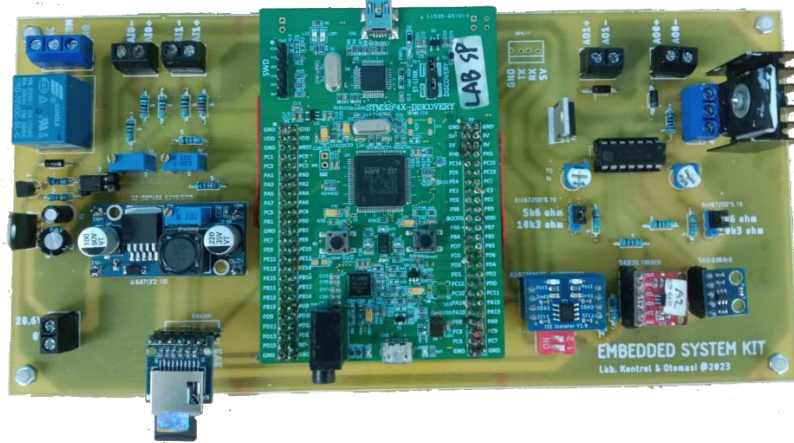


BUKU PETUNJUK

EMBEDDED SYSTEM KIT

Menggunakan Mikrokontroler STM32F407 Discovery



Disusun oleh Tim Tugas Akhir PCT100:

Muhammad Faris Zuhairi (07111940000164)

Hakhi Gya Yektianto (07111940000022)

Reno

Laboratorium Kontrol dan Otomasi
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya

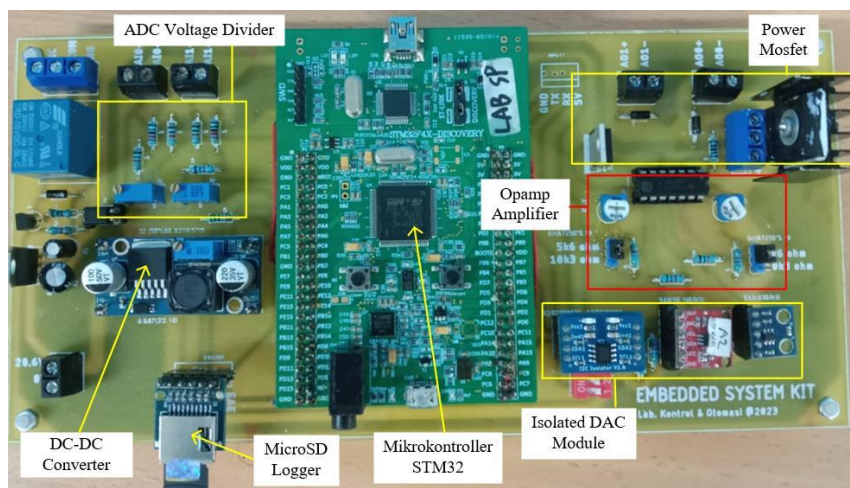
Agustus, 2023

Daftar Isi

Daftar Isi	2
BAB 1 Pengenalan Modul Embedded System	3
1.1 Studi Kebutuhan Sistem.....	3
1.2 Modul Kontroler	3
1.3 Modul Terminal	4
BAB 2 Desain Embedded System	5
2.1 Konfigurasi Pinout Mikrokontroler	5
2.2 Rangkaian Analog Input.....	5
2.3 Rangkaian Analog Output dan Relay.....	6
BAB 3 Persiapan Embedded System	8
3.1 Konfigurasi Power Supply	8
3.2 Kalibrasi Analog Input.....	8
3.3 Kalibrasi Analog Output	9
3.4 Wiring	11
BAB 4 Modifikasi Program Embedded System	13
4.1 Instalasi STM32CubeIDE	13
4.2 Repository Github.....	13
4.3 Import Project di STM32CubeIDE	14
4.4 Penjelasan isi dari program main.c	15
4.5 Penyimpanan Data melalui MicroSD	17
4.6 Program Kontroler	19
4.7 Interface STM32	22
BAB 5 Troubleshooting	24
5.1 Tegangan Pembacaan ADC Tidak Sesuai.....	24
5.2 Address DAC MCP4725 Tidak Terbaca	24
5.3 Kartu SD Tidak Terdeteksi.....	25
5.4 Data TXT pada MicroSD Tidak Muncul Setelah Running	25
5.5 Tegangan MOSFET IRFP250 tidak berubah saat Adjustment	25
5.6 MOSFET IRFP250 Overheat atau Berbau	25

BAB 1 Pengenalan Modul Embedded System

Pada modul embeded system ini modul kontroler dan modul terminal. Modul kontroler merupakan PCB utama yang berisi mikrokontroler untuk menanamkan program yang telah dibuat. Modul terminal ditambahkan agar memudahkan pengguna dalam memilih actuator dan sensor yang dikendalikan. Pada tugas akhir yang menggunakan prototipe ini, konektor dapat menghubungkan pin DB25 PCT-100 atau sinyal lain (EXT/eksternal). Gambar berikut menampilkan modul kontroler dan komponen-komponen di dalamnya.



Gambar 1.1 Modul Kontroler

1.1 Studi Kebutuhan Sistem

Sistem tertanam merujuk pada sistem komputer yang didesain untuk menjalankan tugas atau fungsi tertentu dalam satu unit tunggal. Desain sistem tertanam meliputi perancangan *hardware* dan program dalam *software* IDE. Desain ini dimulai dengan mendefinisikan kebutuhan dari fungsi alat dan rangkaian yang sesuai untuk menyelesaikan kebutuhan. Berikut merupakan kebutuhan dan jenis rangkaian yang sesuai.

Tabel 1.1 Kebutuhan Sistem dan Rangkaian yang Sesuai

No.	Kebutuhan	Bagian Blok
1.	Analog input (sensor level dan flow)	ADC (Analog to Digital Converter) dengan voltage divider
2.	Analog output (flow valve dan motor drain valve)	DAC (Digital to Analog Converter), operational amplifier, dan mosfet regulator (power electronics)
3.	Pemrosesan algoritma kontrol	Mikrokontroler
4.	Switching ON/OFF sinyal referensi (yang dijaga tetap saat mulai running)	Isolator dan relay
5.	Penyimpanan data	MicroSD data logger
6.	Monitoring melalui PC	UART TTL Communication

Berdasarkan

1.2 Modul Kontroler

Berdasarkan kebutuhan sistem yang telah disebutkan dipilih beberapa komponen Pada modul kontroler antara lain:

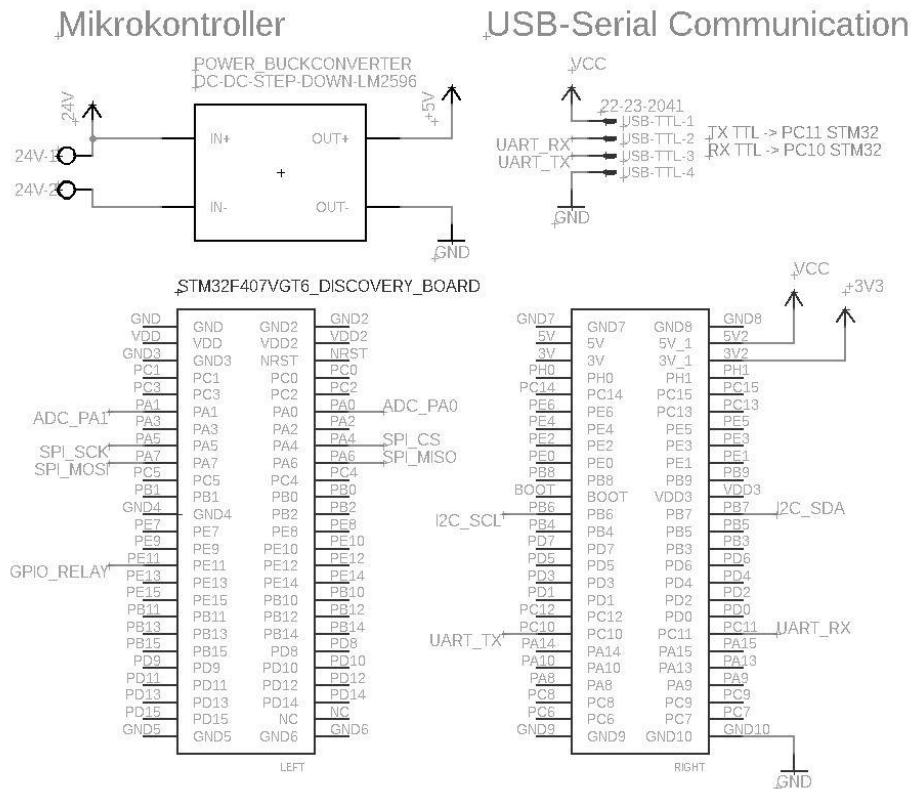
Tabel 1.2 Komponen Penyusun Modul Kontroler

No	Komponen	Fungsi
1	STM32F407VG	Sebagai mikrokontroler tempat program dijalankan

BAB 2 Desain Embedded System

2.1 Konfigurasi Pinout Mikrokontroler

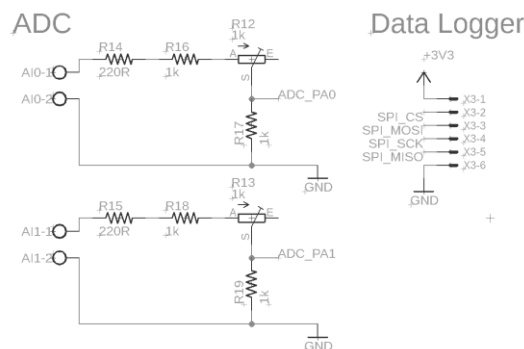
Mikrokontroler merupakan unit pemrosesan digital yang menjalankan komputasi digital sesuai algoritma fungsi yang diberikan. Input program dapat berasal dari tegangan pembacaan sensor melalui ADC atau informasi yang dikirimkan PC melalui komunikasi UART pada USB-to-TTL adapter. Output hasil program dikirimkan menuju aktuator melalui protokol I2C DAC sekaligus disimpan dalam file pada microSD melalui protocol SPI. Supply mikrokontroler dan keseluruhan modul diberikan oleh voltage regulator LM2596 yang dapat menyuplai tegangan 5 VDC hingga 3A.



