

TUGAS AKHIR - EE184801

DESAIN KONTROLER BERBASIS ALGORITME PSO PADA STASIUN PENGISIAN DAYA UNTUK ELECTRIC VEHICLE DENGAN BATERAI NIMH

NUH ENOLA

NRP 07111840000140

Dosen Pembimbing

Eka Iskandar, S.T., M.T.

NIP 198005282008121001

Ir. Rusdhianto Effendi, AK., M.T.

NIP 195704241985021001

Program Studi Sarjana Teknik Elektro

Departemen Teknik Elektro

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2022



TUGAS AKHIR - EE184801

**DESAIN KONTROLER BERBASIS ALGORITME PSO
PADA STASIUN PENGISIAN DAYA UNTUK ELECTRIC
VEHICLE DENGAN BATERAI NIMH**

NUH ENOLA

NRP 07111840000140

Dosen Pembimbing

Eka Iskandar, S.T., M.T.

NIP 198005282008121001

Ir. Rusdhianto Effendi, AK., M.T.

NIP 195704241985021001

Program Studi Sarjana Teknik Elektro

Departemen Teknik Elektro

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2022



FINAL PROJECT - EE184801

**CHARGING STATION CONTROLLER DESIGN USING
PSO ALGORITHM FOR ELECTRIC VEHICLE WITH
NIMH BATTERY**

NUH ENOLA

NRP 07111840000140

Advisor

Eka Iskandar, S.T., M.T.

NIP 198005282008121001

Ir. Rusdhianto Effendi AK., M.T.

NIP 195704241985021001

Electrical Engineering Undergraduate Program

Department of Electrical Engineering

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2022

LEMBAR PENGESAHAN

DESAIN KONTROLER BERBASIS ALGORITME PSO PADA STASIUN PENGISIAN DAYA UNTUK ELECTRIC VEHICLE DENGAN BATERAI NIMH

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Teknik pada
Program Studi S-1 Teknik Elektro
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : **NUH ENOLA**

NRP. 07111840000140

Disetujui oleh Tim Penguji Tugas Akhir :

1. Eka Iskandar, S.T., M.T.

Pembimbing



2. Ir. Rusdhianto Effendi AK., M.T.

Ko-pembimbing



3. Prof. Dr. Ir. Achmad Jazidie, M.Eng.

Penguji



4. Dr. Trihastuti Agustinah, S.T., M.T.

Penguji



5. Dr. Ir. Ari Santoso, DEA.

Penguji



SURABAYA

Juni, 2022

(Halaman ini sengaja dikosongkan)

APPROVAL SHEET

CHARGING STATION CONTROLLER DESIGN USING PSO ALGORITHM FOR ELECTRIC VEHICLE WITH NIMH BATTERY

FINAL PROJECT

Submitted to full fil one of the requirements

For obtaining a bachelor's degree at

Undergraduate Study Program of Electrical Engineering

Department of Electrical Engineering

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

By : **NUH ENOLA**

NRP. 07111840000140

Approved by Final Project Examiner Team :

1. Eka Iskandar, S.T., M.T.

Advisor



2. Ir. Rusdhianto Effendi AK., M.T.

Co-Advisor



3. Prof. Dr. Ir. Achmad Jazidie, M.Eng.

Examiner



4. Dr. Trihastuti Agustinah, S.T., M.T.

Examiner



5. Dr. Ir. Ari Santoso, DEA.

Examiner



SURABAYA

June, 2022

(Halaman ini sengaja dikosongkan)

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP : Nuh Enola / 07111840000140
Departemen : Teknik Elektro
Dosen Pembimbing / NIP : Eka Iskandar, S.T., M.T. / 198005282008121001

dengan ini menyatakan bahwa Tugas Akhir dengan judul “Desain Kontroler Berbasis Algoritme PSO pada Stasiun Pengisian Daya Untuk Electric Vehicle Dengan Baterai NiMH” adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 10 Juni 2022

Mengetahui
Dosen Pembimbing



Eka Iskandar, S.T., M.T.
NIP. 198005282008121001

Mahasiswa



Nuh Enola
NRP. 07111840000140

(Halaman ini sengaja dikosongkan)

STATEMENT OF ORIGINALITY

The undersigned below:

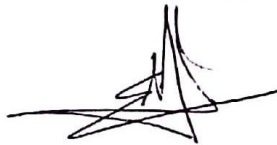
Name of student / NRP : Nuh Enola / 07111840000140
Department : Electrical Engineering
Advisor / NIP : Eka Iskandar, S.T., M.T. / 198005282008121001

hereby declare that the Final Project with the title of “Charging Station Controller Design using PSO Algorithm for Electric Vehicle with NiMH Battery” is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

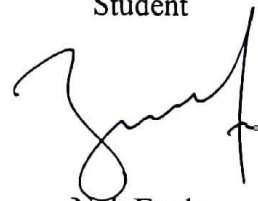
Surabaya, 10 June 2022

Acknowledged
Advisor



Eka Iskandar, S.T., M.T.
NIP. 198005282008121001

Student



Nuh Enola
NRP. 07111840000140

(Halaman ini sengaja dikosongkan)

ABSTRAK

DESAIN KONTROLER BERBASIS ALGORITME PSO PADA STASIUN PENGISIAN DAYA UNTUK ELECTRIC VEHICLE DENGAN BATERAI NIMH

Nama Mahasiswa / NRP : Nuh Enola / 07111840000140
Departemen : Teknik Elektro FTEIC - ITS
Dosen Pembimbing : Eka Iskandar, S.T., M.T.

Abstrak

Kendaraan listrik merupakan suatu penemuan yang tengah berkembang pesat saat ini. Berbagai penelitian mengenai kendaraan listrik tidak berhenti dilakukan, termasuk mengenai sistem pengecasan. Banyak jenis baterai yang digunakan untuk kendaraan listrik, salah satunya adalah baterai NiMH. Penelitian mengenai bagaimana mengoptimalkan pengecasan baterai juga telah banyak dilakukan, salah satunya dengan memanfaatkan *intelligent algorithm*. Namun, cara ini jarang diimplementasikan secara *real-time* dan hanya melalui bantuan *software* komputer. Pada tugas akhir ini, dibahas mengenai implementasi *intelligent algorithm* PSO (*Particle Swarm Optimization*) sebagai metode pengoptimalan secara *real time* pada *charger controller* dengan harapan mampu memberikan solusi terkait pengoptimalan pengecasan sesuai kebutuhan pengguna. Untuk 3 kondisi pengecasan pengoptimalan pada *prototype* menghasilkan perhitungan dengan error *cost* masing-masing 0.33%, 7.22%, dan 5.55% dibandingkan dengan hasil pada simulasi. Dengan nilai ini, implementasi PSO pada sistem *real-time* telah mencapai tingkat keberhasilan sebesar 95%.

Kata kunci: NiMH, PSO, Kendaraan Listrik, *Charger Controller*.

(Halaman ini sengaja dikosongkan)

ABSTRACT

CHARGING STATION CONTROLLER DESIGN USING PSO ALGORITHM FOR ELECTRIC VEHICLE WITH NIMH BATTERY

Student Name / NRP : Nuh Enola / 07111840000140
Department : Electrical Engineering ELECTICS - ITS
Advisor : Eka Iskandar, S.T., M.T.

Abstract

Electric vehicles are an invention that is currently developing rapidly. Various studies on electric vehicles do not stop, including the charging system. Many types of batteries are used for electric vehicles, including NiMH Battery. There is also a lot of research on how to optimize battery charging, one of which is by using intelligent algorithms. However, this method is rarely implemented in real-time and only through the help of the computer software. In this final project, we discuss the implementation of the intelligent algorithm PSO (Particle Swarm Optimization) as a real time optimization method on the charger controller with the hope of providing solutions according to the user needs . For the 3 optimization charging conditions the *Prototype* results in calculations with error costs of 0.33%, 7.22%, and 5.55%, respectively, compared to the results in the simulation. With this value, the implementation of PSO in real-time systems has achieved a 95% success rate.

Keywords: NiMH, PSO, *Electric Vehicle, Charger Controller.*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Segala puji syukur penulis panjatkan kepada Allah S.W.T yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat membuat dan menyelesaikan tugas akhir ini dengan baik dan tepat waktu. Kegiatan tugas akhir ini merupakan bagian dari penyelesaian studi S-1 Departemen Teknik Elektro Institut Teknologi Sepuluh Nopember, dan laporan tugas akhir ini disusun untuk melengkapi hasil capaian dari tugas akhir yang telah dilaksanakan. Dalam pembuatan tugas akhir ini, penulis telah dibantu dan didukung oleh banyak pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Orang tua serta keluarga yang telah memberikan dukungan dan doa kepada penulis.
2. Bapak Eka Iskandar, S.T., M.T. selaku dosen pembimbing yang telah memberi arahan, bimbingan, dan saran yang membangun pada penulis selama proses pengerjaan tugas akhir ini.
3. Bapak Almarhum Ir. Rusdhianto Effendi AK., M.T. yang telah membantu pengerjaan tugas akhir ini di awal hingga akhir hidup beliau.
4. Bapak Dr. Ir. Ari Santoso, DEA selaku dosen yang telah membantu memberi arahan, bimbingan, serta saran yang membangun selama pengerjaan tugas akhir ini.
5. Dosen dan tenaga pendidik Departemen Teknik Elektro ITS, khususnya bidang studi Teknik Sistem Pengaturan.
6. Alya, Matrix, Dheo, Angga, dan seluruh teman bidang studi Teknik Sistem Pengaturan yang terus memberikan motivasi serta dukungan selama proses pengerjaan tugas akhir ini.
7. Teman-teman penulis lain dan seluruh pihak yang berkontribusi yang namanya tidak bisa disebutkan satu persatu, yang telah memberikan dukungan teknis maupun non teknis selama proses pengerjaan tugas akhir ini.

Penulis menyadari, bahwa masih terdapat kekurangan dari laporan tugas akhir ini dikarenakan keterbatasan dan kurangnya pengetahuan serta pengalaman penulis. Penulis berharap tugas akhir ini dapat bermanfaat bagi pembaca. Saran dan kritik juga penulis harapkan untuk pengembangan tugas akhir ini.

Surabaya, 10 Juni 2022

Nuh Enola

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

LEMBAR PENGESAHAN	i
APPROVAL SHEET	iii
PERNYATAAN ORISINALITAS	v
STATEMENT OF ORIGINALITY	vii
ABSTRAK	ix
ABSTRACT	xi
KATA PENGANTAR	xiii
DAFTAR ISI	xv
DAFTAR GAMBAR	xvii
DAFTAR TABEL	xviii
DAFTAR SIMBOL	xix
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Batasan Masalah	2
1.4 Tujuan	2
1.5 Manfaat	2
BAB 2 TINJAUAN PUSTAKA	3
2.1 Hasil Penelitian Terdahulu	3
2.2 Dasar Teori	3
2.2.1 Baterai Nickel Metal Hydride	3
2.2.2 Particle Swarm Optimization	5
2.2.3 Charger Controller	7
2.2.4 PID Controller	8
2.2.5 Transistor	9
BAB 3 METODOLOGI	11
3.1 Metode yang digunakan	11
3.2 Bahan dan peralatan yang digunakan	11
3.3 Urutan pelaksanaan penelitian	12
3.3.1 Penentuan Model Matematika Baterai NiMH	12
3.3.2 Perancangan Sistem <i>Charger Controller</i>	15
3.3.3 Perancangan <i>PI Controller</i> pada Sistem	15
3.3.4 Perancangan Metode PSO untuk Baterai NiMH	15
	xv

3.3.5	Perancangan Simulasi dengan MATLAB	17
3.3.6	Perancangan <i>Prototype Charger Controller</i>	19
3.3.7	Pengambilan Data Percobaan	20
BAB 4	HASIL DAN PEMBAHASAN	23
4.1	Estimasi parameter baterai	23
4.2	Pengujian algoritme PSO	24
4.3	Hasil Desain PI <i>Controller</i>	26
4.4	Hasil pengujian Simulasi MATLAB	26
4.5	Hasil pengujian <i>Prototype</i>	29
4.6	Perbandingan Simulasi dan <i>Prototype</i>	31
BAB 5	KESIMPULAN DAN SARAN	39
5.1	Kesimpulan	39
5.2	Saran	39
	DAFTAR PUSTAKA	41
	LAMPIRAN	43
1.	Program Simulasi	43
2.	Program <i>Prototype</i> dengan Mikrokontroler AVR	48
3.	Datasheet baterai	62
4.	Datasheet Komponen	63
4.1.	Datasheet ATmega32A	63
4.2.	Datasheet TIP142	64
	BIODATA PENULIS	67

DAFTAR GAMBAR

Gambar 2.1 Rangkaian ekuivalen untuk baterai	4
Gambar 2.2 Contoh <i>charging rules</i> untuk baterai NiMH	5
Gambar 2.3 Rangkaian DAC R-2R	8
Gambar 3.1 Rangkaian <i>Simulink pulse discharging</i>	13
Gambar 3.2 Rangkaian <i>Simulink</i> Hubungan Voc-SOC	14
Gambar 3.3 Diagram Blok Sistem <i>Charger Controller</i>	15
Gambar 3.4 Diagram Alir Algoritme PSO	17
Gambar 3.5 Rancangan Simulasi dengan <i>Simulink MATLAB</i>	18
Gambar 3.6 Skema Rangkaian <i>Prototype Charger Controller</i>	19
Gambar 3.7 Diagram Alir Cara Kerja Sistem	20
Gambar 3.8 Fitur <i>Recorder</i> pada Osiloskop Digital	22
Gambar 3.9 Skema (kiri) dan Rangkaian (kanan) Tambahan	22
Gambar 4.1 Grafik Pengecasan Kondisi 1 pada Simulasi <i>MATLAB</i>	27
Gambar 4.2 Grafik Pengecasan Kondisi 2 pada Simulasi <i>MATLAB</i>	27
Gambar 4.3 Grafik Pengecasan Kondisi 3 pada Simulasi <i>MATLAB</i>	28
Gambar 4.4 Grafik Pengecasan Kondisi 1 pada <i>Prototype</i>	29
Gambar 4.5 Grafik Pengecasan Kondisi 2 pada <i>Prototype</i>	30
Gambar 4.6 Grafik Pengecasan Kondisi 3 pada <i>Prototype</i>	30
Gambar 4.7 Perbandingan Grafik Arus Simulasi dan <i>Prototype</i> Kondisi 1	33
Gambar 4.8 Perbandingan Grafik Arus Simulasi dan <i>Prototype</i> Kondisi 2	33
Gambar 4.9 Perbandingan Grafik Arus Simulasi dan <i>Prototype</i> Kondisi 3	34
Gambar 4.10 Perbandingan Grafik SOC Simulasi dan <i>Prototype</i> Kondisi 1	34
Gambar 4.11 Perbandingan Grafik SOC Simulasi dan <i>Prototype</i> Kondisi 2	35
Gambar 4.12 Perbandingan Grafik SOC Simulasi dan <i>Prototype</i> Kondisi 3	35
Gambar 4.13 Perbandingan Grafik Tegangan Simulasi dan <i>Prototype</i> Kondisi 1	36
Gambar 4.14 Perbandingan Grafik Tegangan Simulasi dan <i>Prototype</i> Kondisi 2	36
Gambar 4.15 Perbandingan Grafik Tegangan Simulasi dan <i>Prototype</i> Kondisi 3	37

DAFTAR TABEL

Tabel 3.1 Daftar Bahan yang Digunakan	11
Tabel 3.2 Daftar Peralatan yang Digunakan	12
Tabel 3.3 Parameter awal estimasi	13
Tabel 3.4 <i>Range</i> Arus setiap <i>Stage</i>	16
Tabel 4.1 Nilai parameter sistem	23
Tabel 4.2 Rata-Rata Cost untuk Kombinasi Parameter	24
Tabel 4.3 Parameter PI <i>Controller</i> untuk setiap <i>Stage</i> Pengecasan	26
Tabel 4.4 Parameter Kondisi Pengecasan	26
Tabel 4.5 Data Pengujian Simulasi	27
Tabel 4.6 Data Pengujian <i>Prototype</i>	29
Tabel 4.7 Perbandingan Optimasi Simulasi dan <i>Prototype</i>	32

DAFTAR SIMBOL

V_T	: Tegangan terminal baterai
V_{cp}	: Tegangan cabang RC
V_{ceq}	: Tegangan <i>equivalent open circuit</i> baterai
I_{ch}	: Arus pengecasan
I	: Arus tersimpan di baterai
R_0	: Hambatan dalam baterai
R_p	: Hambatan cabang RC
C_p	: Kapasitansi cabang RC
Δt	: Waktu sampling pengecasan
A	: Amplitudo area eksponensial
B	: Invers konstanta area eksponensial
K	: Resistansi polaritas
E_0	: Tegangan konstan baterai
Q	: Kapasitas baterai sebenarnya
Q_{max}	: Kapasitas baterai maksimum
i^*	: Arus terfilter
V_{exp}	: Tegangan eksponensial
SOC	: <i>State of charge</i> baterai
$SOCOCV$: Polinomial hubungan SOC-OCV
η_b	: Efisiensi baterai
x_b	: State model baterai
p_n	: Konstanta polinomial SOCOCV ke-n
J_b	: <i>Cost</i> keseluruhan pengecasan
J_1	: <i>Cost</i> waktu pengecasan
J_2	: <i>Cost loss</i> pengecasan
J_3	: <i>Cost</i> tegangan terminal akhir pengecasan
α_b	: Faktor bobot kepentingan relatif baterai (waktu)
β_b	: Faktor bobot kepentingan relatif baterai (<i>loss</i>)

T_{min}	: Waktu pengecasan minimum
T_{max}	: Waktu pengecasan maksimum
L_{min}	: <i>Loss</i> pengecasan minimum
L_{max}	: <i>Loss</i> pengecasan maksimum
V_{Tmin}	: Tegangan terminal baterai minimum
V_{Tmax}	: Tegangan terminal baterai maksimum
T	: Waktu pengecasan
L	: <i>Loss</i> pengecasan
V_{TL}	: Tegangan terminal akhir pengecasan
$u(t)$: Sinyal kontrol
$e(t)$: Error
$r(t)$: <i>Setpoint</i>
$y(t)$: <i>Output</i>
K_p	: Parameter proporsional
K_i	: Parameter integral
K_D	: Parameter diferensial
G_{pid}	: Fungsi transfer PID
t_{sim}	: Waktu simulasi sistem
h	: Waktu sampling PID
J	: <i>Cost</i> PID
α	: Faktor bobot kepentingan relatif PID (<i>IAE</i>)
β	: Faktor bobot kepentingan relatif PID (<i>TV</i>)
v	: Kecepatan partikel
w	: Momen inersia partikel
$c_{1,2}$: Faktor <i>learning</i>
$r_{1,2}$: Faktor random
p_{best}	: Solusi optimal suatu partikel
g_{best}	: Solusi optimal keseluruhan partikel
x	: Posisi partikel

t	: Variabel aljabar dari evolusi (iterasi)
t_{max}	: Iterasi maksimum
f_{avg}	: <i>Fitness</i> partikel saat itu
f	: <i>Fitness</i> partikel
n	: Jumlah partikel <i>swarm</i>
σ^2	: Variasi <i>fitness</i> sekumpulan partikel
C	: Nilai acak terdistribusi <i>Cauchy</i>
η	: Konstanta <i>step size</i> variasi

(Halaman ini sengaja dikosongkan)

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa dekade terakhir, penggunaan *Electric Vehicle* atau EV sangat berkembang pesat, apalagi dengan adanya kenaikan emisi dari kendaraan bermotor dengan bahan bakar minyak. *Electric Vehicle* memiliki banyak macam, baik itu berbentuk mobil, sepeda, hingga kendaraan besar seperti truck dan beberapa kendaraan industri seperti *forklift*. Penggunaan EV tentu tidak akan terlepas dari proses *charging* sebagai sumber energinya. Proses tentu memerlukan waktu yang cukup lama dikarenakan butuh kapasitas baterai yang cukup besar untuk mampu menggerakkan suatu kendaraan listrik. Hal inilah yang seringkali masih menjadi permasalahan dalam penggunaan kendaraan listrik. Proses *charging* tidak boleh dilakukan sembarangan agar dapat menjaga keawetan baterai. Terlebih lagi, proses *charging* EV yang tidak dikontrol dapat memberikan dampak negatif pada jaringan distribusi energi listrik (Shafiee et al., 2012). EV sendiri memiliki banyak jenis baterai yang digunakan, salah satunya adalah jenis baterai NiMH atau *Nickel Metal Hydride*.

Baterai NiMH merupakan jenis baterai yang menggunakan nikel sebagai bahan utamanya. Indonesia sendiri merupakan salah satu negara penghasil nikel terbesar di dunia sehingga jenis baterai dengan bahan dasar NiMH dapat menjadi pendapatan besar bagi negara Indonesia (U.S. Geological Survey, 2021). Sekalipun jenis baterai NiMH lebih jarang digunakan daripada baterai Li-Ion, jenis baterai ini diperkirakan masih akan terus digunakan dalam berbagai aplikasi termasuk pada EV. *Range* dari temperatur kerja dari baterai NiMH telah diperluas hingga 100°C (lebih tinggi dari jenis baterai dengan *Li cell*) sehingga cocok digunakan dalam aplikasi di bidang otomotif. Jenis baterai ini juga memiliki *life cycle* yang lebih panjang dibandingkan dengan baterai *Li cell*. Proses kontrol pengisian daya pada jenis baterai ini dapat meningkatkan *life span* dan memaksimalkan *life cycle* dari baterai ini. Namun, NiMH dikenal memiliki cara yang cukup kompleks untuk pengisian sehingga penelitian mengenai proses kontrol pada jenis baterai ini masih terus dilakukan, baik dengan metode analitis maupun cerdas (*intelligent algorithm*) (Arya & Verma, n.d.).

Pada umumnya, pengecasan hanya dilakukan secara langsung dengan menggunakan charger yang memiliki tegangan lebih tinggi untuk mengisi daya baterai dengan tegangan yang lebih rendah. Biasanya, metode yang digunakan dalam proses ini adalah switching biasa. Beberapa penelitian juga menerapkan proses pengecasan dengan menggunakan prinsip kontrol PID. Namun, metode ini belum tentu dapat mengoptimalkan proses pengecasan. Banyak penelitian juga telah menggunakan *intelligent algorithm* untuk melakukan pengecasan namun kebanyakan dari penelitian ini masih berbasis simulasi dengan jenis baterai Li-Ion. Hal ini dikarenakan *battery management system* memerlukan estimasi SOC yang akurat agar proses pengisian dapat dilakukan secara maksimal dalam kondisi real-time. Dari berbagai penelitian yang telah ada, proses pengoptimalan *charging* seringkali dilakukan dengan media komputer sebelum di proses oleh charger controller itu sendiri.

1.2 Rumusan Masalah

Bagaimana cara mengimplementasikan pengoptimalan dengan memanfaatkan *intelligent algorithm* pada suatu sistem terpisah dengan komputer, serta dapat dilakukan dengan mempertimbangkan kebutuhan pengguna dari segi waktu pengecasan.

1.3 Batasan Masalah

Dalam penelitian tugas akhir ini diterapkan beberapa batasan masalah antara lain sebagai berikut:

1. Sistem dirancang dengan bahasa pemrograman C untuk microcontroller AVR
2. Sistem dirancang hanya untuk jenis baterai NiMH
3. *Prototype* dirancang untuk jenis baterai dengan kapasitas kecil (12V 10Ah)
4. Kapasitas baterai dianggap kosong 0% pada 10 Volt dan penuh 100% pada 14.5 Volt
5. Model sistem dirancang dengan mengabaikan *loss* dari komponen selain baterai.
6. Perhitungan *loss* hanya sebagai objektivitas dan tidak diukur dalam simulasi maupun *prototype*.
7. *Prototype* dirancang dengan sumber tegangan 240Watt 10A dengan kemampuan supply arus sebesar 7A.
8. Waktu input minimal dari pengguna adalah 1800 detik dengan kondisi awal baterai lebih dari 50%.

1.4 Tujuan

Tujuan dari penelitian ini adalah untuk merancang suatu charge controller yang mampu mengidentifikasi pengoptimalan sesuai dengan kebutuhan pengguna dengan memanfaatkan *intelligent algorithm* yang kemudian akan diimplementasikan pada microcontroller agar dapat diuji secara real-time dan real-system.

1.5 Manfaat

Manfaat dari penelitian tugas akhir ini adalah untuk memberikan solusi baru terkait implementasi desain smart *charging* serta membuka peluang penelitian baru yang masih saling berkaitan. Selain itu, penelitian tugas akhir ini juga memiliki manfaat sebagai media pembelajaran serta pengimplementasian ilmu yang diperoleh penulis selama masa perkuliahan.

BAB 2 TINJAUAN PUSTAKA

2.1 Hasil Penelitian Terdahulu

Pengecasan atau pengisian daya baterai merupakan salah satu proses yang penting dalam *Battery Management System* (BMS). Salah satu metode umum yang digunakan dalam pengecasan baterai adalah metode *constant current-constant voltage* (CC-CV) dimana pada awal pengecasan baterai akan diisi dengan arus konstan hingga tegangan naik dan mencapai batas atas tegangan lalu metode akan diubah menjadi tegangan konstan. Namun fase tegangan konstan ini membutuhkan waktu lama sehingga secara keseluruhan waktu pengecasan akan meningkat dan menyebabkan *life cycle* baterai yang lebih singkat. Oleh karena itu, banyak penelitian yang telah dilakukan untuk mencari cara meningkatkan efisiensi pengecasan termasuk dari segi meminimalkan waktu pengecasan.

