PROYEK AKHIR – VE 180626

PERANCANGAN *ARM REMOTE TESTER* UNTUK MENGUJI PERGERAKAN *ARM* ROBOT RHINO MENGGUNAKAN MIKROKONTROLER STM32

Andi Octriyanto Bahar
NRP 10311600000036

Pembimbing
Imam Arifin, S.T., M.T.
Dr. Eng. Imam Wahyudi Farid, S.T., M.T.
Muhammad Fajar Adityo, S.T.

DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

PROYEK AKHIR – VE 180626

PERANCANGAN *ARM REMOTE TESTER* UNTUK MENGUJI PERGERAKAN *ARM* ROBOT RHINO MENGGUNAKAN MIKROKONTROLER STM32

Andi Octriyanto Bahar
NRP 10311600000036

Pembimbing
Imam Arifin, S.T., M.T.
Dr. Eng. Imam Wahyudi Farid, S.T., M.T.
Muhammad Fajar Adityo, S.T.

DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

FINAL PROJECT – VE 180626

DESIGN OF ARM REMOTE TESTER FOR ARM RHINO ROBOT MOVEMENT TESTING USING MICROCONTROLLER STM32

Andi Octriyanto Bahar
NRP 10311600000036

Supervisor
Imam Arifin, S.T., M.T.
Dr. Eng. Imam Wahyudi Farid, S.T., M.T.
Muhammad Fajar Adityo, S.T.

DEPARTMENT OF ELECTRICAL AUTOMATION ENGINEERING
Vocational Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2019

PERNYATAAN KEASLIAN PROYEK AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun kesuluruhan Proyek Akhir saya dengan judul **"PERANCANGAN *ARM REMOTE TESTER* UNTUK MENGUJI PERGERAKAN *ARM* ROBOT RHINO MENGGUNAKAN MIKROKONTROLER STM32"** adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 12 Agustus 2019

Andi Octriyanto Bahar
NRP. 10311600000036

Halaman ini sengaja dikosongkan

PERANCANGAN *ARM REMOTE TESTER* UNTUK MENGUJI PERGERAKAN *ARM* ROBOT RHINOMENGGUNAKAN MIKROKONTROLER STM32

PROYEK AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Ahli Madya Teknik
Pada
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui:

Pembimbing 1	Pembimbing 2
Imam Arifin, S.T., M.T.	Dr. Eng. Imam Wahyudi Farid
NIP.197302222002121001	NPP. 199020191186

28/08/'19

SURABAYA
JULI, 2019

Halaman ini sengaja dikosongkan

PERANCANGAN *ARM REMOTE TESTER* UNTUK MENGUJI PERGERAKAN *ARM* ROBOT RHINO MENGGUNAKAN MIKROKONTROLER STM32 DI PT BHIMASENA RESEARCH AND DEVELOPMENT

PROYEK AKHIR

Disusun oleh:

Andi Octriyanto Bahar NRP. 10311600000036

Menyetujui,

Kepala *Human Resources Department*, Pembimbing Perusahaan,

BHIMASENA Research & Development

M. Nur Cahyo Utomo
NIK. 021042017

M. Fajar Adityo, S.T.
NIK. 020322016

Halaman ini sengaja dikosongkan

PERANCANGAN *ARM REMOTE TESTER* UNTUK MENGUJI PERGERAKAN *ARM* ROBOT RHINO MENGGUNAKAN MIKROKONTROLER STM32

Andi Octriyanto Bahar
10311600000036

Pembimbing 1 : Imam Arifin, S.T., M.T.
Pembimbing 2 : Dr. Eng. Imam Wahyudi Farid, S.T., M.T.
Pembimbing 3 : Muhammad Fajar Adityo, S.T.

ABSTRAK

Robot Rhino merupakan *Explosive Ordnance Disposal Robot* (EOD Robot) yang digunakan untuk penanganan barang berbahaya dengan kendali jarak jauh. Robot EOD pada umumnya terdiri dari dua bagian utama yaitu bagian *chassis* dan *arm*. Saat ini, pengujian *arm* (lengan robot) perlu menggunakan laptop, *development board* dan CAN *transceiver* untuk memberikan sebuah instruksi dari *gamepad* ke *arm*. Pengujian *arm* saat ini dirasa kurang fleksibel karena banyaknya peralatan yang mengakibatkan pengujian tidak dapat dipindah tempatkan dengan mudah serta banyaknya konektifitas kabel yang dibutuhkan.

Alat *Arm Remote Tester* merupakan sebuah solusi untuk meningkatkan efisiensi pengujian *arm*, karena semua input/ouput, mikrokontroler dan CAN *transceiver* dirancang menjadi satu alat. Tahapan yang perlu dilakukan untuk membuat alat *Arm Remote Tester* adalah dengan menentukan fitur alat, merancang *Printed Circuit Board*, memprogram mikrokontroler STM32, pengujian alat, pengambilan data dan pembuatan laporan dari hasil data yang di dapat.

Hasil yang didapat saat menggunakan alat *Arm Remote Tester* adalah pengujian *arm* dapat lebih mudah dilakukan karena alat ini dirancang lebih *compact* sehingga mudah untuk dibawa dan hanya memerlukan konektifitas kabel komunikasi ke arm, kabel *power supply* untuk alat dan kabel *power supply* untuk *arm*.

Kata kunci : *Explosive Ordnance Disposal Robot* (EOD Robot), *chassis*, *arm* (lengan robot), *development board*, CAN *transceiver*, *Printed Circuit Board* (PCB), mikrokontroler STM32.

Halaman ini sengaja dikosongkan

DESIGN OF ARM REMOTE TESTER FOR ARM RHINO ROBOT MOVEMENT TESTING USING MICROCONTROLLER STM32

Andi Octriyanto Bahar
10311600000036

Supervisor 1 : Imam Arifin, S.T., M.T.
Supervisor 2 : Dr. Eng. Imam Wahyudi Farid, S.T., M.T.
Supervisor 3 : Muhammad Fajar Adityo, S.T.

ABSTRACT

The Rhino Robot is an Explosive Ordnance Disposal Robot (EOD Robot) which is used for handling dangerous items with a remote control. EOD robots generally consist of two main parts, namely the chassis and arm. At present, the arm (robot arm) test needs to use a laptop, development board and CAN transceiver to provide instructions from the gamepad to the arm. The arm testing is currently considered to be less flexible because the large number of equipment that causes testing cannot be easily moved and the amount of cable connectivity needed.

The Arm Remote Tester tool is a solution to improve the efficiency of arm testing, because all inputs / outputs, microcontrollers and CAN transceivers are designed into one device. The steps that need to be done to make the Arm Remote Tester tool is to determine the features of the tool, design a Printed Circuit Board, program the microcontroller STM32, test tools, retrieve data and make reports from the results of the data obtained.

The results obtained when using the Arm Remote Tester tool are arm testing can be more easily done because this tool is designed to be more compact so that it is easy to carry and only requires communication cable connectivity to the arm, power supply cable for tools and power supply cables for the arm.

Keyword : Explosive Ordnance Disposal Robot (EOD Robot), chassis, arm, development board, CAN transceiver, Printed Circuit Board (PCB), microcontroller STM32.

Halaman ini sengaja dikosongkan

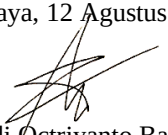
KATA PENGANTAR

Puji syukur atas kehadirat Tuhan Yang Maha Esa, atas rahmatnya dan karunia-Nya penulis dapat menyelesaikan Proyek Akhir dengan judul **" Perancangan *Arm Remote Tester* untuk Menguji Pergerakan *Arm* Rhino Robot Menggunakan Mikrokontroler STM32"** untuk memenuhi persyaratan menyelesaikan pendidikan Diploma 3 di Departemen Teknik Elektro Otomasi, Fakultas Vokasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam proses penyusunan penelitian ini, penulis telah banyak didukung oleh berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin mengucapkan terima kasih kepada seluruh keluarga yang selalu memberikan dukungan dan doa untuk keberhasilan dan kelancaran. Setelah itu, kepada Bapak Imam Arifin, S.T., M.T. selaku Dosen Pembimbing sekaligus Kepala Laboratorium Cyber Physical, Otomasi dan Robot Industri yang telah membimbing penulis dalam pengerjaan laporan ini, kepada bapak Dr. Eng. Imam Wahyudi Farid, S.T., M.T. selaku Dosen Pembimbing yang telah memberikan arahan serta nasihat kepada penulis dan juga kepada bapak Muhammad Fajar Adityo, S.T. selaku Pembimbing Lapangan di PT Bhimasena Research and Development yang selalu memberikan arahan serta saran dalam segala pekerjaan selama 10 bulan. Selain itu, kepada teman-teman anggota Laboratorium Cyber Physical, Otomasi dan Robot Industri angkatan 2016, 2017 dan 2018 atas pemberian semangat dan bantuan, serta semua pihak yang telah membantu secara langsung maupun tidak langsung dalam proses penyelesaian penelitian ini yang tidak dapat disebutkan satu-persatu.

Penulis menyadari bahwa buku Proyek Akhir ini belum sempurna, sehingga saran dan masukan sangat diharapkan untuk perbaikan kedepannya. Semoga buku ini dapat bermanfaat bagi pembaca dan masyarakat pada umumnya.

Surabaya, 12 Agustus 2019



Andi Octriyanto Bahar
NRP. 10311600000036

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring perkembangan teknologi yang semakin maju, semua hal yang biasa dilakukan oleh manusia mulai perlahan digantikan oleh peran robot. Hal tersebut dimaksudkan untuk membantu pekerjaan manusia dalam melakukan sesuatu yang tidak dapat dilakukan oleh manusia atau melakukan sesuatu yang memiliki resiko yang besar bagi manusia. Dengan demikian penggunaan robot sangat dibutuhkan pada beberapa bidang untuk mengurangi resiko kecelakaan, salah satunya adalah di bidang keamanan dalam menangani benda-benda berbahaya bagi manusia menggunakan robot *Explosive Ordnance Disposal* (EOD). Fungsi robot EOD tersebut adalah untuk mendekati dan mengambil benda-benda yang dianggap berbahaya [1].

PT Bhimasena Research and Development melakukan riset robot EOD bernama Rhino dengan tujuan untuk membantu manusia dalam mengamankan suatu benda yang dianggap berbahaya. Robot tersebut terdiri dari dua bagian utama yaitu *chasis* dan *arm*. *Chasis* memiliki fungsi untuk mobilitas robot (berpindah tempat) dan *arm* sebagai lengan untuk mengambil benda yang akan diamankan.

Robot Rhino dirancang untuk dapat bekerja dalam jarak yang cukup jauh dari pengendalinya dengan koneksi *wireless* atau *fiber optics cable*. Perangkat yang digunakan untuk mengendalikan robot bernama *Operator Control Unit* (OCU) [2]. Pada alat OCU terdapat *monitor*, tombol dan *joystick* untuk mengendalikan pergerakan robot.

Pada proses produksi dari robot ini, khususnya pada bagian *arm* sangat perlu dilakukan pengujian untuk meminimalkan *error* yang terjadi pada saat pengoprasian. Proses pengujian yang masih digunakan saat ini kurang fleksibel, karena perlu menggunakan perangkat laptop dan *development board* untuk dapat menghubungkan *gamepad* dengan *arm*.

Pada proyek akhir ini akan dibuat rancangan alat *Arm Remote Tester* yang nantinya dapat menggantikan peran laptop, *development board* beserta *gamepad* dalam proses pengujian *arm*.

1.2 Permasalahan

Proses pengujian *arm* (lengan) robot Rhino yang digunakan saat ini perlu menggunakan perangkat laptop dan *development board* untuk menghubungkan *gamepad* dengan *arm*. Pengujian seperti ini dirasa kurang fleksibel karena banyaknya alat yang digunakan, terutama jika digunakan pada saat produksi robot tersebut kedepannya.

1.3 Batasan Masalah

Arm (lengan) robot Rhino menggunakan komunikasi *Controller Area Network* (CAN) Bus untuk memberikan instruksi dan mendapatkan informasi dari *arm*. Alat *Arm Remote Tester* yang dibuat dapat memberikan instruksi dengan tombol, potensio dan *joystick*. Selain sebagai pemberi instruksi, alat tersebut juga dapat menampilkan informasi dari *arm* menggunakan LCD dengan komunikasi *Inter-Integrated Circuit* (I2C) Bus.

1.4 Tujuan

Tujuan dari proyek akhir ini adalah merancang dan mengimplementasikan alat *Arm Remote Tester* yang digunakan untuk menguji *arm* (lengan) pada robot Rhino. Dengan adanya alat *Arm Remote Tester* diharapkan dapat membantu serta mempermudah proses pengujian *arm* di waktu yang akan datang.

1.5 Metodologi

Metodologi yang digunakan terdiri dari studi literatur, perancangan alat, pengujian dan penyusunan buku proyek akhir. Pada tahap studi literatur, dasar teori yang menunjang dikumpulkan untuk dapat membantu dalam penyelesaian masalah yang timbul saat perancangan alat. Studi literatur berupa rujukan dalam bentuk jurnal internasional maupun nasional, buku dan artikel di internet.

Tahapan selanjutnya perancangan alat yang meliputi perangkat keras dan lunak. Pada perancangan perangkat keras dilakukan pemilihan komponen yang sesuai dengan fitur dari alat kemudian komponen disusun sedemikian rupa agar mudah dalam penggunaannya. Sedangkan pada perancangan perangkat lunak, program bahasa C dibuat menggunakan *software* CooCox IDE untuk membuat program mikrokontroler STM32F103C8T6 Blue Pill dengan

konfigurasi masukan digital beserta analog, *Direct Memory Access* (DMA), *Real-Time Operating System* (RTOS), komunikasi *Inter-Integrated Circuit* (I2C) dan komunikasi *Controller Area Network* (CAN).

Setelah melakukan perancangan alat, maka dilakukan tahap pengujian dan analisa tiap fungsi dari alat. Pengujian alat dilakukan untuk mengetahui kinerja pengiriman dan pembacaan data *arm*. Hasil pengujian kemudian dibandingkan dengan fungsi yang digunakan apakah sesuai dengan respons *arm* terhadap instruksi yang diberikan.

Tahapan yang terakhir melakukan penyusunan buku proyek akhir, buku proyek akhir disusun setelah perancangan alat telah berhasil dibuat, dapat berfungsi sebagaimana mestinya, pengambilan data pengujian terpenuhi dan permasalahan yang timbul dapat terselesaikan. Dengan adanya buku ini diharapkan dapat bermanfaat bagi pembaca dan bisa dijadikan pedoman dalam melanjutkan pengembangan alat yang telah dibuat.

1.6 Sistematika Penulisan

Untuk pembahasan lebih lanjut, laporan proyek akhir ini disusuh dengan sistematika sebagai berikut:

BAB I PENDAHULUAN

Berisi tentang latar belakang, perumusan masalah, batasan masalah, tujuan, metodologi penelitian, sistematika laporan buku proyek akhir.

BAB II DASAR TEORI

Menjelaskan teori-teori yang digunakan sebagai landasan dalam perancangan penelitian ini.

BAB III PERANCANGAN ALAT

Membahas tentang perancangan perangkat keras dan perangkat lunak. Perangkat keras meliputi pemilihan komponen, dan desain *Printed Circuit Board* (PCB). Sedangkan pada perangkat lunak berisi *flowchart* program yang dibuat.

BAB IV PENGUJIAN DAN ANALISA

Memaparkan hasil pengujian dan analisa data yang telah diperoleh.

BAB V PENUTUP

Memaparkan kesimpulan yang didapat dari pengujian yang telah dilakukan dan saran-saran untuk pengembangan alat lebih lanjut.

BAB II
DASAR TEORI

Bab ini membahas mengenai materi-materi hasil studi literatur yang dapat menunjang dalam tahap perancangan dan implementasi alat. Dasar teori yang didapatkan dari berbagai macam sumber yang digunakan.

2.1 Robot *Explosive Ordnance Disposal* (EOD)

Perkembangan robot pada era saat ini sangat pesat, salah satunya adalah dengan adanya *Explosive Ordnance Disposal* (EOD) robot dalam dunia militer. Robot tersebut dapat menggantikan manusia untuk mengintai, memusnahkan dan menangani bahan peledak atau barang berbahaya lainnya secara langsung [3].

Menurut sejarahnya, EOD robot pertama kali ditemukan pada tahun 1972 oleh Letnan Kolonel Peter Miller. Konsep awal robot tersebut berbentuk gerobak anti ledakan yang dilengkapi dengan pengait. Pengait tersebut berfungsi untuk menarik mobil yang akan dipindahkan ke lokasi lebih aman tanpa membahaykan keselamatan petugas penjinak bom [4].