Gambar 2.1 Rangkaian Modul Mikrokontroler

2.2 Rangkaian Analog Input

Signal conditioning berupa rangkaian pembagi tegangan diberikan agar tegangan output sensor (0-10VDC) dapat dibaca ADC pada range kerjanya (0-3.3VDC). Pembagi tegangan menggunakan rangkaian seri resistor dan adjustable trimmer.



Gambar 2.2 Rangkaian Penurun Tegangan ADC dan MicroSD Logger

$$\frac{V_{out}}{V_{in}} = \frac{R_{17}}{R_{17} + R_{14} + R_{16} + R_{12}}$$

Agar tegangan keluaran $V_{out} = 3.3V$, rasio V_{out}/V_{in} ditentukan bernilai $\frac{V_{out}}{V_{in}} = \frac{3.3V}{10V} \approx \frac{1}{3}$. Hambatan resistor ditetapkan $R_{16} = R_{17} = 1k\Omega$, $R_{14} = 220\Omega$. Resistansi trimmer diperoleh dengan substitusi nilai tersebut ke persamaan pembagi tegangan.

$$\frac{1}{3} = \frac{1k}{1k + 1k + 220 + R_{12}}$$

Aktuator yang digunakan memiliki range tegangan berbeda. Aktuator mendapat sinyal kontrol 12bit dari mikrokontroler, kemudian oleh IC DAC MCP4725 diubah menjadi tegangan 0-5V. Sinyal I2C perlu diisolasi agar MCP4725 tidak menyebabkan *drop voltage* pada mikrokontroler. Keluaran DAC ini perlu dikuatkan oleh rangkaian non-inverting opamp sehingga menghasilkan 0-10V atau 0-24V. Kedua nilai ini dapat disesuaikan dengan mengatur potentiometer adjustment yang akan dibahas pada bab selanjutnya. Agar dihasilkan sinyal yang linear dan stabil, dibutuhkan regulator power mosfet yang difeedback melalui voltage follower opamp. Jangkauan tegangan kontrol dari aktuator disajikan pada tabel berikut.

No.	Aktuator	Range Input
1.	Flow control valve	0-24VDC (max current 1A)
2.	Motor drain valve	0-10VDC
3.	Pompa	0-10VDC

DAC	Range tegangan pada PCT100	Address I2C
GY4725 (biru)	0-24V	0x61
MCP4725 (merah)	0-10V	0x60

Penguatan tegangan untuk motor flow control valve dan drain valve menggunakan rangkaian noninverting amplifier. Untuk motor flow control valve, penguatan menggunakan resistor feedback TM_1 dan resistor inverting R_3 . Nilai keduanya dihitung berdasarkan rumus berikut.

$$\frac{V_{out \text{ flow valve}}}{V_{in \text{ DAC flow valve}}} = 1 + \frac{TM_1}{R_3}$$

Diketahui $V_{out \text{ flow valve max}} = 24V$, $V_{in \text{ DAC flow valve max}} = 5V$, dan $R_3 = 1k\Omega$.

$$\begin{aligned}\frac{24}{5} &= 1 + \frac{TM_1}{1k} \\ \frac{19}{5} &= \frac{TM_1}{1k} \\ TM_1 &= 3.8k\end{aligned}$$

Untuk motor drain valve, penguatan menggunakan resistor feedback TM_2 dan resistor inverting R_6 . Nilai keduanya dihitung berdasarkan rumus berikut.

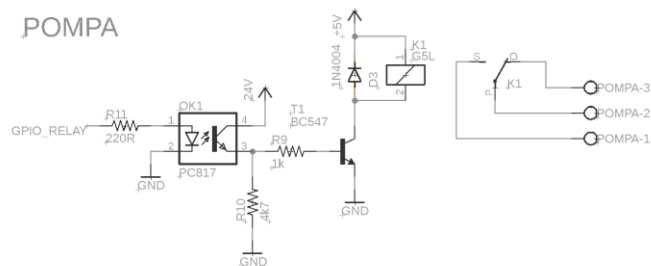
$$\frac{V_{out \text{ drain valve}}}{V_{in \text{ DAC drain valve}}} = 1 + \frac{TM_2}{R_6}$$

Substitusi parameter kondisi tegangan maksimum untuk mendapat resistansi trimmer TM_2 .

$V_{out \text{ drain valve}} = 10V$, $V_{in \text{ DAC drain valve}} = 5V$, $R_6 = 1k\Omega$

$$\begin{aligned}\frac{10}{5} &= 1 + \frac{TM_2}{1k} \\ 1 &= \frac{TM_2}{1k} \\ TM_2 &= 1k\Omega\end{aligned}$$

Aktuator pompa digerakkan oleh supply tegangan eksternal 0-10V. Tegangan eksternal ini oleh mikrokontroler hanya memiliki 2 kondisi, yakni OFF (0V) dan ON (tegangan eksternal). Relay diberikan sebagai *switch* yang digerakkan oleh pin digital GPIO mikrokontroler.



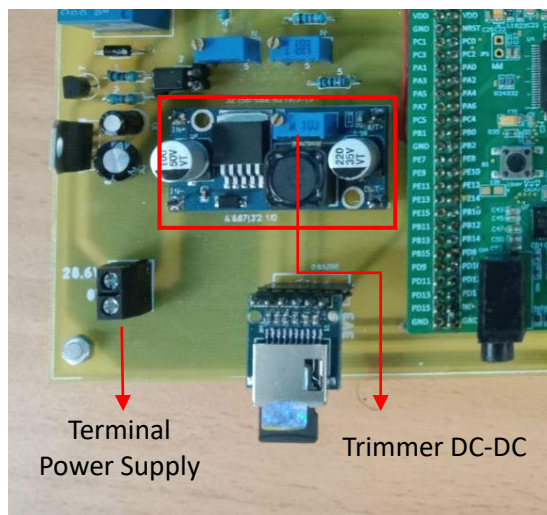
Gambar 3.7. Rangkaian switching relay pompa

BAB 3 Persiapan Embedded System

3.1 Konfigurasi Power Supply

Sebelum menjalankan modul diperlukan pengecekan dan konfigurasi pada power supply agar tegangan yang masuk kedalam rangkaian sesuai sehingga tidak beresiko merusak alat. Dalam melakukan konfigurasi power supply diperlukan multimeter untuk membaca tegangan dan power supply external untuk memberikan supply kedalam rangkaian. Adapun untuk langkah kalibrasi analog input sebagai berikut.

1. Lepas STM32 dari modul embedded system
2. Nyalakan modul embedded system dengan cara menghubungkan power supply external pada terminal blok “PWR” pada modul terminal.
3. Atur power supply external untuk memberikan tegangan 28.6 V.
4. Ukur tegangan keluaran DC-DC converter pada pin “OUT +” DAN “OUT -” menggunakan multimeter.
5. Lakukan adjustment tegangan dengan cara memutar trimmer pada DC-DC converter sampai tegangan output pada DC-DC converter menjadi 5 volt.
6. Apabila kalibrasi telah selesai matikan power supply
7. Lepas kabel power supply dari modul terminal dan kembalikan seperti semula.