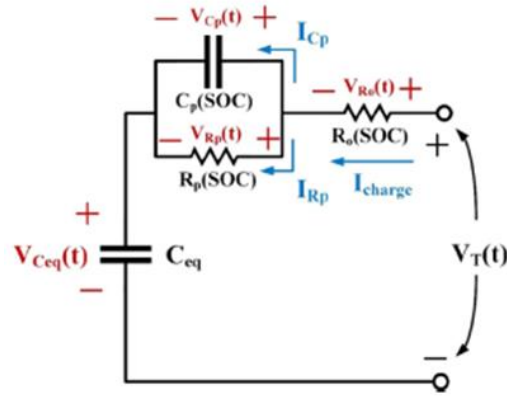
Salah satu penelitian mengenai pengoptimalan pengecasan yang telah dilakukan adalah dengan menentukan *current pattern* berdasarkan algoritme *grey wolf optimizer* pada baterai jenis Li-Ion. Metode ini memanfaatkan cara pengecasan *Multi-Stage Constant Current* (MSCC) yang memiliki beberapa keuntungan yaitu mudah diimplementasikan, kenaikan temperatur rendah, dan efisiensi pengecasan tinggi. Dari penelitian yang telah dilakukan, terbukti metode ini mampu meningkatkan sistem sebanyak 5.33% pada waktu pengecasan, 25.99% pada kenaikan suhu maksimum, 19.59% pada rata-rata kenaikan suhu, dan 0.48% pada efisiensi pengecasan. (Chen et al., 2021).

Cara lain dalam pengoptimalan pengecasan adalah dengan meminimalkan waktu respons serta *cost* dari kontroler. Baterai memiliki bentuk sistem yang tidak linear dan juga memiliki histeresis antara proses *charging* dan *discharging* sehingga mendapatkan parameter ideal untuk proses pengecasan sulit tercapai. Salah satu cara mengoptimalkan parameter pengecasan ini adalah dengan memanfaatkan *intelligent algorithm*. Salah satu penelitian telah menerapkan *Particle Swarm Optimization* yang telah dikembangkan dalam menentukan nilai parameter Kp, Ki, dan Kd pada kontroler PID untuk pengecasan baterai. Pada penelitian ini, diperoleh hasil berupa peningkatan efisiensi baterai dari 86.44% menjadi 91.47% dan penurunan kenaikan suhu pengecasan sebanyak 1°C. (Wu et al., 2020)

2.2 Dasar Teori

2.2.1 Baterai Nickel Metal Hydride

Baterai NiMH atau *Nickel Metal Hydride* merupakan salah satu jenis baterai yang menggunakan nikel sebagai bahan utamanya. Jenis baterai ini merupakan perkembangan dari baterai NiCad atau *Nickel Cadmium*. Baterai NiMH merupakan salah satu baterai yang populer digunakan karena memiliki beberapa kelebihan seperti kepadatan energi yang tinggi, harga murah, tidak beracun dan aman, serta siklus hidup panjang (Baterijom, 2021), (Windarko & Choi, 2010). Secara umum, baterai sendiri dapat digambarkan dalam rangkaian ekuivalen seperti pada Gambar 2.1. (Chen et al., 2021)



Gambar 2.1 Rangkaian ekuivalen untuk baterai

Dari rangkaian equivalent diatas, output tegangan terminal baterai V_T dan tegangan cabang paralel RC V_{cp} dapat diperoleh dan dituliskan dalam bentuk persamaan dinamis sebagaimana tertera pada persamaan (2.1) dan (2.2). (Khanum et al., 2021)

$$V_T(k) = V_{ceq}(k) - V_{cp}(k) - I_{ch}(k)R_0 \quad (2.1)$$

$$V_{cp}(k + 1) = e^{\frac{-\Delta t}{R_p C_p}} V_{cp}(k) + R_p \left(1 - e^{\frac{-\Delta t}{R_p C_p}} \right) I_{ch}(k) \quad (2.2)$$

Dengan V_{ceq} , I_{ch} , R_0 , R_p , C_p dan Δt secara berurutan merupakan tegangan ekuivalen *open circuit*, arus pengecasan, hambatan dalam, resistansi cabang paralel RC, kapasitansi cabang paralel RC, dan waktu sampling data. Perbedaan utama jenis baterai lain dengan baterai NiMH adalah adanya fenomena histeresis antara tahap *charging* dan *discharging*. Baterai NiMH memiliki persamaan tegangan eksponensial yang dapat dituliskan dalam persamaan (2.3). Persamaan tegangan terminal spesifik pada baterai NiMH sendiri ditunjukkan oleh persamaan (2.4) untuk proses *discharging* dan (2.5) untuk proses *charging* (Mars et al., 2017).

$$\dot{V}_{exp}(t) = B|I(t)| \left(-V_{exp}(t) + Au(t) \right) \quad (2.3)$$

$$V_T(t) = E_0 - K \frac{Q_{max}}{Q_{max} - Q(t)} (Q(t) + i^*(t)) + V_{exp}(t) - R_0 \cdot I(t) \quad (2.4)$$

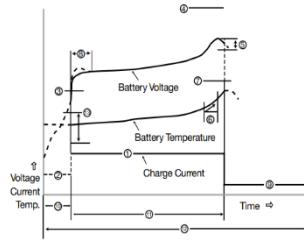
$$V_T(t) = E_0 - K \frac{Q_{max}}{|Q(t)| - 0.1Q_{max}} i^*(t) - K \frac{Q_{max}}{Q_{max} - Q(t)} + V_{exp}(t) - R_0 \cdot I(t) \quad (2.5)$$

Dimana E_0 merupakan tegangan konstan baterai, V_{exp} merupakan tegangan eksponensial baterai, B merupakan invers konstanta area eksponensial waktu, A merupakan amplitudo area eksponensial, K merupakan resistansi polarisasi, Q_{max} merupakan kapasitas maksimum baterai, Q merupakan kapasitas baterai yang sebenarnya, i^* merupakan arus terfilter, $u(t)$ merupakan sinyal kontrol (1 untuk *charging* dan 0 untuk *discharging*) dan I merupakan arus pada baterai. Baterai NiMH juga memiliki sifat dimana ketika berada telah terisi penuh atau mencapai SOC atau *State of Charge* 100%, tegangan akan mengalami drop. SOC atau persentase kapasitas baterai saat ini dibandingkan kapasitas baterai maksimum dapat dituliskan dalam persamaan dinamis (2.6).

$$SOC(k + 1) = SOC(k) + \frac{I(k)\Delta t}{Q_{max} * 3600} \quad (2.6)$$

Untuk mendapatkan proses pengecasan yang optimal, baterai memiliki beberapa *charging rules* dimana arus yang digunakan untuk mengisi baterai tidak akan sama dari SOC 0% hingga 100%. Proses pengisian ini biasanya dipisah menjadi 3 tahap yaitu *bulk*, *absorption*, dan *float*. Terkadang ada tahap tambahan yaitu *qualification* atau *equalization*. Selain itu, ada pula pengecasan yang hanya melibatkan 2 tahap yaitu *bulk* dan *float* (Bogno et al., 2017). Setiap baterai memiliki karakteristik yang berbeda-beda untuk penerapan ketiga *stage* ini, begitu pula untuk jenis baterai NiMH. Contoh *charging rules* untuk baterai NiMH antara lain ditunjukkan oleh Gambar 2.2 (Panasonic Corporation, 2014).

1. Rapid charge current	Max. 1CmA to 0.5CmA
2. Rapid charge transition voltage restoration current	0.2 to 0.3CmA
3. Rapid charge start voltage	Approx. 0.8V/cell
4. Charge terminating voltage	1.8V/cell
5. ΔV value	5 to 10mV/cell
6. Battery temperature rising rate dT/dt value	1 to 2°C/min
7. Maximum battery temperature TCO	60°C (for L-A, L-fatA and SC size) 55°C (for A, AA and D size) 50°C (for QA, AAA and prismatic size)
8. Initial - ΔV detection disabling timer	5 to 10 min
9. Trickle current (after rapid charge)	0.033 to 0.05CmA
10. Rapid charge transfer timer	60 min
11. Rapid charge timer	90 min (at 1CmA charge)
12. Total timer	10 to 20 hours
13. Rapid charge temperature range	0° to 40°C



Gambar 2.2 Contoh *charging rules* untuk baterai NiMH

2.2.2 Particle Swarm Optimization

Particle Swarm Optimization atau yang biasa disingkat PSO merupakan metode optimasi yang dikenalkan oleh Kennedy dan Eberhart pada tahun 1995. Metode ini di dasarkan pada populasi partikel yang memiliki 2 parameter yaitu kecepatan dan lokasi. Lokasi akan merepresentasikan solusi sementara kecepatan akan menentukan arah dan jarak partikel dalam menemukan solusi dari permasalahan optimasi. Posisi dan kecepatan partikel ini dapat dimodifikasi dan dimanipulasi berdasarkan persamaan (2.6) dan (2.7).

$$v(t + 1) = wv(t) + c_1r_1(p_{best} - x(t)) + c_2r_2(g_{best} - x(t)) \quad (2.7)$$

$$x_{ij}(t + 1) = x_{ij}(t) + v_{ij}(t + 1) \quad (2.8)$$

Dimana t merupakan variabel aljabar dari evolusi (iterasi) saat ini, c_1 dan c_2 merupakan faktor *learning*, r_1 dan r_2 merupakan faktor *random* yang terdistribusi pada *range* [0,1], p_{best} merupakan solusi optimal suatu partikel, dan g_{best} merupakan solusi optimal seluruh populasi. Sementara itu, x merupakan lokasi partikel dan v merupakan kecepatan partikel.

Kecepatan partikel pada metode PSO dipengaruhi oleh dua parameter penting lainnya yaitu berat inersia dan percepatan partikel. Kedua parameter ini juga dapat di sesuaikan berdasarkan persamaan (2.9), (2.10), dan (2.11).

$$w = (w_{start} - w_{end}) \arctan \frac{t_{max} - t}{t_{max}} \quad (2.9)$$

$$c_1 = 1.3 + 1.2 \cos \pi \frac{t}{t_{max}} \quad (2.10)$$

$$c_2 = 2 - 1.2 \cos \pi \frac{t}{t_{max}} \quad (2.11)$$

Dimana w merupakan koefisien inersia dan t merupakan jumlah iterasi. Jarak partikel pada metode PSO sebagai solusi dari permasalahan optimasi dapat diperoleh dengan cara mengobservasi perubahan keseluruhan dalam hal *fitness* atau kesesuaian antar partikel dalam populasi. Variasi *fitness* ini dapat dituliskan dalam persamaan (2.12), (2.13), (2.14), dan (2.15).

$$f_{avg} = \frac{1}{n} \sum_{i=1}^n f_i \quad (2.12)$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n \left(\frac{f_i - f_{avg}}{f} \right)^2 \quad (2.13)$$

$$f = \max\{\max|f_i - f_{avg}|, 1\} (i \in [1, n]) \quad (2.14)$$

$$\lim_{t \rightarrow +\infty} X(t) = P \quad (2.15)$$

Dimana n merupakan jumlah partikel swarm, f_i merupakan *fitness* untuk partikel ke- i , f_{avg} merupakan *fitness* dari partikel saat itu, σ^2 merupakan variasi *fitness* dari sekumpulan partikel yang ada dan P merupakan posisi tetap dari partikel. Parameter kecepatan dan jarak yang didapatkan dari persamaan berikutnya menuntun algoritme PSO mendapatkan nilai variasi yang semakin kecil dan menyebabkan populasi semakin konvergen hingga akhirnya menemukan konvergensi global atau lokal. Solusi ini dinotasikan dengan g_{best} yang dapat berupa solusi optimal local atau global dengan cara membandingkan dengan f_{best} . Namun tidak hanya itu, g_{best} juga dipengaruhi oleh Cauchy mutation yang memiliki *operating formula* sebagaimana tertera pada persamaan (2.16).

$$x_{ij} = x_{ij} + \eta * C(0,1) \quad (2.16)$$

Dimana $j = 1, 2, 3, \dots, \eta$ adalah konstanta yang mengatur *step size* dari variasi, dan $C(0,1)$ adalah nomor acak yang dihasilkan oleh fungsi distribusi Cauchy dengan $T = 1$. Mutasi Cauchy ini membawa beberapa permasalahan. Untuk mengatasi ini, diusulkanlah suatu metode adaptif untuk mutasi Cauchy yang menggunakan kecepatan rata-rata dari kelompok partikel sebagai parameternya. Kecepatan rata-rata sendiri dideskripsikan oleh persamaan (2.17).

$$\bar{v}_j = \frac{1}{n} \sum_{i=1}^n v_{ij} \quad (2.17)$$

Dimana n merupakan jumlah partikel dan v_{ij} merupakan kecepatan dari partikel ke- i pada dimensi ke- j . Dari serangkaian perhitungan yang ada, dapat diperoleh persamaan untuk solusi dari algoritme PSO sebagaimana tertera pada persamaan (2.18) (Wu et al., 2020).

$$g_{best} = g_{best} + \bar{v}_j * C(x_{min}, x_{max}) \quad (2.18)$$

2.2.3 Charger Controller

Proses pengecasan memerlukan controlling agar tidak terjadi *overcharging*. Pada dasarnya, suatu charger control bekerja dengan prinsip mengalirkan arus dari tegangan sumber yang lebih tinggi menuju baterai. Dalam pengecasan baterai, terdapat beberapa metode yang dapat digunakan antara lain dengan menggunakan tegangan konstan, arus konstan, campuran tegangan dan arus konstan, metode pengecasan dengan *pulse*, serta pengecasan dengan metode *float*. Metode tegangan konstan merupakan metode standar dimana tegangan pengecasan sepanjang proses akan tetap sama. Sementara itu, pada metode arus konstan baterai dihubungkan membentuk kelompok kecil dan setiap kelompok di charge dari sumber DC melalui hambatan beban. Arus pengecasan akan dijaga konstan dengan mengurangi resistansi rangkaian seiring naiknya tegangan baterai. Kedua metode ini memiliki kelebihan dan kekurangan masing-masing sehingga untuk meningkatkan proses pengecasan dilakukan suatu metode gabungan tegangan dan arus konstan. Pada metode ini, arus konstan digunakan di awal pengecasan sementara tegangan konstan akan digunakan ketika tercapai tegangan baterai tertentu.

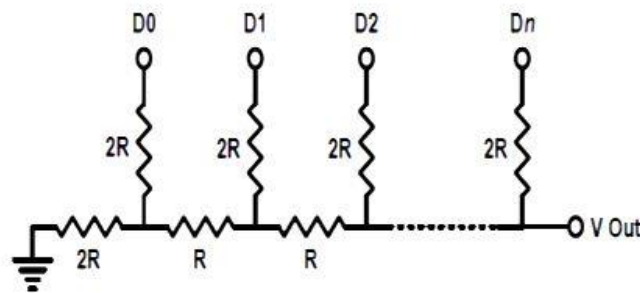
Selain mengatur tegangan dan arus, terdapat pula metode pengecasan dengan *pulse*. Pada metode ini, terdapat saat dimana proses pengecasan berhenti sehingga dapat membuat elektrolit baterai menjadi lebih uniform dan meningkatkan efisiensi pengecasan. Metode selanjutnya yaitu metode *float* yaitu metode yang biasa digunakan sebagai *backup* energi apabila baterai tidak digunakan. Pada metode ini, *charger* akan beroperasi pada tegangan rendah yang biasanya berada dibawah 2.4V per cell. Hal ini bertujuan untuk menjaga arus pengecasan tetap rendah dan meminimalkan kerusakan baterai akibat *overcharging*.

Terdapat beberapa jenis metode yang dapat digunakan di charger controller antara lain simple on off, PWM, dan MPPT. Simple on off charger controller memanfaatkan penggunaan relay dan menyambungkan dengan sumber tenaga apabila telah tercapai tegangan tertentu. Kelebihan dari metode charger ini adalah kualitas yang tidak tergoyahkan. Sementara itu, metode PWM menggunakan transistor untuk mengatur frekuensi sinyal sedemikian rupa untuk dikirimkan menuju baterai agar dapat menjaga suatu kondisi tegangan tertentu. Metode ini memiliki beberapa kerugian seperti menyebabkan noise dari frekuensi PWM dan juga menghasilkan residu berupa panas. Di sisi lain, metode MPPT merupakan metode yang seringkali digunakan untuk charger controller dengan sumber panel surya.

Metode MPPT merupakan metode yang bertujuan untuk mencari titik dimana dapat diperoleh daya keluaran maksimum. Metode ini berdasar dari daya rata-rata yang dapat dihasilkan dengan mengalikan nilai tegangan yang cukup konstan dengan arus rata-rata pengisian baterai. Dengan begitu, metode MPPT dapat dilakukan dengan memaksimalkan arus rata-rata pengisian baterai (Parthasarathy & Vijayaraj, 2020).

Selain PWM dan MPPT, terdapat pula cara lain untuk mengontrol proses pengecasan yaitu dengan menggunakan DAC atau Digital to Analog Converter. DAC merupakan salah satu

metode yang dapat mengubah sinyal digital menjadi sinyal analog. Berbeda dengan PWM yang mendapatkan besaran analog dengan melakukan *on off* dalam frekuensi tertentu, DAC mendapatkan besaran analog dengan cara mengombinasikan output 0-1 dari rangkaian sehingga mendapatkan kombinasi biner yang mewakili nilai analog tertentu. Cara ini memiliki keunggulan yaitu memberikan noise yang jauh lebih kecil daripada sistem dengan PWM. Namun, DAC membutuhkan banyak pin output untuk dapat menghasilkan besaran analog yang lebih presisi. Rangkaian DAC yang paling umum yaitu rangkaian R-2R sebagaimana tertera pada Gambar 2.3.



Gambar 2.3 Rangkaian DAC R-2R

2.2.4 PID Controller

PID (Proportional, Integral, Derivative) Controller merupakan salah satu jenis teknik kontroler yang terus digunakan sekalipun telah banyak perkembangan teknik kontrol lainnya. Secara umum, suatu PID controller dapat dituliskan dalam persamaan (2.19) dan (2.20).

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_D \frac{de(t)}{dt} \quad (2.19)$$

$$e(t) = r(t) - y(t) \quad (2.20)$$

dimana $e(t)$ adalah error, $r(t)$ adalah nilai referensi, $u(t)$ adalah output dari kontroler, dan K_p, K_i, K_d secara berurutan adalah nilai gain proporsional, integral, dan derivatif. Sementara itu, pada PID controller dalam suatu sistem tertutup (dengan feedback), dapat dituliskan transfer function sebagaimana tertera pada persamaan (2.21).

$$G_{pid} = K_p + \frac{K_i}{s} + sK_d \quad (2.21)$$

Pada PID, sebagaimana kontroler lainnya, terdapat banyak cara untuk mengevaluasi kinerjanya sehingga diperoleh performa maksimal yang dapat mengurangi beban kerja kontroler. Pada umumnya, dalam domain waktu ada beberapa jenis pendekatan tidak langsung, seperti Integral of Square Error (ISE), Integral of Absolute Error (IAE), Integral of Time Weighted Absolute Error (ITAE), and Integral Weighted Square Error (ITSE) yang mana masing-masing secara berurutan dituliskan dalam persamaan (2.22), (2.23), (2.24), dan (2.25).

$$ISE = \int_0^{t_{sim}} e^2(t) dt \quad (2.22)$$

$$IAE = \int_0^{t_{sim}} |e(t)| dt \quad (2.23)$$

$$ITAE = \int_0^{t_{sim}} t|e(t)| dt \quad (2.24)$$

$$ITSE = \int_0^{t_{sim}} te^2(t) dt \quad (2.25)$$

dimana t_{sim} merepresentasikan waktu berjalannya sistem. Selain itu, terdapat pula suatu cara untuk mengevaluasi kehalusan sinyal yang bernama Total Variation (TV) index pada persamaan (2.26).

$$TV_u = \int_0^{t_{sim}} \left| \frac{du}{dt} \right| dt \quad (2.26)$$

dan dapat di lakukan pendekatan dengan persamaan (2.27).

$$TV_u = \sum_{k=0}^{\frac{t_{sim}}{h}} |u(k+1) - u(k)| \quad (2.27)$$

dimana h merupakan waktu sampling. Kedua kriteria ini kemudian dapat dituliskan menjadi suatu fungsi cost function dengan pendekatan yang lebih sederhana sebagaimana tertulis dalam persamaan (2.28).

$$J = \alpha IAE + \beta TV \quad (2.28)$$

Dengan α dan β merepresentasikan faktor bobot yang menentukan kepentingan relatif dari masing-masing kriteria (de Moura Oliveira et al., 2020).

2.2.5 Transistor

Transistor merupakan salah satu jenis perangkat elektronik semikonduktor yang biasa digunakan sebagai penguat atau saklar dari sinyal listrik. Terdapat beberapa jenis transistor, antara lain *Bipolar transistor (BJT)*, *Field effect transistors (FET)*, *metal-oxide-semiconductor field-effect transistor (MOSFET)*. Selain itu, terdapat pula satu jenis transistor yang merupakan kombinasi dari 2 BJT, yang dinamakan dengan *Darlington Transistor*. Pada *Darlington Transistor*, bagian emitter salah satu BJT terhubung ke base dari BJT lainnya sementara bagian collector keduanya disambungkan. Darlington transistor digunakan untuk menguatkan aliran arus dari BJT satu ke BJT lainnya. Peran dari *Darlington Transistor* sendiri cukup banyak, salah satunya juga digunakan sebagai pengontrol arus pada *charging system*.

(Halaman ini sengaja dikosongkan)

BAB 3 METODOLOGI

3.1 Metode yang digunakan

Pada tugas akhir ini, secara umum digunakan 2 metode yaitu simulasi dan eksperimen. Pada simulasi, akan dilakukan beberapa metode khusus yaitu identifikasi sistem, pemodelan sistem, perancangan *controller*, dan diakhiri dengan pengambilan data simulasi. Sementara itu, pada eksperimen, akan dilakukan metode antara lain perancangan alat, perancangan *controller*, dan pengambilan data percobaan. Metode identifikasi sistem pada simulasi akan dilakukan dengan cara menerapkan *pulse discharging estimation* untuk mendapatkan parameter dari baterai yang digunakan. Perancangan *controller* akan dilakukan dengan menerapkan algoritme PSO untuk optimasi set point dan parameter PI.

3.2 Bahan dan peralatan yang digunakan

Bahan yang digunakan pada tugas akhir sebagian besar merupakan komponen yang terdapat pada *prototype* sistem sebagaimana ditunjukkan oleh Tabel 3.1.

Tabel 3.1 Daftar Bahan yang Digunakan

No.	Nama Bahan	Nilai/Tipe	Keterangan
1.	Power Supply	24 Volt 10A	Sumber tegangan pengecasan
2.	Relay	12V	<i>Switch</i> sistem
3.	Dioda bridge	-	Penyearah tegangan dari trafo
4.	Darlington transistor	TIP142	Aktuator untuk mengontrol arus ke baterai
		TIP122	Aktuator untuk mengontrol drain baterai
5.	Shunt resistor	0.13443Ω	Sense resistor untuk pembacaan arus ke <i>microcontroller</i>
6.	Kapasitor Elco	100V 1000uF	Filter tegangan
		16V 470uF	Filter tegangan
		10V 100uF	Filter ADC
7.	LCD 16x2	-	Display
8.	Voltage regulator	L7805	Penurun tegangan 24DC-5DC
9.	Kapasitor keramik	100nF	Filter tegangan
10.	Resistor	47Ω	-
		100Ω	-
		220Ω	-
		1kΩ	-
		1k2Ω	-
		2k2Ω	-
		4k7Ω	-
		15kΩ	-
		20kΩ	-
11.	Op-Amp	LM358	Penguatan tegangan bacaan dari sense resistor
12.	Dioda	IN4007	Pelindung dari polaritas terbalik baterai
13.	Zener	5v2	Pelindung <i>microcontroller</i> dari tegangan tinggi
14.	ULN2003A	-	-
15.	Optocoupler	-	-
16.	Microcontroller	ATmega32	Spesifikasi 32Mb Flash, 8Kb EEPROM, 16MhZ <i>speed</i> .

Tabel 3.1 Daftar Bahan yang Digunakan (Lanjutan)

No.	Nama Bahan	Nilai/Tipe	Keterangan
17.	Baterai NiMH	12V 10Ah	Plant sistem
18.	Kabel	-	-

Sementara itu, peralatan yang digunakan pada tugas akhir ini meliputi alat serta *software* untuk perancangan sistem, simulasi, pembuatan *prototype*, serta pengambilan data sebagaimana ditunjukkan pada Tabel 3.2.

Tabel 3.2 Daftar Peralatan yang Digunakan

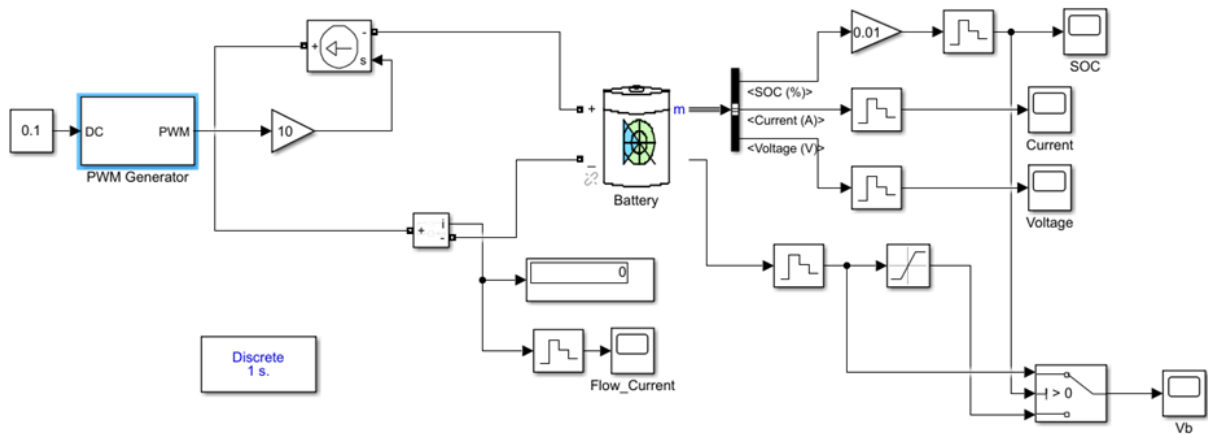
No.	Nama Alat	Keterangan
1.	Laptop	-
2.	MATLAB 2021b	Media untuk melakukan estimasi dan <i>modelling</i>
3.	Simulink	Media untuk melakukan simulasi
4.	Arduino IDE 1.8.19	Media untuk melakukan <i>coding</i> program <i>prototype</i>
5.	Osiloskop	Media untuk pengambilan data arus
7.	Solder	Alat untuk pembuatan <i>prototype</i>
8.	AVR ISP MKVII Programmer	Alat untuk memasukkan program dari Arduino IDE ke dalam <i>microprocessor</i>
9.	Multimeter	Alat ukur untuk kalibrasi

3.3 Urutan pelaksanaan penelitian

Pada tugas akhir ini, pelaksanaan penelitian dimulai dari penentuan model matematika baterai NiMH, perancangan sistem *charger controller*, perancangan metode PSO untuk pengecasan baterai NiMH, perancangan simulasi dengan MATLAB, dan perancangan *prototype charger controller*.

