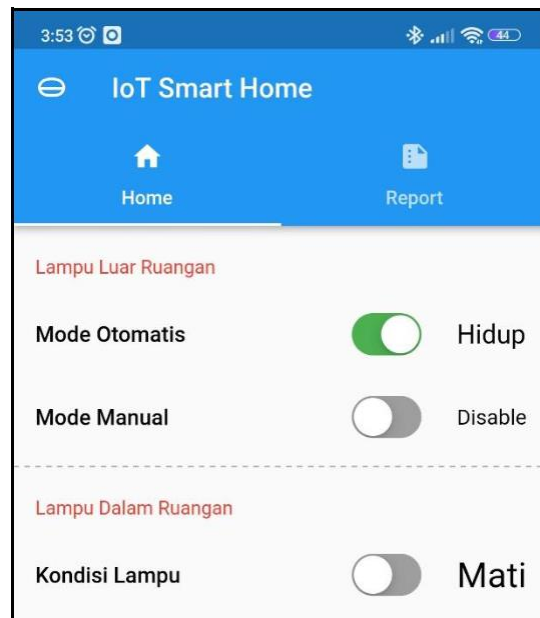
Pengujian dilakukan setelah proses implementasi sistem telah berhasil. Pengujian ini berguna untuk mengetahui apakah sistem yang telah dirancang dan diterapkan, dapat berjalan dengan sebagaimana mestinya. Pengujian yang dilakukan oleh penulis terbagi menjadi beberapa bagian yaitu:

1. Pengujian *mode* otomatis untuk menghidupkan lampu dengan menggunakan nilai digital dari modul sensor LDR.

2. Pengujian satu per satu dari setiap perangkat saat dihidupkan melalui aplikasi yang telah dibuat sebelumnya.
3. Pengujian perangkat saat dihidupkan seluruhnya dengan menggunakan aplikasi yang telah dibuat sebelumnya.
4. Pengujian perekaman kWh dan biaya yang dikeluarkan pada *Firebase Realtime Database*.

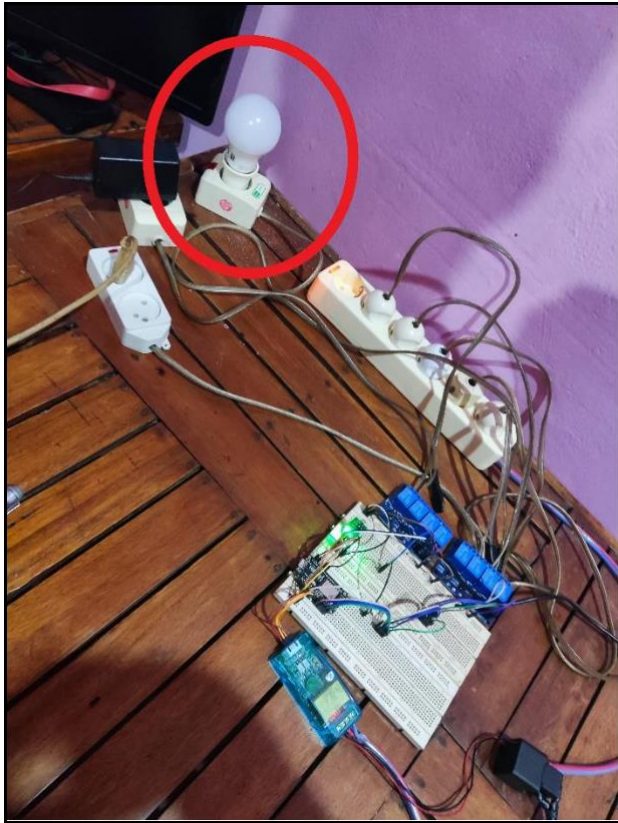
5.1.5 Hasil Pengujian Sistem

Pengujian *mode* otomatis untuk menghidupkan lampu dilakukan dengan kondisi cahaya yang terang, sehingga kondisi awal lampu dalam keadaan mati. Lalu penulis menutup modul sensor LDR dengan buku agar cahaya tidak dapat ditangkap oleh modul sensor dan hasilnya lampu akan menyala.