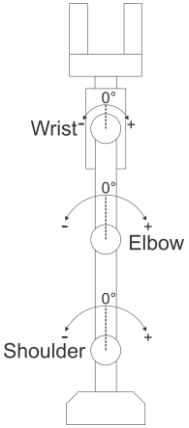
2.2 Robot Rhino

PT Bhimasena Research and Development melakukan riset robot EOD yang bernama Rhino robot. Rhino robot dibuat dengan tujuan untuk membantu manusia dalam mengamankan suatu benda yang dianggap berbahaya [1]. Robot tersebut terdiri dari dua bagian utama yaitu *chassis* dan *arm*. Tiap bagian robot memiliki fungsi tersendiri, *chassis* memiliki fungsi untuk mobilitas robot (berpindah tempat) dan *arm* sebagai lengan untuk mengambil benda yang akan diamankan.

Robot ini masih dalam tahap pengembangan, sehingga robot tersebut belum dapat dipublikasikan dalam bentuk gambar. Namun secara konsep robot Rhino didesain hampir sama dengan desain robot EOD yang sudah ada sekarang ini, pada Gambar 2.1 merupakan salah satu contohnya.

Gambar 2.1 Robot EOD
(Sumber : media.irobot.com)

Arm robot Rhino pada Gambar 2.2 dirancang berada di sudut 0 derajat sejajar tegak lurus antar *link* di setiap *joint*-nya, bernilai sudut positif jika ke arah kanan dan bernilai sudut negatif jika ke arah kiri.



Gambar 2.2 *Arm* robot Rhino

Ada pula *range* gerak dari *arm* robot Rhino yang terbatas agar tidak terjadi tabrakan antar *link* sebagai berikut.

Tabel 2.1 *Range* gerak *arm* robot Rhino

No	Nama Bagian *Arm*	Range Gerak	Ruang Gerak
1	*Turntable*	-180° sampai 180°	360°
2	*Shoulder*	0° sampai 180°	180°
3	*Elbow*	-170° sampai 0°	170°
4	*Wrist*	0° sampai 170°	170°
5	*Gripper Grab*	-180° sampai 180°	360°
6	*Gripper Rotation*	-180° sampai 180°	360°
7	*Camera Pan*	-180° sampai 180°	360°
8	*Camera Tilt*	-180° sampai 0°	180°

Tabel 2.1 merupakan batas pengoprasian tiap bagian *arm* yang meliputi range dan ruang geraknya. *Range* gerak merupakan batas sudut gerak yang diatur agar *arm* tidak mengalami tabrakan antar bagiannya, contoh pada bagian *elbow* memiliki *range* maksimal -170° agar bagian *shoulder* dan *wrist* memiliki jarak 10°. Ruang gerak merupakan besar sudut bebas setiap bagian *arm* saat dioperasikan.

2.3 Pemrograman Bahasa C

Bahasa C adalah termasuk dalam bahasa komputer tingkat tinggi yang instruksinya mudah untuk dipahami [6]. Penulisan dalam bahasa C bisa dibagi menjadi beberapa bagian sebagai berikut.

2.3.1 Konstanta dan Variabel

Konstanta adalah nilai yang tidak pernah berubah, sebaliknya variabel dapat berubah-ubah nilainya saat program dieksekusi [6]. Penulisan konstanta bisa dalam format desimal (basis 10), biner (basis 2), hexadesimal (basis 16), atau oktal (basis 8).

Tabel 2.2 Penulisan konstanta sesuai formatnya

Nama Format	Format Awal	Bilangan	Penulisan
Desimal		0,1,2,…,9	12
Biner	0b	0 dan 1	0b01101001
Hexadesimal	0x	0,1,2,..,9,A,B,C,...,F	0x1F
Oktal	0	0,1,2,….,7	011

Selain berupa angka, konstanta dapat ditulis dalam bentuk karakter yang kemudian oleh *compiler* akan diubah menjadi angka sesuai nilainya dalam table ASCII [6]. Penulisan karakter sebagai konstanta diawali dan diakhiri dengan *aphostrop* (‘).

Variabel sebelum digunakan harus dideklarasikan terlebih dahulu [6]. Deklarasi variabel terdiri dari tipe data dan nama variabel. Berdasarkan tipe data yang digunakan maka *compiler* akan mengalokasikan seberapa besar memori yang diperlukan.

Tabel 2.3 Tipe data dalam bahasa C

Tipe Data	Range Nilai	Alokasi Memori
unsigned char	0 .. 255	1 Byte
char	-128 .. 127	1 Byte
unsigned int	0 .. 65535	2 Byte
int	-32768 .. 32767	2 Byte
unsigned short	0 .. 65535	2 Byte
short	-32768 .. 32767	2 Byte
unsigned long	0 .. 4294967295	4 Byte
long	-2147483648 .. 2147483647	4 Byte
float	-3.402E+38 .. 3.402E+38	4 Byte
double*	-1.797E+308 .. 1.797E+308	8 Byte

*pada beberapa *compiler* hanya mendukung sampai tipe data float

Pendeklarasian variabel dapat dituliskan sesuai keinginan asalkan tidak mengandung beberapa hal berikut :

- Tidak mengandung simbol-simbol kecuali garis bawah (_)
- Tidak diawali dengan angka
- Penulisan tidak sama dengan operator-operator bahasa C
- Penulisan tidak sama dengan keyword bahasa C

2.3.2 Komentar

Komentar berfungsi untuk memberikan catatan dalam program yang dibuat [6]. Apa yang dituliskan dalam komentar diabaikan oleh *compiler*. Dengan demikian penambahan komentar tidak menambah besarnya kode hasil kompilasi dan tidak mempengaruhi alur program yang dibuat. Penulisan komentar dapat diawali ”/*” diakhiri ”*/” dan juga dapat diawali ”//” jika komentar hanya satu baris.

2.3.3 Pernyataan dan Blok Pernyataan

Merupakan sebuah instruksi lengkap yang sudah siap dieksekusi *compiler* untuk diubah menjadi bahasa mesin [6]. Pernyataan dapat berupa deklarasi variabel, ekspresi, pemanggilan fungsi, dan juga penggunaan operator lainnya yang selalu diakhiri dengan tanda ";".

Blok pernyataan adalah sekelompok pernyataan yang diawali "{" dan diakhiri "}" [6]. Blok pernyataan dibuat untuk mengelompokkan semua instruksi yang merupakan satu kesatuan pernyataan.

2.3.4 Ekspresi

Merupakan kombinasi antar variabel, konstanta dan operator untuk membentuk sebuah operasi yang dikehendaki. Operator adalah suatu fungsi untuk melakukan operasi tertentu dan melibatkan satu lebih *operand*. *Operand* sendiri adalah masukan (dapat berapa variabel atau konstanta) yang diolah oleh operator [6].

- Operator aritmatika

Operator aritmatika digunakan untuk mengerjakan fungsi-fungsi aritmatika desar.

Tabel 2.4 Operator aritmatika

Operator	Fungsi	Penulisan
+	Penjumlahan	A + B
-	Pengurangan dan negasi	A – B dan -A
*	Perkalian	A * B
/	Pembagian	A / B
%	Modulus (sisa pembagian)	A % B
++	Inkemen (penambahan 1 nilai)	A++ atau ++A
--	Dikremen (pengurangan 1 nilai)	A-- atau --A

- Operator relasi

Operator relasi digunakan untuk mendapatkan hasil perbandingan dari dua nilai.

Tabel 2.5 Operator relasi

Operator	Fungsi	Penulisan
==	Persamaan	A == B
>	Lebih besar	A > B
<	Lebih kecil	A < B
>=	Lebih besar sama dengan	A >= B
<=	Lebih kecil sama dengan	A <= B
!=	Pertidaksamaan	A != B

- Operator manipulasi bit

Operator manipulasi bit digunakan untuk mengolah data dengan orientasi per bit dari *operand*-nya.

Tabel 2.6 Operator manipulasi bit

Operator	Fungsi	Penulisan
|	Fungsi OR bit	A | B
&	Fungsi AND bit	A & B
~	Fungsi komplemen bit	~A
^	Fungsi XOR bit	A ^ B
>>	Geser bit ke kanan	A >> B
<<	Geser bit ke kiri	A << B

- Operator logika

Operator logika digunakan untuk operasi logika dengan memperlakukan nilai *operand*-nya sebagai nilai logika.

Tabel 2.7 Operator logika

Operator	Fungsi	Penulisan
||	Fungsi logika OR	A || B
&&	Fungsi logika AND	A && B
!	Fungsi logika negasi/inverter	!A

- Operator penugasan

Operator penugasan berfungsi untuk mengisi variabel sebelah kiri operator dengan nilai sebelah kanan operator. Dalam bahasa C operator ini dapat dikombinasikan dengan operator aritmatika dan juga operator manipulasi bit untuk meringkas penulisannya.

Tabel 2.8 Operator penugasan

Operator	Fungsi	Penulisan
=	A sama dengan B	A = B
+=	A sama dengan A + B	A += B
-=	A sama dengan A – B	A -= B
*=	A sama dengan A * B	A *= B
/=	A sama dengan A / B	A /= B
%=	A sama dengan A % B	A %= B
&=	A sama dengan A & B	A &= B
|=	A sama dengan A | B	A |= B
^=	A sama dengan A ^ B	A ^= B
<<=	Geser bit A ke kiri sebanyak B	A <<= B
>>=	Geser bit A ke kanan sebanyak B	A >>= B

2.3.5 Percabangan dan Seleksi Kondisi

Tak jarang sebuah program membutuhkan percabangan baik itu bersyarat ataupun tanpa syarat. Percabangan bersyarat meliputi proses seleksi atau pengujian kondisi dari suatu masukan untuk menentukan alur program berikutnya.

- if – else

Seleksi kondisi ini melakukan pengecekan pada ekspresi logika untuk menjalankan sebuah blok pernyataan.

```
if (ekspresi logika 1)
{
        pernyataan1
}
else if (ekspresi logika 2)
{
        pernyataan2
}
else
{
        pernyataan3
}
```

- switch – case

Seleksi kondisi ini digunakan dengan menguji persamaan antara nilai dalam variabel dengan konstanta-konstanta tertentu.

```
switch (variabel)
{
        case konstanta1 :
                pernyataan1
        break
        case konstanta2 :
                pernyataan2
        break
        case konstanta3 :
                pernyataan3
        break
        default :
                pernyataan4
        break
}
```

- goto

Percabangan ini adalah percabangan tanpa syarat. Instruksi ini menyebabkan eksekusi dilanjutkan ke alamat yang ditunjukkan menggunakan label. Label sendiri adalah alamat dari program yang akan dituji oleh instruksi ini [6].

```
goto Label;
Label;
Pernyataan
```

2.3.6 Perulangan

Dalam bahasa C terdapat *keyword* perulangan untuk mengulang sebuah proses jika dibutuhkan pada program, dengan demikian program tidak perlu dituliskan secara berulang.

- for

Perulangan ini digunakan untuk mengulang sebuah pernyataan atau pernyataan blok selama logika masih benar. Perulangan ini memerlukan inisialisai dan ekspresi perulangan untuk memberikan batas perulangan tersebut dapat dilakukan [6].

```
for (inisialisasi; syarat perulangan; ekspresi perulangan)
{
        pernyataan perulangan
}
```

- while

Perulangan ini untuk mengulang sebuah pernyataan atau blok pernyatan sesuai dengan syarat perulangan.

```
while (syarat perulangan)
{
        pernyataan perulangan
}
```

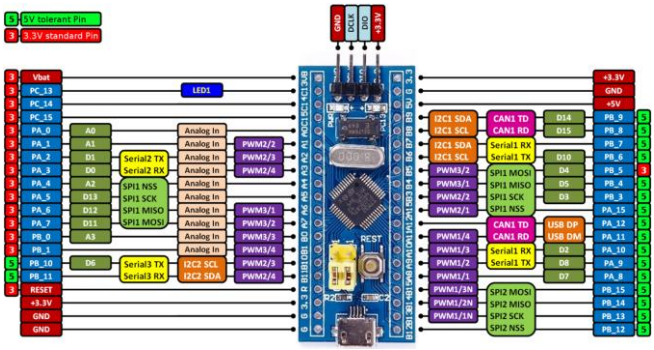
- do – while

perulangan ini dapat melakukan perulangan sebuah pernyataan atau blok pernyataan dengan menguji syarat diakhir perulangan, sehingga pernyataan selalu dieksekusi minimal satu kali.

```
do
        pernyataan perulangan
while (syarat perulangan)
```

2.4 Mikrokontroler

Merupakan alat yang digunakan jika proses kontrol melibatkan operasi yang kompleks baik itu aritmatika, logika, pewaktuan, atau lainnya yang akan sangat rumit bila diimplementasikan dengan komponen-komponen diskrit [5]. Salah satu contohnya adalah *development board Blue Pill* yang menggunakan mikrokontroler STM32F103C8T6. Modul tersebut memiliki beberapa fitur diantaranya I/O digital, ADC, PWM, USART, SPI, I2C, CAN dan *supply* tegangan yang dibutuhkan modul ini sebesar 5 *volt* DC.



Gambar 2.3 STM32F103C8T6 *Blue Pill*
(Sumber : os.mbed.com)

STM32F103C8T6 *Blue Pill* tidak dilengkapi dengan *downloader*, sehingga untuk memasukkan program pada STM32F103C8T6 perlu menambahkan modul ST-Link yang dapat dilihat pada Gambar 2.4. Modul ST-Link tersebut dihubungkan pada pin +3.3V, DIO, DCLK dan GND.

Gambar 2.4 *ST-Link* V2
(Sumber : www.bukalapak.com)

2.4.1 *Analog to Digital Converter* (ADC)

Merupakan sebuah alat yang mengubah sinyal / besaran analog menjadi satuan digital sesuai ketelitian dari perangkat ADC (banyak bit dari ADC) [5].

Sampling pada ADC adalah pengambilan data dalam satu titik waktu tertentu. Semakin banyak titik sampling dalam kurun waktu yang sama, semakin teliti data digital yang dihasilkan[5]. Sebagai contoh ADC 12 bit, maka nilai digital yang dihasilkan memiliki rentang dari 0-4095. Hasil konversi ADC juga dapat dihitung dengan persamaan 1 berikut.

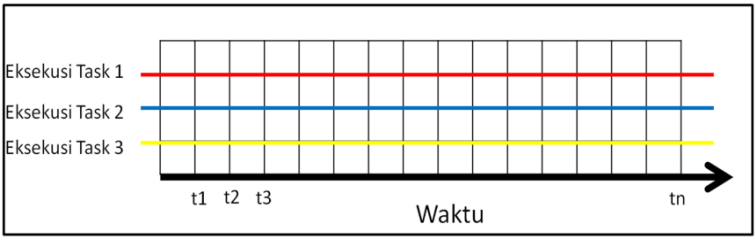
$$ADC = \frac{V_{in} \times ADC_{max}}{V_{ref}} \quad [1]$$

2.4.2 *Real-Time Operating System* (RTOS)

Merupakan sebuah sistem operasi yang memiliki komponen utama untuk mengatur adanya *multitasking*. *Multitasking* sendiri ialah kemampuan sistem operasi untuk menjalankan banyak *task* dalam rentang waktu tertentu. Setiap task yang ada terbentuk dari setiap proses yang harus dijalankan oleh sebuah komputer ataupun Embedded system [8].

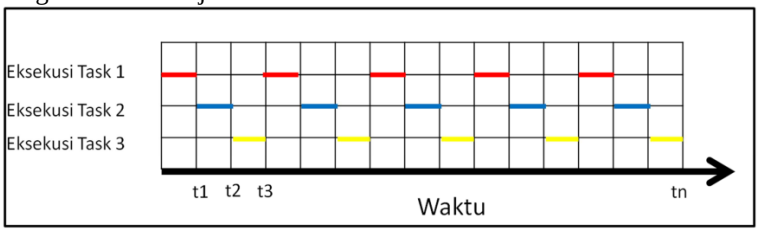
Konsep *multitasking* dari sistem operasi akan memudahkan desain sebuah aplikasi yang kompleks. Terdapat beberapa keuntungan dari *multitasking* sebagai berikut:

- Memungkinkan sebuah aplikasi yang kompleks dibagi menjadi bagian lebih kecil, sederhana, dan mudah diatur.
- Bagian-bagian kecil dari sebuah aplikasi dapat dengan mudah diuji coba, ditelusuri, dan didaur ulang untuk kebutuhan lain.
- Detail alur dari aplikasi serta pengaturan waktu eksekusi yang kompleks dapat dihilangkan dari kode aplikasi. Hal tersebut sudah menjadi tanggung jawab dari sistem operasi itu sendiri.



Gambar 2.5 Konsep RTOS

Gambar 2.5 menunjukkan bagaimana konsep multitasking bekerja. Tiga task akan berjalan dalam waktu yang sama dan waktu eksekusinya sama. Akan tepi pada prosesor konvensional, konsep tersebut belum dapat dijalankan. Namun setiap task yang ada secara bergantian akan dijalankan secara konkurensi.



Gambar 2.6 Konsep konkurensi

Konsep konkurensi digambarkan pada Gambar 2.6, dimana semua *task* akan dieksekusi sesuai prioritasnya dan setiap *task* akan dieksekusi dalam rentang waktu yang kecil hingga proses perpindahan antar eksekusi *task* seolah-olah melakukan beberapa proses secara bersamaan.