3.3.1 Penentuan Model Matematika Baterai NiMH

Dalam tugas akhir ini, untuk menerapkan algoritme optimasi, diperlukan model matematika dari baterai NiMH dalam bentuk transfer function yang dapat diperoleh melalui persamaan model dari baterai sebagaimana tertera pada dasar teori. Namun, persamaan sistem yang kompleks menyebabkan transfer function sulit diperoleh sehingga digunakan pendekatan lain untuk memperoleh transfer function yaitu menggunakan estimasi parameter sistem dengan menggunakan data *pulse discharging*. Data ini dapat diperoleh melalui simulasi dimana baterai yang digunakan memiliki persamaan serupa dengan yang telah tertera di dasar teori. Adapun rangkaian simulasi untuk memperoleh data ini ditunjukkan oleh Gambar 3.1.



Gambar 3.1 Rangkaian *Simulink pulse discharging*

Hasil data ini kemudian akan diolah dengan menggunakan *Discharge Pulse Estimation MATLAB* dengan beberapa parameter awal sebagaimana tertera pada Tabel 3.1. Parameter awal ini diperoleh dari datasheet baterai NiMH. Dari hasil estimasi, diperoleh data nilai resistansi dan kapasitansi estimasi dari baterai yang kemudian nilai tersebut akan digunakan untuk persamaan (2.1), (2.2), dan (2.6). Model baterai pada tugas akhir ini menggunakan V_T sebagai output dan I_{ch} sebagai input. Namun, pada persamaan (2.6), $I(k)$ yang merupakan arus tersimpan pada baterai memiliki nilai yang berbeda dengan $I_{ch}(k)$ bergantung pada efisiensi η_b atau kemampuan dari baterai dalam menangkap daya yang diberikan. Semakin baik kondisi baterai, nilai efisiensi baterai akan semakin tinggi. Hubungan dari $I(k)$ dan $I_{ch}(k)$ ditunjukkan oleh persamaan (3.1) sehingga persamaan (2.6) berubah menjadi persamaan (3.2).

Tabel 3.3 Parameter awal estimasi

No	Parameter	Nilai
1.	Jumlah pasangan RC	1
2.	Estimasi awal E0	13.032
3.	Batas E0	9.8 – 14.5
4.	Estimasi awal R0	0.012
5.	Batas R0	0.0001 – 0.1
6.	Estimasi awal Rx	0.01
7.	Batas Rx	0.0001 – 0.5
8.	Estimasi awal Tx	100
9.	Batas Tx	75 – 360

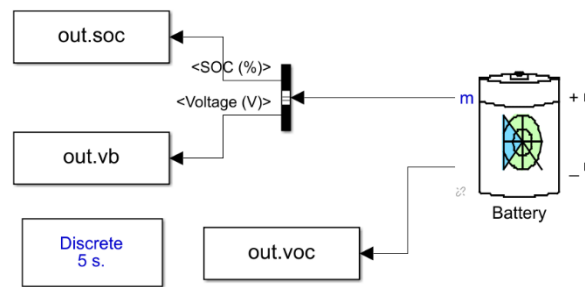
$$I(k) = \eta I_{ch}(k) \quad (3.1)$$

$$SOC(k + 1) = SOC(k) + \frac{\eta_b I_{ch}(k) \Delta t}{3600 Q_m} \quad (3.2)$$

Pada baterai terdapat suatu hubungan pasti antara nilai SOC dengan V_{OC} yang dapat dituliskan dalam bentuk polinomial. Untuk itu, diperlukan tabel data berisi hubungan SOC-OCV. Pada tugas akhir ini, data tersebut diperoleh melalui simulasi dengan jenis baterai yang sama seperti simulasi untuk memperoleh parameter baterai. Adapun rangkaian simulasi ditunjukkan oleh Gambar 3.2 yang dijalankan sesuai dengan *pseudocode* pada Algoritme 1.

ALGORITME 1. Algoritme Nilai SOC-OCV

- Input:** SOC
Output: SOC-OCV
1. **Data :** SOC (1:1:100)
 2. **Set :** posisi awal partikel, kecepatan partikel, $i = 0, j = 0$.
 3. **For** SOC=1:100
 4. **Set :** SOC Baterai simulasi sama dengan SOC
 5. Workspace Simulink menjadi workspace utama
 6. Jalankan simulasi dengan perintah sim untuk mendapatkan output simulasi pengecasan
 7. OCV(SOC) sama dengan output OCV
 8. **end**



Gambar 3.2 Rangkaian *Simulink* Hubungan Voc-SOC

Data dari tabel hubungan SOC-OCV kemudian akan diolah memanfaatkan fungsi *polyfit* pada MATLAB sehingga diperoleh polinomial SOCOCV pada persamaan (3.3) dan (3.4) sehingga persamaan (2.1) dapat diubah menjadi persamaan (3.5).

$$SOCOCV(SOC(k)) = p_1 + p_2 SOC(k) + p_3 SOC^2(k) + \dots + p_n SOC^{n-1}(k) \quad (3.3)$$

$$SOCOCV(SOC(k)) = E_0 + V_{ceq} \quad (3.4)$$

$$V_T(k) = (SOCOCV(SOC(k)) - V_{cp}(k) - I_{ch}(k)R_0) \quad (3.5)$$

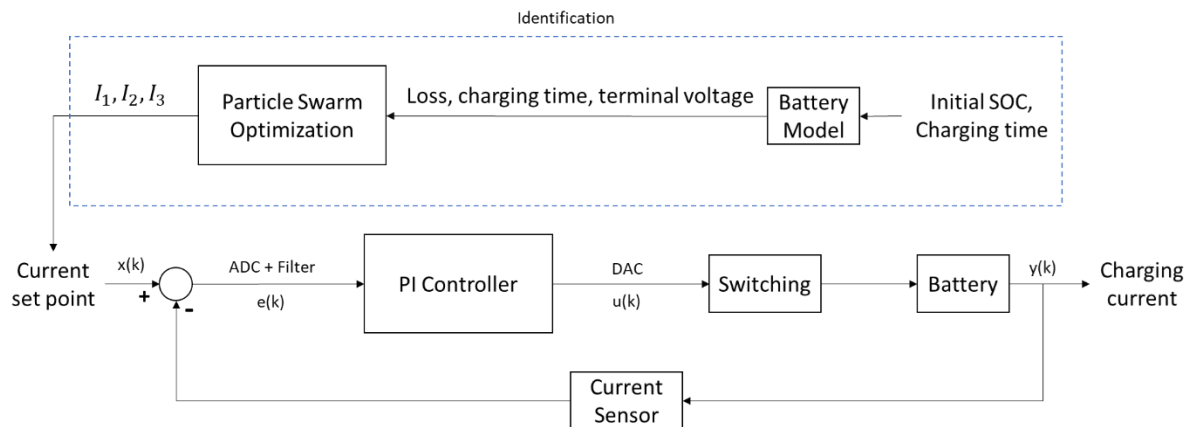
Dengan p_n merupakan konstanta polinomial, dan $n = 1, 2, \dots$ merupakan orde polinomial ditambah 1. Persamaan (3.2) dan (3.5) kemudian dapat dibentuk menjadi suatu persamaan state space dinamis (3.6) dan (3.7) dengan $x_b(k) = [SOC(k) \quad V_{cp}(k)]$; $y(k) = V_T(k)$; serta $u(k) = I_{ch}(k)$

$$\begin{bmatrix} x_{b_1}(k+1) \\ x_{b_2}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{-\Delta t}{R_p C_p}} \end{bmatrix} \begin{bmatrix} x_{b_1}(k) \\ x_{b_2}(k) \end{bmatrix} + \begin{bmatrix} \frac{\eta_b \Delta t}{3600 Q_m} \\ R_p \left(1 - e^{\frac{-\Delta t}{R_p C_p}} \right) \end{bmatrix} u(k) \quad (3.6)$$

$$y(k) = [0 \quad -1] \begin{bmatrix} x_{b_1}(k) \\ x_{b_2}(k) \end{bmatrix} - R_0 u(k) + (SOCOCV(x_{b_1}(k))) \quad (3.7)$$

3.3.2 Perancangan Sistem *Charger Controller*

Pada dasarnya, charger bekerja dengan cara mengalirkan arus dari tegangan yang lebih tinggi ke baterai dengan tegangan yang lebih rendah. Baterai NiMH memiliki karakteristik yang cukup berbeda dengan baterai lainnya dimana baterai ini sangat sensitif terhadap *overcharging* dan juga tegangan puncaknya akan turun ketika penuh. Oleh karena itu untuk jenis rapid *charging*, baterai NiMH memerlukan suatu charger controller dengan arus konstan. Pada tugas akhir ini, arus dijaga konstan dengan menggunakan DAC dari kontroler yang akan mengatur basis dari transistor (*switching*). DAC yang dikeluarkan sendiri akan diatur menggunakan PI Controller yang mendapatkan input berupa error dari set point arus yang ditetapkan dengan bacaan sensor yang ada. Secara garis besar, sistem dapat digambarkan dalam diagram blok yang tertera pada Gambar 3.3.



Gambar 3.3 Diagram Blok Sistem *Charger Controller*

Pada tugas akhir ini, charger controller yang digunakan akan menerapkan 3 set point arus sesuai dengan arahan pada datasheet baterai yang digunakan. Setiap set point arus sendiri akan ditentukan berdasarkan hasil algoritme optimasi. Waktu perubahan set point atau perubahan *stage charging* sendiri ditentukan berdasarkan SOC baterai sehingga sistem juga akan mendeteksi tegangan atau SOC dari baterai.

3.3.3 Perancangan PI Controller pada Sistem

Pada tugas akhir ini, *controller* yang dipilih adalah *controller* PI atau *Proportional Integral*. Kontroler tidak menerapkan diferensial dikarenakan sistem menjadi tidak stabil apabila jenis ini diterapkan. Penetapan parameter K_p dan K_i pada kontroler adalah dengan menggunakan Algoritme PSO dengan kriteria nilai sebagai berikut:

1. K_p maksimum untuk 3 *stage* bernilai 10
2. K_p minimum untuk 3 *stage* bernilai 1
3. K_i maksimum untuk 3 *stage* bernilai 100
4. K_i minimum untuk 3 *stage* bernilai 10

Kriteria kontroler PI optimal adalah menggunakan IAE dengan faktor bobot $\alpha = 0.5$ dan $\beta = 0.5$. Sistem dijalankan dengan rancangan simulasi sebagaimana tertera pada Gambar 3.5.

3.3.4 Perancangan Metode PSO untuk Baterai NiMH

Sesuai dengan dasar teori yang ada, performa sistem pengecasan dapat dilihat dari beberapa objektivitas seperti durasi pengecasan, *loss*, *battery aging*, temperatur, dan lain sebagainya.

Dalam tugas akhir ini, dirancang optimasi pengecasan dengan 3 objektivitas yaitu meminimalkan waktu (T), meminimalkan $loss$ (L), dan memaksimalkan tegangan terminal akhir (V_{TL}) sesuai dengan persamaan (3.8), (3.9), (3.10), (3.11), dan (3.12). (Chen et al., 2021)

$$J_b = \sqrt{\alpha_b J_1^2 + \beta_b J_2^2 + (1 - \alpha_b - \beta_b) J_3^2} \quad (3.8)$$

dengan:

$$J_1 = \frac{T - T_{min}}{T_{max} - T_{min}} \quad (3.9)$$

$$J_2 = \frac{L - L_{min}}{L_{max} - L_{min}} \quad (3.10)$$

$$J_3 = \frac{V_{TL} - V_{Tmax}}{V_{Tmax} - V_{Tmin}} \quad (3.11)$$

$$L(k) = I_{ch}^2(k)(R_0 + R_p)\Delta t \quad (3.12)$$

Dengan T_{max} , T_{min} , L_{max} , L_{min} , V_{Tmax} , dan V_{Tmin} masing-masing adalah waktu pengecasan maksimum, waktu pengecasan minimum, $loss$ maksimum, $loss$ minimum, tegangan terminal maksimum, dan tegangan terminal minimum. Nilai tegangan terminal maksimum dan minimum diperoleh dari nilai OCV untuk SOC 100% dan 0%. Sementara itu waktu pengecasan minimum dan $loss$ maksimum diperoleh dari *running* model untuk arus maksimum sesuai datasheet, dan sebaliknya waktu pengecasan maksimum dan $loss$ minimum diperoleh dari *running* model untuk arus minimum sesuai datasheet. Adapun arus minimum dan maksimum untuk setiap *stage* pengecasan tertera pada Tabel 3.4.

Tabel 3.4 Range Arus setiap Stage

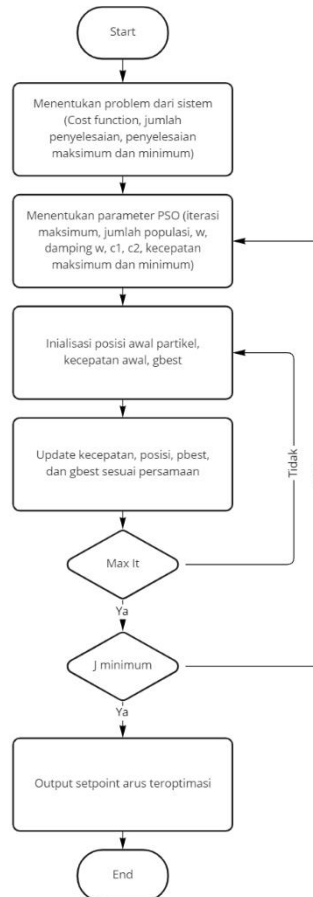
Stage	Arus Minimum (A)	Arus Maksimum (A)
1	2	3
2	5	10
3	0.33	0.5

Sesuai dengan tujuan tugas akhir dimana hasil akhir rancangan *Prototype* diharapkan mampu mengoptimalkan *charging* untuk waktu tertentu yang diinginkan pengguna tanpa mengurangi *lifespan* baterai, pada tugas akhir ini digunakan 3 kondisi untuk memperoleh cost function yaitu:

1. User tidak memasukkan input waktu
Pada kondisi ini, maka diberlakukan 3 objektivitas. Namun, karena tidak terdapat batasan waktu, maka V_{TL} yang diperoleh akan selalu maksimal sehingga J_3 selalu bernilai 0 dan seakan-akan hanya terdapat 2 objektivitas.
2. User memasukkan input waktu dengan $T_{in} > T_{max}$
Pada kondisi ini, karena waktu yang dimiliki user lebih besar dari waktu yang diperlukan maka akan tercapai kondisi seakan-akan user tidak memasukkan batasan waktu (sama seperti kondisi 1)

3. User memasukkan input waktu dengan $T_{min} < T_{in} < T_{max}$ atau $T_{in} < T_{min}$
 Pada kondisi ini, karena waktu yang dimiliki user terbatas, maka $T_{max} = T_{in}$. Dengan
 begini, nilai objektivitas meminimalkan waktu akan selalu bernilai 1 dan menjadi tidak
 berfungsi lagi. Untuk itu, pada kondisi ini, nilai α diatur menjadi 0 sehingga hanya
 terdapat 2 objektivitas yaitu meminimalkan loss dan memaksimalkan V_{TL} dalam kurun
 waktu yang ditentukan user.

Secara umum algoritme PSO yang digunakan dapat digambarkan oleh Gambar 3.4.



Gambar 3.4 Diagram Alir Algoritme PSO

3.3.5 Perancangan Simulasi dengan MATLAB

Pada proses pembuatan rancang bangun *charger controller* dengan Algoritme PSO ini diperlukan suatu simulasi untuk apakah sistem sudah bekerja sesuai yang diinginkan. Dalam proses simulasi, dipilih MATLAB sebagai media dikarenakan aplikasi ini dapat digunakan untuk merancang baik secara elektronika yaitu dengan memanfaatkan SimScape pada Simulink dan juga secara matematis dengan memanfaatkan model-model matematika yang ada. Agar menyerupai rangkaian sebenarnya yang akan digunakan, maka dalam perancangan simulasi digunakan SimScape yang terdiri dari beberapa komponen yaitu:

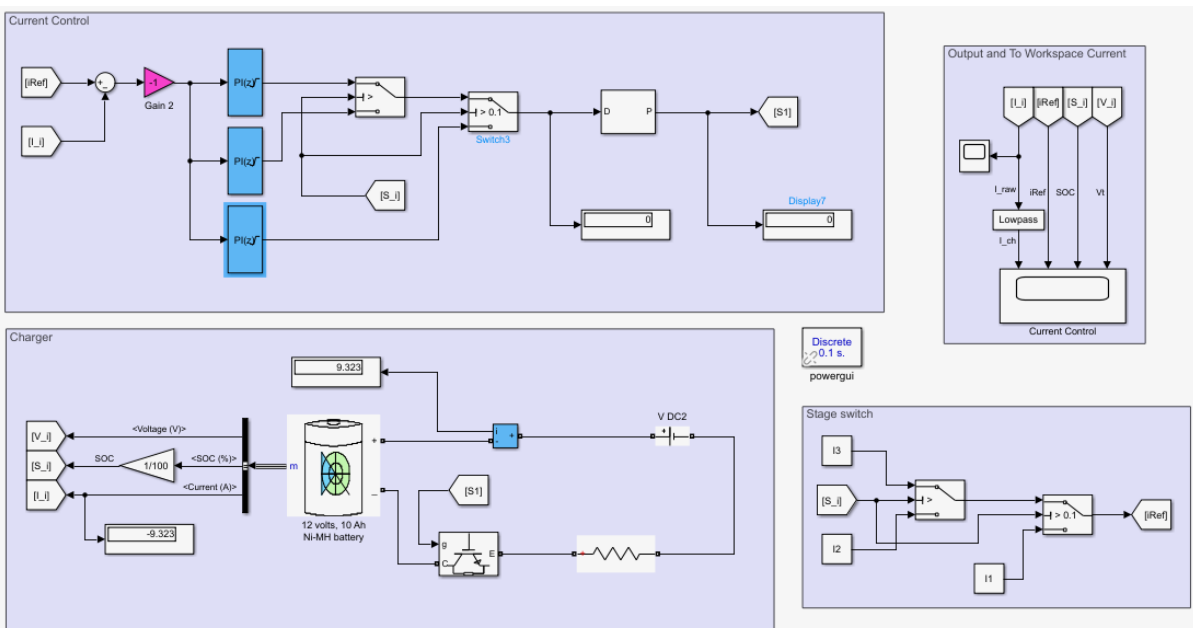
1. IGBT
2. Resistor
3. Sumber tegangan DC
4. Baterai

Pada simulasi, digunakan sumber berupa tegangan DC 30V untuk melakukan pengecasan pada baterai. Untuk memudahkan pendeteksian arus yang melewati baterai selaku *feedback* untuk kontroler, maka bagian positif baterai langsung disambungkan dengan sumber tegangan sementara bagian negatif dari baterai dihubungkan ke *ground* oleh transistor yang dikendalikan oleh kontroler. Dengan begitu, apabila *gate* transistor diberikan sinyal penuh atau terbuka, maka arus yang mengalir akan maksimal dan sebaliknya, apabila *gate* transistor tidak diberi sinyal atau tertutup maka arus tidak akan mengalir ke baterai. Namun, pada simulasi dikarenakan ada beberapa keterbatasan dalam segi komponen dimana tidak terdapat *BJT* maupun *Darlington Transistor* sehingga digunakan IGBT sebagai pengganti sesuai dengan instruksi MATLAB.

Selain itu, pada perancangan simulasi ini juga digunakan komponen Simulink lain sebagai kontroler yang akan mengontrol transistor yang terdiri dari komponen yaitu:

1. PI controller diskrit
2. PWM generator

PI controller diskrit digunakan untuk mengontrol arus sesuai dengan set point yang diberikan. Sesuai dengan penjelasan sebelumnya, set point dari sistem akan ditentukan melalui algoritme PSO. Namun, seperti yang telah dijelaskan sebelumnya terkait keterbatasan komponen MATLAB, IGBT hanya bisa di atur dengan sinyal PWM sehingga pada simulasi tidak digunakan DAC melainkan menggunakan PWM yang kemudian akan dilewatkan melalui *low pass filter* untuk mendapatkan hasil hitungan analog (Microchip Technology Inc., 1997). Secara keseluruhan bentuk rancangan dari simulasi ini ditunjukkan oleh Gambar 3.5.

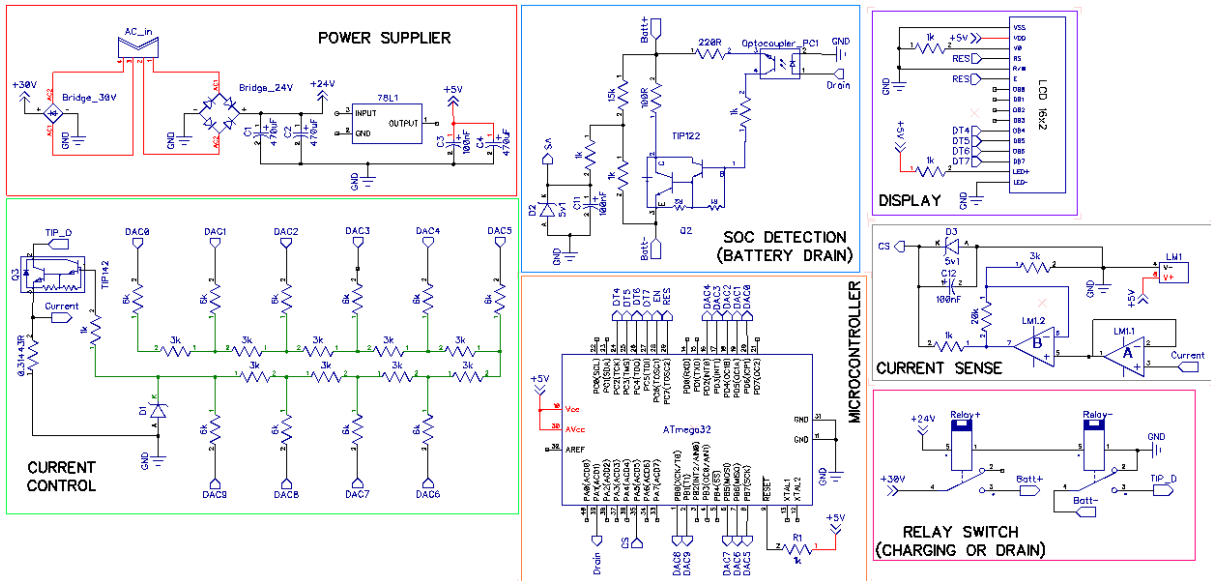


Gambar 3.5 Rancangan Simulasi dengan Simulink MATLAB

Sementara itu, program PSO yang digunakan sendiri dipisah menjadi 3 program yaitu 1 main program dengan 2 sub-program. Adapun sub-program pertama yaitu program algoritme PSO itu sendiri sementara sub-program kedua berisikan *cost function* sistem. Main program digunakan untuk mendefinisikan permasalahan, parameter, dan solusi yang ingin dicari.

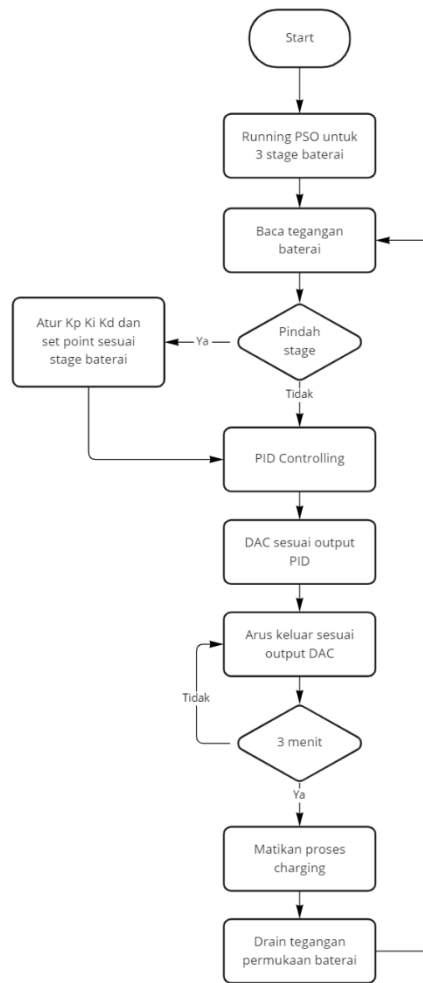
3.3.6 Perancangan *Prototype Charger Controller*

Prototype charger controller pada tugas akhir ini dirancang sedemikian rupa mengikuti perancangan simulasi yang telah dilakukan. Namun, untuk memungkinkan sistem bekerja sesuai yang diharapkan dilakukan pula penyesuaian-penyesuaian pada komponen yang digunakan. Berbeda dengan simulasi, pada *Prototype* sumber yang digunakan dirancang menggunakan sumber tegangan AC yang kemudian diolah menjadi tegangan DC dengan memanfaatkan *power supply switching* 24V 10A. Adapun skema rangkaian untuk *prototype* dapat dilihat pada Gambar 3.6.