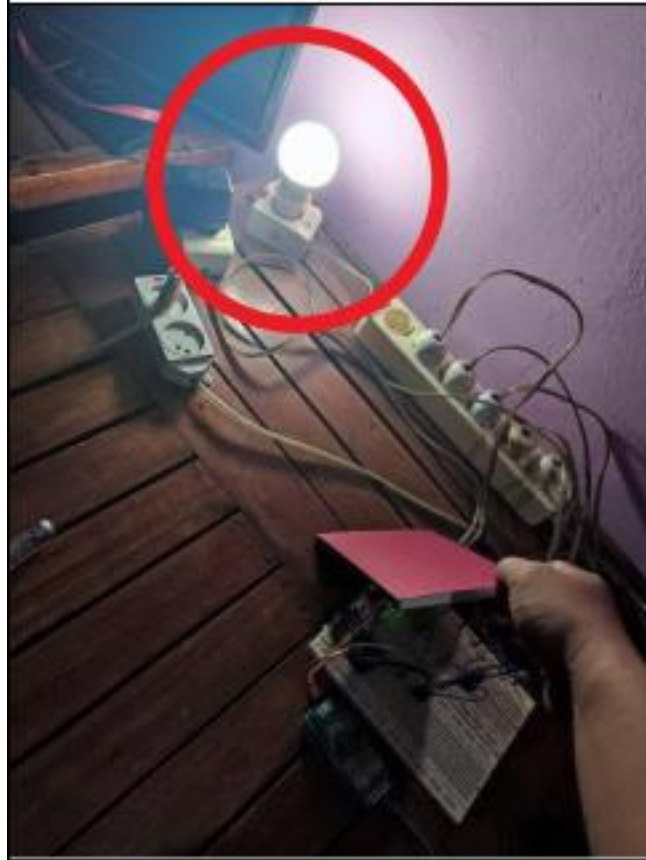
Sumber: diolah sendiri

Gambar 5.10 Menghidupkan Fungsi Otomatis pada Aplikasi



Sumber: diolah sendiri

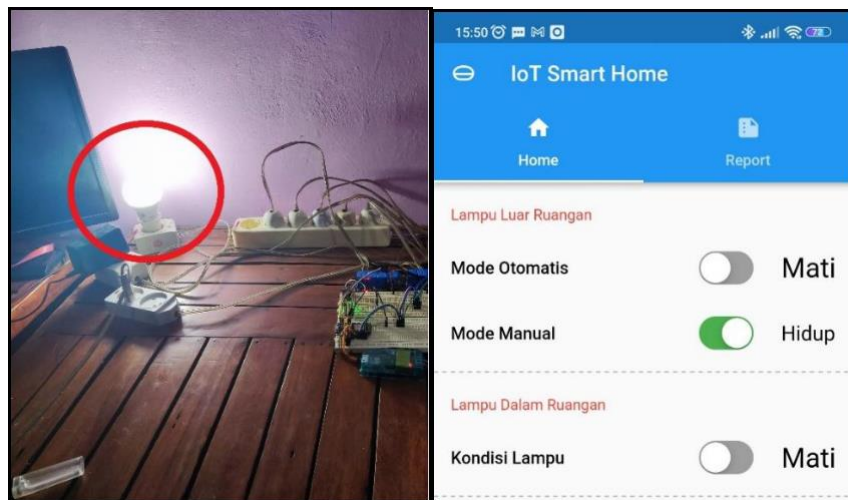
Gambar 5.11 Kondisi Lampu Luar Saat Kondisi Cahaya Terang



Sumber: diolah sendiri

Gambar 5.12 Kondisi Lampu Saat Kondisi Cahaya Gelap

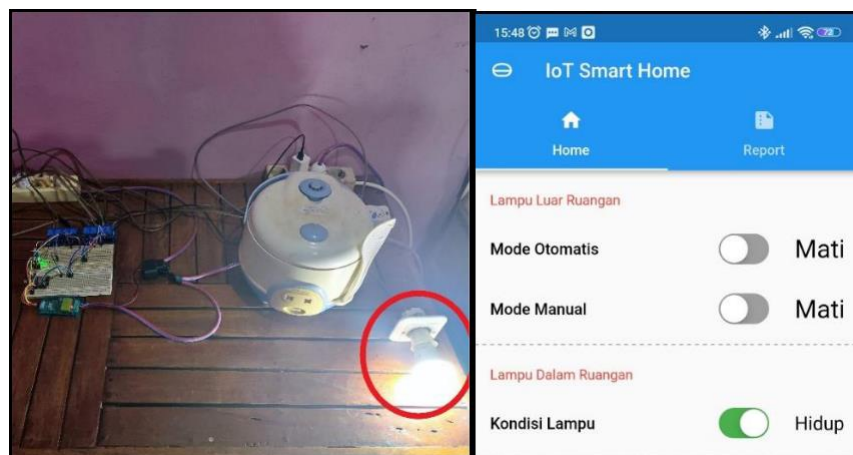
Pada Gambar 5.11 adalah kondisi lampu mode otomatis saat terdapat cahaya yang cukup sehingga perangkat tidak perlu menghidupkan lampu, sedangkan pada Gambar 5.12 adalah kondisi dimana modul sensor LDR tidak mendapatkan cahaya sehingga perangkat akan menghidupkan lampu tersebut. Hal ini dilakukan penulis dengan cara menutup modul sensor LDR agar tidak mendapatkan cahaya.