2.4.3 *Direct Memory Access* (DMA)

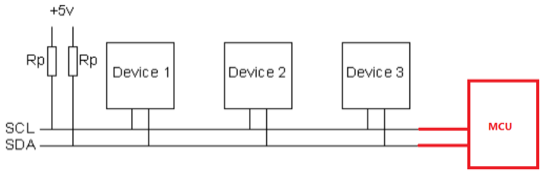
Merupakan suatu alat pengendali khusus yang disediakan untuk memungkinkan transfer blok data langsung antar perangkat eksternal dan memori utama, tanpa intervensi terus menerus dari prosesor [7].

Dengan adanya DMA maka prosesor dapat mengerjakan proses yang lain tanpa harus menunggu proses transfer data ke memori. Pengaplikasian DMA dapat digunakan untuk transfer data ADC, komunikasi I2C, komunikasi SPI dan lain sebagainya.

2.4.4 *Inter-Integrated Circuit* (I2C)

Merupaakan cara berkomunikasi atau protokol komunikasi antara perangkat secara serial dengan 2 kabel, yaitu serial data (SDA) dan serial clock (SCL) [5].

Dalam komunikasi I2C ada bagian yang disebut *master* (perangkat pengendali) dan *slave* (perangkat yang dikendalikan). *Master* umumnya adalah berupa mikrokontroler, sedangkan *slave* adalah perangkat lainnya (contoh adaptor LCD 16x2). Tipikal koneksi master dan slave ke bus I2C sesuai Gambar 2.7 dengan nilai resistor pull-up 10k Ohm.

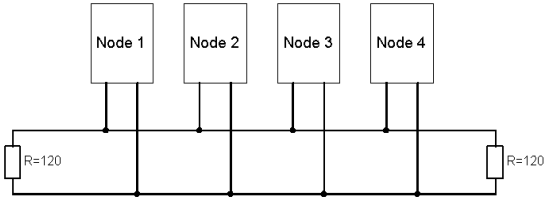


Gambar 2.7 Konfigurasi *Inter-Integrated Circuit* (I2C)

2.4.5 *Controller Area Network* (CAN)

Secara umum komunikasi ini digunakan pada bidang otomotif, tetapi juga dapat diterapkan pada bidang lainnya. komunikasi CAN adalah multi-master serial bus yang dapat melakukan transmisi data dengan efisiensi lebih baik antar perangkatnya [9]. Komunikasi ini bersifat *broadcast*, dimana setiap kontroler dapat melakukan *broadcast* data pada CAN bus yang berupa 2 kabel CAN *High* dan CAN *Low*.

Dalam komunikasi CAN, data yang ditransmisikan memiliki identitas yang digunakan untuk membedakan data. Sehingga untuk dapat mengakses data, kontroler membutuhkan *filter* data untuk memilah data menurut ID yang sudah ditetapkan. Tipikal koneksi antar kontroler ke komunikasi CAN sesuai Gambar 2.8 dan CAN High dengan CAN Low harus dipisah menggunakan resistor 120 Ohm.



Gambar 2.8 Konfigurasi *Controller Area Network* (CAN)

BAB III
PERANCANGAN *ARM REMOTE TESTER*

Pada Bab ini dijelaskan mengenai proses perancangan dan pembuatan alat *Arm Remote Tester* yang digunakan untuk pengujian *arm* Rhino robot. Perancangan alat terdiri dari *design requirement*, blok fungsional alat, perancangan perangkat keras dan perancangan perangkat lunak.

3.1 *Design Requirement*

Alat *Arm Remote Tester* digunakan untuk menguji *arm* (lengan robot) pada Rhino robot dengan memberikan masukan berupa sudut untuk tiap *joint* dan intensitas cahaya untuk LED pada *arm*. Target pembuatan alat yang ingin dicapai dalam perancangan ini terdiri dari fitur-fitur sebagai berikut:

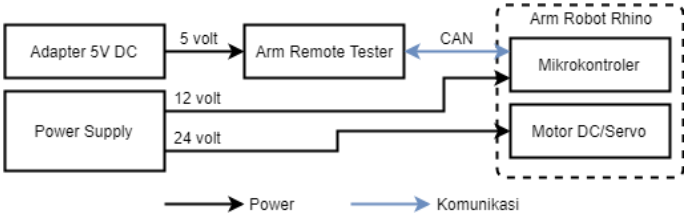
- Memiliki LCD, untuk menampilkan informasi *arm* dan menampilkan sudut atau intensitas cahaya yang akan diberikan.
- Memiliki tombol mode LCD, karena keterbatasan ruang LCD maka tampilan LCD dapat diubah sesuai kebutuhan.
- Memiliki tombol pilih joint, untuk memilih joint mana yang akan diberikan instruksi.
- Memiliki tombol mode potensio, untuk mengatur masukan potensio sebagai instruksi sudut *joint* atau intensitas cahaya LED.
- Memiliki tombol menerapkan nilai potensio.
- Memiliki 2 tombol inkremental, untuk memberikan instruksi sudut secara inkremental positif atau negatif.
- Memiliki potensio, untuk mengatur nilai instruksi sudut joint atau intensitas cahaya LED.
- Memiliki *joystick* 2 axis, untuk sumbu X dan Y pada sistem invers kinematik.
- Memiliki 2 tombol sumbu Z, untuk sistem invers kinematik.
- Memiliki tombol emergency.
- Memiliki *Controller Area Network* (CAN) *transceiver*.

Dari semua fitur yang diinginkan, mikrokontroler yang dirasa sesuai digunakan untuk alat ini adalah STM32F103C8T6 Blue Pill. Pada mikrokontroler tersebut memiliki pin I/O yang cukup untuk

semua masukan berserta keluaran, memiliki *interface* I2C Bus dan *interface* CAN Bus.

3.2 Blok Fungsional Alat

Pada alat *Arm Remote Tester* terdapat tiga cara pengujian *arm* yaitu dengan memberikan instruksi *absolute*, *incremental* dan *inverse kinematic.* Selain digunakan untuk memberikan instruksi, alat ini juga dapat menampilkan informasi yang terdapat pada *arm* seperti posisi, kecepatan, arus listrik dan kondisi motor/servo pada *arm*.



Gambar 3.1 Konfigurasi alat *Arm Remote Tester*

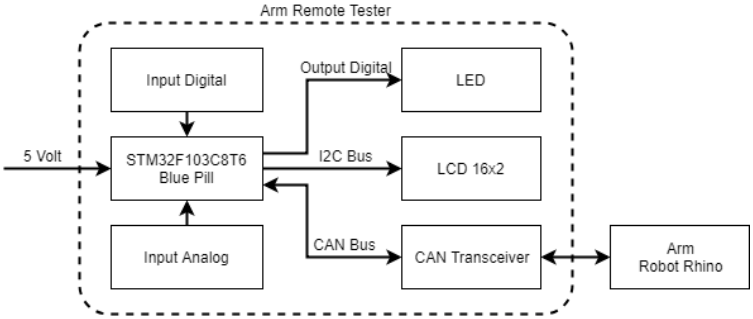
Sumber *power* untuk alat *Arm Remote Tester* perlu menggunakan adaptor 5 *volt* dengan konektor keluaran USB Mikro C yang langsung dihubungkan ke mikrokontroler. Sumber *power* yang diperlukan *arm* menggunakan *power supply* dengan keluaran 12 volt untuk mikrokontroler pada *joint* dan 24 volt untuk *motor* atau *servo*.

Keluaran sinyal yang dihasilkan alat *Arm Remote Tester* dikirimkan ke *arm* menggunakan komunikasi *Controller Area Network* (CAN) Bus, dimana setiap data yang dikirimkan memiliki ID tersendiri sesuai dengan instruksi apa yang diberikan oleh alat tersebut untuk *arm*. Sedangkan masukan alat *Arm Remote Tester* yang didapatkan dari *arm* berupa data informasi posisi, kecepatan, arus listrik pada motor dan kondisi *arm* yang nantinya akan ditampilkan pada *Liquid Crystal Display* (LCD) 16x2. Selain untuk menampilkan informasi *arm*, LCD 16x2 juga dapat digunakan untuk melihat sudut yang akan diberikan untuk menggerakkan *arm*.

3.3 Perancangan Perangkat Keras

Perancangan alat *Arm Remote Tester* memiliki beberapa fitur utama yang berupa masukan digital dan masukan analog yang

digunakan untuk memberikan instruksi pada arm pada saat pengujian. Selain masukan, alat tersebut memiliki keluaran LED, LCD dan CAN *transciver* untuk komunikasi antara alat dengan *arm* robot Rhino.



Gambar 3.2 Rancangan alat *Arm Remote Tester*

Perangkat keras pada alat ini didesain memiliki ukuran yang tidak terlalu besar dan mudah untuk digunakan, untuk memenuhi aspek tersebut maka dibuat lah perancangan perangkat keras pada alat *Arm Remote Tester* yang terdiri dari pemilihan komponen dan desain PCB.

3.3.1 Pemilihan Komponen

Pemilihan komponen diperlukan untuk mempertimbangkan komponen apa yang sesuai dengan fitur yang diperlukan dan juga mempertimbangkan ukuran dari komponen tersebut.

- *Tactile Switch*

Gambar 3.3 *Tactile Switch* 12x12mm
(Sumber : www.bukalapak.com)

Tactile Switch tersebut memiliki ukuran 12x12mm sehingga tidak terlalu kecil dan tidak terlalu besar untuk digunakan pada alat ini. Dengan ukuran tersebut, pengguna akan lebih mudah untuk menggunakan setiap tombol pada alat ini.

- *Potentiometer*

Gambar 3.4 *Potentiometer 10K Ohm*
(Sumber : www.bukalapak.com)

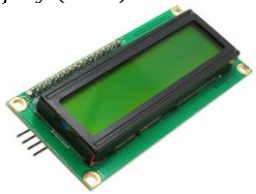
Potentiometer tersebut memiliki bentuk dengan *knob* menghadap ke atas, dengan bentuk tersebut maka penggunaan potensio dapat lebih mudah untuk diubah nilainya.

- *Joystick*

Gambar 3.5 *Joystick 2 axis*
(Sumber : www.bukalapak.com)

Joystick tersebut memiliki ukuran yang tidak terlalu besar, dengan bentuk tersebut maka tempat yang dibutuhkan untuk menempatkan *Joystick* tersebut tidak terlalu luas dan tetap nyaman untuk digunakan.

- *Liquid Crystal Display* (LCD)



Gambar 3.6 LCD 16x2 dengan adaptor I2C
(Sumber : www.bukalapak.com)

Liquid Crystal Display (LCD) yang digunakan memiliki 2 baris dan tiap baris dapat menuliskan 16 karakter. LCD tersebut membutuhkan modul adaptor I2C untuk mengurangi jumlah pin mikrokontroler yang digunakan untuk menggunakan LCD tersebut.

- *Light-Emitting Diode* (LED)

Gambar 3.7 LED 2mm
(Sumber : www.bukalapak.com)

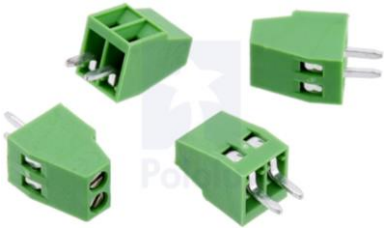
Light-Emitting Diode (LED) memiliki ukuran yang cukup untuk digunakan sebagai indikator pada alat ini.

- *Controller Area Network* (CAN) *Transceiver*

Gambar 3.8 *Controller Area Network* (CAN) *Transceiver* MCP2551
(Sumber : www.digikey.com)

Controller Area Network (CAN) *Transceiver* MCP2551 yang digunakan pada alat ini berbentuk *Small Outline Integrated Circuit* (SOIC) *package* sehingga tempat yang diperlukan untuk memasang komponen tersebut tidak terlalu luas.

- Terminal Block



Gambar 3.9 *Terminal Block*
(Sumber : www.digikey.com)

Terminal Block yang digunakan memiliki 2 pin yang nantinya digunakan sebagai keluaran CAN Bus untuk *arm* dan untuk opsi power jika menggunakan *power supply*.

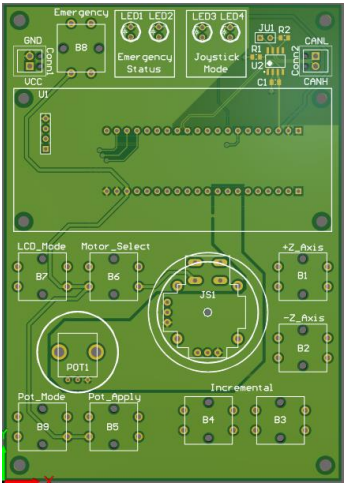
- Mikrokontroler STM32F103C8T6 *Blue Pill*

Gambar 3.10 Mikrokontroler STM32F103C8T6 *Blue Pill*
(Sumber : www.bukalapak.com)

Mikrokontroler yang digunakan selain memiliki ukuran yang tidak terlalu besar, jumlah pin yang digunakan juga disesuaikan dengan I/O analog dan digital, terdapat komunikasi I2C Bus dan terdapat komunikasi CAN Bus.

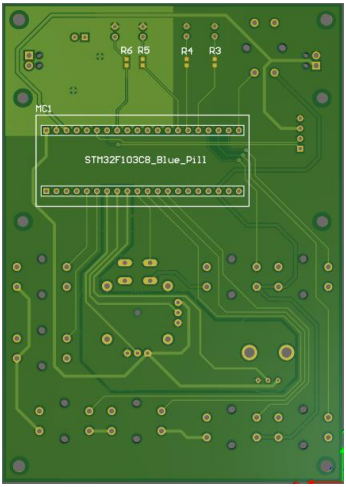
3.3.2 Desain *Printed Circuit Board* (PCB)

Desain *Printed Circuit Board* yang dibuat memiliki ukuran panjang 120 mm dan lebar 85 mm. Desain alat ini dibuat sekecil mungkin untuk mempermudah penggunaan yang hanya dipegang dengan satu tangan dan tangan lainnya untuk mengoprasikan alat.



Gambar 3.11 Desain *Top layer* PCB *Arm Remote Tester*

Bagian atas PCB terdapat 9 tombol, 1 potensio, 1 *joystick* 2 *axis*, 4 LED indikator, rangkaian CAN *Transceiver* dan 2 terminal blok. Semua komponen disusun sesuai dengan kemudahan dalam penggunaan alat *Arm Remote Tester.*



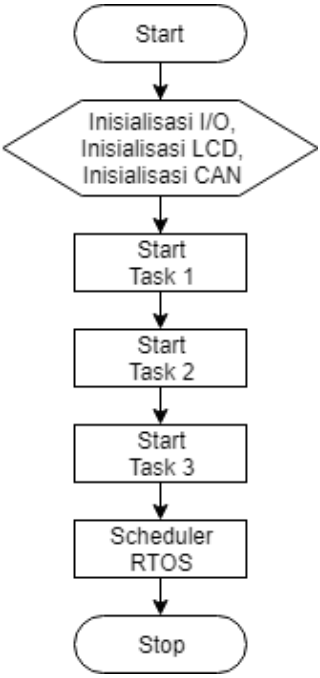
Gambar 3.12 Desain *Bottom layer* PCB *Arm Remote Tester*

Bagian bawah PCB terdapat mikrokontroler STM32F103C8T6 *Blue Pill* yang berfungsi sebagai kontroler alat ini.

3.4 Perancangan Perangkat Lunak

Perangkat lunak pada alat *Arm Remote Tester* dirancang untuk memiliki respons *real time* agar tidak mengganggu kinerja alat pada saat melakukan pengujian *arm* robot Rhino. Dengan adanya fitur (*Real-Time Operating System*) RTOS pada STM32 maka hal tersebut dapat dilakukan.

Pemanfaatan RTOS dapat digunakan untuk melakukan *multitasking* program. Dengan demikian program yang dibuat dibagi menjadi 3 *task* (tugas). *Task* 1 berisi program I2C untuk mengirimkan data ke LCD 16x2, *task* 2 berisi program pengecekan tombol dan ADC dari *potentiometer* dan *joystick*, *task* 3 berisi program CAN untuk mengirimkan instruksi serta meminta informasi dari arm.