Gambar 3.6 Skema Rangkaian *Prototype Charger Controller*

Pada tugas akhir ini, charger controller dirancang agar dapat melakukan perhitungan optimasi sesuai algoritme PSO yang telah dijelaskan pada bagian sebelumnya. Secara umum, cara kerja dari alat ini ditunjukkan pada Gambar 3.7



Gambar 3.7 Diagram Alir Cara Kerja Sistem

3.3.7 Pengambilan Data Percobaan

Pada tugas akhir ini, pemerolehan data percobaan akan dipisah menjadi 2 yaitu secara simulasi dan secara eksperimen. Pada eksperimen, parameter PSO yang digunakan akan didasarkan pada parameter PSO terbaik menurut simulasi pada MATLAB untuk baterai dengan anggapan kondisi awal SOC 0% dan tanpa waktu input dari pengguna. Adapun parameter terbaik ini akan didapatkan dari menjalankan program optimasi sebanyak 5 kali untuk 80 kombinasi parameter yang terdiri dari 5 iterasi (3, 5, 10, 50, 100), 3 populasi (3, 5, 10), 2 c_1 (0.5, 1), 2 c_2 (0.5, 1), dan 2 w (0.8, 0.9). Data *cost* dari hasil setiap kombinasi parameter dalam 1 kali *running* program akan di rata-rata dan parameter dengan rata-rata nilai *cost* paling kecil akan dipilih sebagai parameter PSO dalam seluruh rangkaian percobaan. Algoritme PSO dengan parameter yang telah di tentukan sebelumnya akan digunakan untuk memperoleh data percobaan baik secara simulasi maupun eksperimen. Pada simulasi, rancangan *Simulink* yang telah dijelaskan sebelumnya akan dijalankan dengan Algoritme 2.

ALGORITME 2. Algoritme Pengambilan Data Simulasi

Input: Parameter PSO, SOC awal, Kapasitas baterai, Waktu input

Output : Set point arus, Cost, Loss, Waktu pengecasan, Tegangan akhir

1. Inisialisasi parameter PSO dan kondisi awal baterai
2. **Set :** posisi awal partikel, kecepatan partikel, $i = 0, j = 0$.

3. g_{best} sama dengan posisi awal
4. **If** iterasi ke- i kurang dari iterasi maksimal
5. **If** populasi ke- j kurang dari jumlah populasi
6. Update kecepatan partikel
7. Aplikasikan limit kecepatan
8. Update posisi partikel
9. Aplikasikan limit posisi
10. Evaluasi $cost$
11. **If** $cost$ lebih kecil dari $cost$ populasi sebelumnya
12. p_{best} sama dengan posisi saat ini ($x(k)$)
13. **end**
14. **end**
15. **If** $cost$ untuk p_{best} lebih kecil dari $cost$ iterasi sebelumnya
16. g_{best} sama dengan posisi p_{best}
17. **end**
18. **end**
19. Setpoint arus sama dengan g_{best}
20. Hitung loss, waktu pengecasan, dan tegangan terminal akhir untuk setpoint arus
21. **Set** : SOC Baterai simulasi sama dengan SOC inisial
22. Workspace Simulink menjadi workspace utama
23. Jalankan simulasi dengan perintah sim untuk mendapatkan output simulasi pengecasan

Sementara itu, pada eksperimen, pengambilan data akan dilakukan dengan memanfaatkan media serial komunikasi dan fitur *recorder* pada osiloskop digital sebagaimana tertera pada Gambar 3.8. Media serial komunikasi yang digunakan adalah Arduino dengan menambahkan rangkaian proteksi dan *voltage divider* sebagai pengukur tegangan baterai selama pengecasan sebagaimana ditunjukkan oleh Gambar 3.9. Data serial yang dikirimkan Arduino kemudian akan diterima dan disimpan oleh MATLAB menggunakan Algoritme 4.2.

ALGORITME 3. Algoritme Pengambilan Data Serial *Prototype*

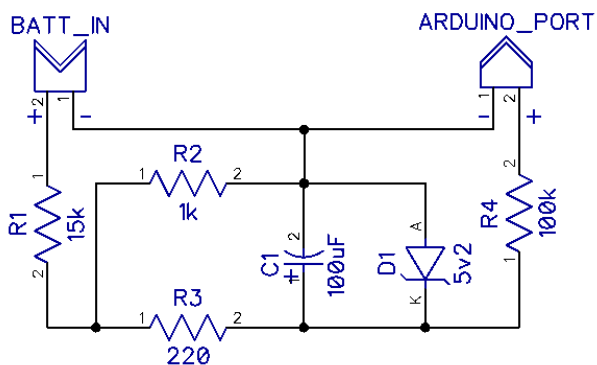
Input: Nomor Port Serial, Baud Rate Serial, Waktu Sampling

Output : Tegangan baterai *Prototype*

1. Inialisasi parameter serial
 2. Buka port serial
 3. **Set** : nilai $i=1$
 4. **while true**
 5. Simpan data serial dalam array tegangan ke- i
 6. Update nilai i dengan $i=i+1$
 7. Pause selama waktu sampling
 8. **end**
 9. Tutup port serial
-



Gambar 3.8 Fitur Recorder pada Osiloskop Digital



Gambar 3.9 Skema (kiri) dan Rangkaian (kanan) Tambahan

Sementara itu, osiloskop digital akan digunakan sebagai media untuk mendapatkan arus yang masuk ke baterai. Osiloskop hanya mampu mendeteksi tegangan sehingga hasil akhir data percobaan nantinya akan diolah kembali secara matematis untuk mendapatkan nilai arus sebenarnya.

BAB 4 HASIL DAN PEMBAHASAN

4.1 Estimasi parameter baterai

Dari proses estimasi menggunakan *pulse discharging* dengan nilai inialisasi parameter awal sebagaimana tertera pada Tabel 3.2, diperoleh nilai parameter sesuai yang telah tertera pada Tabel 4.1.

Tabel 4.1 Nilai parameter sistem

No	Parameter	Nilai	Satuan
1.	R_0	0.0846	Ohm
2.	R_p	0.009	Ohm
3.	C_p	2.5508e+04	Farad
4.	E_0	13.0232	Volt
5.	η	0.9	-
6.	Δt	1	Detik
7.	Q_{max}	10	Ampere Hour (Ah)
8.	p_1	-4526.3365	-
9.	p_2	25264.6955	-
10.	p_3	-60569.6609	-
11.	p_4	81886.4051	-
12.	p_5	-68849.1071	-
13.	p_6	37452.6285	-
14.	p_7	-13313.2871	-
15.	p_8	3067.3029	-
16.	p_9	-451.0154	-
17.	p_{10}	42.5251	-
18.	p_{11}	10.3660	-

Parameter R_0 , R_p , dan C_1 merupakan nilai estimasi resistansi dan kapasitansi ekuivalen pada baterai. Ketiga nilai ini sangat berpengaruh pada proses pengoptimalan sistem yang mempertimbangkan *loss* sebagai objektivitas. Hal ini disebabkan oleh perhitungan *loss* yang melibatkan penjumlahan nilai resistansi internal R_0 dan resistansi cabang RC R_p , seperti yang telah dijelaskan oleh persamaan (3.12).

Parameter pada tugas akhir ini lainnya adalah η yaitu nilai efisiensi coulomb, dimana η berada diantara nilai 0 dan 1 serta bernilai semakin besar apabila kondisi baterai semakin baik. Baterai NiMH sesungguhnya kemudian di *charge* dengan arus 5A selama satu jam. Secara teori untuk jenis baterai 10Ah maka baterai akan terisi sebanyak 50%. Namun, baterai hanya terisi sebanyak 45% sehingga dipilih nilai η yaitu 0.9. Sementara itu, nilai p_1 hingga p_{11} merupakan konstanta polinomial orde 10 yang diperoleh dari *polyfit* antara SOC dan OCV baterai NiMH.

4.2 Pengujian algoritme PSO

Pada algoritme PSO, dipilih beberapa parameter PSO yang kemudian diterapkan untuk memperoleh nilai set point arus setiap *stage*. Pada pengujian algoritme PSO, optimasi dilakukan untuk kondisi dimana tidak terdapat masukan berupa waktu maksimal pengecasan yang ada serta kondisi awal SOC baterai dianggap 0%. Program PSO dijalankan sebanyak 5 kali untuk memastikan keakuratan setiap kombinasi parameter PSO yang kemudian hasilnya akan dirata-rata untuk diperoleh 1 kombinasi parameter PSO yang dianggap terbaik. Nilai *cost* rata-rata untuk 80 kombinasi tersebut tertera pada Tabel 4.2.

Tabel 4.2 Rata-Rata Cost untuk Kombinasi Parameter

Kombinasi ke-	Iter	Pop	C1	C2	w	wd	Cost	Running Time	Objektivitas		
1	3	5	1	1	0.8	0.99	0.41573503	1.10420958	0.788569		
2				0.5			0.39942949	1.07440558	0.59886		
3				0.5			1	0.41219421	1.0754233	0.747373	
4							0.5	0.38336718	1.0943394	0.411982	
5			1	0.9			1	0.40316872	1.14873353	0.642365	
6							0.5	0.39352271	1.13334243	0.530138	
7							0.5	1	0.42949698	1.17729478	0.948683
8								0.5	0.40658343	1.18137855	0.682094
9		10	0.8		1		1	0.3756702	2.31987168	0.322433	
10							0.5	0.36859857	2.16761395	0.240157	
11					0.5		1	0.39606427	2.2198507	0.559708	
12							0.5	0.39242834	2.22331655	0.517406	
13			1	0.9	1		0.37760328	2.21970423	0.344923		
14					0.5		0.3795406	2.39126383	0.367463		
15					0.5		1	0.38073488	2.7642023	0.381359	
16							0.5	0.38524789	2.55574955	0.433865	
17	5	5	1		1	0.99	0.38657365	1.77796605	0.449288		
18					0.5		0.39810265	1.77182248	0.583424		
19					0.5		1	0.38603685	1.88523525	0.443043	
20							0.5	0.40168786	1.75361638	0.625136	
21			1	0.9	1		0.3797163	1.7904063	0.369506		
22					0.5		0.38104073	1.80168153	0.384915		
23					0.5		1	0.37602764	1.90521893	0.32659	
24							0.5	0.39282617	1.71441743	0.522034	
25		10	0.8		1		1	0.38078021	3.67802928	0.381889	
26							0.5	0.35764326	3.34417823	0.112709	
27					0.5		1	0.38063993	3.2938488	0.380256	
28							0.5	0.36485983	3.274709	0.196664	
29			1	0.9	1		0.35743193	3.4005888	0.110251		
30					0.5		0.35390211	3.47166005	0.069194		
31					0.5		1	0.36593676	3.39662365	0.209194	
32							0.5	0.36021116	3.3753212	0.142583	
33	10	5	1		1	0.99	0.37325774	2.97210823	0.294367		
34					0.5		0.37977892	3.14205843	0.370238		
35					0.5		1	0.36351508	3.1825394	0.181019	
36							0.5	0.3657245	3.5779549	0.206726	
37			1	0.9	1		0.36523387	3.285401	0.201016		
38					0.5		0.37706331	3.16752273	0.338643		
39					0.5		1	0.3631606	3.16243773	0.176895	
40							0.5	0.35108982	3.11741668	0.036484	
41		10	0.8		1		1	0.36237878	7.76512948	0.167866	
42							0.5	0.35492899	8.2073625	0.081291	
43					0.5		1	0.35483062	6.15628185	0.080063	

Tabel 4.2 Rata-Rata Cost untuk Kombinasi Parameter (Lanjutan)

Kombinasi ke-	Iter	Pop	C1	C2	w	wd	Cost	Running Time	Objektivitas	
44			1	0.5	0.9		0.34864106	5.8669883	0.008745	
45				1			0.34931553	5.87552905	0.016217	
46			0.5	0.35445068			6.24662043	0.075651		
47			1	0.35525108			5.77717543	0.084937		
48			0.5	0.34841148			6.30769745	0.028967		
49	50	5	1	0.5	0.8	0.99	0.37466011	12.9692461	0.310804	
50				1			0.34796257	13.388357	0.009115	
51			0.5	0.34796832			13.1875919	0.00897		
52			1	0.35653051			13.7397715	0.100188		
53			0.5	0.34862339			13.2598739	0.011895		
54		1	0.5	0.5	0.9		0.34796209	13.0118662	0.008841	
55							0.35166113	12.8586492	0.04397	
56		0.5	0.5	0.5	0.9		0.34797027	13.2030941	0.008981	
57							0.35021416	25.8556081	0.031958	
58		10	1	0.5	0.8		0.99	0.34795793	25.9323208	0.01827
59				1				0.35036765	26.9576504	0.033886
60			0.5	0.34795749	31.1356834			0.022067		
61			1	0.34796003	30.2718042			0.021437		
62			0.5	0.34795875	28.414995			0.020081		
63		1	0.5	0.5	0.9		0.34795898	28.7890421	0.020354	
64	0.3479608					29.7565143	0.02106			
65	100	5	1	0.5	0.8	0.99	0.35420061	28.3076304	0.075345	
66				1			0.34798037	28.807782	0.02037	
67			0.5	0.3539048			27.1423936	0.071801		
68			1	0.34798836			29.1111159	0.020593		
69			0.5	0.34895584			26.5308642	0.022022		
70		1	0.5	0.5	0.9		0.34796121	26.98908	0.019041	
71							0.3479581	30.2665841	0.021433	
72		0.5	0.5	0.5	0.9		0.34795757	31.1190547	0.022055	
73							0.34795748	59.1815494	0.042534	
74		1	0.5	0.5	0.8		0.34795878	52.7814433	0.037864	
75							0.35262291	48.9527991	0.064628	
76		0.5	0.5	0.5	0.9		0.34795738	49.1840326	0.035238	
77							0.34795735	52.2646237	0.037486	
78		1	0.5	0.5	0.9		0.34795703	51.0110242	0.036572	
79							0.34795744	47.0249385	0.033663	
80	0.5	0.34795697	47.9973003	0.034372						

Algoritme PSO atau Particle Swarm Optimization merupakan algoritme cerdas (*intelligent algorithm*) yang sangat bergantung pada parameter yang digunakan meliputi jumlah iterasi dan populasi, nilai momen inersia, serta nilai faktor acak. Pemilihan parameter ini cukup mempengaruhi performa dari algoritme PSO. Nilai parameter PSO pada Tabel 4.2 menunjukkan bahwa nilai c_1 , c_2 , dan w yang semakin besar memberikan hasil optimasi yang semakin baik dan berlaku sebaliknya. Hal ini disebabkan karena dengan semakin besarnya parameter c_1 , c_2 dan w maka partikel akan semakin cepat bergerak mencari titik optimal. Sementara itu, dari tabel terlihat bahwa semakin banyak iterasi maka nilai *cost* akan semakin kecil yang menandakan sistem semakin optimal. Namun, dari segi waktu, semakin banyak iterasi menyebabkan waktu eksekusi optimasi semakin lama. Waktu ini tidak akan terlalu berpengaruh apabila sistem tidak mempertimbangkan waktu. Namun, untuk sistem pengecasan dengan input waktu, *running time* dari PSO ini juga dipertimbangkan. Dikarenakan fokus pada

tugas akhir ini hanya mempersingkat waktu pengecasan, maka dipilih faktor bobot untuk objektivitas *running time* hanya 0.1 sementara *cost* 0.9 yang menandakan bahwa nilai *cost* lebih penting daripada *running time*. Dari Tabel 4.2 terlihat bahwa dengan objektivitas ini, diperoleh parameter paling optimal pada kombinasi ke 44 yaitu 10 iterasi, 10 populasi, momen inersia w 0.8, konstanta c_1 1, konstanta c_2 1, dan damping inersia 0.99 dengan nilai *cost* 0.3486 dan *running time* 5.87s.

4.3 Hasil Desain PI Controller

Dari proses *tuning* untuk perolehan parameter PI Controller dengan menggunakan Algoritme PSO, diperoleh nilai parameter untuk 3 *stage* sebagai berikut:

Tabel 4.3 Parameter PI Controller untuk setiap Stage Pengecasan

Stage	Kp	Ki
1	8.2582	61.1057
2	9.0094	67.4419
3	5.4203	48.9966

Tabel 4.3 menunjukkan bahwa nilai Kp dan Ki memiliki besaran yang berbanding lurus dengan batas *setpoint* setiap *stage* dimana *stage* ketiga memiliki rentang *setpoint* yang paling kecil yaitu 0.33A-0.5A, diikuti oleh *stage* pertama dengan rentang *setpoint* 2A-3A, dan terakhir dengan *stage* ketiga dengan rentang *setpoint* arus paling besar yaitu 5A-10A. Nilai Kp dan Ki ini berbanding lurus dikarenakan untuk *setpoint* yang lebih besar diperlukan respons sistem yang lebih cepat. Dengan begitu, nilai Kp akan semakin besar. Namun, dengan semakin besarnya nilai Kp maka *error steady state* dari sistem juga akan semakin besar yang dapat dikompensasi dengan nilai parameter Ki. Oleh karena itu, seiring dengan naiknya nilai Kp, Ki juga akan meningkat.

4.4 Hasil pengujian Simulasi MATLAB

Pengujian simulasi MATLAB dilakukan dengan menentukan kondisi pengecasan baterai sebagaimana tertera pada Tabel 4.4.

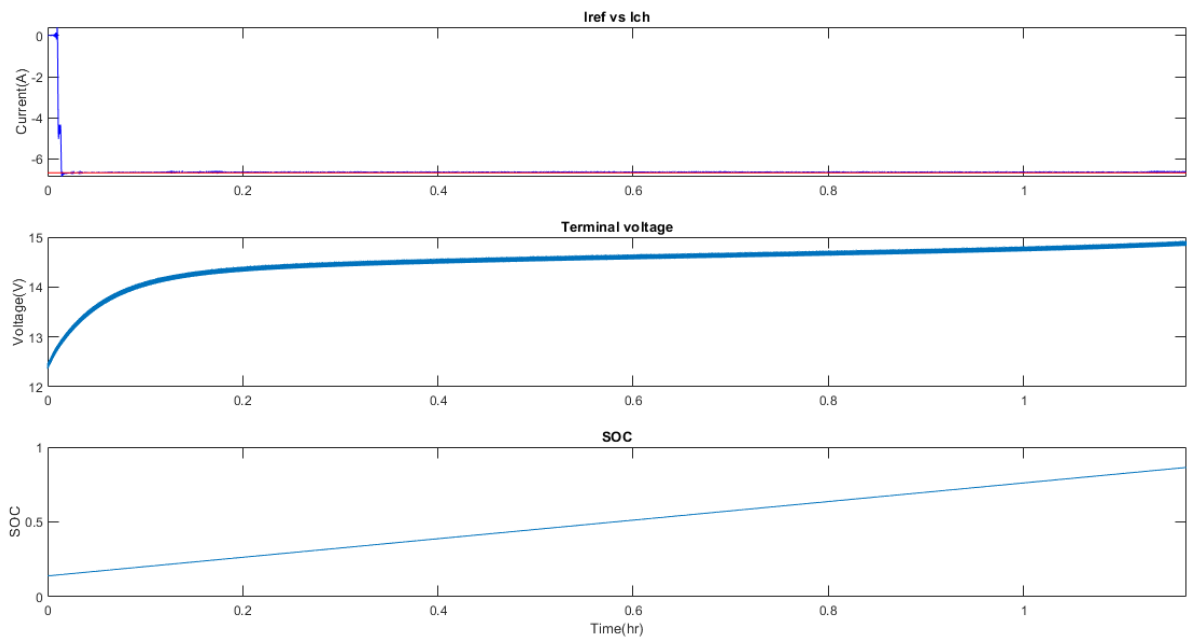
Tabel 4.4 Parameter Kondisi Pengecasan

Kondisi	SOC Awal	Waktu Input
1.	14%	4200 s (1 jam 10 menit)
2.	6%	6000 s (1 jam 40 menit)
3.	9%	-

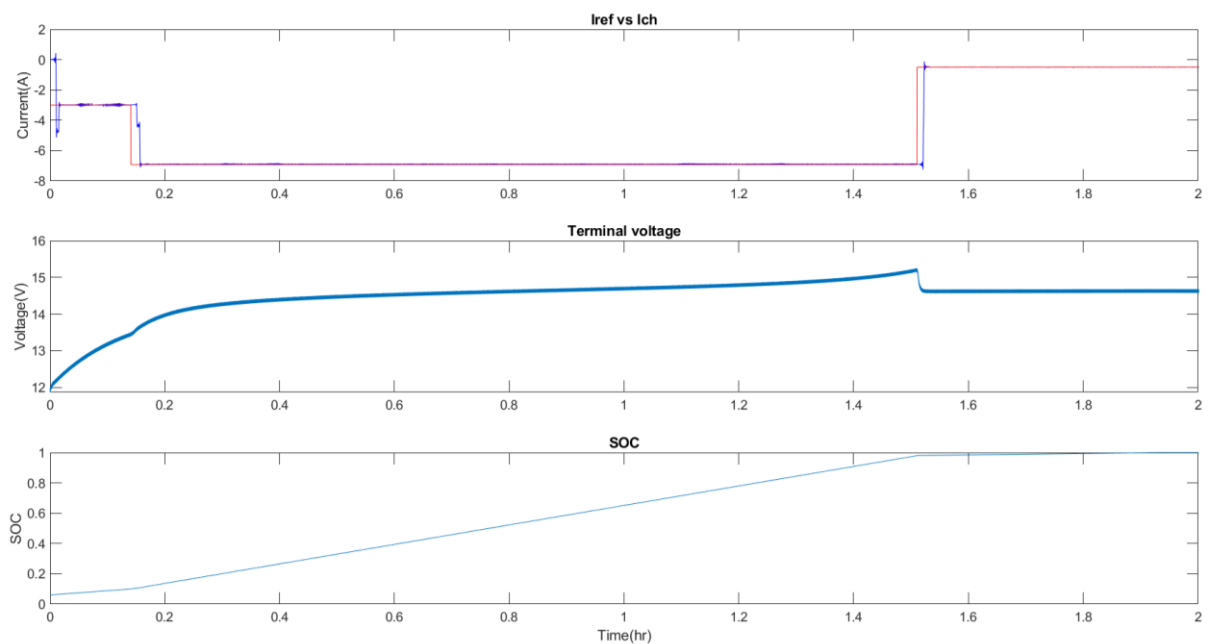
Pada pengujian simulasi MATLAB, diperoleh data hasil optimasi dan data pengecasan pada Tabel 4.3. Sementara itu, data grafik pengecasan untuk kondisi 1, 2, dan 3, secara berurutan ditunjukkan oleh Gambar 4.1, 4.2, dan 4.3. Analisa dilakukan dengan membandingkan perhitungan optimasi dengan keadaan sebenarnya pada simulasi.

Tabel 4.5 Data Pengujian Simulasi

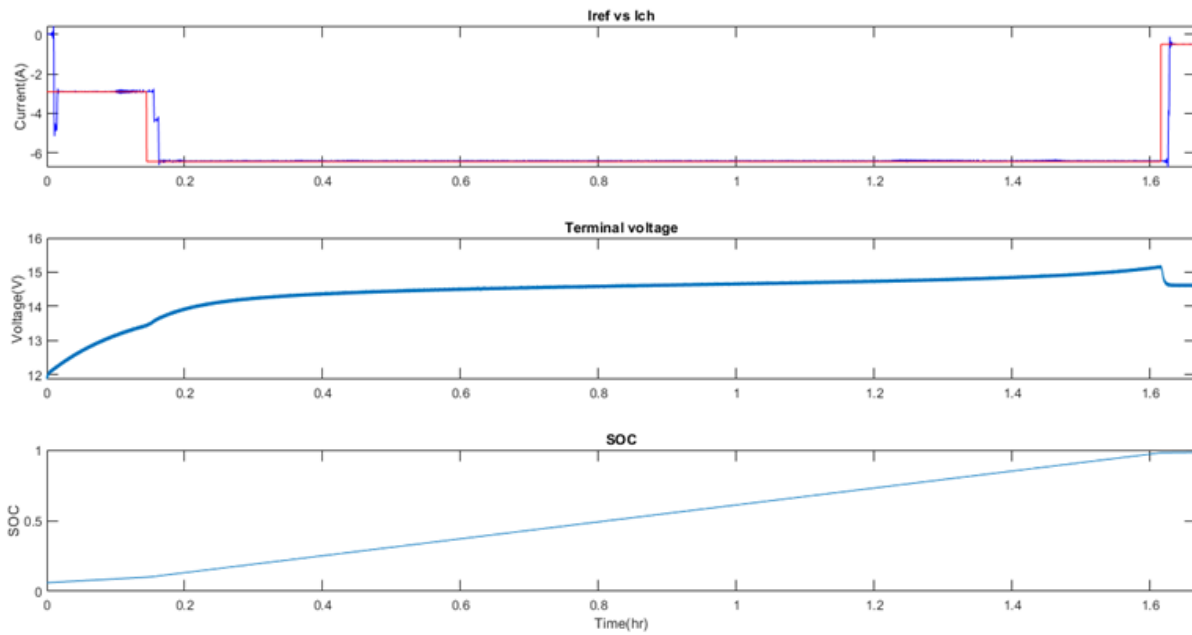
Kondisi	I1 (A)	I2 (A)	I3 (A)	Waktu (s)	SOC Final (%)
Hasil Optimasi					
1.	2.32	6.685	0.38	4200	84
2.	2.917	6.449	0.5	6000	97.9
3.	3	6.926	0.5	7205	100
Data Pegecasan					
1.	-	6.58	-	4200	86
2.	2.738	6.381	1.869	6000	98.2
3.	2.81	6.85	0.647	7205	100



Gambar 4.1 Grafik Pegecasan Kondisi 1 pada Simulasi MATLAB



Gambar 4.2 Grafik Pegecasan Kondisi 2 pada Simulasi MATLAB



Gambar 4.3 Grafik Pengecasan Kondisi 3 pada Simulasi MATLAB

Berdasarkan Tabel 4.4, pada kondisi pertama SOC dari simulasi MATLAB menunjukkan nilai yang lebih tinggi sebanyak 2%, dibandingkan dengan SOC yang dihitung dari proses optimasi. Sementara itu, arus dari simulasi memiliki error sebesar 0.105 Ampere lebih rendah. Pada kondisi pengecasan kedua, terlihat bahwa kembali terdapat error sebesar 0.179 Ampere, 0.068 Ampere, dan 1.369 Ampere untuk secara berurutan *stage* 1, 2 dan 3. Dari grafik pengecasan kondisi kedua, terlihat bahwa error ini disebabkan oleh adanya delay dari respons sistem saat terjadi perubahan *stage*. Ketika pengecasan beralih dari *stage* pertama ke *stage* kedua, set point arus berubah dari 2 menjadi 5. Selain itu, waktu pengecasan *stage* kedua adalah yang paling lama sehingga delay sistem sebanyak 1 menit tidak terlalu berdampak. Namun, ketika perubahan *stage* kedua menuju *stage* ketiga, selisih perbedaan *set point* arus keduanya sangat tinggi. Waktu pengecasan *stage* 3 juga yang paling cepat dilalui sehingga delay 30 detik sangat berpengaruh pada hasil bacaan data pengecasan. Pada grafik juga terlihat bahwa terdapat overshoot yang cukup besar untuk *stage* pertama yaitu sebesar 70%, namun tidak untuk *stage* kedua dan ketiga. Pada pengecasan baterai, sesuai dengan dasar teori, PI Controller harus dirancang memiliki step response berupa respons underdamp. Hal ini dikarenakan overshoot arus yang terlalu tinggi dapat menyebabkan penurunan usia baterai. Fakta bahwa masih terdapat overshoot menandakan masih terdapatnya ketidaksesuaian dalam perancangan dan penentuan nilai parameter PI pada proses designing controller untuk proses pengecasan *stage* pertama. Selain error arus, pada kondisi kedua juga terdapat error SOC dimana pada perhitungan optimasi, diprediksi tegangan baterai akan mencapai 0.979 sementara hasil simulasi menunjukkan error sebesar 0.003 atau 0.3%.

