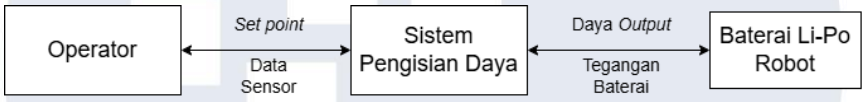
BAB III

PERANCANGAN DAN IMPLEMENTASI SISTEM

3.1 Tinjauan Desain Sistem

3.1.1 Desain Sistem Keseluruhan

Data flow diagram (DFD) dapat digunakan untuk menggambarkan alur sistem secara keseluruhan, khususnya alur pengolahan data.



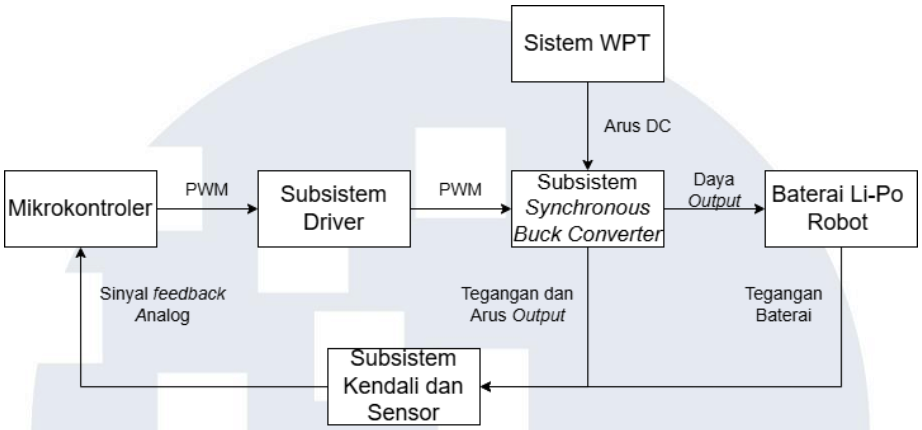
Gambar 3.1 DFD Level 0 Sistem Pengisian Daya

Tabel 3.1 Penjelasan DFD Level 0 Sistem Pengisian Daya

Parameter	Keterangan
Input	• *Set point* tegangan dan arus dari operator • Kondisi tegangan baterai
Output	• Daya ke Baterai • Data sensor arus dan tegangan
Fungsi	• Mengatur mode yang akan dijalankan oleh sistemaat pengisian daya berlangsung • Menampilkan data sensor untuk *monitoring*

Gambar 3.1 menjelaskan DFD level 0 untuk sistem secara keseluruhan. Fungsi DFD tersebut dijelaskan pada Tabel 3.1, di mana *set point* tegangan dan arus *output* dapat diatur oleh pengguna. Pengguna juga dapat membaca data sensor tegangan dan arus melalui UART yang terhubung dengan laptop untuk mengkalibrasi kedua sensor. Setelah sensor telah terkalibrasi dan *set point* telah ditentukam, sistem tidak membutuhkan operator lagi untuk dapat berjalan, namun operator tetap dapat ada untuk memonitor keamanan sistem.

3.1.2 Desain Subsistem



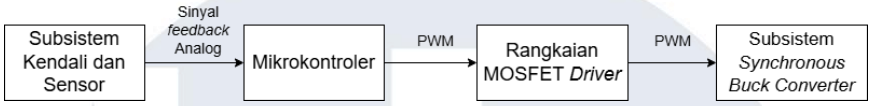
Gambar 3.2 DFD Level 1 Sistem Pengisian Daya

Tabel 3.2 Penjelasan DFD Level 1 Sistem Pengisian Daya

Parameter	Keterangan
Input	• Arus DC dari sistem WPT • Tegangan Baterai • Data sensor tegangan dan arus
Output	• Daya ke Baterai
Fungsi	• Mengubah arus AC menjadi arus DC yang tekontrol untuk pengisian daya baterai robot.

Gambar 3.2 merupakan DFD level 1 pada sistem pengsian daya. DFD tersebut menjelaskan hubungan antar subsistem, input, dan output dari sistem. Tabel 3.2 menjelaskan input dari subsistem *synchronous buck converter* merupakan arus DC hasil penyearah, selain itu tegangan baterai juga menjadi input pada subsistem kendali dan sensor mengetahui kondisi baterai. Setelah itu, data dari sensor akan diolah oleh mikrokontroler untuk menjalankan kontrol CC-CV. *Output* dari sistem tentunya merupakan daya yang telah terkontrol yang berfungsi untuk mengisi daya baterai robot.

3.1.2.1 Desain Subsistem *Driver*



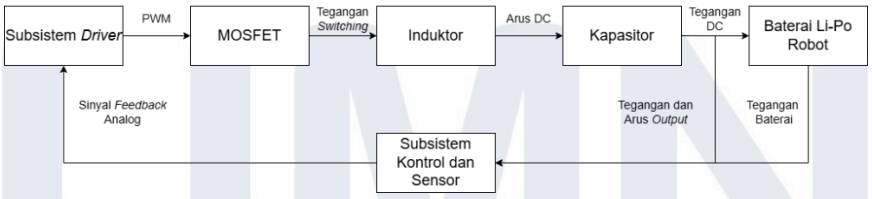
Gambar 3.3 DFD Level 2 Subsistem *Driver*

Tabel 3.3 Penjelasan DFD Level 2 Subsistem *Driver*

Parameter	Keterangan
Input	• Data dari subsistem Kendali dan Sensor
Output	• Sinyal PWM dengan *duty cycle* yang sudah diatur oleh mikrokontroler
Fungsi	• Mengendalikan *switching* MOSFET pada subsistem *synchronous buck converter*

Pada Gambar 3.3 alur data pada subsistem ini ditunjukkan. Tabel 3.3 menjelaskan bahwa input dari rangkaian MOSFET *driver* merupakan sinyal PWM yang *duty cycle* nya disesuaikan berdasarkan hasil *feedback* dari subsistem kendali dan sensor. Fungsi dari subsistem ini merupakan memperkuat sinyal PWM yang dihasilkan oleh mikrokontroler agar mampu mengendalikan *switching* MOSFET pada subsistem *synchronous buck converter*.

3.1.2.2 Desain Subsistem Synchronous Buck Converter



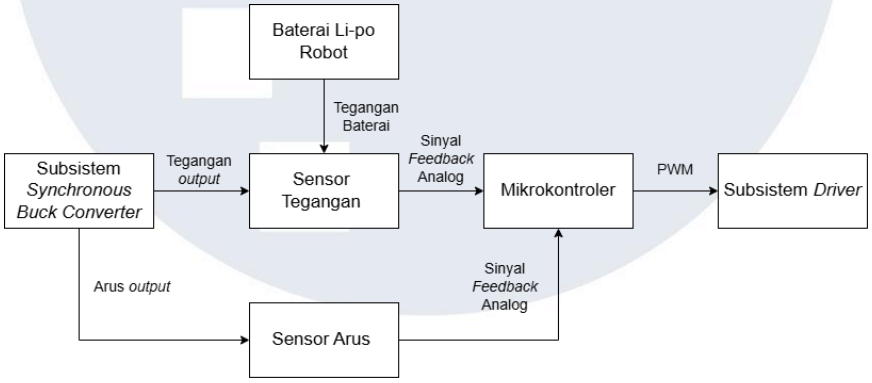
Gambar 3.4 DFD Level 2 Subsistem *Synchronous Buck Converter*

Tabel 3.4 Penjelasan DFD Level 2 Subsistem *Synchronous Buck Converter*

Parameter	Keterangan
Input	• Sinyal PWM dari subsistem *driver*
Output	• Daya DC yang sudah terkontrol • Tegangan dan arus *output* ke subsistem kontrol dan sensor
Fungsi	• Mengisi daya baterai robot dengan metode CC-CV

Gambar 3.4 menjelaskan alur data dari subsistem *synchronous buck converter*. Tabel 3.4 menjelaskan bahwa input dari subsistem ini adalah sinyal PWM yang digunakan untuk *switching* kedua MOSFET. Setelah itu, proses *switching* menghasilkan tegangan berpulsa, yang kemudian diratakan oleh komponen-komponen pasif, yaitu induktor dan kapasitor. Output dari komponen ini merupakan daya DC yang sudah terkontrol dan digunakan untuk mengisi daya baterai robot. Selain itu, *output* juga akan masuk ke subsistem kendali dan sensor untuk diproses.