Sumber: diolah sendiri

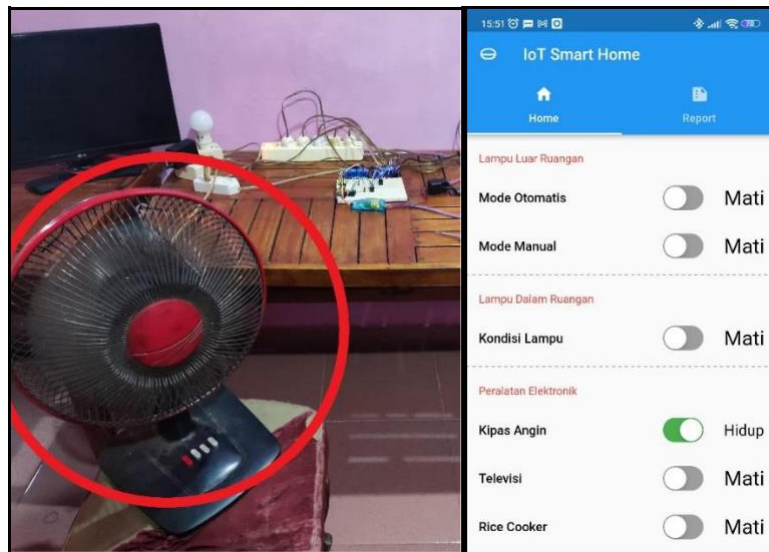
Gambar 5.13 Kondisi Lampu Luar Saat Mode Manual

Pada Gambar 5.13, dilakukan pengujian dari Lampu Luar ruangan dengan menggunakan mode manual yaitu dengan menekan *switch* yang ada pada aplikasi. Untuk menggunakan *mode* manual ini, harus terlebih dahulu mematikan *mode* otomatisnya. Jika mode otomatis masih dalam kondisi hidup, maka mode manual tidak akan dapat digunakan.