Terakhir, pada kondisi ketiga, terlihat bahwa error arus menunjukkan angka 0.107 Ampere, 0.076 Ampere, dan 0.147 Ampere untuk masing-masing *stage* 1, 2 dan 3. Sama seperti kondisi kedua, error ini disebabkan oleh adanya delay saat terjadi perpindahan *stage*. Namun, pada kondisi ini terlihat bahwa SOC akhir dari perhitungan optimasi dan simulasi sudah sesuai yaitu mencapai angka 1 dengan batas waktu yang sama yaitu 7205 detik atau 1 jam 5 detik. Dari

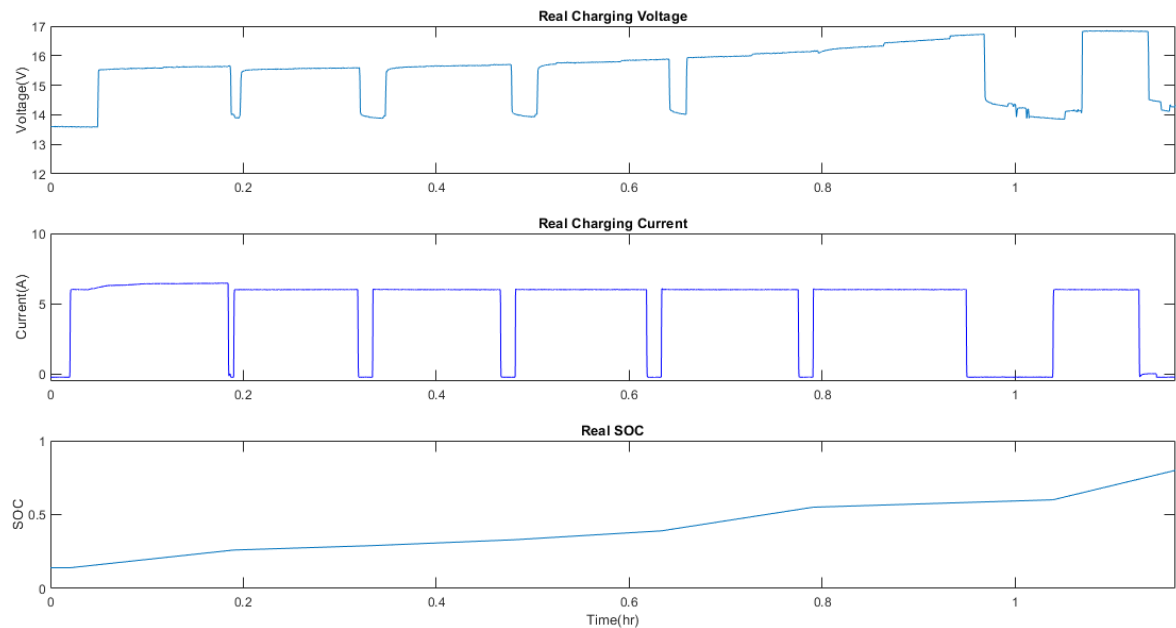
ketiga kondisi pengecasan, terlihat bahwa error SOC yang semakin kecil saat input waktu dinaikkan.

4.5 Hasil pengujian *Prototype*

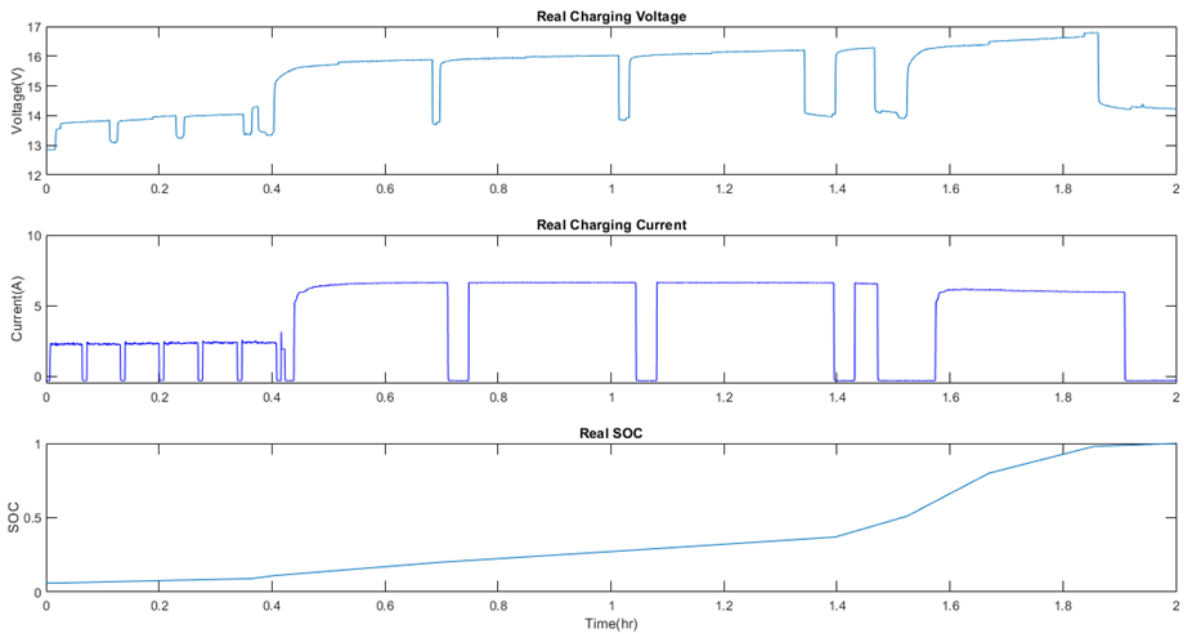
Pada pengujian *prototype*, sistem dijalankan pada 3 kondisi yang sama seperti yang telah tertera pada Tabel 4.3. Dari pengujian tersebut, diperoleh data hasil optimasi dan data pengecasan pada Tabel 4.4. Sementara itu, data grafik pengecasan untuk kondisi 1, 2, dan 3, secara berurutan ditunjukkan oleh Gambar 4.1, 4.2, dan 4.3. Analisa kembali dilakukan dengan membandingkan perhitungan optimasi dan pengecasan *Prototype*.

Tabel 4.6 Data Pengujian *Prototype*

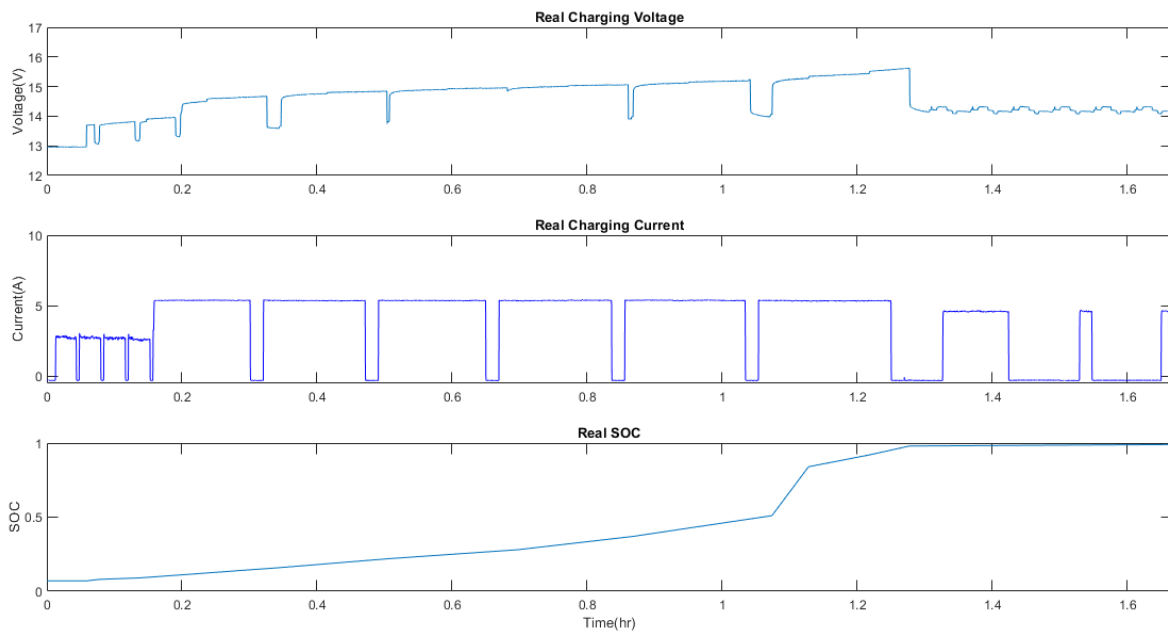
Kondisi	I1 (A)	I2 (A)	I3 (A)	Waktu (s)	SOC Final (%)
Hasil Optimasi					
1.	2.63	5.99	0.46	4200	76
2.	2.8	5.36	0.33	6000	98
3.	2.4	6.6	0.33	8148	100
Data Pengecasan					
1.	-	6.05	-	4200	80
2.	2.7	5.37	4.5	6000	98
3.	2.35	6.43	0.33	7202	100



Gambar 4.4 Grafik Pengecasan Kondisi 1 pada *Prototype*



Gambar 4.5 Grafik Pengisian Kondisi 2 pada *Prototype*



Gambar 4.6 Grafik Pengisian Kondisi 3 pada *Prototype*

Berdasarkan Tabel 4.5, pada kondisi pertama SOC pada pengujian *prototype* menunjukkan angka 4% lebih tinggi dibandingkan dengan perhitungan optimasi yang telah dilakukan. Sementara itu, dalam 1 siklus pengisian pertama, terlihat bahwa arus sistem masih tidak stabil dan bernilai lebih tinggi sebesar 0.5 ampere. Namun, mulai dari siklus kedua, arus stabil dan hanya memiliki error steady state sebesar 0.01 ampere. Pada grafik terlihat bahwa pada waktu ke 1 jam, tegangan bacaan dari baterai menjadi tidak stabil. Pada detik yang sama arus bernilai -0.3 ampere yang berarti sistem tengah melalui proses pembacaan SOC, yang artinya tegangan

seharusnya tetap stabil tanpa naik ataupun turun. Hal ini dapat disebabkan oleh error bacaan ataupun *noise* dari luar.

Pada kondisi pengecasan kedua, SOC hasil perhitungan optimasi dan SOC akhir *real system* telah menunjukkan angka yang sama yaitu 0.98 atau 98%. Dari grafik, terlihat bahwa untuk *stage* pertama sistem masih terlihat mengalami overshoot. Dari grafik juga terlihat bahwa tegangan mengalami kenaikan yang stabil seiring dengan penambahan waktu dan kenaikan SOC. Dari sisi arus, error yang terbentuk juga sangat kecil yaitu bernilai 0.01 ampere. Pada kondisi ini, terlihat ada kesalahan bacaan arus pada *stage* ketiga dimana tegangan telah menunjukkan masuknya sistem ke *stage* 3 atau *trickle* sementara bacaan arus menunjukkan nilai turun dibandingkan arus *stage* kedua, namun sangat tinggi dibandingkan *stage* ketiga. Bacaan grafik arus ini juga tidak sesuai dengan bacaan yang tertera pada LCD dan amperemeter analog di *prototype* sehingga kesalahan bacaan ini kemungkinan besar diakibatkan kesalahan bacaan dari alat ukur osiloskop.

Terakhir, pada kondisi ketiga, terlihat bahwa sistem memiliki respons yang hampir serupa dengan kondisi kedua dimana pada *stage* pertama, masih terdapat overshoot dari arus sebesar 2%, namun sistem telah stabil tanpa overshoot untuk *stage* kedua dan ketiga, dengan error masing-masing untuk setiap *stage* adalah 0.01 ampere. Pada kondisi ketiga ini, dari data pada Tabel 4.5 terlihat bahwa terdapat perbedaan yang cukup besar terkait dengan waktu pengecasan. Menurut perhitungan optimasi, baterai akan penuh dalam waktu 8148 detik atau 2 jam 15 menit 48 detik sementara pada keadaan sesungguhnya baterai telah penuh dalam kurun waktu 2 jam 2 detik. Error waktu yang terjadi adalah 13 menit 20 detik, atau senilai 10.05%. Nilai error ini cukup besar untuk error perhitungan waktu.

Dari beberapa data yang ada, tampak bahwa model sistem dalam perhitungan SOC, dan desain controller PI sudah sangat mendekati dengan keadaan sesungguhnya dari sistem untuk kondisi kedua dan ketiga pada *stage* kedua. Sementara itu, desain *controller* PI pada *stage* pertama tampak masih terdapat kekurangan yang menyebabkan timbulnya overshoot pada arus. Di sisi lain, pada *stage* ketiga yang mana hanya terdapat di pengecasan kondisi ketiga, terlihat bahwa sebenarnya *modelling* masih kurang sesuai, melihat waktu pemenuhan baterai hanya sampai detik ke 7205 sementara dari perhitungan memerlukan waktu hingga 8148.

4.6 Perbandingan Simulasi dan *Prototype*

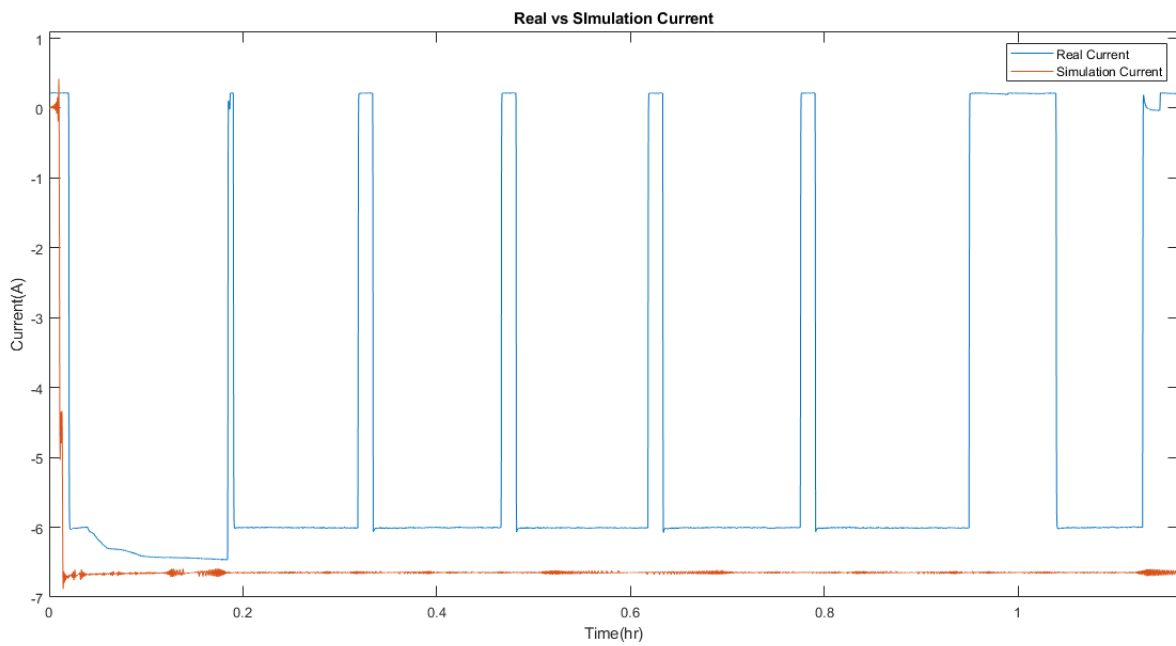
Berdasarkan hasil simulasi dan pengujian *prototype*, dapat dilakukan analisis mengenai kesesuaian antara keduanya, dari sisi hasil optimasi, grafik tegangan, grafik arus, dan grafik SOC. Untuk hasil optimasi, nilai yang difokuskan adalah perbandingan cost, lalu berlanjut ke perbandingan waktu, SOC Final, Loss, dan arus. Perbandingan hasil optimasi antara simulasi dan *prototype* dapat dilihat pada Tabel 4.6 sementara grafik perbandingan arus dapat dilihat pada Gambar 4.7, 4.8, 4.9, grafik perbandingan tegangan dapat dilihat pada Gambar 4.10, 4.11, 4.12, dan grafik perbandingan SOC dapat dilihat pada Gambar 4.13, 4.14, 4.15.

Tabel 4.7 Perbandingan Optimasi Simulasi dan *Prototype*

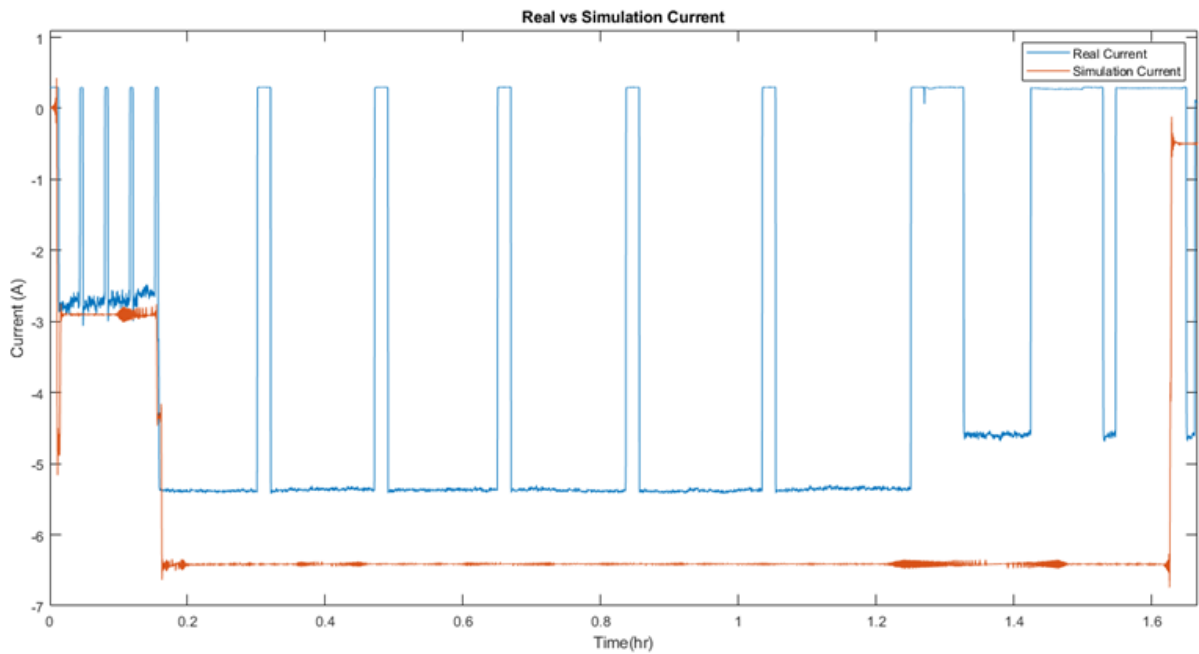
	I1 (A)	I2 (A)	I3 (A)	Waktu (s)	Loss (W)	SOC Final (%)	Cost
Simulasi Kondisi 1	2.32	6.685	0.38	4200	17576.28	0.84	0.3612
<i>Prototype</i> Kondisi 1	2.63	5.99	0.46	4200	13802.90	0.76	0.36
Error	0.31	-0.695	0.08	0	-3773.38	-0.08	-0.0012
Simulasi Kondisi 2	2.917	6.449	0.5	6000	21663.50	0.979	0.194
<i>Prototype</i> Kondisi 2	2.8	5.36	0.33	6000	22702.60	0.98	0.18
Error	-0.117	-1.089	-0.17	0	1039.1	0.001	-0.014
Simulasi Kondisi 3	3	6.926	0.5	7205	23308.35	1	0.36
<i>Prototype</i> Kondisi 3	2.4	6.6	0.35	8148	20122.80	1	0.38
Error	-0.6	-0.326	-0.15	943	-3185.55	0	0.02

Pada Tabel 4.6, tampak untuk beberapa kondisi hasil perhitungan optimasi *cost* sudah saling menyerupai dimana untuk masing-masing kondisi pengecasan, terdapat error secara berurutan yaitu 0.0012, 0.014, dan 0.02 atau dalam bentuk persentase terhadap optimasi pada simulasi yaitu 0.33%, 7.2%, dan 5.5%. Dari sisi arus, error paling besar terdapat pada kondisi pengecasan kedua. Pada kondisi ini, error yang besar terlihat dari penentuan *setpoint* arus kedua yang memiliki perbedaan lebih dari 1 ampere, yaitu 1.089 ampere. Perbedaan nilai hasil optimasi antara MATLAB dan *prototype* ini kemungkinan besar disebabkan oleh r_1 dan r_2 , yaitu nilai faktor random dalam persamaan algoritme PSO yang telah dijelaskan pada persamaan 2.1. Pada MATLAB, keacakan nilai yang diberikan cukup *reliable* sementara *microcontroller* AVR yang digunakan tidak dapat memberikan nilai acak yang benar-benar acak. Nilai acak pada *microcontroller* AVR akan berulang setiap 1 loop terlewat dan akan memberikan nilai acak yang sama apabila dilakukan reset pada *microcontroller*. Kejadian ini dapat diatasi dengan menggantungkan nilai acak pada bacaan analog dari kaki *microcontroller* yang tidak disambungkan, sehingga tercipta faktor acak baru setiap kali *microcontroller* mengalami perubahan loop atau *reset*, bergantung pada keacakan sinyal analog yang tertangkap. Di sisi lain, perbedaan nilai hasil optimasi juga dapat disebabkan oleh perbedaan perhitungan pada MATLAB dan *microcontroller*. Pada MATLAB, hubungan SOC dan tegangan terminal dituliskan dalam sebuah polinomial orde 11 sementara pada *microcontroller* nilai ini dipetakan menjadi suatu tabel data berukuran 101x2 yang mana berisikan data SOC dengan interval 1% serta tegangan terminal untuk SOC tersebut. Kedua perbedaan tersebut menyebabkan MATLAB memiliki perhitungan hubungan SOC dan tegangan yang lebih akurat dibandingkan dengan *microcontroller*. Sebagai perbandingan mudah, dimisalkan dalam perhitungan dicapai nilai SOC 0.125 yang berarti telah tercapai SOC 12.5%. Pada MATLAB, nilai tegangan akan dihitung dengan polinomial dengan input 0.125 yang berarti lebih besar daripada tegangan saat 12% dan lebih kecil dari tegangan saat 13%. Sementara itu, pada *microcontroller* akan didapatkan tegangan yang sama dengan tegangan pada SOC 12% dikarenakan tabel data SOC yang dituliskan dalam interval 1% SOC. Perbedaan ini tidak akan terlalu berdampak dan tampak untuk waktu pengecasan yang kecil dimana SOC akhir juga tidak berbeda jauh dengan SOC awal. Namun, seiring meningkatnya waktu pengecasan pada optimasi, perbedaan ini akan semakin tampak. Kemungkinan terakhir penyebab perbedaan hasil optimasi adalah tingkat kepresisian yang berbeda antara MATLAB dan *microcontroller* AVR yang digunakan, dimana MATLAB memiliki 16 digit kepresisian sementara AVR hanya memiliki 7 digit kepresisian.