3.1.2.3 Desain Subsistem Kendali dan Sensor



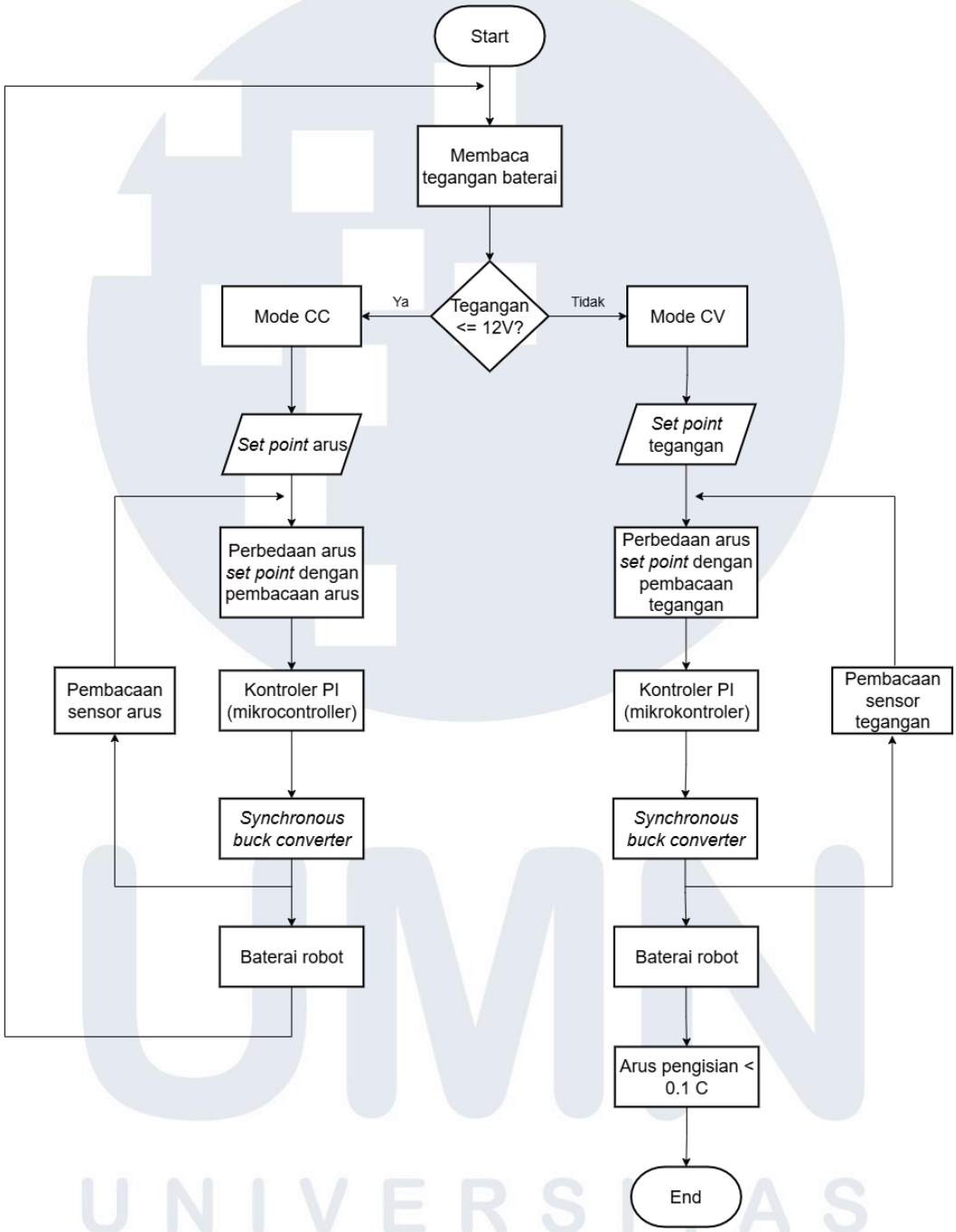
Gambar 3.5 DFD Level 2 Subsistem Kendali dan Sensor

Tabel 3.5 Penjelasan DFD Level 2 Subsistem Kendali dan Sensor

Parameter	Keterangan
Input	• Tegangan *output* • Arus *output*
Output	• Sinyal *feedback* analog sensor tegangan • Sinyal *feedback* analog sensor arus
Fungsi	• Mengukur tegangan dan arus *output* • Mengontrol tegangan dan arus *output* dengan algoritma kontrol PI

Gambar 3.5 menunjukkan alur data pada subsistem kendali dan sensor. Selanjutnya, Tabel 3.5 menjelaskan input dari subsistem ini adalah tegangan dan arus *output* dalam bentuk sinyal analog. Mikrokontroler akan memproses sinyal analog menjadi digital (ADC) dan memproses dengan algoritma kontrol PI untuk menghasilkan PWM dengan *duty cycle* yang telah disesuaikan.

Untuk mengerti alur kerja kontrol PI yang digunakan pada sistem, dibutuhkan sebuah *flowchart* sebagai berikut.