Gambar 3.1 Lokasi Kalibrasi Power Supply

3.2 Kalibrasi Analog Input

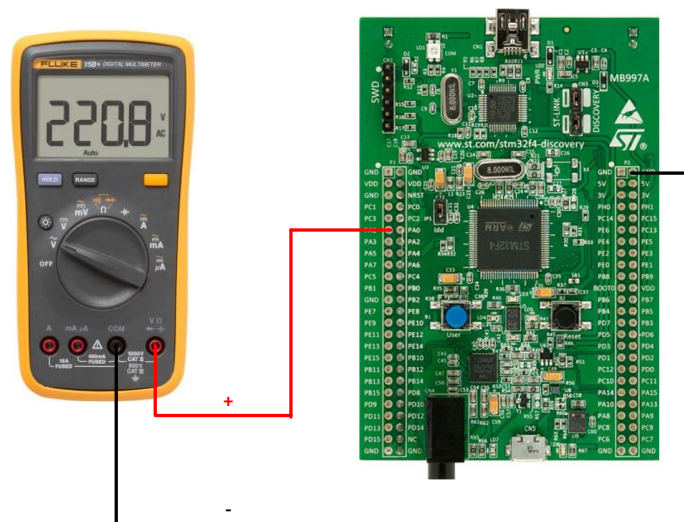
Sebelum menjalankan modul diperlukan pengecekan dan kalibrasi pada analog input agar sinyal yang dikirim oleh PCT dapat diterima oleh kontroler dengan benar. Sinyal yang dikirim oleh PCT berupa tegangan DC 0-10 volt harus dirubah menjadi tegangan 0-3.3 volt oleh rangkaian ADC voltage divider agar dapat terbaca oleh kontroler. Dalam melakukan kalibrasi analog input diperlukan multimeter untuk membaca tegangan dan power supply external untuk memberikan supply kedalam rangkaian. Adapun untuk langkah kalibrasi analog input sebagai berikut.

1. Pasang Mikrokontroler STM 32 pada modul kontroler.
2. Ubah selector AI0 pada modul terminal menjadi mode “EXT”.
3. Hubungkan power supply external pada terminal AI0.
4. Nyalakan power supply dan beri tegangan 10 V.
5. Pastikan mikrokontroler STM 32 dalam keadaan mati dengan cara menatur dipswitch ke posisi “DEBUG MODE”
6. Ukur tegangan pada keluaran rangkaian voltage divider dengan bantuan kabel jumper male-female, untuk jalur AI0 pengukuran pada pinout PA0 dengan ground STM32 dan untuk jalur AI1 pengukuran pada pinout PA1 dengan ground STM32 untuk AI1 seperti pada gambar 3.3.

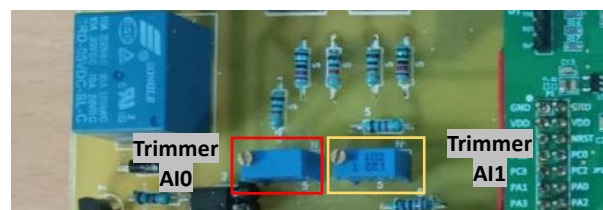
7. Lakukan adjustment tegangan dengan cara memutar trimmer sampai tegangan pada jalur pengukuran AI0 keluaran rangkaian voltage divider menjadi 3.3 volt.
8. Apabila tanganan terukur sudah sesuai lakukan kalibrasi pada terminal AI1 dengan mengulang langkah 2-7. Matikan power supply 10V ketika melepas terminal screw AI.
9. Apabila kedua terminal sudah dikalibrasi matikan power supllly
10. Lepas kabel power supply dari terminal AI
11. Kembalikan posisi selector pada modul terminal menjadi mode “PCT.FT dan PCT.LT”.



Gambar 3.2 Lokasi Selector AI pada Modul Terminal



Gambar 3.3 Ilustrasi Pengukuran Tegangan pada Kalibrasi AI



Gambar 3.4 Lokasi Kalibrasi Voltage Divider AI

3.3 Kalibrasi Analog Output

Sebelum menjalankan modul diperlukan pengecekan dan kalibrasi pada analog output agar sinyal yang dikirim oleh kontroler dapat diterima oleh PCT dengan benar. Sinyal kontrol yang dikirim oleh STM32 berupa sinyal bit 0-4095 dirubah terlebih dahulu menjadi tegangan 0-5 volt oleh MCP4725 pada terminal AO1 serta oleh GY4725 pada terminal AO0. Sinyal kontrol yang telah diubah menjadi analog perlu dikuatkan agar sesuai dengan spesifikasi alat (PCT-100) agar dapat menggerakkan aktuator. Pada kasus PCT-100 terdapat dua aktuator yaitu flow control valve yang membutuhkan sinyal kontrol 0-24 V dan drain valve yang membutuhkan sinyal kontrol 0-10 V. Dalam melakukan kalibrasi analog output diperlukan multimeter untuk membaca tegangan dan kabel jumper untuk menghubungkan sinyal kontrol dengan tegangan 5 V DC-DC converter.

Pada prototipe ini AO0 digunakan untuk mengontrol control valve dan AO1 untuk pump. Untuk menggerakkan control valve, diperlukan sinyal kontrol 0-22V serta arus hingga 1.2 A. sedangkan

pump membutuhkan tegangan 0-10V. kalibrasi keduanya dilakukan secara terpisah seperti ditunjukkan oleh prosedur di bawah ini.

Kalibrasi AO0 / FV (Control Valve)

1. Lepas STM32 dari modul embedded system
2. Lepas DAC (GY4725) dari modul kontroler.
3. Hubungkan power supply external ke terminal blok PWR pada modul terminal dan beri tegangan 28.6 V.
4. Ubah posisi selector AO0 pada modul terminal menjadi mode “EXT”.
5. Jumper pin output 5v dc-dc converter ke pin header ‘OUT’ dari GY4725 (AO0) menggunakan kabel jumper male-female.

Pin “OUT +” dari DC-DC Converter terhubung dengan pin “OUT” dari header GY4725. Sedangkan, pin “OUT -” dari DC-DC Converter terhubung dengan pin “GND” dari header GY4725.

6. Pada modul kontroler, posisikan selektor pada posisi label 5k6.
7. Hubungkan control valve ke terminal block AO0 pada modul terminal. Pemasangan kabel control valve ditunjukkan seperti pada gambar xx.
8. Ukur tegangan pada terminal output AO0.

Tips:

Pada saat menghubungkan dengan beban, terkadang tegangan pada power supply turun (terlihat dari display PSU atau tegangan PWR) disertai bunyi kontaktor PSU. Hal ini disebabkan oleh fitur constant current pada power supply. Ketika arus yang menuju beban berlebih dari batas atas constant current, tegangan PSU tidak dapat naik. Arus konstan ini berakibat valve tidak dapat membuka sepenuhnya.

Untuk mengatasi hal ini, naikan batas atas constant current dengan memutar knob “CURRENT” power supply hingga tegangan output PSU kembali ke 28.6V. Arus yang disuplai berbanding lurus dengan bukaan valve.

9. Lakukan adjustment tegangan dengan cara memutar trimmer AO0 sampai tegangan pada keluaran terminal block AO0 menjadi 22 volt, yaitu tegangan maksimal output yang diinginkan (di kasus ini yakni tegangan kontrol maksimal flow control valve PCT-100).
10. Matikan power supply apabila tegangan terukur sudah sesuai.
11. Lepas kabel jumper dan kembalikan GY4725 pada kondisi semula.

Kalibrasi AO1 / Pump

1. Lepas STM32 dari modul embedded system
2. Lepas DAC (MCP4725) dari modul kontroler.
3. Hubungkan power supply external ke terminal blok PWR pada modul terminal dan beri tegangan 28.6 V.
4. Ubah posisi selector AO1 pada modul terminal menjadi mode EXT.
5. Jumper pin output 5v dc-dc converter ke pin header ‘OUT’ dari MCP4725 (AO1) menggunakan kabel jumper male-female.

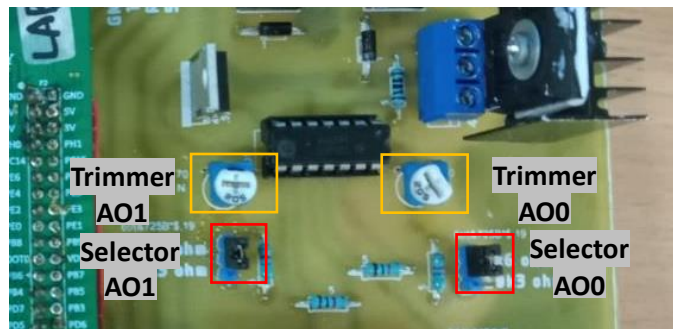
Pin “OUT +” dari DC-DC Converter terhubung dengan pin “OUT” dari header MCP4725. Sedangkan, pin “OUT -” dari DC-DC Converter terhubung dengan pin “GND” dari header MCP4725.

6. Pada modul kontroler, posisikan selektor AO1 pada posisi label 5k6.
7. Ukur tegangan pada terminal output AO1.

8. Lakukan adjustment tegangan dengan cara memutar trimmer AO1 sampai tegangan pada keluaran terminal block AO1 menjadi 10 volt, yaitu tegangan maksimal output yang diinginkan (di kasus ini yakni tegangan kontrol maksimal pump PCT-100).
9. Matikan power supply apabila tegangan terukur sudah sesuai.
10. Lepas kabel jumper dan kembalikan MCP4725 pada kondisi semula.
11. Kembalikan selector AO1 pada modul terminal ke mode “PCT.PUMP”
12. Pasang mikrokontroler untuk mulai memprogram.



