



INSTITUT TEKNOLOGI BANDUNG

PROGRAM STUDI TEKNIK ELEKTRO

JALAN GANESHA NO. 10 Gedung Labtek V Lantai 2 **(022)2508135-36, (022)250 0940**
BANDUNG 40132

Dokumentasi Produk Tugas Akhir

Lembar Sampul Dokumen

Judul Dokumen

**TUGAS AKHIR TEKNIK ELEKTRO:
Sistem Pengisian Daya Portabel Baterai
Kendaraan Listrik PT. Oyika Powered Solution**

Jenis Dokumen

IMPLEMENTASI

Catatan: Dokumen ini dikendalikan penyebarannya oleh Prodi Teknik Elektro ITB

Nomor Dokumen

B400-TA2122.01.024

Nomor Revisi

01

Nama File

B400-TA2122.01.024-002

Tanggal Penerbitan

26 June 2022

Unit Penerbit

Prodi Teknik Elektro - ITB

Jumlah Halaman

101

(termasuk lembar sampul ini)

Data Pemeriksaan dan Persetujuan

Ditulis Oleh	Nama Tanggal	Dhanurangga Al Fadh 26 June 2022	Jabatan Tanda Tangan	Anggota
		Danu Ihza Pamungkas 26 June 2022	Jabatan Tanda Tangan	
		Kevin Naoko 26 June 2022	Jabatan Tanda Tangan	

Diperiksa Oleh	Nama	Dr. Muhammad Amin Sulthoni, S.T. M.T.	Jabatan	Dosen Pembimbing
	Tanggal	26 June 2022	Tanda Tangan	
Disetujui Oleh	Nama	Dr. Eng. Arwindra Rizqiawan, S.T., M.T.	Jabatan	Dosen Pembimbing
	Tanggal	26 June 2022	Tanda Tangan	
	Nama	Dr. Muhammad Amin Sulthoni, S.T. M.T.	Jabatan	Dosen Pembimbing
	Tanggal	26 June 2022	Tanda Tangan	

DAFTAR ISI

DAFTAR ISI.....	3
CATATAN SEJARAH PERBAIKAN DOKUMEN.....	4
PROPOSAL PROYEK PENGEMBANGAN	5
1 PENGANTAR	5
1.1 RINGKASAN ISI DOKUMEN	5
1.2 TUJUAN PENULISAN DAN APLIKASI/KEGUNAAN DOKUMEN	5
1.3 REFERENSI	5
1.4 DAFTAR SINGKATAN.....	5
2 IMPLEMENTASI.....	7
2.1 SUB-SISTEM CHARGER.....	7
<i>2.1.1 Implementasi</i>	<i>7</i>
<i>2.1.2 Pengujian</i>	<i>23</i>
2.2 SUB-SISTEM KONTROL.....	25
<i>2.2.1 Implementasi</i>	<i>25</i>
<i>2.2.2 Pengujian</i>	<i>42</i>
2.3 SUB-SISTEM INTERFACE.....	43
<i>2.3.1 Implementasi</i>	<i>43</i>
<i>2.3.2 Pengujian</i>	<i>45</i>
2.4 SUB-SISTEM IoT	45
<i>2.4.1 Implementasi</i>	<i>45</i>
<i>2.4.2 Pengujian</i>	<i>90</i>
2.5 SUB-SISTEM SENSING	100
<i>2.5.1 Implementasi</i>	<i>100</i>
<i>2.5.2 Pengujian</i>	<i>103</i>
2.6 SUB-SISTEM POWER SUPPLY.....	103
<i>2.6.1 Implementasi</i>	<i>103</i>
<i>2.6.2 Pengujian</i>	<i>104</i>
2.7 CASING	104
3 ANALISIS PENGERJAAN IMPLEMENTASI.....	105
4 HASIL AKHIR.....	105
5 LAMPIRAN.....	108

Catatan Sejarah Perbaikan Dokumen

Versi, Tgl, Oleh	Perbaikan
1.0, 12 Januari 2022, KN, DIP, DAF	Dokumen dibuat
1.2, 12 Maret 2022, KN, DIP, DAF	Menambahkan bagian subsistem power supply dan sensing
1.3, 13 Maret 2022, KN, DIP, DAF	Menambahkan penjelasan dan kode pada bagian subsistem kontrol
1.4, 15 Maret 2022, KN, DIP, DAF	Menambahkan pin mapping pada setiap subsistem dan menambahkan bagian penjelasan subsistem kontrol dan charger
1.5, 17 Maret 2022, KN, DIP, DAF	Menghapus bagian template dokumen yang tidak diperlukan, menambahkan hasil akhir
2.0, 4 April 2022, KN, DIP, DAF	Mengupdate subsistem IoT, khususnya di metode publish produk ke server
2.1, 15 April 2022, KN, DIP, DAF	Mengupdate bagian casing Mengupdate bagian subsistem IoT, terkait data yang perlu ditampilkan pada dashboard
2.2 14 Mei 2022, KN, DIP, DAF	Mengupdate kode subsistem kontrol
2.2 26 Juni 2022, KN, DIP, DAF	Mengupdate proses implementasi dan pengujian subsistem charger, mengubah formatting, menambahkan keterangan pada gambar UI dasboard di subsistem IoT

Proposal Proyek Pengembangan

1 Pengantar

1.1 Ringkasan Isi Dokumen

Dokumen B400 ini berisi implementasi dari konsep sistem yang telah dirancang pada dokumen B100 hingga B300 sebelum ini. Metode implementasi pada dokumen ini mengacu pada desain sistem yang telah dituliskan pada dokumen B300 dan juga spesifikasi yang telah ditentukan pada B200. Pada dokumen ini juga akan dijelaskan mengenai cara pengujian untuk setiap subsistem. Lalu akan dianalisis setiap subsistem yang telah diimplementasikan. Pengujian pada dokumen ini dilakukan untuk membandingkan hasil implementasi dengan spesifikasi yang telah ditentukan sebelumnya. Analisis pada dokumen ini dibuat untuk memperdalam pemahaman mengenai pengembangan proyek tugas akhir ini dan dapat mengatasi masalah yang muncul.

1.2 Tujuan Penulisan dan Aplikasi/Kegunaan Dokumen

Berisi tujuan/maksud penulisan dokumen, dan ditujukan kepada siapa.

1. Sebagai dokumen yang berisi penjelasan mengenai implementasi tiap subsistem yang telah dirancang
2. Sebagai dokumentasi pengerjaan proyek tugas akhir
3. Sebagai dokumen yang menjadi acuan dalam proses pengembangan proyek tugas akhir. Dokumen ini mencakup kesesuaian spesifikasi desain dan jadwal pengerjaan tugas akhir.

Dokumen ini ditujukan kepada dosen pembimbing Tugas Akhir, serta tim Tugas Akhir Program Studi Teknik Elektro ITB sebagai bahan penilaian Tugas Akhir.

1.3 Referensi

- [1] <https://mosquitto.org/>
- [2] <https://flask.palletsprojects.com/en/2.0.x/>
- [3] <https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>
- [4] [\(PDF\) On the 3D printing of recycled ABS, PLA and HIPS thermoplastics for structural applications \(researchgate.net\)](#)
- [5] [3D Printing: Bahan-Bahan yang Digunakan untuk Print 3D \(fomustudio.com\)](#)
- [6] https://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf, Halaman 101
- [7] https://www.elecrow.com/wiki/images/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf, Halaman 198
- [8] <https://www.gme.cz/data/attachments/dsh.775-083.1.pdf>, Figure 8, Halaman 21
-

1.4 Daftar Singkatan

SINGKATAN	ARTI
AC	Alternative Current
API	Application Programming Interface
CGI	Cell Global Identity
BMS	Battery Management System

SINGKATAN	ARTI
BTS	Base Transceiver Station
DC	Direct Current
GCE	Google Compute Engine
IoT	Internet of Things
LED	Light Emitting Diode
MQTT	Message Queuing Telemetry Transport
PCB	Printed Circuit Board
PWM	Pulse Width Modulation
SCL	Serial Clock
SDA	Serial Data
VM	Virtual Machine

2 Implementasi

Sistem pengisian daya baterai yang dirancang dapat dibagi menjadi 6 subsistem sesuai dengan fungsinya. Subsistem tersebut terdiri dari subsistem charger, subsistem kontrol, subsistem IoT, subsistem interface, subsistem power supply, dan subsistem sensing. Masing-masing subsistem tersebut dapat diklasifikasikan kembali sesuai fungsinya dengan lebih spesifik. Proses perancangan dan manufaktur dilakukan dengan membuat sistem secara modular. Sistem modular tersebut digunakan untuk memudahkan dalam proses *research and development* serta memudahkan dalam proses manufakturing. Berikut ini penjelasan lebih detail dari masing-masing subsistem.

2.1 Sub-Sistem Charger

Subsistem charger mencakup high voltage untuk pengisian daya serta low voltage untuk driver MOSFET. Subsistem charger ini memiliki komunikasi dengan subsistem kontrol berupa PWM untuk mosfet dan PWM untuk mengosongkan daya yang tersisa pada komponen setelah digunakan. Subsistem charger ini memiliki board PCB sendiri. Pada board tersebut terdapat juga bagian dari subsistem sensing yaitu sensor suhu untuk melakukan pembacaan suhu pada mosfet. Desain dari board subsistem charger ini dapat dilihat lebih detail pada Subbab 2.1.1.

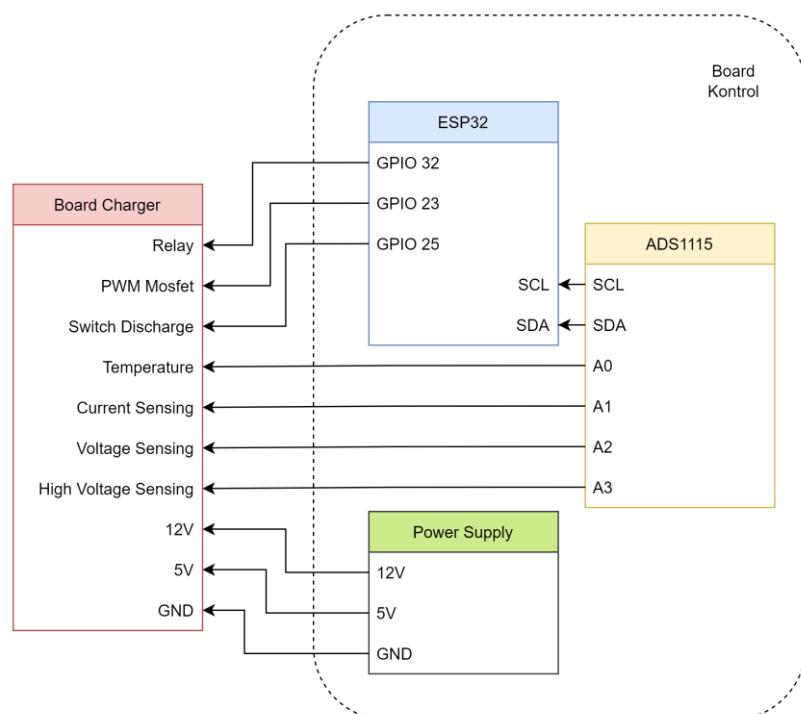
2.1.1 Implementasi

Pada subbab ini, akan dijelaskan mengenai proses implementasi subsistem charger selama semester genap oleh tim penulis. Secara umum, terdapat banyak sekali kendala yang menghambat progress pengembangan subsistem charger ini, namun setelah dilakukan troubleshooting dan debugging, subsistem ini berhasil direalisasikan sesuai dengan spesifikasi output arus yang diperlukan, yakni 10A. Perlu diketahui bahwa terdapat penggantian beberapa bagian pada subsistem ini, yakni induktor dan rangkaian MOSFET driver, sehingga akan terdapat beberapa penggunaan komponen yang berbeda dengan desain di B300. Pembahasan mengenai proses implementasi ini akan dibagi menjadi 4 bagian. Tiap bagiannya melambangkan revisi rangkaian subsistem charger yang dilakukan oleh tim penulis. Selain itu, daftar komponen untuk subsistem ini hanya akan dicantumkan pada Subbab 2.1.1.4, yakni perancangan board charger revisi terakhir.

Pada perancangan awal, perangkat keras subsistem charger dibuat terpisah dengan subsistem kontrol untuk meningkatkan modularitas dari produk sekaligus mempermudah troubleshooting saat tahap perancangan. Subsistem charger ini dirancang pada board sendiri, terpisah dari subsistem lainnya. Hal tersebut karena tegangan yang digunakan pada subsistem charger merupakan tegangan *high voltage*, sehingga perlu dilakukan pemisahan antara charger dan kontrol. Pemisahan tersebut dilakukan untuk mendukung aspek keamanan dan kemudahan manufaktur.

Subsistem charger memiliki hubungan dengan subsistem kontrol dan power supply. Subsistem power supply berfungsi untuk menyalurkan tegangan 12V dan 5V untuk mosfet driver pada subsistem charger. Pada subsistem kontrol, charger dikendalikan melalui PWM yang dikeluarkan oleh ESP32. Oleh karena itu, diperlukan sebuah penghubung antara board charger dan board kontrol. Board penghubung tersebut berisi 6 koneksi untuk satu board charger. Untuk menghubungkan kedua board tersebut digunakan koneksi JST XH 2.54. Konektor tersebut digunakan karena banyak di pasaran dan mudah untuk dipasang dan

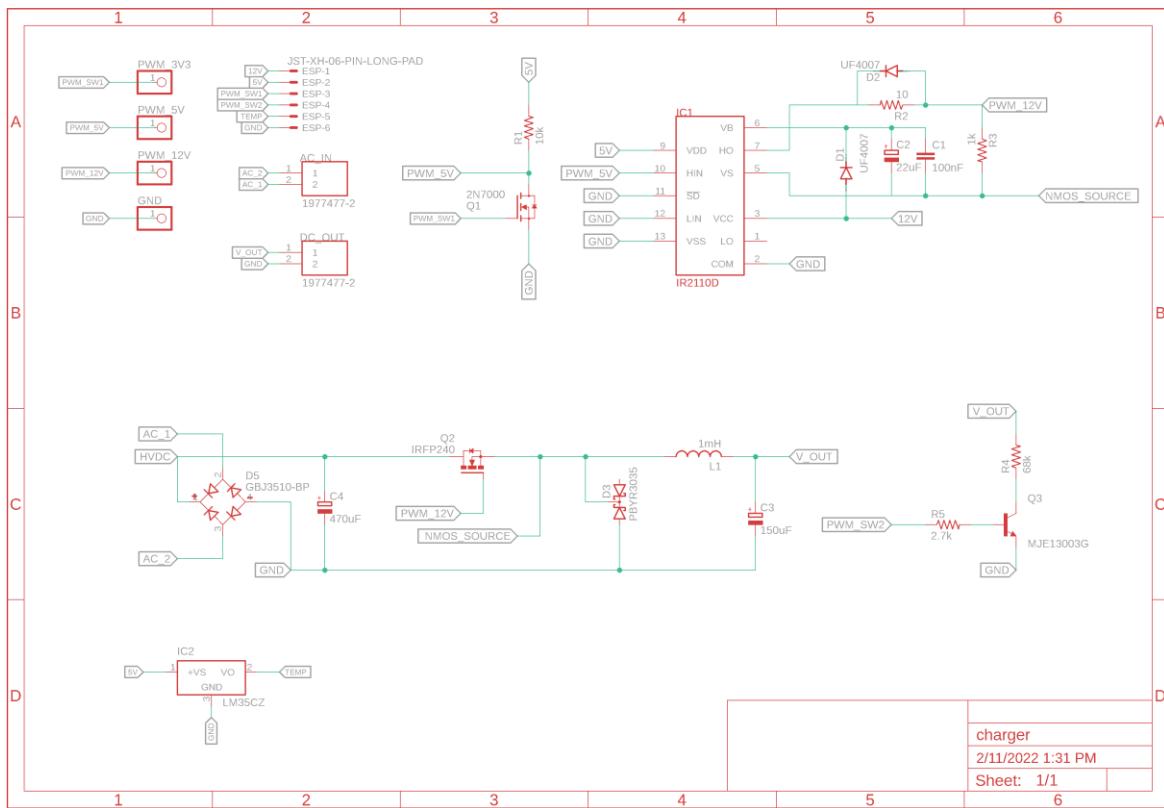
dilepas. Untuk hubungan subsistem charger dengan subsistem lainnya dapat dilihat lebih jelas pada Gambar 2.1.1 dibawah ini.



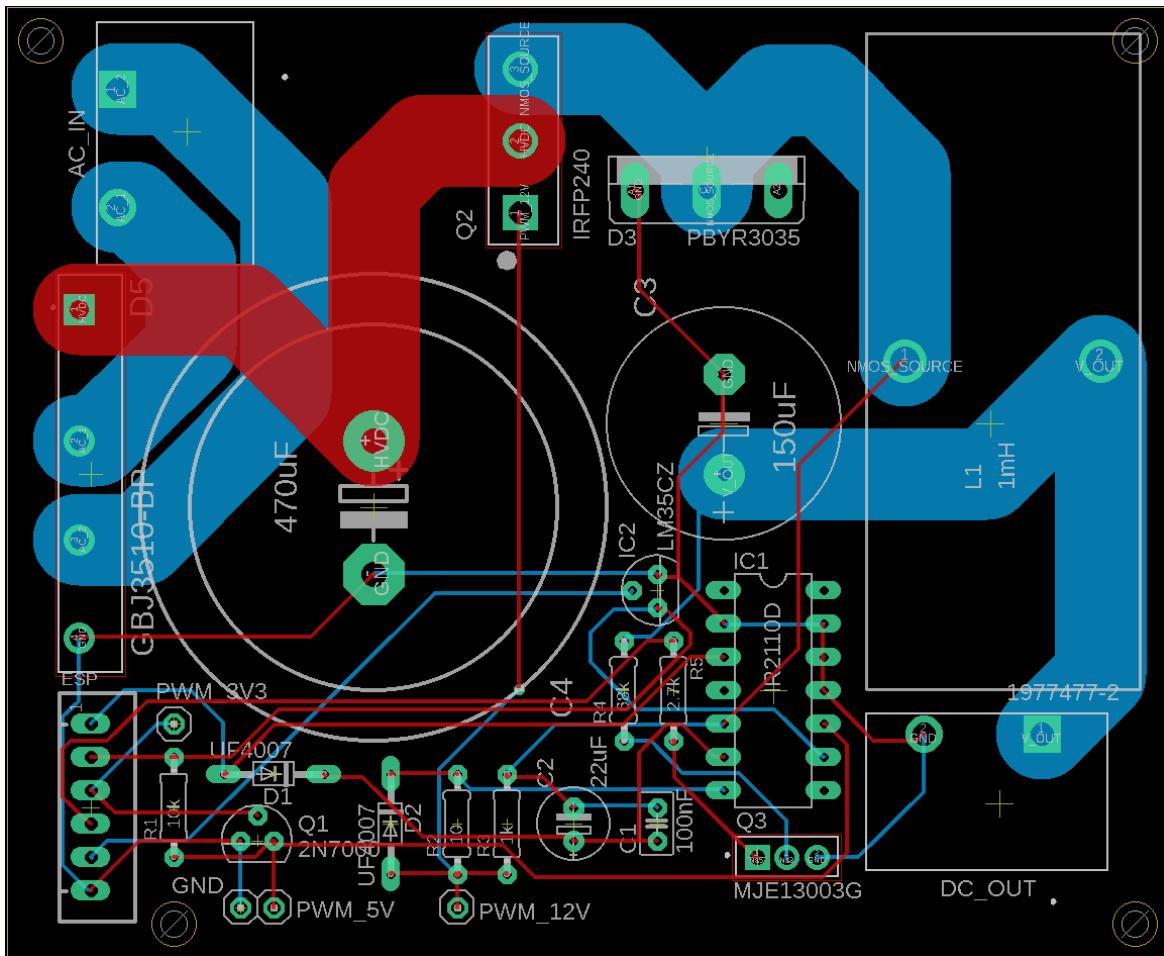
Gambar 2.1.1 Pin Mapping Subsistem Charger

2.1.1.1 Rangkaian Charger Revisi 1

Skematik dan layout dari board charger dapat dilihat pada Gambar 2.1.2 dan Gambar 2.1.3 berikut ini.



Gambar 2.1.2 Skematik Subsistem Charger revisi ke-1



Gambar 2.1.3 Layout subsistem charger revisi ke-1

Dari Gambar 2.1.3 diatas, terlihat ada jalur yang memiliki ukuran yang cukup besar dibandingkan dengan jalur lainnya pada PCB. Jalur yang memiliki ukuran besar tersebut merupakan sisi *high voltage* yang memiliki tegangan dan arus yang cukup besar (hingga maksimum 20A), sehingga diperlukan jalur serta *clearance* yang cukup besar dengan jalur-jalur yang berada di sebelahnya.

Namun terdapat kendala pada board ini, yakni ground plane yang cukup dekat dengan seluruh jalur, sehingga saat dilakukan percobaan, terdapat kejadian dimana tegangan 311V dari output rectifier terhubung singkat dengan ground. Namun karena pencetakan board ini dilakukan sebanyak 3 buah, percobaan menggunakan board ini masih dapat dilakukan. Setelah menyolder dengan berhati hati, dapat dilakukan pengujian terhadap rangkaian charger ini. Pengujian output yang dilakukan ialah kendali *openloop* sederhana untuk melihat tegangan yang dikeluarkan oleh subsistem charger berdasarkan beberapa variasi duty cycle tertentu. Load yang digunakan ialah resistor sederhana dengan resistansi 50Ω . Hasil pengujian *openloop* dapat dilihat pada Tabel 2.1.1

Tabel 2.1.1 Output tegangan dan arus subsistem charger saat pengujian open loop

Duty Cycle	Tegangan Ideal	Tegangan Terukur	ΔV_o (%)	Arus Terukur	Suhu MOSFET (T+30 sekon)
5.8%	18.29	14.88	81.34	0.33	33.6
7.8%	24.39	19.8	81.17	0.45	34.8
9.8%	30.49	31.8	104.30	0.72	50
11.7%	36.59	51.2	139.94	1.02	60

Pada Tabel 2.1.1 terlihat bahwa tegangan output yang terukur untuk variasi duty cycle 5.8% dan 7.8% memiliki ΔV_o yang sama, yaitu 81%. Hal ini disebabkan karena tegangan hasil rectifier tidak sepenuhnya DC, meskipun adanya *smoothing capacitor* di output rectifier. Secara umum, pengujian pada kedua variasi duty cycle ini memiliki output dan suhu yang stabil seiring berjalannya waktu.

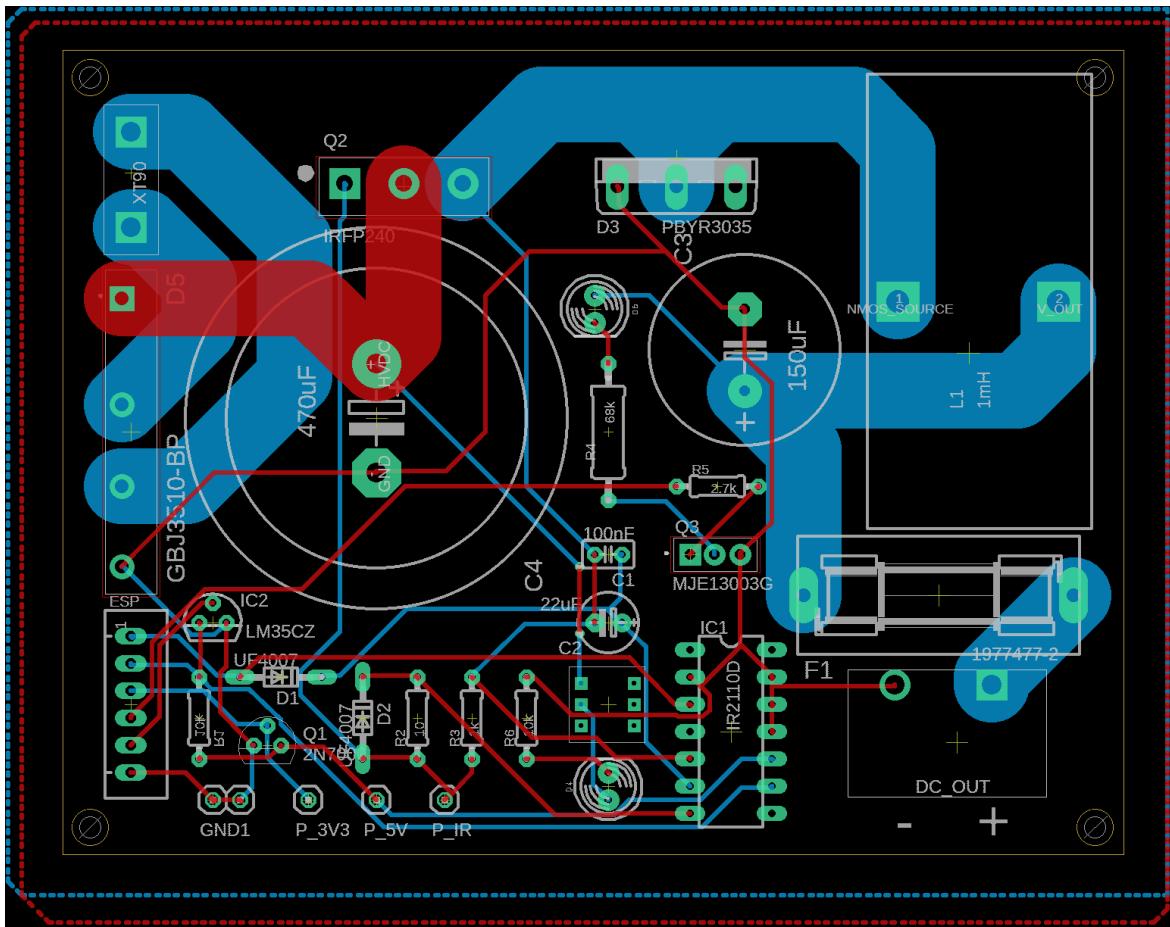
Namun untuk duty cycle 9.8% dan 11.7%, tegangan output yang terukur memiliki ΔV_o lebih dari 100%, yang artinya terdapat kesalahan pada proses konversi tegangan. Pengujian pada duty cycle 9.8% tidak memiliki output yang stabil, dalam arti seiring berjalannya waktu, tegangan output charger perlahan meningkat terus menerus.

Perlu diketahui pula, setelah kurang lebih 10 detik setelah duty cycle diganti ke 11.7%, tegangan output terus meningkat hingga 200V lebih, yang menandakan bahwa MOSFET telah rusak. Hipotesis tim penulis untuk permasalahan ini ialah heatsink yang kurang memadai pada bagian MOSFET.

Setelah MOSFET mengalami *failure*, kerap kali rangkaian MOSFET *driver* pada PCB juga mengalami kerusakan, sehingga perlu dilakukan pencabutan terhadap beberapa komponen di rangkaian *driver*, hingga penyolderan ulang dari seluruh rangkaian charger di PCB baru. Untuk rancangan PCB revisi ke-1, dilakukan sebanyak 3 kali penyolderan dan pengujian, dan rangkaian charger selalu mengalami kerusakan setelah pengoperasian beberapa menit di arus dan tegangan rendah.

2.1.1.2 Rangkaian Charger Revisi 2

Berdasarkan kendala yang ditemukan pada PCB revisi ke-1, yaitu kesulitan untuk melakukan *debugging* pada rangkaian driver dan *ground plane* yang terlalu dekat, telah dirancang PCB revisi ke-2, yang memuat beberapa *probe points* agar dapat dilakukan pemantauan tegangan pada titik tertentu. Selain itu, jarak *ground plane* dengan jalur lain juga diperbesar untuk meminimalisir kemungkinan terjadi *short circuit*. Board PCB versi ke-2 ini dapat dilihat pada Gambar 2.1.4 berikut.



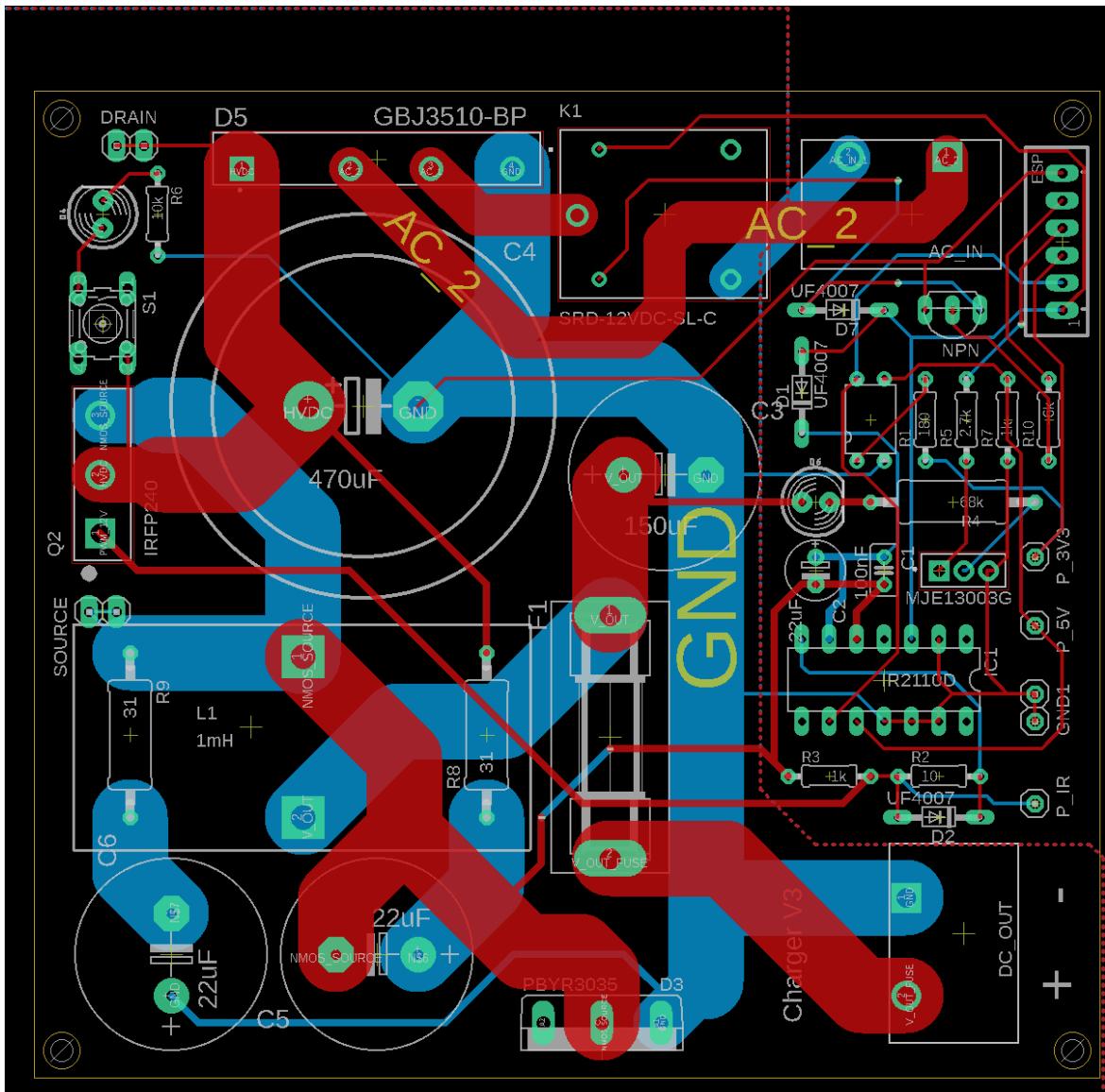
Gambar 2.1.4 Layout Subsistem Charger revisi ke-2

Setelah dilakukan pemasangan komponen pada PCB, pengujian dilanjutkan kembali dengan output berupa *resistor load* yang sama seperti sebelumnya, yakni 50Ω . Permasalahan yang sama juga masih terjadi. Setelah dilakukan troubleshooting, ditemukan bahwa suhu pada MOSFET mengalami peningkatan yang cukup tinggi, meskipun output arus dari rangkaian hanya sebesar 500mA.