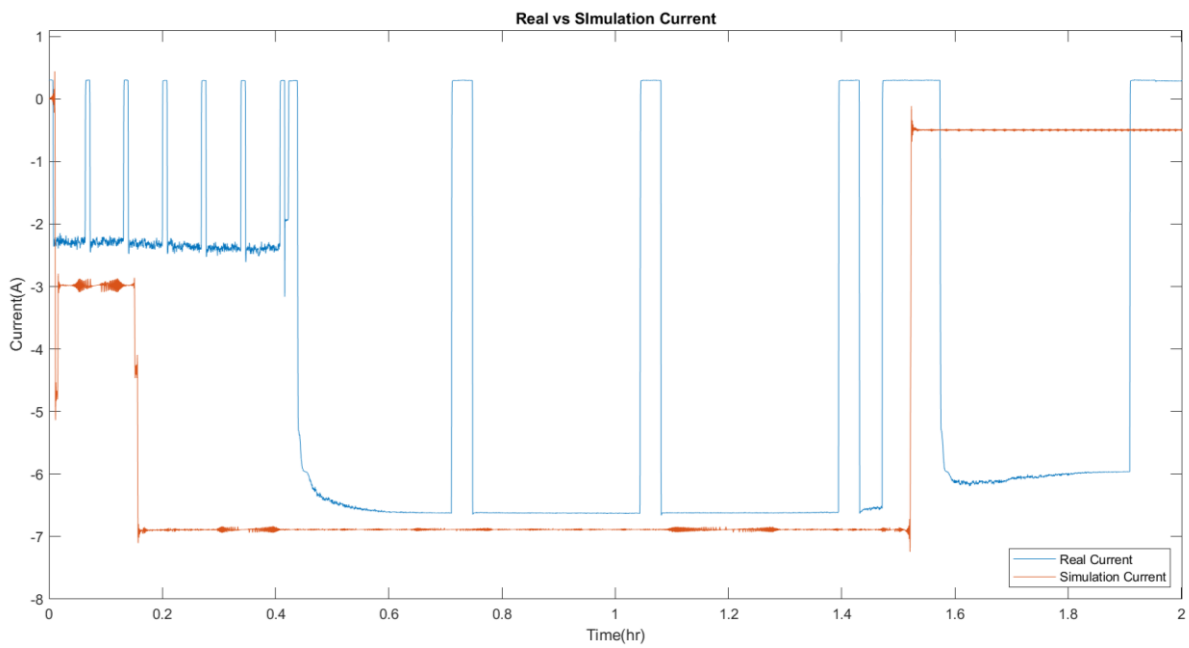
Perhitungan algoritme PSO melibatkan nilai-nilai desimal sehingga semakin tinggi kepresisian variabel yang disimpan maka hasil optimasi juga akan semakin baik.



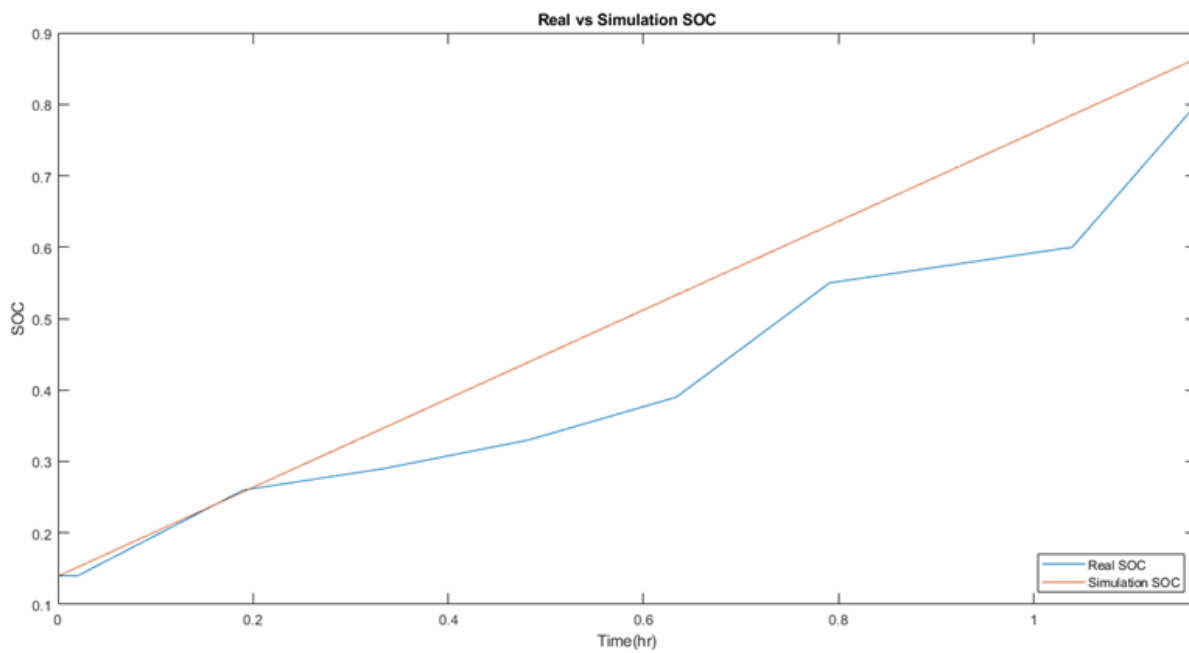
Gambar 4.5 Perbandingan Grafik Arus Simulasi dan *Prototype* Kondisi 1



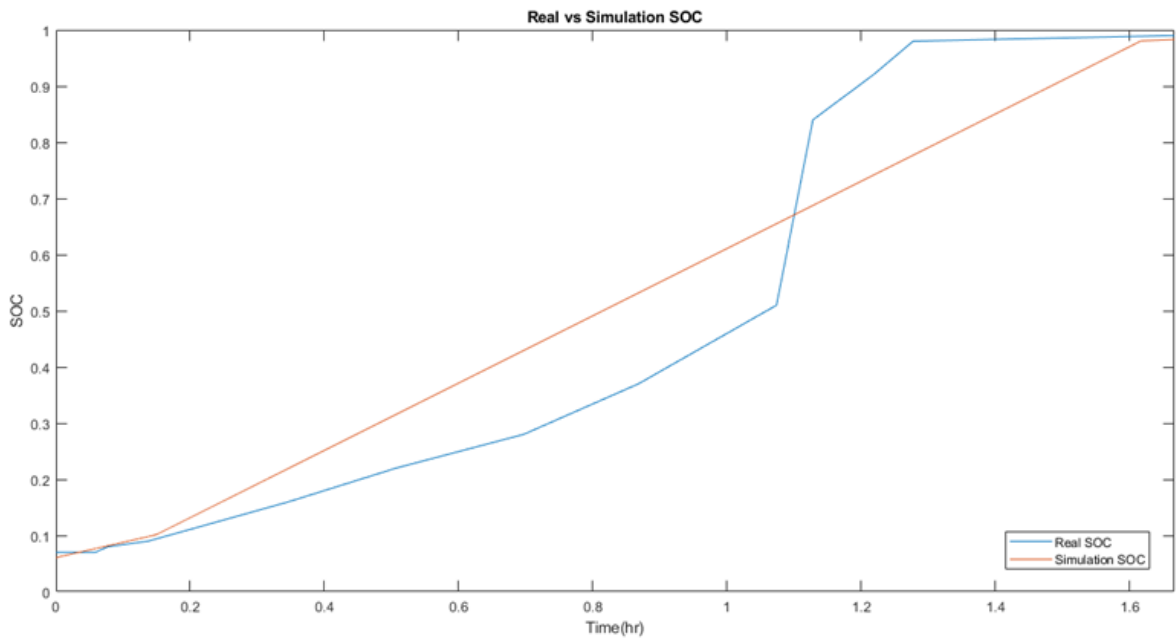
Gambar 4.6 Perbandingan Grafik Arus Simulasi dan *Prototype* Kondisi 2



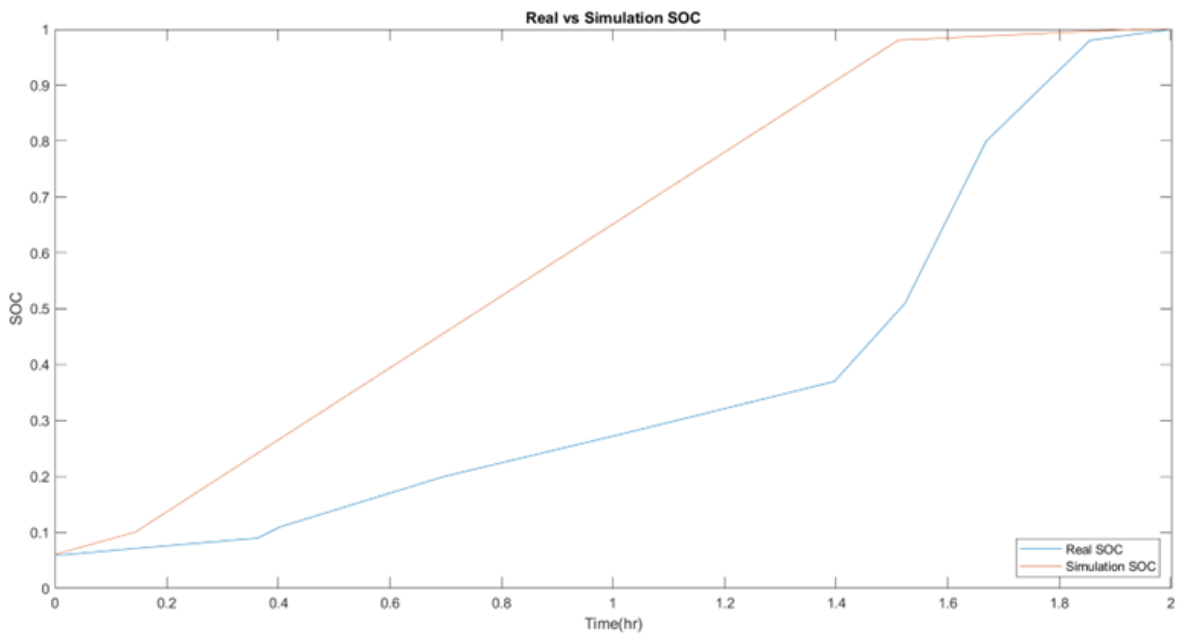
Gambar 4.7 Perbandingan Grafik Arus Simulasi dan *Prototype* Kondisi 3



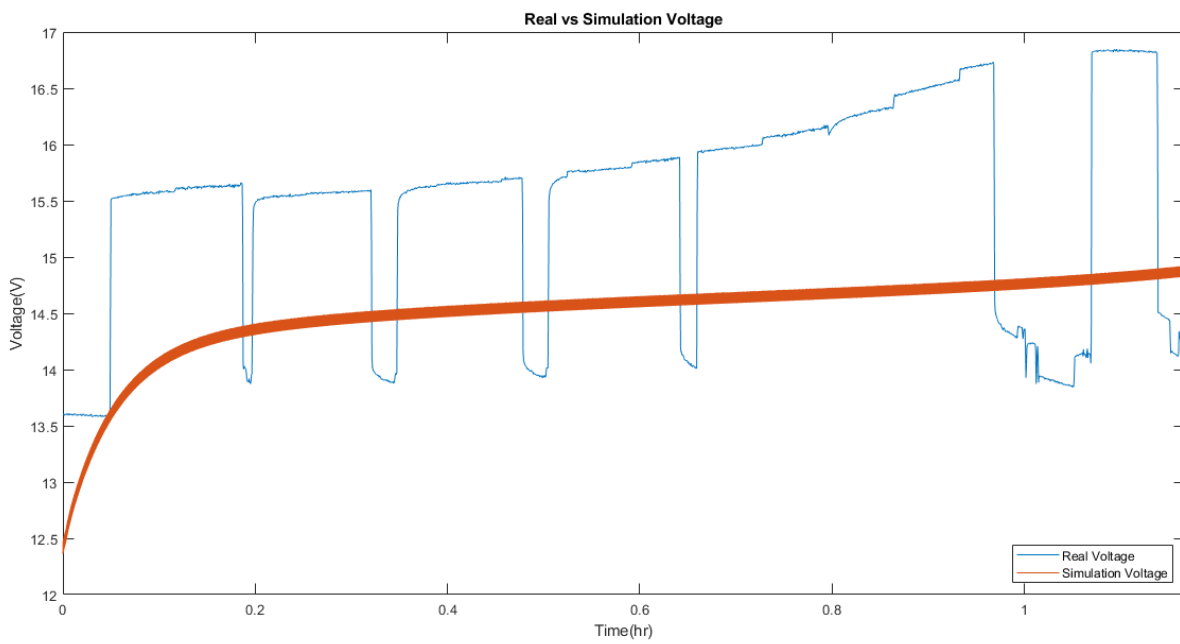
Gambar 4.8 Perbandingan Grafik SOC Simulasi dan *Prototype* Kondisi 1



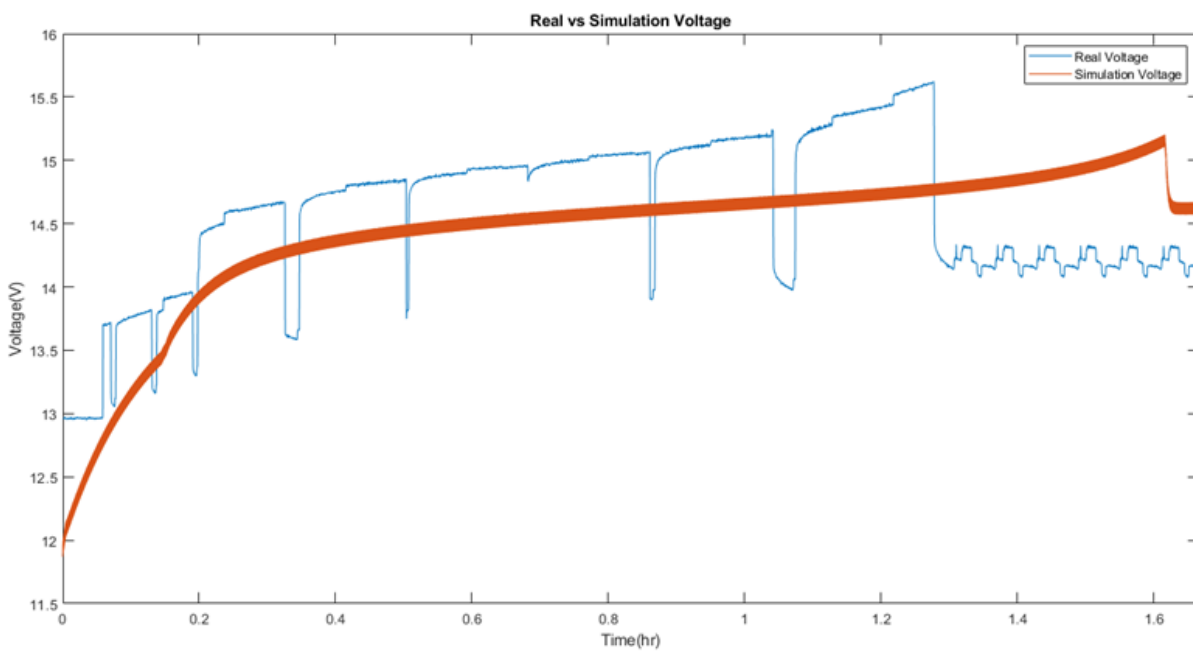
Gambar 4.9 Perbandingan Grafik SOC Simulasi dan *Prototype* Kondisi 2



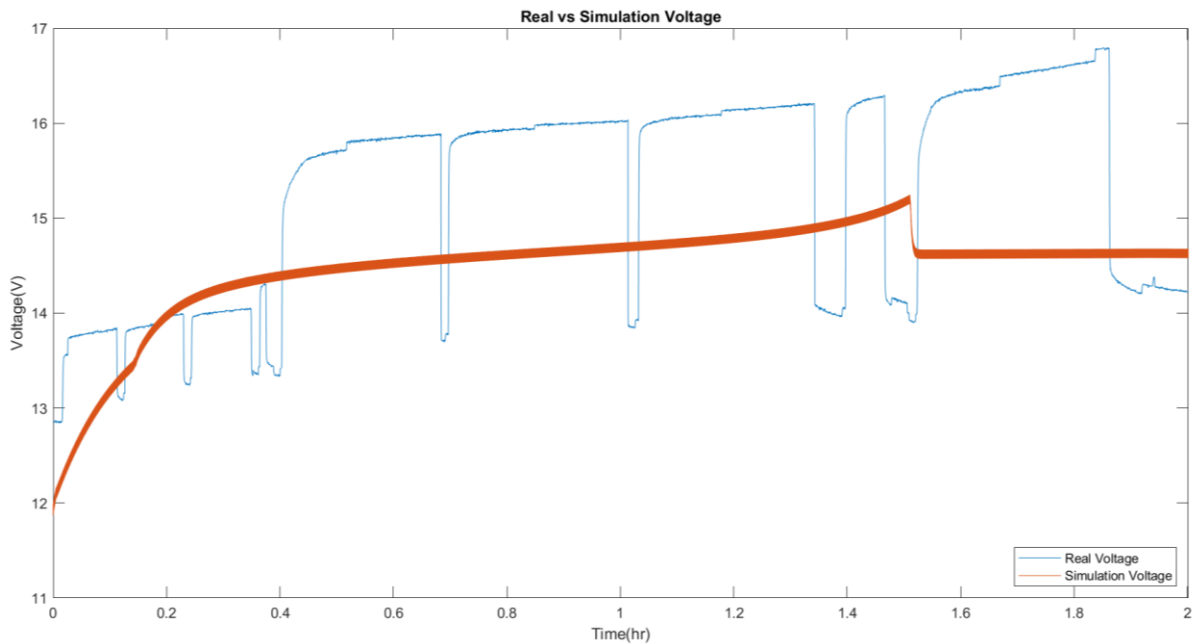
Gambar 4.10 Perbandingan Grafik SOC Simulasi dan *Prototype* Kondisi 3



Gambar 4.11 Perbandingan Grafik Tegangan Simulasi dan *Prototype* Kondisi 1



Gambar 4.12 Perbandingan Grafik Tegangan Simulasi dan *Prototype* Kondisi 2



Gambar 4.13 Perbandingan Grafik Tegangan Simulasi dan *Prototype* Kondisi 3

Melihat perbandingan grafik pada Gambar 4.7 terlihat bahwa arus pada simulasi dan *prototype* tidak ada yang mengalami perubahan *stage*. Hal ini sesuai dengan kondisi pada subbab 4.3 dan 4.4 dimana baik pada simulasi maupun *Prototype* sistem hanya mengalami *stage* pengecasan kedua dengan SOC awal 14% dan SOC akhir sekitar 80%. Dari grafik SOC pada Gambar 4.13, SOC pada *prototype* tidak linear seperti pada simulasi, namun keduanya masih saling menyerupai. Perbedaan ini dapat disebabkan oleh perbedaan cara pembacaan SOC sebagaimana telah dijelaskan sebelumnya. Sementara itu, pada Gambar 4.8, terlihat bahwa pada waktu 1 jam 18 menit arus pada *prototype* telah mengalami penurunan atau pergantian *stage* sementara pada simulasi perpindahan *stage* ini baru terjadi pada waktu ke 1 jam 36 menit. Hal ini berarti pemenuhan SOC pada *prototype* terjadi lebih cepat daripada simulasi untuk kondisi kedua ini. Hal tersebut didukung oleh grafik SOC pada Gambar 4.14 dimana SOC pada *prototype* selalu lebih rendah hingga pada waktu ke 1 jam 4 menit SOC naik drastis. Namun, kenaikan SOC ini diiringi dengan landainya perubahan SOC pada detik ke 1 jam 12 menit hingga akhir yang menyebabkan hasil akhir SOC pada *Prototype* maupun simulasi menjadi saling mendekati. Terakhir, berkebalikan dengan kondisi kedua dimana perubahan *stage* terjadi lebih dulu pada *prototype*, pada kondisi ketiga perubahan *stage* ini terjadi lebih dulu pada simulasi. Seperti yang dapat dilihat pada Gambar 4.9, pada waktu ke 10 menit arus dari simulasi telah berubah menjadi *stage 2* sementara pada *prototype* hal ini baru terjadi di menit ke 20. Begitu pula proses perubahan *stage* dari 2 ke 3 dimana pada simulasi terjadi di waktu ke 1 jam 30 menit sementara *prototype* terjadi di waktu ke 1 jam 36 menit. Waktu perpindahan *stage* ini terhitung tidak konstan, dimana selisih waktu dari perpindahan *stage 1* ke 2 lebih lama daripada *stage 2* ke 3. Berdasarkan grafik SOC pada Gambar 4.15, hal ini disebabkan oleh bacaan SOC yang lebih landai untuk *stage* pertama dibandingkan dengan *stage* kedua.

Analisa berikutnya adalah mengenai grafik tegangan baterai, dimana pada grafik pengecasan *prototype* terlihat bahwa tegangan pengecasan jauh lebih tinggi dibandingkan dengan pengecasan pada simulasi. Dapat dilihat bahwa pada Gambar 4.10, error antara

tegangan pengecasan simulasi dan *Prototype* mencapai nilai terbesar yaitu 12%. Pada bagian awal pengecasan, terlihat bahwa tegangan baterai sesungguhnya lebih tinggi daripada tegangan baterai simulasi. Namun, di akhir grafik, terlihat bahwa tegangan simulasi lebih tinggi daripada tegangan baterai sesungguhnya. Hal ini menunjukkan masih terdapat ketidaksesuaian antara baterai yang digunakan di *prototype* dengan baterai yang digunakan pada simulasi. Namun, dapat dilihat baik pada Gambar 4.10, 4.11, maupun 4.12 bahwa bacaan tegangan *prototype* lebih landai daripada simulasi. Dari perbedaan bentuk grafik ini, kemungkinan besar penyebab adanya perbedaan adalah pemilihan titik eksponensial baterai yang juga dapat menjadi alasan ketidaksesuaian yang telah dijelaskan di bagian-bagian sebelumnya. Kendati demikian, proses pengoptimalan dan pengecasan yang dilakukan pada tugas akhir ini telah bernilai saling mendekati. Penyebab lain ketidaksesuaian yang ada dikarenakan pada proses penentuan parameter, nilai parameter hanya diestimasi berdasarkan baterai pada simulasi yang nilainya telah disesuaikan dengan baterai yang Panasonic, mengikuti datasheet yang ada. Di sisi lain, baterai yang digunakan pada tugas akhir ini adalah jenis baterai yang memiliki karakteristik menyerupai jenis baterai Panasonic dan bukan menggunakan merk yang sama. Dari hal ini, terlihat bahwa parameter baterai sangat berpengaruh pada *charger controller* ini, baik dalam proses pengoptimalannya maupun proses pengisiannya.

BAB 5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada tugas akhir ini, dapat diperoleh beberapa kesimpulan antara lain:

1. Parameter PSO cukup berdampak pada hasil optimasi. Parameter PSO yang optimal pada tugas akhir ini adalah iterasi 10, populasi 10, c_1 1, c_2 1, w 0.8, dan w_d 0.99.
2. Algoritme PSO dapat diimplementasikan pada *prototype* sebagai media pengoptimalan arus pengecasan dengan input waktu dari pengguna dan mencapai tingkat keberhasilan 95% dengan error cost untuk 3 kondisi pengecasan masing-masing sebesar 0.33%, 7.22%, dan 5.55%.
3. Error arus antara optimasi simulasi dan *Prototype* disebabkan perbedaan kepresisian perhitungan.
4. Nilai parameter baterai sangat berpengaruh pada perhitungan pengecasan. Error tegangan pengecasan senilai 12% dapat disebabkan penentuan parameter yang kurang tepat.