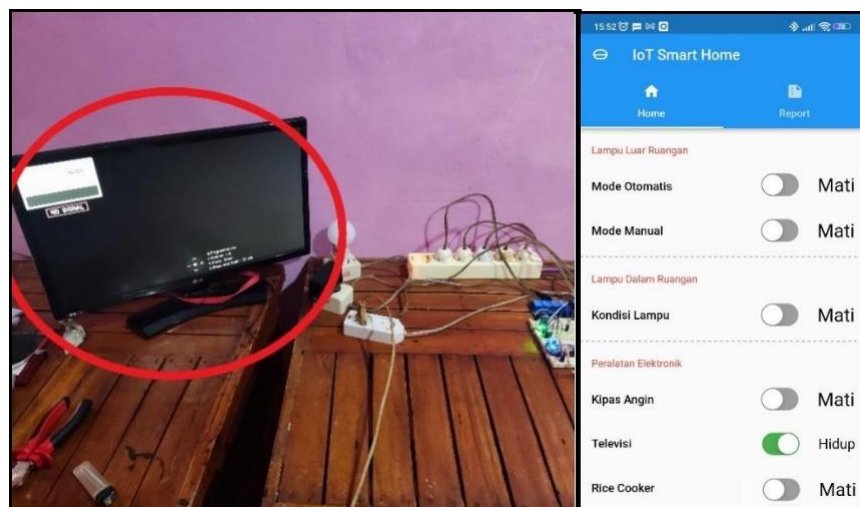
Sumber: diolah sendiri

Gambar 5.14 Kondisi Lampu Dalam Saat Dihidupkan



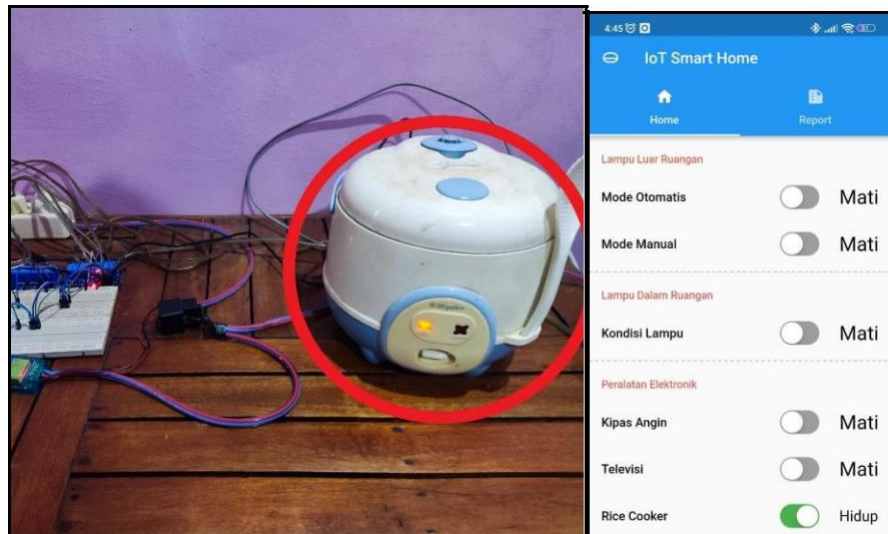
Sumber: diolah sendiri

Gambar 5.15. Kondisi Kipas Angin Saat Dihidupkan



Sumber: diolah sendiri

Gambar 5.16. Kondisi Televisi Saat Dihidupkan

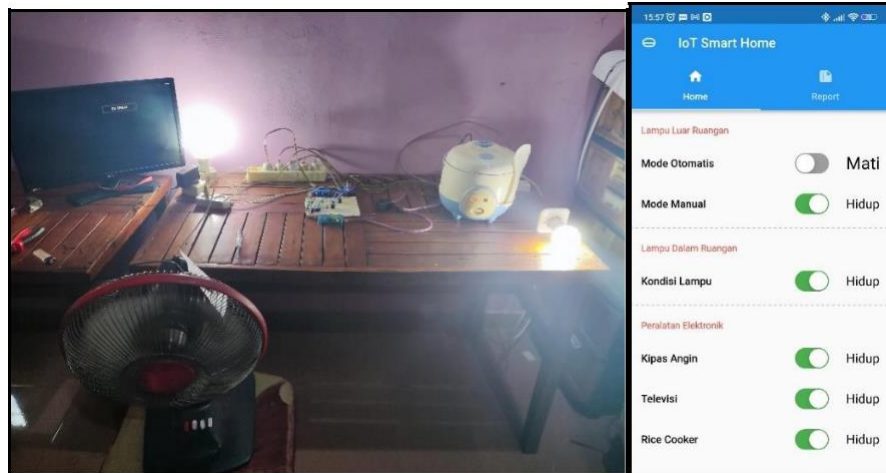


Sumber: diolah sendiri

Gambar 5.17. Kondisi Televisi Saat Dihidupkan

Pada pengujian satu persatu, penulis menghidupkan semua perangkat secara bergantian dan hasilnya semua perangkat berhasil hidup dengan normal.

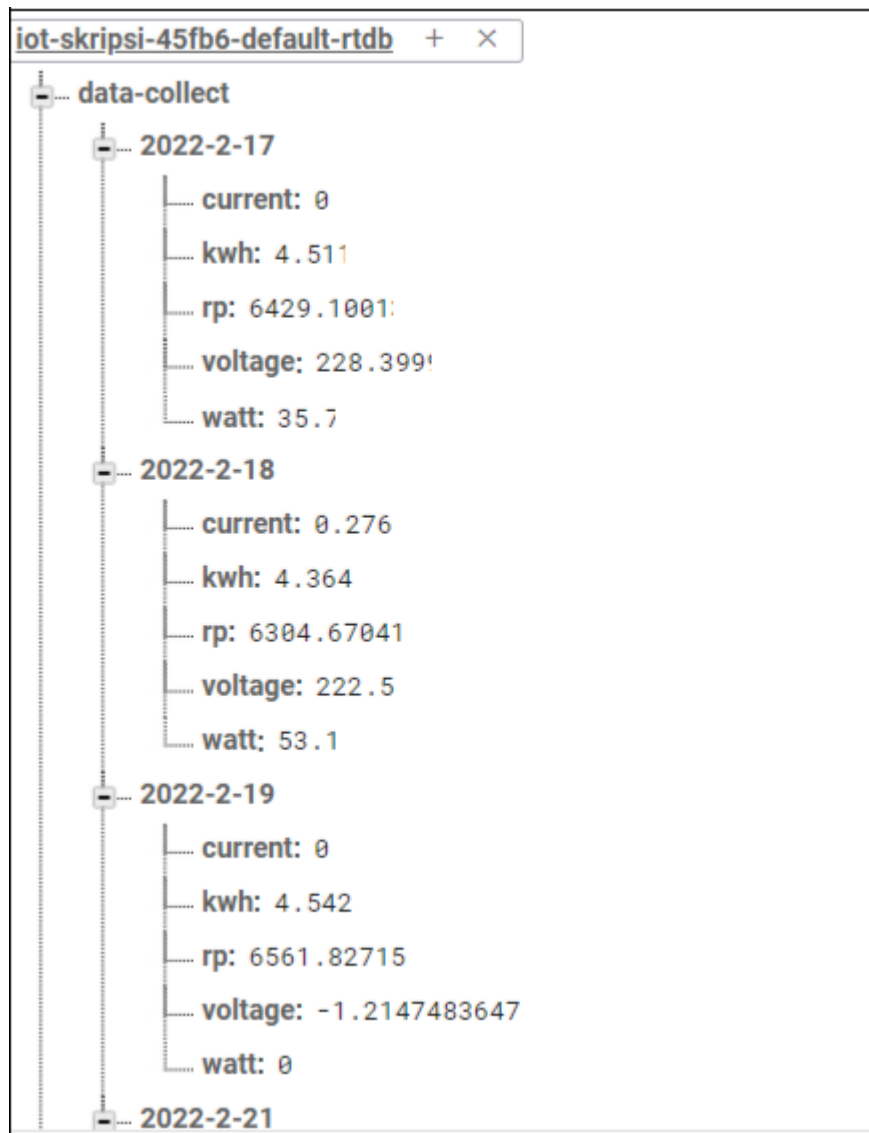
Pengujian selanjutnya adalah menghidupkan semua perangkat secara bersamaan yang hasilnya ditunjukkan pada Gambar 5.18



Sumber: diolah sendiri

Gambar 5.18. Kondisi Semua Perangkat Saat Dihidupkan

Pengujian sistem perekaman penggunaan kWh serta biaya listrik yang dikeluarkan menggunakan modul sensor PZEM-004t terjadi secara otomatis saat perangkat sistem dihidupkan karena pada perangkat NodeMCU yang telah dibuat, telah tertanam proses sinkronisasi *realtime* dengan *Firestore Real Time Database* dan *NTP Server*. *NTP (Network Time Protocol) Server* digunakan untuk menyesuaikan waktu yang ada pada NodeMCU dengan waktu yang ada pada internet sesuai GMT yang kita atur terlebih dahulu.