Menimbang jarak yang cukup dekat antara MOSFET dengan *power diode*, diputuskan untuk tidak melakukan pemasangan MOSFET secara langsung di board, melainkan melayang menggunakan kabel. Setelah dibubuh *heatsink* dan kipas, hal yang serupa masih terjadi. MOSFET kerap mengalami failure dan berhenti berfungsi, dan rangkaian driver yang digunakan pun ikut mengalami kerusakan.

2.1.1.3 Rangkaian Charger Revisi 3

Kemudian dilakukan perancangan terhadap PCB revisi ke-3, dimana MOSFET, dioda, serta rectifier diletakkan pada sisi yang berbeda beda agar dapat dihubunkan heatsink yang cukup besar pada tiap komponen tersebut. Layout dari PCB tersebut dapat dilihat pada Gambar 2.1.5



Gambar 2.1.5 Layout Subsistem Charger revisi ke-3

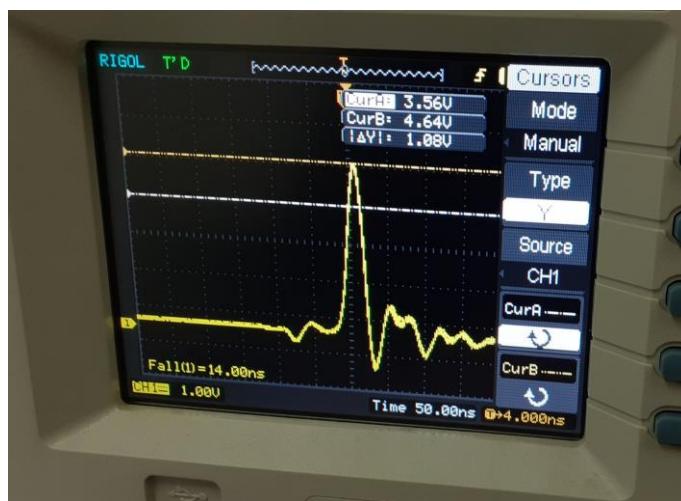
Pada revisi ketiga, terdapat pula rangkaian *relay* yang dapat digunakan untuk memutus daya secara mekanik apabila diperlukan, mengingat MOSFET saat proses implementasi masih kerap mengalami kerusakan. Pengujian menggunakan PCB revisi ke-3 ini masih membuat hasil yang sama seperti pada percobaan sebelumnya, yakni MOSFET yang *overheat*, kemudian mengalami kerusakan. Pengujian yang dilakukan ialah pengujian *closed loop* untuk melihat output charger terhadap perubahan setpoint arus yang diberikan. Hasil pengujian *closed loop* dapat dilihat pada Tabel 2.1.2

Tabel 2.1.2 Output tegangan dan arus subsistem charger saat pengujian closed loop

Setpoint arus (mA)	Arus terukur (mA)	Tegangan terukur (V)	Suhu terukur
100	100	4.8	29.1
200	200	9.1	28

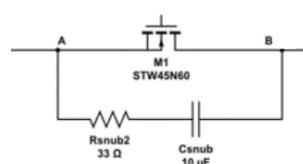
300	300	13.2	30
400	390	17.2	40
500	480	21.3	29.4
600	590	26.2	47
700	690	29.8	50+

Pada Tabel 2.1.2 terlihat bahwa secara umum, subsistem charger dapat menyesuaikan arus outputnya dengan setpoint yang diberikan. Permasalahan yang sama juga terjadi seperti pada percobaan *openloop*, yakni arus output yang melebihi 600mA belum memungkinkan, karena terjadi kenaikan suhu yang drastis, meskipun sudah dibubuhinya heatsink yang cukup besar. Setelah melakukan *troubleshooting* lebih lanjut pada Lab Dasar, ditemukan bahwa terdapat spiking pada node V_S . Spike tersebut bisa dilihat pada Gambar 2.1.6



Gambar 2.1.6 Spike tegangan di V_S MOSFET saat sebelum switch tertutup

Tim penulis menduga bahwa kerusakan terjadi karena hal ini, sehingga untuk meredam spike ini, dipasangkan rangkaian snubber RC secara paralel dengan MOSFET. Lebih jelasnya, peletakkan snubber dapat dilihat pada Gambar 2.1.7



Gambar 2.1.7 Konfigurasi rangkaian snubber RC untuk menekan spike tegangan

Pemilihan nilai R dan C tidak dapat dilakukan dengan perhitungan, karena spike tegangan ini terjadi akibat induktansi parasitik yang terdapat pada rangkaian. Namun, dapat digunakan suatu persamaan sebagai titik acuan dalam pemilihan R dan C untuk pertama kali. Dari nilai awal tersebut, dapat dilakukan *tuning* manual agar spike teredam secara

optimal. Persamaan yang digunakan untuk menentukan nilai R dan C dapat dilihat pada bagian dibawah ini

$$R_s = \frac{V_{off}}{I_D} = \frac{311}{10} = 31\Omega$$

$$C_s = \frac{1}{V_{off}^2 \times f_s} = \frac{1}{311^2 \times 50 \times 10^3} = 206.7 \text{ pF}$$

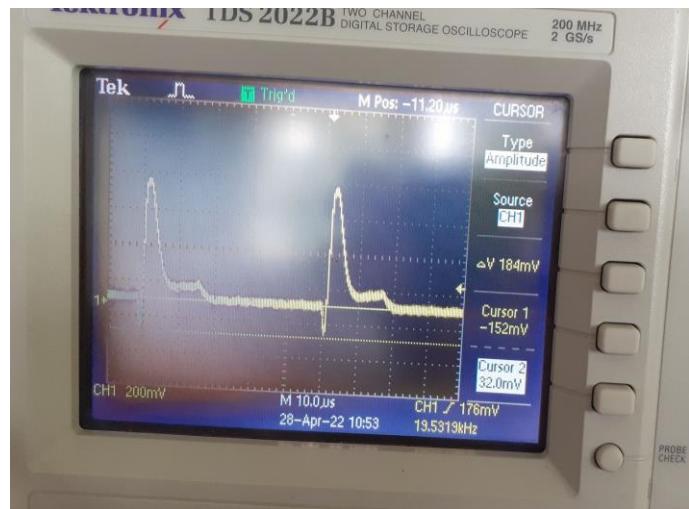
Setelah didapatkan nilai R dan C acuan, dilakukan percobaan dengan beberapa kombinasi nilai komponen RC. Hasil percobaan tersebut dapat dilihat pada Tabel 2.1.3

Tabel 2.1.3 Perubahan spike tegangan dengan beberapa variasi kombinasi nilai RC

R (Ω)	C (F)	Spike (V)
-	-	5
100	100p	4.9
100	1n	4.2
100	10n	3.7
31	10n	2.1
9	10n	2.1
31	100n	2.1

Merujuk pada Tabel 2.1.3terlihat bahwa spike tegangan tidak dapat turun dibawah 2.1V, meskipun nilai resistansi dan kapasitansi dinaikkan. Dengan itu, akan digunakan kombinasi nilai RC sebesar 31Ω dan 10nF .

Meskipun telah digunakan snubber RC untuk meredam spike tegangan yang terjadi, kegagalan pada MOSFET masih terjadi pada nominal arus output 600mA. Tim penulis kemudian menggunakan peralatan yang terdapat pada Lab Konversi untuk melihat karakteristik arus yang mengalir di rangkaian pada setiap *branch*, dan ditemukan bahwa terdapat *spike arus* yang sangat tinggi pada I_D . Saat output rangkaian berada di 50mA, arus I_D maksimal berada di kisaran 700mA, sebesar 14 kali lipat dibanding arus outputnya. Karakteristik spike arus ini dapat dilihat lebih lanjut pada Gambar 2.1.8

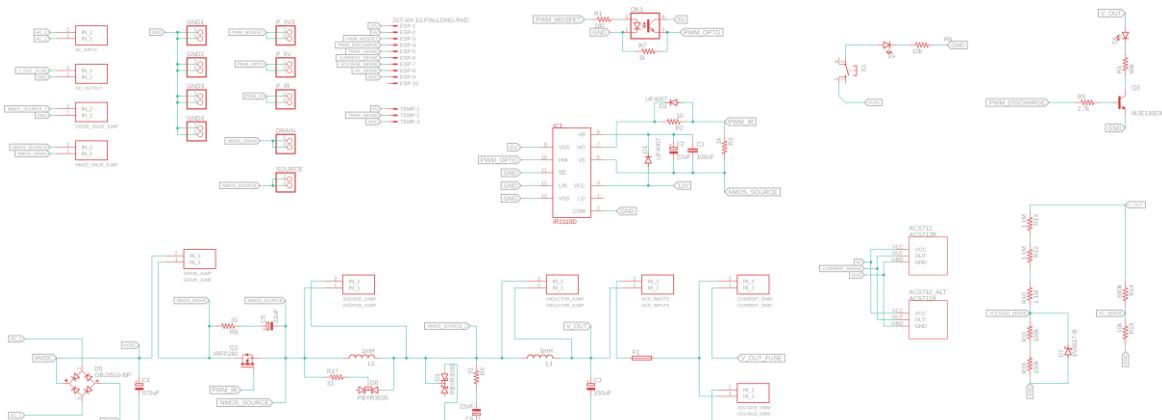


Gambar 2.1.8 Spike arus saat switch menutup

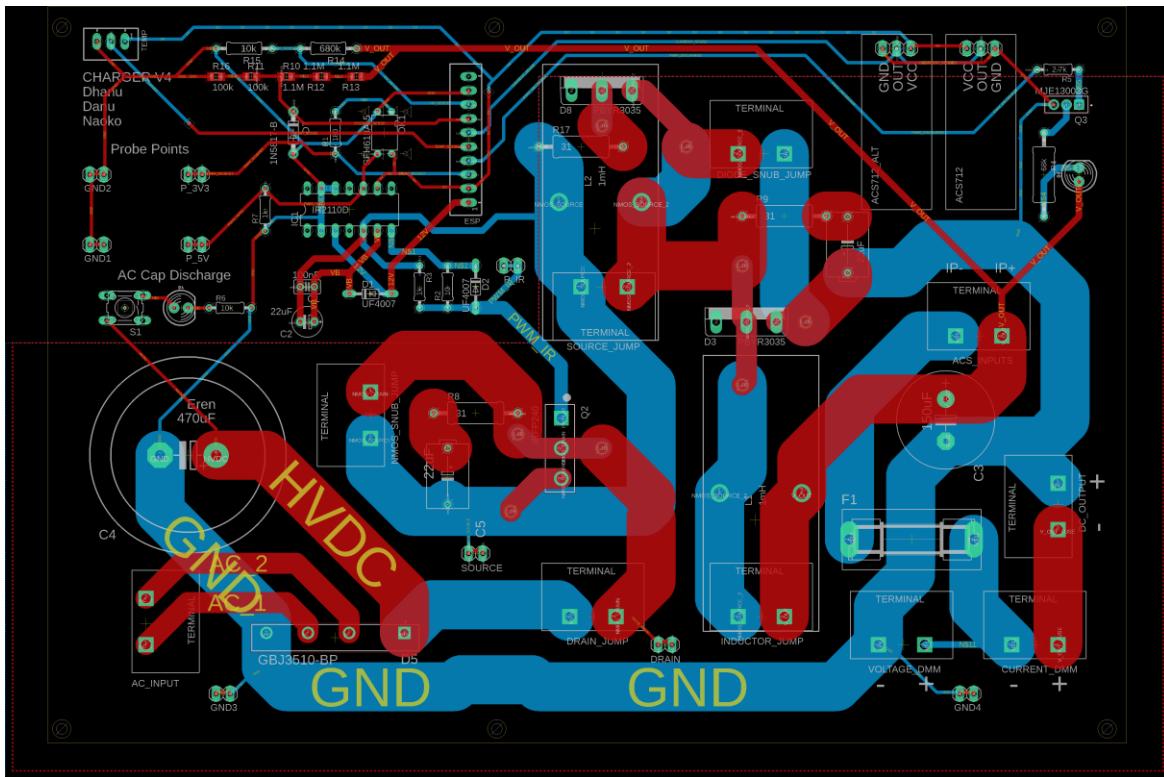
Salah satu upaya tim penulis dalam meredam spike ini ialah dengan memasang snubber RLD secara seri dengan MOSFET untuk menyerap sebagian spike yang terjadi pada I_D . Namun agar pengujian rangkaian snubber ini dapat dilakukan lebih leluasa, akan dilakukan perancangan PCB revisi ke-4.

2.1.1.4 Rangkaian Charger Revisi 4 (Final)

Pada perancangan PCB revisi ke-4, terdapat perubahan drastis dari segi ukuran dan tata letak komponen. Ukuran PCB ini 4 kali lebih besar dibanding sebelumnya, yang berujuan untuk meletakkan beberapa terminal block yang dapat digunakan sebagai jumper untuk rangkaian snubber dan jumper untuk pengamatan arus. Adapun hasil perancangan PCB yang mengakomodasi komponen snubber ini dapat dilihat pada Gambar 2.1.9 dan Gambar 2.1.10



Gambar 2.1.9 Skematik Board Charger Revisi ke-4



Gambar 2.1.10 Desain board charger revisi ke-4

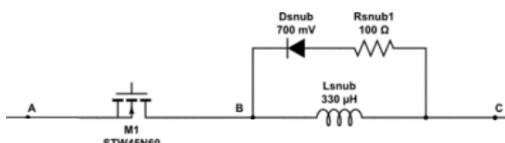
Pada board ini juga sudah disisipkan ruang untuk meletakkan komponen sensing, dengan tujuan merampingkan desain. Meninjau board yang telah dirancang pada Gambar 2.1.4, daftar komponen yang diperlukan untuk board charger ini dapat dilihat pada Tabel 2.1.4

Tabel 2.1.4 Daftar komponen untuk subsistem charger

Kategori	Komponen	Jumlah
AC to DC	Rectifier [GBJ3510]	1
	Elco 470uF 500V	1
Buck	PWM MOSFET [STW45N60]	1
	Power Diode [PA905C4]	1
	Induktor 1mH	1
	Elco 150uF 500V	1
MOSFET DRIVER	Lihat Subbab 2.1.1.5	-
IR2110 Driver	Optocoupler 2501	1
	R 5.6k	1

Discharge	Discharge BJT [MJE13003]	1
	R 68k 1W	1
	R 2.7k	1
Sensing	LM35	1
	ACS712	1
	R 680k	1
	R 10k	1
	R SMD 1.1M	3
	R SMD 100k	2
Lain lain	Generic terminal block (pitch 7.62 mm)	10
	JST XH10	1
	IC Socket DIP14	1
	IC Socket DIP8	1
	Heatsink besar untuk MOSFET	1
	Heatsink kecil untuk dioda	1
	Fuse holder	1

Setelah merangkai PCB tersebut, dilakukan pemasangan snubber RLD pada terminal block yang sesuai. Peletakan tersebut dapat dilihat pada Gambar 2.1.11



Gambar 2.1.11 Topologi snubber arus (RLD) yang digunakan

Berdasarkan percobaan yang telah dilakukan, perubahan nilai R pada rangkaian RLD tidak menyebabkan perubahan yang signifikan terhadap peredaman spike arus. Hal ini disebabkan karena rangkaian RD hanya digunakan sebagai pembuang muatan dari induktor snubber ketika sedang diluar posisi transien penutupan dan pembukaan switch. Yang penting disini ialah pemilihan nilai resistor agar *time constant* kombinasi rangkaian RL yang dimiliki lebih rendah dari frekuensi switching yang digunakan.

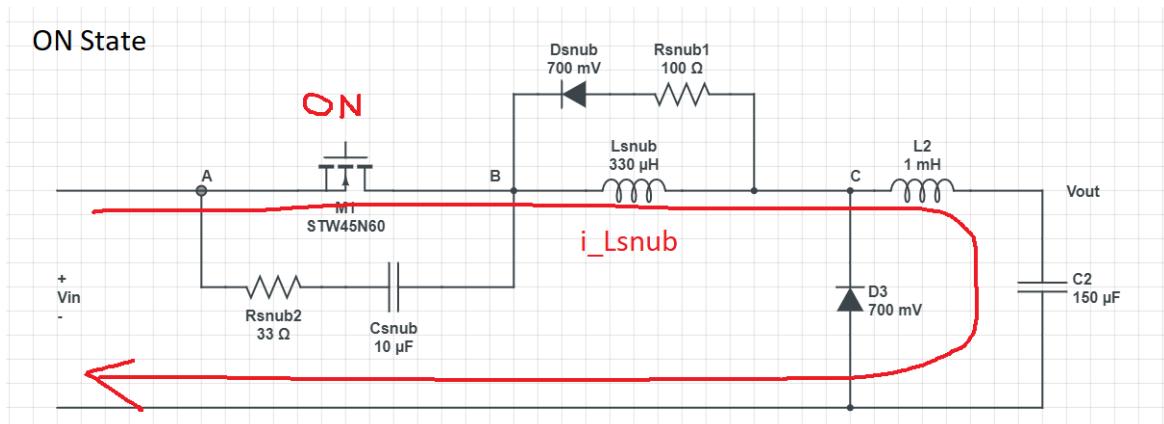
Kemudian, hasil percobaan beberapa kombinasi nilai induktor yang telah dilakukan dapat dilihat pada Tabel 2.1.5

Tabel 2.1.5 Perubahan spike tegangan dengan beberapa variasi kombinasi nilai RL

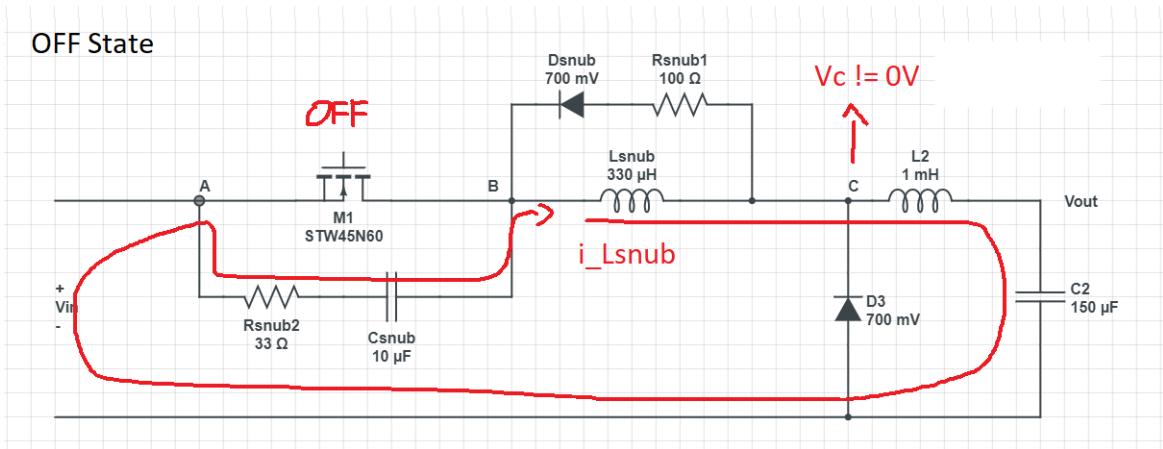
R (Ω)	L (uH)	I_{avg} (mA)	I_{spike} (mA)
-	-	51	990
500	112	51	180
1000	112	51	180
500	182	51	120
500	360	51	96
500	360	88	190
500	1000	51	67

Apabila dilihat hasil kombinasi RL pada Tabel 2.1.5, secara intuitif dapat dikatakan bahwa seiring peningkatannya nilai L, maka spike arus yang terjadi semakin rendah, dan seharusnya dipilih nilai L sebesar 1000uH. Namun, berdasarkan percobaan tim penulis, didapatkan bahwa seiring peningkatannya nilai L pada snubber, terjadi pembatasan tegangan output, sehingga tegangan output tidak dapat meningkat lagi setelah melewati batas tertentui. Hal ini terjadi karena pengisian induktor snubber menjadi lebih lambat.

Setelah pengujian rangkaian snubber arus dan tegangan secara independen selesai, tim penulis mencoba menggabungkan kedua rangkaian snubber tersebut. Namun ditemukan kendala bahwa penggunaan snubber arus tidak dapat dilakukan bersamaan dengan snubber tegangan. Lebih jelasnya, penyebab dari keterbatasan ini dapat dilihat pada Gambar 2.1.12 dan Gambar 2.1.13



Gambar 2.1.12 aliran arus saat MOSFET dalam kondisi ON



Gambar 2.1.13 Aliran arus saat MOSFET dalam kondisi OFF

Pada **Error! Reference source not found.** dapat dilihat bahwa saat switch dalam kondisi OFF, induktor snubber dapat membuang muatannya melalui rangkaian snubber RC yang berperan sebagai snubber tegangan. Hal ini menyebabkan tegangan di Node C tidak sama dengan 0V, yang menyebabkan D3 tidak dapat masuk ke mode forward biased. Karena itu, tegangan output dari rangkaian dengan kombinasi dua snubber ini tidak bisa mencapai tegangan yang seharusnya.

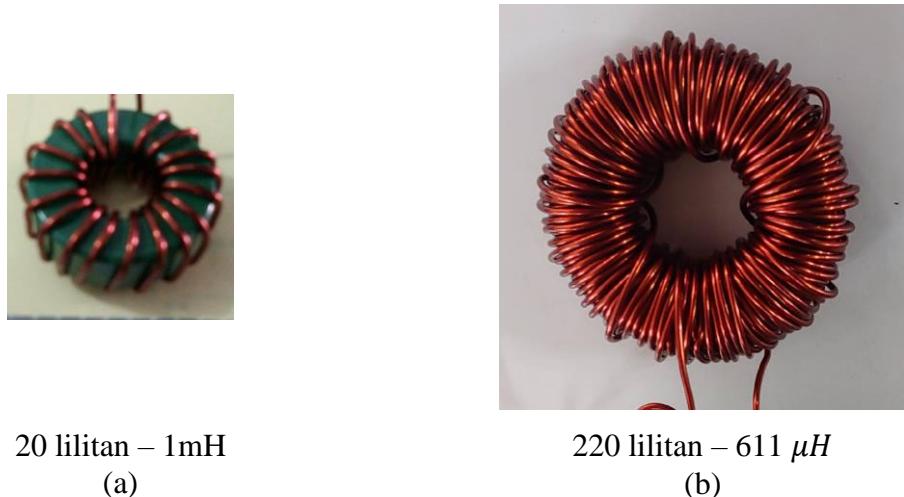
Mengingat bahwa spike tegangan yang terjadi pada MOSFET ternyata tidak terlalu tinggi (maksimum 5V untuk variasi input tegangan tertentu), maka diputuskan bahwa snubber tegangan tidak lagi digunakan, dan hanya akan digunakan snubber arus. Setelah diputuskan bahwa hanya akan digunakan snubber arus, pengujian kemudian berangkat ke tahap selanjutnya, yaitu menggunakan tegangan AC untuk mengamati spike arus yang terjadi untuk input tegangan lebih dari 20VDC.

Percobaan dengan tegangan AC kemudian dilakukan menggunakan trafo tegangan AC yang tersedia di lab konversi, agar peningkatan tegangan AC dapat dilakukan secara perlahan, sambil mengamati spike arus dan suhu dari MOSFET. Seiring dengan ditingkatkannya tegangan AC, output tegangan DC yang teramat juga ikut meningkat sesuai dengan input PWM. Namun, masih terdapat batas tegangan saturasi untuk PWM tertentu. Sebagai contoh, dengan duty cycle sebesar 25%, maksimum tegangan yang dapat dicapai adalah 8VDC, meskipun tegangan input AC nya sudah mencapai 60VDC. Pengamatan suhu dengan *heat gun* yang tersedia di lab konversi menunjukkan bahwa suhu MOSFET meningkat dengan cepat, sampai 50°C dalam waktu kurang dari 30 detik. Perlu diketahui bahwa MOSFET mencapai suhu tersebut saat tegangan output masih 10V, dan arus output hanya sebesar 160mA, sehingga tim penulis memutuskan untuk menghentikan percobaan, dan melakukan *troubleshooting* lebih lanjut.

Setelah melakukan riset secara daring, ditemukan bahwa *power inductor* yang digunakan sebagai induktor utama pada rangkaian *buck converter* memiliki arus saturasi yang sangat kecil, terlihat dari jumlah lilitan yang cukup sedikit untuk mendapatkan induktansi yang besar, yakni 1mH. Jumlah lilitan yang diperlukan dipengaruhi oleh faktor permeabilitas (μ_R) yang dimiliki oleh *core* induktor yang digunakan. Perlu digunakan induktor yang memiliki μ_R yang rendah, sehingga arus saturasi yang dimiliki oleh induktor tersebut tinggi.

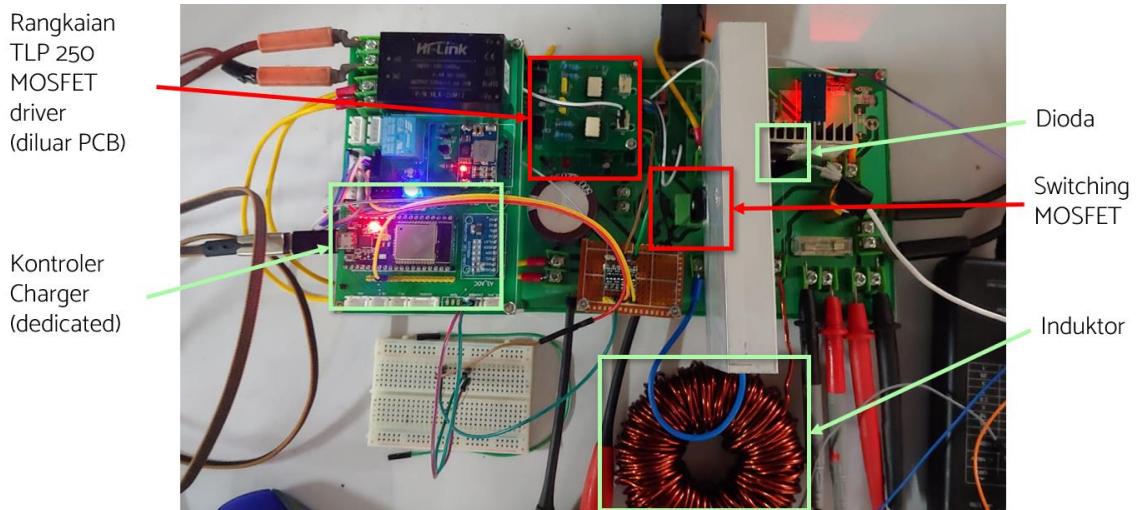
Pada proses desain yang ideal, terdapat datasheet untuk core induktor yang digunakan, sehingga dapat dilihat kurva B-H yang dapat memberikan informasi arus saturasi

maksimum dari suatu induktor. Namun pada pasar Indonesia, sangat sulit untuk menemukan core khusus yang memiliki datasheet, sehingga tim penulis memutuskan untuk memilih core dengan μ_R sekecil-kecilnya agar tidak masuk ke daerah operasi saturasi. Akibatnya, diperlukan jumlah kumparan yang sangat banyak untuk mencapai induktansi 1mH, yakni sebanyak 397 lilitan. Karena keterbatasan waktu dan kawat tembaga, hanya dapat dibuat induktor dengan 220 lilitan, yang menghasilkan induktansi sebesar $611 \mu H$. Induktor awal dan induktor pengganti dapat dilihat pada Gambar 2.1.14



Gambar 2.1.14 (a) induktor dengan μ_R yang tinggi. (b) induktor dengan μ_R yang rendah

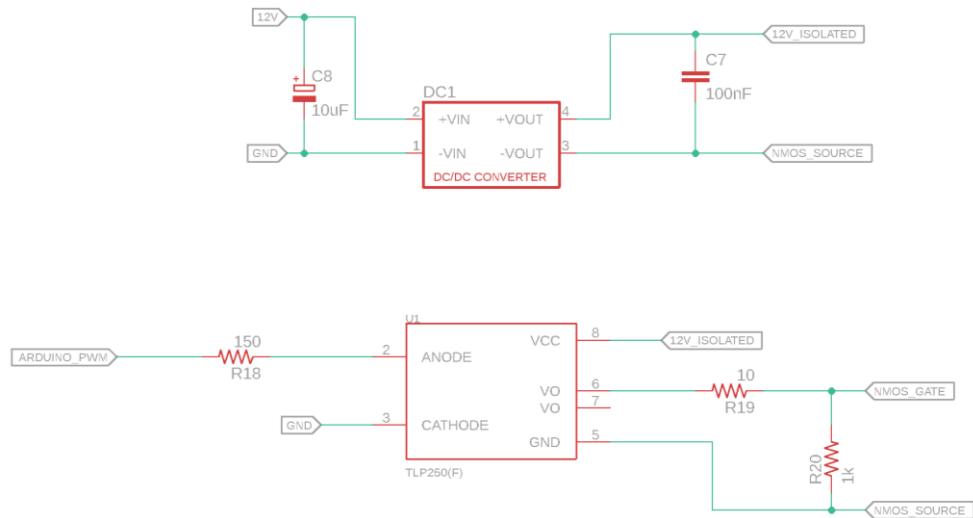
Setelah memasangkan induktor tersebut ke rangkaian, terdapat kendala lain yakni MOSFET *driver* yang sekali lagi mengalami malfungsi. Pada saat ini, diputuskan bahwa operasi MOSFET driver berbasis IR2110 tidak cukup konsisten dalam perancangan subsistem ini, sehingga diputuskan untuk mengganti rangkaian driver menjadi berbasis TLP250. Karena keterbatasan waktu, pembuatan dan pencetakan PCB dengan rangkaian driver TLP250 yang terintegrasi di PCB tidak memungkinkan, sehingga rangkaian driver tersebut berada di luar PCB charger. Selain itu, diputuskan pula untuk menggunakan mikrokontroler *dedicated* (ESP32) untuk meminimalisir terjadinya kesalahan pada kendali *closed loop*. Percobaan singkat dengan komponen yang diganti & ditambahkan menunjukkan hasil yang menjanjikan, dimana arus spike pada induktor sudah berada dalam batas normal, serta dapat mengeluarkan output arus yang tinggi. Dengan ini, proses implementasi subsistem charger telah usai, dan akan dilakukan beberapa pengujian untuk mengkonfirmasi fungsionalitas dari subsistem charger ini. Rangkaian akhir hasil implementasi dapat dilihat pada Gambar 2.1.15



Gambar 2.1.15 Rangkaian hasil implementasi subsistem charger

2.1.1.5 Rangkaian Driver TLP250

Karena chip driver IR2110 tidak lagi digunakan, rangkaian untuk driver tersebut tidak akan ditampilkan. Yang akan ditampilkan adalah implementasi driver dengan chip TLP250. Secara umum, driver ini merupakan optocoupler yang menggunakan LED untuk mengendalikan CMOS. Rangkaian driver ini dapat dilihat pada Gambar 2.1.16



Gambar 2.1.16 Skematik rangakaian driver TLP250

Pada Gambar 2.1.16 dapat dilihat bahwa driver ini cukup sederhana, dimana saat input PWM berada pada logic level ON, maka VO akan sama dengan VCC. Sedangkan saat PWM berada pada logic level OFF, maka VO akan sama dengan GND. Untuk memberi tegangan referensi pada VCC dan GND, digunakan sebuah isolated DC-DC power supply. Daftar komponen yang diperlukan untuk rangkaian driver ini dapat dilihat pada Tabel 2.1.6

Tabel 2.1.6 Daftar komponen untuk rangkaian driver

Komponen	Jumlah
Optocoupler CMOS [TLP250]	1
DC-DC Isolated Power Supply	1
Elco 10uF 25V	1
Kapasitor mika 100nF	1
R 1k	1
R 10	1
R 180	1

2.1.2 Pengujian

Setelah mengganti power inductor dan rangkaian driver pada rangkaian buck converter, subsistem charger sudah dapat mengeluarkan output arus yang tinggi, tanpa merusak MOSFET. Pengujian closed loop kemudian dilakukan untuk beberapa setpoint arus. Hasil pengujian pada beberapa nilai output arus dapat dilihat pada Tabel 2.1.7

Tabel 2.1.7 Pengujian Closed Loop dengan load Resistor $\pm 12\Omega$

Setpoint Arus	V_o (V)	I_o (A)	$I_{D_{MAX}}$ (A)	P_o (W)
1	12.8	0.9	1.6	11.9
2	25.6	2.0	3.7	51.0
3	36.6	2.9	5.3	106.6
4	48.8	3.8	7.2	186.6
5	61.5	4.8	9.4	296.3
6	72.9	6.0	11.8	433.9
7	84.2	6.9	14.0	583.5
7.5	89.1	7.4	15.6	655.8

Salah satu kekurangan dari pengujian menggunakan resistor sebagai load ialah kenaikan tegangan yang sebanding dengan kenaikan arus. Dengan resistor sebesar 12Ω sebagai load, tidak dapat dilakukan pengujian untuk output fast charging (60V, 10A). Namun, pengujian untuk output slow charging (60V, 5A) dapat dilakukan, yang dapat dilihat pada Tabel 2.1.7 berwarna hijau.

