UNIVERSITAS INDONESIA

PENGEMBANGAN PERANGKAT LUNAK MESIN *RAPID PROTOTYPING* DENGAN METODE FDM (*FUSED DEPOSITIONING MODELLING*)

SKRIPSI

Andry Sulaiman
0606072925

FAKULTAS TEKNIK
DEPARTEMEN TEKNIK MESIN
DEPOK
DESEMBER 2010

UNIVERSITAS INDONESIA

PENGEMBANGAN PERANGKAT LUNAK MESIN *RAPID PROTOTYPING* DENGAN METODE FDM (*FUSED DEPOSITIONING MODELLING*)

SKRIPSI

Andry Sulaiman
0606072925

FAKULTAS TEKNIK
DEPARTEMEN TEKNIK MESIN
DEPOK
DESEMBER 2010

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar

Nama	**: Andry Sulaiman**
NPM	**: 0606072925**
Tanda Tangan	**:** 
Tanggal	**: 6 Januari 2011**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama : Andry Sulaiman

NPM : 0606072925

Program Studi : Teknik Mesin

Judul Skripsi : PENGEMBANGAN PERANGKAT LUNAK MESIN *RAPID PROTOTYPING* DENGAN METODE FDM (*FUSED DEPOSITIONING METHOD*)

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian dari persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi, Teknik Mesin Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

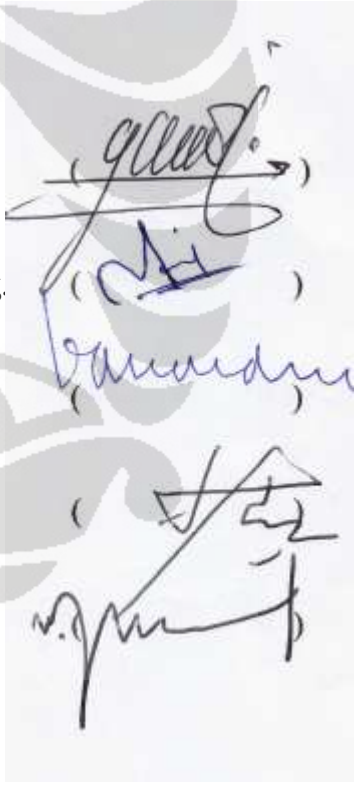
Pembimbing : Dr. Ir. Gandjar Kiswanto, M.Eng

Penguji : Dr. Ir. Ario Sunar Baskoro, ST., MT., M.Eng.

Penguji : Dr. Ir. Danardono AS.

Penguji : Ir. Hendri DS Budiono, M.Eng.

Penguji : Ir. Henky S. Nugroho, MT.



Ditetapkan di : Depok

Tanggal : 6 Januari 2011

ABSTRAK

Nama : Andry Sulaiman

Program Studi : Teknik Mesin

Judul :Pengembangan Perangkat Lunak Mesin Rapid Prototyping Dengan Metode FDM (*Fused Deposition Modelling*)

Penelitian ini bertujuan untuk merancang sebuah sistem pengendali berupa perangkat lunak yang mampu mengontrol sebuah mesin *rapid prototyping* dengan 3 derajat kebebasan. Dalam penelitian ini digunakan software atau *compiler* Turbo-C dan Codevision AVR untuk meng-*compile* perangkat lunak yang sudah dirancang. Alogaritma perangkat lunak secara umum adalah PC mengirim data koordinat ke mikrokontroler kemudian mikrokontroler menggerakan motor stepper pada masing-masing sumbu sesuai data koordinat yang sudah diterima. Motor penggerak yang di kontrol oleh mikrokontroler berfungsi untuk mengendalikan *nozzle* yang digunakan untuk meng-*extrude* material, sehingga *nozzle* dapat bergerak dan membentuk pola sesuai koordinat yang di berikan. Untuk memudahkan pengendalian dirancang program user interface sehingga pengguna dapat dengan mudah memberi perintah kepada mesin. Pilihan perintah yang dapat di pilih oleh pengguna adalah mode posisi awal dan mode pengiriman data. Pilihan perintah ini yang kemudian akan menentukan apa yang dilakukan oleh mikrokontroler. Perangkat lunak yang didesain dapat mengirim dan merima data dengan cepat dan memiliki keakurasian sebesar 0.5-1 mm. Dari hasil pengujian didapat *error* akibat perubahan nilai *float* ke *integer* sebesar 0.42%.