Gambar 3.5 Lokasi Selector AO pada Modul Terminal

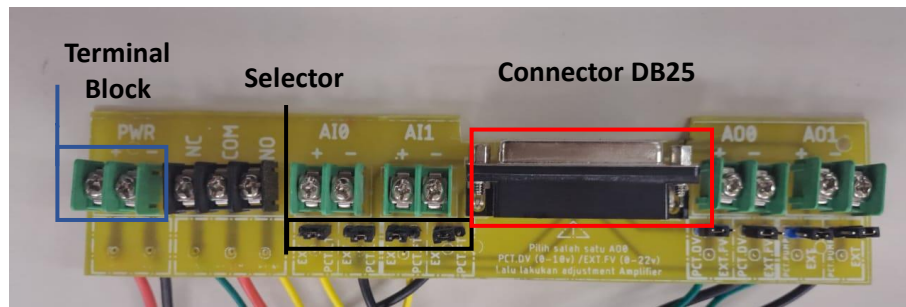


Gambar 3.6 Lokasi Kalibrasi AO

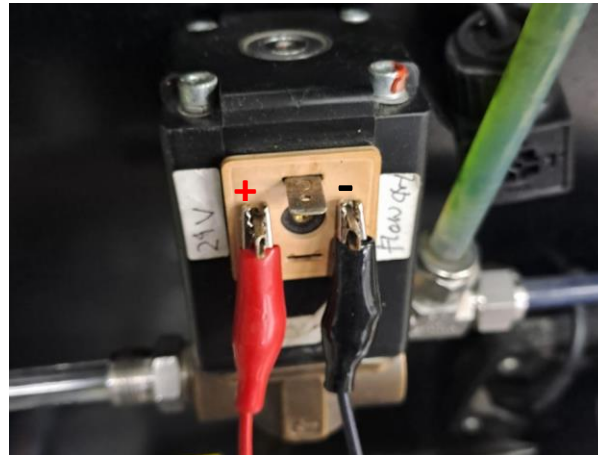
3.4 Wiring

Setelah semua sistem telah dikalibrasi dan dikonfigurasi dilakukan pengkabelan atau wiring untuk menghubungkan PCT ke modul terminal dan modul terminal ke modul kontroler. Untuk melakukan wiring dibutuhkan obeng untuk membuka dan mengencangkan kembali terminal blok. Adapun langkah langkah yang dibutuhkan sebagai berikut.

1. Hubungkan kabel yang berada dibalik modul konektor ke terminal blok modul kontroler sesuai dengan label yang berada pada masing-masing modul.
2. Pasangkan Kabel power dari power supply external ke terminal “PWR” (pastikan kutub positif dan negatif tidak tertukar).
3. Hubungkan konektor male DB25 dari PCT ke konektor female DB25 yang berada Di modul konektor.
4. Kencangkan baut yang berada pada konektor DB25 agar tidak mudah terlepas.
5. Hubungkan control valve ke terminal blok AO0 menggunakan kabel buaya (pastikan kutub positif dan negatif tidak tertukar).
6. Atur posisi Selector AI pada posisi PCT.FT atau PCT.LT (sebelah kanan)
7. Atur posisi Selector AO0 ke posisi ext FV (sebelah kiri)
8. Nyalakan power supply external dan beri tegangan sebesar 28.6 V.



Gambar 3.7 Penjelasan Modul Konektor



Gambar 3.8 Pengkabelan Control Valve

BAB 4 Modifikasi Program Embedded System

4.1 Instalasi STM32CubeIDE

Untuk dapat memprogram STM32 diperlukan beberapa software dan alat untuk memulainya antara lain:

- STM32CubeIDE, IDE gratis untuk mengembangkan perangkat lunak STM32
- STM32F407VG sebagai mikrokontroler pada PCT

Untuk dapat melakukan instalasi STM32cube IDE dapat mengikuti langkah berikut:

1. Membuat akun pada ST.com pada link berikut [register](#).
2. Download software STM32CubeIDE pada link berikut [STM32CubeIDE download](#)
3. Anda akan diminta login sebelum bisa mendownload software, gunakan akun yang telah dibuat sebelumnya.
4. Ekstrak file yang telah didownload.
5. Jalankan file installer yang berada dalam file yang telah diunzip.
6. Atur file direktori instalasi software STM32CubeIDE.
7. Biarkan choose component dalam keadaan default.
8. STM32CubeIDE telah berhasil diinstall.

4.2 Repository Github

Langkah pertama yang perlu dilakukan adalah membuka situs [Download GitHub directory \(download-directory.github.io\)](#) kemudian masukkan link github, yakni <https://github.com/fariszuh/TA-PCT100>. Setelah itu, tekan enter dan proses download akan dimulai secara otomatis (lihat gambar 3.1).



Gambar 4.1 Halaman Download Direktori Github

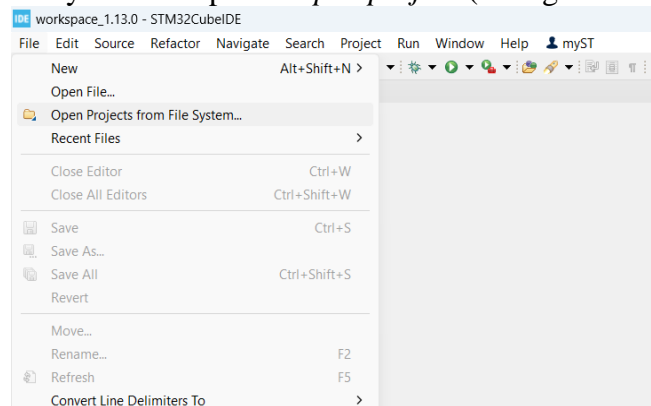
Setelah ter-download, file zip kemudian di-extract dan akan tampil file serta folder program **STM32** dengan project bernama **MultichannelADC** seperti gambar 3.2. File **main.c** yang berupa program utama berada di folder **Core>Src**.

fariszu TA-PCT100 master STM32-Multic...	7/29/2023 12:49 PM
MultichannelADC Debug.launch	7/29/2023 5:49 AM
MultichannelADC.ioc	7/29/2023 5:49 AM
STM32F407VGTX_FLASH.Id	7/29/2023 5:49 AM
STM32F407VGTX_RAM.Id	7/29/2023 5:49 AM
.cproject	7/29/2023 5:49 AM
.mxproject	7/29/2023 5:49 AM
.project	7/29/2023 5:49 AM
Drivers	7/29/2023 5:49 AM
FATFS	7/29/2023 5:49 AM
Middlewares	7/29/2023 5:49 AM
.settings	7/29/2023 5:49 AM
Core	7/29/2023 5:49 AM
Debug	7/29/2023 5:49 AM

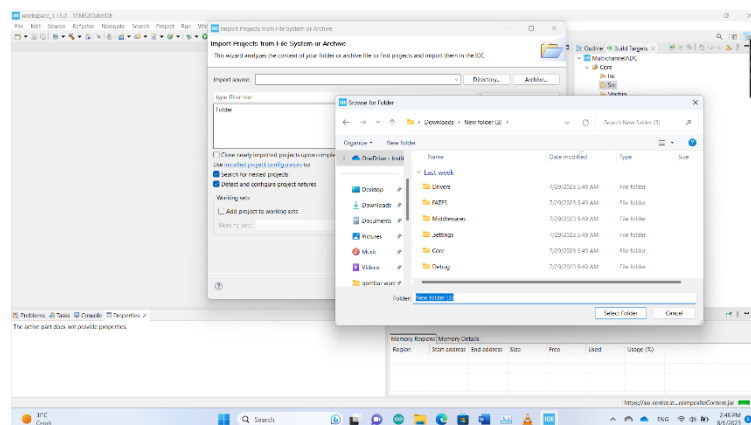
Gambar 4.2 Ekstraksi Folder Program

4.3 Import Project di STM32CubeIDE

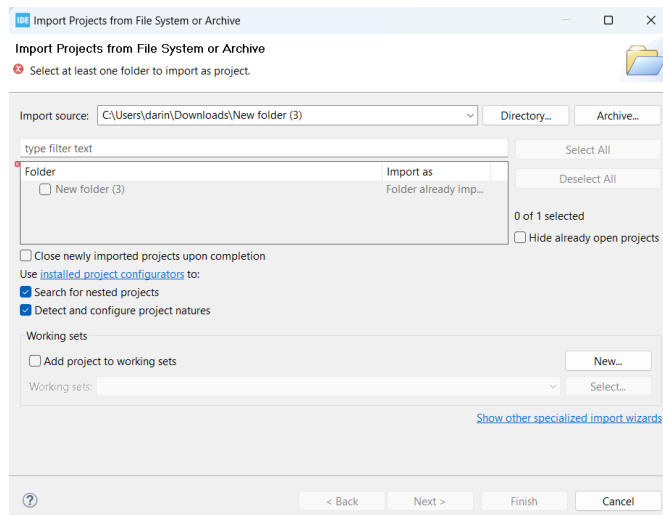
Langkah selanjutnya, yakni membuka folder berisi program yang telah diekstrak dengan cara memilih file>open project from file system. Kemudian pilih **Directory** dan sesuaikan dengan folder hasil ekstraksi yang telah dilakukan. Apabila sudah maka pilih next dan kemudian finish untuk menyelesaikan proses *open project* (lihat gambar 3.3, 3.4, dan 3.5).