Namun setelah berkonsultasi dengan pihak Lab Konversi, terdapat satu resistor load lagi yang bernilai kurang lebih 12Ω pula, sehingga kedua load tersebut dapat diparalel sehingga menghasilkan load 6Ω , dan digunakan untuk menguji output fast charging. Hasil pengujian dengan load 6Ω dapat dilihat pada Tabel 2.1.8, dan karakteristik termal untuk beberapa komponen krusial dapat dilihat pada Tabel 2.1.9

Tabel 2.1.8 Pengujian Closed Loop dengan load Resistor $\pm 6\Omega$

Setpoint Arus	V_o (V)	I_{AVG} (A)	$I_{D_{MAX}}$ (A)	P_o (W)
5	45.4	7	10.3	318.1
6	52.8	8.1	12.0	428.0
7	58.9	9.1	13.4	535.9
10	67.5	10.3	15.2	655.8

Tabel 2.1.9 Informasi termal untuk pengujian pada Tabel 2.1.8

Setpoint Arus	MOSFET	Dioda	Induktor	PCB Rail
5	29	35	34	30
6	30	-	-	31
7	30	41	-	32
10	33	49	41	37

Terlihat pada Tabel 2.1.8 bahwa subsistem charger berhasil mengeluarkan output fast charging, yakni 67.5V dengan arus 10.3A. Saat output subsistem charger nominal tersebut, I_D tidak menunjukkan adanya *spike* yang lebih tinggi dari 1.5 kali dari I_o . Pengamatan waveform I_D dapat dilihat lebih lanjut pada Gambar 2.1.17



Gambar 2.1.17 Waveform V_{GS} (atas) dan I_D (bawah), dengan $I_{AVG} = 10A$

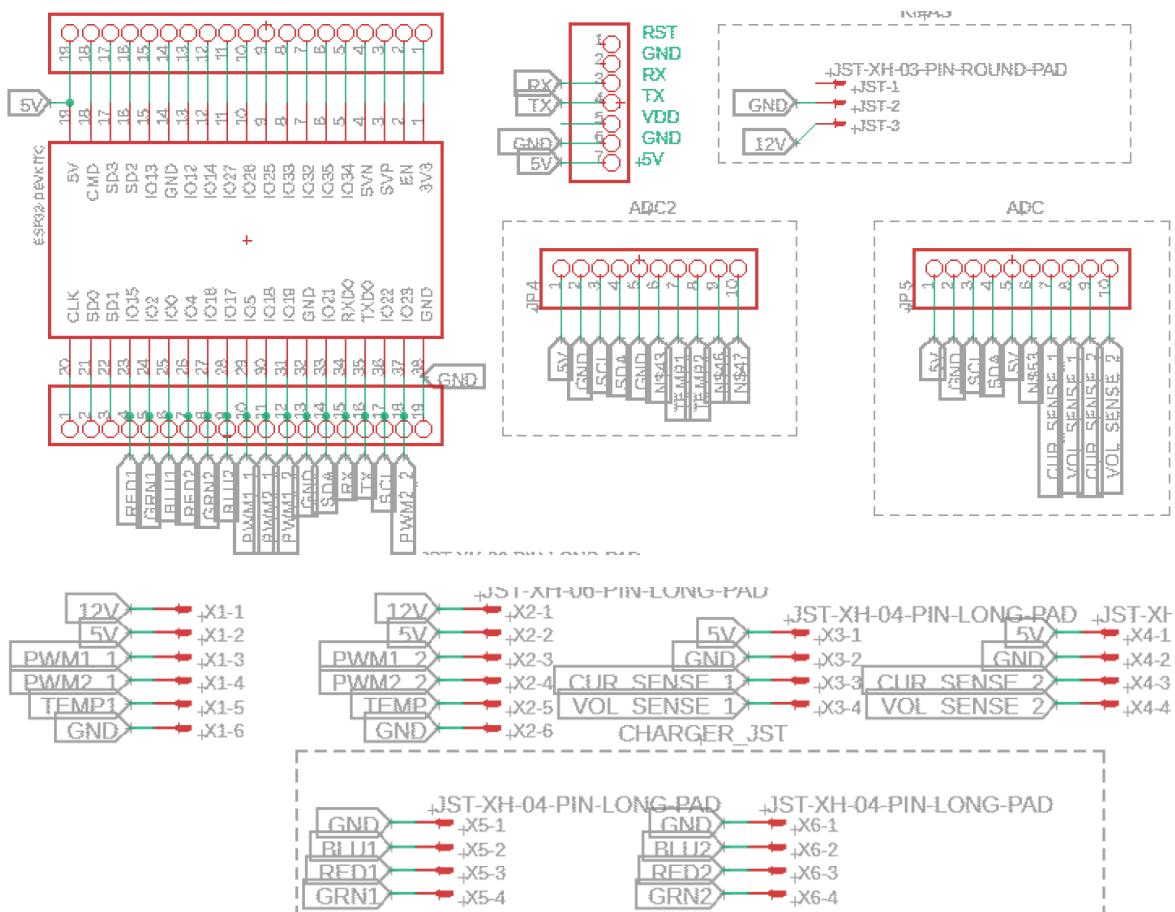
Pada Gambar 2.1.17, terlihat bahwa saat arus output 10A, spike yang terjadi pada I_D hanya mencapai 1.52 kalinya, yakni 15.2A. Namun pada gambar tersebut juga dapat dilihat bahwa waveform V_{GS} yang memiliki spike dan ringing. Karakteristik tersebut tidak dapat dihindari karena rangkaian driver berada diluar PCB, sehingga koneksi antara output rangkaian driver

dengan gate MOSFET dihubungkan menggunakan kabel tipis yang tentunya memiliki induktansi parasitik sehingga terjadi spiking dan ringing. Meskipun itu, tegangan spike masih berada di batas normal dan masih dapat ditolerir. Data termal yang terdapat pada Tabel 2.1.9 juga menunjukkan nilai yang stabil. Heatsink yang digunakan pada dioda dapat diperbesar, namun untuk penggunaan sekarang, heatsink tersebut masih mencukupi.

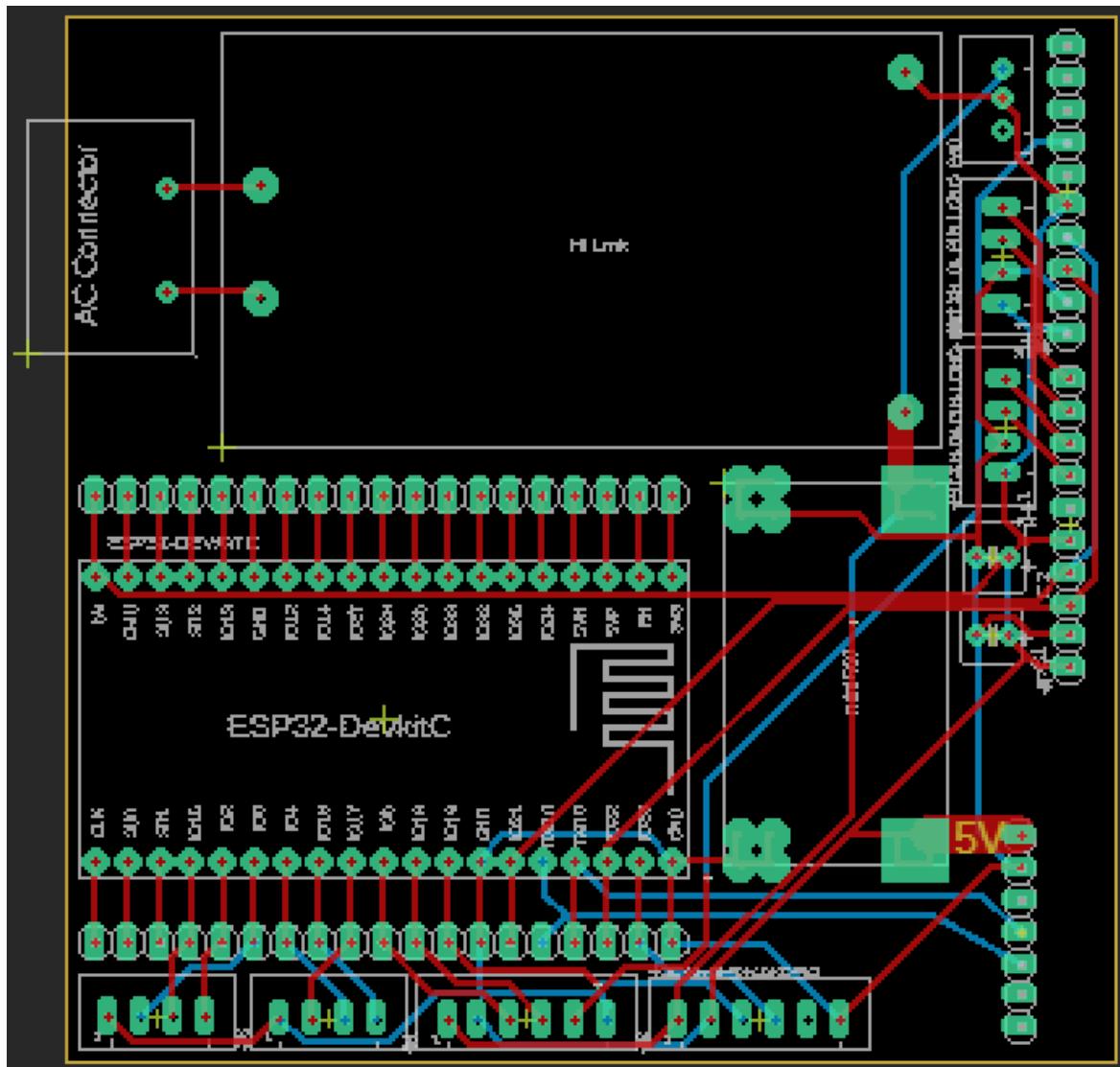
2.2 Sub-Sistem Kontrol

2.2.1 Implementasi

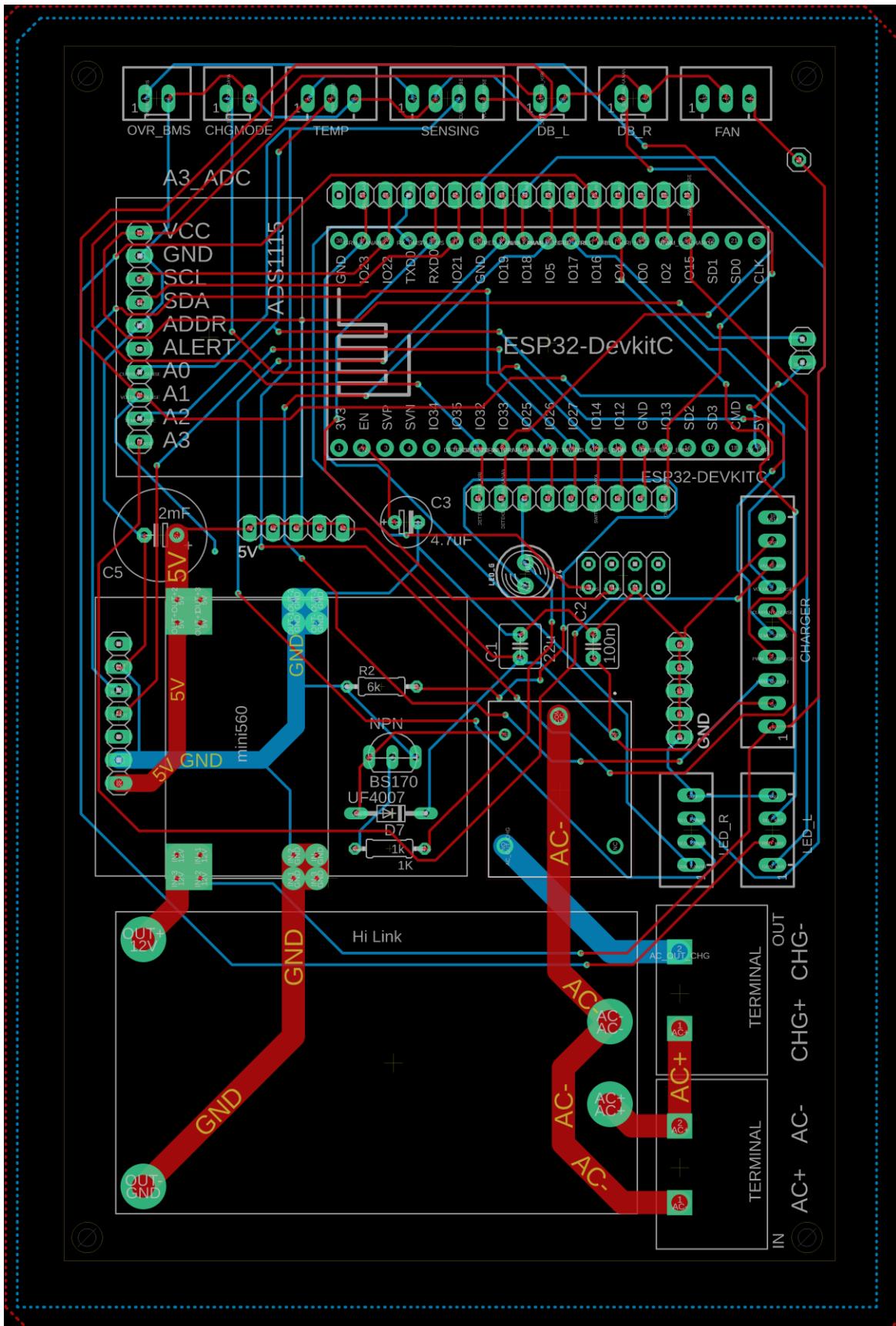
2.2.1.1 Perangkat Keras



Gambar 2.2.1 Skematik Subsistem Kontrol



Gambar 2.2.2 Layout Subsistem Kontrol



Gambar 2.2.3 Layout Subsistem Kontrol revisi 2

Pada board kontrol revisi kedua, sudah dilengkapi dengan relay untuk memutus arus ke charger. Selain itu, pada revisi ini juga ditambahkan beberapa konektor JST-XH untuk koneksi button-button, LED, serta sensor suhu. Selain itu juga dilakukan penambahan konektor TTL untuk komunikasi serial dengan *tester board*. Meninjau board yang telah dirancang pada Gambar 2.2.3, daftar komponen yang diperlukan untuk board kontrol ini dapat dilihat pada Tabel 2.2.1.

Tabel 2.2.1 Daftar komponen untuk board Charger

Kategori	Komponen	Jumlah
Power	Hilink 220VAC - 12VDC	1
	Mini 560	1
Modul	ESP32	1
	ADS1115	1
	SIM800L	1
Relay	Songle Relay 250VAC	1
	BS170	1
	Diода	1
	R 1k	1
	R 6k	1
	LED	1
Kapasitor Smoothing	C 22uF	1
	C 100nF	1
Kapasitor ESP	C 10uF	1
Lain lain	JST XH4	3
	JST XH6	2
	JST XH3	2
	JST XH10	1
	JST XH2	4
	Female Header 1x20	2

	Female Header 1x10	2
	Female Header 1x7	1
	Terminal block	2

2.2.1.2 Source Code

Kode yang digunakan pada subsistem kontrol dibuat menggunakan Real Time OS atau biasa disingkat FreeRTOS. FreeRTOS digunakan karena berukuran kecil dan memiliki cara kerja yang lebih advance dibandingkan dengan super loop. Dengan menggunakan FreeRTOS, keseluruhan sistem dapat dibagi-bagi menjadi task-task sederhana sesuai dengan fungsinya, sehingga kode akan lebih mudah untuk di debug jika terdapat kesalahan.

Kode yang digunakan dalam alat ini dibagi menjadi 4 task. Pengaturan perpindahan state dan output tiap state dari baterai kiri dan kanan dibagi menjadi dua task berbeda. Kemudian, terdapat satu task yang berfungsi untuk melakukan sampling arus dan suhu setiap saat. Terakhir, terdapat task untuk interkoneksi dengan web. Berikut ini akan dijelaskan lebih rinci penerapan task untuk perpindahan state. Untuk task sampling, akan dijelaskan secara detail di bagian subsistem sensing dan task interkoneksi dengan web akan dijelaskan secara detail pada bagian subsistem IoT. Sebelum masuk ke penjelasan task, terdapat pendeklarasian pin-pin yang akan digunakan. Pendeklarasian pin yang digunakan yaitu sebagai berikut. Tabel 2.2.2 menunjukkan alokasi pin untuk setiap komponen pada setiap subsistem. Implementasi source code mengikuti pin mapping yang ada pada Tabel 2.2.2.

Tabel 2.2.2 Pin Mapping ESP32

Subsistem	Komponen	GPIO
Interface	LED Right (R, G, B)	19, 18, 23
	LED Left (R, G, B)	16, 4, 17
	Override BMS	13
	Switch Mode Daya	14
	Deteksi Baterai Right	33
	Deteksi Baterai Left	32
Tester Board	TX BMS	1
	RX BMS	3
Charger	PWM MOSFET	-
	Switch Discharge	-
	Relay	25

Sensing (ADS 1015)	SDA	21
	SCL	22
	ADDR	GND
IoT (SIM800L)	TX GSM	26
	RX GSM	27

Di bawah ini merupakan definisi dari *pin mapping* dan juga definisi state dari sistem keseluruhan.

```
// State Definition
#define STARTUP 9
#define IDLE 0
#define RETRIEVE_SERIAL 1
#define TRIGGER 2
#define INIT_CHARGING 3
#define CHARGING 5
#define FINISH_CHARGING 4
```

```
//Right Side
#define batteryButtonR 33
#define chargerSW1R 32
#define chargerSW2R 33
#define redR 19
#define greenR 18
#define blueR 23
```

```
//Left Side
#define batteryButtonL 32
#define chargerSW1L 23
#define chargerSW2L 25
#define redL 16
#define greenL 4
#define blueL 17
```

```
//Others
#define chargingModePin 14
#define SDA 21
#define SCL 22
#define txBMS 1
#define rxBMS 3
#define txGSM 26
#define rxGSM 27
#define relayCharger 25
#define triggerButton 13
```

Setelah pendeklarasian pin yang akan digunakan, selanjutnya mendeklarasikan global variable. Global variable merupakan variable yang akan digunakan untuk beberapa task. Pendeklarasian global variable dibagi menjadi 2 untuk menyimpan nilai yang dibutuhkan oleh pengisian baterai kiri dan kanan.

```
// Global Variables
// Right Side
byte stateR = 0;
int PWM_FREQUENCYR = 50000;
int PWM_CHANNELR = 0;
int PWM_RESOLUTIONR = 8;
unsigned short SoHR;
unsigned short remainingCapacityR;
unsigned short totalCapacityR;
double SoCR;
double voltR;
char msgR[200];
char serialNumberR[16] = "";
byte cmd_sendSlotR = 0;

// Left Side
byte stateL = 0;
int PWM_FREQUENCYL = 50000;
int PWM_CHANNELL = 0;
int PWM_RESOLUTIONL = 8;
unsigned short SoHL;
unsigned short remainingCapacityL;
unsigned short totalCapacityL;
double SoCL;
double voltL;
char msgL[200];
char msgBattInfoL[200];
char serialNumberL[16] = "";
byte cmd_sendSlotL = 0;
```

Pada kode arduino, terdapat void setup() untuk melakukan inisialisasi penggunaan pin apakah sebagai input maupun output. Pada bagian setup() ini juga digunakan untuk mendeklarasikan komunikasi serial yang digunakan dan pendeklarasian task yang akan digunakan pada FreeRTOS. Kode void setup() yang digunakan sebagai berikut.

```
void setup()
{
    // Right Side Pin Configuration
    pinMode(batteryButtonR, INPUT_PULLUP);
    //pinMode(chargerSW1R, OUTPUT);
    //pinMode(chargerSW2R, OUTPUT);
    pinMode(redR, OUTPUT);
    pinMode(greenR, OUTPUT);
    pinMode(blueR, OUTPUT);

    // Left Side Pin Configuration
    pinMode(batteryButtonL, INPUT_PULLUP);
    //pinMode(chargerSW1L, OUTPUT);
    //pinMode(chargerSW2L, OUTPUT);
    pinMode(redL, OUTPUT);
    pinMode(greenL, OUTPUT);
    pinMode(blueL, OUTPUT);
```

```

// Other pins configuration
pinMode(triggerButton, INPUT_PULLUP);
pinMode(chargingModePin, INPUT_PULLUP);
pinMode(relayCharger, OUTPUT);

// Initialize serial communication
Serial.begin(9600);

//ADS1115 initialization
if (!adsR.begin(0x48)) {
    Serial.println("Failed to initialize ADS.");
    while (1);
}

if (!adsL.begin(0x49)) {
    Serial.println("Failed to initialize ADS.");
    while (1);
}

xTaskCreatePinnedToCore(LeftCode, "Left", 10000, NULL, 1, NULL, 0);
xTaskCreatePinnedToCore(Sampling, "Sampling", 10000, NULL, 1, NULL, 0);
xTaskCreatePinnedToCore(RightCode, "Right", 10000, NULL, 1, NULL, 1);
xTaskCreatePinnedToCore(MQTTLoop, "MQTTLoop", 10000, NULL, 2, NULL, 1);

}

```

Bagian pertama yang akan dijelaskan yaitu untuk menentukan state pada subsistem charger. Terdapat 6 state pada masing-masing subsistem charger seperti yang ada pada Gambar 2.2.4 dan telah dijelaskan pada B300. Terdapat masing-masing satu task untuk penentuan state baterai kiri dan kanan. Kode yang digunakan untuk memanggil fungsi pada state tersebut yaitu sebagai berikut.

<pre> void LeftCode(void *parameter) { for (;;) { if(overheatFlag overcurrentFlag !isEnabledCmd){ stateFaultL(); } else if(triggerFlag){ stateTriggerBMS(); } else{ if (stateL == IDLE) { stateIdleL(); idleTransitionL(); } else if (stateL == RETRIEVE_SERIAL) { stateRetrieveSerialL(); } } } } </pre>	<pre> void RightCode(void *parameter) { for (;;) { if(overheatFlag overcurrentFlag !isEnabledCmd){ stateFaultR(); } else if(triggerFlag){ // Tidak melakukan apapun } else{ if (stateR == IDLE) { stateIdleR(); idleTransitionR(); } else if (stateR == RETRIEVE_SERIAL) { stateRetrieveSerialR(); } } } } </pre>
--	--

```

        else if (stateL == INIT_CHARGING)
        {
            initChargingL();
        }
        else if (stateL == FINISH_CHARGING)
        {
            stateFinishChargeL();
        }
        else if (stateL == CHARGING)
        {
            stateChargingL();
        }

        vTaskDelay(3 / portTICK_PERIOD_MS);
    }
}

```

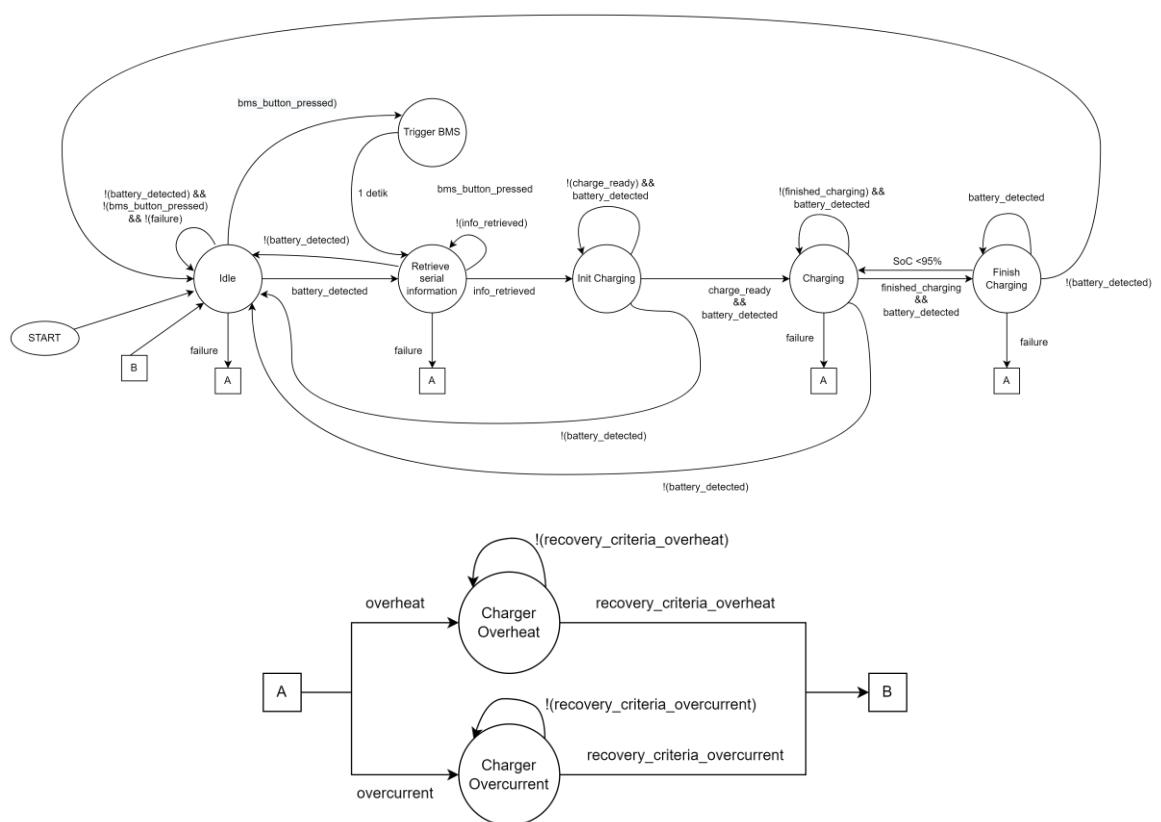


```

        }
        else if (stateR == INIT_CHARGING)
        {
            initChargingR();
        }
        else if (stateR == FINISH_CHARGING)
        {
            stateFinishChargeR();
        }
        else if (stateR == CHARGING)
        {
            stateChargingR();
        }

        vTaskDelay(3 / portTICK_PERIOD_MS);
    }
}

```



Gambar 2.2.4 FSM sistem keseluruhan

Sistem pertama-tama akan masuk ke state Idle. Pada state ini, sistem akan menunggu baterai untuk dipasangkan. Selama menunggu, sistem akan memberikan indikator LED warna merah.

```
void stateIdleL() {
    // Membuka transistor SW2
    digitalWrite(chargeSW2L, HIGH);

    // Memberika indikator LED merah
    LED_1L(HIGH, LOW, LOW);

    // Delay
    vTaskDelay(50/portTICK_PERIOD_MS);
}

void idleTransitionL()
{
    byte triggerDetectedL; // Untuk mendekripsi
trigger button ditekan
    byte battSWL = digitalRead(batteryButtonL); // Untuk mendekripsi
switch pada dudukan baterai ditekan
    int16_t adc;
    float vBattery = 0;

    triggerDetectedL = digitalRead(triggerButton);

    // Apabila terdeteksi ada baterai, pindah ke state retrieve
serial. Selain itu, pindah ke state 2
    if (voltage > 40000)
    {
        if(battSWL == 1 && stateR != RETRIEVE_SERIAL && stateR != INIT_CHARGING){
            stateL = RETRIEVE_SERIAL;
        }
    }
    else
    {
        stateL = IDLE;
    }
}
```