Kata kunci:

Rapid prototyping, perangkat lunak

ABSTRACT

Name : Andry Sulaiman

Study Program : Mechanical Engineering

Title : Software Development for Rapid Prototyping Machine with FDM method (Fused Deposition Modelling)

This research is aimed to design and develop two controlling softwares for a rapid prototyping machine with three degree of freedom. The first one is installed in the PC to send data to microcontrollers, the other one is coded in microcontrollers for receiving data from PC and execute it to stepper motor driver. The general software alogarithm is PC sent data coordinate to microcontrollers and then microcontrollers move stepper motor in each axis according to data coordinate that has been received. The driving motor that is controlled by microcontrollers is used to control nozzle that is used to extrude material, so nozzle can move and create a pattern according to the given coordinate. To simplifies the program a user interface is designed, so the user can easily give an order to the machine. The order that can be chosen by the user is default position mode and sending data. This given order that will determind what the machine will do. The software is able to receive and sending data instantly with 0.5-1 mm accuracy. The testing result show the error from float to integer conversion is 0.42%.

Keywords:

Rapid prototyping, software

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Mesin pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, penyusunan skripsi ini sangatlah sulit bagi saya. Oleh karena itu, saya mengucapkan terima kasih kepada:

1) Orang tua dan paman saya yang telah memberikan dukungan moril dan materiil;
2) Dr. Ir. Gandjar Kiswanto, M.Eng selaku dosen pembimbing yang telah meluangkan waktunya untuk menyemangati saya dan menginspirasi saya dalam pengerjaan skripsi ini;
3) Dr. Ir.Harinaldi, M.Eng selaku kepala Departemen Teknik Mesin;
4) Teman-teman tim skripsi dan Lab Manufaktur, Rendi Kurniawan, L. Sriyanto, dan Hadi Maryadi, Achmad Saroni, Jediel Billy R, Teguh Santoso, Hendra P.S. S.T, Erwin Nugraha Bayuaji, yang telah menemani dan membantu penulis dalam mengerjakan skripsi ini;
5) Alvin Pradipta yang telah membantu pembuatan program dalam skripsi ini;
6) Pak Martinus Herlinsen, yang telah membantu pembuatan mesin *Rapid Prototyping;*
7) Teman-teman yang telah menemani penulis menghabiskan waktu luang dalam kesempitan;
8) Cerita abstrak menggugah hati, yang sanggup memberi inspirasi pada penulis;

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Depok, Desember 2010

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR ISTILAH

LCD **:** Liquid Crystal Display
USART **:** Universal Serial Asynchronous Receiver-Transmitter
CAD **:** Computer Aided Design
STL **:** Stereolithography
MC **:** Mikrokontroler
PC **:** Personal Computer

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pada dunia industri pengembangan desain produk merupakan bagian yang vital mengingat begitu banyaknya pesaing yang dengan cepat membuat produk yang sejenis, sehingga produk yang kita miliki harus terus berkembang dan memiliki keunikan atau perbedaan dari produk pesaing. Dalam hal desain ini ada tahapan dimana seorang teknisi desain membuat *prototype* dari desain yang sudah dirancang olehnya agar dapat di lihat secara langsung untuk di nilai kesesuaian desain dengan permintaan konsumen. Tanpa *prototype* ini maka seorang teknisi desain akan sangat kesulitan untuk mengevaluasi bentuk produk yang telah didesainya. Apalagi jika hanya dengan gambar teknik dua dimensi dimana sisi belakang, depan, atas, bawah berada pada gambar yang berbeda, sehingga akan sangat sulit bagi orang awam untuk menilai kesesuaian produk tersebut dengan permintaan konsumen.