Sumber: diolah sendiri

Gambar 5.19. Hasil Perekaman PZEM-004t

Hasil perekaman dari modul sensor PZEM-004t ini dapat dilihat pula pada aplikasi yang telah dibuat pada lama *Report*, serta dapat pula di *export* dengan mengakses *website console* dari *Firebase Realtime Database* dimana hasil yang akan dikeluarkan berbentuk *file JSON*.

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan hasil penelitian, dapat disimpulkan bahwa pengembangan perangkat pengendali peralatan elektronik rumah tangga berbasis IoT dan Android dapat membantu bagi pemilik rumah yang sering bepergian untuk dapat memastikan bahwa peralatan elektronik rumah yang ditinggalkan, dalam keadaan tidak menyala untuk menghindari pemakaian listrik yang berlebihan dan kemungkinan terjadinya hal yang tidak diinginkan karena kondisi alat elektronik yang terus menyala. Pemilik rumah juga dapat melihat penggunaan kWh dan biaya listrik dari peralatan rumah tangga yang digunakan.

6.2 Saran

Pada Sistem kontrol peralatan elektronik rumah tangga berbasis Internet of Things dan Android ini masih terdapat kekurangan yang kelak sekiranya dapat diperbaiki oleh pengembang atau peneliti berikutnya. Beberapa pengembangan yang dapat dilakukan adalah sebagai berikut:

1. Menambah perangkat yang dapat dikendalikan dan di pantau pemakaiannya.
2. Menambahkan sistem analisis otomatis yang dapat memprediksi perilaku pengguna perangkat.

DAFTAR PUSTAKA

Alfian, Raviki Dwi, Subuh Isnur Haryudo, Unit Three Kartini, dan Nur Kholis.

2021. *RANCANG BANGUN ALAT MONITORING PEMAKAIAN TARIF LISTRIK DAN KONTROL DAYA LISTRIK PADA RUMAH KOS BERBASIS INTERNET OF THINGS*. Jurnal Teknik Elektro UNESA 10(3):661–69.

Anggiawan, D. D., Emerensye S. Y. Pandie, dan Meiton Boru. 2018. *Sistem Informasi Pelayanan Publik Kelurahan Bakunase Kota Kupang Untuk Peningkatan Kualitas Pelayanan Berbasis Web*. J-Icon 6(2):8–13.

Atmoko, R. A. 2019. *Dasar Implementasi Protokol MQTT Menggunakan Python dan NodeMCU*. Mokosoft Media.

Badan Pusat Statistik. 2020. *Konsumsi Listrik per Kapita (MWH/Kapita), 2018-2020*. Badan Pusat Statistik. Diambil 25 Desember 2021 (<https://www.bps.go.id/indicator/7/1156/1/konsumsi-listrik-per-kapita.html>).

Bolloor, Adith Jagadish. 2015. *Arduino by Example*. Birmingham: Packt Publishing.

Dewi, Nurul Hidayati Lusita, Mimin F. Rohmah, dan Soffa Zahara. 2019. *PROTOTYPE SMART HOME DENGAN MODUL NODEMCU ESP8266 BERBASIS INTERNET OF THINGS (IOT)*. 3.

Hamdani, H., J. Budiarto, dan S. Hadi. 2020. *Sistem Kendali Peralatan Elektronik Rumah Tangga Berbasis Internet Of Things Menggunakan Protokol MQTT*.” Jurnal Bumigora Informasi Technology (BITe) 2(1):1–11. doi:

10.30812/bite.v2i1.799.

Iqbal, Javed, Murad Khan, Muhammad Talha, Haleem Farman, Bilal Jan, Arshad Muhammad, dan Hasan Ali Khattak. 2018. *A generic internet of things architecture for controlling electrical energy consumption in smart homes.*

Sustainable Cities and Society 43(September):443–50. doi: 10.1016/j.scs.2018.09.020.

Kurnia, Rizki, dan Ahmad Chusyairi. 2021. *RANCANG BANGUN DISPENSER PENUANGAN AIR MINUM OTOMATIS BERBASIS ARDUINO MENGGUNAKAN METODE PROTOTYPE.* *Aisyah Journal of Informatics and Electrical Engineering* 3(2):153–62.

Nugraha, Alpin, Dede Cahyadi, Dini Oktarina, Dwi Handayani, Program Studi, dan Teknik Informatika. 2019. *Sistem Monitor Dan Kontrol Konsumsi Listrik Rumah Tangga.* 8(01):9–16.

Nuryana, Arief, Pawito Pawito, dan Prahastiwi Utari. 2019. *Pengantar metode penelitian kepada suatu pengertian yang mendalam mengenai konsep fenomenologi.* *ESAINS* 2:19–24. doi: <https://doi.org/10.31848/ensains.v2i1.148>.

Prabowo, Mei. 2020. *METODOLOGI PENGEMBANGAN SISTEM INFORMASI.* diedit oleh A. W. Budyastomo. Salatiga: LP2M IAIN Salatiga.