Gambar 4.3 Membuka File Program

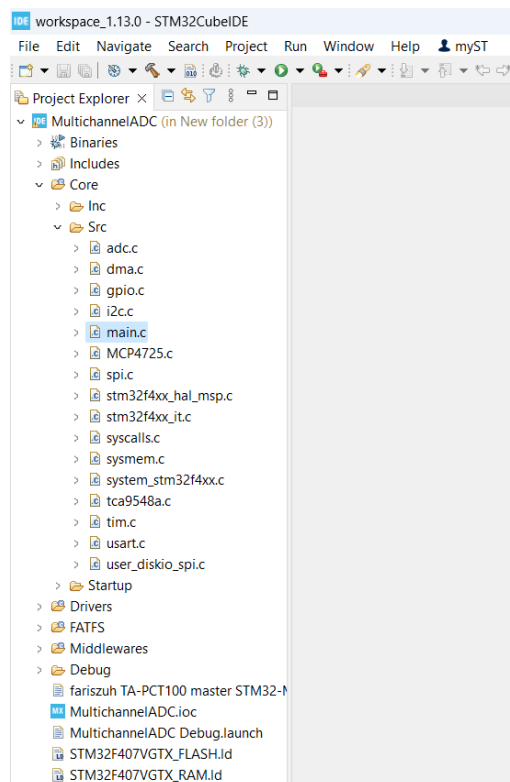


Gambar 4.4 Impor File Program



Gambar 4.5 Impor File Program Lanjutan

Setelah berhasil *load project* atau folder maka tampilan stm 32 akan seperti berikut yang artinya semua program yang telah di-*download* dari github telah berhasil di *load* (lihat gambar 3.6).



Gambar 4.6 Tampilan *Project Explorer* dari Program yang Diimpor

4.4 Penjelasan isi dari program **main.c**

Pertama, buka file program **main.c** dan didalamnya berisikan beberapa *library* yang perlu di-*import* sebagai berikut (lihat gambar 3.7). Pada baris 20 s.d 28 adalah *library* untuk komponen *hardware*, untuk “*adc.h*” berguna untuk device *analog digital converter* pada *hardware*. *Library* “*fatfs.h*” untuk keperluan *micro sd* dalam penyimpanan data.

Kemudian untuk “*i2c.h*” digunakan untuk keperluan DAC atau *digital analog converter*. Selain itu, “*spi.h*” atau Serial Peripheral Interface (SPI) juga digunakan untuk keperluan komunikasi pada *micro sd*. Selanjutnya terdapat “*usart.h*” untuk komunikasi dengan usb ttl dan “*gpio.h*” untuk aktivasi komponen *relay*. Pada baris 32 s.d. 37 mendukung tipe data yang

digunakan pada program. Selain itu, terdapat library “string.h” untuk keperluan string pada IO. Selanjutnya untuk “MCP4725” adalah library untuk I2C MCP4725 yang berupa komponen DAC.

```

20 #include "main.h"
21 #include "adc.h"
22 #include "dma.h"
23 #include "fatfs.h"
24 #include "i2c.h"
25 #include "spi.h"
26 #include "tim.h"
27 #include "usart.h"
28 #include "gpio.h"
29
30 /* Private includes -----
31  */
32 #include "stm32f4xx_hal.h"
33 #include <stdio.h>
34 #include <string.h>
35 #include <stdarg.h> //for va_list var arg functions
36 #include <stdlib.h>
37 #include <MCP4725.h>
38

```

Gambar 4.7 Library yang Diperlukan

Langkah selanjutnya, yaitu melakukan inisialisasi data yang harus dilakukan didalam *section* `/* USER CODE BEGIN PV */`. Pada contoh program ini, digunakan *controller neural network* (NN) maka diinisialisasi matriks untuk jumlah neuron pada *hidden layer* dan neuron pada input, yakni 3. Selain itu, diinisialisasi tiap variabel beserta tipe datanya, sebagai contoh `float w_input[num_neuronHidden][num_neuronInput];` yang berupa variabel weight input dengan tipe data *float* dalam bentuk array. Begitu pula `v_hidden` yang merupakan weight output. Inisialisasi dapat disesuaikan dengan kebutuhan program *controller* masing-masing.

Delta pada baris 68 adalah *local* gradien pada output, sedangkan `x_input` adalah inputnya yang dalam bentuk *array* dengan jumlah 3. Pada baris 71 s.d.74 didefinisikan variabel `y` dan `Qin` serta parameter *setpoint* `SP` baik dalam unit milimeter dan volt.

Terdapat pula `uint32_t` pada baris 76 yang artinya tipe data unsigned integer 32 bit dan variabel `my_adc[3]` yang berfungsi sebagai *storing* ADC. Pada baris selanjutnya, didefinisikan 3 analog input (AI) *channel*, tetapi hanya 2 *channel* yang dibutuhkan, sebagaimana dua analog *output* (AO) `AO0_volt` dan `AO1_volt`.

```

61 /* USER CODE BEGIN PV */
62 // Initialize matrix for weight input and hidden layer
63 #define num_neuronHidden 3 // sebagai baris
64 #define num_neuronInput 3 // sebagai kolom
65 // baris: neuronHidden ke-, kolom: neuronInput ke-
66 float w_input[num_neuronHidden][num_neuronInput];
67 float v_hidden[num_neuronHidden];
68 float delta2[num_neuronHidden], hj[num_neuronHidden];
69 float x_input[num_neuronInput];
70
71 float y, Qin; // dalam volt
72 float SP = 100; // dalam mm
73 float SP_volt = 0;
74 uint8_t dey_deu;
75
76 uint32_t my_adc[3];
77 uint32_t AI0_bit, AI1_bit, AI2_bit;
78 float AO0_volt, AO1_volt, AO2_volt;
79 float AO0_volt = 0; // valve flow control Qin
80 float AO1_volt = 4; // pompa, drain valve
81 uint16_t AO0_bit, AO1_bit = 0;
82

```

Gambar 4.8 Inisialisasi Tipe Data Variabel dan Nilainya

Pada baris 83 s.d. 86 terdapat buffer Tx dan Rx untuk keperluan komunikasi serial ke PC. Kemudian perlu digaris bawahi, penambahan baris inisialisasi harus berada diantara *section*

`/* USER CODE BEGIN PV */` dan `/* USER CODE END PV */` pada baris 102 agar tidak hilang saat dilakukan *build*.

```

83 char buffer[100];
84 char Tx_buffer[100], Rx_buffer[100];
85 char Rx_buffer_string[];
86 char Rx_buffer_status[];
87
88 //char data1, data2, data3;
89 //uint32_t prevSP, SP, KP, KI = 0;
90
91 //float KI_nopointer;
92 //int startTime, diffTime;
93
94 // Initialize learning rate, alpha, delta1, and control signal u
95 float lr = 0.00001;
96 float alpha = 0.1;
97 float delta1 = 0;
98 float u = 0;
99 // Initialize error
100 float e_0, e_1, e_2 = 0;
101
102 /* USER CODE END PV */

```

Gambar 4.9 Inisialisasi Variabel dan Parameter yang Dibutuhkan

4.5 Penyimpanan Data melalui MicroSD

Hal yang perlu dilakukan sebelum beranjak kedalam topik *read and write* adalah mounting drive, yakni menghubungkan micro sd pada sistem stm32 kepada modul dari microsd. Proses mounting dilakukan pada baris 201 (lihat gambar 3.10) oleh fungsi **f_mount** dengan objeknya, yakni **Fatfs** yang telah diinisialisasi di awal. Kondisi *if* pada baris 202 akan mengeluarkan *output* jika hasil mounting gagal atau terjadi *error*. Apabila proses *mounting* sukses maka kondisi *else* akan mengeluarkan *output*-nya.

```

200 //Mount drive
201 fres = f_mount(&FatFs, "", 1); //1=mount now
202 if (fres != FR_OK) {
203     myprintf("f_mount error (%i)\r\n", fres);
204 // while(1);
205 }
206 else{
207     myprintf("SD CARD mounted successfully...\r\n");
208 }

```

Gambar 4.10 Mounting Drive atau Penyimpanan Micro SD

Selanjutnya, juga perlu dipastikan apakah proses membaca dan menulis dapat dilakukan. Langkah pertama, yakni melakukan cek *free space* pada micro sd. Dengan fungsi **f_getFree**, apabila **fres** dibaca ok (**FR_OK**) maka akan mengeluarkan *output* pada komunikasi *serial* berupa komentar seperti pada baris 221. Sebaliknya, jika terjadi *error* maka akan tampil komentar pada baris 216 gambar 3.11.

```

210 /* Check free space */
211 DWORD free_clusters, free_sectors, total_sectors;
212 FATFS* getFreeFs;
213 fres = f_getfree("", &free_clusters, &getFreeFs);
214
215 if (fres != FR_OK) {
216     myprintf("f_getfree error (%i)\r\n", fres);
217     while(1);
218 } else{
219     total_sectors = (getFreeFs->n_fatent - 2) * getFreeFs->csize;
220     free_sectors = free_clusters * getFreeFs->csize;
221     myprintf("SD card stats:\r\n%10lu KiB total drive space.\r\n%10lu KiB available.\r\n", total_sectors / 2,
222 }