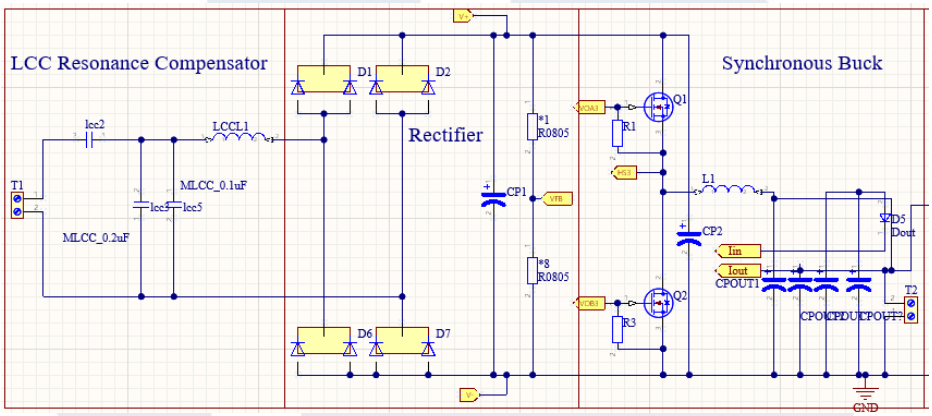


Gambar 3.6 *Flowchart* Pengisian Daya Metode CC-CV dengan Kontrol PI

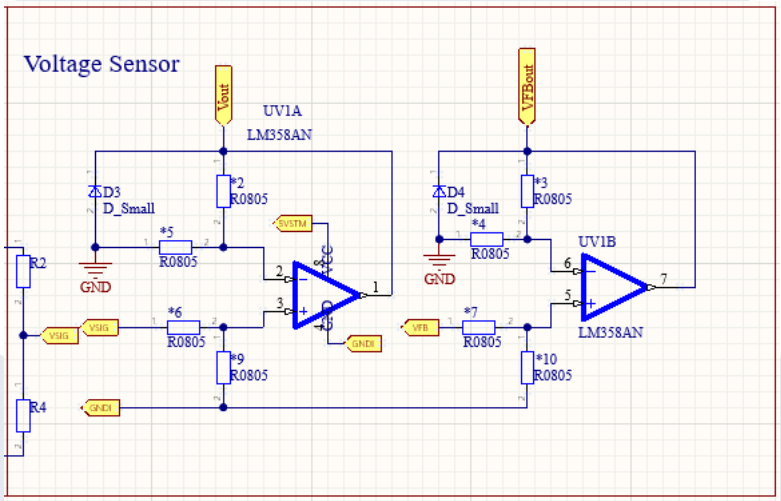
Gambar 3.6 menjelaskan *flowchart* dari metode CC-CV dengan menggunakan kontroler PI. Sensor tegangan dan arus masing-masing akan memberikan *feedback* ke mikrokontroler. Jika tegangan baterai kurang dari 12 V, maka mode CC akan dijalankan. Mode CC akan membandingkan *set point* arus, yaitu 3 A dengan pembacaan arus *output* dari *synchronous buck converter*. Jika tidak sesuai, maka algoritma kontrol PI yang ada di mikrokontroler akan menyesuaikannya. Ketika baterai sudah mencapai 12 V, maka mode CV akan berlangsung. Sama seperti mode CC, pada mode CV nilai pembacaan sensor tegangan akan dibandingkan dengan *set point* tegangan, yaitu 12 V. Jika tidak sesuai, maka algoritma kontrol PI akan menyesuaikannya. Sistem pengisian daya akan selesai jika sudah berada pada mode CV dan arus sudah mencapai kondisi *cut off,* yaitu 0.1 C atau sekitar 300 mA.

3.1.3 Diagram Sistem

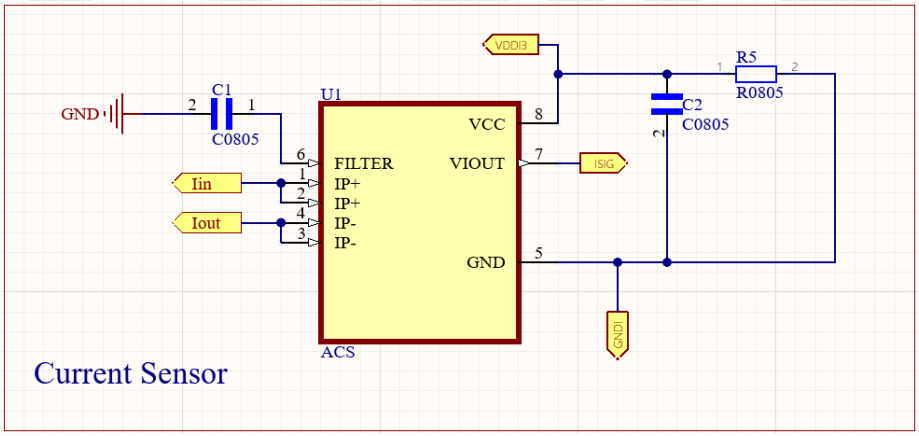
Sistem akan menggunakan PCB untuk menghubungkan seluruh subsistem. Berikut merupakan desain skematik dari sistem.



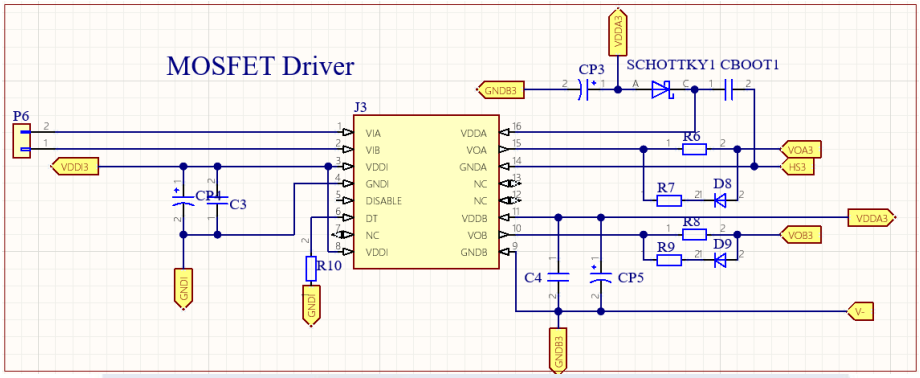
(a)



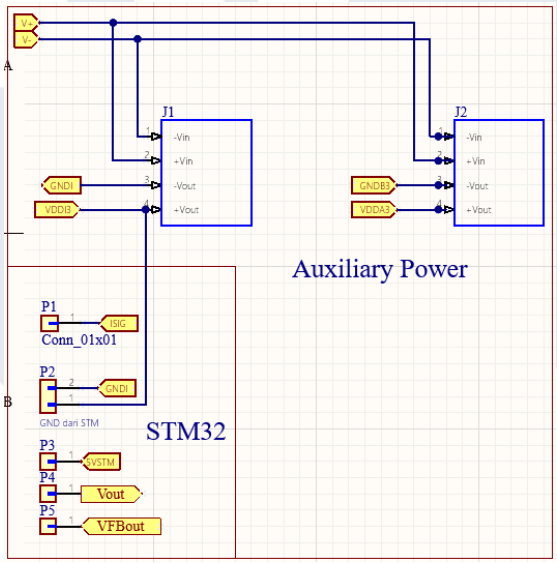
(b)



(c)



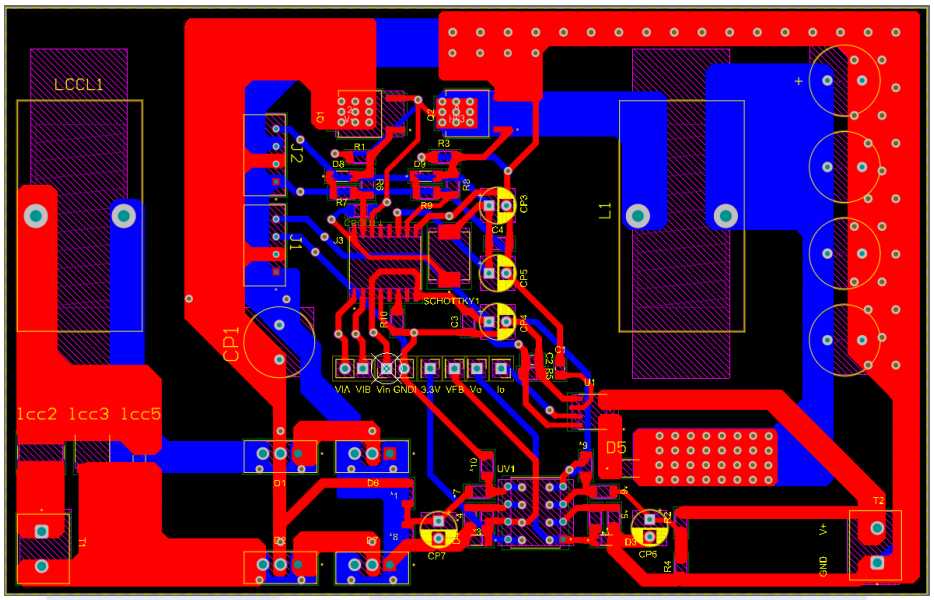
(d)