Pratama, Satria Bagus, Rendy Munadi, dan Akhmad Syauqi. 2018. *ANALISIS PERFORMANSI PROTOKOL COAP DAN MQTT-SN PADA SISTEM SMARTHOME DENGAN COOJA NETWORK SIMULATOR.* *Informatics*

Journal 5(2):1982–91.

PT PLN Persero. 2022. *Penetapan Penyesuaian Tarif Tenaga Listrik*. Diambil 1 Februari 2022 (<https://web.pln.co.id/pelanggan/tarif-tenaga-listrik/tariff-adjustment>).

Rahman, Budi, dan I. Imelda. 2020. *Prototipe Sistem Kontrol Smart Home Berbasis IoT Dengan Metode MQTT Menggunakan Google Asisstant*. *Rekayasa Sistem dan Teknologi Informasi (RESTI)* 4(3):303–10.

Rahman, Mochamad Bagus Arif. 2019. *SISTEM KENDALI PERALATAN ELEKTRONIK RUMAH TANGGA BERBASIS INTERNET OF THINGS (IoT) MENGGUNAKAN NODEMCU*. *Ubiquitous: Computers and its Applications Journal* 2:99–104. doi: 10.51804/ucaiaj.v2i2.99-104.

Santoso, Arif Dwi, dan Muhammad Agus Salim. 2019. *Penghematan Listrik Rumah Tangga dalam Menunjang Kestabilan Energi Nasional dan Kelestarian Lingkungan*. *Jurnal Teknologi Lingkungan* 20(2):263–70. doi: 10.29122/jtl.v20i2.3242.

Siswanto, Siswanto, Thoha Nurhadiyan, dan Muhamad Junaedi. 2020. *PROTOTYPE SMART HOME DENGAN KONSEP IOT (INTERNET OF THING) BERBASIS NODEMCU DAN TELEGRAM*. *Jurnal Sistem Informasi dan Informatika (Simika)* 3(1):85–93. doi: 10.47080/simika.v3i1.850.

Syifa, Fikra Titan, Gilang Prayoga, dan Muntaqo Alfin Amanaf. 2020. *Sistem Pengaman Kunci Kontak Sepeda Motor Melalui Android Berbasis NodeMCU ESP8266*. *JOURNAL OF TELECOMMUNICATION*,

ELECTRONICS, AND CONTROL ENGINEERING (JTECE) 02(01):24–
34.

Tanto, Tanto, dan Darmuji Darmuji. 2019. *Penerapan Internet of Things (IoT) Pada Alat Monitoring Energi Listrik*. Jurnal Elektronika Listrik dan Teknologi Informasi Terapan 1(1):45–51.

Windryani, Nindithia Putri, Nyoman Bogi A. K, dan Ratna Mayasari. 2019. *ANALISA PERBANDINGAN PROTOKOL MQTT DENGAN HTTP PADA IOT PLATFORM PATRIOT*. Hal. 3192–99 in e-Proceeding of Engineering. Vol. 6.

DAFTAR LAMPIRAN

1. PEMROGRAMAN PADA NODEMCU

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <time.h>
#include <TZ.h>
#include <FS.h>
#include <LittleFS.h>
#include <WiFiUdp.h>
#include <NTPClient.h>
#include <CertStoreBearSSL.h>
#include <FirebaseESP8266.h>
#include <PZEM004Tv30.h>

const char* ssid = "ssid_wifi";
const char* password = "password";
const char* mqtt_server = "url_mqtt_broker";

#define FIREBASE_HOST "url_firebase_realtime_database"
#define FIREBASE_AUTH "auth_firebase_realtime_database"

int toggleState_1 = 1; //Define integer to remember the toggle state for relay 1
int toggleState_2 = 1; //Define integer to remember the toggle state for relay 2
int toggleState_3 = 1; //Define integer to remember the toggle state for relay 3
int toggleState_4 = 1; //Define integer to remember the toggle state for relay 4
int toggleState_5 = 1; //Define integer to remember the toggle state for relay 5
```

```

int toggleState_6 = 0; //Define integer to remember the toggle state for relay 6

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP, "pool.ntp.org");

String months[12]={"January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December"};

#define sub1 "switch1"
#define sub2 "switch2"
#define sub3 "switch3"
#define sub4 "switch4"
#define sub5 "switch5"
#define sub6 "switch6"
#define pub1 "switch1_status"
#define pub2 "switch2_status"
#define pub3 "switch3_status"
#define pub4 "switch4_status"
#define pub5 "switch5_status"
#define pub6 "switch6_status"

BearSSL::CertStore certStore;

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

WiFiClientSecure espClient;

PubSubClient * client;

unsigned long lastMsg = 0;

```

```

#define MSG_BUFFER_SIZE (500)

char msg[MSG_BUFFER_SIZE];
int value = 0;

String path = "/data-collect";

PZEM004Tv30 pzem(14, 12);
float harga_KWh = 1444.70; //harga per KWh

void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi
  network Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");

```