```

Gambar 4.11 Pengecekan Ruang Kosong di Micro SD

Kemudian dilanjutkan dengan proses pembacaan file .txt seperti terlihat pada baris 225 s.d 231. Dicontohkan di sini “textbaca.txt” akan dibaca kemudian dibuka melalui **f_open** (lihat gambar 3.12).

```

224  /*-----READ TEST.TXT by f_open method-----*/
225  fres = f_open(&fil, "testBaca.txt", FA_READ); //Try to open file
226  if (fres != FR_OK) {
227      myprintf("f_open error (%i)\r\n");
228      while(1);
229  } else{
230      myprintf("I was able to open 'testBaca.txt' for reading!\r\n");
231  }

```

Gambar 4.12 Pengecekan Terbacanya File (testBaca.txt) yang Ada Didalam *Micro SD*

Setelah file .txt berhasil dibaca atau diteteki maka selanjutnya adalah membaca isi dari file atau string .txt tersebut. Seperti pada gambar dibawah, pada baris 223 hingga 244, pertama-tama fungsi **f_gets** akan melakukan pembacaan dari readBuf (baca baris 237 gambar 3.13). Untuk **readBuf**, adalah suatu *buffer* untuk menyimpan file-file yang kemudian akan dipecah kembali. Seperti sebelumnya, apabila terjadi error maka akan mengeluarkan *output* komentar error. Jika berhasil maka outputnya akan membaca string dari file .txt (textbaca.txt).

```

233  /*-----READ TEST.TXT by f_gets method-----*/
234  BYTE readBuf[30];
235  //We can either use f_read OR f_gets to get data out of files
236  //f_gets is a wrapper on f_read that does some string formatting for us
237  TCHAR* rres = f_gets((TCHAR*)readBuf, 30, &fil);
238  if(rres != 0) {
239      myprintf("Read string from 'testBaca.txt' contents: %s\r\n", readBuf);
240  } else {
241      myprintf("f_gets error (%i)\r\n", fres);
242      while(1);
243  }
244  f_close(&fil); //Close file, don't forget this!

```

Gambar 4.13 Pembacaan Isi Dari File testBaca.txt

Setelah proses pembacaan file selesai kemudian dilanjutkan dengan proses menulis (*write*). Dalam contoh ini file untuk dilakukan write didalamnya bernama "write.txt". Nantinya, seluruh data atau kode akan ditulis kedalam file yang didefinisikan berdasarkan serangkaian kode pada gambar 3.14 dibawah.

```

246  /*-----OPEN WRITE.TXT READY TO WRITE-----*/
247  fres = f_open(&fil, "write.txt", FA_WRITE | FA_OPEN_ALWAYS | FA_CREATE_ALWAYS);
248  if(fres == FR_OK) {
249      myprintf("I was able to open 'write.txt' for writing\r\n");
250  } else {
251      myprintf("f_open error (%i)\r\n", fres);
252      while(1);
253  }

```

Gambar 4.14 Kode Penulisan/Write ke Dalam Sebuah File (write.txt)

Langkah berikutnya adalah memberikan nilai inisial *set value* pada DAC, baik untuuk AO0 maupun AO1, yakni awalnya bernilai nol (dalam miliVolt). Hal yang sama berlaku pada *relay* yang terlihat pada baris 259 pada gambar di bawah. Saat ini digunakan gpio untuk mengontrol *relay* sehingga tidak menggunakan pwm. Dengan fungsi **HAL_GPIO_WritePin**, inisial gpio adalah bernilai nol atau keadaan awal mati melalui **GPIO_PIN_RESET** (lihat gambar 3.15).

```

255  setValue_AO0(0); // 0mV di awal
256  setValue_AO1(0); // 0mV di awal
257  prevTime = 0;
258
259  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_11, GPIO_PIN_RESET);
260

```

Gambar 4.15 Pemberian Nilai Inisial pada DAC dan GPIO

4.6 Program Kontroler

Pada baris 263 dan 264 (lihat gambar 3.16) memiliki arti, yakni apabila SP (*setpoint*) kurang dari 42.177 (merupakan batas bawah) maka SP tersebut sama dengan batas bawah tersebut. Hal yang sama berlaku apabila saat SP ditentukan lebih dari batas atas. Alasannya, yakni titik mulai operasi level *measurement* adalah batas bawah dan titik sesaat sebelum terjadinya *overflow* pada tangki adalah batas atas tersebut.

```
261 // NN Initialization
262
263 if(SP < 42.1779848866499){
264     SP = 42.1779848866499;
265 }
266 else if(SP > 217.324332493703){
267     SP = 217.324332493703;
268 }
269
```

Gambar 4.16 Setting Batas Atas dan Bawah Setpoint

Selanjutnya pada baris 272 hingga 287 (lihat gambar 3.17) bertujuan untuk menggenerasi matriks berisikan nilai random. Dalam program ini, pada baris 273 dilakukan generasi matriks *wight* secara random untuk hidden layer-output. Sedangkan pada baris 282 dilakukan generasi matriks ukuran 3x3 untuk weight inputnya.

```
270
271 // Generate random 0-1 for initial weight and
272 for(int neuronHidden=0; neuronHidden<num_neuronHidden; ++neuronHidden){
273     v_hidden[neuronHidden] = (float)rand()/RAND_MAX;
274 // printf("v_hidden[%d] = %f \n",neuronHidden,v_hidden[neuronHidden]);
275     delta2[neuronHidden] = 0;
276     hj[neuronHidden] = 0;
277
278     for(int neuronInput=0; neuronInput<num_neuronInput; ++neuronInput){
279         if(neuronHidden == 0){
280             x_input[neuronInput] = 0;
281         }
282         w_input[neuronHidden][neuronInput] = (float)rand()/RAND_MAX;
283 // printf("w [%d][%d] = %f \n",neuronHidden,neuronInput,w_input[neuronHidden]
284     }
285 // printf("delta2[%d] = %f \n",neuronHidden,delta2[neuronHidden]);
286 // printf("hj[%d] = %f \n",neuronHidden,hj[neuronHidden]);
287 }
288
```

Gambar 4.17 Generasi Random Matriks untuk Neural Network

Sebagai tambahan, **while (1)** berperan sebagai iterasi dari *microkontroler*, yakni *void loop* pada arduino sehingga while disini untuk mengulangi eksekusi kode secara terus-menerus.

```
291 /* Infinite loop */
292 /* USER CODE BEGIN WHILE */
293 while (1)
294
```

Gambar 4.18 Void Loop dengan While (1)

Kode pada *void loop* (dalam arduino) dapat ditulis didalam section **/*USER CODE BEGIN 3*/**. Didalamnya, terdapat *diffTime*, yakni selisih waktu saat ini (perhitungan dilakukan oleh **HAL_GetTick()**) dengan waktu awal *microcontroller* itu berjalan. Dengan arti lainnya, *diffTime* adalah waktu berapa lama program berjalan. Dengan demikian, *diffTime* akan terus bertambah seiring dengan bertambahnya waktu *microcontroller* beroperasi (lihat gambar di bawah).

Langkah berikutnya adalah melakukan konversi unit dari Volt ke bit yang akan dikirim ke MCP4725, yakni pada baris 301 dan 302. Untuk **AO0_bit** adalah hasil konversi 0-4095 bit

(bit tegangan) dari 0-24 Volt, sedangkan **A01_bit** adalah hasil konversi 0-4095 bit dari 0-10 Volt. Selanjutnya pada baris 304 s.d. 318 adalah kode untuk pengecekan koneksi DAC ke stm32 (lihat gambar 3.19).

```

297  /* USER CODE BEGIN 3 */
298  {
299      diffTime = HAL_GetTick() - startTime; // dalam ms
300
301      A00_bit = (uint16_t) 4095 * A00_volt / 24; // divided by 24V karena A00 mengirim tegangan max 24V
302      A01_bit = (uint16_t) 4095 * A01_volt / 10; // divided by 10V karena A00 mengirim tegangan max 10V
303
304      if(!MCP4725_isConnected(&myMCP4725_A00)){
305          /* Print that the DAC is connected */
306          errorDAC = 1;
307          myprintf("MCP A0 not connected | ");
308      }
309      if(!MCP4725_isConnected(&myMCP4725_A01)){
310          /* Print that the DAC is connected */
311          errorDAC = 1;
312          myprintf("MCP A1 not connected | ");
313      }
314
315      if(MCP4725_isConnected(&myMCP4725_A00) && MCP4725_isConnected(&myMCP4725_A01)){
316          errorDAC = 0; /* Print that the DAC is NOT connected */
317          myprintf("Both MCP4725 s Connected -- ");
318      }