Pembuatan *prototype* berperan sangat penting namun proses pembuatan *prototype* ini memakan waktu yang cukup lama dan harga yang cukup mahal, untuk sebuah *prototype*. Sebagai contoh, jika seorang pengrajin tanah liat yang diminta untuk membuat *prototype* tersebut maka ia perlu memahami terlebih dahulu setiap seluk beluk desain yang dibuat. Barulah ia mulai memoles model dari tanah liat hingga sesuai dengan desain, pengrajin tanah liat ini mungkin dapat mengerjakan desain tersebut hingga berhari-hari atau bahkan berminggu-minggu untuk model yang rumit. Maka selama itulah waktu terbuang, belum terhitung jika dalam proses setelahnya desain tersebut ditolak dan harus di buat lagi *prototype* yang lain maka proses pengembangan sebuah produk akan sangat lama. Kasus lainnya jika *prototype* dipilih dibuat langsung dengan proses pemesinan maka biaya proses pemesinan yang cukup tinggi akan sangat membebani perusahaan pada tahap pengembangan produk.

Dengan perkembangan teknologi yang semakin pesat pembuatan *prototype* yang tidak memakan banyak waktu dan biaya dapat terwujud dengan menggunakan mesin *rapid prototyping* yang dapat membuat *prototype* dalam waktu yang singkat dan biaya

yang murah dibandingkan pembuatan *prototype* secara konvensional. Mesin *rapid prototyping* ini pun menjadi alat vital dalam dunia industri. Namun untuk industri di Indonesia belum banyak digunakan dikarenakan harga mesin tersebut terbilang mahal untuk industri-industri berkembang di Indonesia, maka dari itu Laboratorium Manufaktur dan Otomasi DTM FTUI mengembangkan mesin *Rapid Prototyping* dengan teknik *Fused Deposition Modelling* sebagai basis dasar mesin tersebut, dengan harga yang jauh lebih terjangkau.

Gambar 1.1 Contoh Produk

1.2 Tujuan Penelitian

Penelitian ini merupakan bagian riset pada Laboratorium Manufaktur dan Otomasi DTM UI yang ditujukan untuk mengembangkan sebuah desain mesin Rapid Prototyping dengan teknik *Fused Deposition Modelling* dengan biaya rendah. Bagian dari penelitian yang dilakukan ini adalah mengembangkan perangkat lunak yang dapat mengontrol mesin *rapid prototyping* tersebut dan mengirimkan data proses pembuatan produk dengan akurat.

1.3 Perumusan Masalah

Pada mesin rapid prototyping dengan metode FDM (*Fused Deposition Modelling*) yang memiliki tiga derajat kebebasan diperlukan perangkat lunak untuk

mengontrol motor penggerak pada setiap sumbu. Dimana motor sumbu X dan sumbu Y harus dapat bergerak bersamaan untuk membuat membentuk garis miring atau pola berkontur. Perangkat lunak ini juga harus mampu mengirimkan perintah dari PC (*personal computer*) ke mikrokontroler dengan cepat, presisi dan akurat. Sehingga tidak terjadi kesalahan pada pergerakan motor. Selain untuk mengontrol motor ketiga sumbu, perangkat lunak juga harus memiliki *user interface* yang mudah dimengerti dan digunakan dengan baik oleh pengguna.

1.4 Pembatasan Masalah

Penelitian ini membatasi permasalahan pada pengembangan perangkat lunak yang digunakan untuk mengontrol mesin rapid prototyping, mengirim data dari *personal computer* ke mikrokontroler untuk menggerakan motor *stepper* yang mengendalikan pergerakan *nozzle*. Perangkat lunak pada PC dibuat dengan compiler Turbo-C yang dan menggunakan Codevision AVR untuk perangkat lunak yang di-*compile* ke dalam mikrokontroler.

1.5 Metodologi Penelitian

Metodologi penelitian yang diterapkan adalah sebagai berikut:

1. Melakukan studi lapangan melalui pencarian informasi dan akses *web*.
2. Melakukan studi literatur mengenai *rapid prototyping*.
3. Studi literatur mengenai pemrograman dan sistem kontrol dengan mikrokontroler.
4. Pengembangan perangkat lunak.
5. Pengujian dan analisa pada perangkat lunak.

1.6 Sistematika Penulisan

BAB 1. PENDAHULUAN

Pada bab ini dijelaskan mengenai latar belakang penelitian, tujuan penelitian, perumusan masalah, pembatasan masalah, metodologi penelitian, dan sistematika penulisan.