```

Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void reconnect() {
  // Loop until we're reconnected
  while (!client->connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client - MyClient";
    if (client->connect(clientId.c_str(), "username_mqtt_client", "password_mqtt"))
    {
      Serial.println("connected");
      client->subscribe(sub1);
      client->subscribe(sub2);
      client->subscribe(sub3);
      client->subscribe(sub4);
      client->subscribe(sub5);
      client->subscribe(sub6);
    } else {
      Serial.print("failed, rc = ");
      Serial.print(client->state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
}

```

```

void setDateTime() {

    configTime(TZ_Europe_Berlin, "id.pool.ntp.org", "time.nist.gov");

    Serial.print("Waiting for NTP time sync: ");
    time_t now = time(nullptr);
    while (now < 8 * 3600 * 2) {
        delay(100);
        Serial.print(".");
        now = time(nullptr);
    }
    Serial.println();
    struct tm timeinfo;
    gmtime_r(&now, &timeinfo);
    Serial.printf("%s %s", tzname[0], asctime(&timeinfo));
}

void callback(char* topic, byte* payload, unsigned int length) {
    payload[length] = '\0';
    String strTopic = String(topic);
    String strPayload = String((char *) payload);

    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    Serial.println();

    if(strTopic == sub1)

```

```

{
  if(strPayload == "0") {
    digitalWrite(D1, LOW);
    toggleState_1 = 0;
    client->publish(pub1, "0");
  }

  if(strPayload == "1") {
    digitalWrite(D1, HIGH);
    toggleState_1 = 1;
    client->publish(pub1, "1");
  }
}

else if(strstr(topic, sub2)) {

  if(strPayload == "0") {
    digitalWrite(D2, LOW);
    toggleState_2 = 0;
    client->publish(pub2, "0");
  }

  if(strPayload == "1") {
    digitalWrite(D2, HIGH);
    toggleState_2 = 1;
    client->publish(pub2, "1");
  }
}

```



```
} else if(strstr(topic, sub3)) {  
  
    if(strPayload == "0") {  
        digitalWrite(D0, LOW);  
        toggleState_3 = 0;  
        client->publish(pub3, "0");  
    }  
  
    if(strPayload == "1") {  
        digitalWrite(D0, HIGH);  
        toggleState_3 = 1;  
        client->publish(pub3, "1");  
    }  
} else if(strstr(topic, sub4)) {  
  
    if(strPayload == "0") {  
        digitalWrite(D4, LOW);  
        toggleState_4 = 0;  
        client->publish(pub4, "0");  
    }  
  
    if(strPayload == "1") {  
        digitalWrite(D4, HIGH);  
        toggleState_4 = 1;  
        client->publish(pub4, "1");  
    }  
} else if(strstr(topic, sub5)) {
```

```

if(strPayload == "0") {
    digitalWrite(D7, LOW);
    toggleState_5 = 0;
    client->publish(pub5, "0");
}

if(strPayload == "1") {
    digitalWrite(D7, HIGH);
    toggleState_5 = 1;
    client->publish(pub5, "1");
}
}

else if(strstr(topic, sub6)) {
    if(strPayload == "0") {
        digitalWrite(D7, HIGH);
        toggleState_6 = 0;
        client->publish(pub6, "0");
    }
    if(strPayload == "1") {
        toggleState_6 = 1;
        client->publish(pub6, "1");
    }
}
else
{
    Serial.println("unsubscribed topic");
}
}

```

```
}
```

```
void setup() {  
    // put your setup code here, to run once:  
    delay(500);  
    // When opening the Serial Monitor, select 9600 Baud  
    Serial.begin(115200);  
    delay(500);  
  
    LittleFS.begin();  
    setup_wifi();  
    setDateTime();  
  
    timeClient.begin();  
    // Set offset time in seconds to adjust for your timezone, for example:  
    // GMT +1 = 3600  
    // GMT +8 = 28800  
    // GMT -1 = -3600  
    // GMT0=0  
    timeClient.setTimeOffset(25200);  
    // pinMode(LED_BUILTIN, OUTPUT); // Initialize the LED_BUILTIN pin as  
    // an output  
    pinMode(D1, OUTPUT);  
    pinMode(D2, OUTPUT);  
    pinMode(D3, OUTPUT);  
    pinMode(D4, OUTPUT);  
    pinMode(D0, OUTPUT);  
    pinMode(D7, OUTPUT);  
}
```

```

pinMode(D8, INPUT);

pinMode(SwitchPin1, INPUT_PULLUP);
pinMode(SwitchPin2, INPUT_PULLUP);
pinMode(SwitchPin4, INPUT_PULLUP);

//During Starting all Relays should TURN OFF
digitalWrite(D1, HIGH);
digitalWrite(D2, HIGH);
digitalWrite(D0, HIGH);
digitalWrite(D3, HIGH);
digitalWrite(D4, HIGH);
digitalWrite(D7, HIGH);

// you can use the insecure mode, when you want to avoid the
certificates //espclient->setInsecure();

int numCerts = certStore.initCertStore(LittleFS, PSTR("/certs.idx"),
PSTR("/certs.ar"));
Serial.printf("Number of CA certs read: %d\n", numCerts);
if (numCerts == 0) {
    Serial.printf("No certs found. Did you run certs-from-mozilla.py and upload the
LittleFS directory before running?\n");
    return; // Can't connect to anything w/o certs!
}