Terdapat fungsi bernama LED_1L yang banyak digunakan pada state-state lainnya. Fungsi ini berfungsi untuk mengatur keluaran warna merah, hijau, dan biru dari LED. Karakter L di akhir mengindikasikan bahwa fungsi tersebut ditujukan untuk LED pada baterai sebelah kiri. Adapun untuk baterai sebelah kanan dinamakan LED_1R dan berisi kode dengan alur yang sama.

```
void LED_1L(int red_light_value_1, int green_light_value_1, int
blue_light_value_1){
    digitalWrite(redL, red_light_value_1);
    digitalWrite(greenL, green_light_value_1);
    digitalWrite(blueL, blue_light_value_1);
}
```

State selanjutnya adalah membaca informasi serial. Pada state ini, ESP32 akan mengirimkan *request* ke BMS baterai untuk mendapatkan informasi nomor serial baterai dan informasi

baterai seperti SoC dan SoH. *Request* dikirimkan dengan komunikasi UART dengan *baud rate* 9600. Perintah yang dituliskan untuk meminta nomor serial baterai adalah “LSerial_Number” untuk port sebelah kiri dan “RSerial_Number” untuk port sebelah kanan. Format respon dari BMS adalah “[L atau R]+[Serial_number]+[Serial_number]”. Pengecekan pesan respon dapat dilakukan dengan mengecek apakah terdapat dua karakter “+” dari pesan yang diterima. Apabila iya, nomor serial dapat dibaca. Untuk *request* informasi baterai, digunakan perintah ”[L/R]Batt_Info”.

```

void stateRetrieveSerialL(){
    LED_1L(HIGH, HIGH, LOW);
    int serialValid = 0;
    int battDataValid = 0;
    int counter = 0;
    byte battSWL = digitalRead(batteryButtonL);

    while(!(serialValid) && battSWL){
        serialValid = readSerial('L');
        battSWL = digitalRead(batteryButtonL);
        vTaskDelay(700 / portTICK_PERIOD_MS);
    }

    // Sampai sini, data serial sudah diperoleh

    while(!(battDataValid) && counter < MAX_RETRIEVE && battSWL){
        battDataValid = readBattData('L');
        battSWL = digitalRead(batteryButtonL);
        counter++;
        vTaskDelay(1500 / portTICK_PERIOD_MS);
    }

    // Sampai sini, data baterai mungkin sudah diperoleh atau belum
    // Diberikan counter agar tidak stuck ketika tidak dapat membaca
    info baterai

    if (battSWL == 1 && voltage > 40000)
    {
        stateL = INIT_CHARGING;
    }
    else
    {
        stateL = IDLE;
    }
}

int readSerial(char LorR){
    int readIdx = 0;
    int msgIdx = 0;
    int plusCount = 0;
    int serialNumberCount = 0;
    int serialNumberIdx = 0;

    // Mengirim request serial number. request dikirimkan dua kali
    // karena ada delay respon dari BMS sebesar satu request
    if(LorR == 'L'){
        Serial.write("LSerial_Number");
    }
}

```

```

else if(LorR == 'R'){
    Serial.write("RSerial_Number");
}
else{
    Serial.println("LorR tidak didefinisikan!");
}

// Membaca respon dari request
while (Serial.available() > 0)
{
    if(LorR == 'L'){
        msgL[readIdx] = Serial.read();
    }
    else if(LorR == 'R'){
        msgR[readIdx] = Serial.read();
    }
    else{
        Serial.println("LorR tidak didefinisikan!");
    }
    readIdx++;
}

// Parsing pesan
if( LorR == 'L' ){
    if(msgL[0] == 'L'){
        // Menghitung karakter '+'
        while (msgIdx < readIdx + 1)
        {
            if (plusCount == 1)
            {
                serialNumberCount++;
            }

            if (msgL[msgIdx] == '+')
            {
                plusCount++;
                if (plusCount == 1)
                {
                    serialNumberIdx = msgIdx + 1;
                }
            }
            msgIdx++;
        }

        msgIdx = 0;
    }

    // Memindahkan nomor serial baterai ke variabel
    serialNumberL
    while (msgL[serialNumberIdx] != '+' && serialNumberIdx < readIdx + 1)
    {
        serialNumberL[msgIdx] = msgL[serialNumberIdx];
        serialNumberIdx++;
        msgIdx++;
    }

    // Terminasi string
    serialNumberL[msgIdx] = '\0';
}

```

```

        // Untuk troubleshooting
        Serial.println();
        Serial.print("Serial Number Slot Kiri: ");
        Serial.println(serialNumberL);
    }
    else{
        Serial.println("Data yang diterima tidak valid");
    }
}
else if( LorR == 'R' ){
    if(msgR[0] == 'R'){
        // Menghitung karakter '+'
        while (msgIdx < readIdx + 1)
        {
            if (plusCount == 1)
            {
                serialNumberCount++;
            }

            if (msgR[msgIdx] == '+')
            {
                plusCount++;
                if (plusCount == 1)
                {
                    serialNumberIdx = msgIdx + 1;
                }
            }

            msgIdx++;
        }

        msgIdx = 0;
    }

    // Memindahkan nomor serial baterai ke variabel
    serialNumberL
    while (msgR[serialNumberIdx] != '+' && serialNumberIdx <
readIdx + 1)
    {
        serialNumberR[msgIdx] = msgR[serialNumberIdx];
        serialNumberIdx++;
        msgIdx++;
    }

    // Terminasi string
    serialNumberR[msgIdx] = '\0';

    // Untuk troubleshooting
    Serial.println();
    Serial.print("Serial Number Slot Kanan: ");
    Serial.println(serialNumberR);
}
else{
    Serial.println("Data yang diterima tidak valid");
}
}

if(plusCount == 2 && serialNumberCount > 1){
    return 1;
}
else{

```

```

        return 0;
    }

int readBattData(char LorR){
    int readIdx = 0;
    int msgIdx = 0;
    int plusCount = 0;
    int serialNumberCount = 0;
    int checksum = 0;

    // Mengirimkan request
    if(LorR == 'L'){
        Serial.write("LBatt_Info");
    }
    else if(LorR == 'R'){
        Serial.write("RBatt_Info");
    }
    else{
        Serial.println("LorR tidak terdefinisi");
    }

    // Membaca respon dari request
    while (Serial.available() > 0)
    {
        if(LorR == 'L'){
            msgL[readIdx] = Serial.read();
        }
        else if(LorR == 'R'){
            msgR[readIdx] = Serial.read();
        }
        readIdx++;
    }

    while(msgIdx < readIdx+1){
        Serial.print(msgBattInfoL[msgIdx]);
        Serial.print(" , ");
        Serial.println(msgBattInfoL[msgIdx], HEX);

        msgIdx++;
    }

    for(int i = 1; i < readIdx-3; i++){
        checksum += msgL[i];
    }

    // Parsing respon dari BMS
    if(LorR == 'L'){
        if( strncmp(msgL, "Error", 6) == 0 ){
            Serial.println("Data battery info tidak valid!");
            SoHL = -1;
            remainingCapacityL = -1;
            totalCapacityL = -1;
            SoCL = -1;
        }
        else{
            if(checksum % 256 == 0){
                SoHL = (msgL[80] << 8) | (msgL[81]);
                remainingCapacityL = (msgL[76] << 8) | (msgL[77]);
                totalCapacityL = (msgL[78] << 8) | (msgL[79]);
            }
        }
    }
}

```

```

        SoCL = (double) remainingCapacityL / (double)
totalCapacityL;
    }
    else{
        Serial.println("Data salah");
    }
}

Serial.println(SoHL);
Serial.println(remainingCapacityL);
Serial.println(totalCapacityL);
Serial.println(SoCL);
}
else if(LorR == 'R'){
    if( strncmp(msgR, "Rerror", 6) == 0 ){
        Serial.println("Data battery info tidak valid!");
        SoHL = -1;
        remainingCapacityL = -1;
        totalCapacityL = -1;
        SoCL = -1;
    }
    else{
        if(checksum % 256 == 0){
            SoHR = (msgR[80] << 8) | (msgR[81]);
            remainingCapacityR = (msgR[76] << 8) | (msgR[77]);
            totalCapacityR = (msgR[78] << 8) | (msgR[79]);
            SoCR = (double) remainingCapacityR / (double)
totalCapacityR;
        }
        else{
            Serial.println("Data salah");
        }
    }
}

Serial.println(SoHR);
Serial.println(remainingCapacityR);
Serial.println(totalCapacityR);
Serial.println(SoCR);
}

if(checksum % 256 == 0){
    return 1;
}
else{
    return 0;
}
}

```

Setelah informasi serial baterai telah didapat, ESP32 akan mengirim *request* ke BMS untuk mengubah mode baterai menjadi mode *charging*. Format respon dari BMS adalah “[L atau R][ok/error]”. Apabila pesan yang diterima adalah “Rok” atau “Lok”, maka baterai siap untuk di-*charging*.

```

void initChargingL(){
    byte battSW = digitalRead(batteryButtonL);
    int chargeReady = 0;
    char stringHolder[256];
    char blank[1] = "";

```

```

// Mengirimkan request mengubah mode charging
while(!(chargeReady) && battSW){
    // Indikator blink biru
    LED_1L(LOW, LOW, LOW);
    vTaskDelay(500 / portTICK_PERIOD_MS);
    LED_1L(LOW, LOW, HIGH);
    vTaskDelay(500 / portTICK_PERIOD_MS);
    chargeReady = requestStartCharging('L');
    battSW = digitalRead(batteryButtonL);
    vTaskDelay(1000 / portTICK_PERIOD_MS);
}

// Transisi State
if (battSW == 1 && voltage > 40000)
{
    if (chargeReady)
    {
        stateL = CHARGING;
        cmd_sendSlotL = 1;
    }
}
else
{
    stateL = IDLE;
}

```

State selanjutnya adalah *charging*. Pada state ini, akan dilakukan pengisian daya baterai. Tegangan dan arus dari subsistem charger dikontrol dengan kontrol proporsional.

```

void stateChargingL()
{
    // Serial.print("L Charging\n");
    byte battSWL = digitalRead(batteryButtonL);
    float tegangan;
    float arus;

    LED_1L(LOW, LOW, HIGH);

    // Mengaktifkan relay
    digitalWrite(relayCharger, RELAY_ON);

    vTaskDelay(delayChargingState / portTICK_PERIOD_MS); // delay
charging

    // Transisi state
    if (battSWL == 1 && voltage > 40000)
    {
        // Kode untuk demo alat
        if (DEMO == 1){
            if(stateR != CHARGING){
                digitalWrite(relayCharger, RELAY_OFF);
            }
            stateL = FINISH_CHARGING;
            cmd_sendSlotL = 1;
        }
    }
}

```

```

// Akan selesai bila arus yang ke baterai bernilai 0
else {
    if(current < 0){
        if(stateR != CHARGING){
            digitalWrite(relayCharger, RELAY_OFF);
        }
        stateL = FINISH_CHARGING;
        cmd_sendSlotL = 1;
    }
    else{
        stateL = CHARGING;
    }
}
}

// Saat baterai dicabut
else
{
    stateL = IDLE;
    digitalWrite(relayCharger, RELAY_OFF);
    cmd_sendSlotL = 1;
}
}

```

Setelah pengisian daya selesai dilakukan, state selanjutnya adalah finish charging. Pada state ini, sistem akan terus mengecek tegangan baterai. Apabila tegangan baterai telah turun mencapai batas tertentu, akan dilakukan pengisian daya kembali.

```

void stateFinishChargeL()
{
    // Serial.print("L Finish Charging\n");
    byte battSWL = digitalRead(batteryButtonL);

    LED_1L(LOW, HIGH, LOW);

    // Transisi state
    if (battSWL == 1 && voltage > 40000)
    {
        stateL = FINISH_CHARGING;
    }
    else
    {
        stateL = IDLE;
        cmd_sendSlotL = 1;
    }
    vTaskDelay(50 / portTICK_PERIOD_MS);
}

```

Apabila terdapat kasus dimana kapasitas daya yang tersisa di baterai sudah 0%, maka tegangan baterai tidak akan dapat terdeteksi oleh sistem. Untuk menyalakannya kembali, state trigger BMS diperlukan. State ini berisi

```

void stateTriggerBMS()
{
    // Serial.print("L Triggered\n");
}

```

```

LED_1L(HIGH, HIGH, LOW);
LED_1R(HIGH, HIGH, LOW);
vTaskDelay(500 / portTICK_PERIOD_MS);
LED_1L(LOW, LOW, LOW);
LED_1R(LOW, LOW, LOW);
vTaskDelay(500 / portTICK_PERIOD_MS);
LED_1L(HIGH, HIGH, LOW);
LED_1R(HIGH, HIGH, LOW);
vTaskDelay(500 / portTICK_PERIOD_MS);
LED_1L(LOW, LOW, LOW);
LED_1R(LOW, LOW, LOW);
vTaskDelay(500 / portTICK_PERIOD_MS);
LED_1L(HIGH, HIGH, LOW);
LED_1R(HIGH, HIGH, LOW);

digitalWrite(relayCharger, RELAY_ON);

vTaskDelay(1000 / portTICK_PERIOD_MS);

digitalWrite(relayCharger, RELAY_OFF);

stateL = IDLE;
stateR = IDLE;
triggerFlag = 0;
}

```

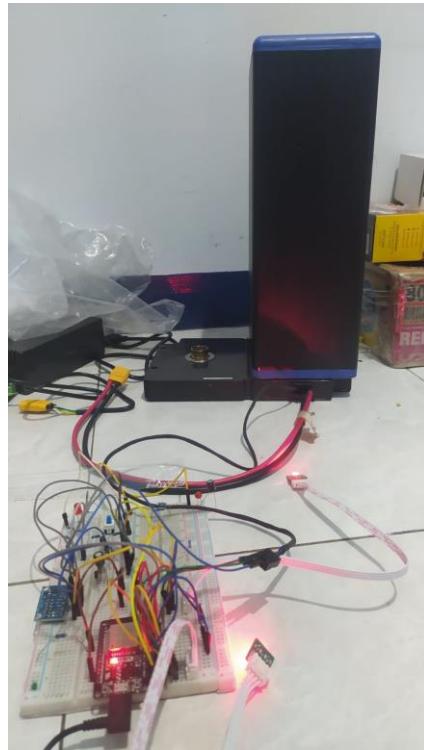
2.2.2 Pengujian

Pengujian subsistem kontrol dilakukan dengan menguji keberhasilan setiap *state*. Tabel 2.2.3 menunjukkan hasil pengujian subsistem kontrol. Parameter keberhasilan *state idle* adalah ketika sistem mengeluarkan LED merah dan akan berubah menjadi kuning ketika dihubungkan baterai pada terminal output. Kemudian, parameter keberhasilan *state retrieve serial number* adalah ketika sistem berhasil mendapatkan informasi serial baterai dan ditandai dengan nyala LED kuning. Lalu, parameter keberhasilan *state trigger BMS* adalah sistem mengeluarkan tegangan nominal baterai selama 1 detik dan ditandai dengan nyala LED kuning berkedip. Setelah itu, parameter keberhasilan *state init charging* adalah ketika sistem mampu mengubah mode BMS yang ditandai dengan LED biru berkedip. Kemudian, parameter keberhasilan *state charging* adalah ketika sistem mampu mengisi daya baterai yang ditandai dengan adanya arus yang masuk dan LED berwarna biru. Selanjutnya, parameter keberhasilan *state finish charging* adalah ketika baterai telah selesai diisi dayanya dan ditandai dengan LED berwarna hijau. Untuk state *failure overheat* dan *overcurrent* serta *disable*, parameter keberhasilannya ditandai dengan sistem tidak dapat beroperasi sampai *failure* teratasi dan ditandai dengan LED merah berkedip.

Tabel 2.2.3 Pengujian subsistem kontrol

State	Status
Idle	Berhasil
Retrieve Serial Number	Berhasil
Trigger BMS	Berhasil
Init Charging	Berhasil
Charging	Berhasil

Finish charging	Berhasil
Charger overheat	Berhasil
Charger overcurrent	Berhasil
Charger disable	Berhasil



Gambar 2.2.5 Dokumentasi pengujian subsistem kontrol

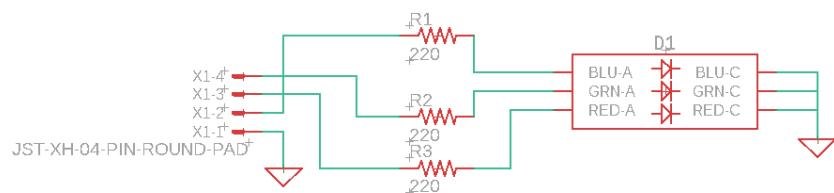
2.3 Sub-Sistem Interface

2.3.1 Implementasi

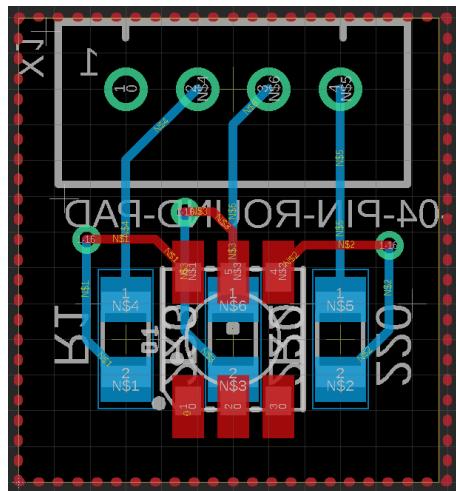
2.3.1.1 Desain Perangkat Keras Subblok Input Pengguna

Pada keseluruhan sistem, dibutuhkan 4 tombol untuk menangani input fisik maupun untuk mendeteksi keberadaan baterai secara fisik. Keempat tombol tersebut yaitu tombol switch mode daya, tombol deteksi baterai kiri dan kanan, dan tombol trigger baterai. Konfigurasi tombol yang digunakan yaitu pullup.

2.3.1.2 Desain Perangkat Keras Subblok Output Pengguna



Gambar 2.3.1 Skematik Indikator LED



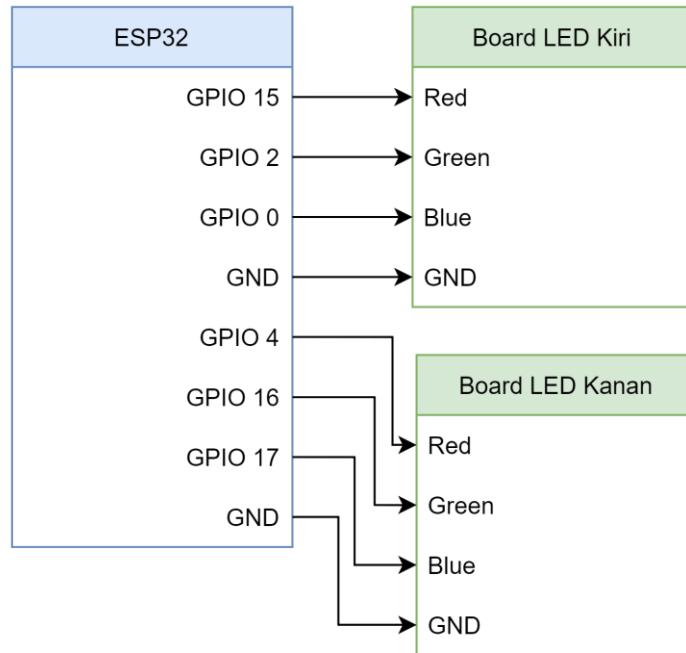
Gambar 2.3.2 Perancangan board untuk indikator LED

Dalam sebuah produk diperlukan sebuah indikator agar pengguna dapat memantau alatnya. Indikator yang ada haruslah sederhana agar mudah dipahami namun juga harus tetap informatif. Pada sistem charger ini, indikator yang digunakan yaitu lampu LED RGB. LED RGB dipilih karena dapat memberikan informasi state secara jelas berdasarkan warnanya.

Meninjau board yang telah dirancang pada Gambar 2.3.2, daftar komponen yang diperlukan untuk board charger ini dapat dilihat pada Tabel 2.3.1

Tabel 2.3.1 Daftar komponen untuk board interface LED

Kategori	Komponen	Jumlah
LED	LED SMD 5050	1
Current Limiting Resistor	R 1206 68Ω	1
	R 1206 15Ω	2
	JST XH4	1



Gambar 2.3.3 Pinout ESP32 yang terhubung ke board indikator LED

2.3.2 Pengujian

Pengujian interface dilakukan dengan menguji nyala setiap nyala LED untuk output dan kontinuitas dari saklar untuk pengujian input. Pengujian nyala LED dan kontinutas saklar dilakukan dengan menggunakan multimeter.

Tabel 2.3.2 Hasil pengujian subsistem interface

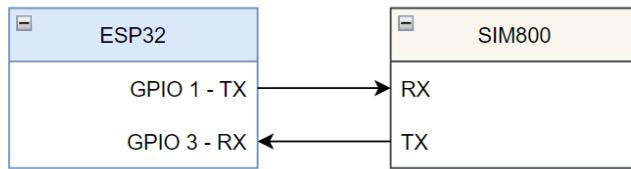
Komponen	Status
LED kiri	LED merah, hijau, dan biru dapat menyala
LED kanan	LED merah, hijau, dan biru dapat menyala
Tombol-tombol	Semua tombol telah teruji kontinuitasnya

2.4 Sub-Sistem IoT

2.4.1 Implementasi

2.4.1.1 Perangkat Keras

Perangkat keras yang digunakan pada subsistem ini hanya modul SIM800L untuk mengirim data ke server. Protokol yang dipilih untuk mengirim data adalah MQTT, menyesuaikan dengan standar yang digunakan oleh Oyika. Maka, diperlukan juga komputer yang menjalankan broker MQTT. Selain itu, penampilan data akan dilakukan di web. Software yang digunakan sebagai broker adalah Mosquitto [1], sedangkan, backend dari web akan dirancang menggunakan Python Flask [2].



Gambar 2.4.1 Pinout ESP32 yang terhubung ke modul GSM SIM800L

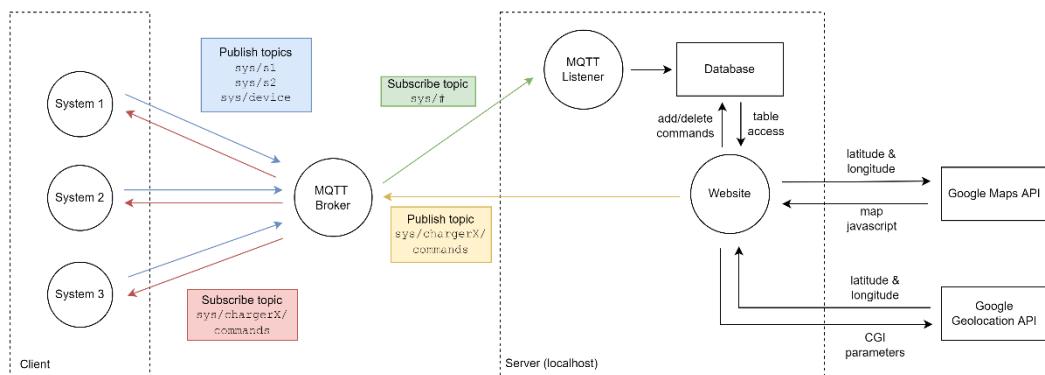
Saat pertama kali melakukan koneksi ke jaringan, modul SIM800L memerlukan arus yang cukup besar dalam waktu yang singkat (~577uS) [8]. Bila demand arus ini tidak dapat terpenuhi tegangan catu daya akan turun, sehingga modul akan melakukan restart dan mencoba melakukan koneksi lagi. Berdasarkan pengujian yang dilakukan, modul Mini 560 pada subsistem power supply tidak dapat merespon lonjakan arus yang diperlukan dengan waktu yang cepat, sehingga pada node 5V perlu diletakkan kapasitor (4400uF) untuk memenuhi demand arus dari modul SIM800L.

Proses publish topic yang dilakukan oleh produk dilakukan menggunakan ESP32, bersamaan dengan code kontrol charging, interface, dsb. Melalui koneksi GPRS, subsistem IoT mem-publish topic ke broker MQTT yang dihost di GCE (Google Compute Engine), kemudian server men-subscribe ke topic tersebut dan mengolah data yang masuk. Proses ini lebih jelasnya dapat dilihat pada Subbab 2.4.1.2.

2.4.1.2 Perangkat Lunak

2.4.1.2.1 Struktur Data

Secara umum, alur data sistem ini dapat digambarkan sebagai berikut



Gambar 2.4.2 Alur data subsistem IoT

Protokol MQTT memiliki konsep publish dan subscribe, dimana publish merupakan pengiriman data ke broker, dan subscribe merupakan penerimaan data dari broker. Broker yang digunakan adalah Mosquitto. Broker ini dijalankan pada Google Compute Engine Virtual Machine Instance (GCE VM) agar data dapat dipublish melalui jaringan GPRS. Kemudian, datalogger yang dijalankan di localhost melakukan subscribe ke broker yang sama, agar data yang dipublish oleh tiap produk dapat dimasukkan ke database.

2.4.1.2.2 Client-side (Produk)

2.4.1.2.2.1 Pengiriman data ke server

Sesuai pada alur data yang terdapat di Gambar 2.4.2, produk sebagai client akan mempublish data ke broker. Data tersebut akan diformat dalam bentuk JSON string yang berisikan parameter produk. Secara umum, ada 3 topik yang akan dipublish oleh satu produk. Rincian mengenai data yang dikirim untuk tiap topic tersebut dapat dilihat pada Tabel 2.4.1

Tabel 2.4.1 Daftar topik dan informasi yang dipublish ke topik yang bersangkutan

Topic	Data yang dikirim
sys/s1	Product ID, Status, Serial Number, Time, SoC, SoH, Voltage dari slot 1
sys/s2	Product ID, Status, Serial Number, Time, SoC, SoH, Voltage dari slot 2
sys/device	Product ID, mode pengisian daya, dan lokasi produk (CGI)

Bentukan JSON untuk topik diatas dapat dilihat pada bagian dibawah ini

Contoh JSON String untuk sys/s1 dan sys/s2	Contoh JSON String untuk sys/device
{ "Product ID" : 1, >Status" : 0, "SN" : "0123456789abcdef", "Time" : "22/04/07,19:12:42", "SoC" : 50, "SoH" : 75, "Volt" : 59, "Charging Mode" : 0 }	{ "Product ID" : 1, "Charging Mode": 0, "MCC" : 510, "MNC" : 10, "LAC" : 14016, "Cell ID" : 65372 }

Penjelasan mengenai database dan web ini dapat dilihat lebih lanjut pada Subbab 2.4.1.2.4 dan 2.4.1.2.6.

Dalam JSON String slot 1 dan slot 2, terdapat 8 *dict* (*JSON dictionary*) yang menyimpan informasi terkait penggunaan slot. Adapun *dict* tersebut ialah

- Status, berisi integer yang menandakan kondisi slot. Nilai 0 menandakan bahwa slot sedang idle, nilai 1 sedang mengisi daya, dan nilai 2 menadakan bahwa pengisian daya sudah selesai, namun baterai belum dicabut dari slot.
- Product ID, berisi ID produk yang merupakan integer unik dari setiap produk, untuk keperluan penampilan data singular. Product ID ini di *define* pada kode sebagai PRODUCT_ID, yang nantinya dapat diubah sesuai keperluan.

- SN, memuat string 16 karakter yang merupakan nomor serial dari baterai. Nomor serial tersebut didapatkan dengan mengirim command melalui komunikasi serial ke BMS di baterai oyika (lebih lengkapnya dapat dilihat di subsistem kontrol pada Subbab 2.2.1.2).
- Time, memuat timestamp yang didapatkan dari modul SIM800L saat dilakukan upload. Format waktu yang digunakan ialah DD/MM/YYYY, hh:mm:ss.
- SoC, memuat SoC dari slot yang bersangkutan. Nilai SoC merupakan indikator kapasitor baterai berbentuk integer dengan rentang 0 – 100 (persen) yang menandakan persentase kapasitas baterai
- SoH, memuat SoH dari slot yang bersangkutan. Nilai SoH merupakan integer dengan rentang 0 – 100 (persen) yang menandakan persen kapasitas baterai sekarang, dibandingkan dengan kapasitas baterai saat awal manufaktur.
- Voltage, memuat tegangan nominal baterai yang terbaca dalam satuan Volt.
- Charging Mode, memuat mode pengisian daya dari slot. Pada dasarnya, kedua slot akan memiliki mode pengisian daya yang sama. Namun informasi terkait charging mode ini perlu disertakan juga di JSON Dict ini agar pengolahan data riwayat baterai yang pernah terhubung ke charger.

Untuk mendapatkan parameter Time, akan dikirimkan *command* ke modul GSM melalui serial, dan modul GSM akan mengembalikan respons berupa string yang mengandung informasi tanggal dan waktu. Adapun command yang digunakan adalah

```
> AT+CCLK?
[response] = 22/04/07,19:14:28
```

Command tersebut akan mengembalikan informasi tanggal dan waktu, sesuai dengan penjabaran pada Gambar 2.4.3

Write Command	Response
AT+CCLK=<time>	OK If error is related to ME functionality: +CME ERROR: <err>
	Parameters <time> String type(string should be included in quotation marks) value; format is "yy/MM/dd,hh:mm:ss±zz", where characters indicate year (two last digits),month, day, hour, minutes, seconds and time zone (indicates the difference, expressed in quarters of an hour, between the local time and GMT; range -47...+48). E.g. 6th of May 2010, 00:01:52 GMT+2 hours equals to "10/05/06,00:01:52+08".

Gambar 2.4.3 Keterangan mengenai parameter <time>, dan output command AT+CCLK [6]

Kemudian dalam JSON String device, terdapat 5 *dict* yang menyimpan informasi terkait penggunaan produk, yakni mode pengisian daya dan lokasi produk. Sesuai dengan proses desain yang dilakukan di dokumen B300, lokasi akan didapatkan melalui modul SIM800L, khususnya ialah dengan parameter Cell Global Identity (CGI) dari Base Transceiver Station

(BTS) suatu network. Parameter CGI terdiri atas 4 informasi, yaitu MCC, MNC, LAC, dan Cell ID.