BAB 2. PENGENALAN RAPID PROTOTYPING

Pada bab ini dijelaskan mengenai *rapid prototyping* dan berbagai metode *rapid prototyping* yang biasa digunakan dalam dunia industri.

BAB 3. PENGEMBANGAN PERANGKAT LUNAK KOMUNIKASI MIKROKONTROLER DENGAN PERSONAL COMPUTER

Pada bab ini dijelaskan mengenai perancangan dan pembuatan alogaritma program, *user inferface*, dan format data yang di gunakan.

BAB 4. PENGEMBANGAN INTERPOLATOR UNTUK BENDA BERKONTUR

Pada bab ini dijelaskan mengenai teori interpolasi, perancangan dan pembuatan alogaritma program interoplasi untuk membuat benda berkontur

BAB 5. ANALISA DAN PENGUJIAN PERANGKAT LUNAK

Pada bab ini membahas cara pengiriman data dan menampilkan hasil pengujian keakurasian pada pengiriman data dari PC ke mikrokontroler

BAB 6. KESIMPULAN DAN SARAN PENELITIAN LEBIH LANJUT

Pada bab ini berisi kesimpulan penelitian dan saran untuk penelitian selanjutnya.

BAB 2

PENGENALAN *RAPID PROTOTYPING*

Rapid Prototyping (RP) dapat didefinisikan sebagai metode-metode yang digunakan untuk membuat model berskala (prototipe) dari mulai bagian suatu produk (*part*) ataupun rakitan produk (*assembly*) secara cepat dengan menggunakan data *Computer Aided Design* (CAD) tiga dimensi. *Rapid Prototyping* memungkinkan visualisasi suatu gambar tiga dimensi menjadi benda tiga dimensi asli yang mempunyai volume. Selain itu produk-produk *rapid prototyping* juga dapat digunakan untuk menguji suatu *part* tertentu. Metode RP pertama ditemukan pada tahun 1986 di California, USA yaitu dengan metode *Stereolithography*. Setelah penemuan metode tersebut berkembanglah berbagai metode lainnya yang memungkinkan pembuatan prototipe dapat dilakukan secara cepat.

Saat ini, pembuatan prototipe menjadi syarat tersendiri pada beberapa perusahaan dalam upaya penyempurnaan produknya. Beberapa alasan mengapa *rapid prototyping* sangat berguna dan diperlukan dalam dunia industri adalah:

- Meningkatkan efektifitas komunikasi di lingkungan industri atau dengan konsumen.
- Mengurangi kesalahan-kesalahan produksi yang mengakibatkan membengkaknya biaya produksi.
- Mengurangi waktu pengembangan produk.
- Meminimalisasi perubahan-perubahan mendasar.
- Memperpanjang jangka pakai produk misalnya dengan menambahkan beberapa komponen fitur atau mengurangi fitur-fitur yang tidak diperlukan dalam desain.

Rapid Prototyping mengurangi waktu pengembangan produk dengan memberikan kesempatan-kesempatan untuk koreksi terlebih dahulu terhadap produk yang dibuat (prototipe). Dengan menganalisa prototipe, insinyur dapat mengkoreksi beberapa kesalahan atau ketidaksesuaian dalam desain ataupun memberikan sentuhan-