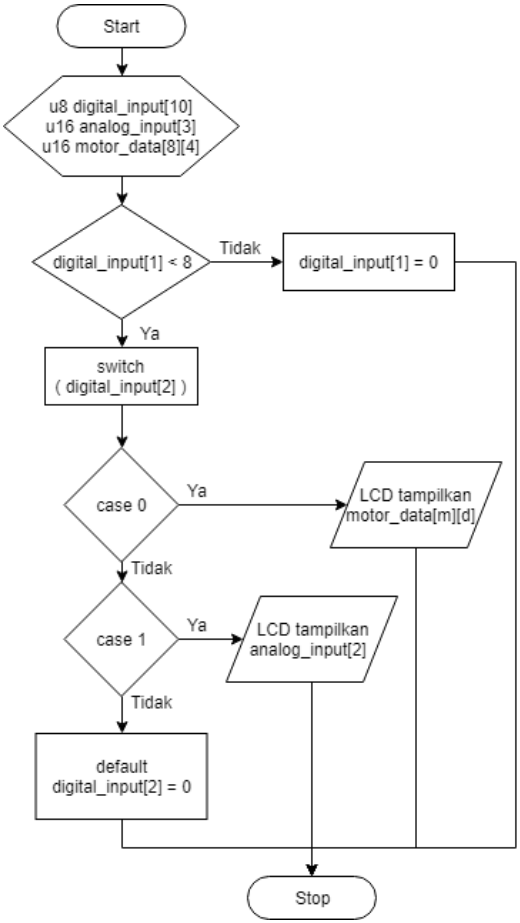
Gambar 3.13 Diagram alir program *main*

3.4.1 *Task* 1

Semua penugasan yang digunakan untuk menuliskan data pada LCD dikerjakan oleh *task* 1. Pada program *task* 1, terdapat variabel yang berfungsi untuk memberikan instruksi dan informasi yang akan ditampilkan oleh LCD.

Data yang pertama kali diterima oleh LCD merupakan data informasi dari *joint arm* yang berupa sudut untuk posisi, kecepatan untuk pergerakan dan arus pada motor/servo *joint arm*.

Data yang kedua akan ditampilkan jika tombol mode LCD ditekan. Isi dari data dua berupa nilai konversi potensio menjadi sudut dari -180 sampai 180 derajat dan menjadi persentase 0 sampai 100. Data sudut dan persentase tersebut digunakan untuk memberikan instruksi menggerakkan arm dengan nilai sudut dan mengatur intensitas cahaya menggunakan nilai persentase.

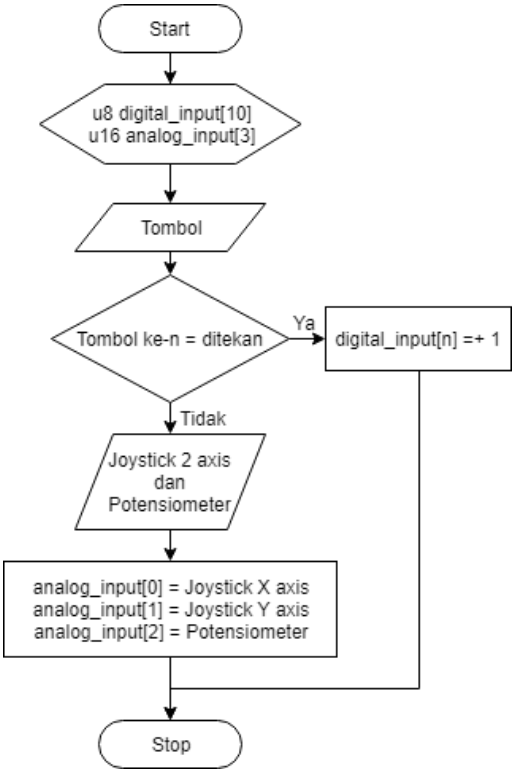


Gambar 3.14 Diagram alir program *task* 1

3.4.2 *Task* 2

Semua masukan untuk memberikan instruksi diproses oleh *task* 2, dimana semua masukan dari tombol, potensiometer dan *joystick* disimpat pada variabel penampung data digital dan data analog. Variabel penampung data digital memiliki 10 elemen yang setiap elemennya mewakili setiap tombol. Variabel penampung data analog memiliki 3 elemen yang setiap elemennya mewakili komponen yang menghasilkan data analog.

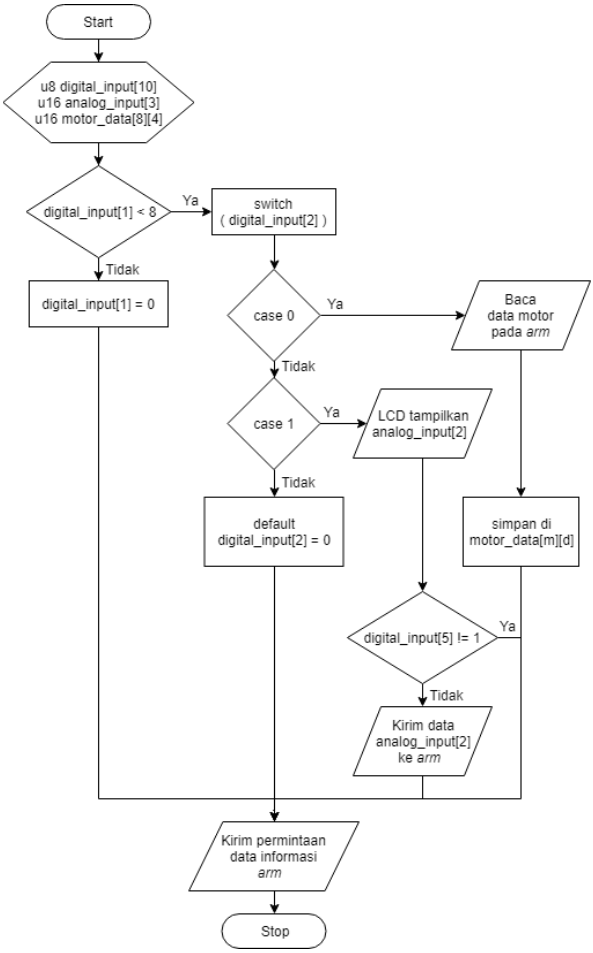
Semua tombol pada alat ini akan dicek secara berkala. Jika terdapat tombol yang ditekan, maka tombol tersebut akan menghasilkan penambahan nilai 1 dan nilai tersebut disimpan pada variabel penampung data digital.



Gambar 3.15 Diagram alir program *task* 2

3.4.3 *Task* 3

Proses pengiriman instruksi dan permintaan data informasi dilakukan oleh *task* 3. Instruksi yang dikirimkan berdasarkan dari masukan pada *task* 2. Instruksi yang dikirimkan memiliki 2 macam yaitu instruksi *absolute* dengan mengirimkan data sudut yang diinginkan dan instruksi *incremental* dengan mengirimkan data penambahan atau pengurangan posisi saat ini. Sedangkan permintaan informasi dilakukan setiap 500 ms.

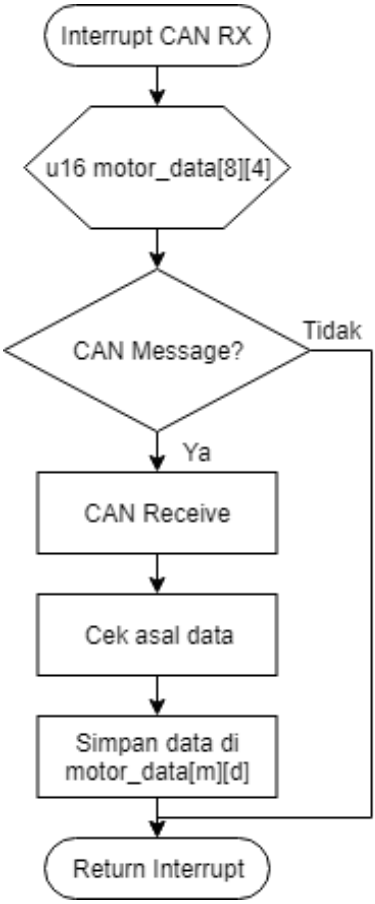


Gambar 3.16 Diagram alir program *task* 3

3.4.4 *Interrupt* CAN RX

Proses penerimaan data informasi dilakukan dengan program *interrupt*, dimana setiap ada data yang ditujukan untuk alat *Arm Remote Tester* maka penerimaan data akan lebih diutamakan terlebih dahulu. Selain menerima data, program ini juga memilah data yang

diterima menurut asal dari data tersebut dan disimpan pada variabel penampung.



Gambar 3.17 Diagram alir program *interrupt* CAN RX

BAB IV
PENGUJIAN DAN ANALISIS

Pada Bab ini akan memaparkan hasil dari pengujian alat *Arm Remote Tester* dan juga analisis dari hasil pengujian yang dilakukan. Pengujian yang dilakukan dapat dibagi menjadi pengujian fungsional alat dan pengujian respons alat saat digunakan.

4.1 Fungsional Alat

Dalam pengujian ini, akan diuji fungsional setiap instruksi yang diberikan ke *arm* dan informasi yang didapatkan dari *arm*. Hasil yang didapat akan dianalisis gangguan pada pemberian instruksi dan penerimaan data informasi serta apa yang mempengaruhi masalah tersebut dan bagaimana cara mengatasinya.

4.1.1 Instruksi

Pengujian ini dilakukan dengan menghubungkan alat *Arm Remote Tester* ke *arm* robot Rhino. Dengan pemasangan tersebut dapat dilihat instruksi yang diberikan apakah dapat berfungsi atau tidak. Hasil yang didapat telah dituliskan pada Tabel 4.1 dan Tabel 4.2.

Tabel 4.1 Pengujian fungsional pengiriman instruksi

No	Instruksi	Fungsional	Keterangan
1	Tombol emergency	Belum Berfungsi	-
2	Tombol mode LCD	Berfungsi	Dapat merubah tampilan LCD
3	Tombol pilih motor dc/servo	Berfungsi	Dapat merubah mode motor dc/servo yang akan diberikan instruksi
4	Tombol mode potensio	Berfungsi	Dapat merubah mode potensio sebagai sudut/ intensitas cahaya

Tabel 4.2 Pengujian fungsional pengiriman instruksi (Lanjutan)

No	Instruksi	Fungsional	Keterangan
5	Tombol terapkan nilai potensio	Berfungsi	Dapat melakukan pengiriman nilai potensio yang sudah diatur
6	Tombol inkremental negatif	Berfungsi	Dapat memberikan instruksi inkremental negatif
7	Tombol inkremental positif	Berfungsi	Dapat memberikan instruksi inkremental positif
8	Tombol sumbu Z negatif	Belum Berfungsi	-
9	Tombol sumbu Z positif	Belum Berfungsi	-
10	Potensio	Berfungsi	Dapat memberikan nilai sudut dan intensitas cahaya
11	*Joystick* 2 axis	Belum Berfungsi	-

Secara fungsional semua tombol dapat berfungsi, kecuali fitur yang memang masih belum memiliki program untuk fungsinya. Gangguan yang terjadi terdapat pada pemberian masukan digital dari tombol terkadang tidak berfungsi jika penekanan dilakukan terlalu cepat, tetapi hal ini sangat jarang sekali terjadi. Dalam mengatasi masukan yang ada kemungkinan tidak berfungsi tersebut usahakan penekanan tombol yang dilakukan tidak terlalu cepat.

4.1.1.1 Instruksi Tombol Mode LCD

Mode LCD pada alat *Arm Remote Tester* yang dibuat memiliki 2 mode yaitu mode informasi yang berisi data dari *arm* berupa posisi

sudut, kecepatan motor, arus motor dan kondisi *joint.* Sedangkan mode yang kedua berisi informasi nilai potensio sebagai masukan sudut atau intensitas cahaya LED. Berikut merupakan hasil pengujian tombol mode LCD pada alat *Arm Remote Tester*.

Tabel 4.3 Hasil pengujian tombol mode LCD

No	Hasil Pengujian	Keterangan
1		Mode informasi motor
2		Mode informasi nilai potensio sebagai masukan

4.1.1.2 Instruksi Tombol Pilih Motor

Mode pilih motor pada alat *Arm Remote Tester* terdiri dari 8 mode yaitu *Turntable* (T), *Shoulder* (S), *Elbow* (E), *Wrist* (W), *Gripper Grab* (GG), *Gripper Rotation* (GR), *Camera Pan* (CP) dan *Camera Tilt* (CT). Berikut merupakan hasil pengujian tombol pilih motor pada alat *Arm Remote Tester* yang ditandai dengan perunbahan keterangan motor pada LCD bagian paling kiri.

Tabel 4.4 Hasil pengujian tombol pilih motor

No	Hasil Pengujian	Keterangan
1		Mode informasi motor *Turntable*
2		Mode informasi motor *Shoulder*

Tabel 4.5 Hasil pengujian tombol pilih motor (Lanjutan)

No	Hasil Pengujian	Keterangan
3		Mode informasi motor *Elbow*
4		Mode informasi motor *Wrist*
5		Mode informasi motor *Gripper Grab*
6		Mode informasi motor *Gripper Rotation*
7		Mode informasi servo *Camera Pan*
8		Mode informasi servo *Camera Tilt*

4.1.1.3 Instruksi Tombol Mode Potensio

Mode potensio terdiri dari mode sebagai masukan sudut *joint* dan masukan intensitas cahaya LED. Berikut merupakan hasil pengujian tombol mode potensio pada alat *Arm Remote Tester* yang ditandai dengan tanda panah di sebelah sudut dan LED.

Tabel 4.6 Hasil pengujian tombol mode potensio

No	Hasil Pengujian	Keterangan
1		Mode pilih nilai potensio sebagai sudut
2		Mode pilih nilai potensio sebagai intensitas cahaya LED

4.1.2 Informasi

Pengujian ini dilakukan untuk melihat fungsi LCD sebagai *monitring arm* pada alat *Arm Remote Tester* yang menampilkan informasi *arm* berupa data posisi, kecepatan, arus listrik dan kondisi pada motor DC/servo. Hasil pengujian yang didapat telah dituliskan pada Tabel 4.8.

Tabel 4.7 Pengujian pembacaan data informasi

No	Informasi	Fungsional	Keterangan
1	Posisi motor DC/servo	Berfungsi	Dapat menampilkan informasi posisi dari -180 sampai 180 derajat
2	Kecepatan motor DC/servo	Berfungsi	Dapat menampilkan informasi kecepatan dari 0 sampai 10000 rpm
3	Arus listrik motor DC/servo	Belum Berfungsi	-
4	Kondisi motor DC/servo	Belum Berfungsi	-

Secara fungsional LCD sudah dapat menampilkan posisi dan kecepatan motor DC/servo. Untuk informasi arus masih belum berfungsi dikarenakan sensor arus pada *arm* masih terjadi kerusakan, sedangkan untuk informasi kondisi masih belum befungsi dikarenakan mikrokontroler pada *joint* masih belum dapat melakukan diagnosa kondisi dari motor DC/servo. Gangguan yang terjadi pada pengujian ini adalah jika terdapat 2 data yang dituliskan ke LCD secara bersamaan maka pengiriman data akan mengalami error yang menyebabkan seluruh program dari alat ini berhenti, hal ini dapat terjadi karena LCD menggunakan komunikasi serial dan jika data yang dikirimkan belum sempurna maka data pengiriman tersebut menyebabkan *error* pada pngiriman selanjutnya. Selain terjadi gangguan, pada pembacaan data juga mengalami delay yang menyebabkan penulisan informasi sedikit terlambat jika dibandingkan dengan data informasi yang sudah diterima oleh mikrokontroler pada *joint*.

4.1.2.1 Informasi Posisi Motor DC/Servo

Berikut merupakan hasil pengujian pembacaan informasi posisi motor DC/servo pada alat *Arm Remote Tester* yang ditandai dengan perubahan angka dengan satuan derajat.

Tabel 4.8 Hasil pengujian pembacaan posisi motor DC/servo

No	Hasil Pengujian	Keterangan
1		Informasi motor *Elbow* saat pengujian berada di posisi sudut -160°
2		Informasi motor *Elbow* saat pengujian berada di posisi sudut -91°
3		Informasi motor *Elbow* saat pengujian berada di posisi sudut -10°

4.1.2.2 Informasi Kecepatan Motor DC/Servo

Berikut merupakan hasil pengujian pembacaan informasi kecepatan motor DC/servo pada alat *Arm Remote Tester* yang ditandai dengan perubahan angka dengan satuan rpm.

Tabel 4.9 Hasil pengujian pembacaan kecepatan motor DC/servo

No	Hasil Pengujian	Keterangan
1		Informasi motor *Elbow* saat pengujian berputar dengan kecepatan 2804 rpm
2		Informasi motor *Elbow* saat pengujian berputar dengan kecepatan 8147 rpm
3		Informasi motor *Elbow* saat pengujian berputar dengan kecepatan 4886 rpm

4.2 Transfer Data

Pengujian ini dilakukan dengan cara mengirimkan instruksi absolute dan inkremental dari *Arm Remote Tester* ke *arm* robot Rhino, kemudian data yang diterima oleh *arm* dibandingkan dengan data instruksi yang dikirimkan *Arm Remote Tester.* Berikut hasil pengujian yang didapat.

Tabel 4.10 Hasil pengujian transfer data instruksi absolute

No	Instruksi absolute ART	Diterima *Arm*
1	-160°	-16000
2	-140°	-14000
3	-120°	-12000
4	-100°	-10000
5	0°	0

Tabel 4.10 merupakan hasil yang didapatkan dari pengujian transfer data absolute, dapat dilihat bahwa nilai yang diterima oleh *arm* berkelipatan 100 karena mikrokontriler pada *arm* sudah diatur sedemikian rupa agar dua digit paling belakang bisa mewakili nilai belakang koma.