Untuk mendapatkan parameter CGI, dapat dikirim perintah berikut ke modul SIM800L melalui komunikasi serial

```
> AT+CNETSCAN=1
[response] = OK
> AT+CNETSCAN
[response]
Operator:"EXCELCOM",MCC:510,MNC:11,Rxlev:41,Cellid:06DA,Arfcn:51
4,Lac:6214,Bsic:03
Operator:"51089",MCC:510,MNC:89,Rxlev:51,Cellid:6EE8,Arfcn:673,L
ac:290E,Bsic:2A
Operator:"TELKOMSEL",MCC:510,MNC:10,Rxlev:49,Cellid:FE14,Arfcn:7
76,Lac:36C8,Bsic:0A
Operator:"TELKOMSEL",MCC:510,MNC:10,Rxlev:44,Cellid:9C3C,Arfcn:7
80,Lac:36FB,Bsic:2B
```

Pertama diatur terlebih dahulu parameter CNETSCAN=1 agar CGI yang didapatkan mengandung 4 parameter yang dibutuhkan. Setelah itu, respon yang diberikan oleh modul GSM akan berbentuk seperti pada Gambar 2.4.4

Execution Command AT+CNETSCA N	Response If format's value is 0: Operator:<Network_Operator_name>,MCC:<MCC>,MNC:<MNC> ,Rxlev:<Rxlev>,Cellid:<CellID>,Arfcn:<Arfcn> <CR><LF> Operator: "<Network_Operator_name2>",MCC:<MCC2>,MNC:<MNC2>,Rxlev <Rxlev2>,Cellid:<CellID2>,Arfcn:<Arfcn2>[...] If format's value is 1: Operator:<Network_Operator_name>,MCC:<MCC>,MNC:<MNC> ,Rxlev:<Rxlev>,Cellid:<CellID>,Arfcn:<Arfcn>,Lac:<Lac>,Bsic:<Bsic > <CR><LF> Operator:<Network_Operator_name2>,MCC:<MCC2 >,MNC:<MNC2>,Rxlev:<Rxlev2>,Cellid:<CellID2>,Arfcn:<Arfcn2>,L ac:<Lac2>,Bsic:<Bsic2>[...] OK
---	---

Gambar 2.4.4 Respons modul GSM saat diberi command AT+CNETSCAN [7]

Terlihat pada cuplikan kode bahwa perintah CNETSCAN mengembalikan beberapa kombinasi parameter CGI. Yang akan diupload hanya set parameter CGI pertama saja. Setelah set parameter tersebut diparse, data tersebut disispkjan ke *dict* yang sesuai, dan menunggu diupload. Namun karena proses ini memakan waktu cukup lama, pengambilan data CGI ini akan dilakukan saat awal produk menyala pertama kali, dan hanya dilakukan sekali saja. Bila terdapat perubahan pada slot 1 atau slot 2, data CGI beserta data lainnya baru dipublish ke broker. Dengan adanya parameter CGI, dapat dilakukan aproksimasi lokasi produk. Aproksimasi tersebut dilakukan dengan mengirimkan JSON payload berisikan CGI ke Google Geolocation API.

Selain dari itu, terdapat juga beberapa command yang dapat dikirim ke SIM800L yang membantu proses pengambilan data. Salah satunya ialah command untuk mengecek apakah SIM800L sudah terhubung ke suatu jaringan, yang dilakukan dengan

```
> AT+CREG?  
[response] = +CREG: 0,0
```

2.4.1.2.2.2 Penerimaan data dari server

Agar produk dapat dikendalikan secara remote dari server, diperlukan sistem untuk menerima data (perintah) dari server yang juga dilakukan melalui protokol MQTT. Pada dasarnya, produk akan mensubscribe ke topik yang bersangkutan, dan server akan mengirimkan perintah di topik tersebut agar produk dapat melaksanakan perintah yang dikirimkan. Untuk saat ini, perintah yang dikirimkan masih sebatas perintah untuk mematikan dan menyalakan produk, sehingga belum memerlukan JSON string sebagai struktur penyusun payload yang dikirimkan. Adapun topik yang digunakan ialah

```
sys/charger [N] /commands
```

Selanjutnya, bila perintah sudah diterima oleh produk, maka produk akan mengirimkan *callback response* ke server melalui topic yang berbeda. Callback ini hanya berisi string “received”, agar server mendapatkan konfirmasi bahwa produk sudah menerima perintah. Penggunaan topic yang berbeda ditujukan agar tidak terjadi konflik pada saat penerimaan perintah. Adapun topik yang digunakan ialah

```
sys/charger [N] /commandsCallback
```

2.4.1.2.3 MQTT Broker

Broker yang digunakan ialah Mosquitto versi 2.0.14. Untuk menjalankan service broker ini, perlu diunduh terlebih dahulu file yang diperlukan dari situs Mosquitto[1]. Agar produk dapat mem-*publish* topic menggunakan koneksi GPRS dari mana saja, service Mosquitto harus dijalankan di perangkat yang memiliki IP Public. Maka untuk sekarang, service Mosquitto dijalankan di Google Compute Engine Virtual Machine (GCE-VM) instance. VM yang dijalankan memiliki sistem operasi Linux, dengan IP Public 34.101.49.52.

Perlu diketahui bahwa agar VM dapat menerima koneksi dari luar, perlu dilakukan pengaturan terhadap Firewall dari VM instance yang digunakan. Untuk keperluan tugas akhir ini, telah dialokasikan whitelist port 1883 untuk melakukan koneksi ke broker.

Kemudian, Mosquitto dapat diinstal di VM melalui terminal, dengan perintah sebagai berikut

Command	Keterangan
> sudo su > cd .. > cd ..	Memindahkan working directory ke system root
> sudo apt-get update	Update packages
> sudo apt-get install mosquitto	Install mosquitto

Setelah aplikasi Mosquitto sudah dimasukkan ke VM instance, aplikasi mosquito.exe dapat dijalankan pada cmd dengan perintah berikut

```
> mosquitto -c mosquitto.conf -v
```

Perintah -c menandakan bahwa akan digunakan config khusus untuk pengoperasian service broker, sedangkan perintah -v menandakan bahwa broker akan berjalan dalam mode debug, sehingga akan muncul pesan error apabila terdapat error. Pada file config tersebut, ditambahkan dua perintah berikut

Setelah setup broker selesai, produk baru dapat mempublish data ke broker. Kemudian datalogger juga akan men-*subscribe* ke setiap topic yang dipublish produk. Sesuai dengan pembahasan pada Subbab 2.4.1.2.2, setiap produk akan mempublish beberapa data ke tiga topic yang berbeda setiap terjadinya perubahan status pengisian daya (*charging*, *idle*, *finished charging*), atau perubahan mode pengisian daya (*slow charging* atau *fast charging*).

2.4.1.2.4 Datalogger

Untuk menyimpan informasi yang telah dipublish oleh tiap produk, digunakan sebuah Python Script yang men-*subscribe* ke topic sys/# agar dapat menerima informasi dari setiap produk. Data yang diterima akan disisipkan ke tabel yang bersangkutan pada database berdasarkan path dari topic yang diterima (sys/s1, sys/s2, atau sys/device). Penyimpanan data nya sendiri akan dilakukan menggunakan SQLite, agar query database bisa dilakukan lebih mudah.

Pada database, terdapat 3 tabel utama yang masing masing menyimpan informasi mengenai slot 1, slot 2, dan parameter umum untuk setiap device yang terhubung. Tabel tersebut tidak dipisah mengingat keterbatasan SQLite yang hanya bisa memuat maksimal sebanyak 64 tabel. Pada Tabel 2.4.2 dan Tabel 2.4.3 dapat dilihat struktur tabel yang digunakan untuk menyimpan data yang dipublish produk.

Tabel 2.4.2 Bentukan tabel charger_s1 dan charger_s2

#	Product ID	Status	Serial Number	Time	SoC	SoH	Voltage	Charging Mode

Tabel 2.4.3 Bentukan tabel charger_device

#	Product ID	Charging Mode	MCC	MNC	LAC	Cell ID

2.4.1.2.5 APIs

2.4.1.2.5.1 Google Geolocation API

Informasi lokasi yang didapatkan oleh modul SIM800L masih berbentuk parameter CGI, yang sulit diartikan dalam bentuk lokasi. Dengan menggunakan Geolocation API yang disediakan oleh Google, parameter CGI tersebut dapat diubah menjadi sepasang koordinat latitude dan longitude. API tersebut pada dasarnya merupakan database Google yang memuat informasi koordinat berdasarkan CGI yang diberikan. Link tujuan post request, isi payload, serta callback response yang didapatkan dapat dilihat pada

Tabel 2.4.4 Keterangan mengenai Google Geolocation API

Keterangan	Isi
http	<code>https://www.googleapis.com/geolocation/v1/geolocate?key=<API_KEY></code>
payload	<code>{ "considerIp": true, "cellTowers": [{ "cellId": 28392, "locationAreaCode": 10510, "mobileCountryCode": 510, "mobileNetworkCode": 89 }] }</code>
callback	<code>{ "location": { "lat": -6.8759115,</code>

	<pre> "lng": 107.6119818 }, "accuracy": 988 } </pre>
--	--

2.4.1.2.5.2 Google Maps API

Untuk memudahkan proses pemantauan, koordinat latitude dan longitude yang didapatkan akan disajikan pada map yang terdapat di web dashboard. Untuk itu, akan digunakan Google Maps API untuk menampilkan peta pada web. Untuk mempermudah proses penampilan map, digunakan script untuk membuat class map, kemudian langsung disisipkan ke body page html.

Tabel 2.4.5 Keterangan mengenai Google Maps API

Keterangan	Isi
http	<a href="https://maps.googleapis.com/maps/api/js?key=<API_KEY>&callback=initMap">https://maps.googleapis.com/maps/api/js?key=<API_KEY>&callback=initMap
script	<pre> function initMap() { var coords = { lat: {{ lat }}, lng: {{ lon }} }; var map = new google.maps.Map(document.getElementById('map'),{ center: coords, disableDefaultUI: true, zoom: 16 }); var marker1 = new google.maps.Marker({ position: coords, map: map }); } </pre>
Tampilan Map	

2.4.1.2.6 Webserver

Web dashboard akan dijalankan menggunakan webserver Flask yang dijalankan menggunakan python script di localhost. Server ditempatkan di localhost agar proses perancangan dashboard dapat dilakukan dengan lebih mudah. Pada tugas akhir ini, framework Flask dipilih sebagai backend web dashboard berdasarkan aspek kemudahan pemrograman, mengingat kode backend nya dimuat pada file yang sama. Pada sisi frontend,

terdapat kode Jinja2 yang dapat digunakan ditengah tengah kode HTML untuk menampilkan variabel dari backend, atau sekedar melakukan pemrograman backend sederhana (seperti logika if statement, for loop, dsb).

Secara umum, fitur yang terdapat pada web server yang telah dirancang ialah

- Halaman utama yang memuat informasi produk secara umum, seperti status terkini dari slot pada produk, mode pengisian daya, dan tanggal terakhir server menerima data dari produk
- Halaman khusus bagi setiap produk untuk melihat riwayat informasi yang telah dikirimkan oleh produk, yang mencakup riwayat nomor seri, *timestamp*, SOC, SOH, dan tegangan nominal dari baterai yang terhubung ke slot 1 maupun slot 2. Selain itu, terdapat juga informasi lokasi produk
- Data processing sederhana yang berupa rekap historis pengisian daya baterai yang pernah diisi menggunakan sistem yang dirancang berdasarkan nomor seri baterai tersebut.
- Fitur kontrol sederhana untuk mengaktifkan atau menonaktifkan produk secara *remote*
- Fitur user management, seperti login, add user, delete user, edit user bagi petugas Oyika agar data tidak dapat diakses oleh oknum yang tidak berwenang.

Fitur-fitur tersebut akan dijelaskan lebih lanjut pada bagian source code.

Pada bagian user interface, digunakan CSS Framework Bootstrap 4 untuk mempermudah proses desain UI. Selain itu, digunakan juga jQuery versi 3.6.0, khususnya fitur dataTables nya agar penyajian tabel dapat dibuat lebih responsif.

Hasil implementasi fitur-fitur tersebut dapat dilihat pada Subbab pengujian.

2.4.1.3 Source Code

Untuk menjalankan kode dibawah ini, terdapat beberapa dependensi yang perlu diinstall terlebih dahulu, baik pada sisi client (ESP32) maupun pada sisi server (Python). Dependensi yang diperlukan pada sisi client dapat diinstal melalui library manager yang terdapat di Arduino IDE. Adapun dependensi yang diperlukan pada sisi client ialah

- TinyGsmClient.h
- PubSubClient.h
- Adafruit_ADS1X15.h
- Wire.h
- SoftwareSerial.h

Dependensi yang diperlukan pada sisi server seluruhnya merupakan library python yang dapat diinstal menggunakan pip. Adapun dependensi tersebut ialah

- Paho-mqtt
- Flask
- Requests
- Sqlite3
- hashlib

2.4.1.3.1 **Publish ke Broker**

Program publish ke broker dijalankan di ESP32 bersamaan dengan kode untuk kontrol pengisian daya, interface, dsb. Sementara, pengembangan subsistem IoT dilakukan terpisah dengan subsistem lainnya, sehingga variabel yang digunakan pada subsistem ini mungkin berbeda dengan variabel yang dijelaskan pada subsistem lainnya. Perlu diketahui pula bahwa koneksi ke broker MQTT dilakukan dengan library eksternal bernama <PubSubClient.h> yang mempermudah proses koneksi. Inisialisasi library tersebut dapat dilihat pada potongan kode dibawah

Kode inisialisasi modul dan definition untuk subsistem IoT
<pre>#define PRODUCT_ID 3 #define TINY_GSM_MODEM_SIM800 // Modem is SIM800L #define Sim8001 Serial1 #include <TinyGsmClient.h> TinyGsm modem(Sim8001); TinyGsmClient clientelle(modem); PubSubClient client(clientelle); char topics[3][20]; char commandTopic[25]; sprintf(topics[0], "sys/charger%d/s1", PRODUCT_ID); sprintf(topics[1], "sys/charger%d/s2", PRODUCT_ID); sprintf(topics[2], "sys/charger%d/device", PRODUCT_ID); sprintf(commandTopic, "sys/charger%d/commands", PRODUCT_ID); void setup() { // Initialize serial communication Serial.begin(9600); Sim8001.begin(9600, SERIAL_8N1, MODEM_RX, MODEM_TX); // Connect to GPRS if (checkGsmNetwork() == 1){ gprsConnectFuction();</pre>

```

    }
    else{
        Serial.println("Network unavailable, retrying GPRS in 30secs");
    }

    // mqtt params
    client.setServer(brokerIP, 1883);
    client.setCallback(callback);

}

```

Sementara waktu, proses publish data ke broker masih dilakukan dengan WiFi untuk mempermudah proses perancangan. Setelah perancangan mendekati tahap akhir, baru akan dilakukan publish data melalui jaringan GSM.

Pada bagian setup, dilakukan inisialisasi setCallback dan setClient untuk proses MQTT, serta penyettingan IP dan Port untuk server broker MQTT. Port default untuk protokol MQTT adalah 1883, dan IP address yang digunakan adalah IP VM (Virtual Machine) yang dihost pada GCE yang meng-host broker MQTT.

Kemudian, proses utama subsistem ini diletakkan pada Task baru, yaitu Task MQTTLoop. Kode penyusun task ini dapat dilihat pada bagian dibawah

Kode loop pada produk untuk proses publish data

```

void MQTTLoop(void *parameter)
{
    for (;;)
    {
        char blank[1] = "";
        char stringHolder[256];

        int gsmStatus = checkGsmNetwork();
        int modemStatus = modem.isGprsConnected();

        // Setiap 15 detik, apabila koneksi GPRS belum ada, dipanggil
        // fungsi koneksi GPRS
        if (millis() - lastGprsAttempt > 15000 && gsmStatus == 1 &&
modemStatus == 0){
            if (modem.isGprsConnected()){
                Serial.println("GPRS Connected");
            }
            else{
                Serial.println("GPRS not connected, connecting...");
                lastGprsAttempt = millis();
                gprsConnectFuction();
                if (modem.isGprsConnected()){

                }
        }
    }
}

```

```

        }

    }

    // Bila sudah terkoneksi GPRS namun belum terkoneksi ke broker
    // MQTT dipanggil fungsi koneksi ke broker
    if (!client.connected() && modem.isGprsConnected()){
        reconnectmqttserver();
    }

    // fungsi untuk keberjalanannya modul PubSubClient.h
    client.loop();

    // Sampling lokasi apabila informasi belum didapatkan
    if (!(isGetLocation))
    {
        if (checkGsm() == 1){
            getLocation(locDetails);
            Serial.println("Loc:");
            for (int l = 0; l < 4; l++){
                Serial.println(locDetails[l]);
            }

            isGetLocation = 1;
        }
    }

    // Proses bila sudah terkoneksi ke broker
    // terdapat beberapa variabel command yang diset 1 pada state
    // tertentu di Task Left dan Right battery
    // proses publish harus dilakukan terpisah dengan Task Left dan
    // Right, karena proses komunikasi dengan SIM800L harus ada di
    // task yang level prioritas nya lebih tinggi
    if (client.connected()){
        // command publish data slot kiri (slot 1)
        if (cmd_sendSlotL == 1){
            strcpy(stringHolder, blank);
            buildStringBatteryInfo(0, stringHolder);
            Serial.println(stringHolder);
            client.publish(topics[0], stringHolder);
            strcpy(stringHolder, blank);
            buildStringDeviceInfo(stringHolder);
            Serial.println(stringHolder);
            client.publish(topics[2], stringHolder);

            cmd_sendSlotL = 0;
        }
        // command publish data slot kiri (slot 2)
        if (cmd_sendSlotR == 1){
    
```

```

        strcpy(stringHolder, blank);
        buildStringBatteryInfo(1, stringHolder);
        Serial.println(stringHolder);
        client.publish(topics[1], stringHolder);
        strcpy(stringHolder, blank);
        buildStringDeviceInfo(stringHolder);
        Serial.println(stringHolder);
        client.publish(topics[2], stringHolder);

        cmd_sendSlotR = 0;
    }
}

// bila koneksi ke broker belum ada, command publish ditahan dulu
// publish dilakukan setelah dilakukan koneksi ke broker
else{
    Serial.println("Can't upload, MQTT / GPRS unavailable");
    Serial.print("Network status: ");
    Serial.println(gsmStatus);
    Serial.print("GPRS status: ");
    Serial.println(modemStatus);
}

vTaskDelay(2 / portTICK_PERIOD_MS);
}
}

```

Secara umum, pada task MQTTLLoop ini terdapat 3 proses besar yang dilakukan, yakni:

- Pertama, task melakukan pengecekan apakah SIM800L sudah berhasil terhubung ke jaringan provider dan koneksi GPRS. Bila belum terhubung, setiap 15 detik sekali akan dipanggil fungsi `gprsConnectFunction` yang bertujuan untuk melakukan koneksi ke internet via GPRS.
- Kedua, task melakukan pengecekan apabila produk belum terhubung ke broker, akan dipanggil fungsi `reconnectmqttserver` untuk mencoba melakukan koneksi. Untuk keberjalanannya library `PubSubClient.h`, dilakukan juga pemanggilan fungsi `client.loop`.
- Ketiga, bila informasi lokasi belum didapatkan, maka produk akan menjalankan fungsi untuk mendapatkan parameter lokasi (empat parameter CGI). Perlu diketahui bahwa proses akuisisi parameter lokasi memakan waktu yang cukup lama (sekitar 15 hingga 45 detik). Maka dari itu, pada kondisi dimana produk dinyalakan dan langsung digunakan (dicolok baterai), maka produk tidak dapat mengupload data slot sebelum parameter lokasi berhasil didapatkan. Meskipun pada implementasinya dapat dibuat agar tidak menunggu akuisisi data lokasi, tim penulis memutuskan untuk menerapkan sistem seperti demikian, agar data yang dikirimkan oleh produk memiliki informasi lokasi yang tepat.

- Dan keempat, task akan lakukan pengecekan apakah sudah terkoneksi ke broker MQTT. Bila sudah terkoneksi, task akan melakukan publish data ke broker, sesuai dengan variabel command yang diset oleh task LeftCode dan RightCode. Proses publish ini harus dipisah dari kedua task tersebut karena terdapat proses komunikasi dengan SIM800L yang harus dijalankan pada task yang memiliki level prioritas paling tinggi. Bila tidak, command serial yang dikirimkan tidak akan terbaca oleh modul SIM800L.

Namun bila produk belum terkoneksi ke broker, tidak akan dilakukan apapun. Task akan menunggu proses pertama dan keduanya berhasil melakukan koneksi.

Pada proses ketiga, terdapat proses publish informasi ke broker. Informasi yang dipublish memiliki format JSON, sehingga perlu diformat terlebih dahulu menggunakan fungsi tertentu. Pemrosesan string untuk mengirim data mengenai slot baterai dilakukan dengan fungsi `buildStringBatteryInfo`, sedangkan pengiriman data mengenai informasi produk (mode pengisian daya dan lokasi) diproses dengan fungsi `buildStringDeviceInfo`. Hasil pemrosesan string disimpan dalam variabel `stringHolder`, kemudian isi dari variabel holder dipublish ke topic yang bersangkutan.

Isi dari fungsi `buildStringBatteryInfo` dapat dilihat pada bagian dibawah ini

Fungsi <code>buildStringBatteryInfo</code>
<pre>void buildStringBatteryInfo(int battSelect, char *holder) { char dbIndex[8][20] = { "\"Product ID\": ", "\"Status\": ", "\"Serial Number\": ", "\"Time\": ", "\"SoC\": ", "\"SoH\": ", "\"Volt\": ", "\"Charging Mode\": "}; char sep[3] = ", "; char buildSN[18]; char blank[1] = ""; char ob = '{'; char cb = '}'; char stringHolder[30]; // Openbracket strncat(holder, &ob, 1); // Product ID strncat(holder, dbIndex[0]); sprintf(stringHolder, "%d", PRODUCT_ID); strncat(holder, stringHolder); strcpy(stringHolder, blank); strncat(holder, sep); // Status strncat(holder, dbIndex[1]);</pre>

```

if (battSelect == 0)
{
    if (initialUpload == 0){
        sprintf(stringHolder, "%d", -1);
    }
    else{
        if(stateL == CHARGING){
            sprintf(stringHolder, "%d", 1);
        }
        else if(stateL == FINISH_CHARGING){
            sprintf(stringHolder, "%d", 2);
        }
        else{
            sprintf(stringHolder, "%d", 0);
        }
    }
}

else
{
    if (initialUpload == 0){
        sprintf(stringHolder, "%d", -1);
    }
    else{
        if(stateR == CHARGING){
            sprintf(stringHolder, "%d", 1);
        }
        else if(stateR == FINISH_CHARGING){
            sprintf(stringHolder, "%d", 2);
        }
        else{
            sprintf(stringHolder, "%d", 0);
        }
    }
}

strcat(holder, stringHolder);
strcpy(stringHolder, blank);
strcat(holder, sep);

// Serial Number
strcat(holder, dbIndex[2]);
if (battSelect == 0)
{
    sprintf(stringHolder, "\"%s\"", serialNumberL);
    strcat(holder, stringHolder);
}

```

```

else
{
    sprintf(stringHolder, "\"%s\"", serialNumberR);
    strcat(holder, stringHolder);
}

strcpy(stringHolder, blank);
strcat(holder, sep);

// Time
strcat(holder, dbIndex[3]);
strcpy(stringHolder, "\"\"");
getCurrentTime(stringHolder);
strcat(holder, stringHolder);
strcpy(stringHolder, blank);
strcat(holder, sep);

// SOC
strcat(holder, dbIndex[4]);
if (battSelect == 0)
{
    sprintf(stringHolder, "%.2f", SoCL*100);
}
else
{
    sprintf(stringHolder, "%.2f", SoCR*100);
}
strcat(holder, stringHolder);
strcpy(stringHolder, blank);
strcat(holder, sep);

// SOH
strcat(holder, dbIndex[5]);
if (battSelect == 0)
{
    sprintf(stringHolder, "%.2f", float(SoHL)/10);
}
else
{
    sprintf(stringHolder, "%.2f", float(SoHR)/10);
}
strcat(holder, stringHolder);
strcpy(stringHolder, blank);
strcat(holder, sep);

// Volt
strcat(holder, dbIndex[6]);
if (battSelect == 0)
{
}

```

```

        sprintf(stringHolder, "%.2f", voltage/1000);
    }
    else
    {
        sprintf(stringHolder, "%.2f", voltage/1000);
    }
    strcat(holder, stringHolder);
    strcpy(stringHolder, blank);
    strcat(holder, sep);

    // Charging Mode
    strcat(holder, dbIndex[7]);
    sprintf(stringHolder, "%d", chargingMode);
    strcat(holder, stringHolder);
    strcpy(stringHolder, blank);

    // closebracket
    strncat(holder, &cb, 1);
}

```

Kemudian, fungsi `buildStringDeviceInfo` dapat dilihat pada bagian dibawah ini. Fungsi ini bertujuan untuk membuat JSON string yang memuat informasi produk seperti yang telah dijelaskan diatas

Fungsi <code>buildStringDeviceInfo</code>
<pre> void buildStringDeviceInfo(char *holder) { char dbIndex[6][20] = { "\"Product ID\": ", "\"Charging Mode\": ", "\"MCC\": ", "\"MNC\":", ", \"LAC\": ", "\"Cell ID\": "; char sep[3] = ","; char blank[1] = ""; char ob = '{'; char cb = '}'; char stringHolder[30]; // Openbracket strncat(holder, &ob, 1); // Product ID strcat(holder, dbIndex[0]); sprintf(stringHolder, "%d", PRODUCT_ID); strcat(holder, stringHolder); strcpy(stringHolder, blank); strcat(holder, sep); } </pre>

```

// Charging Mode
strcat(holder, dbIndex[1]);
sprintf(stringHolder, "%d", chargingMode);
strcat(holder, stringHolder);
strcpy(stringHolder, blank);
strcat(holder, sep);

// Location
// getLocation(locDetails);

for (int i = 0; i < 4; i++)
{
    strcat(holder, dbIndex[i + 2]);
    sprintf(stringHolder, "%d", locDetails[i]);
    strcat(holder, stringHolder);
    strcpy(stringHolder, blank);
    if (i < 3)
    {
        strcat(holder, sep);
    }
}

// closebracket
strncat(holder, &cb, 1);
}

```

Kedua fungsi penyusun JSON string diatas memiliki beberapa fungsi bantuan, salah satunya adalah fungsi `getCurrentTime`, yang mendapatkan waktu melalui jaringan GSM.

Fungsi <code>getCurrentTime</code>
<pre> void getCurrentTime(char *currentTime) { if (checkGsm() == 0) { Serial.println("SIM800L offline, cannot get time"); return; } int intercharTime = 0; Serial.println("Get time info ..."); Sim8001.print("AT+CCLK?\r"); String buffer2; while (Sim8001.available()) { </pre>

```

        Serial.write(Sim8001.read());
    }

    while ((intercharTime) < MAX_DELAY_MS_TIME)
    {
        if (Sim8001.available())
        {
            int c = Sim8001.read();
            // Serial.print((char)c);
            buffer2.concat((char)c);
            intercharTime = 0;
        }
        else
        {
            intercharTime += LOOP_DELAY_MS;
        }
        vTaskDelay(LOOP_DELAY_MS / portTICK_PERIOD_MS);
    }

    int str_len = buffer2.length() + 1;
    char strProcess[str_len];
    buffer2.toCharArray(strProcess, str_len);

    // Parsing (+CCLK: "03/01/01,00:01:58+28")
    char *token;
    token = strtok(strProcess, "\\");
    token = strtok(NULL, "+");
    if (token == NULL)
    {
        Serial.println("\nBuffer error, force quit fx\n");
        return;
    }
    // the sprintf solution
    char timeString[str_len + 2];
    sprintf(timeString, sizeof timeString, "%s\\", token);
    strcpy(currentTime, timeString);
    return;
}

```

Terdapat pula fungsi `checkGsm` yang melakukan pengecekan apakah modul GSM sedang bekerja dengan baik. Kode yang merealisasikan fungsi tersebut dapat dilihat pada bagian dibawah ini

Fungsi <code>checkGsm</code>			
<code>int checkGsm()</code>			
Nomor Dokumen: B400- TA2122.01.024 Nomor Revisi: 01 Tanggal: 6/26/2022 Halaman 64 dari 108			
© 2022 Prodi Teknik Elektro-ITB. Pengungkapan dan penggunaan seluruh isi dokumen hanya dapat dilakukan atas ijin tertulis Prodi Teknik Elektro - ITB Jalan Ganesha 10 Bandung, 40132 Indonesia.			

```

{
    String buffer2;
    int intercharTime = 0;
    Sim8001.print("AT\r");

    while ((intercharTime) < 100)
    {
        if (Sim8001.available())
        {
            int c = Sim8001.read();
            //Serial.print((char)c);
            buffer2.concat((char)c);
            intercharTime = 0;
        }
        else
        {
            intercharTime += 1;
        }
        vTaskDelay(1 / portTICK_PERIOD_MS);
    }

    if (buffer2.indexOf("OK") > 0)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

```

Selain itu, terdapat juga fungsi `checkGsmNetwork` yang bertugas untuk melakukan pengecekan apakah produk sudah terhubung ke jaringan GSM. Kode yang merealisasikan fungsi tersebut dapat dilihat pada bagian dibawah ini.

Fungsi `checkGsmNetwork`

```

int checkGsmNetwork(){
    String buffer2;
    int intercharTime = 0;
    Sim8001.print("AT+CREG?\r");

    while ((intercharTime) < 100)
    {
        if (Sim8001.available())
        {

```

```

        int c = Sim8001.read();
        // Serial.print((char)c);
        buffer2.concat((char)c);
        intercharTime = 0;
    }
    else
    {
        intercharTime += LOOP_DELAY_MS;
    }

    delay(LOOP_DELAY_MS);
}

if (buffer2.indexOf("+CREG: 0,1")>0){
    return 1;
}
else{
    return 0;
}
}

```

Bila produk sudah terhubung ke jaringan GSM, maka dijalankan fungsi gprsConnectFunction yang bertugas untuk menghubungkan produk ke jaringan GPRS agar dapat mengirimkan data ke server melalui internet. Kode yang merealisasikan fungsi tersebut dapat dilihat pada bagian dibawah ini

Fungsi gprsConnectFunction

```

void gprsConnectFuction(){
    Serial.println("Initializing modem...");
    modem.init();

    String modemInfo = modem.getModemInfo();
    Serial.print("Modem Info: ");
    Serial.println(modemInfo);

    Serial.print("Connecting to APN: ");
    Serial.print(apn);
    if (!modem.gprsConnect(apn)) {
        Serial.println(" fail");
        // try again in 30 secs
        // ESP.restart();
    }
    else {
        Serial.println(" -- OK");
    }
}

```

```

if (modem.isGprsConnected()) {
    Serial.println("GPRS connected");
}
}

```

Selanjutnya, dibawah ini dapat dilihat fungsi yang bertugas untuk melakukan koneksi ke server broker

Fungsi yang dipanggil untuk melakukan koneksi ke MQTT broker
<pre> void reconnectmqttserver() { while (!client.connected()) { Serial.print("Attempting MQTT connection..."); String clientId = "ESP32Client-"; clientId += String(random(0xffff), HEX); if (client.connect(clientId.c_str())) { Serial.println("connected"); } else { Serial.print("failed, rc="); Serial.print(client.state()); Serial.println(" try again in 5 seconds"); delay(5000); } } } </pre>

Fungsi client.connect digunakan untuk melakukan koneksi ke broker MQTT dengan IP adress dan Port yang telah dideklarasikan di bagian setup. Bila koneksi tidak dapat dilakukan, program akan mencoba melakukan koneksi secara periodik tiap 5 detik.