```

Gambar 4.19 Konversi Volt ke Bit dan Pengecekan Koneksi DAC

Sebagai tambahan fungsi setValue pada baris 321 dan 322 (lihat gambar 3.20) berperan untuk mengirim nilai konversi bit tegangan kedalam komponen DAC. Sehingga DAC dapat mengeluarkan tegangan yang sesuai dengan tegangan yang diinginkan dalam program. Jika diringkas, kode pada baris 301 s.d 322 memiliki alur, yakni konversi tegangan (Vvolt ke bit), pengecekan koneksi komponen DAC, dan mengirimkan tegangan (dalam bit ke DAC).

```

321      setValue_A00(A00_bit);
322      setValue_A01(A01_bit);

```

Gambar 4.20 Mengirimkan Nilai Hasil Konversi Ke Komponen DAC

Berikutnya didefinisikan dt, yakni selisih antara diffTime dan waktu iterasi sebelumnya (lihat gambar 3.21). Dengan prevTime = HAL_GetTick(). Kemudian saat dt lebih besar daripada waktu sample (timeSampling) maka akan dijalankan program *controller* sekaligus menulis atau menyimpan data. Dalam hal ini timeSampling di-set 100 milidetik (ms). Maka saat iterasi waktu sekarang dikurangi waktu lalu lebih dari timeSampling maka yang dilakukan adalah mengirimkan sinyal kontrol baru.

```

324  /*-----STORING TO MICRO SD-----*/
325  uint32_t bytesWrote;
326  dt = diffTime - prevTime; // juga digunakan dalam integrator, derivative PID
327  if(dt >= timeSampling){
328      fres = f_write(&fil, stringKirim, strlen(stringKirim), &bytesWrote);
329      k++;

```

Gambar 4.21 Definisi dt dan Penyimpanan Ke Micro SD

Adapun apabila perubahan *setpoint* diinginkan maka kode pada baris 332 s.d 362 dengan kondisi diffTime yang terus bertambah (lihat gambar 3.22).

```

331 // NN: Forward and Backward
332 if(diffTime >= 3000){
333     if(diffTime < 243000){
334         SP = 50; // naik dari 0s ke 50s
335     }
336     else if(diffTime >= 243000 && diffTime < 483000){
337         SP = 100; // naik dari 50s ke 75s
338     }
339     else if(diffTime >= 483000 && diffTime < 723000){
340         SP = 150; // naik dari 75s ke 100s
341     }
342     else if(diffTime >= 723000 && diffTime < 963000){
343         SP = 200; // naik dari 100s ke 125s
344     }
345     else if(diffTime >= 963000 && diffTime < 1203000){
346         SP = 150; // naik dari 125s ke 150s
347     }
348     else if(diffTime >= 1203000 && diffTime < 1443000){
349         SP = 100; // naik dari 150s ke 175s
350     }
351     else if(diffTime >= 1443000 && diffTime < 1683000){
352         SP = 50; // naik dari 175s ke 200s
353     }

```

Gambar 4.22 Perubahan *Setpoint* dalam Kondisi yang Berbeda

Untuk *diffTime* lebih dari 1683000 ms maka akan dilakukan stopping, yakni--yang pertama—melakukan close file *write.txt* (baris 355). Kemudian pada baris 356 dilakukan *demounting* drive. Lalu menormalkan *value* DAC menjadi nol (baris 358 dan 359) dan meresetpin relay (baris 360 pada gambar 3.23).

```

354     else if(diffTime >= 1683000){
355         f_close(&fil); //Close file, don't forget this!
356         f_mount(NULL, "", 0); //De-mount drive
357         // normalkan kondisi semua off
358         AO0_volt = 0;
359         AO1_volt = 0;
360         HAL_GPIO_WritePin(GPIOE, GPIO_PIN_11, GPIO_PIN_RESET);
361         while(1);
362     }

```

Gambar 4.23 Penutupan File dan *Demounting* Micro SD serta Reset nilai AO dan GPIO

Kemudian beranjak ke bagian konversi pembacaan dari ADC untuk analog input (AI). Konversi tersebut akan mengubah hasil pembacaan (AO0_bit dan AO1_bit) bit dari *microcontroller* ke bentuk analog, yaitu tegangan AI dari 0-10 Volt (lihat gambar 3.24).

```

438 timestamp = HAL_GetTick() - startTime;
439 /* KONVERSI bit ke Volt */
440 // scaling langsung dikali 10V bukan 3.3V, karena sudah dlm bentuk bit
441 AI0_volt = (float) AI0_bit * 10 / 4096; // PA0
442 AI1_volt = (float) AI1_bit * 10 / 4096; // PA1
443 AI2_volt = (float) AI2_bit * 10 / 4096; // PA2

```

Gambar 4.24 Konversi ADC dari Bit ke Tegangan

Pada baris selanjutnya (baris 444 pada gambar 3.25) pada gambar di bawah, dilakukan untuk proses menyimpan data ke dalam file *.txt*. Sebagai tambahan, pada baris 450 *HAL_UART_Receive* untuk mengirimkan string ke dalam uart atau serial di komputer (PC).

```

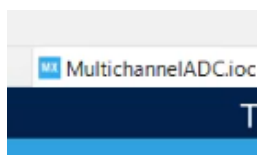
444 sprintf(stringKirim,"%d, %d, %f, %f, %f, %f, %f, %f, %f, %f, %f\n",k,timestamp, AO0_volt,
445
446 if(fres == FR_OK) {
447     myprintf("Status: %s",stringKirim)]
448 } else {
449     if(awal==0){ myprintf("f_write error (%i)\r\n"); }
450     HAL_UART_Receive(&huart4, (char*)Rx_buffer, 50, 500);
451     sprintf(Rx_buffer_string,"%s",Rx_buffer);
452     char* Rx_buffer_print = strtok(Rx_buffer_string,"\r\n");
453 // sscanf(Rx_buffer_print, "[SP=%d],[KP=%d],[KI=%d]", &SP, &KP, &KI);
454 // myprintf("SP = %d | KP = %d | KI = %d\n",SP, KP, KI);
455 }

```

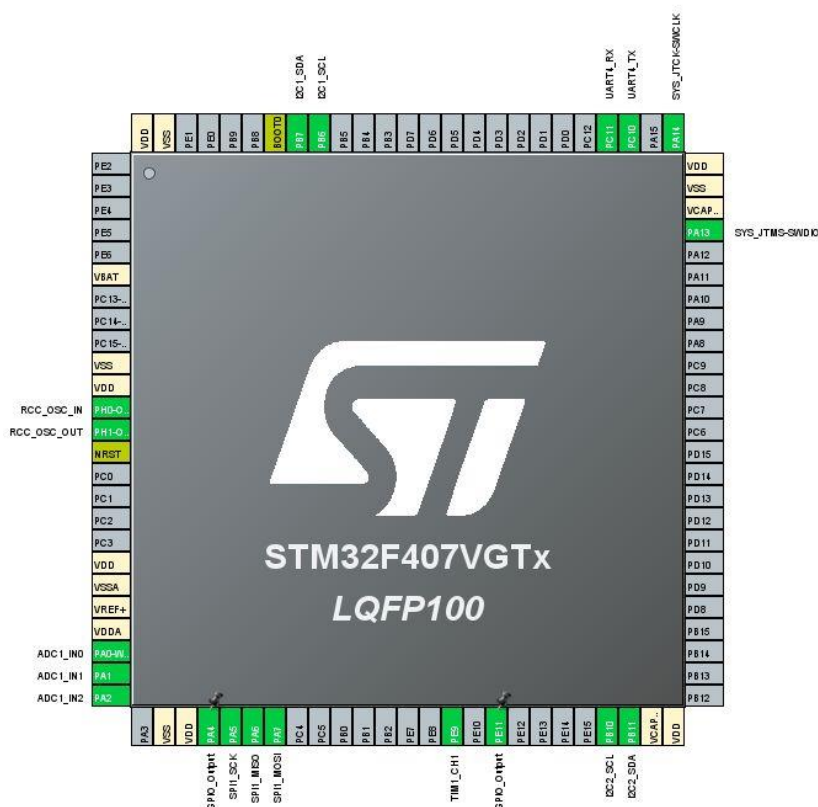
Gambar 4.25 Penulisan atau Pengiriman *String*.

4.7 Interface STM32

Gambar 3.26 dibawah merupakan interface.ioc, yakni **MultichanneADC.ioc**. Pada pin PA0, PA1, dan PA2 digunakan untuk komponen ADC atau analog input (lihat gambar 3.27). Namun, pin yang digunakan hanya dua saja, yakni PA0 dan PA1 sesuai dengan jumlah AI yang digunakan. Kemudian untuk PA4 s.d PA7 digunakan untuk komponen *micro sd*. Adapun pada pin PE(sebagai pin timer untuk komponen *relay* Berikutnya adalah pin UART4_RX dan UART4_TX digunakan untuk komunikasi serial (usb ttl) ke PC. Terakhir pin PB7 dan PB6 digunakan untuk komunikasi *microcontroller* ke komponen DAC.