5.2 Saran

Pada tugas akhir ini, terdapat beberapa saran untuk penelitian mendatang antara lain:

1. Pemilihan parameter algoritme PSO dapat kembali dikaji dengan menaikkan iterasi dan mengatasi *running time* dengan menggunakan *microprocessor* yang memiliki kecepatan lebih tinggi.
2. Parameter baterai dapat ditentukan berdasarkan *discharging pulse* pada baterai sesungguhnya untuk mendapatkan hasil yang lebih akurat.
3. Bacaan SOC dapat dilakukan dengan mengimplementasikan estimasi SOC dengan berbagai metode yang ada seperti Kalman filter pada *microcontroller*.
4. Perhitungan *loss* dapat diterapkan pada *prototype* dengan menambahkan sensor suhu serta mempertimbangkan suhu pengecasan sebagai objektivitas optimasi.
5. Dapat dilakukan implementasi serupa dengan melihat dampaknya pada efisiensi sumber tegangan.
6. Dapat dilakukan implementasi serupa dan menambahkan database parameter dari berbagai jenis baterai yang ada sehingga dapat menyesuaikan kebutuhan pengguna dari segi jenis kendaraan atau baterai.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

- Arya, S., & Verma, S. (n.d.). *Nickel-Metal Hydride (Ni-MH) Batteries*. 131–175.
- Baterijom, V. S. A. N. (2021). *A MECHANICAL PERFORMANCE STUDY AND SIMULATION OF A HYBRID ELECTRIC VEHICLE POWERED BY Ni-MH BATTERY VEHICLE POWERED BY Ni-MH BATTERY*. November.
- Bogno, B., Sawicki, J. P., Salame, T., Aillerie, M., Saint-Eve, F., Hamandjoda, O., & Tibi, B. (2017). Improvement of safety, longevity and performance of lead acid battery in off-grid PV systems. *International Journal of Hydrogen Energy*, 42(5), 3466–3478. <https://doi.org/10.1016/j.ijhydene.2016.12.011>
- Chen, G. J., Liu, Y. H., Wang, S. C., Luo, Y. F., & Yang, Z. Z. (2021). Searching for the optimal current pattern based on grey wolf optimizer and equivalent circuit model of Li-ion batteries. *Journal of Energy Storage*, 33(April 2020), 101933. <https://doi.org/10.1016/j.est.2020.101933>
- de Moura Oliveira, P. B., Hedengren, J. D., & Solteiro Pires, E. J. (2020). Swarm-based design of proportional integral and derivative controllers using a compromise cost function: An arduino temperature laboratory case study. *Algorithms*, 13(12). <https://doi.org/10.3390/a13120315>
- Khanum, F., Louback, E., Duperly, F., Jenkins, C., Kollmeyer, P. J., & Emadi, A. (2021). A Kalman filter based battery state of charge estimation MATLAB function. *2021 IEEE Transportation Electrification Conference and Expo, ITEC 2021*, 484–489. <https://doi.org/10.1109/ITEC51675.2021.9490163>
- Mars, N., Krouz, F., Louar, F., & Sbita, L. (2017). Comparison study of different dynamic battery model. *International Conference on Green Energy and Conversion Systems, GECS 2017*. <https://doi.org/10.1109/GECS.2017.8066241>
- Microchip Technology Inc. (1997). Using PWM to Generate Analog Output. *Technology*, 1, 1997–1999.
- Panasonic Corporation. (2014). *NI-MH Handbook*.
- Parthasarathy, K., & Vijayaraj, S. (2020). *An Overview of Battery Charging Methods , Charge Controllers , and Design of MPPT Controller based on Aduino Nano for Solar Renewable Storage Energy System*. 9(11), 430–439. <https://www.ijert.org/an-overview-of-battery-charging-methods-charge-controllers-and-design-of-mppt-controller-based-on-adruino-nano-for-solar-renewable-storage-energy-system>
- Shafiee, S., Fotuhi-Firuzabad, M., & Rastegar, M. (2012). Impacts of controlled and uncontrolled PHEV charging on distribution systems. *IET Conference Publications*, 2012(616 CP). <https://doi.org/10.1049/cp.2012.2160>
- U.S. Geological Survey. (2021). *Mineral Commodity Summaries 2021 : Sand and Gravel (Industrial)*.
- Windarko, N. A., & Choi, J. (2010). SOC estimation based on OCV for NiMH batteries using an improved Takacs model. *Journal of Power Electronics*, 10(2), 181–186. <https://doi.org/10.6113/JPE.2010.10.2.181>

Wu, T., Zhou, C., Yan, Z., Peng, H., & Wu, L. (2020). Application of PID optimization control strategy based on particle swarm optimization (PSO) for battery *charging* system. *International Journal of Low-Carbon Technologies*, 15(4), 528–535. <https://doi.org/10.1093/ijlct/ctaa020>

LAMPIRAN

1. Program Simulasi

a. Fungsi PSO

```
1 function [I1,I2,I3,costf] = sppsoin(it,pop,w,wd,c1,c2,SOC,Qm,Tin)
2
3     if nargin==8
4         Tin=32000;
5     end
6
7     % INITIALIZE SOME PARAMETERS
8     nv=3;
9     nvmax1 = 0.2*Qm;
10    nvmin1 = 0.3*Qm;
11    nvmax2 = 0.5*Qm;
12    nvmin2 = Qm;
13    nvmax3 = 0.05*Qm;
14    nvmin3 = 0.033*Qm;
15
16    vmax = (nvmax3-nvmin3);
17    vmin = -vmax;
18
19    % INITIALIZE POSITIONS AND VELOCITIES
20
21    % Generate random solution
22    position_now(1,:) = nvmin1+((nvmax1-nvmin1)*rand(1,pop));
23    position_now(2,:) = nvmin2+((nvmax2-nvmin2)*rand(1,pop));
24    position_now(3,:) = nvmin3+((nvmax3-nvmin3)*rand(1,pop));
25
26    % Generate initial velocity
27    velocity = zeros(nv,pop);
28
29    % Evaluate
30    cost=zeros(1,pop);
31    for p=1:pop
32        I1 = position_now(1,p);
33        I2 = position_now(2,p);
34        I3 = position_now(3,p);
35        cost(p) = fcostin(I1,I2,I3,Qm,SOC,Tin);
36    end
37
38    % Find local best cost among population
39    position_LB = position_now;
40    cost_LB = cost;
41
42    % Set best cost into global best
43    [cost_GB,xnv] = min(cost_LB);
44
```

```

45 % Set best global position
46 position_GB = position_LB(:,xnv);
47
48 % PSO Main Loop
49 for iter=1:it
50
51 % Update velocity
52 velocity = w*velocity + c1*(rand(nv,pop).*(position_LB- ...
53     position_now)) + c2*(rand(nv,pop).*(position_GB-position_now));
54
55 % Apply velocity limits
56 velocity=min(velocity, vmax);
57 velocity=max(velocity, vmin);
58
59 % Update position
60 position_now = position_now + velocity;
61
62 % Apply solution limits
63 position_now(1,:) = min(position_now(1,:), nvmax1);
64 position_now(1,:) = max(position_now(1,:), nvmin1);
65 position_now(2,:) = min(position_now(2,:), nvmax2);
66 position_now(2,:) = max(position_now(2,:), nvmin2);
67 position_now(3,:) = min(position_now(3,:), nvmax3);
68 position_now(3,:) = max(position_now(3,:), nvmin3);
69
70 % Evaluate
71 for p=1:pop
72     I1 = position_now(1,p);
73     I2 = position_now(2,p);
74     I3 = position_now(3,p);
75     cost(p) = fcostin(I1,I2,I3,Qm,SOC,Tin);
76 end
77
78 % Update personal best
79 for p=1:pop
80     if cost(p) < cost_LB(p)
81         cost_LB(p) = cost(p);
82         position_LB(:,p) = position_now(:,p);
83     end
84 end
85
86 [cost_GBn,xnv] = min(cost_LB);
87
88 if cost_GBn < cost_GB
89     cost_GB = cost_GBn;
90     position_GB=position_LB(:,xnv);
91 end
92
93 % Update Coefficient
94 w = w * wd;

```



```

95     end
96     I1 = position_GB(1);
97     I2 = position_GB(2);
98     I3 = position_GB(3);
99     costf=cost_GB;
100 end

```

b. Fungsi Cost

```

1  function [J,T,L,Yf]=fcostin(I1,I2,I3,Qm,SOC_Init,Tin)
2
3      load('vbvoc_exp12915.mat');
4
5      % SOC-OCV RELATION
6      SOCOCV = polyfit(0:0.01:1,Voc,10);
7
8      % PARAMETER USED
9      Rp = 0.009;
10     Cp = 2.5508e+04;
11     Ro = 0.0846;
12     E0=13.0232;
13     Ts=1;
14     simTime=32000;
15     X      = [SOC_Init; 0;];
16     DeltaT = Ts; % sample time in seconds
17     Qn_rated = 3600*Qm; % Ah to Amp-seconds
18     Tau_1   = Rp*Cp;
19     a1      = exp(-DeltaT/Tau_1);
20     b1      = Rp * (1 - exp(-DeltaT/Tau_1));
21     eta     = 0.9;
22     Umax    = [-0.3*Qm -Qm -0.05*Qm];
23     Umin    = [-0.2*Qm -0.5*Qm -0.033*Qm];
24     U       = [-I1 -I2 -I3];
25     Lmax=0;
26     Lmin=0;
27     Tmax=0;
28     Tmin=0;
29     Ymax=14.5240;
30     Ymin=10.3408;
31     L=0;
32     T=0;
33     smp     = (simTime/Ts)+1;
34
35     % MIN T MAX L
36     Xtmin=X;
37     for loop=1:smp
38         if (0<=Xtmin(1))&&(Xtmin(1)<=0.1)
39             s=1;
40         elseif (0.1<Xtmin(1))&&(Xtmin(1)<=0.98)
41             s=2;
42         elseif (0.98<Xtmin(1))&&(Xtmin(1)<=1)

```

```

43         s=3;
44     elseif Xtmin(1)>1
45         break
46     end
47
48     A = [1 0; 0 a1;];
49     B = [-(eta * DeltaT/Qn_rated); b1;];
50     C = [polyval(SOCOVCV,Xtmin(1)) -1];
51     D = -Ro;
52
53     Ytmin=C(1)+C(2)*Xtmin(2)+D*Umax(s);
54     Xtmin=A*Xtmin+B*Umax(s);
55
56     Tmin=Tmin+Ts;
57     Lmax=Lmax+(Ts*Umax(s)*Umax(s)*(Ro+Rp));
58 end
59
60 % MIN L MAX T
61 Xlmin=X;
62 for loop=1:smp
63     if (0<=Xlmin(1))&&(Xlmin(1)<=0.1)
64         s=1;
65     elseif (0.1<Xlmin(1))&&(Xlmin(1)<=0.98)
66         s=2;
67     elseif (0.98<Xlmin(1))&&(Xlmin(1)<=1)
68         s=3;
69     elseif Xlmin(1)>1
70         break
71     end
72
73     A = [1 0; 0 a1;];
74     B = [-(eta * DeltaT/Qn_rated); b1;];
75     C = [polyval(SOCOVCV,Xlmin(1)) -1];
76     D = -Ro;
77
78     Ylmin=C(1)+C(2)*Xlmin(2)+D*Umin(s);
79     Xlmin=A*Xlmin+B*Umin(s);
80
81     Tmax=Tmax+Ts;
82     Lmin=Lmin+(Ts*Umin(s)*Umin(s)*(Ro+Rp));
83 end
84
85 % INPUT SETTING
86 finsimt=Tin;
87 if Tin>=Tmax
88     ST=1;
89 elseif Tin<Tmin
90     ST=0;
91 elseif (Tin>=Tmin)&&(Tin<Tmax)
92     ST=0;

```

```

93     end
94
95     smps      = (finsimt/Ts)+1;
96
97     % OPTIMIZATION
98     Y = zeros(1,smps);
99     for loop=1:smps
100        if (0<=X(1))&&(X(1)<=0.1)
101            s=1;
102        elseif (0.1<X(1))&&(X(1)<=0.98)
103            s=2;
104        elseif (0.98<X(1))&&(X(1)<=1)
105            s=3;
106        elseif X(1)>1
107            break
108        end
109
110        A = [1 0; 0 a1;];
111        B = [-(eta * DeltaT/Qn_rated); b1;];
112        C = [polyval(SOCOVC,X(1)) -1];
113        D = -Ro;
114
115        Y(loop)=C(1)+C(2)*X(2)+D*U(s);
116        X=A*X+B*U(s);
117
118        T=T+Ts;
119        L=L+(Ts*U(s)*U(s)*(Ro+Rp));
120    end
121
122    % COST
123    if ST==1
124        alpha=0.6;
125        beta=0.4;
126    else
127        alpha=0;
128        beta=0.4;
129    end
130
131    Y=Y(1:loop-1);
132    Yf=Y(length(Y));
133    J1=(T-Tmin)/(Tmax-Tmin);
134    J2=(L-Lmin)/(Lmax-Lmin);
135    J3=(Yf-Ymax)/(Ymin-Ymax);
136    J=sqrt((alpha*J1*J1)+(beta*J2*J2)+((1-(alpha+beta))*J3*J3));
137 end

```

c. Pengambilan Data Simulasi

1. clear;
2. clc;

```

3. % Load data parameter PSO
4. load('1_2_11.mat','iter','pop','c1','c2','w','wd');
5.
6. SOCIn = 0;
7. Cap = 10;
8. Tin=32000;
9.
10. tic
11. [I1,I2,I3]=sppsoin(iter,pop,w,wd,c1,c2,SOCIn,Cap,Tin);
12. runtime=toc;
13. [J,t,Loss,Yf]=fcostin(I1,I2,I3,Cap,SOCIn,Tin);
14.
15. I1=-I1;
16. I2=-I2;
17. I3=-I3;
18.
19. SOC_Batt=SOCIn;
20. options = simset('SrcWorkspace','current');
21. out=sim('TA05_Charging_Simulation.slx',[0 t],options);

```

2. Program *Prototype* dengan Mikrokontroler AVR

```

1  /* INCLUDE LIBRARY */
2  #include <LiquidCrystal.h>
3  #include <PID_v1.h>
4
5  /* LCD ASSIGNMENT */
6  LiquidCrystal lcd(23, 22, 21, 20, 19, 18);
7
8
9  /* SETTING VARIABLES */
10 float b_cap;
11 int t_max;
12 byte stateButton, lastState, wrt;
13
14 /* PSO PARAMETERS */
15 byte nv=3;
16 float min1;
17 float max1;
18 float min2;
19 float max2;
20 float min3;
21 float max3;
22 byte it=10;
23 byte pop=10;
24 float w=0.8;
25 float wd=0.99;
26 float c1=1;
27 float c2=1;
28 float vmin,vmax;
29 float randc1,randc2;
30
31 float position_now[3][10] = {
32     {0,0,0,0,0,0,0,0,0,0},
33     {0,0,0,0,0,0,0,0,0,0},
34     {0,0,0,0,0,0,0,0,0,0},
35 };

```

```

36 float position_LB[3][10] = {
37     {0,0,0,0,0,0,0,0,0,0},
38     {0,0,0,0,0,0,0,0,0,0},
39     {0,0,0,0,0,0,0,0,0,0},
40 };
41 float velocity[3][10] = {
42     {0,0,0,0,0,0,0,0,0,0},
43     {0,0,0,0,0,0,0,0,0,0},
44     {0,0,0,0,0,0,0,0,0,0},
45 };
46
47 float position_GB[3];
48 float cost[10];
49 float cost_LB[10];
50 float cost_GB;
51 float cost_GBn;
52 float opt_soc;
53 float opt_loss;
54 float opt_time;
55
56 byte p,q, iter;
57 byte xnv;
58
59 /* COST FUNCTION VARIABLES */
60 float X1,X2,Xtmin1,Xtmin2,Xlmin1,Xlmin2,Ytmin,Ylmin;
61 float a1,b1,eta,finsimt;
62 float Rp=0.009;
63 float Cp=2.5508e+04;
64 float Ro=0.0846;
65 float Ts=1;
66 float simTime=32000;
67 float Lmax,Lmin,Tmax,Tmin,Ymax,Ymin,L,T,Y;
68 float alpha,beta,J,J1,J2,J3;
69 int loops,s,ST;
70
71 float Umin[3] = {0,0,0};
72 float Umax[3] = {0,0,0};
73 float U[3] = {0,0,0};
74
75 const static PROGMEM float
SOC_Conv[202]={0,10.4506,0.01,10.8875,0.02,11.1978,0.03,11.4295,0.04,11.6092,0.05,11.7526,0.06,11.8696,0.07
,11.967,0.08,12.0493,0.09,12.1197,0.1,12.1807,0.11,12.234,0.12,12.281,0.13,12.3228,0.14,12.3601,0.15,12.393
7,0.16,12.4241,0.17,12.4517,0.18,12.4769,0.19,12.5,0.2,12.5212,0.21,12.5408,0.22,12.5589,0.23,12.5757,0.24,
12.5914,0.25,12.606,0.26,12.6197,0.27,12.6325,0.28,12.6445,0.29,12.6559,0.3,12.6665,0.31,12.6766,0.32,12.68
62,0.33,12.6952,0.34,12.7038,0.35,12.7119,0.36,12.7196,0.37,12.727,0.38,12.734,0.39,12.7407,0.4,12.7472,0.6
1,12.8368,0.62,12.8397,0.63,12.8425,0.64,12.8452,0.65,12.8478,0.66,12.8503,0.67,12.8528,0.68,12.8552,0.69,1
2.8576,0.7,12.8599,0.71,12.8621,0.72,12.8643,0.73,12.8665,0.74,12.8686,0.75,12.8707,0.76,12.8728,0.77,12.87
49,0.78,12.877,0.79,12.8792,0.8,12.8815,0.81,12.884,0.82,12.8868,0.83,12.8899,0.84,12.8936,0.85,12.898,0.86
,12.9036,0.87,12.9108,0.88,12.9203,0.89,12.933,0.9,12.9502,0.91,12.9736,0.92,13.006,0.93,13.0509,0.94,13.11
32,0.95,13.2002,0.96,13.3218,0.97,13.4918,0.98,13.73,0.99,14.0637,1,14.5315};
76 const static PROGMEM float
Vb_Conv[202]={0,10.1524,0.01,10.9538,0.02,11.5152,0.03,11.9283,0.04,12.2391,0.05,12.4772,0.06,12.6629,0.07,
12.81,0.08,12.9283,0.09,13.025,0.1,13.1051,0.11,13.1724,0.12,13.2297,0.13,13.2791,0.14,13.3221,0.15,13.3599
,0.16,13.3935,0.17,13.4236,0.18,13.4508,0.19,13.4754,0.2,13.4979,0.21,13.5186,0.22,13.5376,0.23,13.5553,0.2
4,13.5717,0.25,13.587,0.26,13.6012,0.27,13.6146,0.28,13.6272,0.29,13.6391,0.3,13.6503,0.31,13.6609,0.32,13.
671,0.33,13.6805,0.34,13.6896,0.35,13.6982,0.36,13.7065,0.37,13.7144,0.38,13.7219,0.39,13.7291,0.4,13.7361,
0.61,13.9078,0.62,13.9178,0.63,13.9278,0.64,13.9378,0.65,13.9478,0.66,13.9578,0.67,13.9678,0.68,13.9778,0.6
9,13.9878,0.7,13.9978,0.71,14.0078,0.72,14.0178,0.73,14.0278,0.74,14.0378,0.75,14.0478,0.76,14.0578,0.77,14
.0678,0.78,14.0778,0.79,14.0878,0.8,14.0978,0.81,14.1078,0.82,14.1178,0.83,14.1278,0.84,14.1378,0.85,14.147
8,0.86,14.1578,0.87,14.1678,0.88,14.1778,0.89,14.1878,0.9,14.1978,0.91,14.2078,0.92,14.2178,0.93,14.2278,0.
94,14.2378,0.95,14.2478,0.96,14.2578,0.97,14.2678,0.98,14.2778,0.99,14.2878,1,14.2978};
77
78 /* BUTTON ASSIGNMENT */
79 #define CLK 17
80 #define SW 16

```

```

81 #define DT 15
82
83 int CLKstate;
84 int DTstate;
85 int SWstate;
86
87 /* PID ASSIGNMENT */
88 double Setpoint1, Setpoint2, Setpoint3, Input, Output;
89 PID b1PID(&Input, &Output, &Setpoint1, 8.2582, 61.1057, 0, DIRECT);
90 PID absPID(&Input, &Output, &Setpoint2, 9.0094, 67.4419, 0, DIRECT);
91 PID trPID(&Input, &Output, &Setpoint3, 5.4203, 48.9966, 0, DIRECT);
92
93 /* DEFINE DAC PIN */
94 #define DAC0 14
95 #define DAC1 13
96 #define DAC2 12
97 #define DAC3 11
98 #define DAC4 10
99 #define DAC5 7
100 #define DAC6 6
101 #define DAC7 5
102 #define DAC8 0
103 #define DAC9 1
104
105 /* DEFINE INPUT OUTPUT PIN */
106 #define IN0 A5
107 #define IN1 A6
108 #define OUT0 A7
109 #define OUT1 2
110 #define OUT2 3
111
112 /* CHARGING SYSTEM VARIABLES */
113 int input;
114 float Io;
115 float V0;
116 float Ii;
117 float Vb;
118 float I;
119 float SOC;
120 int UPDATE;
121 long READV;
122 byte state;
123
124 void setup() {
125     //Serial.begin(9600);
126     lcd.begin(16,2);
127
128     pinMode(DAC0, OUTPUT);
129     pinMode(DAC1, OUTPUT);
130     pinMode(DAC2, OUTPUT);
131     pinMode(DAC3, OUTPUT);
132     pinMode(DAC4, OUTPUT);
133     pinMode(DAC5, OUTPUT);
134     pinMode(DAC6, OUTPUT);
135     pinMode(DAC7, OUTPUT);
136     pinMode(DAC8, OUTPUT);
137     pinMode(DAC9, OUTPUT);
138

```

```

139 pinMode(IN0, INPUT);           //Current sensor
140 pinMode(IN1, INPUT);         //Read Battery Voltage
141 pinMode(OUT0, OUTPUT);       //Output to transistor
142 pinMode(OUT1, OUTPUT);       //Output relay batt
143 pinMode(OUT2, OUTPUT);       //Output relay power
144 pinMode(CLK,INPUT_PULLUP);   //Tombol up
145 pinMode(DT,INPUT_PULLUP);    //Tombol down
146 pinMode(SW, INPUT_PULLUP);   //Tombol center
147
148 /*INITIAL VALUE*/
149 input=0;
150 V0=0;
151 I=0;
152 Io=0;
153 Vb=0;
154 UPDATE=0;
155 READV=1;
156 state=0;
157 lastState=0;
158 stateButton=0;
159
160 blPID.SetMode(AUTOMATIC);
161 absPID.SetMode(AUTOMATIC);
162 trPID.SetMode(AUTOMATIC);
163
164 b_cap=10;
165 t_max=32000;
166
167 analogReference(INTERNAL);
168
169 /*INITIAL RELAY CONDITION*/
170 digitalWrite(OUT1, LOW);
171 digitalWrite(OUT2, LOW);
172
173 digitalWrite(OUT0, HIGH);
174 delay(8000);
175 digitalWrite(OUT0, LOW);
176
177 delay(1000);
178
179 V0=analogRead(IN1);
180 Vb=(V0/1023)*(2.235)*(15.95/0.95)*1.12;
181
182 /*If battery voltage is abnormal, check battery connection */
183 if(Vb<=1) {state=1;}
184
185 /*If not, check whether user want optimization or not */
186 else
187 {
188     state=0;
189

```

```

190     /*PID Parameter set*/
191     Setpoint1=2.56;
192     Setpoint2=5.61;
193     Setpoint3=0.37;
194
195     state=0;
196
197     /* Check center button pressed or not */
198     if (digitalRead(SW)==LOW) {state=2;}
199 }
200
201     SOC=SOCRead(0,Vb);
202
203     randomSeed(analogRead(A0));
204 }
205
206 void loop() {
207     if (state==0)
208     {
209         digitalWrite(OUT1,HIGH);
210
211         if(UPDATE==2000){UPDATE=0;}
212
213         SOC=SOCRead(0,Vb);
214
215         Input=I;
216
217         if(SOC<=10)                {blPID.Compute(); if(READV==180000){READV=0;}}
218         else if(SOC>10&&SOC<=98)    {absPID.Compute();
if(READV==750000){READV=0;}}
219         else if(SOC>98)            {trPID.Compute(); if(READV==100000){READV=0;}}
220
221         if(Vb<=14.5){
222             DAC(Output);
223         }
224         else if(Vb>14.5)
225         {
226             DAC(0);
227         }
228
229         Io=analogRead(IN0);
230         I=(Io/1023)*2.235*2*1.77;
231
232         /*Check batt voltage every x time*/
233         if (READV==0)
234         {
235             DAC(0);
236
237             delay(500);
238
239             digitalWrite(OUT1, LOW);

```



```

240
241     digitalWrite(OUT0,HIGH);
242     if(SOC<=10)           {delay(8000);}
243     else if(SOC>10&&SOC<=98) {delay(35000);}
244     else if(SOC>98)       {delay(5000);}
245     delay(8000);
246     digitalWrite(OUT0,LOW);
247
248     if(SOC<=10)           {delay(1000);}
249     else if(SOC>10&&SOC<=98) {delay(2000);}
250     else if(SOC>98)       {delay(1000);}
251
252     V0=analogRead(IN1);
253     Vb=(V0/1023)*(2.235)*(15.95/0.95)*1.12;
254     digitalWrite(OUT1,HIGH);
255 }
256
257 /* Show value on LCD every x time */
258 if(UPDATE==0)
259 {
260     lcd.setCursor(0,0);
261     lcd.print("I:   ");
262     lcd.setCursor(3,0);
263     lcd.print(I);
264     lcd.setCursor(8,0);
265     lcd.print("V:   ");
266     lcd.setCursor(11,0);
267     lcd.print(Vb);
268     lcd.setCursor(0,1);
269     lcd.print("%:   ");
270     lcd.setCursor(3,1);
271     lcd.print(SOC);
272     lcd.setCursor(8,1);
273     lcd.print(Output);
274
275 }
276
277 UPDATE++;
278 READV++;
279 }
280
281 /* ABNORMAL BATTERY VOLTAGE */
282 else if(state==1)
283 {
284     lcd.setCursor(0,0);
285     lcd.print(" CHECK BATTERY");
286     lcd.setCursor(0,1);
287     lcd.print(" CONNECTION ");
288 }
289
290 /* ASK COST ORIENTATION */

```

```

291 else if(state==2){
292
293     lcd.setCursor(0,0);
294     lcd.print("CAP : DEFAULT");
295     lcd.setCursor(0,1);
296     lcd.print("MAXt: DEFAULT");
297
298     while (stateButton==0){
299         if (digitalRead(SW) == LOW) {
300             lastState=1;
301         }
302         if (digitalRead(SW) == HIGH && lastState == 1){
303             stateButton=1;
304             lastState=0;
305             break;
306         }
307     }
308
309     /* SET CAPACITY */
310     while(stateButton==1){
311         if(digitalRead(DT)==LOW)      {b_cap=b_cap+0.1; delay(100); wrt=1;}
312         else if(digitalRead(CLK)==LOW){b_cap=b_cap-0.1; delay(100); wrt=1;}
313
314         if(b_cap>10)      {b_cap=10;}
315         else if(b_cap<1.2) {b_cap=1.2;}
316
317         if (wrt==1){
318             lcd.setCursor(6,0);
319             lcd.print("      ");
320             lcd.setCursor(6,0);
321             lcd.print(b_cap);
322             wrt=0;
323         }
324
325         if (digitalRead(SW) == LOW) {
326             lastState=1;
327         }
328         if (digitalRead(SW) == HIGH && lastState == 1){
329             stateButton=2;
330             lastState=0;
331             break;
332         }
333     }
334
335     delay(100);
336
337     /* SET MAX TIME */
338     while(stateButton==2){
339         if(digitalRead(DT)==LOW)      {t_max=t_max+60; delay(100); wrt=1;}
340         else if(digitalRead(CLK)==LOW){t_max=t_max-60; delay(100); wrt=1;}
341