Selanjutnya, terdapat pula fungsi callback yang dijalankan ketika terdapat pesan baru di topic yang telah disubscribe. Kode yang merealisasikan fungsi tersebut dapat dilihat pada bagian dibawah ini

fungsi callback
<pre> void callback(char *topic, byte *payload, unsigned int length) { Serial.print("Message arrived on topic: "); Serial.print(topic); Serial.print(". Message: "); String MQTT_DATA = ""; for (int i=0;i<length;i++) { MQTT_DATA += (char)payload[i]; } } </pre>

```

Serial.print(MQTT_DATA);

isEnabledCmd = MQTT_DATA.toInt();
EEPROM.write(0, isEnabledCmd);
EEPROM.commit();

client.publish(commandTopicCallback, "received");
Serial.println();
}

```

2.4.1.3.2 Subscribe ke Broker dan Data Logging

Subblok subscribe ke broker ini dilakukan di server menggunakan bahasa Python. Subblok ini harus selalu menyala dan men-subscribe ke semua topic yang dikirim oleh produk. Pada protokol MQTT, subscribe terhadap topic yang banyak dilakukan dengan karakter '#', sehingga komponen ini hanya perlu melakukan subscribe ke 1 topic, yaitu topic sys/#, dan hasil publish semua produk dapat dibaca, dan dapat disisipkan ke file database.db.

Python script untuk menerima informasi (subscribe) dari broker MQTT
<pre> import paho.mqtt.client as mqtt from dbHandler import dataHandler # MQTT Parameters broker = "192.168.1.102" port = 1883 keepAliveInterval = 45 masterTopic = "sys/#" # Subscribe to topic def on_connect(mosq, obj, rc, properties=None): mqttc.subscribe(masterTopic, 0) # Save to DB table def on_message(mosq, obj, msg): print ("MQTT Data Received...") print ("MQTT Topic: " + msg.topic) print ("Data: ",end='') print (msg.payload) dataHandler(msg.topic, msg.payload) def on_subscribe(mosq, obj, mid, granted_qos): pass # Deklarasi obj mqttc </pre>

```

mqttc = mqtt.Client()

# Assign callback
mqttc.on_message = on_message
mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe

# Connect
mqttc.connect(broker, int(port), int(keepAliveInterval))

# Network loop
mqttc.loop_forever()

```

Secara umum, kode diatas bertugas untuk men-subscribe ke semua topic yang dipublish oleh produk. Bila terdapat action publish dari salah satu produk, maka kode akan menjalankan fungsi `on_message`, yang didalamnya terdapat pemanggilan fungsi `dataHandler` untuk menyisipkan data ke database. Fungsi `dataHandler` ini menyisipkan data sesuai dengan topic dimana data itu diterima, sehingga dapat bekerja untuk semua produk. Adapun implementasi fungsi `dataHandler` dapat dilihat dibawah

Fungsi menyisipkan data ke file database
<pre> import json import sqlite3 # SQLite DB Name DB_Name = "IoT.db" # Functions def slotHandler(jsonData, chargerPrefix, slotSuffix): jsonRows = json.loads(jsonData) chgStatus = jsonRows['Status'] battSN = jsonRows['Serial Number'] time = jsonRows['Time'] battSoC = jsonRows['SoC'] # Sisipkan data ke db dbObj = dbManager() tableName = chargerPrefix + "_" + slotSuffix sqliteQuery = "INSERT INTO " + tableName + " (Status, SN, Time, SoC) values (?, ?, ?, ?)" dbObj.insertRecord(sqliteQuery, [chgStatus, battSN, time, battSoC]) del dbObj print ("Inserted " + slotSuffix + " into " + tableName) print ("") </pre>

```

def deviceHandler(jsonData, chargerPrefix):
    jsonRows = json.loads(jsonData)
    chgMode = jsonRows['Charging Mode']
    locMCC = jsonRows['MCC']
    locMNC = jsonRows['MNC']
    locLAC = jsonRows['LAC']
    locCellID = jsonRows['Cell ID']

    # Sisipkan data ke db
    dbObj = dbManager()
    tableName = chargerPrefix + "_device"
    sqliteQuery = "INSERT INTO " + tableName + " (chargeMode, MCC, MNC, LAC, CellID) values (?, ?, ?, ?, ?)"
    dbObj.insertRecord(sqliteQuery, [chgMode, locMCC, locMNC, locLAC, locCellID])
    del dbObj

    print ("Inserted device info into " + tableName)
    print ("")

def dataHandler(Topic, jsonData):
    concatTopic = Topic.split("/")

    if concatTopic[2] == "s1" or concatTopic[2] == "s2":
        slotHandler(jsonData, concatTopic[1], concatTopic[2])
    elif concatTopic[2] == "device":
        deviceHandler(jsonData, concatTopic[1])

```

Fungsi dataHandler secara umum berperan untuk memindahkan informasi yang terdapat pada string yang memiliki struktur JSON, dan memasukkannya ke variabel masing masing, yang kemudian akan disisipkan ke database. Pada fungsi ini, digunakan class dbManager untuk merapikan struktur kode. Kedepannya, bagian kode lain juga akan menggunakan class ini agar struktur kode lebih seragam. Adapun class yang digunakan dapat dilihat dibawah ini.

Class yang digunakan untuk pengaksesan database

```

class dbManager():
    # load db, enable foreign keys, set cursor
    def __init__(self):
        self.conn = sqlite3.connect(DB_Name)
        self.conn.execute('PRAGMA foreign_keys = on')
        self.conn.commit()
        self.cur = self.conn.cursor()

    # commit data ke newline di db
    def insertRecord(self, query, values=()):

```

```

        self.cur.execute(query, values)
        self.conn.commit()
        return

    # close cursor, close db
    def __del__(self):
        self.cur.close()
        self.conn.close()

```

2.4.1.3.3 Inisialisasi Tabel

Sebelum subblok datalogger dapat memasukkan data ke tabel, perlu dilakukan inisialisasi tabel didalam database terlebih dahulu. Sementara ini, inisialisasi tabel dilakukan secara manual dengan mengeksekusi kode di cmd. Namun kedepannya akan diintegrasikan dengan *frontend* website agar penambahan produk dapat dilakukan dengan mudah. Kode yang digunakan dapat dilihat dibawah ini.

Fungsi inisialisasi tabel dalam suatu database

```

import sqlite3

# SQLite DB Name
DB_Name = "IoT.db"

tableS1 = "charger_s1"
tableS2 = "charger_s2"
tableDevice = "charger_device"

def createDB(s1Name, s2Name, deviceName):
    # SQLite DB Table Schema
    TableSchema=f"""
        drop table if exists {s1Name} ;
        create table {s1Name} (
            id integer primary key autoincrement,
            productId INTEGER,
            Status INTEGER,
            SN TEXT,
            Time TEXT,
            SoC INTEGER,
            SoH INTEGER,
            Volt REAL
        );

        drop table if exists {s2Name} ;
        create table {s2Name} (
            id integer primary key autoincrement,
            productId INTEGER,
            Status INTEGER,

```

```

SN TEXT,
Time TEXT,
SoC INTEGER,
SoH INTEGER,
Volt REAL
);

drop table if exists {deviceName} ;
create table {deviceName} (
id integer primary key autoincrement,
productId INTEGER,
chargeMode INTEGER,
MCC INTEGER,
MNC INTEGER,
LAC INTEGER,
CellID INTEGER
);
"""

#Connect or Create DB File
conn = sqlite3.connect(DB_Name)
curs = conn.cursor()

#Create Tables
sqlite3.complete_statement(TableSchema)
curs.executescript(TableSchema)

#Close DB
curs.close()
conn.close()

createDB(tableS1, tableS2, tableDevice)

```

2.4.1.3.4 Website

Website yang dirancang menggunakan Python Flask sebagai pemrosesan *backend*-nya. Pemrograman *frontend* dilakukan dengan html, js, dan css. Implementasi website pada saat ini sudah mencakup beberapa fitur, yakni pemantauan produk secara keseluruhan, pemantauan data satu produk, riwayat baterai yang pernah terhubung ke seluruh produk, menonaktifkan dan mengaktifkan produk secara remote, serta sistem user logon. Masing masing fitur ini akan dijelaskan pada beberapa subbab dibawah ini.

Pada beberapa subbab dibawah ini, hanya akan ditampilkan cuplikan kode terkait kode *backend* yang digunakan. Kode frontend html dinilai terlalu banyak untuk ditampilkan di dokumen ini. Kode frontend, beserta seluruh kode yang digunakan (baik di sisi client maupun server) nantinya dapat dilihat di *github repository* kelompok TA 024. Link untuk *repository* ini akan kami cantumkan di bagian lampiran.

2.4.1.3.4.1 Laman pemantauan produk secara keseluruhan

Pada laman ini, terdapat tabel yang menampilkan informasi terkini dari setiap produk yang terdapat di database. Kode *backend* yang digunakan untuk mengakses data ini dapat dilihat pada cuplikan dibawah ini.

```
Kode backend untuk memproses penampilan produk secara keseluruhan

@app.route("/home", methods=["POST", "GET"])
def home():
    if "username" in session:
        # request handle
        if request.method == "POST":
            # button deleteIndex
            if request.form['submit_button'] == 'deleteIndexSubmit':
                print("DEL")
                print("=====")
                deviceSelect = request.form["deleteIndex"]

                con = sqlite3.connect(dbName)
                cur = con.cursor()
                cur.execute("DELETE FROM charger_s1 WHERE productId=?",
(deviceSelect,))
                cur.execute("DELETE FROM charger_s2 WHERE productId=?",
(deviceSelect,))
                cur.execute("DELETE FROM charger_device WHERE
productId=?",
(deviceSelect,))
                con.commit()
                cur.close()
                print('index dropped')

                flash('Charger ' + deviceSelect + ' deleted', 'success')
                return redirect(url_for('home'))

            # button change on/off
            if request.form['submit_button'] == 'changeEnableSubmit':
                print("change status")
                print("=====")
                deviceSelect = request.form["changeEnableIndex"]

                con = sqlite3.connect(dbName)
                cur = con.cursor()
                # ['id', 'productId', 'productEnable']
                # 0      1          2
                cur.execute("SELECT * FROM charger_enable WHERE
productId=?",
(deviceSelect,))
                row = cur.fetchone()
```

```

        if row[2] == 1:
            setEn = 0
            strFlash = ' disabled'
        else:
            setEn = 1
            strFlash = ' enabled'

        query = "Update charger_enable set productEnable = ?
where productId = ?"
        cur.execute(query,[setEn,deviceSelect])
        con.commit()
        cur.close()
        print('info changed')

        # publish command
        try:
            commandTopic = "sys/charger" + str(deviceSelect) +
"/commands"
            publish.single(commandTopic, setEn,
hostname=mqttHost, port = mqttPort)

            flash('Charger ' + deviceSelect + strFlash,
'success')

        except:
            flash('MQTT server unreachable. Failed to change
device status', 'warning')

        return redirect(url_for('home'))

    else:
        pass

con = sqlite3.connect(dbName)
cur = con.cursor()

# source https://pagehalffull.wordpress.com/2012/11/14/python-
script-to-count-tables-columns-and-rows-in-sqlite-database/

# Get unique pids:
uniqueIDs1 = []
uniqueIDs2 = []
uniqueIDdev = []

for row in cur.execute("SELECT DISTINCT productId FROM charger_s1
ORDER BY productId" ):
    # print (row)

```

```

uniqueIDs1.append(row[0])

        for row in cur.execute("SELECT DISTINCT productId FROM charger_s2
ORDER BY productId" ):
            # print (row)
            uniqueIDs2.append(row[0])

        for row in cur.execute("SELECT DISTINCT productId FROM
charger_device ORDER BY productId" ):
            # print (row)
            uniqueIDdev.append(row[0])

combineIDs = uniqueIDs1 + uniqueIDs2 + uniqueIDdev
list_set = set(combineIDs)
uniqueIDs = (list(list_set))

print('uid')
print(uniqueIDs)

productCount = len(uniqueIDs)
print(productCount)
latestData = [ dict() for i in range(len(uniqueIDs)) ]

for i in range(len(uniqueIDs)):

    # cek apakah sudah ada data di charger_enable
    # kalau belum, insert data lalu inisialisasi dengan value 1
(ON)
    # kalau sudah, ambil info device_enable
    # ['id', 'productId', 'productEnable']
    # 0           1           2
    cur.execute('SELECT * FROM charger_enable WHERE productId =
?', [str(uniqueIDs[i]),])
    lastRow = cur.fetchone()

    if (lastRow == None):
        query = "INSERT INTO charger_enable (productId,
productEnable) values (?,?)"
        cur.execute(query, [uniqueIDs[i],1])
        con.commit()
        latestData[i]['enable'] = 1
    else:
        latestData[i]['enable'] = lastRow[2]

    latestData[i]['pid'] = uniqueIDs[i]
    # ambil latest row dari charger_s1 produk ke i
    # ['id', 'productId', 'Status', 'SN', 'Time', 'SoC', 'SoH',
'Volt']


```

```

#      0          1          2          3          4          5          6
7
        cur.execute('SELECT * FROM charger_s1 WHERE productId = ' +
str(uniqueIDs[i]) + ' ORDER BY id DESC LIMIT 1')
        lastRow = cur.fetchone()

        if (lastRow == None):
            latestData[i]['s1_stat'] = '-'
            latestData[i]['s1_soc'] = '-'
            latestData[i]['s1_latest'] = '-'

        else:
            latestData[i]['s1_stat'] = lastRow[2]
            latestData[i]['s1_soc'] = lastRow[5]
            latestData[i]['s1_latest'] = lastRow[4]

# ambil latest row dari charger_s2 produk ke i
# ['id', 'productId', 'Status', 'SN', 'Time', 'SoC', 'SoH',
'Volt']
#      0          1          2          3          4          5          6
7
        cur.execute('SELECT * FROM charger_s2 WHERE productId = ' +
str(uniqueIDs[i]) + ' ORDER BY id DESC LIMIT 1')
        lastRow = cur.fetchone()

        if (lastRow == None):
            latestData[i]['s2_stat'] = '-'
            latestData[i]['s2_soc'] = '-'
            latestData[i]['s2_latest'] = '-'

        else:
            latestData[i]['s2_stat'] = lastRow[2]
            latestData[i]['s2_soc'] = lastRow[5]
            latestData[i]['s2_latest'] = lastRow[4]

# ambil latest row dari charger_device produk ke i
# ['id', 'productId', 'chargeMode', 'MCC', 'MNC', 'LAC', 'ID']
#      0          1          2          3          4          5          6

        cur.execute('SELECT * FROM charger_device WHERE productId = ' +
+ str(uniqueIDs[i]) + ' ORDER BY id DESC LIMIT 1')
        lastRow = cur.fetchone()

        if (lastRow == None):
            latestData[i]['chgMode'] = '-'

        else:
            latestData[i]['chgMode'] = lastRow[2]

```

```

# proses tanggal
try:
    dateS1 = datetime.strptime(latestData[i]['s1_latest'],
'%y/%m/%d,%H:%M:%S')
    dateS2 = datetime.strptime(latestData[i]['s2_latest'],
'%y/%m/%d,%H:%M:%S')
    if (dateS1>dateS2):
        latestData[i]['latestSlot'] = 'S1'
        timestampStr = dateS1.strftime("%d-%B-%Y (%H:%M:%S)")

    else:
        latestData[i]['latestSlot'] = 'S2'
        timestampStr = dateS2.strftime("%d-%B-%Y (%H:%M:%S)")

    latestData[i]['latestTime'] = timestampStr

except:
    latestData[i]['latestSlot'] = '-'
    latestData[i]['latestTime'] = '-'


for x in latestData:
    print(x)

return render_template('displayGeneral.html',
latest_data=latestData, product_count=productCount)

else:
    return redirect(url_for('login'))

```

Kode frontend tidak ditampilkan pada bagian ini, namun dapat dilihat pada bagian lampiran.

2.4.1.3.4.2 Laman pemantauan data satu produk

Informasi yang ditampilkan pada web adalah informasi mengenai kondisi produk saat ini (kondisi slot 1, slot 2, mode pengisian daya yang digunakan, dan lokasi produk), serta riwayat kedua slot baterai. Proses *backend* untuk memproses webpage bagian ini dapat dilihat pada potongan kode dibawah.

Kode backend untuk memproses display satu produk pada laman web

```

@app.route('/devices/<deviceSelect>', methods=["POST", "GET"])
def display(deviceSelect):
    if "username" in session:
        if request.method == "POST":
            if request.form['submit_button'] == 'changeEnableSubmit':
                print("change status")

```

```

        print("====")
        deviceSelect = request.form["changeEnableIndex"]

        con = sqlite3.connect(dbName)
        cur = con.cursor()
        # ['id', 'productId', 'productEnable']
        # 0 1 2
        cur.execute("SELECT * FROM charger_enable WHERE
productId=? , (deviceSelect,))
        row = cur.fetchone()

        if row[2] == 1:
            setEn = 0
            strFlash = ' disabled'
        else:
            setEn = 1
            strFlash = ' enabled'

        # publish command
        try:
            commandTopic = "sys/charger" + str(deviceSelect) +
"/commands"
            publish.single(commandTopic, setEn, hostname =
mqttHost, port = mqttPort)

        except:
            flash('MQTT server unreachable. Can\'t change device
status', 'warning')
            params = "charger" + str(deviceSelect)
            return redirect(url_for('display',
deviceSelect=params))

            query = "Update charger_enable set productEnable = ?
where productId = ?"
            cur.execute(query,[setEn,deviceSelect])
            con.commit()
            cur.close()
            print('info changed')

            params = "charger" + str(deviceSelect)

            flash('Charger ' + deviceSelect + strFlash, 'success')
            return redirect(url_for('display', deviceSelect=params))

        else:

```

```

    pass

    con = sqlite3.connect(dbName)
    cur = con.cursor()
    # reading all table names
    table_list = [a for a in cur.execute("SELECT name FROM
sqlite_master WHERE type = 'table'")]

    deviceName = (deviceSelect[0:7] + " " +
deviceSelect[7:]).capitalize()
    deviceSelection = deviceSelect[7:]
    print("TEST:",end='')
    print(deviceSelection)
    querySelect = [ [0] for i in range (3) ]
    querySelect[0] = "charger_s1"
    querySelect[1] = "charger_s2"
    querySelect[2] = "charger_device"

    lenIndex = [ [0] for i in range(3) ]

    cur.execute("SELECT * FROM " + querySelect[0] + " WHERE productId
= ?", (deviceSelection,))
    lenIndex[0] = len(cur.fetchall())

    cur.execute("SELECT * FROM " + querySelect[1] + " WHERE productId
= ?", (deviceSelection,))
    lenIndex[1] = len(cur.fetchall())

    s1data = [ dict() for i in range(lenIndex[0]+1)]
    s2data = [ dict() for i in range(lenIndex[1]+1)]
    latestInfo = dict()

    # get charger_s1 data
    loopCount = 0
    # ['id', 'productId', 'Status', 'SN', 'Time', 'SoC', 'SoH',
    'Volt']
    # 0 1 2 3 4 5 6 7
    for row in cur.execute("SELECT * FROM " + querySelect[0] + "
WHERE productId = ? ORDER BY id DESC", (deviceSelection,)):
        if row[2] == -1:
            continue
        s1data[loopCount]['pid'] = row[1]
        s1data[loopCount]['stat'] = row[2]
        s1data[loopCount]['sn'] = row[3]
        s1data[loopCount]['time'] = row[4]
        s1data[loopCount]['soc'] = row[5]
        s1data[loopCount]['soh'] = row[6]
        s1data[loopCount]['volt'] = row[7]

```

```

        loopCount += 1

    # get charger_s2 data
    loopCount = 0
    # ['id', 'productId', 'Status', 'SN', 'Time', 'SoC', 'SoH',
'Volt']
    #      0         1         2         3         4         5         6         7
    for row in cur.execute("SELECT * FROM " + querySelect[1] + "
WHERE productId = ? ORDER BY id DESC", (deviceSelection,)):
        # print("HALO")
        # print(row)
        if row[2] == -1:
            continue
        s2data[loopCount]['pid'] = row[1]
        s2data[loopCount]['stat'] = row[2]
        s2data[loopCount]['sn'] = row[3]
        s2data[loopCount]['time'] = row[4]
        s2data[loopCount]['soc'] = row[5]
        s2data[loopCount]['soh'] = row[6]
        s2data[loopCount]['volt'] = row[7]

        # print(s2data[loopCount])
        loopCount += 1

    # get charger_device data
    # ['id', 'productId', 'chargeMode', 'MCC', 'MNC', 'LAC',
'CellID']
    #      0         1         2         3         4         5         6

    cur.execute('SELECT * FROM ' + querySelect[2] + " WHERE productId
= ? ORDER BY id DESC LIMIT 1", (deviceSelection,))
    row = cur.fetchone()
    print('latestinfo')
    latestInfo['pid'] = row[1]
    latestInfo['chgMode'] = row[2]
    latestInfo['mcc'] = row[3]
    latestInfo['mnc'] = row[4]
    latestInfo['lac'] = row[5]
    latestInfo['cid'] = row[6]

    print(latestInfo)

    # get charger status
    # cek apakah sudah ada data di charger_enable
    # kalau belum, insert data lalu inisialisasi dengan value 1 (ON)
    # kalau sudah, ambil info device_enable
    # ['id', 'productId', 'productEnable']

```

```

# 0      1      2
cur.execute('SELECT * FROM charger_enable WHERE productId = ?',
[str(deviceSelection),])
row = cur.fetchone()
deviceEn = row[2]

cur.close()
con.close()

# get latitude longitude from CGI with geolocation API
geoloc_url =
"https://www.googleapis.com/geolocation/v1/geolocate?key=AIzaSyD62AE0gl0C
jwL8dGwc14fLG5IoiwHHghw"
headers = CaseInsensitiveDict()
headers["Content-Type"] = "application/json"
mcc=latestInfo['mcc']
mnc=latestInfo['mnc']
lac=latestInfo['lac']
cid=latestInfo['cid']

api_payload = """
{
    "considerIp": true,
    "cellTowers": [
        {
            "cellId": %s,
            "locationAreaCode": %s,
            "mobileCountryCode": %s,
            "mobileNetworkCode": %s
        }
    ]
}
"""
%(cid, lac, mcc, mnc)

# default lat lon kalau geolocApi dicomment
lat= -6.8770328
lon= 107.6123459

# api request, comment dulu pas proses development
resp = requests.post(geoloc_url, headers=headers,
data=api_payload)
print(resp.text)

pythonObj = json.loads(resp.text)

lat = pythonObj['location']['lat']
lon = pythonObj['location']['lng']

```

```

        print(lat)
        print(lon)

        # print(s2data[0])
        return render_template('displayOneDevice.html', len=lenIndex,
s1_data=s1data, s2_data=s2data, device_info=latestInfo,
device_id=deviceSelection, lat=lat, lon=lon, device_en = deviceEn)

    else:
        return redirect(url_for('login'))

```

Secara umum, proses yang dilakukan pada *backend* bagian ini ialah memuat riwayat data slot 1 dan slot 2, serta data terakhir terkait lokasi dan mode pengisian daya. Data lokasi yang terdapat pada database produk berbentuk CGI, sehingga parameter CGI tersebut perlu dicari di database lokasi untuk mendapatkan koordinat latitude dan longitude. Setelah pencarian selesai, koordinat tersebut beserta riwayat data slot dan info pengisian daya dikirimkan ke file displayOneDevice.html untuk ditampilkan di website. Tampilan *frontend* dari kode html tersebut dapat dilihat pada Subbab 2.4.1.3.4.

2.4.1.3.4.3 Laman pemantauan riwayat setiap baterai yang terhubung

Kode backend untuk memproses rekап semua baterai yang pernah diisi dayanya
<pre> @app.route("/batteries", methods=["POST", "GET"]) def batteries(): if "username" in session: # request handle if request.method == "POST": pass else: pass con = sqlite3.connect(dbName) cur = con.cursor() # sauce https://pagehalffull.wordpress.com/2012/11/14/python- # script-to-count-tables-columns-and-rows-in-sqlite-database/ # Get unique pids: battSNs1 = [] battSNs2 = [] for row in cur.execute("SELECT DISTINCT SN FROM charger_s1"): # print (row) </pre>

```

        battSNs1.append(row[0])

    for row in cur.execute("SELECT DISTINCT SN FROM charger_s2 "):
        # print (row)
        battSNs2.append(row[0])

    combineSNs = battSNs1 + battSNs2
    list_set = set(combineSNs)
    uniqueSNs = (list(list_set))

    uniqueSNs = [x for x in uniqueSNs if x != '']

    print('usn')
    print(uniqueSNs)

    battCount = len(uniqueSNs)
    print(battCount)

    latestData = [ dict() for i in range(len(uniqueSNs)) ]

    for i in range(len(uniqueSNs)):

        latestData[i]['battSN'] = uniqueSNs[i]
        # ambil latest row dari charger_s1 produk ke i
        # ['id', 'productId', 'Status', 'SN', 'Time', 'SoC', 'SoH',
'Volt']
        #     0          1          2          3          4          5          6
        7
        cur.execute('SELECT * FROM charger_s1 WHERE SN = ? ORDER BY
id DESC LIMIT 1', (str(uniqueSNs[i]),))
        lastRow = cur.fetchone()

        if (lastRow == None):
            latestData[i]['s1_sn'] = '-'
            latestData[i]['s1_sn_latest'] = '-'
            latestData[i]['s1_sn_stat'] = '-'

        else:
            latestData[i]['s1_sn'] = lastRow[3]
            latestData[i]['s1_sn_latest'] = lastRow[4]
            latestData[i]['s1_sn_stat'] = lastRow[2]

        # ambil latest row dari charger_s2 produk ke i
        # ['id', 'productId', 'Status', 'SN', 'Time', 'SoC', 'SoH',
'Volt']
        #     0          1          2          3          4          5          6
        7

```

```

        cur.execute('SELECT * FROM charger_s2 WHERE SN = ? ORDER BY
id DESC LIMIT 1', (str(uniqueSNs[i]),))
        lastRow = cur.fetchone()

        if (lastRow == None):
            latestData[i]['s2_sn'] = '-'
            latestData[i]['s2_sn_latest'] = '-'
            latestData[i]['s2_sn_stat'] = '-'

        else:
            latestData[i]['s2_sn'] = lastRow[3]
            latestData[i]['s2_sn_latest'] = lastRow[4]
            latestData[i]['s2_sn_stat'] = lastRow[2]

        # proses slot mana yang lebih latest
        try:
            dateS1 = datetime.strptime(latestData[i]['s1_sn_latest'],
'%y/%m/%d,%H:%M:%S')
            dateS2 = datetime.strptime(latestData[i]['s1_sn_latest'],
'%y/%m/%d,%H:%M:%S')

            if (dateS1>dateS2):
                latestData[i]['battSN_latestSlot'] = 'S1'
                latestData[i]['battSN_status'] =
latestData[i]['s1_sn_stat']
                timestampStr = dateS1.strftime("%d-%B-%Y (%H:%M:%S)")

            else:
                latestData[i]['battSN_latestSlot'] = 'S2'
                latestData[i]['battSN_status'] =
latestData[i]['s2_sn_stat']
                timestampStr = dateS2.strftime("%d-%B-%Y (%H:%M:%S)")

            latestData[i]['battSN_latestTime'] = timestampStr

        #asumsi slot1 yang paling latest
        except:
            latestData[i]['battSN_latestSlot'] = 'S1'
            latestData[i]['battSN_latestTime'] = 'N/A'

        for x in latestData:
            print(x)

    return render_template('displayBatteries.html',
latest_data=latestData, product_count=battCount)

```

```

    else:
        return redirect(url_for('login'))

```

2.4.1.3.4.4 Laman pemantauan riwayat satu baterai

Kode backend untuk memproses riwayat satu baterai pada laman web

```

@app.route('/connectedBatteries/<battSN>')
def batteryList(battSN):
    if "username" in session:
        con = sqlite3.connect(dbName)
        cur = con.cursor()
        # reading all table names
        table_list = [a for a in cur.execute("SELECT name FROM
sqlite_master WHERE type = 'table'")]

        batteryName = "Battery #" + str(battSN)

        lenIndex = [ [0] for i in range(2)]

        cur.execute("SELECT * FROM charger_s1 WHERE SN = ?", (battSN,))
        lenIndex[0] = len(cur.fetchall())

        cur.execute("SELECT * FROM charger_s2 WHERE SN = ?", (battSN,))
        lenIndex[1] = len(cur.fetchall())

        s1data = [ dict() for i in range(lenIndex[0]+1)]
        s2data = [ dict() for i in range(lenIndex[1]+1)]
        battData = [ dict() for i in range(lenIndex[0]+lenIndex[1])]

        # get SN data in slot 1 (if available)
        loopCount = 0
        # ['id', 'productId', 'Status', 'SN', 'Time', 'SoC', 'SoH',
        'Volt', 'ChargeMode']
        # 0 1 2 3 4 5 6 7
        # 8
        for row in cur.execute("SELECT * FROM charger_s1 WHERE SN = ?",
        (battSN,)):
            s1data[loopCount]['stat'] = row[2]
            s1data[loopCount]['time'] = row[4]
            s1data[loopCount]['soc'] = row[5]
            s1data[loopCount]['soh'] = row[6]
            s1data[loopCount]['volt'] = row[7]
            s1data[loopCount]['chargeMode'] = row[8]

```

```

        loopCount += 1

    # get SN data in slot 2 (if available)
    loopCount = 0
    # ['id', 'productId', 'Status', 'SN', 'Time', 'SoC', 'SoH',
    'Volt', 'ChargeMode']
    #      0          1          2          3          4          5          6          7
    #      8
    for row in cur.execute("SELECT * FROM charger_s2 WHERE SN = ?", (battSN,)):
        s2data[loopCount]['stat'] = row[2]
        s2data[loopCount]['time'] = row[4]
        s2data[loopCount]['soc'] = row[5]
        s2data[loopCount]['soh'] = row[6]
        s2data[loopCount]['volt'] = row[7]
        s2data[loopCount]['chargeMode'] = row[8]

        # print(s2data[loopCount])
        loopCount += 1


validChargingCycle = 0
invalidChargingCycle = 0

for i in s1data:
    print(i)

# process data valid per 1 charging cycle di slot 1
for i in range(lenIndex[0]-1):
    space = 1
    # syarat kondisi 1 charging cycle
    # setelah stat nya 1, setelahnya harus langsung 2 atau 1
    if s1data[i]['stat'] == 1:
        # kalau 1, berarti lagi ngubah mode pengisian daya, jadi
        ditraverse sampai ketemu 2
        if s1data[i+1]['stat'] == 1:
            chargeCondition = 'Partial'
        elif s1data[i+1]['stat'] == 2:
            chargeCondition = 'Full'
        elif s1data[i+1]['stat'] == 0:
            chargeCondition = 'Partial'
        elif s1data[i+1]['stat'] == -1:
            invalidChargingCycle+=1
            continue

            battData[validChargingCycle]['condition'] =
chargeCondition