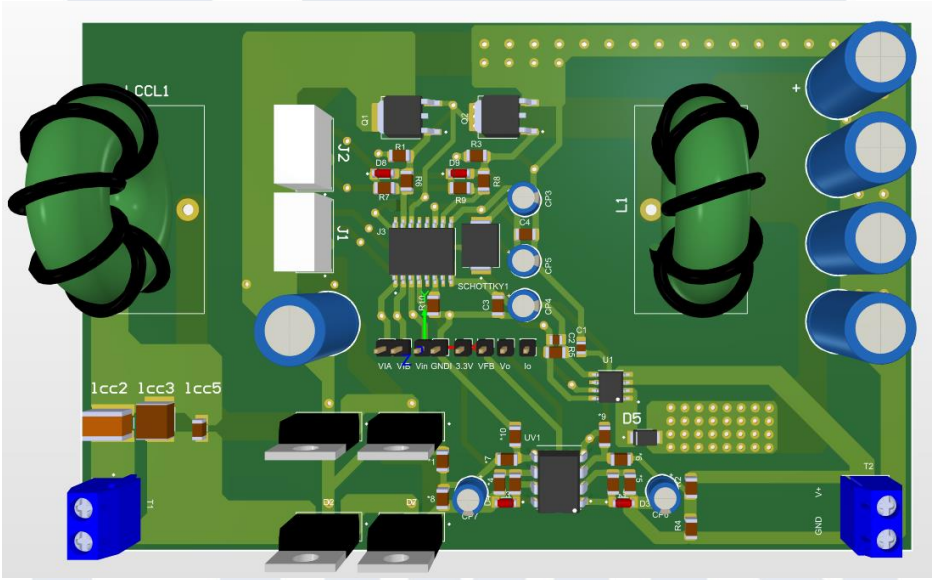
(e)

Gambar 3.7 Desain Skematik Sistem, (a) Subsistem *Synchronous Buck Converter* yang terhubung dengan kompensasi pada sisi penerima WPT dan *rectifier*, (b) Subsistem Sensor Tegangan, (c) Subsistem Sensor Arus, (d) Subsistem *Driver*, (e) *Auxiliary Power*

Gambar 3.7 merupakan desain skematik sistem yang dibagi per subsistem. Desain PCB akan dibuat dari desain skematik ini.



(a)



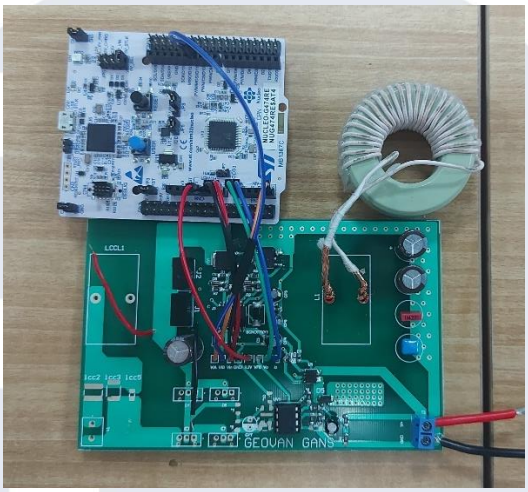
(b)

Gambar 3.8 Desain PCB Sistem, (a) Tampilan 2D, (b) Tampilan 3D

Gambar 3.8 menunjukkan desain PCB yang digunakan pada sistem *synchronous buck converter* yang digunakan sebagai pengisian daya baterai robot. Desain ini mencangkup seluruh subsistem yang terintegrasi pada satu papan.

3.2 Implementasi Sistem

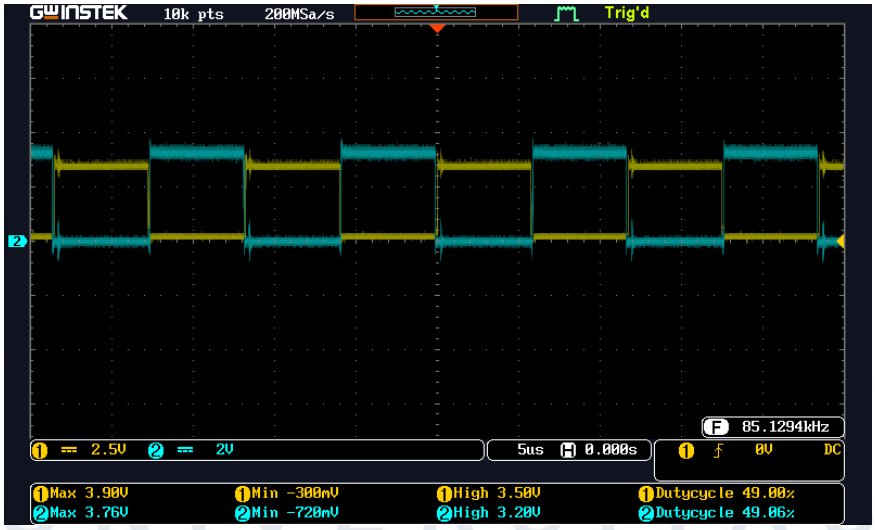
3.2.1 Hasil Implementasi



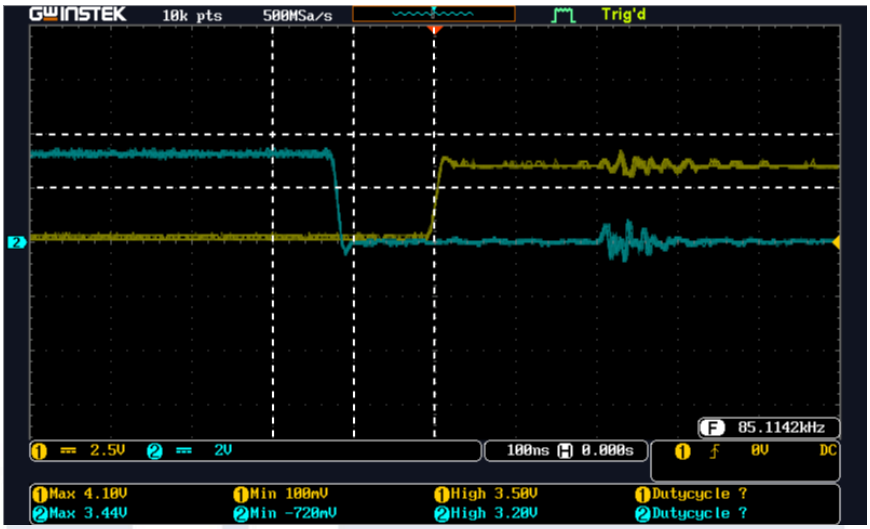
Gambar 3.9 PCB *Hardware*

Gambar 3.9 merupakan *hardware* dari desain PCB yang telah dibuat. STM32 terhubung dengan PCB dengan menggunakan kabel *jumper* dan posisi *core* induktor tidak terpasang pada PCB saat tahap percobaan.

3.2.1.1 Hasil Implementasi Subsistem *Driver*



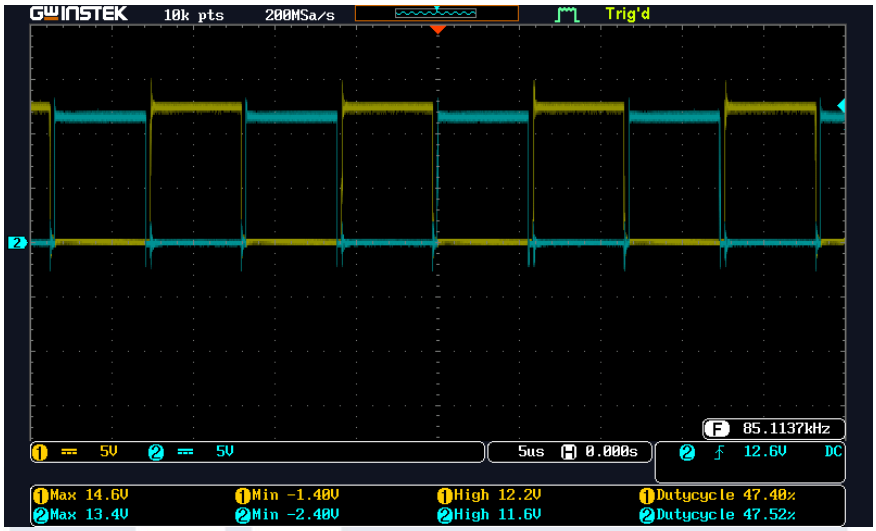
(a)