```

```

342     if(t_max>32000)    {t_max=1;}
343     else if(t_max<1)  {t_max=32000;}
344
345     if (wrt==1){
346         lcd.setCursor(6,1);
347         lcd.print("      ");
348         lcd.setCursor(6,1);
349         lcd.print(t_max);
350         wrt=0;
351     }
352
353     if (digitalRead(SW) == LOW) {
354         lastState=1;
355     }
356     if (digitalRead(SW) == HIGH && lastState == 1){
357         stateButton=0;
358         lastState=0;
359         break;
360     }
361 }
362
363 /* PARAMETER UNTUK KALKULASI MIN MAX T-L */
364
365 a1      = exp(-Ts/(Rp*Cp));
366 b1      = Rp * (1 - exp(-Ts/(Rp*Cp)));
367 eta     = 0.9;
368
369 Umax[0] = -0.3*b_cap;
370 Umax[1] = -b_cap;
371 Umax[2] = -0.05*b_cap;
372 Umin[0] = -0.2*b_cap;
373 Umin[1] = -0.5*b_cap;
374 Umin[2] = -0.033*b_cap;
375
376 /* MAX TIME AND MIN LOSS CALCULATION */
377 Xtmin1=(float)SOC/100;
378 Xtmin2=0;
379
380 for (loops=0; loops<simTime; loops++){
381
382     if ((0<=Xtmin1)&&(Xtmin1<=0.1))    {s=0;}
383     else if ((0.1<Xtmin1)&&(Xtmin1<=0.98)) {s=1;}
384     else if ((0.98<Xtmin1)&&(Xtmin1<=1))  {s=2;}
385     else if (Xtmin1>1)                    {break;}
386
387     Xtmin1=Xtmin1-((eta * Ts/(3600*b_cap))*Umax[s]);
388     Xtmin2=a1*Xtmin2+b1*Umax[s];
389
390     Tmin=Tmin+Ts;
391     Lmax=Lmax+(Ts*Umax[s]*Umax[s]*(Ro+Rp));
392

```

```

393     }
394
395     /* MAX TIME AND MIN LOSS CALCULATION */
396     Xlmin1=(float)SOC/100;
397     Xlmin2=0;
398
399     for (loops=0; loops<simTime; loops++){
400
401         if ((0<=Xlmin1)&&(Xlmin1<=0.1))           {s=0;}
402         else if ((0.1<Xlmin1)&&(Xlmin1<=0.98))    {s=1;}
403         else if ((0.98<Xlmin1)&&(Xlmin1<=1))      {s=2;}
404         else if (Xlmin1>1)                        {break;}
405
406         Xlmin1=Xlmin1-((eta * Ts/(3600*b_cap))*Umin[s]);
407         Xlmin2=a1*Xlmin2+b1*Umin[s];
408
409         Tmax=Tmax+Ts;
410         Lmin=Lmin+(Ts*Umin[s]*Umin[s]*(Ro+Rp));
411
412     }
413
414     /* RUNNING PSO */
415
416     lcd.clear();
417     lcd.setCursor(0,0);
418     lcd.print("OPTIMIZING ");
419     lcd.print(SOC);
420
421
422     /* INITIALIZE SOME PARAMETERS */
423
424     min1=-Umin[0]*100;
425     max1=-Umax[0]*100;
426     min2=-Umin[1]*100;
427     max2=-Umax[1]*100;
428     min3=-Umin[2]*100;
429     max3=-Umax[2]*100;
430     vmax = ((max3/100)-(min3/100));
431     vmin = -vmax;
432
433     // INITIALIZE POSITIONS AND VELOCITIES
434     for (p=0;p<pop;p++){
435     {
436         // Generate random solution
437         position_now[0][p] = ((float)random(min1,max1))/100;
438         position_now[1][p] = ((float)random(min2,max2))/100;
439         position_now[2][p] = ((float)random(min3,max3))/100;
440
441         //Generate initial velocity
442         velocity[0][p] = 0;
443         velocity[1][p] = 0;

```

```

444     velocity[2][p] = 0;
445
446
447     // Evaluate
448     Costfunc(p,b_cap,((float)SOC/100),t_max);
449     cost[p]=J;
450 }
451
452 xnv=0;
453 for (p=0;p<pop;p++)
454 {
455     for (q=0;q<nv;q++)
456     {
457         position_LB[q][p] = position_now [q][p];
458         cost_LB[p] = cost[p];
459
460         if (p>0 && cost_LB[p]<cost_LB[p-1])
461         {
462             cost_GB = cost_LB[p];
463             xnv = p;
464         }
465         position_GB[q]=position_LB[q][xnv];
466     }
467 }
468
469 lcd.clear();
470
471 // PSO Main Loop
472 for (iter=1;iter<=it;iter++)
473 {
474     //Serial.println(iter);
475     //lcd.setCursor(0,0);
476     //lcd.print(iter);
477
478     for (p=0;p<pop;p++)
479     {
480         for (q=0;q<nv;q++)
481         {
482             // Update velocity
483             randc1=((float)random(100))/100;
484             randc2=((float)random(100))/100;
485             velocity[q][p] = w*velocity[q][p] + c1*(randc1*(position_LB[q][p]-
position_now[q][p]))+ c2*(randc2*(position_GB[q]-position_now[q][p]));
486
487             // Apply velocity limits
488             velocity[q][p]=min(velocity[q][p], vmax);
489             velocity[q][p]=max(velocity[q][p], vmin);
490
491             // Update
492             position_now[q][p] = position_now[q][p] + velocity[q][p];
493         }

```

```

494 // Apply solution limits
495 position_now[0][p] = min(position_now[0][p], max1/100);
496 position_now[0][p] = max(position_now[0][p], min1/100);
497 position_now[1][p] = min(position_now[1][p], max2/100);
498 position_now[1][p] = max(position_now[1][p], min2/100);
499 position_now[2][p] = min(position_now[2][p], max3/100);
500 position_now[2][p] = max(position_now[2][p], min3/100);
501
502 // Evaluate
503 Costfunc(p,b_cap,((float)SOC/100),t_max);
504 cost[p]=J;
505 }
506 xnv=0;
507 for (p=0;p<pop;p++)
508 {
509     for (q=0;q<nv;q++)
510     {
511         if (cost[p]<cost_LB[p])
512         {
513             cost_LB[p] = cost[p];
514             position_LB[q][p] = position_now [q][p];
515         }
516
517         if (p>0 && cost_LB[p]<cost_LB[p-1])
518         {
519             cost_GBn = cost_LB[p];
520             if (cost_GBn < cost_GB)
521             {
522                 xnv = p;
523                 cost_GB=cost_GBn;
524                 opt_time=T;
525                 opt_loss=L;
526                 opt_soc=(int)(X1*100);
527             }
528             position_GB[q]=position_LB[q][xnv];
529         }
530     }
531 }
532
533 // Update Coefficient
534 w = w * wd;
535
536 lcd.setCursor(0,0);
537 lcd.print(cost_GB);
538 lcd.setCursor(9,0);
539 lcd.print(opt_loss);
540 lcd.setCursor(0,1);
541 lcd.print(opt_time);
542 lcd.setCursor(9,1);
543 lcd.print(opt_soc);
544 }

```

```

545
546     Setpoint1=position_GB[0];
547     Setpoint2=position_GB[1];
548     Setpoint3=position_GB[2];
549
550     delay(1000);
551     lcd.clear();
552     lcd.setCursor(0,0);
553     lcd.print("      DONE      ");
554     lcd.setCursor(0,1);
555     lcd.print(Setpoint1);
556     lcd.setCursor(6,1);
557     lcd.print(Setpoint2);
558     lcd.setCursor(12,1);
559     lcd.print(Setpoint3);
560
561     delay(2000);
562
563     lcd.clear();
564
565     /* START CHARGING */
566     state=0;;
567 }
568 }
569
570 float Costfunc(byte S, float Qm, float SOC_Init, float Tin)
571 {
572
573     X1          = SOC_Init;
574     X2          = 0;
575     a1          = exp(-Ts/(Rp*Cp));
576     b1          = Rp * (1 - exp(-Ts/(Rp*Cp)));
577     eta         = 0.9;
578
579     U[0] = -position_now[0][S];
580     U[1] = -position_now[1][S];
581     U[2] = -position_now[2][S];
582
583     Ymax=14.5240;
584     Ymin=SOCCRead(1,SOC_Init);
585
586     L=0;
587     T=0;
588
589     finsimt=Tin;
590
591     if (Tin>=Tmax)           {ST=1;}
592     else if (Tin<Tmin)       {ST=0;}
593     else if ((Tin>=Tmin)&&(Tin<Tmax)) {ST=0;}
594
595     /* TIME AND LOSS CALCULATION */

```

```

596
597 for (loops=0; loops<finsimt; loops++){
598
599     if ((0<=X1)&&(X1<=0.1))        {s=0;}
600     else if ((0.1<X1)&&(X1<=0.98)) {s=1;}
601     else if ((0.98<X1)&&(X1<=1))   {s=2;}
602     else if (X1>1)                {break;}
603
604     Y=SOCRead(1,X1)-X2-Ro*U[s];
605     X1=X1-((eta * (Ts/(3600*Qm)))*U[s]);
606     X2=a1*X2+b1*U[s];
607
608     T=T+Ts;
609     L=L+(Ts*U[s]*U[s]*(Ro+Rp));
610
611 }
612
613 /* COST CALCULATION */
614
615 if (ST==1){
616     alpha=0.6;
617     beta=0.4;
618 }
619 else{
620     alpha=0;
621     beta=0.4;
622 }
623
624 J1=(T-Tmin)/(Tmax-Tmin);
625 J2=(L-Lmin)/(Lmax-Lmin);
626 J3=(Y-Ymax)/(Ymin-Ymax);
627
628 J=sqrt((alpha*J1*J1)+(beta*J2*J2)+((1-(alpha+beta))*J3*J3));
629
630 return J,T,L,X1;
631 }
632
633 float SOCRead(int type, float in)
634 {
635     int i;
636     float out;
637     float data;
638
639     if (type==0){
640
641         for (i=1; i<202; i=i+2)
642         {
643             data=pgm_read_float_near(Vb_Conv+i);
644             if (data>=in)
645             {
646                 out=pgm_read_float_near(Vb_Conv+(i-1))*100;

```



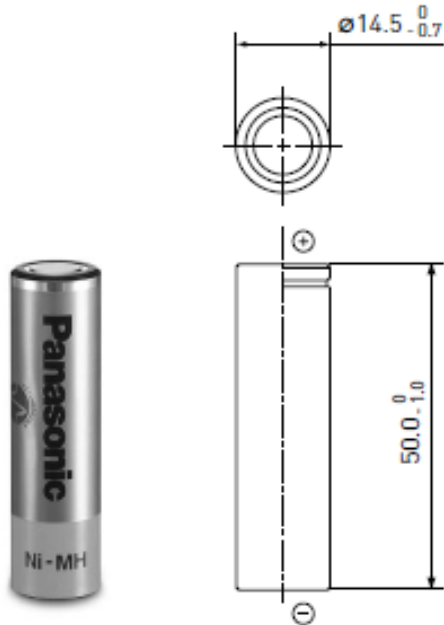
```

647         break;
648     }
649 }
650 }
651 else if (type==1){
652     for (i=0; i<202; i=i+2)
653     {
654         data=pgm_read_float_near(SOC_Conv+i);
655         if (data>=in)
656         {
657             out=pgm_read_float_near(SOC_Conv+(i+1));
658             break;
659         }
660     }
661 }
662
663 return out;
664 }
665
666 void DAC(int input)
667 {
668     int out;
669     int s;
670     int a;
671     int N[10];
672
673     for (a=0;a<10;a++)
674     {
675         s = input%2;
676         input = input/2;
677         if (s==0) {N[a]=0;}
678         else if (s==1){N[a]=1;}
679     }
680
681     digitalWrite(DAC0, N[0]);
682     digitalWrite(DAC1, N[1]);
683     digitalWrite(DAC2, N[2]);
684     digitalWrite(DAC3, N[3]);
685     digitalWrite(DAC4, N[4]);
686     digitalWrite(DAC5, N[5]);
687     digitalWrite(DAC6, N[6]);
688     digitalWrite(DAC7, N[7]);
689     digitalWrite(DAC8, N[8]);
690     digitalWrite(DAC9, N[9]);
691 }

```

3. Datasheet baterai

DIMENSIONS (MM)



SPECIFICATIONS

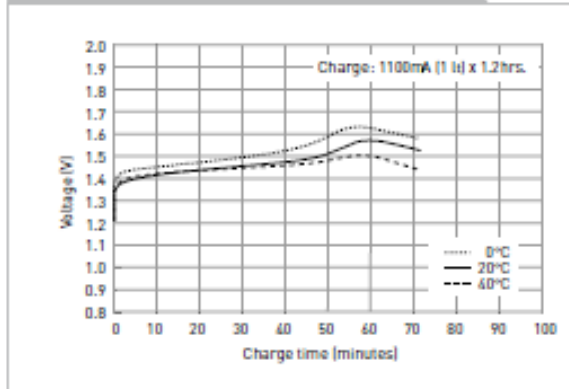
Name		HHR-110AA/FT
Diameter (mm)		14.5 +0/-0.7
Height (mm)		50.0 +0/-1.0
Approximate weight (g)		24
Nominal voltage (V)		1.2
Discharge capacity**	Average** (mAh)	1,180
	Rated/min. (mAh)	1,100
Approx. internal impedance at 1,000Hz at charged state (mΩ)		16
Charge	Standard (mA x hrs.)	110 x 16
	Rapid** (mA x hrs.)	1,100 x 1.2
Ambient temperature	Charge (°C)	Standard: 0 to +45 Rapid: 0 to +40
	Discharge (°C)	-10 to +65
Storage (°C)	<1 year	-20 to +35
	<3 months	-20 to +45
	<1 month	-20 to +55

** After charging at 0.1I for 16 hours, discharging at 0.2I.

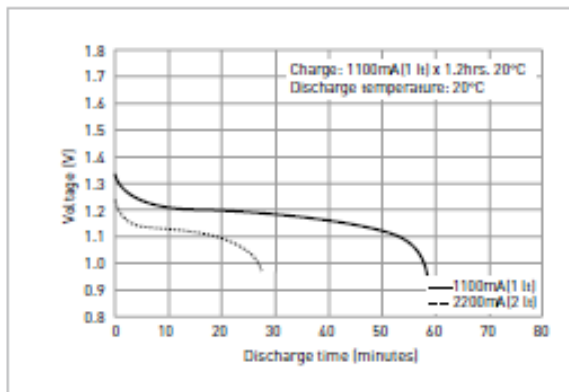
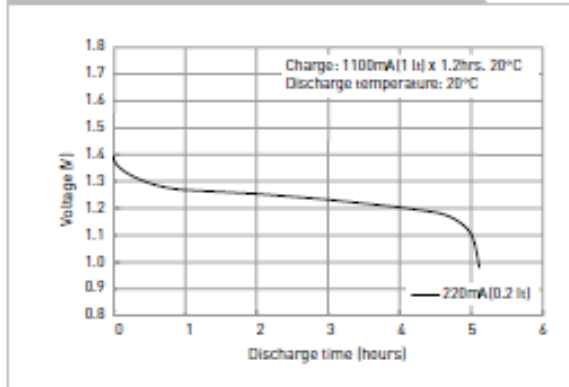
** For reference only.

** Need specially designed control system. Please contact Panasonic for details.

TYPICAL CHARGE CHARACTERISTICS



TYPICAL DISCHARGE CHARACTERISTICS



4. Datasheet Komponen

4.1. Datasheet ATmega32A

Features

- High-performance, Low-power AVR[®] 8-bit Microcontroller
- Advanced RISC Architecture
 - 131 Powerful Instructions – Most Single-clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16 MHz
 - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
 - 32K Bytes of In-System Self-programmable Flash program memory
 - 1024 Bytes EEPROM
 - 2K Byte Internal SRAM
 - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/100 years at 25°C⁽¹⁾
 - Optional Boot Code Section with Independent Lock Bits
 - In-System Programming by On-chip Boot Program
 - True Read-While-Write Operation
 - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
 - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
 - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Four PWM Channels
 - 8-channel, 10-bit ADC
 - 8 Single-ended Channels
 - 7 Differential Channels in TQFP Package Only
 - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
 - Byte-oriented Two-wire Serial Interface
 - Programmable Serial USART
 - Master/Slave SPI Serial Interface
 - Programmable Watchdog Timer with Separate On-chip Oscillator
 - On-chip Analog Comparator
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated RC Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
 - 32 Programmable I/O Lines
 - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
 - 2.7 - 5.5V for ATmega32A
- Speed Grades
 - 0 - 16 MHz for ATmega32A
- Power Consumption at 1 MHz, 3V, 25°C for ATmega32A
 - Active: 0.6 mA
 - Idle Mode: 0.2 mA
 - Power-down Mode: < 1 µA



8-bit **AVR[®]**
Microcontroller
with 32K Bytes
In-System
Programmable
Flash

ATmega32A

Summary



8155AS-AVR-06/08

4.2. Datasheet TIP142



DARLINGTON COMPLEMENTARY SILICON POWER TRANSISTORS

...designed for general-purpose amplifier and low speed switching applications

FEATURES:

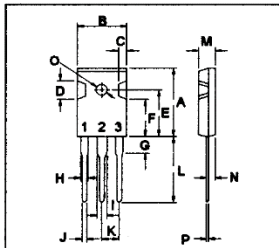
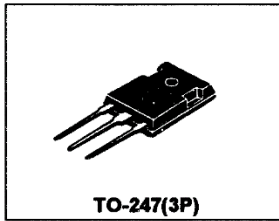
- * Collector-Emitter Sustaining Voltage-
 $V_{CE(sus)}$ = 60 V (Min) - TIP140, TIP145
 = 80 V (Min) - TIP141, TIP146
 = 100 V (Min) - TIP142, TIP147
- * Collector-Emitter Saturation Voltage
 $V_{CE(sat)}$ = 2.0 V (Max.) @ $I_C = 5.0$ A
- * Monolithic Construction with Built-in Base-Emitter Shunt Resistor

NPN	PNP
TIP140	TIP145
TIP141	TIP146
TIP142	TIP147

10 AMPERE DARLINGTON COMPLEMENTARY SILICON POWER TRANSISTORS
 60-100 VOLTS
 125 WATTS

MAXIMUM RATINGS

Characteristic	Symbol	TIP140 TIP145	TIP141 TIP146	TIP142 TIP147	Unit
Collector-Emitter Voltage	V_{CEO}	60	80	100	V
Collector-Base Voltage	V_{CBO}	60	80	100	V
Emitter-Base Voltage	V_{EBO}	5.0			V
Collector Current-Continuous -Peak	I_C I_{CM}	10 15			A
Base Current	I_B	0.5			A
Total Power Dissipation @ $T_C = 25^\circ\text{C}$ Derate above 25°C	P_D	125 1.0			W W/ $^\circ\text{C}$
Operating and Storage Junction Temperature Range	T_J, T_{STG}	- 65 to +150			$^\circ\text{C}$

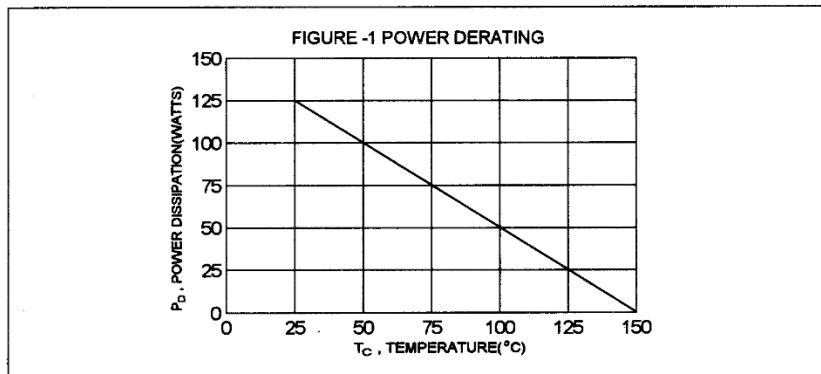


PIN 1.BASE
 2.COLLECTOR
 3.EMITTER

DIM	MILLIMETERS	
	MIN	MAX
A	20.63	22.38
B	15.38	16.20
C	1.90	2.70
D	5.10	6.10
E	14.81	15.22
F	11.72	12.84
G	4.20	4.50
H	1.82	2.46
I	2.92	3.23
J	0.89	1.53
K	5.26	5.66
L	18.50	21.50
M	4.68	5.36
N	2.40	2.80
O	3.25	3.65
P	0.55	0.70

THERMAL CHARACTERISTICS

Characteristic	Symbol	Max	Unit
Thermal Resistance Junction to Case	$R_{\theta jc}$	1.0	$^\circ\text{C/W}$



TIP140, TIP141, TIP142 NPN / TIP145, TIP146, TIP147 PNP

ELECTRICAL CHARACTERISTICS ($T_C = 25^\circ\text{C}$ unless otherwise noted)

Characteristic	Symbol	Min	Max	Unit
----------------	--------	-----	-----	------

OFF CHARACTERISTICS

Collector - Emitter Sustaining Voltage (1) ($I_C = 30\text{ mA}$, $I_B = 0$)	TIP140, TIP145 TIP141, TIP146 TIP142, TIP147	$V_{CE(sus)}$	60 80 100	V
Collector Cutoff Current ($V_{CE} = 30\text{ V}$, $I_B = 0$) ($V_{CE} = 40\text{ V}$, $I_B = 0$) ($V_{CE} = 50\text{ V}$, $I_B = 0$)	TIP140, TIP145 TIP141, TIP146 TIP142, TIP147	I_{CEO}	2.0 2.0 2.0	mA
Collector Cutoff Current ($V_{CE} = 60\text{ V}$, $I_B = 0$) ($V_{CE} = 80\text{ V}$, $I_B = 0$) ($V_{CE} = 100\text{ V}$, $I_B = 0$)	TIP140, TIP145 TIP141, TIP146 TIP142, TIP147	I_{CBO}	1.0 1.0 1.0	mA
Emitter Cutoff Current ($V_{EB} = 5.0\text{ V}$, $I_C = 0$)		I_{EBO}	2.0	mA

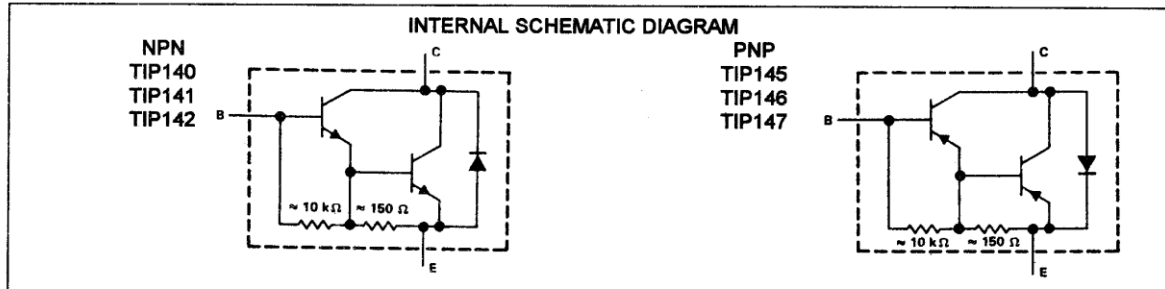
ON CHARACTERISTICS (1)

DC Current Gain ($I_C = 5.0\text{ A}$, $V_{CE} = 4.0\text{ V}$) ($I_C = 10\text{ A}$, $V_{CE} = 4.0\text{ V}$)	h_{FE}	1000 500		
Collector-Emitter Saturation Voltage ($I_C = 5.0\text{ A}$, $I_B = 10\text{ mA}$) ($I_C = 10\text{ A}$, $I_B = 40\text{ mA}$)	$V_{CE(sat)}$		2.0 3.0	V
Base-Emitter Saturation Voltage ($I_C = 10\text{ A}$, $I_B = 40\text{ mA}$)	$V_{BE(sat)}$		3.5	V
Base-Emitter On Voltage ($I_C = 10\text{ A}$, $V_{CE} = 4.0\text{ V}$)	$V_{BE(on)}$		3.0	V

SWITCHING CHARACTERISTICS

Delay Time	$V_{CC} = 30\text{ V}$, $I_C = 5.0\text{ A}$ $I_{B1} = -I_{B2} = 20\text{ mA}$, $t_p = 20\text{ us}$, Duty Cycle $\leq 2.0\%$	t_d	0.15(Typ)		us
Rise Time		t_r	0.55(Typ)		us
Storage Time		t_s	2.5(Typ)		us
Fall Time		t_f	2.5(Typ)		us

(1) Pulse Test: Pulse width = 300 us , Duty Cycle $\leq 2.0\%$



(Halaman ini sengaja dikosongkan)

BIODATA PENULIS



Nuh Enola, lahir di Malang, 04 Desember 2001. Penulis telah menyelesaikan pendidikan formal di SDN Pagerwojo Sidoarjo (2008 – 2014), MTsN 3 Malang (2014 – 2016), dan SMAN 1 Lawang (2016 – 2018). Kemudian, pada tahun 2018, penulis memutuskan untuk melanjutkan studi S-1 di Departemen Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS). Pada semester 4 perkuliahan, penulis memilih Teknik Sistem Pengaturan sebagai bidang studi karena ketertarikan penulis terhadap perkembangan teknologi di bidang studi tersebut. Selama masa perkuliahan, penulis aktif mengikuti kegiatan organisasi dan kepanitiaan yang terdapat di kampus, seperti Himpunan Mahasiswa Teknik Elektro ITS, UKM

Teater Tiyang Alit ITS, Pemandu FTEIC ITS, dan beberapa kegiatan lainnya. Selain itu, penulis juga aktif sebagai asisten laboratorium di Lab. Kontrol dan Otomasi AJ-104 Teknik Sistem Pengaturan ITS. Untuk menghubungi penulis, dapat melalui alamat email berikut nuhenola@gmail.com.