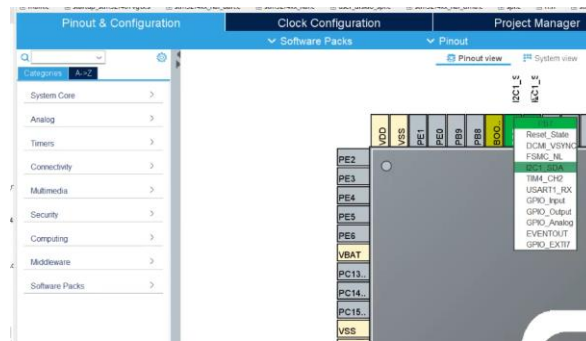


Gambar 4.26 Tab Terbukanya MultichanneADC.ioc



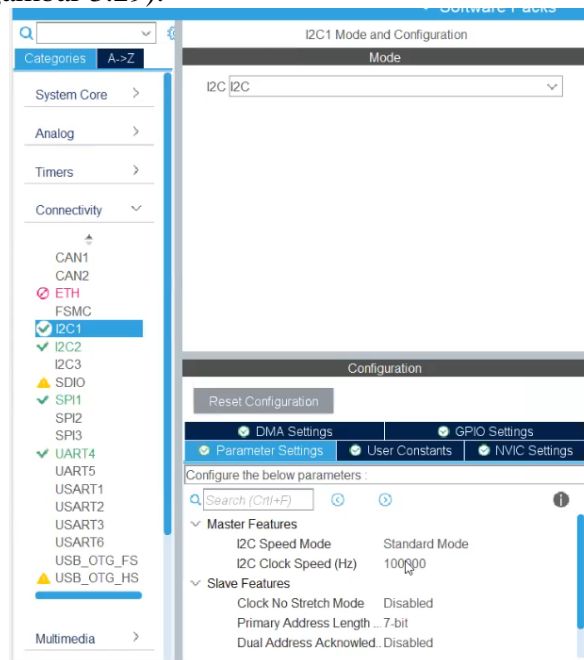
Gambar 4.27 Pin di STM32

Sebagai tambahan, untuk melihat spesifikasi atau melakukan konfigurasi pada pin yang tersambung dengan komponen tertentu dapat dilakukan sebagai berikut. Pada *window MultichanneADC.ioc*, temukan *dropdown "connectivity"* di sebelah kanan (lihat gambar 3.28)



Gambar 4.28 *Project dan Configuration*

Misalkan ingin melihat konfigurasi dari komponen I2C1 maka klik “connectivity” dan kemudian klik “I2C1” (gambar 3.29).

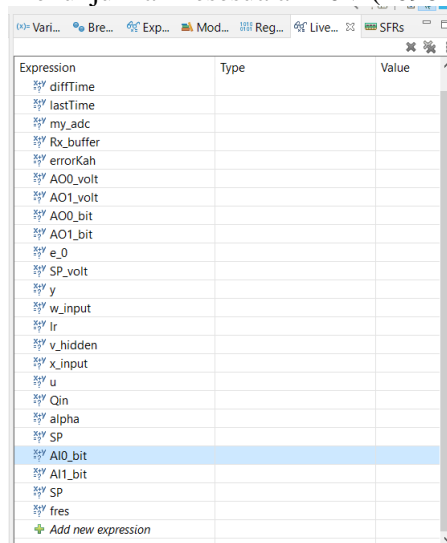


Gambar 4.29

BAB 5 Troubleshooting

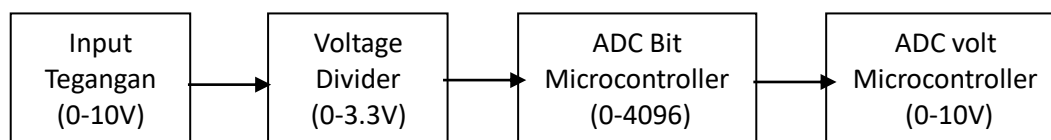
5.1 Tegangan Pembacaan ADC Tidak Sesuai

Tegangan pembacaan ADC yang tidak sesuai diketahui dengan memberikan tegangan supply 10V ke modul analog input (AI). Kegagalan ini tampak pada program debug “Live Expression” variable AI0_bit atau AI1_bit yang tidak menunjukkan kesesuaian 10V (4096 bit).



Gambar 5.1. Live Expression AO0_volt dan AO1_volt

Solusi permasalahan ini yakni mengecek kembali konversi tegangan power supply ke mikrokontroler, tegangan pembacaan analog input dengan cara mengatur trimmer. Berikut merupakan diagram alir yang menunjukkan proses konversi untuk pembacaan tegangan.



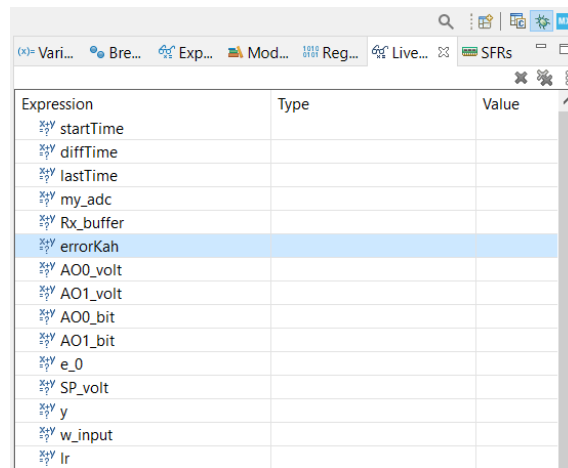
Gambar 5.2. Diagram Alir Konversi Tegangan Input

5.2 Address DAC MCP4725 Tidak Terbaca

Pastikan GY 0x61 dan MCP 0x60 terpasang dengan benar dan tidak terbalik. Jangan run program ketika laptop terpasang charger karena akan menimbulkan aliran ground ganda. Langkah dengan debugging melalui window Live Expression debug pada variable errorDAC. Apabila terindikasi adanya kegagalan pembacaan address errorDAC=1 dan text “MCP not connected” pada komunikasi serial. Program untuk mengidentifikasi error address DAC diberikan pada gambar berikut.

```
304 if(!MCP4725_isConnected(&myMCP4725_A00)){
305     /* Print that the DAC is connected */
306     errorDAC = 1;
307     myprintf("MCP A0 not connected | ");
308 }
309 if(!MCP4725_isConnected(&myMCP4725_A01)){
310     /* Print that the DAC is connected */
311     errorDAC = 1;
312     myprintf("MCP A1 not connected | ");
313 }
314
315 if(MCP4725_isConnected(&myMCP4725_A00) && MCP4725_isConnected(&myMCP4725_A01)){
316     errorDAC = 0; /* Print that the DAC is NOT connected */
317     myprintf("Both MCP4725 s Connected -- ");
318 }
319
```

Gambar 5.3. Kode Identifikasi errorDAC



Expression	Type	Value
startTime		
diffTime		
lastTime		
my_adc		
Rx_buffer		
errorKah		
AO0_volt		
AO1_volt		
AO0_bit		
AO1_bit		
e_0		
SP_volt		
y		
w_input		
Ir		

Gambar 5.4. Live Expression errorDAC

Selain itu, error address dapat dilihat pada serial monitor pada COM USB-TTL. Error address ini dapat diselesaikan dengan menghentikan run program/debug, mencabut keseluruhan supply USB STM dan power supply external, kemudian menyalakan prototype kembali.

5.3 Kartu SD Tidak Terdeteksi

Kartu SD tidak terdeteksi ditandai dengan tidak adanya tegangan keluaran pada terminal analog output setelah sekian lama menggunakan voltmeter (misal. pada program baru berjalan setelah 3 detik). Hal ini dapat diselesaikan dengan menekan tombol reset button pada STM32 untuk menjalankan ulang program. Kemungkinan lain yang dapat terjadi dikarenakan sekuen error DAC mendahului sekuen penulisan ke microSD. Solusinya yakni menyelesaikan permasalahan DAC terlebih dahulu.

5.4 Data TXT pada MicroSD Tidak Muncul Setelah Running

Jika data yang terbaca pada mikro SD terjadi masalah maka diperlukan pengambilan data ulang. Langkah lain yang dapat dicoba ialah memperbaiki baris kode if ($\text{diffTime} < \text{_waktu_}$). Hindari penggunaan HAL_Delay() atau delay() karena eksekusi program akan dipaksa berhenti untuk beberapa saat.

5.5 Tegangan MOSFET IRFP250 tidak berubah saat Adjustment

Tegangan Power Mosfer IRFP250 tidak berubah setelah adjustment dapat disebabkan adanya kerusakan pada bahan semikonduktor Mosfet. Langkah yang diperlukan yakni mengganti Mosfet dengan tipe atau spesifikasi yang sama.

5.6 MOSFET IRFP250 Overheat atau Berbau

Kejadian overheat merupakan kondisi yang wajar apabila arus drain-source Mosfet mencapai lebih dari 0.5A. Untuk meminimisasi temperature disipasi ini, diberikan kipas 12V. Langkah berupa memvalidasi dengan pengukuran arus beban perlu dilakukan. Percobaan yang telah dilakukan untuk tugas akhir sebelumnya memperlihatkan bahwa Mosfet ini mampu mengalirkan arus hingga 1.2A. Apabila dijumpai Mosfet berbau, aliran angin dapat diperbesar dengan mengganti kipas.