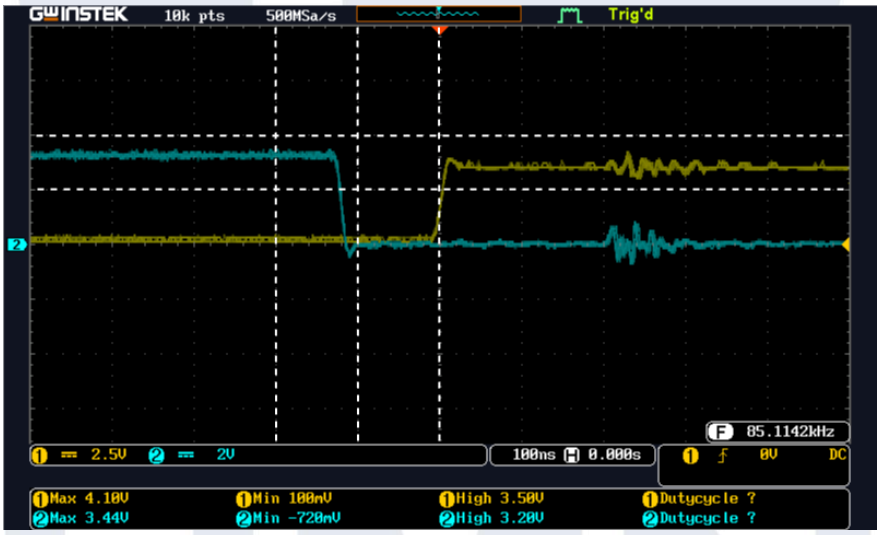
(b)

Gambar 3.10 Sinyal PWM pada Input *Driver*, (a) Sinyal Input *High-Side* (CH1) dan *Low-Side* (CH2) [*Time*: 5 µs/div; *Scale*: 2.5 V/div (CH1), 2 V/div (CH2)], (b) Sinyal Input *High-Side* (CH1) dan *Low-Side* (CH2) [*Time*: 100 ns/div; *Scale*: 2.5 V/div (CH1), 2 V/div (CH2)]

Sinyal pada Gambar 3.10 (a) merupakan sinyal PWM yang ada di sisi input *driver*, atau sinyal PWM yang dihasilkan oleh STM32, sedangkan Gambar 3.9 (b) merupakan versi *zoom in* dari Gambar 3.10 (a). Sinyal tersebut masih berada di sekitar 3.3 V sesuai dengan tegangan PWM maksimum yang dapat dihasilkan oleh STM32. *Dead time* dari sinyal PWM adalah sekitar 120 ns atau sekitar 1% dari periode *switching*. Nilai ini masih dapat diterima pada sistem yang menggunakan aplikasi *high-side/low-side driver*. Frekuensi yang digunakan adalah 85 kHz dengan *duty cycle* yang diatur oleh STM32 sebesar 50%.



(a)



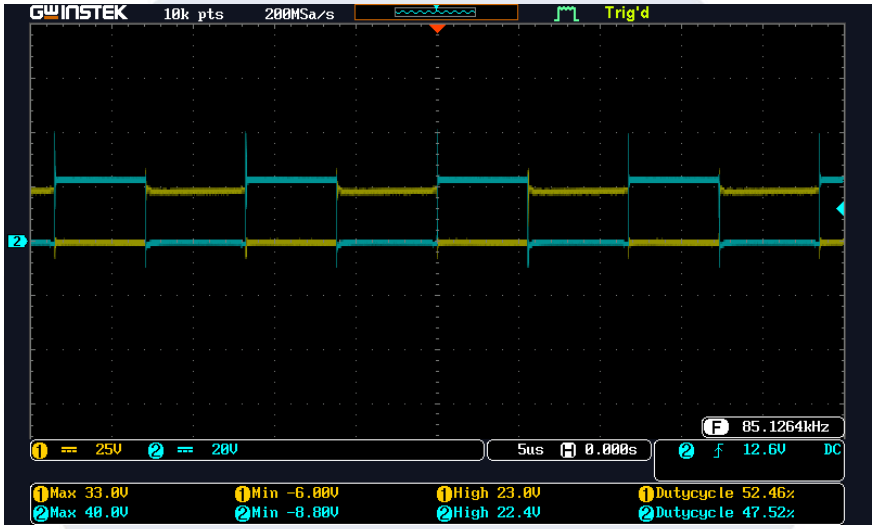
(b)

Gambar 3.11 Sinyal PWM pada *Gate-Source* (V_{GS}), (a) Sinyal V_{GS} *High-Side* (CH1) dan *Low-Side* (CH2) [*Time*: 5 µs/div; *Scale*: 5 V/div], (b) Sinyal V_{GS} *High-Side* (CH1) dan *Low-Side* (CH2) [*Time*: 100 ns/div; *Scale*: 5V/div]

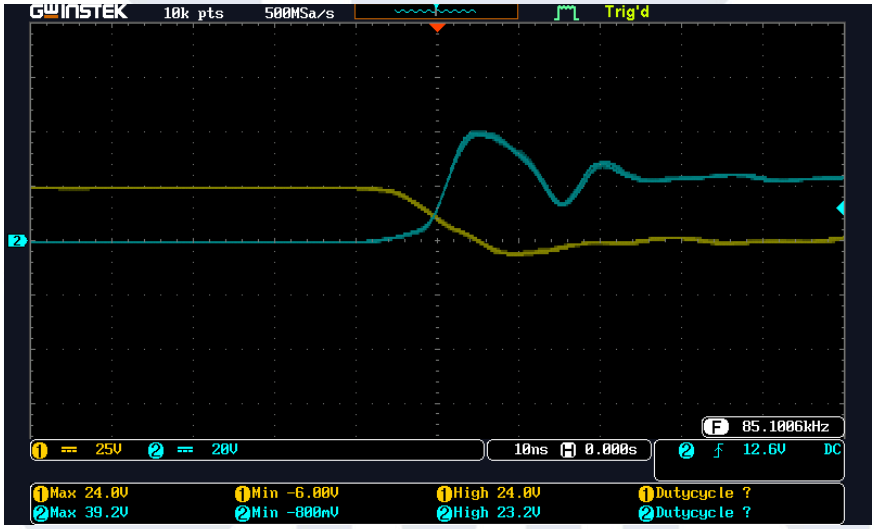
Gambar 3.11 menunjukkan hasil PWM yang dihasilkan oleh rangkaian *driver*. Sinyal tersebut merupakan sinyal dari kedua PWM pada *high-side* dan *low-side* MOSFET. Tegangan sesuai dengan suplai tegangan pada sisi *output driver*, yaitu sekitar 12 V. *Overshoot* yang dimiliki sinyal tidak terlalu tinggi, masih di batas toleransi sistem. *Dead time* ada pada sekitar 220 ns atau sekitar 2% dari periode *switching*. *Dead time* yang sebelumnya sudah diberikan oleh mikrokontroler ditambah kembali di *driver* untuk memastikan keamanan *switching*. Frekuensi yang didapatkan pada sinyal V_{GS} masih sama dengan yang dihasilkan oleh

mikrokontroler, sedangkan *duty cycle* berkurang karena ditambahkan *dead time* yang tentunya mempengaruhi nilai *duty cycle*.