Tabel 4.11 Hasil pengujian transfer data instruksi inkremental

No	Instruksi inkremental ART	Diterima *Arm*
1	Positif	100
2	Negatif	-100

Tabel 4.11 merupakan hasil yang didapatkan dari pengujian transfer data inkremental, dimana penambahan 1 nilai positif atau negatif perlu dikali 100 terlebih dahulu.

4.3 Validasi Hasil Pengujian

Pada bagian ini, dilakukan validasi dari hasil pengujian terhadap target-target yang telah dijelaskan pada Bab III. Pada Tabel 4.12 dan Tabel 4.13 dituliskan hasil dari validasi perancangan alat yang telah dilakukan. Dengan keterangan tersebut dapat dipastikan bahwa perancangan alat ini telah berhasil dibuat.

Tabel 4.12 Validasi efisiensi perancangan alat

No	Efisiensi	Pengujian saat ini	*Arm Remote Tester*
1	Bentuk	Terdiri dari perangkat-perangkat dan modul-modul	*Compact* dalam satu perangkat dengan ukuran yang dihasilkan 120 x 85 mm dengan tinggi 55 mm
2	Penggunaan	Masukan menggunakan *Gamepad* yang dihubungkan ke laptop	Masukan menggunakan 9 tombol, 1 potensio dan 1 *joystick.* Semua komponen tersebut terpasang dalam satu *board*

Tabel 4.13 Validasi perancangan alat (Lanjutan)

No	Efisiensi	Pengujian saat ini	*Arm Remote Tester*
3	Konektifitas	Antar perangkat perlu konektifitas	Konektifitas hanya perlu kabel power 5V DC untuk alat dan 2 kabel komunikasi CAN untuk *arm*
4	Instruksi Absolute	Pengujian *arm* secara *absolute* menggunakan laptop untuk memberikan nilai sudut yang dituju	Pengujian *arm* secara *absolute* dapat dilakukan dengan mengatur nilai potensio sebagai nilai sudutnya
5	Instruksi Inkremental	Pengujian *arm* secara inkremental menggunakan *gamepad* untuk memberikan instruksi gerak joint kearah positif atau negatif	Pengujian *arm* secara *incremental* dapat dilakukan dengan menekan tombol *incremental* positif atau negatif
6	Informasi *Joint*	Data informasi tiap *joint* dibaca dan ditampilkan menggunakan serial monitor yang ditampilkan laptop	Data informasi tiap *joint* dapat dibaca alat dan ditampilkan pada LCD alat ini

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1 Kesimpulan

Pada Proyek Akhir ini telah dibuat rancangan alat *Arm Remote Tester* untuk menguji *arm* robot Rhino. Alat tersebut dapat memberikan instruksi dan dapat membaca data informasi tiap *joint*-nya. Instruksi yang digunakan untuk menguji *arm* memiliki 2 macam cara yaitu dengan cara memberikan posisi sudut yang akan dituju (*absolute*) dan dengan cara menambahkan setiap 1 sudut kearah positif atau negatif (incremental). Sedangkan data informasi yang diterima alat ini berupa posisi dan kecepatan putar motor DC/servo pada tiap *joint*. Alat yang dibuat menggunakan mikrokontroler STM32F103C8T6, yang dimana memiliki fitur komunikasi I2C untuk mengirimkan data ke LCD, komunikasi CAN untuk melakukan transfer data dengan *arm* dan pin I/O yang mencukupi untuk fungsi dari tiap instruksi.

Berdasarkan data yang diperoleh dari hasil pengujian dan analisis dapat dipastikan alat *Arm Remote Tester* dapat digunakan untuk pengujian *arm* robot Rhino. Hasil yang didapatkan dari perancangan alat ini sudah memenuhi target-target yang diharapkan seperti ukuran dengan panjang 120 mm, lebar 85 mm dan tinggi 55 mm, penggunaan lebih fleksibel, konektifitas kabel tidak terlalu banyak, dapat memberikan instruksi untuk pengujian dan dapat membaca data informasi *arm* robot Rhino.

5.2 Saran

Perancangan alat ini masih memerlukan pengembangan pada fitur *inverse kinematic*, bentuk yang perlu lebih disederhanakan, serta juga dapat dikembangkan penggunaan baterai untuk menyalakan alat ini. Dengan pengembangan tersebut maka pengujian *arm* menggunakan alat ini akan lebih fleksibel dan lebih mudah untuk digunakan saat melakukan produksi.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] J. J. Zeng, R. Q. Yang, W. J. Zhang, X. H. Weng, and Q. Jun, "Research on semi-automatic bomb fetching for an EOD robot,"*Int. J. Adv. Robot. Syst.*, vol. 4, no. 2, pp. 247–252, 2007.

[2] N. Checka, S. Schaffert, D. Demirdjian, J. Falkowski, and D. H. Grollman, "Handheld operator control unit," *Proc. Seventh Annu. ACM/IEEE Int. Conf. Human-Robot Interact. - HRI '12*, p. 137, 2012.

[3] L. Fan, X. Yao, H. Qi, L. Jiang, and W. Wang, "A single-hand and binocular visual system for EOD robot," *Proc. IEEE Int. Conf. Autom. Logist. ICAL 2007*, pp. 1930–1935, 2007.

[4] L. Dams, *Bomb Disposal Robot History*. www.youtube.com, 2014.

[5] Hardana, "Belajar Mudah Mikrokontroler ARM STM32," PT. Mitra Sinergi Optima: Jakarta, Indonesia. 2018.

[6] H.S. Bagus, "Pemrograman Mikrokontroler dengan Bahasa C," C.V Andi Offset: Yogyakarta, 2012.

[7] Margono, "*Direct Memory Access* (DMA)" 2009. Available: https://margono.staff.uns.ac.id/2009/04/08/direct-memory-access-dma/. [Accessed: 9 April 2019].

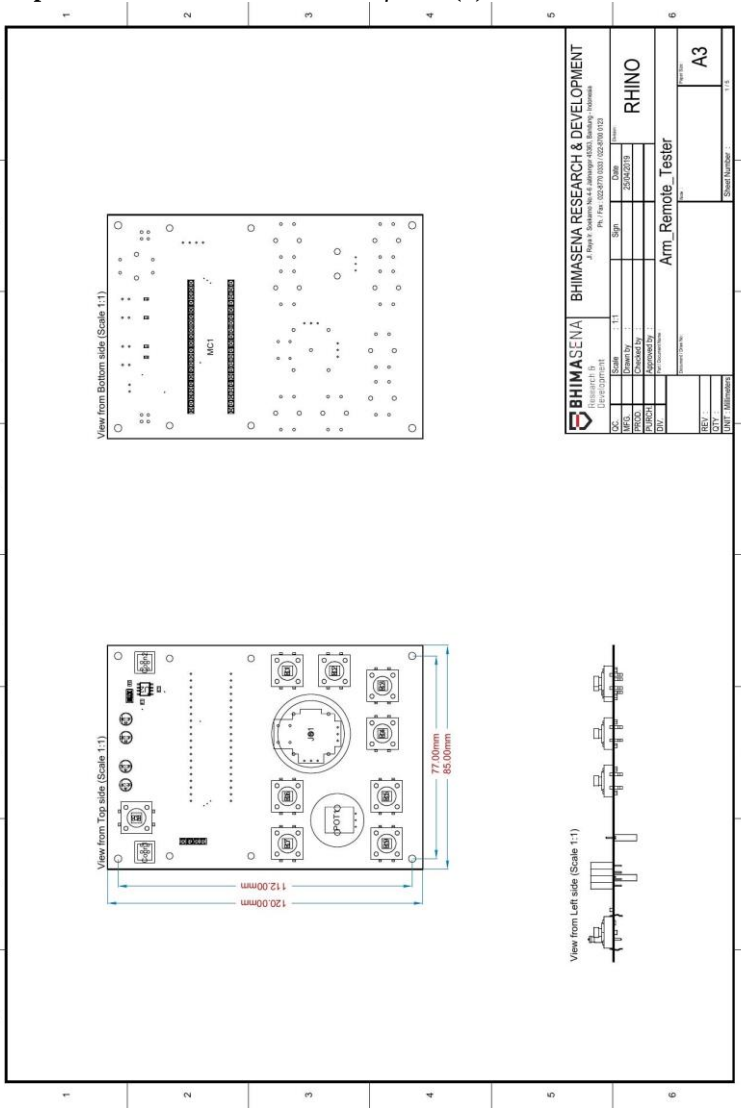
[8] J. Wisnu, P. Mursanto, dkk, "*Real Time Operating System* (RTOS) Teori & Aplikasi," Fakultas Ilmu Komputer, Universitas Indonesia, 2015.

[9] L. N. Wei, K. N. Chee, M. A. Borhanuddin, K. N. Nor, and Z. R. Fakhrul, "Review of Researches in Controller Area Networks Evolution and Applications", Department of Computer and Communication Systems, Faculty of Engineering, Universiti Putra Malaysia, 2010.

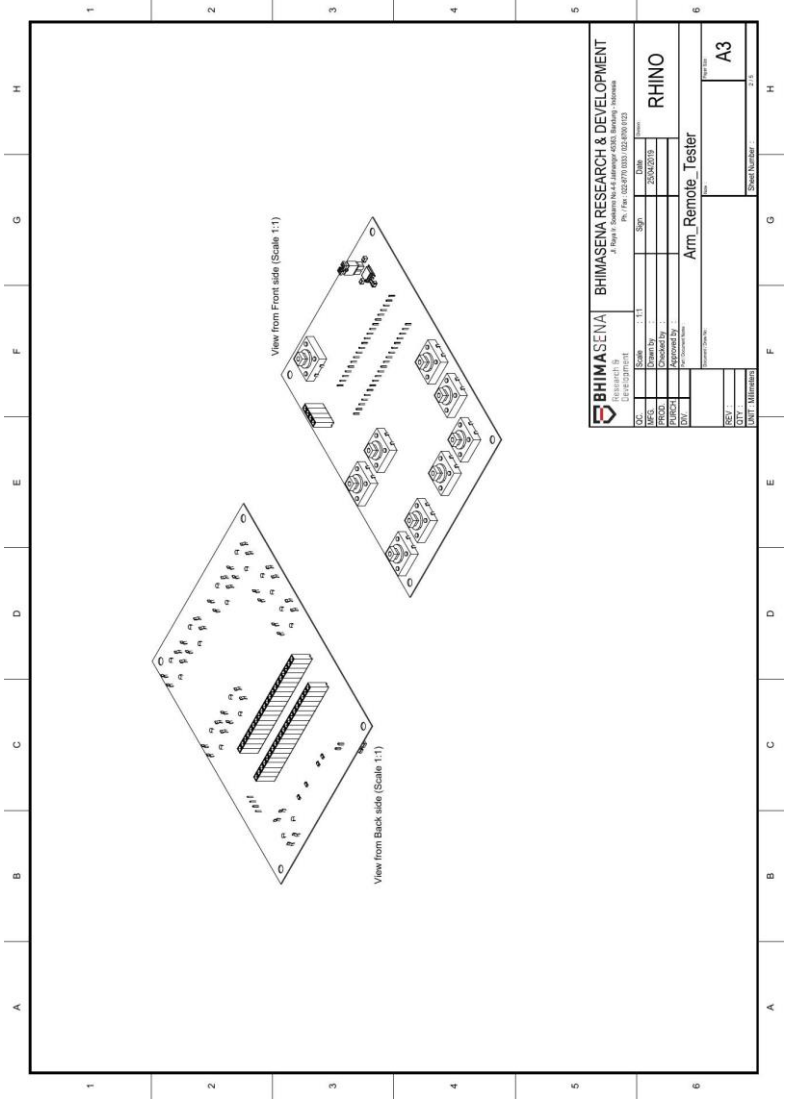
Halaman ini sengaja dikosongkan

LAMPIRAN

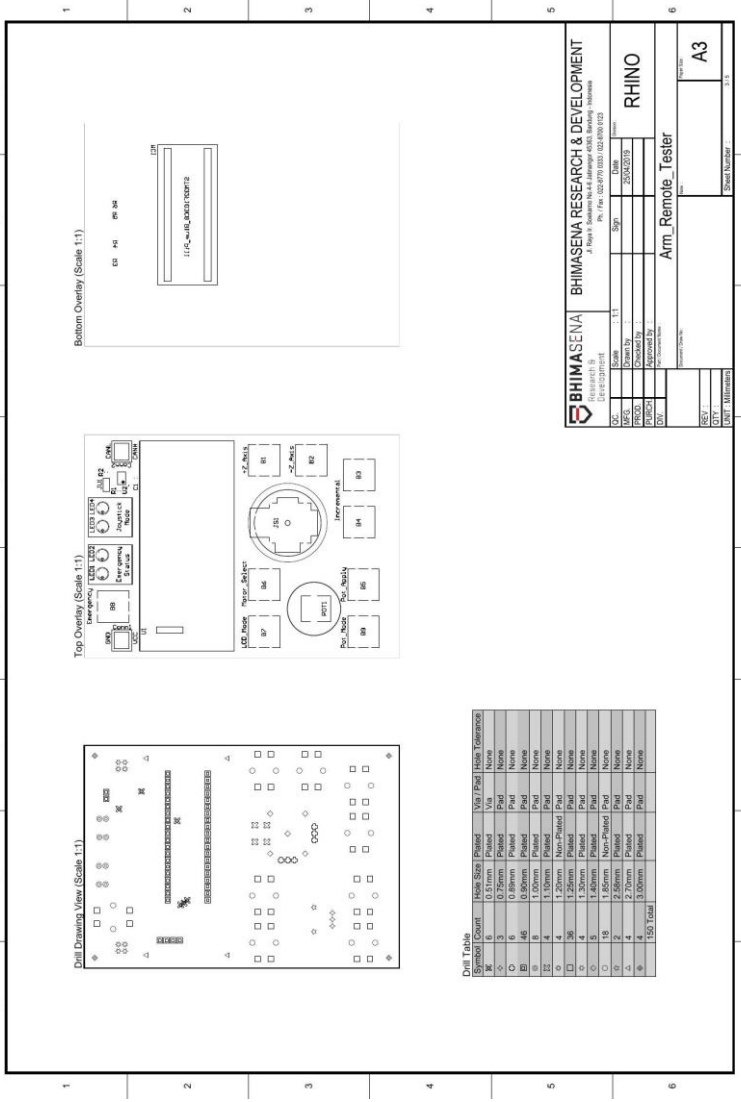
Lampiran 1 *Arm Remote Tester Draftman* (1)



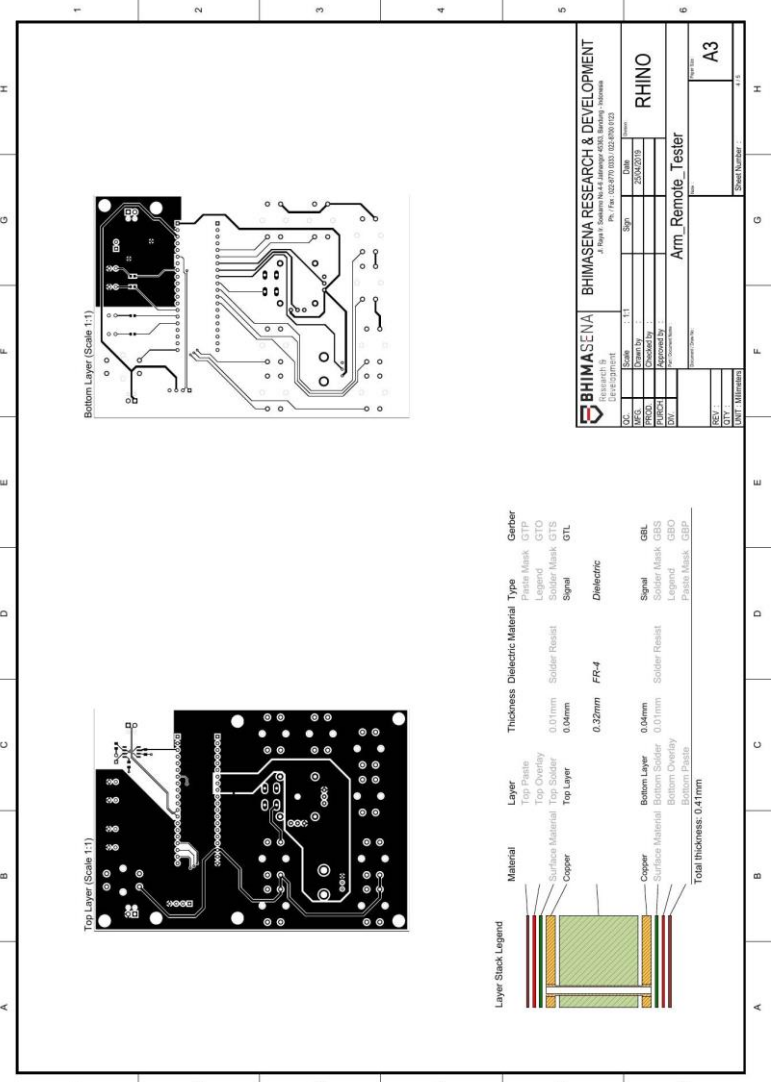
Lampiran 2 *Arm Remote Tester Draftman* (2)