//Firebase.begin(firebase_url, auth); config.host =
FIREBASE_HOST; config.signer.tokens.legacy_token =
FIREBASE_AUTH;

```

```

//Firebase.reconnectWiFi(true);

Firebase.begin(&config, &auth);

BearSSL::WiFiClientSecure *bear = new BearSSL::WiFiClientSecure();
// Integrate the cert store with this connection
bear->setCertStore(&certStore);

client = new PubSubClient(*bear);

client->setServer(mqtt_server, 8883);
client->setCallback(callback);
}

void loop() {
// put your main code here, to run repeatedly:
if (!client->connected()) {
    reconnect();
}
client->loop();

if(toggleState_6 == 1) {
    int ldrStatus = digitalRead(D8);
    if(ldrStatus == 1) {
        digitalWrite(D7, LOW);
    } else if(ldrStatus == 0) {
        digitalWrite(D7, HIGH);
    }
}
}

```

```

}

timeClient.update();
unsigned long epochTime =
timeClient.getEpochTime(); Serial.print("Epoch Time:
"); Serial.println(epochTime);

String formattedTime = timeClient.getFormattedTime();
Serial.print("Formatted Time: ");
Serial.println(formattedTime);

//Get a time structure
struct tm *ptm = gmtime ((time_t *)&epochTime);

int monthDay = ptm->tm_mday;
Serial.print("Month day: ");
Serial.println(monthDay);

int currentMonth = ptm->tm_mon+1;
Serial.print("Month: ");
Serial.println(currentMonth);

String currentMonthName = months[currentMonth-1];
Serial.print("Month name: ");
Serial.println(currentMonthName);

int currentYear = ptm->tm_year+1900;
Serial.print("Year: ");

```

```

Serial.println(currentYear);

//Print complete date:
String currentDate = String(currentYear) + "-" + String(currentMonth) + "-" +
String(monthDay);
Serial.print("Current date: ");
Serial.println(currentDate);

Serial.println("");
delay(1000);

float voltage = pzem.voltage();
float current = pzem.current();
float power = pzem.power();
float energy = pzem.energy();
float totalHarga = energy * harga_KWh;

unsigned long now = millis();

if (now - lastMsg > 2000) {
    if (Firebase.setFloat(fbdo, path + "/" + currentDate + "/current", current))
    {
        Serial.println("Passed");
        Serial.println("Path: " + fbdo.dataPath());
        Serial.println("Type: " + fbdo.dataType());
        Serial.println("Etag: " + fbdo.ETag());
        Serial.print("Value: ");
        //printResult(fbdo);
    }
}

```

```

        Serial.println("-----");
        Serial.println();
    } else { Serial.println("Failed");
        Serial.println("Reason: " + fbdo.errorReason());
        Serial.println("-----");

        Serial.println();
    }

    if (Firebase.setFloat(fbdo, path + "/" + currentDate + "/voltage", voltage))
    {
        Serial.println("Passed");
        Serial.println("Path: " + fbdo.dataPath());
        Serial.println("Type: " + fbdo.dataType());
        Serial.println("Etag: " + fbdo.ETag());
        Serial.print("Value: ");
        //printResult(fbdo);
        Serial.println("-----");
        Serial.println();
    } else { Serial.println("Failed");
        Serial.println("Reason: " +
        fbdo.errorReason()); Serial.println("-----
        -----");
        Serial.println();
    }

    if (Firebase.setFloat(fbdo, path + "/" + currentDate + "/watt", power))
    {

```



```

Serial.println("Passed");
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
    Serial.println("Etag: " + fbdo.ETag());
    Serial.print("Value: ");
    //printResult(fbdo);
    Serial.println("-----");
    Serial.println();
} else { Serial.println("Failed");
    Serial.println("Reason: " +
    fbdo.errorReason()); Serial.println("-----
    -----");
    Serial.println();
}
if (Firebase.setFloat(fbdo, path + "/" + currentDate + "/rp", totalHarga))
{
Serial.println("Passed");
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
    Serial.println("Etag: " + fbdo.ETag());
    Serial.print("Value: ");
    //printResult(fbdo);
    Serial.println("-----");
    Serial.println();
} else { Serial.println("Failed");
    Serial.println("Reason: " +
    fbdo.errorReason()); Serial.println("-----
    -----");
}

```

```

    Serial.println();
}
if (Firebase.setFloat(fbdo, path + "/" + currentDate + "/kwh", energy))
{
    Serial.println("Passed");
    Serial.println("Path: " + fbdo.dataPath());
    Serial.println("Type: " + fbdo.dataType());
    Serial.println("Etag: " + fbdo.ETag());
    Serial.print("Value: ");
    //printResult(fbdo);
    Serial.println("-----");
    Serial.println();
} else {
    Serial.println("Failed");
    Serial.println("Reason: " + fbdo.errorReason());
    Serial.println("-----");
    Serial.println();
}
}
}

```