3.2.1.2 Hasil Implementasi Subsistem Synchronous Buck Converter



(a)

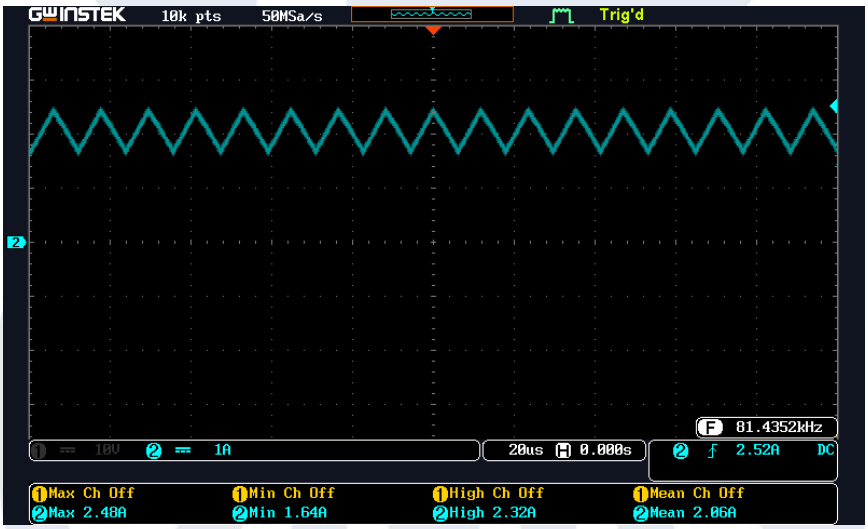


(b)

Gambar 3.12 Sinyal PWM pada *Drain-Source* (V_{DS}), (a) Sinyal V_{DS} *High-Side* (CH1) dan *Low-Side* (CH2) [*Time*: 5 µs/div; *Scale*: 25 V/div (CH1), 20 V/div (CH2)], (b) Sinyal V_{DS} *High-Side* (CH1) dan *Low-Side* (CH2) [*Time*: 10 ns/div; *Scale*: 25V/div (CH1), 20 V/div (CH2)]

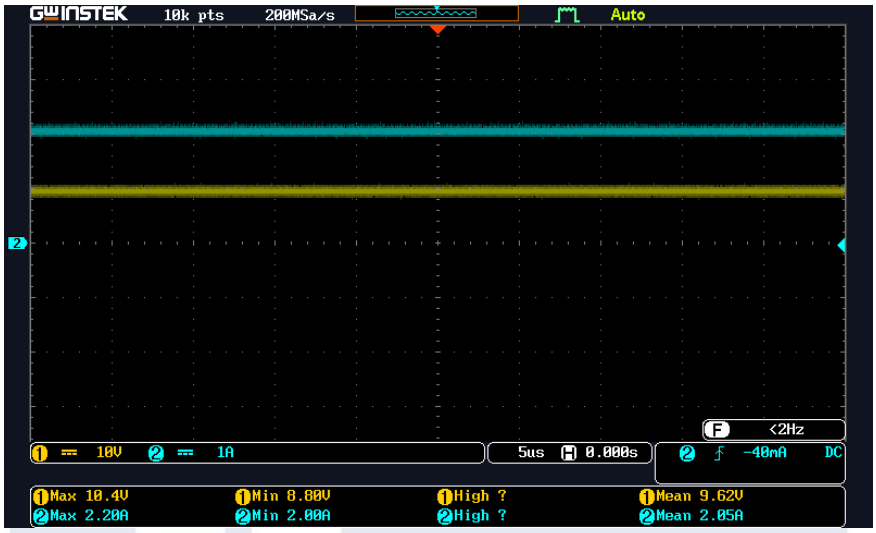
Gambar 3.12 menunjukkan sinyal PWM pada kedua MOSFET selama proses *switching*. Seluruh implementasi pada bagian ini menggunakan beban resisitif sebesar 4 Ω atau dalam keadaan *full load*. *Spike* saat *switching* cukup tinggi dengan nilai maksimal 40 V, namun masih dalam batasan sistem. *Spike* merupakan salah satu efek parasitik yang dimiliki oleh MOSFET. *Duty cycle* yang dimiliki oleh sistem berada pada sekitar 50%, sesuai dengan V_{GS}. Namun, kedua sinyal terjadi *overlap* yang artinya ketika sinyal MOSFET *high-side* belum sepenuhnya OFF, sinyal MOSFET *low-side* sudah ON. Hal ini dapat menyebabkan lonjakan arus dan penurunan efisiensi. *Overlap* dapat terjadi karena efek parasitik pada MOSFET, namun tidak ada *shoot-through* yang terlihat. Sehingga dapat dikatakan bahwa sinyal tersebut terlihat *overlap* karena memang dari karakteristik *switching*.

Selain mengamati sinyal *switching*, arus pada induktor dapat dilihat untuk melihat karakteristik rangkaian. Berikut merupakan arus induktor dengan menggunakan beban 4 Ω yang terhubung pada *output* rangkaian.



Gambar 3.13 Arus Induktor pada *Synchronous Buck Converter*

Gambar 3.13 menunjukkan bahwa induktor bekerja dalam CCM karena sinyal berbentuk segitiga dan tidak menyentuh angka nol. *Ripple* arus pada induktor adalah 0.84 A atau kurang dari 10% dari arus *output*, serupa dengan target *ripple* yang ditentukan ketika menghitung induktansi, ini menunjukkan bahwa induktansi sudah berada pada nilai yang cukup. Selanjutnya, sinyal yang harus dilihat merupakan tegangan dan arus *output*.



Gambar 3.14 Tegangan dan Arus *Output* pada *Synchronous Buck Converter*

Gambar 3.14 menunjukkan bahwa tegangan dan arus pada sisi *output* sudah stabil. Walaupun target *ripple* tegangan ketika menghitung komponen adalah 0.06 V, *ripple* yang didapatkan pada osiloskop sekitar 1.6 V dan masih sangat wajar karena toleransi derau *probe* osiloskop karena frekuensi yang diukur sangat kecil. Selain itu, walau *duty cycle* telah diatur 50% dan tegangan input sebesar 24 V, sistem masih belum mencapai *output* 12 V. Rumus rasio *duty cycle* hanya berlaku pada kondisi ideal, sedangkan pada implementasinya terdapat derau, rugi-rugi pada masing-masing komponen, *switching* ideal, dan sebagainya. Maka dari itu, dibutuhkan sebuah kontrol yang dapat memastikan *output* sesuai dengan *set point*.

3.2.1.3 Hasil Implementasi Subsistem Sensor dan Kendali

Sebelum mengimplementasi sistem kendali, dibutuhkan sensor sebagai *feedback* pada sistem. Untuk sensor arus, IC yang digunakan adalah ACS712 dengan arus maksimum 5 A dan sensor tegangan menggunakan *voltage divider* yang terhubung dengan *op-amp* LM358AN. *Output* kedua sensor akan terhubung dengan pin ADC STM32.

```
for (int i = 0; i < NUM_SAMPLES; i++) {
                //ADC1
                HAL_ADC_Start(&hadc1);
                HAL_ADC_PollForConversion(&hadc1, 10);
                adc1_samples[i] = HAL_ADC_GetValue(&hadc1);

                //ADC2
                HAL_ADC_Start(&hadc2);
                HAL_ADC_PollForConversion(&hadc2, 10);
                adc2_samples[i] = HAL_ADC_GetValue(&hadc2);
        }
        qsort(adc1_samples, NUM_SAMPLES, sizeof(uint16_t), compare_uint16);
        qsort(adc2_samples, NUM_SAMPLES, sizeof(uint16_t), compare_uint16);

        raw1 = adc1_samples[NUM_SAMPLES / 2];
        raw2 = adc2_samples[NUM_SAMPLES / 2];

        //ADC1
        median_buffer1[median_index1++] = raw1;
        if (median_index1 >= N_MEDIANS) {
                median_index1 = 0;
        }
        if (median_count1 < N_MEDIANS) {
                median_count1++;
        }
        //ADC2
        median_buffer2[median_index2++] = raw2;
        if (median_index2 >= N_MEDIANS) {
                median_index2 = 0;
        }
        if (median_count2 < N_MEDIANS) {
                median_count2++;
        }

        //ADC1
        uint32_t sum1 = 0;
        for (int i = 0; i < median_count1; i++) {
                sum1 += median_buffer1[i];
        }
        uint16_t avg_median1 = sum1 / median_count1;

        //ADC2
        uint32_t sum2 = 0;
        for (int i = 0; i < median_count2; i++) {
                sum2 += median_buffer2[i];
        }
        uint16_t avg_median2 = sum2 / median_count2;
```