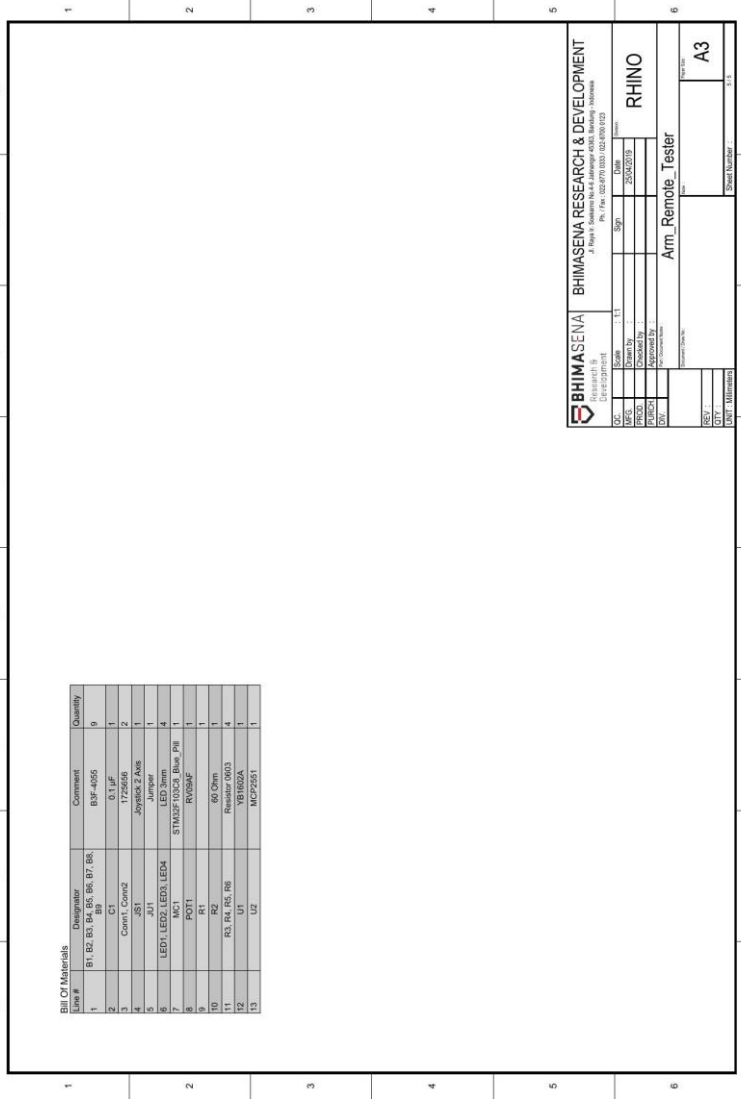
Lampiran 3 *Arm Remote Tester Draftman* (3)



Lampiran 4 *Arm Remote Tester Draftman* (4)



Lampiran 5 *Arm Remote Tester Draftman* (5)



Lampiran 6 File Program main.c

```
#include "stm32f103c8.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_adc.h"
#include "stm32f10x_can.h"
#include "stm32f10x_tim.h"
#include <CoOs.h>

#define STACK_SIZE_DEFAULT 512

OS_STK task1_stk[STACK_SIZE_DEFAULT];
OS_STK task2_stk[STACK_SIZE_DEFAULT];
OS_STK task3_stk[STACK_SIZE_DEFAULT];

/*
 * 0 = Emergency
 * 1 = Motor Selector
 * 2 = LCD Mode
 * 3 = Potensio Mode
 * 4 = Joystick Mode
 * 5 = Potensio Apply
 * 6 = Incremental+
 * 7 = Incremental-
 * 8 = Joystick Axis Z+
 * 9 = Joystick Axis Z-
 */
uint8_t digital_input[10] = {0};

/*
 * 0 = Joystick Axis X
 * 1 = Joystick Axis Y
 * 2 = Potensio
 */
uint16_t analog_input[3];

uint8_t m; //untuk memilih motor
int pot_pos;

int motor_data[8][3]; //data posisi, kecepatan dan status
float motor_data_cur[8]; //data arus motor
char motor_status[2][4] = {"N  ", "PE "}; //nama status motor

uint8_t req_data;

void TIM2_IRQHandler(void)
{
    if(TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
    {
    //Reset Timer
        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);

        req_data = 1;
    }
}

void USB_LP_CAN1_RX0_IRQHandler(void)
{
    CanRxMsg RxMessage;

    CAN_Receive(CAN1, CAN_FIFO0, &RxMessage);

    uint8_t source = RxMessage.ExtId & 0x0F;
    uint8_t object = (RxMessage.ExtId >> 16) & 0x0F;

    int16_t datapos = ((uint16_t)RxMessage.Data[0]) << 8 | ((uint16_t)RxMessage.Data[1]);
    int16_t dataspd = ((uint16_t)RxMessage.Data[2]) << 8 | ((uint16_t)RxMessage.Data[3]);
    int16_t datacur = ((uint16_t)RxMessage.Data[4]) << 8 | ((uint16_t)RxMessage.Data[5]);
```

```
    switch(source)
    {
    case 2:
        motor_data[0][0] = datapos/100;
        motor_data[0][1] = dataspd;
        motor_data_cur[0] = datacur;
        break;

    case 3:
        motor_data[1][0] = datapos/100;
        motor_data[1][1] = dataspd;
        motor_data_cur[1] = datacur;
        break;

    case 4:
        motor_data[2][0] = datapos/100;
        motor_data[2][1] = dataspd;
        motor_data_cur[2] = datacur;
        break;

    case 5:
        motor_data[3][0] = datapos/100;
        motor_data[3][1] = dataspd;
        motor_data_cur[3] = datacur;
        break;

    case 6:
        switch(object)
        {
        case 4:
            motor_data[4][0] = datapos/100;
            motor_data[4][1] = dataspd;
            motor_data_cur[4] = datacur;
            break;

        case 9:
            motor_data[6][0] = datapos/100;
            motor_data[6][1] = dataspd;
            motor_data_cur[6] = datacur;
            break;
        }
        break;

    case 7:
        switch(object)
        {
        case 4:
            motor_data[5][0] = datapos/100;
            motor_data[5][1] = dataspd;
            motor_data_cur[5] = datacur;
            break;

        case 9:
            motor_data[7][0] = datapos/100;
            motor_data[7][1] = dataspd;
            motor_data_cur[7] = datacur;
            break;
        }
        break;
    }
}

/*
 * Task 1 : Tugas yang digunakan untuk menampilkan data ke LCD
 */
void task1 (void* pdata)
{
    char motor_name[8][3] = {"T ", "S ", "E ", "W ", "GG", "GR", "CP", "CT"};

    while(1)
    {
        pot_pos = ((analog_input[2]*convesion_data1) - 180);
        uint16_t pot_led = (analog_input[2]*convesion_data2);
```

```
//Emergency Indikator
    if (digital_input[0] == 0)
    {
        LCD_Write(2, 1 , " ");
        LCD_Data(2, 3 , 0xFF);
    }
    else if (digital_input[0] == 1)
    {
        LCD_Write(2, 1 , "!");
        LCD_Data(2, 3 , 0xFF);
    }
    else if (digital_input[0] > 1)
    {
        digital_input[0] = 0;
    }

//Informasi Motor
    if ((digital_input[1] < 8) && (m < 8))
    {
        LCD_Data(1, 3, 0xFF);

        LCD_Write(2, 2, "  ");
        LCD_Data(2, 3, 0xFF);

    //Nama Motor
        if (digital_input[1] == m)
        {
            LCD_Write(1, 1 , motor_name[m]);

            switch (digital_input[2])
            {
        //Mode 1 LCD
            case 0:
            //Posisi
                if (motor_data[m][0] < 0)
                {
                    if (motor_data[m][0] > -10)
                    {
                        LCD_Write(1, 4, "  ");
                        LCD_WriteInt(1, 6, motor_data[m][0]);
                    }
                    else if ((motor_data[m][0] <= -10) & (motor_data[m][0] > -100))
                    {
                        LCD_Write(1, 4, " ");
                        LCD_WriteInt(1, 5, motor_data[m][0]);
                    }
                    else if (motor_data[m][0] <= -100)
                    {
                        LCD_WriteInt(1, 4, motor_data[m][0]);
                    }
                }
                else
                {
                    if ((motor_data[m][0] >= 0) & (motor_data[m][0] < 10))
                    {
                        LCD_Write(1, 4, "   ");
                        LCD_WriteInt(1, 7, motor_data[m][0]);
                    }
                    else if ((motor_data[m][0] >= 10) & (motor_data[m][0] < 100))
                    {
                        LCD_Write(1, 4, "  ");
                        LCD_WriteInt(1, 6, motor_data[m][0]);
                    }
                    else if (motor_data[m][0] >= 100)
                    {
                        LCD_Write(1, 4, " ");
                        LCD_WriteInt(1, 5, motor_data[m][0]);
                    }
                }

                LCD_Data(1, 8, 0xDF);
                LCD_Write(1, 9, " ");
                LCD_Data(1, 10, 0xFF);
```

```
    //Kecepatan
        if (motor_data[m][1] < 0)
        {
            motor_data[m][1] = motor_data[m][1] * -1;
        }

        if ((motor_data[m][1] >= 0) & (motor_data[m][1] < 10))
        {
            LCD_Write(2, 4, "    ");
            LCD_WriteInt(2, 8, motor_data[m][1]);
        }
        else if ((motor_data[m][1] >= 10) & (motor_data[m][1] < 100))
        {
            LCD_Write(2, 4, "   ");
            LCD_WriteInt(2, 7, motor_data[m][1]);
        }
        else if ((motor_data[m][1] >= 100) & (motor_data[m][1] < 1000))
        {
            LCD_Write(2, 4, "  ");
            LCD_WriteInt(2, 6, motor_data[m][1]);
        }
        else if ((motor_data[m][1] >= 1000) & (motor_data[m][1] < 10000))
        {
            LCD_Write(2, 4, " ");
            LCD_WriteInt(2, 5, motor_data[m][1]);
        }
        else if (motor_data[m][1] >= 10000)
        {
            LCD_WriteInt(2, 4, motor_data[m][1]);
        }

        LCD_Write(2, 9, "rpm");
        LCD_Write(2, 12, " ");
        LCD_Data(2, 13, 0xFF);

    //Arus
        if (motor_data_cur[m] > 0)
        {
            LCD_Write(1, 11 , " ");
            LCD_WriteInt(1, 12, motor_data_cur[m]);
            LCD_Write(1, 16 , "A");
        }
        else
        {
            LCD_Write(1, 11 , " ");
            LCD_WriteInt(1, 12, 0);
            LCD_Write(1, 13 , ".");
            LCD_WriteInt(1, 14, 0);
            LCD_WriteInt(1, 15, 0);
            LCD_Write(1, 16 , "A");
        }

    //Status
        LCD_Write(2, 14 , " ");
        LCD_Write(2, 15 , motor_status[0]);
        break;

//Mode 2 LCD
    case 1:
        LCD_Write(1, 4, " ");
        LCD_Write(1, 6 , "SUDUT:");
        LCD_Data(1, 16, 0xDF);

        LCD_Write(2, 6 , "LED  : ");
        LCD_Write(2, 16, "%");

    //Indikator Potensio Mode
        switch (digital_input[3])
        {
        case 0:
            LCD_Write(2, 5 , " ");
            LCD_Data(1, 5 , 0x7E);
            break;
```

```
            case 1:
                LCD_Write(1, 5 , " ");
                LCD_Data(2, 5 , 0x7E);
                break;

            default:
                digital_input[3] = 0;
                break;
            }

        //Posisi
            if (pot_pos < 0)
            {
                if ((pot_pos < 0) & (pot_pos > -10))
                {
                    LCD_Write(1, 12 , "  ");
                    LCD_WriteInt(1, 14, pot_pos);
                }
                else if ((pot_pos <= -10) & (pot_pos > -100))
                {
                    LCD_Write(1, 12 , " ");
                    LCD_WriteInt(1, 13, pot_pos);
                }
                else if (pot_pos <= -100)
                {
                    LCD_WriteInt(1, 12, pot_pos);
                }
            }
            else
            {
                if ((pot_pos >= 0) & (pot_pos < 10))
                {
                    LCD_Write(1, 12 , "   ");
                    LCD_WriteInt(1, 15, pot_pos);
                }
                else if ((pot_pos >= 10) & (pot_pos < 100))
                {
                    LCD_Write(1, 12 , "  ");
                    LCD_WriteInt(1, 14, pot_pos);
                }
                else if (pot_pos >= 100)
                {
                    LCD_Write(1, 12 , " ");
                    LCD_WriteInt(1, 13, pot_pos);
                }
            }

        //Intensitas LED
            if ((pot_led >= 0) & (pot_led < 10))
            {
                LCD_Write(2, 13 , "  ");
                LCD_WriteInt(2, 15, pot_led);
            }
            else if ((pot_led >= 10) & (pot_led < 100))
            {
                LCD_Write(2, 13 , " ");
                LCD_WriteInt(2, 14, pot_led);
            }
            else if (pot_led >= 100)
            {
                LCD_WriteInt(2, 13, pot_led);
            }
            break;

    //Reset Mode 1 LCD
        default:
            digital_input[2] = 0;
            break;
        }
    }
}
-'--
```

```
        else
        {
            digital_input[1] = 0;
            m = 0;
        }

        GPIO_WriteBit(GPIOC,GPIO_Pin_13,Bit_RESET);
        CoTickDelay(10);
        GPIO_WriteBit(GPIOC,GPIO_Pin_13,Bit_SET);
        CoTickDelay(10);
    }
}

/*
 * Task 2 : Digunakan untuk mengolah input digital dan analog
 */
void task2 (void* pdata)
{
    uint8_t bs_prev[6]; //Button state previous (sebelum)
    uint8_t bs_curr[6]; //Button state currently (saat ini)

    while(1)
    {
    /*Emergency*/
        if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_12) == Bit_SET)
        {
            bs_prev[0] = Bit_SET;
        }
        else if ((bs_prev[0] == Bit_SET) && (bs_curr[0] == Bit_RESET))
        {
            digital_input[0] += 1;
            bs_prev[0] = bs_curr[0];
        }

    /*Motor_Selector*/
        if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_15) == Bit_SET)
        {
            bs_prev[1] = Bit_SET;
        }
        else if ((bs_prev[1] == Bit_SET) && (bs_curr[1] == Bit_RESET))
        {
            Delay(10);
            digital_input[1] += 1;
            m += 1;
            bs_prev[1] = bs_curr[1];
        }

    /*LCD_Mode*/
        if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_13) == Bit_SET)
        {
            bs_prev[2] = Bit_SET;
        }
        else if ((bs_prev[2] == Bit_SET) && (bs_curr[2] == Bit_RESET))
        {
            digital_input[2] += 1;
            bs_prev[2] = bs_curr[2];
        }

    /*Potensio_Mode*/
        if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_14) == Bit_SET)
        {
            bs_prev[3] = Bit_SET;
        }
        else if ((bs_prev[3] == Bit_SET) && (bs_curr[3] == Bit_RESET))
        {
            digital_input[3] += 1;
            bs_prev[3] = bs_curr[3];
        }
```

```
    /*Joystick_Mode*/
        if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_5) == Bit_SET)
        {
            bs_prev[4] = Bit_SET;
        }
        else if ((bs_prev[4] == Bit_SET) && (bs_curr[4] == Bit_RESET))
        {
            digital_input[4] += 1;
            bs_prev[4] = bs_curr[4];
        }

    /*Potensio_Apply*/
        if (GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_0) == Bit_SET)
        {
            bs_prev[5] = Bit_SET;
        }
        else if ((bs_prev[5] == Bit_SET) && (bs_curr[5] == Bit_RESET))
        {
            digital_input[5] += 1;
            bs_prev[5] = bs_curr[5];
        }

    /*Incremental+*/
        if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_6) == Bit_SET)
        {
            digital_input[6] = 1;
        }
        else if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_6) == Bit_RESET)
        {
            digital_input[6] = 0;
        }

    /*Incremental-*/
        if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_7) == Bit_SET)
        {
            digital_input[7] = 1;
        }
        else if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_7) == Bit_RESET)
        {
            digital_input[7] = 0;
        }

    /*Joystick_Axis_Z+*/
        if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0) == Bit_SET)
        {
            digital_input[8] = 1;
        }
        else if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0) == Bit_RESET)
        {
            digital_input[8] = 0;
        }

    /*Joystick_Axis_Z-*/
        if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1) == Bit_SET)
        {
            digital_input[9] = 1;
        }
        else if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1) == Bit_RESET)
        {
            digital_input[9] = 0;
        }

    /*Joystick_Axis_X*/
//      analog_input[0] = Analog_Input_Raw(ADC_Pin_2);

    /*Joystick_Axis_Y*/
//      analog_input[1] = Analog_Input_Raw(ADC_Pin_3);

    /*Potensio*/
        analog_input[2] = (4095 - Analog_Input_Raw(ADC_Pin_4));

        CoTickDelay(5);
    }
}
```

```
/*
 * Task 3 : - Digunakan untuk mengirimkan data ke arm dan membaca data arm
 */
void task3 (void* pdata)
{
    CanTxMsg TxMessage;

//Data kecepatan gerak
    uint8_t msb_speed = 1000 >> 8;
    uint8_t lsb_speed = 1000 & 0x00FF;

//Data incremental +1
    uint8_t msb_incremental_p1 = 100 >> 8;
    uint8_t lsb_incremental_p1 = 100 & 0x00FF;

//Data incremental -1
    int8_t msb_incremental_n1 = -100 >> 8;
    int8_t lsb_incremental_n1 = -100 & 0x00FF;

    while(1)
    {
    //Konversi data ADC ke data arm
        int16_t pot_data_pos = ((analog_input[2] * convesion_data1) - 180) * 100;
        int16_t pot_data_led = 5000 - (analog_input[2] * convesion_data3);

    //Data potensio sebagai sudut
        int8_t msb_position = pot_data_pos >> 8;
        int8_t lsb_position = pot_data_pos & 0x00FF;

    //Data potensio sebagai intensitas cahaya
        uint8_t msb_led = pot_data_led >> 8;
        uint8_t lsb_led = pot_data_led & 0x00FF;

    /*Emergency*/
        switch(digital_input[0])
        {
        case 0:
            GPIO_WriteBit(GPIOA,GPIO_Pin_8,Bit_SET);
            GPIO_WriteBit(GPIOA,GPIO_Pin_9,Bit_RESET);
            break;
        case 1:
            GPIO_WriteBit(GPIOA,GPIO_Pin_8,Bit_RESET);
            GPIO_WriteBit(GPIOA,GPIO_Pin_9,Bit_SET);
            break;
        default:
            digital_input[0] = 0;
            break;
        }

    /*Absolute_Input*/
        switch (digital_input[3])
        {
        case 0:
            if (digital_input[5] == 1)
            {
                TxMessage.IDE = CAN_ID_EXT;
                TxMessage.RTR = CAN_RTR_DATA;
                TxMessage.DLC = 5;
                TxMessage.Data[0] = 0;
                TxMessage.Data[1] = msb_position;
                TxMessage.Data[2] = lsb_position;
                TxMessage.Data[3] = msb_speed;
                TxMessage.Data[4] = lsb_speed;

                switch (digital_input[1])
                {
                case 0:
                    TxMessage.ExtId = 0x1A21BC21;
                    CAN_Transmit(CAN1, &TxMessage);
                    break;
```

```
            case 1:
                TxMessage.ExtId = 0x1A21BC31;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 2:
                TxMessage.ExtId = 0x1A21BC41;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 3:
                TxMessage.ExtId = 0x1A21BC51;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 4:
                TxMessage.ExtId = 0x1A21BC61;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 5:
                TxMessage.ExtId = 0x1A21BC71;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 6:
                TxMessage.ExtId = 0x1A22BC61;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 7:
                TxMessage.ExtId = 0x1A22BC71;
                CAN_Transmit(CAN1, &TxMessage);
                break;
            }
        }
        break;

    case 1:
        if (digital_input[5] == 1)
        {
            TxMessage.ExtId = 0x1A23BC71;
            TxMessage.IDE = CAN_ID_EXT;
            TxMessage.RTR = CAN_RTR_DATA;
            TxMessage.DLC = 2;
            TxMessage.Data[0] = msb_led;
            TxMessage.Data[1] = lsb_led;
            CAN_Transmit(CAN1, &TxMessage);
        }
        break;

    default:
        digital_input[3] = 0;
        break;
    }
    digital_input[5] = 0;

/*Incremental_Input_Positif*/
    while (digital_input[6] == 1)
    {
        TxMessage.IDE = CAN_ID_EXT;
        TxMessage.RTR = CAN_RTR_DATA;
        TxMessage.DLC = 5;
        TxMessage.Data[0] = 1;
        TxMessage.Data[1] = msb_incremental_p1;
        TxMessage.Data[2] = lsb_incremental_p1;
        TxMessage.Data[3] = msb_speed;
        TxMessage.Data[4] = lsb_speed;
```

```
        switch (digital_input[1])
        {
        case 0:
            TxMessage.ExtId = 0x1A21BC21;
            CAN_Transmit(CAN1, &TxMessage);
            break;

        case 1:
            TxMessage.ExtId = 0x1A21BC31;
            CAN_Transmit(CAN1, &TxMessage);
            break;

        case 2:
            TxMessage.ExtId = 0x1A21BC41;
            CAN_Transmit(CAN1, &TxMessage);
            break;

        case 3:
            TxMessage.ExtId = 0x1A21BC51;
            CAN_Transmit(CAN1, &TxMessage);
            break;

        case 4:
            TxMessage.ExtId = 0x1A21BC61;
            CAN_Transmit(CAN1, &TxMessage);
            break;

        case 5:
            TxMessage.ExtId = 0x1A21BC71;
            CAN_Transmit(CAN1, &TxMessage);
            break;

        case 6:
            TxMessage.ExtId = 0x1A22BC61;
            CAN_Transmit(CAN1, &TxMessage);
            break;

        case 7:
            TxMessage.ExtId = 0x1A22BC71;
            CAN_Transmit(CAN1, &TxMessage);
            break;
        }

        TxMessage.ExtId = 0x1A04BC01;
        TxMessage.IDE = CAN_ID_EXT;
        TxMessage.RTR = CAN_RTR_DATA;
        TxMessage.DLC = 0;
        CAN_Transmit(CAN1, &TxMessage);
    }

/*Incremental_Input_Negatif*/
    while (digital_input[7] == 1)
    {
        TxMessage.IDE = CAN_ID_EXT;
        TxMessage.RTR = CAN_RTR_DATA;
        TxMessage.DLC = 5;
        TxMessage.Data[0] = 1;
        TxMessage.Data[1] = msb_incremental_n1;
        TxMessage.Data[2] = lsb_incremental_n1;
        TxMessage.Data[3] = msb_speed;
        TxMessage.Data[4] = lsb_speed;

        switch (digital_input[1])
        {
        case 0:
            TxMessage.ExtId = 0x1A21BC21;
            CAN_Transmit(CAN1, &TxMessage);
            break;

        case 1:
            TxMessage.ExtId = 0x1A21BC31;
            CAN_Transmit(CAN1, &TxMessage);
            break;
```

```
            case 2:
                TxMessage.ExtId = 0x1A21BC41;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 3:
                TxMessage.ExtId = 0x1A21BC51;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 4:
                TxMessage.ExtId = 0x1A21BC61;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 5:
                TxMessage.ExtId = 0x1A21BC71;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 6:
                TxMessage.ExtId = 0x1A22BC61;
                CAN_Transmit(CAN1, &TxMessage);
                break;

            case 7:
                TxMessage.ExtId = 0x1A22BC71;
                CAN_Transmit(CAN1, &TxMessage);
                break;
            }

            TxMessage.ExtId = 0x1A04BC01;
            TxMessage.IDE = CAN_ID_EXT;
            TxMessage.RTR = CAN_RTR_DATA;
            TxMessage.DLC = 0;
            CAN_Transmit(CAN1, &TxMessage);
        }

    /*Joystick_Mode*/
        switch(digital_input[4])
        {
        case 0:
            GPIO_WriteBit(GPIOA,GPIO_Pin_10,Bit_SET);
            GPIO_WriteBit(GPIOB,GPIO_Pin_5,Bit_RESET);
            break;
        case 1:
            GPIO_WriteBit(GPIOA,GPIO_Pin_10,Bit_RESET);
            GPIO_WriteBit(GPIOB,GPIO_Pin_5,Bit_SET);
            break;
        default:
            digital_input[4] = 0;
            break;
        }

    /*Request Data*/
        if (digital_input[2] == 0)
        {
            if (req_data == 1)
            {
                req_data = 0;

                TxMessage.ExtId = 0x1A04BC01;
                TxMessage.IDE = CAN_ID_EXT;
                TxMessage.RTR = CAN_RTR_DATA;
                TxMessage.DLC = 0;
                CAN_Transmit(CAN1, &TxMessage);
            }
        }
    }
}
```

```
int main(void)
{
    Digital_Configuration();
    Analog_Configuration();
    LCD_Configuration();
    CAN_Configuration();
    init_CANControl();
    TIM_Configuration();

    CoInitOS();
    CoCreateTask(task1,0,0,&task1_stk[STACK_SIZE_DEFAULT-1],STACK_SIZE_DEFAULT);
    CoCreateTask(task2,0,1,&task2_stk[STACK_SIZE_DEFAULT-1],STACK_SIZE_DEFAULT);
    CoCreateTask(task3,0,2,&task3_stk[STACK_SIZE_DEFAULT-1],STACK_SIZE_DEFAULT);
    CoStartOS();
    while(1);
}
```