```

```

        battData[validChargingCycle]['init_soc'] =
s1data[i]['soc']
        battData[validChargingCycle]['init_soh'] =
s1data[i]['soh']
        battData[validChargingCycle]['init_volt'] =
s1data[i]['volt']
        battData[validChargingCycle]['init_time'] =
s1data[i]['time']
        battData[validChargingCycle]['chg_mode'] =
s1data[i]['chargeMode']
        battData[validChargingCycle]['fin_soc'] =
s1data[i+space]['soc']
        battData[validChargingCycle]['fin_soh'] =
s1data[i+space]['soh']
        battData[validChargingCycle]['fin_volt'] =
s1data[i+space]['volt']
        battData[validChargingCycle]['fin_time'] =
s1data[i+space]['time']
        validChargingCycle += 1

# process data valid per 1 charging cycle di slot 2
for i in range(lenIndex[1]-1):
    exit = 0
    space = 1
    # syarat kondisi 1 charging cycle
    # setelah stat nya 1, setelahnya harus langsung 2 atau 1
    if s2data[i]['stat'] == 1:

        if s2data[i+1]['stat'] == 1:
            chargeCondition = 'Partial'
        elif s2data[i+1]['stat'] == 2:
            chargeCondition = 'Full'
        elif s2data[i+1]['stat'] == 0:
            chargeCondition = 'Partial'
        elif s2data[i+1]['stat'] == -1:
            invalidChargingCycle+=1
            continue

        battData[validChargingCycle]['condition'] =
chargeCondition
        battData[validChargingCycle]['init_soc'] =
s2data[i]['soc']
        battData[validChargingCycle]['init_soh'] =
s2data[i]['soh']
        battData[validChargingCycle]['init_volt'] =
s2data[i]['volt']
        battData[validChargingCycle]['init_time'] =
s2data[i]['time']

```

```

        battData[validChargingCycle]['chg_mode'] =
s2data[i]['chargeMode']
        battData[validChargingCycle]['fin_soc'] =
s2data[i+space]['soc']
        battData[validChargingCycle]['fin_soh'] =
s2data[i+space]['soh']
        battData[validChargingCycle]['fin_volt'] =
s2data[i+space]['volt']
        battData[validChargingCycle]['fin_time'] =
s2data[i+space]['time']
        validChargingCycle += 1

# print(battData[0])
# proses tanggal, durasi, delta SOC
for i in range(validChargingCycle):
    battData[i]['delta_soc'] = battData[i]['fin_soc'] -
battData[i]['init_soc']
    try:
        dateStart = datetime.strptime(battData[i]['init_time'],
'%y/%m/%d,%H:%M:%S')
        dateFinish = datetime.strptime(battData[i]['fin_time'],
'%y/%m/%d,%H:%M:%S')

        duration = (dateFinish - dateStart).total_seconds()
        if (duration < 3600):
            durationString = "{:.0f} mins".format((duration %
3600) / 60)
        else:
            durationString = "{} hrs {:.0f}"
            mins".format(math.floor(duration / 3600), (duration % 3600) / 60)
        battData[i]['delta_time'] = durationString

        battData[i]['init_time'] = dateStart.strftime("%d-%B-%Y
(%H:%M:%S)")
        battData[i]['fin_time'] = dateFinish.strftime("%d-%B-%Y
(%H:%M:%S)")

    except:
        battData[i]['init_time'] = '-'
        battData[i]['fin_time'] = '-'
        battData[i]['delta_time'] = '-'

    cur.close()
    con.close()

print("inv:{}".format(invalidChargingCycle))

```

```

        return render_template('displayOneBattery.html',
lenValid=validChargingCycle, lenInvalid=invalidChargingCycle,
batt_data=battData, batt_sn=battSN)

else:
    return redirect(url_for('login'))

```

2.4.1.3.4.5 Laman sistem user logon

Kode backend untuk laman login
<pre> @app.route("/login", methods=["POST", "GET"]) def login(): if request.method == "POST": if request.form['submit_btn'] == 'login': uname = request.form["username"] pword = request.form["password"] print(uname) print(pword) con = sqlite3.connect(dbUser) cur = con.cursor() # row contents : id, fullname, username, pass(hash), salt(btn) # index : 0 1 2 3 4 cur.execute("SELECT * FROM userinfo WHERE username = ?", (uname,)) row = cur.fetchone() if (row == None): flash('Invalid username / password. Please try again', 'danger') else: iter_num = 100000 salt = row[4] key = hashlib.pbkdf2_hmac('sha256', pword.encode('utf-8'), salt, iter_num) </pre>

```

        if (key == row[3]):
            # flash('Login successful', 'success')
            print("PASSWORD MATCH")
            session['username'] = uname
            # session['fullname'] = fname
            # session['role']      = urole

            return redirect(url_for('home'))

        else:
            flash('Invalid username / password. Please try again', 'danger')
            print("invalid uname.pw")
            return redirect(url_for('login'))

    return redirect(url_for('login'))

elif request.form['submit_btn'] == 'register':
    print("REGIST")

    flash('Cant register here', 'danger')
    return redirect(url_for('login'))

else:
    pass

return render_template('login.html')

```

2.4.2 Pengujian

2.4.2.1 Client-side data publishing

Pertama, dilakukan pengujian terhadap kapabilitas dari produk untuk mem-*publish* data ke beberapa topic tertentu ke broker MQTT. Dalam pengujian, digunakan beberapa push button sebagai simulasi baterai dipasang atau dicabut, serta simulasi pengubahan mode pengisian daya. Selain itu, untuk mempermudah implementasi subsistem ini, seluruh data yang dikirimkan oleh produk masih merupakan *mock data*.

```

C:\Windows\System32\cmd.exe - mosquitto -c mosquitto.conf -v
Microsoft Windows [Version 10.0.19043.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32\mosquitto -c mosquitto.conf -v
1647272596: mosquitto version 2.0.14 starting
1647272596: Config loaded from mosquitto.conf.
1647272596: Opening ipv4 listen socket on port 1883.
1647272596: Opening ipv4 listen socket on port 1883.
1647272596: mosquitto version 2.0.14 running
1647272598: New connection from 192.168.1.105:53282 on port 1883.
1647272598: New client connected from 192.168.1.105:53282 as ESP32Client-4484 (p2, c1, k15).
1647272598: No will message specified
1647272598: Publishing auto\_30082570\_5683\_7C5B\_C689\_47E9A56A359F to client-4484 (0, 0)
1647272613: Received PINGREQ from ESP32Client-4484
1647272613: Sending PINGRESP to ESP32Client-4484
1647272613: Received PUBLISH from ESP32Client-4484 (d0, q0, r0, m0, 'sys/charger1/s1', ... (90 bytes))
1647272624: Received PUBLISH from ESP32Client-4484 (d0, q0, r0, m0, 'sys/charger1/s2', ... (90 bytes))
1647272628: Received PUBLISH from ESP32Client-4484 (d0, q0, r0, m0, 'sys/charger1/device', ... (73 bytes))
1647272628: Received PINGREQ from ESP32Client-4484
1647272628: Sending PINGRESP to ESP32Client-4484

```

Gambar 2.4.5 Pembacaan topic yang dipublish oleh Charger1 (kanan) ke broker (kiri)

Pada Gambar 2.4.5 terlihat pada sisi kanan bahwa saat tombol pada produk ditekan, pada serial monitor tertulis kata HALO (penanda button ditekan) dan dilanjutkan dengan string yang sudah diformat dalam bentuk JSON. Kemudian di sisi kanan, terlihat bahwa broker menerima data yang dipublish oleh produk pada ketiga topic yang dikirimkan. Setelah pengujian tahap ini selesai, dilanjutkan ke pengujian program datalogger.

2.4.2.2 Server-side data subscribing

```

1647272673: Received PINGREQ from ESP32Client-4484
1647272673: Sending PINGRESP to ESP32Client-4484
1647272675: Received PUBLISH From ESP32Client-4484 (d0, q0, r0, m0, 'sys/charger1/s2', ... (90 bytes))
1647272675: Sending PUBLISH to auto-30082570-5683-7C5B-C689-47E9A56A359F (d0, q0, r0, m0, 'sys/charger1/s2', ... (90 bytes))
1647272676: Received PUBLISH From ESP32Client-4484 (d0, q0, r0, m0, 'sys/charger1/s1', ... (90 bytes))
1647272677: Sending PUBLISH to auto-30082570-5683-7C5B-C689-47E9A56A359F (d0, q0, r0, m0, 'sys/charger1/s1', ... (90 bytes))
1647272678: Received PUBLISH From ESP32Client-4484 (d0, q0, r0, m0, 'sys/charger1/s1', ... (90 bytes))
1647272678: Sending PUBLISH to auto-30082570-5683-7C5B-C689-47E9A56A359F (d0, q0, r0, m0, 'sys/charger1/s1', ... (90 bytes))
1647272678: Received PUBLISH From ESP32Client-4484 (d0, q0, r0, m0, 'sys/charger1/s2', ... (90 bytes))
1647272678: Sending PUBLISH to auto-30082570-5683-7C5B-C689-47E9A56A359F (d0, q0, r0, m0, 'sys/charger1/s2', ... (90 bytes))
1647272678: Received PUBLISH From ESP32Client-4484 (d0, q0, r0, m0, 'sys/charger1/device', ... (73 bytes))
1647272678: Received PUBLISH From ESP32Client-4484 (d0, q0, r0, m0, 'sys/charger1/device', ... (73 bytes))
1647272683: Received PINGREQ from ESP32Client-4484
1647272689: Sending PINGRESP to ESP32Client-4484
1647272689: Received PINGREQ from auto-30082570-5683-7C5B-C689-47E9A56A359F
1647272689: Sending PINGRESP to auto-30082570-5683-7C5B-C689-47E9A56A359F

```

```

Inserted s2 into charger1_s2
MQTT Data Received...
MQTT Topic: sys/charger1/s1
Data: b'{"status": 1, "Serial Number": "0123456789ABCDEF", "Time": "10-Mar-2022 14:59", "SoC": 75}'
Inserted s1 into charger1_s1

MQTT Data Received...
MQTT Topic: sys/charger1/s2
Data: b'{"status": 0, "Serial Number": "0123456789ABCDEF", "Time": "10-Mar-2022 14:59", "SoC": 75}'
Inserted s1 into charger1_s2

MQTT Data Received...
MQTT Topic: sys/charger1/device
Data: b'{"Charging Mode": 0, "MCC": 510, "MNC": 1, "LAC": 9147, "Cell ID": 36665}'
Inserted device info into charger1_device

```

Gambar 2.4.6 Pengujian subscribe ke topic (kiri atas) dan penyimpanan ke database (kiri bawah) oleh Charger1 (kanan)

Pada bagian kiri bawah dari Gambar 2.4.6, dapat dilihat output konsol yang menjalankan kode datalogger. Terlihat bahwa datalogger berhasil menerima string yang dikirim oleh produk melalui subscribe, dan juga berhasil memasukkan data ke database. Verifikasi insersi ke database dapat dilihat lebih lanjut pada Gambar 2.4.7

```

C:\Windows\System32\cmd.exe
C:\Users\andre\Desktop\mqtt\python code>python opendb.py
Slot 1
[{'id': 'Status', 'SN': 'Time', 'SoC'}]
(1, 0, '0123456789ABCDEF', '10-Mar-2022 14:59', 75)
(2, 1, '0123456789ABCDEF', '10-Mar-2022 14:59', 75)
(3, 0, '0123456789ABCDEF', '10-Mar-2022 14:59', 75)
(4, 1, '0123456789ABCDEF', '10-Mar-2022 14:59', 75)
(5, 0, '0123456789ABCDEF', '10-Mar-2022 14:59', 75)
(6, 1, '0123456789ABCDEF', '10-Mar-2022 14:59', 75)
(7, 0, '0123456789ABCDEF', '10-Mar-2022 14:59', 75)

Slot 2
[{'id': 'Status', 'SN': 'Time', 'SoC'}]
(1, 1, 'ABCDEF9876543210', '10-Mar-2022 14:59', 55)
(2, 0, 'ABCDEF9876543210', '10-Mar-2022 14:59', 55)
(3, 1, 'ABCDEF9876543210', '10-Mar-2022 14:59', 55)
(4, 0, 'ABCDEF9876543210', '10-Mar-2022 14:59', 55)
(5, 0, 'ABCDEF9876543210', '10-Mar-2022 14:59', 55)
(6, 1, 'ABCDEF9876543210', '10-Mar-2022 14:59', 55)

Device
[{'id': 'chargeMode', 'MCC': 'MNC', 'LAC', 'CellID'}]
(1, 1, 510, 11, 35117, 593639)
(2, 0, 510, 11, 35117, 593639)
(3, 0, 510, 1, 9147, 36665)
(4, 1, 510, 1, 9147, 36665)
(5, 0, 510, 1, 9147, 36665)

C:\Users\andre\Desktop\mqtt\python code>

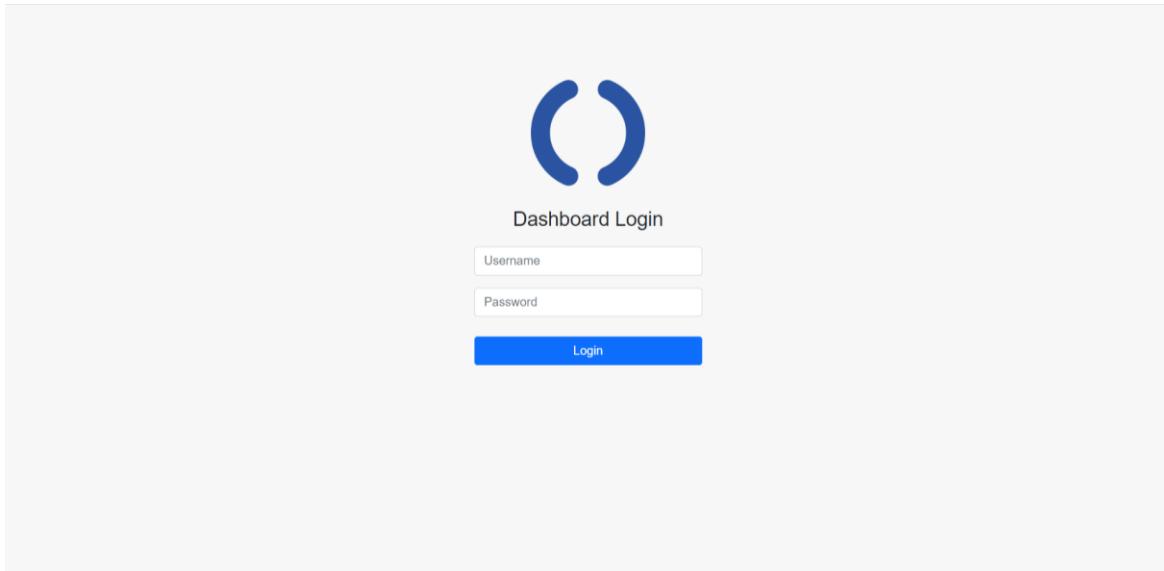
```

Gambar 2.4.7 Isi database untuk Charger 1

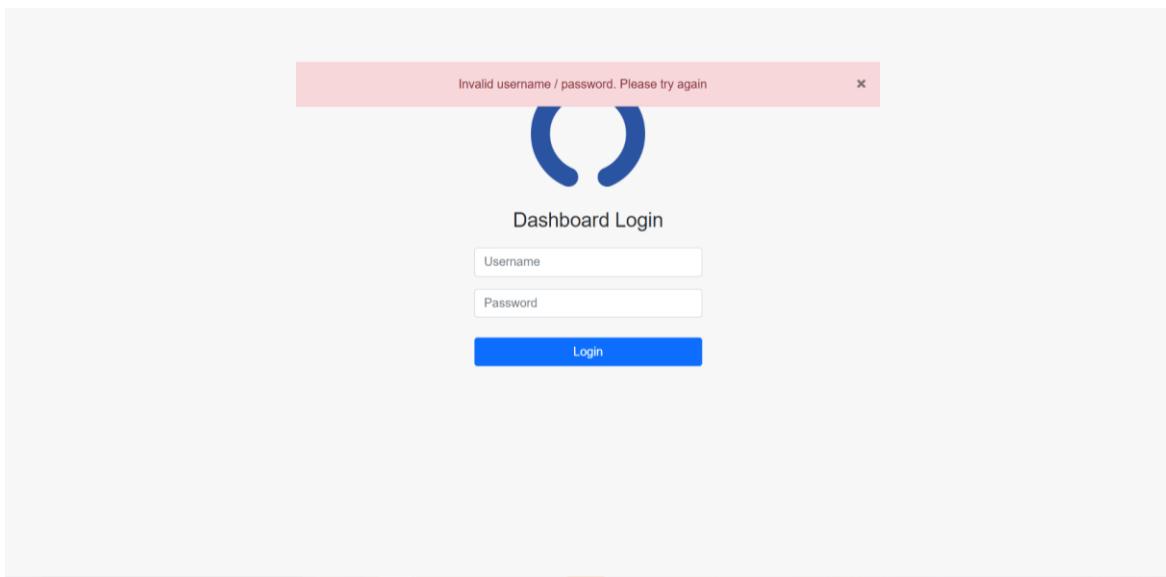
2.4.2.3 Web dashboard

Dibawah ini dapat dilihat beberapa cuplikan laman yang terdapat pada web dashboard yang telah dirancang

2.4.2.3.1 Laman Login



Gambar 2.4.8 Laman login



Gambar 2.4.9 Tampilan laman saat username / password tidak valid

Pada Gambar 2.4.9, dapat dilihat tampilan laman apabila dimasukkan user info yang tidak valid. Laman menampilkan alert bahwa credentials invalid, dan pengguna harus login terlebih dahulu untuk mengakses website. Dari gambar tersebut, disimpulkan bahwa sistem login berhasil diimplementasikan dengan baik.

2.4.2.3.2 Laman Tampilan Produk

The screenshot shows a "Product Dashboard" table with three entries. The columns are labeled: ID, S1 Status, S2 SOC, S2 Status, S2 SOC, Last Updated, Charge Mode, and Actions. The entries are:

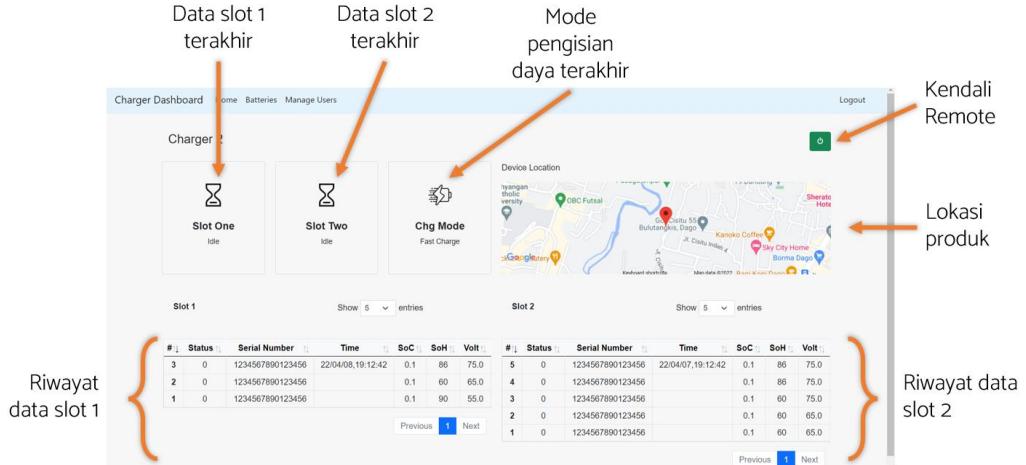
ID	S1 Status	S2 SOC	S2 Status	S2 SOC	Last Updated	Charge Mode	Actions
2	Idle	0.1	Idle	0.1	08-April-2022 (19:12:42) [S1]	Fast Charging	
3	Idle	0	Idle	0	N/A	Fast Charging	
4	N/A	-	N/A	-	N/A	Fast Charging	

Annotations on the right side of the table:

- A blue arrow points from "Riwayat Satu Client" to the "Logout" link at the top right.
- A green arrow points from "Kendali Remote" to the "Edit" icon in the first row's Actions column.
- A red arrow points from "Hapus Riwayat" to the "Delete" icon in the third row's Actions column.
- An orange arrow points from "Data terkini dari tiap sistem" to the bottom of the table.
- Links at the bottom right include "Previous" and "Next".

Gambar 2.4.10 Tampilan laman utama dari dashboard

Pada Gambar 2.4.9, dapat dilihat laman untuk menampilkan informasi produk secara umum. Terdapat entry terakhir dari tiap produk, dan juga terdapat beberapa opsi untuk melihat detail produk secara rinci, opsi untuk mematikan atau menyalaikan produk, serta opsi untuk mendelete seluruh entry produk. Tampilan laman yang menampilkan detail produk dapat dilihat pada Gambar 2.4.11 dibawah ini



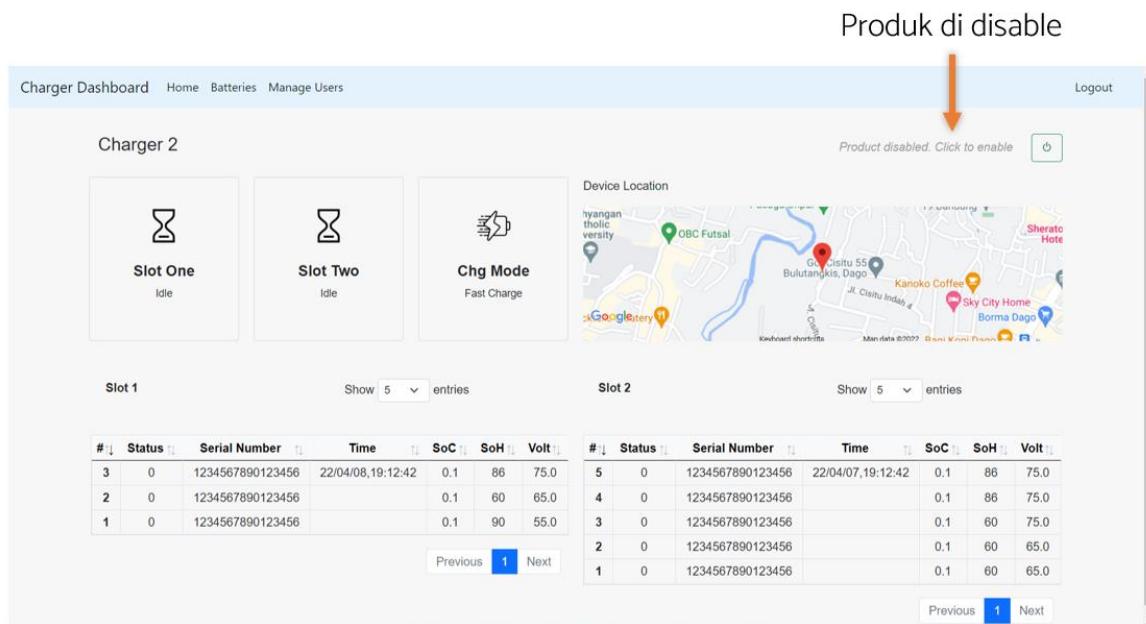
Gambar 2.4.11 Tampilan laman untuk menyajikan data produk Charger 1

Pada laman informasi detail produk, terdapat riwayat slot 1, riwayat slot 2, kondisi slot 1 dan 2, serta mode pengisian daya yang digunakan oleh Charger 1. Selain itu juga terdapat peta yang menunjukkan lokasi penggunaan produk. Perlu diingat bahwa lokasi produk didapatkan bukan menggunakan modul GPS, melainkan menggunakan CGI yang memiliki akurasi yang rendah, sehingga titik yang diberikan tidak sepenuhnya akurat (pada kondisi ekstrim, perbedaan titik lokasi dapat mencapai 1000 m).

Dari pemaparan diatas, dapat dikatakan bahwa fungsionalitas datalogging sudah diimplementasikan dengan baik. Pada dokumen selanjutnya (B500), fungsionalitas ini akan diuji kembali untuk menampilkan data asli yang dikirimkan oleh produk.

2.4.2.3.3 Fungsionalitas pemberian perintah ON/OFF

Kemudian, tombol yang terdapat pada sisi kanan atas dari Gambar 2.4.13 berperan sebagai tombol untuk mematikan atau menyalakan produk secara remote. Bila tombol ditekan, maka server akan mengirimkan perintah untuk mematikan atau menyalakan produk (sesuai dengan kondisi tombol). Pada Gambar 2.4.13, tombol terlihat memiliki warna hijau, yang menandakan bahwa produk masih dalam kondisi ON. Bila tombol ditekan, kondisi produk akan berubah menjadi OFF, seperti yang dapat dilihat pada Gambar 2.4.12. Pada gambar tersebut, terlihat tombol berubah menjadi warna putih dengan *outline* berwarna hijau, yang menandakan bahwa produk sedang dalam kondisi OFF. Untuk mengurangi kekeliruan, terdapat juga keterangan berupa tulisan di sisi kiri tombol yang memberi tahu pengguna bahwa produk sedang dalam kondisi OFF.



Gambar 2.4.12 Perubahan pada tombol ON/OFF apabila ditekan

Dari perubahan tersebut, dapat dikatakan bahwa fungsionalitas ON/OFF dari sisi server sudah diimplementasikan dengan baik. Pada dokumen selanjutnya (B500), baru akan diujicobakan fungsionalitas ini untuk mengontrol kondisi produk.

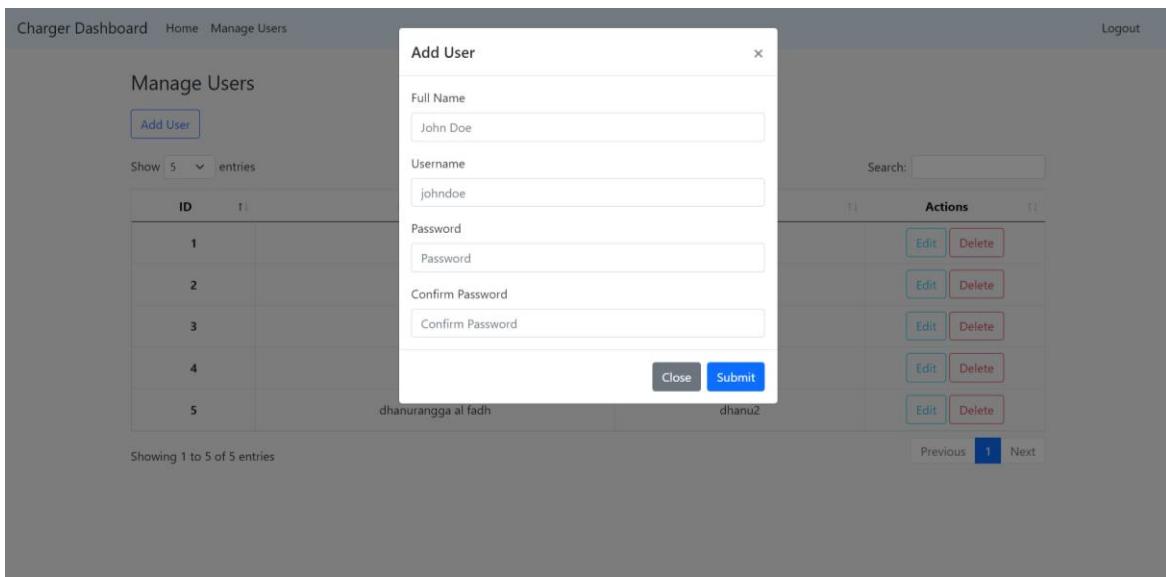
2.4.2.3.4 Fungsionalitas user management

The screenshot shows the 'Manage Users' section of the dashboard. It includes a table of users with columns for ID, Full Name, Username, and Actions (Edit and Delete buttons). A blue arrow points to the 'Add User' button, and another blue arrow points to the 'Edit User' button. A red arrow points to the 'Delete User' button. The table data is as follows:

ID	Full Name	Username	Actions
1	None	admin	Edit Delete
2	None	dhanu	Edit Delete
3	None	naoko	Edit Delete
4	ad	ad	Edit Delete
5	dhanurangga fadh	dhanu2	Edit Delete

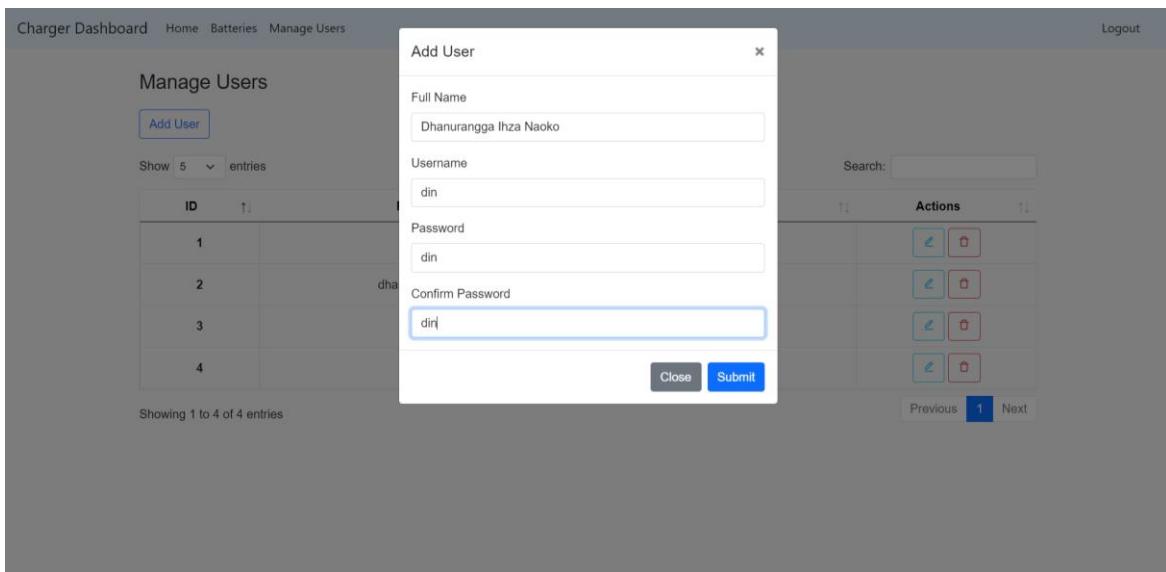
Gambar 2.4.13 Tampilan laman untuk mengedit profil user

Pada Gambar 2.4.13 dapat dilihat tampilan web untuk mengatur user. Terdapat opsi untuk mengubah atau menghapus user. Untuk menambah user, terdapat tombol "add user" di bagian kiri atas laman. Apabila ditekan, muncul pop-up window seperti pada Gambar 2.4.14



Gambar 2.4.14 Tampilan laman saat ingin menambah user baru

Bila beberapa kolom tersebut diisi dengan data seperti pada Gambar 2.4.15, maka akan muncul entry baru yang menampilkan full name dan username dari data yang telah diisi. Entry baru yang muncul dapat dilihat lebih lanjut pada Gambar 2.4.16

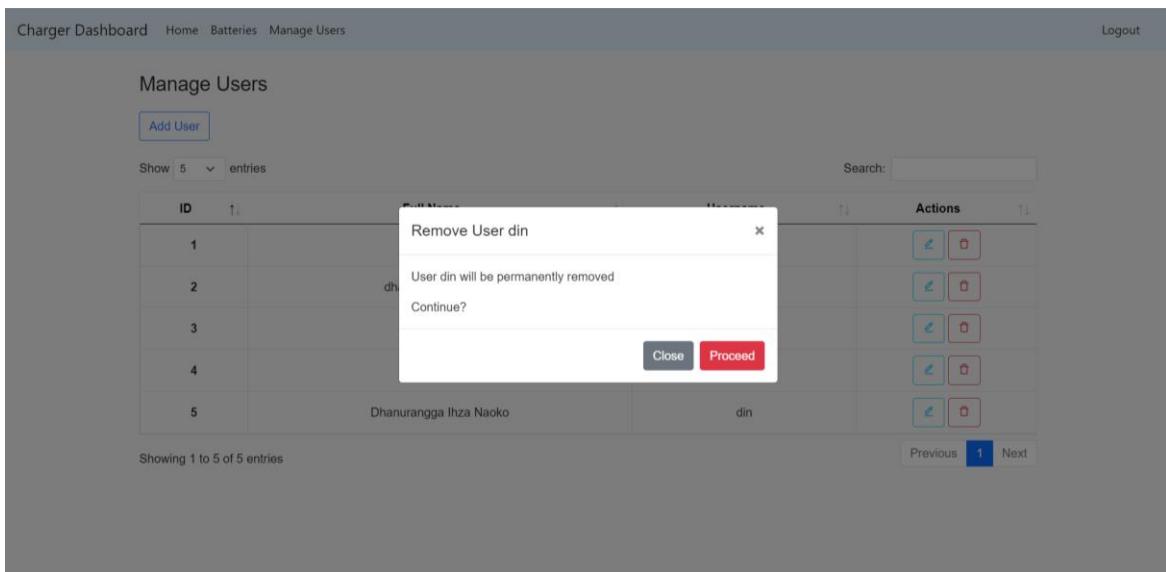


Gambar 2.4.15 Contoh pengisian form add user

ID	Full Name	Username	Actions
1	ahab	naoko	
2	dhanurangga al fadh	dhanu2	
3	ivan	ad	
4	admin	admin	
5	Dhanurangga Ihza Naoko	din	

Gambar 2.4.16 Tampilan saat registrasi berhasil dilakukan

Selanjutnya, bila kolom paling bawah ingin dihapus, dapat ditekan tombol dibagian kanan yang memiliki logo tempat sampah berwarna merah. Nantinya, akan muncul popup confirmation terkait data yang ingin dihapus seperti pada Gambar 2.4.17. Setelah konfirmasi dilakukan, maka data user akan hilang, dan muncul alert disisi atas laman bahwa user sudah dihapus, seperti yang dapat dilihat pada Gambar 2.4.18



Gambar 2.4.17 Pop-up window konfirmasi delete user

ID	Full Name	Username	Actions
1	ahab	naoko	
2	dhanurangga al fadh	dhanu2	
3	ivan	ad	
4	admin	admin	

Gambar 2.4.18 User berhasil didelete

Dari pemaparan diatas, dapat dikatakan bahwa fungsionalitas user management sudah diimplementasikan dengan baik.