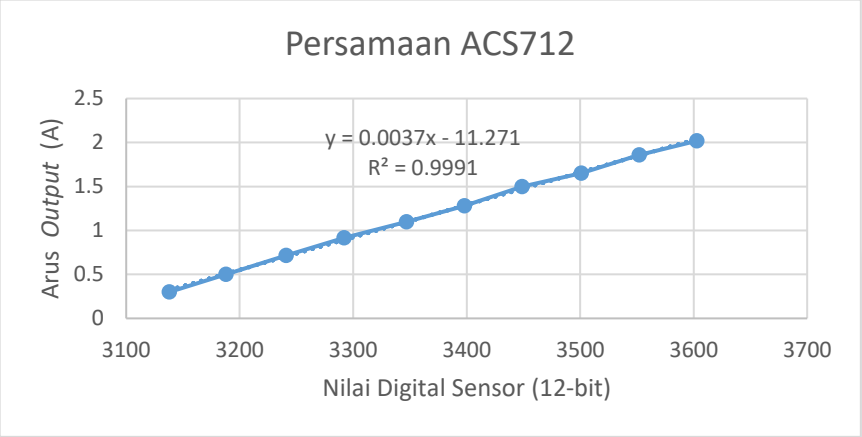
Gambar 3.15 Potongan Kode Pembacaan ADC pada STM32

Gambar 3.15 menjelaskan kode yang digunakan untuk membaca ADC. Pembacaan ADC menggunakan median filter yang kemudian dirata-ratakan kembali untuk memastikan pembacaan stabil.

Agar kontroler dapat membandingkan nilai *feedback* sensor dengan *set point*, nilai ADC harus diubah kembali ke dalam nilai analog. Proses ini memerlukan kalibrasi agar nilai analog dapat merepresentasikan nilai *output* sebenarnya. Berikut merupakan hasil kalibrasi kedua sensor.

Tabel 3.6 Kalibrasi Sensor Arus

Nilai Digital Sensor (12-bit)	Arus *Output* (A)
3138	0.3
3188	0.5
3241	0.715
3292	0.915
3347	1.1
3398	1.28
3449	1.5
3501	1.65
3552	1.86
3603	2.02

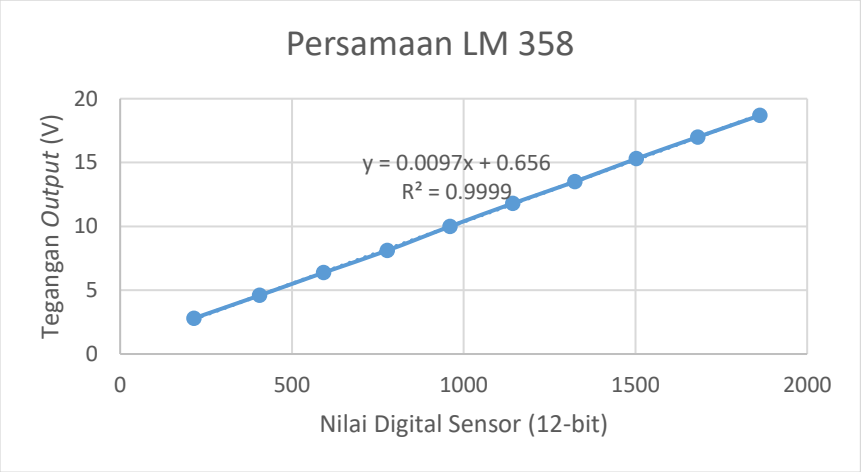


Gambar 3.16 Kurva Persamaan Sensor Arus ACS712

Tabel 3.6 merupakan hasil kalibrasi sensor arus ACS712, percobaan dilakukan sebanyak 10 kali agar hasil kalibrasi semakin akurat. Dari tabel tersebut, didapatkan kurva untuk mendapatkan nilai persamaan yang dimiliki sensor seperti pada Gambar 3.16.

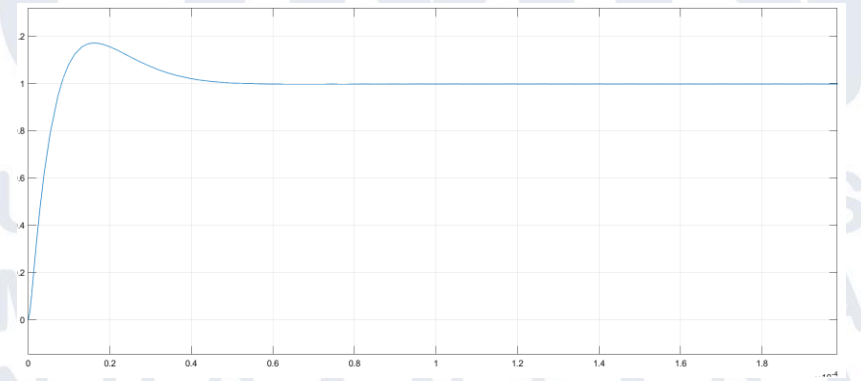
Tabel 3.7 Kalibrasi Sensor Tegangan

Nilai Digital Sensor (12-bit)	Tegangan *Output* (V)
215	2.8
406	4.59
593	6.38
778	8.11
961	10
1143	11.8
1324	13.5
1503	15.3
1682	17
1863	18.7



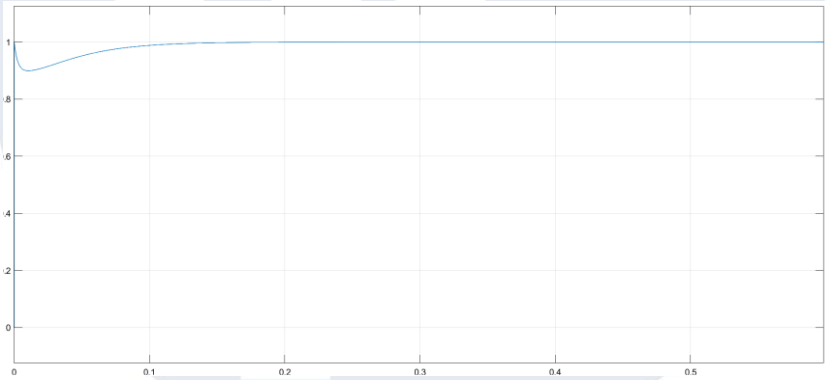
Gambar 3.17 Kurva Persamaan Sensor Tegangan LM358AN

Tabel 3.7 merupakan hasil kalibrasi dari sensor tegangan yang menggunakan LM358. Sama seperti sensor arus, percobaan dilakukan sebanyak 10 kali agar kalibrasi semakin akurat. Dari tabel tersebut, kurva pada Gambar 3.17 digunakan untuk mendapatkan persamaan sensor tegangan. Setelah kedua sensor *feedback* sudah terkalibrasi, kontrol PI dapat diimplementasikan.



Gambar 3.18 Respons Sistem Kontrol Tegangan Menggunakan Simulasi

Sebelum mengimplentasikan langsung kepada rangkaian, dilakukan simulasi untuk melihat respons sistem ketika diberi kontroler PID. Gambar 3.18 merupakan respons sistem ketika menggunakan fungsi alih untuk mengontrol arus dengan simulasi. Nilai K_p, K_i, dan K_d yang didapatkan masing-masing adalah 14.6, 200, dan 0.0002. Secara simulasi, dengan nilai konstanta kontroler PI tersebut sudah sangat optimal, sistem tidak memiliki *overshoot* dan mencapai *steady state* dalam waktu kurang dari 1 detik. Nilai konstanta ini akan digunakan sebagai acuan ketika mengimplementasikan langsung pada rangkaian.