Lampiran 7 File Program stm32f103c8t6.h

```
#include "misc.h"

#define ADC_Pin_2   0x00
#define ADC_Pin_3   0x01
#define ADC_Pin_4   0x02

#define Address 0x27

#define convesion_data1 0.0879120879120879f /* 360/4095 */
#define convesion_data2 0.0244200244200244f /* 100/4095 */
#define convesion_data3 1.221001221001221f  /* 5000/4095 */

u16 ADC_ConvertedValue[3];
u16 ADCBuffer[3];

void Digital_Configuration(void);

void Analog_Configuration(void);
u16 Analog_Input_Raw(u8 ADC_Channel);

void LCD_Configuration(void);
void LCD_Command(u8 command);
void LCD_Data(u8 row, u8 column, char data);
void LCD_Write(u8 row, u8 column, char *text);
void LCD_WriteInt(u8 row, u8 column, int number);

void init_CANControl(void);
void CAN_Configuration(void);

void Delay(u32 time);
void TIM_Configuration(void);
```

Lampiran 8 File Program stm32f103c8t6.c

```
#include "stm32f10x.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_adc.h"
#include "stm32f10x_i2c.h"
#include "stm32f10x_can.h"
#include "stm32f10x_dma.h"
#include "stm32f10x_tim.h"
#include "stm32f103c8.h"

/*
 * Konfigurasi input dan output digital, yang berupa tombol dan led
 */
void Digital_Configuration(void)
{
    GPIO_InitTypeDef GPIO_InitStructure_Input;
    GPIO_InitTypeDef GPIO_InitStructure_Output;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB | RCC_APB2Periph_GPIOC, ENABLE);

    /*Digital_Input_Configuration_Pin_A*/
    GPIO_InitStructure_Input.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_5 | GPIO_Pin_6 |
                                        GPIO_Pin_7;
    GPIO_InitStructure_Input.GPIO_Mode = GPIO_Mode_IPD;
    GPIO_Init(GPIOA, &GPIO_InitStructure_Input);

    /*Digital_Input_Configuration_Pin_B*/
    GPIO_InitStructure_Input.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14 |
                                        GPIO_Pin_15;
    GPIO_InitStructure_Input.GPIO_Mode = GPIO_Mode_IPD;
    GPIO_Init(GPIOB, &GPIO_InitStructure_Input);

    /*Digital_Output_Configuration_Pin_A*/
    GPIO_InitStructure_Output.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure_Output.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure_Output.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_9 | GPIO_Pin_10;
    GPIO_Init(GPIOA, &GPIO_InitStructure_Output);

    /*Digital_Output_Configuration_Pin_B*/
    GPIO_InitStructure_Output.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure_Output.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure_Output.GPIO_Pin = GPIO_Pin_5;
    GPIO_Init(GPIOB, &GPIO_InitStructure_Output);

    /*Digital_Output_Configuration_Pin_C*/
    GPIO_InitStructure_Output.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure_Output.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure_Output.GPIO_Pin = GPIO_Pin_13;
    GPIO_Init(GPIOC, &GPIO_InitStructure_Output);
}

/*
 * Konfigurasi input analog dengan DMA yang berupa potensio dan joystick
 */
void Analog_Configuration(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    ADC_InitTypeDef ADC_InitStructure;
    DMA_InitTypeDef DMA_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_ADC1, ENABLE);
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);

    /*Analog_Input_Configuration_Pin_A*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
```

```
    /*DMA1 channel1 configuration*/
    DMA_DeInit(DMA1_Channel1);
    DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)0x4001244C;
    DMA_InitStructure.DMA_MemoryBaseAddr = (u32)ADCBuffer;
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
    DMA_InitStructure.DMA_BufferSize = 3;
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
    DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_Init(DMA1_Channel1, &DMA_InitStructure);

    /*Enable DMA1 channel1*/
    DMA_Cmd(DMA1_Channel1, ENABLE);

    /*ADC_Configuration*/
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode = ENABLE;
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = 3;
    ADC_Init(ADC1, &ADC_InitStructure);

    ADC_RegularChannelConfig(ADC1, ADC_Channel_2, 1, ADC_SampleTime_7Cycles5);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_3, 2, ADC_SampleTime_13Cycles5);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_4, 3, ADC_SampleTime_28Cycles5);

    /*Enable ADC1 DMA*/
    ADC_DMACmd(ADC1, ENABLE);

    /*Enable ADC1*/
    ADC_Cmd(ADC1, ENABLE);
    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1));
    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1));
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}

/*
 * Untuk menghasilkan data asli dari pembacaaan masukan analog
 * Parameter ADC_Pin : dapat diisi dengan ADC_Pin_2, ADC_Pin_3 atau ADC_Pin_4.
 */
u16 Analog_Input_Raw(u8 ADC_Pin)
{
    for (int i = 0; i < 3; ++i)
    {
        ADC_ConvertedValue[i] = ADCBuffer[i];
    }

    if (ADC_Pin == ADC_Pin_2)
    {
        return ADC_ConvertedValue[0];
    }
    else if (ADC_Pin == ADC_Pin_3)
    {
        return ADC_ConvertedValue[1];
    }
    else if (ADC_Pin == ADC_Pin_4)
    {
        return ADC_ConvertedValue[2];
    }
}
```

```
void init_CANControl(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;

    /* Enable CAN1 RX0 interrupt IRQ channel */
    NVIC_InitStructure.NVIC_IRQChannel = USB_LP_CAN1_RX0_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);

    /* CAN FIFO0 message pending interrupt enable */
    CAN_ITConfig(CAN1, CAN_IT_FMP0, ENABLE);
}

/*
 * Konfigurasi LCD yang menggunakan komunikasi I2C Bus
 */
void LCD_Configuration(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;
    I2C_InitTypeDef  I2C_InitStructure;

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);

    /*LCD_Configuration*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_OD;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    /*I2C_Configuration*/
    I2C_InitStructure.I2C_Mode = I2C_Mode_I2C;
    I2C_InitStructure.I2C_DutyCycle = I2C_DutyCycle_2;
    I2C_InitStructure.I2C_OwnAddress1 = 0x00;
    I2C_InitStructure.I2C_Ack = I2C_Ack_Disable;
    I2C_InitStructure.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit;
    I2C_InitStructure.I2C_ClockSpeed = 100000;
    I2C_Init(I2C1, &I2C_InitStructure);

    I2C_Cmd(I2C1, ENABLE);

    Delay(15000);

    Beginning:

    /*LCD_Start_Initial*/
    I2C_GenerateSTART(I2C1,ENABLE);
    u16 TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT) && TimeOut--);

    I2C_Send7bitAddress(I2C1, Address << 1, I2C_Direction_Transmitter);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED) && TimeOut--);

    I2C_AcknowledgeConfig(I2C1, ENABLE);

    /*LCD_Initial_1*/
    I2C_SendData(I2C1, 0x30 | 0x0C);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
    I2C_SendData(I2C1, 0x30 | 0x08);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

    /*LCD_Initial_2*/
    I2C_SendData(I2C1, 0x30 | 0x0C);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
    I2C_SendData(I2C1, 0x30 | 0x08);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
```

```
/*LCD_Initial_3*/
I2C_SendData(I2C1, 0x30 | 0x0C);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x30 | 0x08);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

/*LCD_Initial_4*/
I2C_SendData(I2C1, 0x20 | 0x0C);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x20 | 0x08);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

/*LCD_Function_Set*/
I2C_SendData(I2C1, 0x20 | 0x0C);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x20 | 0x08);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x80 | 0x0C);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x80 | 0x08);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

/*LCD_Display_On/Off_Control*/
I2C_SendData(I2C1, 0x00 | 0x0C);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x00 | 0x08);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0xC0 | 0x0C);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0xC0 | 0x08);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

/*LCD_Clear_Display*/
I2C_SendData(I2C1, 0x00 | 0x0C);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x00 | 0x08);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x10 | 0x0C);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x10 | 0x08);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

/*LCD_Entry_Mode_Set*/
I2C_SendData(I2C1, 0x00 | 0x0C);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x00 | 0x08);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x60 | 0x0C);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
I2C_SendData(I2C1, 0x60 | 0x08);
TimeOut = 0xFFFF;
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
```