2.4.2.3.5 Fungsionalitas data processing

Pada bagian ini, akan ditelaah mengenai fitur pemrosesan riwayat baterai yang pernah diisi dayanya di sistem produk yang telah dirancang. Terlepas dari posisi slot dan ID produk, informasi mengenai riwayat pengisian daya baterai dapat diakses di dashboard. Untuk mengakses informasi riwayat ini, bagian navbar yang bertuliskan “Batteries” dapat ditekan, dan pengguna akan dibawa ke laman rekап baterai. Laman rekап baterai ini dapat dilihat pada

#	Serial Number	Status	Latest Updated	Actions
1	860548049322453	N/A	01-January-2004 (00:07:23) (S2)	
2	1234567890123456	Idle	08-April-2022 (19:12:42) (S2)	

Gambar 2.4.19 Tampilan rekап riwayat baterai

Apabila tombol yang ber-outline biru di sebelah kanan diklik, maka akan terlihat riwayat historis dari pengisian daya baterai dengan nomor seri yang bersangkutan. Riwayat tersebut dapat dilihat pada gambar dibawah ini

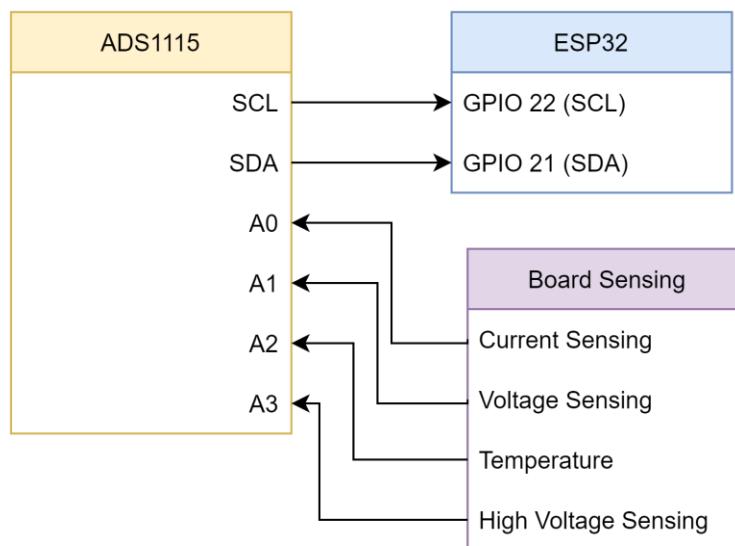
Battery #860548049348987																									
Initial charge													Finish charge												
#	Start Time	SOC	SOH	Volt	Duration	ΔSOC	ChgMode	Condition	#	Start Time	SOC	SOH	Volt	Duration	ΔSOC	ChgMode	Condition								
3	16-May-2022 (21:33:44)	81	88.3	1.02	16-May-2022 (21:38:35)	100	88.3	1.02	5 mins	19	Fast	Full													
2	16-May-2022 (21:31:26)	81	88.3	0.1	16-May-2022 (21:33:44)	81	88.3	1.02	2 mins	-	Fast	Partial													
1	16-May-2022 (21:32:10)	81	88.3	61.4	16-May-2022 (21:31:26)	81	88.3	0.1	59 mins	-	Slow	Partial													

Gambar 2.4.20 Tampilan riwayat historis satu baterai

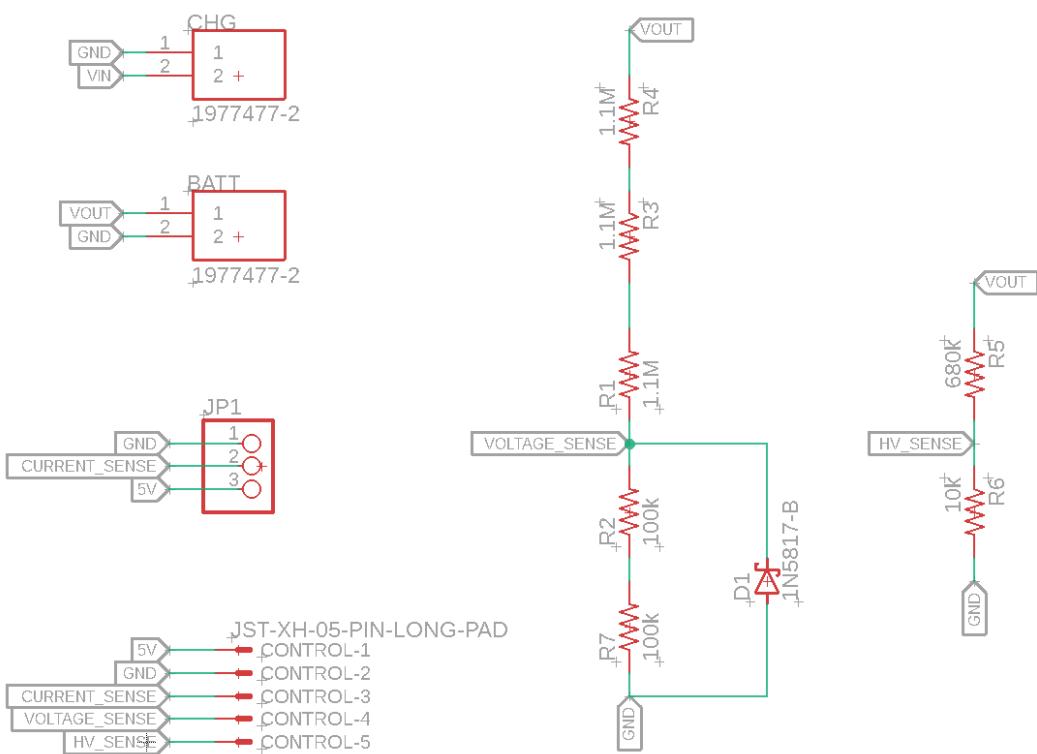
Pada Gambar 2.4.20, terlihat beberapa parameter yang disimpan dalam riwayat pengisian daya baterai, yakni waktu, SOC, SOH, tegangan saat awal dan akhir, durasi pengisian daya, perubahan SOC, charging mode yang digunakan, serta kondisi pengisian daya. Secara khusus, kolom kondisi (Condition) menunjukkan apakah baterai tersebut sudah diisi penuh sebelum dicabut. Bila pengisian daya dibiarkan sampai penuh, maka pada condition akan tertulis “Full”. Sebaliknya bila dicabut sebelum penuh, maka pada condition akan tertulis “Partial”

2.5 Sub-Sistem Sensing

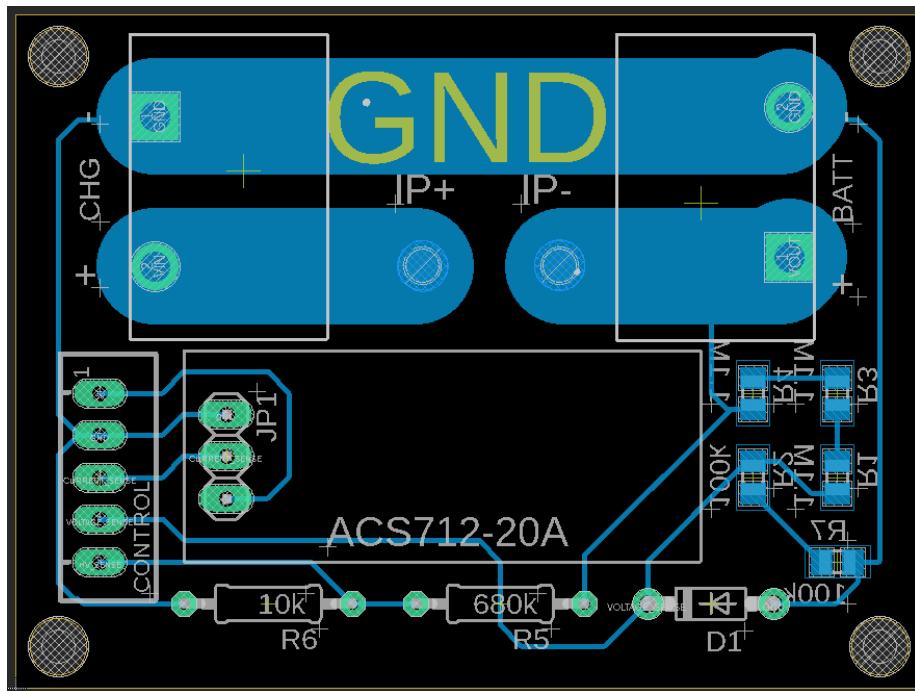
2.5.1 Implementasi



Gambar 2.5.1 Pin Mapping Subsistem Sensing



Gambar 2.5.2 Skematik dari subsistem sensing



Gambar 2.5.3 Hasil perancangan board subsistem sensing

Subsistem sensing ini terdiri dari sensing arus, tegangan, dan suhu. Lalu hasil keluaran analog dari sensor tersebut diolah menjadi digital menggunakan ADS1115 . Sensing suhu dilakukan pada board charger untuk mengetahui suhu pada mosfet. Sensing temperatur tersebut dilakukan menggunakan LM35. Untuk mendapatkan arus digunakan sensor arus ACS712 dengan maksimal pembacaan arus 20A dan resolusi 100mV/A. pembacaan tegangan dilakukan menggunakan pembagi tegangan dengan nilai resistansi yang sangat besar dengan orde $M\Omega$. Pembacaan sensor arus dan tegangan ditempatkan pada board tersendiri antara output subsistem charger dan baterai. Output sensor arus dan tegangan tersebut kemudian dikirimkan ke board kontrol lalu diolah menjadi digital menggunakan ADS1115. Board yang digunakan untuk sensor arus dan tegangan dapat dilihat pada Gambar 2.5.3, dan komponen yang diperlukan untuk board tersebut dapat dilihat pada Tabel 2.5.1

Tabel 2.5.1 Daftar komponen untuk board Sensing

Kategori	Komponen	Jumlah
Divider	R 0805 100k 0.1%	1
	R 0805 1.1M 0.1%	3
Current	ACS 712 20A	1
Lain lain	Female Header 1x3	1
	Terminal Block	2
	JST XH4	1

Di bawah ini merupakan implementasi kode subsistem kontrol untuk mendapatkan informasi arus, tegangan, dan temperatur dari subsistem sensing. Implementasi perangkat lunak menggunakan *task* baru pada FreeRTOS agar sampling dapat dilakukan sepanjang sistem berjalan. Sampling dilakukan setiap interval 3 ms. Selain mengukur tegangan, arus, dan temperatur, pada *task* ini juga di-*sampling* status tombol *trigger*, dan status mode daya.

```

void Sampling(void *parameter)
{
    for (;;)
    {
        int triggerDetected;
        if (millis() - t > 2000){
            t = millis();
        }

        // button ubah mode daya
        byte btnSwitchPowerMode = digitalRead(chargingModePin);
        chargingMode = btnSwitchPowerMode;

        // Deteksi trigger button
        triggerDetected = !(digitalRead(triggerButton));
        if( triggerDetected && stateL == IDLE && stateR == IDLE){
            triggerFlag = 1;
        }

        // Membaca arus
        readAdcCurrent = ads1015.readADC_SingleEnded(0);
        current = ((float( (readAdcCurrent*3 + driftVoltage) ) - float(qov)) ) / 0.093 ;

        // Membaca tegangan
        readAdcVoltage = ads1015.readADC_SingleEnded(1);
        voltage = readAdcVoltage;
        voltage = float( readAdcVoltage*3 );
        voltage *= 34;

        // Membaca temperatur
        readAdcTemperature = ads1015.readADC_SingleEnded(2);

        adc_ads1[0] = ads1015.readADC_SingleEnded(0);
        adc_ads1[1] = ads1015.readADC_SingleEnded(1);
        adc_ads1[2] = ads1015.readADC_SingleEnded(2);
        adc_ads1[3] = ads1015.readADC_SingleEnded(3);

        temperature = (float( readAdcTemperature*3 )/7.2) - 8.5;

        if(temperature > 67){
            Serial.println("OVERHEAT DETECTED");
            overheatFlag = 1;
        }

        if (overheatFlag == 1 && temperature < 57){
            Serial.println("Resume normal operation");
            overheatFlag = 0;
        }

        if(current > 15000){
            Serial.println("OVERCURRENT DETECTED");
            overcurrentFlag = 1;
        }
    }
}

```

```

        t_overcurrentFlag = millis();
    }
    if (overcurrentFlag == 1 && millis() - t_overcurrentFlag >
30000) {
        Serial.println("Resume normal operation");
        overcurrentFlag = 0;
    }

    vTaskDelay(3 / portTICK_PERIOD_MS);
}
}

```

2.5.2 Pengujian

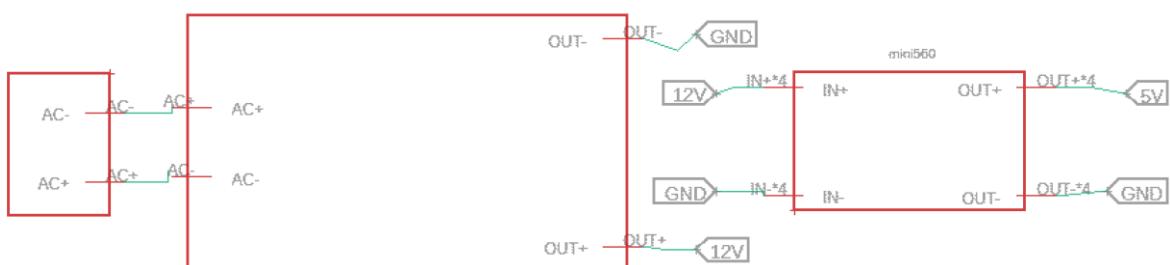
Tabel 2.5.2 Akurasi Voltage Sensing

Tegangan Multimeter (V)	Tegangan Voltage Sensing (V)	Akurasi
63,5	61,62	97,04%
63,5	61,89	97,46%
63,5	61,71	97,18%
63,5	61,91	97,50%
63,7	61,81	97,03%

Hasil pengujian pengukuran tegangan menunjukkan akurasi pembacaan tegangan sebesar 97%.

2.6 Sub-Sistem Power Supply

2.6.1 Implementasi



Gambar 2.6.1 Skematic subsistem power supply

Kebutuhan tegangan pada sisi low voltage membutuhkan tegangan dengan rating 12V dan 5V. sehingga desain supply daya tersebut yaitu mengubah tegangan ac menjadi DC 12V lalu menurunkannya dengan buck converter menjadi 5V.

Penggunaan power supply untuk kontrol dan komponen *low voltage* lainnya besar dari converter AC-DC dengan rating daya maksimum 20W. Converter tersebut mengubah tegangan AC menjadi tegangan DC 12V. converter yang digunakan yaitu Hi-Link dengan bentuk fisik sebagai berikut.

Keluaran 12V tersebut digunakan untuk meyalurkan daya ke komponen pendingin aktif kipas, rangkaian mosfet driver, dan sebagai input daya pada tester board.

Lalu keluaran dari 12V tersebut diturunkan menjadi 5V menggunakan buck converter mini560. Output dari buck converter tersebut digunakan untuk menyalakan mikrokontroler, komponen IoT, board sampling, rangkaian driver mosfet, dan board interface.

2.6.2 Pengujian

Pengujian pada subsistem power supply dilakukan dengan mengukur tegangan pada masing-masing output yaitu 12V dan 5V.

Tabel 2.6.1 Verifikasi output subsistem power supply

Pembanding	Hi-Link (V)	Mini 560 (V)
Datasheet	12	5
PCB	12.2	5.11

Hasil pengujian sudah menunjukkan hasil yang sesuai dengan yang diharapkan walaupun terdapat galat dibawah 2.5%.

2.7 Casing

Agar charger dapat digunakan dengan aman dan nyaman, maka diperlukan casing untuk mengisolasi komponen dengan pengguna. Bahan casing yang digunakan akan dipilih antara plat besi dan alumunium. Kedua bahan tersebut dipilih karena memiliki ketahanan yang kuat. Perbandingan kedua jenis tersebut dapat dilihat lebih detail pada Tabel 2.7.1 berikut.

Tabel 2.7.1 Perbandingan bahan pembuatan casing

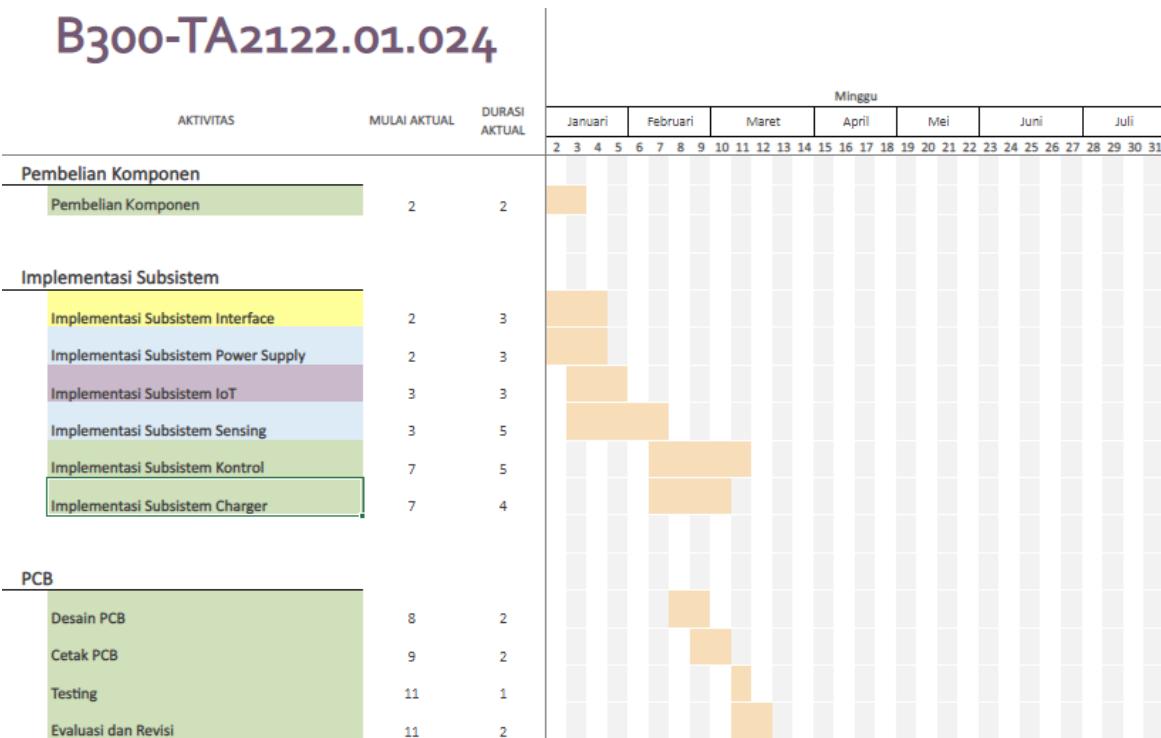
Karakteristik	Plat Besi	Aluminium
Kepadatan (g/cm^3)	7.87	2,7
Kekuatan tekanan (MPa)	240	90
Temperatur ($^\circ\text{C}$)	188	659

Berdasarkan tabel diatas, kepadatan aluminium lebih rendah dari plat besi. Selain itu, kekuatan tekanan plat besi lebih besar dibandingkan dengan alumunium. Oleh karena itu bahan plat besi dipilih sebagai bahan dasar casing.

3 Analisis Pengajaran Implementasi

Tampilkan grafik implementasi dan grafik rencana (gantt chart). Bandingkan dan analisis hasilnya.

Dhanu	Danu	Naoko	Kelompok
-------	------	-------	----------

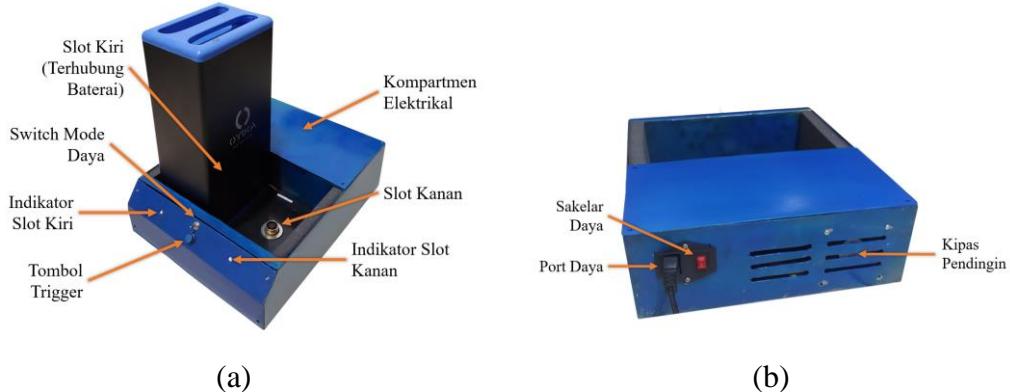


Gambar 2.7.1 Gantt chart rencana jadwal penggerjaan produk

Secara umum, pelaksanaan tugas akhir dari implementasi sampai mencetak PCB sudah sesuai dengan gantt chart yang dibuat di B300. Namun karena terdapat kesalahan dalam desain, proses testing serta evaluasi dan revisi mengalami kemunduran sampai waktu yang belum ditentukan. Pada revisi dokumen ini, subsistem charger masih belum dapat mencapai spesifikasi output yang ditentukan, dan akan diperbaiki lagi pada revisi selanjutnya. Selain dari subsistem ini, semua subsistem sudah dirancang dengan baik.

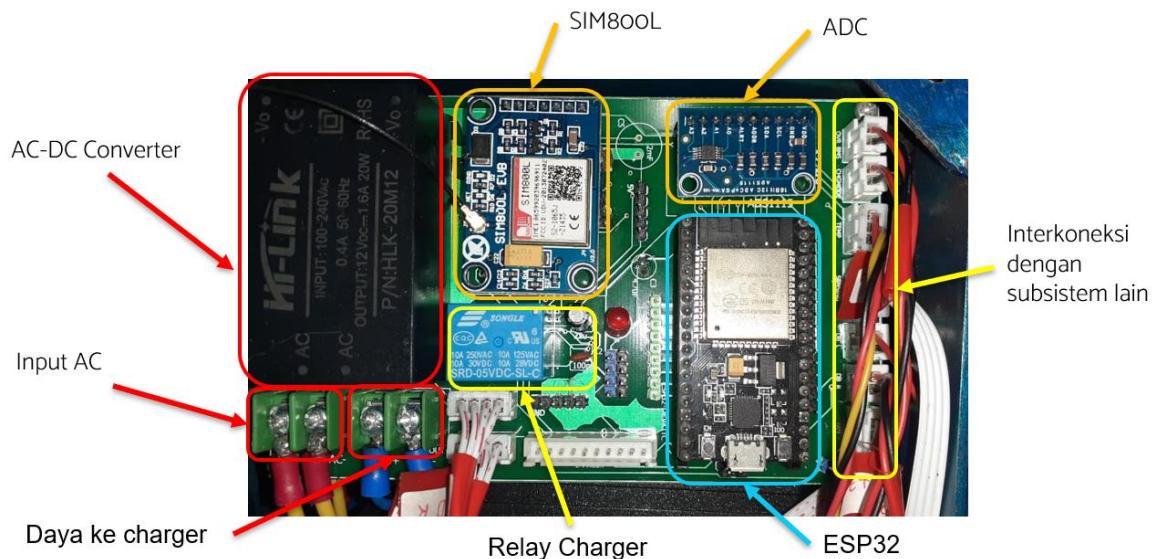
4 Hasil Akhir

Keenam subsistem yang telah dirancang berhasil diintegrasikan dengan baik, dan dikemas di suatu *casing* agar penggunaannya lebih praktis. Sistem yang telah terintegrasi dapat dilihat pada Gambar 2.7.1

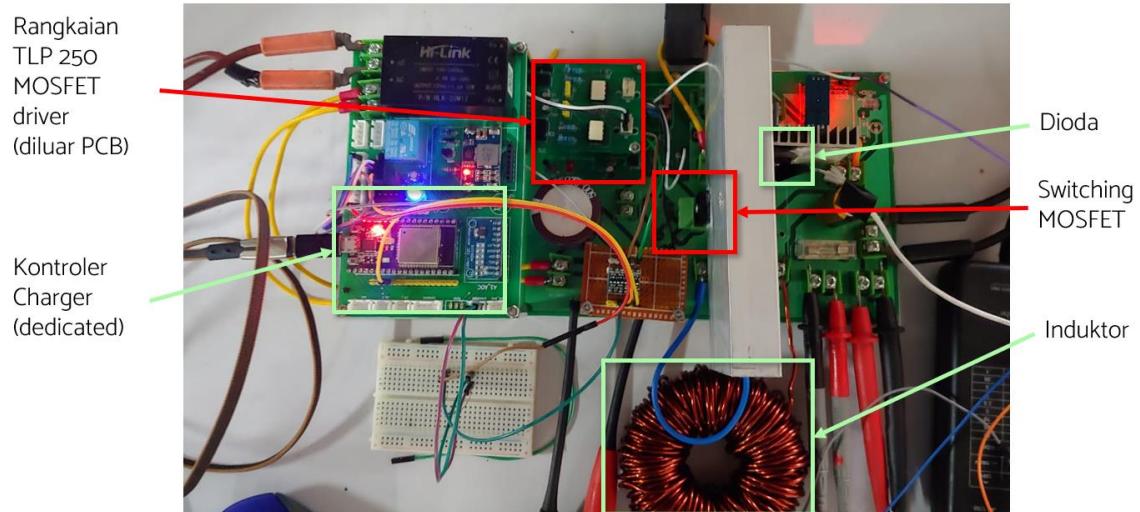


Gambar 2.7.1 Purwarupa produk.
(a) tampak depan dengan baterai pada salah satu slot. (b) tampak belakang

Purwarupa produk dikemas dalam *case* dengan bahan *steel* agar kokoh. Pada bagian depan produk, terdapat dua slot baterai yang dapat diisi oleh baterai Oyika. Selain itu, terdapat dua LED yang bertindak sebagai indikator status masing slot. Terdapat pula switch mode pengisian daya di bagian tengah, serta tombol trigger untuk memberi daya ke BMS baterai apabila daya baterai habis total. Pada bagian belakang, terdapat sakelar daya dan port daya untuk menyalakan sistem. Pada beberapa gambar dibawah ini, dapat dilihat isi dari kompartmen elektrik dari sistem yang dirancang



Gambar 2.7.2 Board Kontrol yang digunakan



Gambar 2.7.3 Hasil implementasi board charger

5 Lampiran

Lampiran A. Setup pengujian subsistem charger untuk kendali closed loop