Gambar 3.19 Respons Sistem Kontrol Arus Menggunakan Simulasi

Seperti dengan kontrol tegangan, simulasi pada kontrol arus juga dilakukan untuk melihat respons sistemnya. Gambar 3.19 menunjukkan respons sistem ketika menggunakan kontroler PID. Respons sistem tersebut menggunakan nilai K_p, K_i, dan K_d masing-masing adalah 1, 30, dan 0.005. Nilai ini akan menjadi acuan ketika mengimplementasikan langsung pada rangkaian.

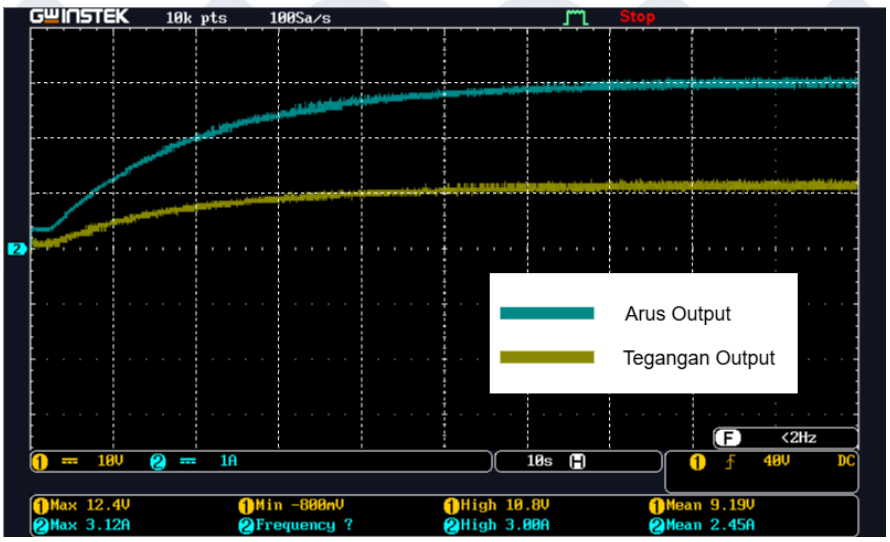
Ketika mengimplementasikan kepada rangkaian asli dengan algoritma kontrol PID menggunakan nilai K_p, K_i, dan K_d sesuai dengan hasil simulasi, respons sistem menjadi tidak stabil. Oleh karena itu diperlukan nilai yang baru dengan metode *fine tuning* berdasarkan respons sistem. Pada akhirnya, respons sistem yang sesuai hanya menggunakan nilai K_p dan K_i, sehingga kontroler yang digunakan adalah PI. Berikut merupakan hasil implementasi kontroler PI pada rangkaian dengan menggunakan STM32.

```
float pid(float error, float dt){
    integral += error * dt;
    float derivative = (error - previous) / dt;
    previous = error;
    float output = kp * error + ki * integral + kd * derivative;

    // Clamp output
    if (output > 1800) output = 1800;
    if (output < 200) output = 200;
    return output;
}
float dt = (now - last_time) / 1000.0f;  // Time step in seconds
last_time = now;
float error;
if (mode == CV_MODE) {
    // Voltage control
    error = setpoint_voltage - v_real;
    kp = 60;
    ki = 20;
    kd = 0;
} else {
    // Current control
    error = setpoint_current - a_real;
    kp = 40;
    ki = 25;
    kd = 0;
}
float pwm_output = pid(error, dt);
__HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pwm_output);
```

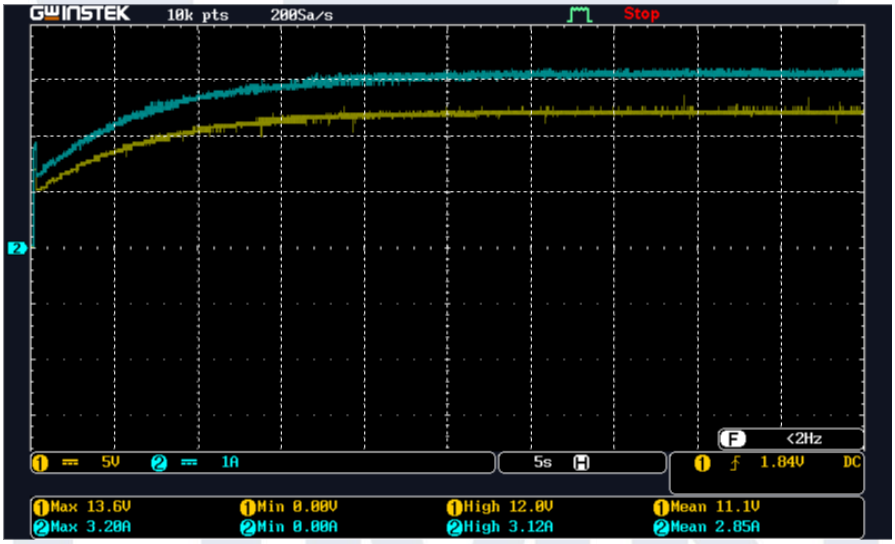
Gambar 3.20 Potongan Kode Kontroler PI pada STM32

Gambar 3.20 merupakan potongan kode pada STM32 yang digunakan sebagai algoritma kontroler PI. Nilai K_p dan K_i yang digunakan adalah 100 dan 30 pada mode CC, sedangkan nilai K_p dan K_i yang digunakan pada mode CV adalah 60 dan 20.



Gambar 3.21 Respons Sistem Mode CC dengan Kontroler PI pada Rangkaian *Synchronous Buck Converter*

Gambar 3.21 merupakan respons sistem pada mode CC dengan menggunakan kontroler PI pada beban 4 Ω. Respons sistem memiliki lonjakan tegangan dan arus ketika sistem dinyalakan. Sistem dinyalakan langsung dengan *power supply*, maka lonjakan pasti terjadi karena adanya perubahan tegangan secara tiba-tiba. Sistem mencapai arus *set point* sekitar 50 detik. Waktu tersebut cukup lama, sehingga perlu menambahkan nilai K_p dan K_i. Namun, ketika nilai tersebut ditambahkan sistem menjadi lebih agresif dan *overshoot* terjadi ketika dalam keadaan beban rendah. Hal ini merupakan sebuah *trade off* di mana sistem tidak dapat mencapai semua target yang diinginkan. *Overshoot* pada pengisian baterai harus dihindarkan karena dapat merusak baterai, sehingga nilai K_p dan K_i tidak diubah.



Gambar 3.22 Respons Sistem Mode CV dengan Kontroler PI pada *Synchronous Buck Converter*

Gambar 3.22 merupakan respons sistem ketika kontrol dalam mode CV dengan beban 4 Ω. Sama seperti mode CC, lonjakan tegangan dan arus terjadi ketika sistem dinyalakan. Sistem mencapai tegangan *set* point, 12 V dalam waktu 20 detik. Respons sistem ini sudah cukup cepat dan tidak perlu menambahkan parameter kontroler kembali.

3.2.2 Hambatan Implementasi

Pada awal implementasi, desain PCB masih memiliki beberapa kesalahan. Kesalahan yang terjadi adalah tidak menghubungkan pin GND pada modul DC-DC 24-12, tidak menghubungkan pin VCC dan GND pada ACS712, VDDB tidak terhubung dengan sumber tegangan 12 V. Selain itu pada subsistem kendali dan sensor, sensor tegangan yang digunakan pada awal implementasi adalah *voltage divider* saja. Jika hanya menggunakan *voltage divider*, terdapat drop tegangan yang signifikan akibat referensi *ground* dengan mikrokontroler sama. Selain itu, pembacaan ADC sensor sangat tidak stabil.

3.2.3 Solusi yang Diterapkan

Solusi pada permasalahan desain PCB adalah memodifikasi desain dan melakukan pencetakan ulang PCB. Selain itu, digunakan LM358AN sebagai sensor tegangan dan menggunakan filter median dalam pembacaan ADC. Hasil implementasi sudah menerapkan solusi-solusi tersebut, sehingga data implementasi sudah akurat.