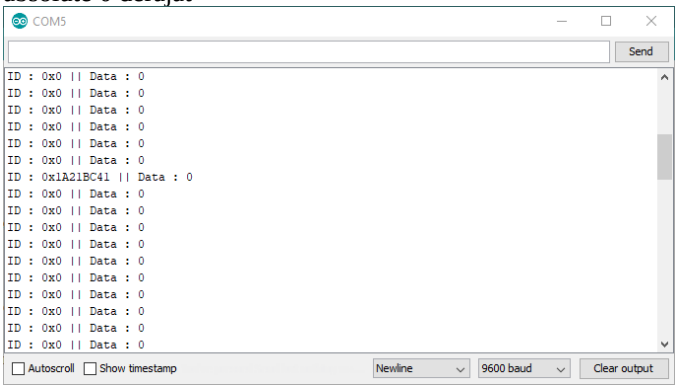
```
    I2C_AcknowledgeConfig(I2C1, DISABLE);

    I2C_GenerateSTOP(I2C1, ENABLE);

    if (TimeOut == 0)
    {
        goto Beginning;
    }
}
/*
 * Digunakan untuk memberikan perintah pada LCD
 * Parameter command : dapat diisi dengan perintah-perintah untuk LCD yang ada pada datasheet.
 */
void LCD_Command(u8 command)
{
    u8 high_bit = command & 0xF0;
    u8 low_bit = (command & 0x0F) << 4;

    Beginning:

    I2C_GenerateSTART(I2C1,ENABLE);
    u16 TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT) && TimeOut--);

    I2C_Send7bitAddress(I2C1, Address << 1, I2C_Direction_Transmitter);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED) && TimeOut--);

    I2C_AcknowledgeConfig(I2C1, ENABLE);

    I2C_SendData(I2C1, high_bit | 0x0C);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
    I2C_SendData(I2C1, high_bit | 0x08);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

    I2C_SendData(I2C1, low_bit | 0x0C);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
    I2C_SendData(I2C1, low_bit | 0x08);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

    I2C_AcknowledgeConfig(I2C1, DISABLE);

    I2C_GenerateSTOP(I2C1, ENABLE);

    if (TimeOut == 0)
    {
        goto Beginning;
    }
}

/*
 * Digunakan untuk menuliskan data pada LCD
 * Parameter row : dapat diisi dengan 1 atau 2, sesuai baris penulisan data pada LCD 16x2.
 * Parameter column : dapat diisi dengan 1 sampai 16, sesuai kolom penulisan data pada LCD 16x2.
 * Parameter data : dapat diisi dengan data yang akan dituliskan di LCD.
 */
void LCD_Data(u8 row, u8 column, char data)
{
    u8 high_bit = data & 0xF0;
    u8 low_bit = (data & 0x0F) << 4;

    Beginning:

    if (row == 1)
    {
        LCD_Command(0x80 + (column - 1));
    }
```

```
    else if(row == 2)
    {
        LCD_Command(0xC0 + (column - 1));
    }

    I2C_GenerateSTART(I2C1,ENABLE);
    u16 TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT) && TimeOut--);

    I2C_Send7bitAddress(I2C1, Address << 1, I2C_Direction_Transmitter);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED) && TimeOut--);

    I2C_SendData(I2C1, 0x08);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

    I2C_AcknowledgeConfig(I2C1, ENABLE);

    I2C_SendData(I2C1, high_bit | 0x0D);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
    I2C_SendData(I2C1, high_bit | 0x09);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

    I2C_SendData(I2C1, low_bit | 0x0D);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);
    I2C_SendData(I2C1, low_bit | 0x09);
    TimeOut = 0xFFFF;
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED) && TimeOut--);

    I2C_AcknowledgeConfig(I2C1, DISABLE);

    I2C_GenerateSTOP(I2C1, ENABLE);

    if (TimeOut == 0)
    {
        goto Beginning;
    }
}

/*
 * Digunakan untuk menulis karakter pada LCD
 * Parameter row : dapat diisi dengan 1 atau 2, sesuai baris penulisan karakter pada LCD 16x2.
 * Parameter column : dapat diisi dengan 1 sampai 16, sesuai kolom penulisan karakter pada LCD 16x2.
 * Parameter text : dapat diisi dengan tulisan karakter yang ingin ditampilkan pada LCD.
 */
void LCD_Write(u8 row, u8 column, char *text)
{
    while(*text)
    {
        LCD_Data(row, column++, *text++);
    }
}

/*
 * Digunakan untuk menulis nilai/angka pada LCD
 * Parameter row : dapat diisi dengan 1 atau 2, sesuai baris penulisan karakter pada LCD 16x2.
 * Parameter column : dapat diisi dengan 1 sampai 16, sesuai kolom penulisan karakter pada LCD 16x2.
 * Parameter number : dapat diisi dengan variabel nilai yang akan ditulis pada LCD.
 */
void LCD_WriteInt(u8 row, u8 column, int number)
{
    char buffer[12];
    char tmp[10];
    u8 i=0, j=0;
    if( number < 0 ){
        buffer[0]='-';
        number = -1 * number;
        i++;
```

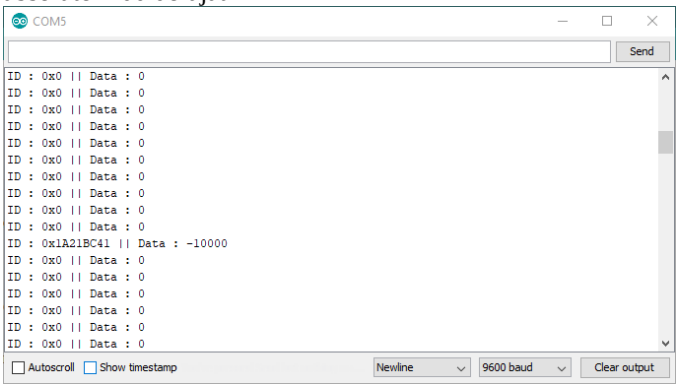
```
    } else
        buffer[0] = 0;
    do{
        tmp[j++] = number % 10;
        number = number / 10;
    } while( number != 0 );
    while(j--){
        buffer[i++] = tmp[j] + 48;
    }
    buffer[i] = 0;
    LCD_Write(row, column, buffer);
}

/*
 * Konfigurasi CAN Bus untuk mengirimkan dan membaca data arm.
 */
void CAN_Configuration(void)
{
    GPIO_InitTypeDef GPIO_CAN_InitStructure;
    CAN_InitTypeDef CAN_InitStructure;
    CAN_FilterInitTypeDef CAN_FilterInitStructure;

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_CAN1, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

    GPIO_CAN_InitStructure.GPIO_Pin = GPIO_Pin_11;
    GPIO_CAN_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_CAN_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_CAN_InitStructure);

    GPIO_CAN_InitStructure.GPIO_Pin = GPIO_Pin_12;
    GPIO_CAN_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_CAN_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_CAN_InitStructure);

    /* CAN register init */
    CAN_DeInit(CAN1);
    CAN_StructInit(&CAN_InitStructure);

    /* CAN cell init */
    CAN_InitStructure.CAN_TTCM = DISABLE;
    CAN_InitStructure.CAN_ABOM = DISABLE;
    CAN_InitStructure.CAN_AWUM = DISABLE;
    CAN_InitStructure.CAN_NART = DISABLE;
    CAN_InitStructure.CAN_RFLM = DISABLE;
    CAN_InitStructure.CAN_TXFP = DISABLE;
    CAN_InitStructure.CAN_Mode = CAN_Mode_Normal;
    CAN_InitStructure.CAN_SJW = CAN_SJW_1tq;
    CAN_InitStructure.CAN_BS1 = CAN_BS1_8tq;
    CAN_InitStructure.CAN_BS2 = CAN_BS2_3tq;
    CAN_InitStructure.CAN_Prescaler = 3;

     /* CAN filter init */
    CAN_FilterInitStructure.CAN_FilterNumber = 0;
    CAN_FilterInitStructure.CAN_FilterMode = CAN_FilterMode_IdMask;
    CAN_FilterInitStructure.CAN_FilterScale = CAN_FilterScale_32bit;
    CAN_FilterInitStructure.CAN_FilterIdHigh = 0x0A00 << 3;
    CAN_FilterInitStructure.CAN_FilterIdLow = 0xBC10 << 3;
    CAN_FilterInitStructure.CAN_FilterMaskIdHigh = 0x0F00 << 3;
    CAN_FilterInitStructure.CAN_FilterMaskIdLow = 0xFFF0 << 3;
    CAN_FilterInitStructure.CAN_FilterFIFOAssignment = CAN_FIFO0;
    CAN_FilterInitStructure.CAN_FilterActivation = ENABLE;
    CAN_FilterInit(&CAN_FilterInitStructure);

    CAN_Init(CAN1, &CAN_InitStructure);
}
```

```
/*
 * Digunakan untuk melakukan jeda pada program.
 * Parameter time : dapat diisi dengan berapa lama jeda yang digunakan.
 */
void Delay(u32 time)
{
    while (time)
    {
        time--;
    }
}


void TIM_Configuration(void)
{
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

    NVIC_InitTypeDef NVIC_InitStructure;
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;

//Enable TIM2 IRQ
    NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);

//Atur timer 2 interrupt jadi 500ms
    TIM_TimeBaseStructure.TIM_Prescaler = 287;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseStructure.TIM_Period = 62499;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);

/* TIM IT enable */
    TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);

/* TIM2 enable counter */
    TIM_Cmd(TIM2, ENABLE);
}
```

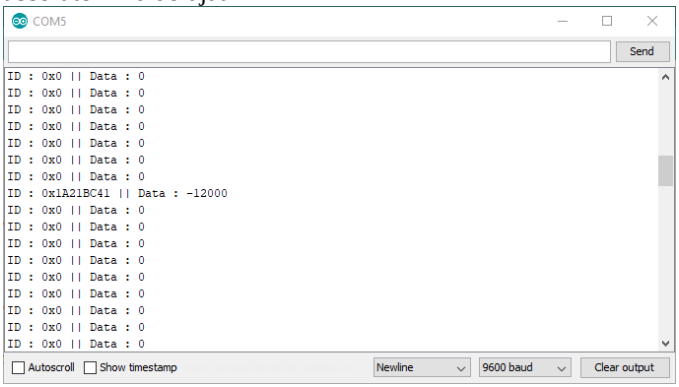
Lampiran 9 Hasil data yang diterima *joint* elbow dengan masukan absolute 0 derajat



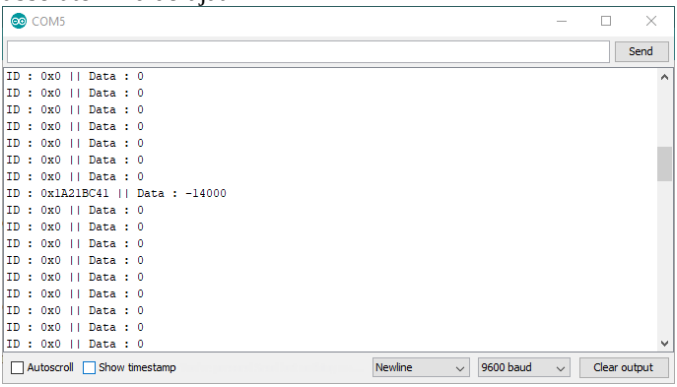
Lampiran 10 Hasil data yang diterima *joint* elbow dengan masukan absolute -100 derajat



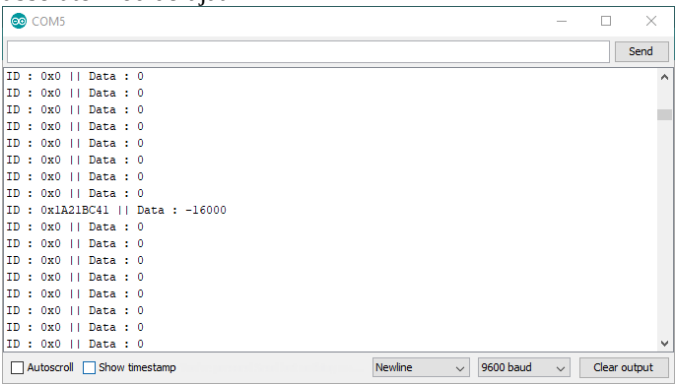
Lampiran 11 Hasil data yang diterima *joint* elbow dengan masukan absolute -120 derajat



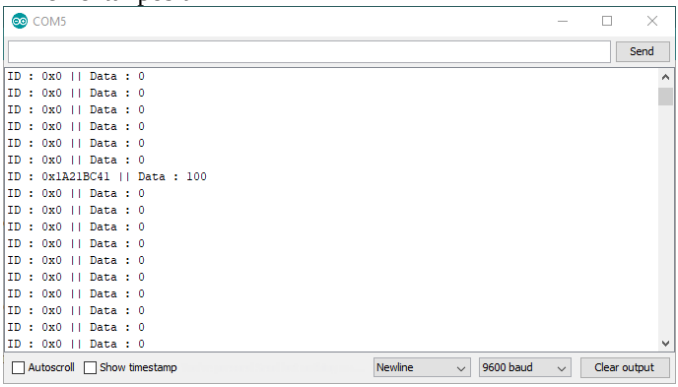
Lampiran 12 Hasil data yang diterima *joint* elbow dengan masukan absolute -140 derajat



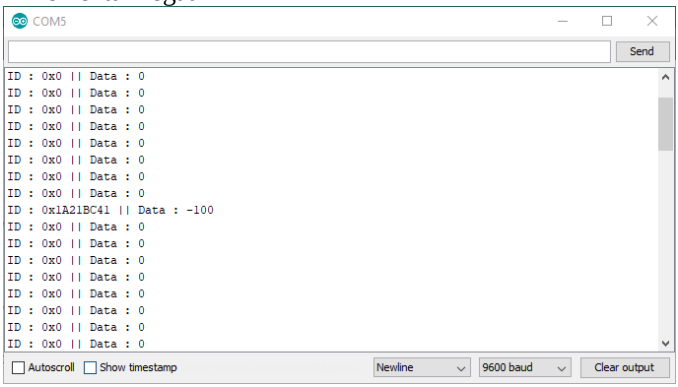
Lampiran 13 Hasil data yang diterima *joint* elbow dengan masukan absolute -160 derajat



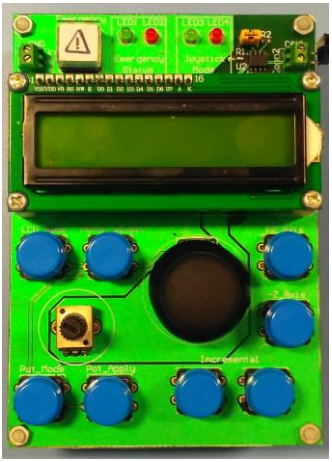
Lampiran 14 Hasil data yang diterima *joint* elbow dengan masukan inkremental positif



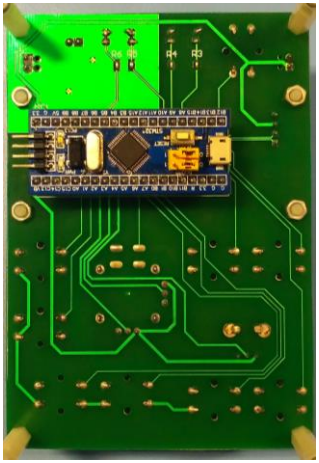
Lampiran 15 Hasil data yang diterima *joint* elbow dengan masukan inkremental negatif



Lampiran 16 Hasil perancangan alat *Arm Remote Tester* (tampak depan)



Lampiran 17 Hasil perancangan alat *Arm Remote Tester* (taampak belakang)



Halaman ini sengaja dikosongkan

RIWAYAT PENULIS

Andi Octriyanto Bahar lahir di kota Surabaya pada tanggal 4 Oktober 1996. Penulis adalah anak ketiga dari tiga bersaudara. Penulis menempuh pendidikan dasar di SD Siti Aminah Surabaya pada tahun 2003-2009, kemudian SMP Negeri 28 Surabaya pada tahun 2009-2012 dan SMK Negeri 1 Surabaya kejuruan Multimedia pada tahun 2012-2015. Pada tahun 2016 penulis diterima sebagai mahasiswa di jurusan D3 Teknik Elektro, Institut Teknologi Sepuluh Nopember (ITS), Surabaya. Semasa kuliah, sejak semester ke-3 penulis mulai aktif sebagai anggota *Automation Computer System Laboratory* (ACSL).

Email : andioctriyanto@gmail.com