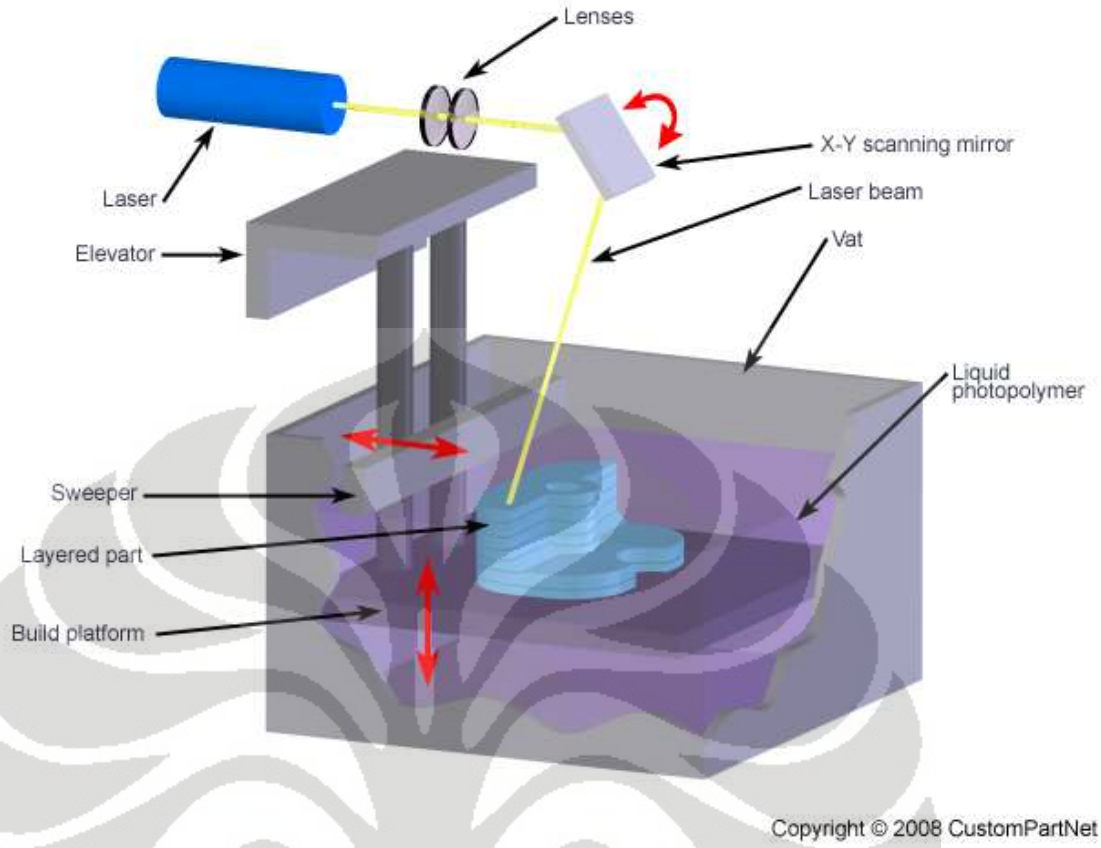
sentuhan *engineering* dalam penyempurnaan produknya. Saat ini tren yang sedang berkembang dalam dunia industri adalah pengembangan variasi dari produk, peningkatan kompleksitas produk, produk umur pakai pendek, dan usaha penurunan biaya produksi dan waktu pengiriman. *Rapid prototyping* meningkatkan pengembangan produk dengan memungkinkannya komunikasi yang lebih efektif dalam lingkungan industri.

Beberapa metode *Rapid Prototyping* yang berkembang saat ini adalah:

1. *Stereolithography* (SLA)
2. *Selective Laser Sintering* (SLS)
3. *Laminated Object Manufacturing* (LOM)
4. *Fused Depsition Modelling* (FDM)
5. *3D Printing* (3DP)

2.1. *Stereolithography* (SLA)

*Stereolithograph*y atau biasa disebut SLA adalah metode *rapid prototyping* yang pertama kali dikembangkan. *3D Systems of Valencia*, CA, USA mengembangkan metode ini pada tahun 1986. Materialnya adalah Resin yang sensitif terhadap cahaya tertentu (*photosensitive resin*) yang ditembakkan sesuai gerakan yang dikontrol oleh sistem komputer dan pembacaan data 3D CAD yang dimasukkan. *Stereolithography* juga biasa disebut *3D-layering*. Menggunakan wadah yang menampung resin bahan baku dan pada bagian tertentu (*elevator*) dapat bergerak vertikal (keatas kebawah atau sumbu-z positif negatif). Cahaya Laser UV jatuh tepat pada layer per layer dan mengeraskan resin yang sensitif. Setelah layer pertama terbentuk, maka laser UV ditembakkan kembali pada resin tersebut untuk membentuk layer kedua. Setiap layer yang telah terbentuk, *elevator* bergerak ke bawah dengan jarak yang sama selama proses berlangsung (pada umumnya 0.003 – 0.002 inch per *layer*). Jadi *layer* kedua terbentuk tepat diatas *layer* pertama dan begitu seterusnya pada *layer* selanjutnya.



Gambar 2.1. *Stereolithography* (SLA)[4]

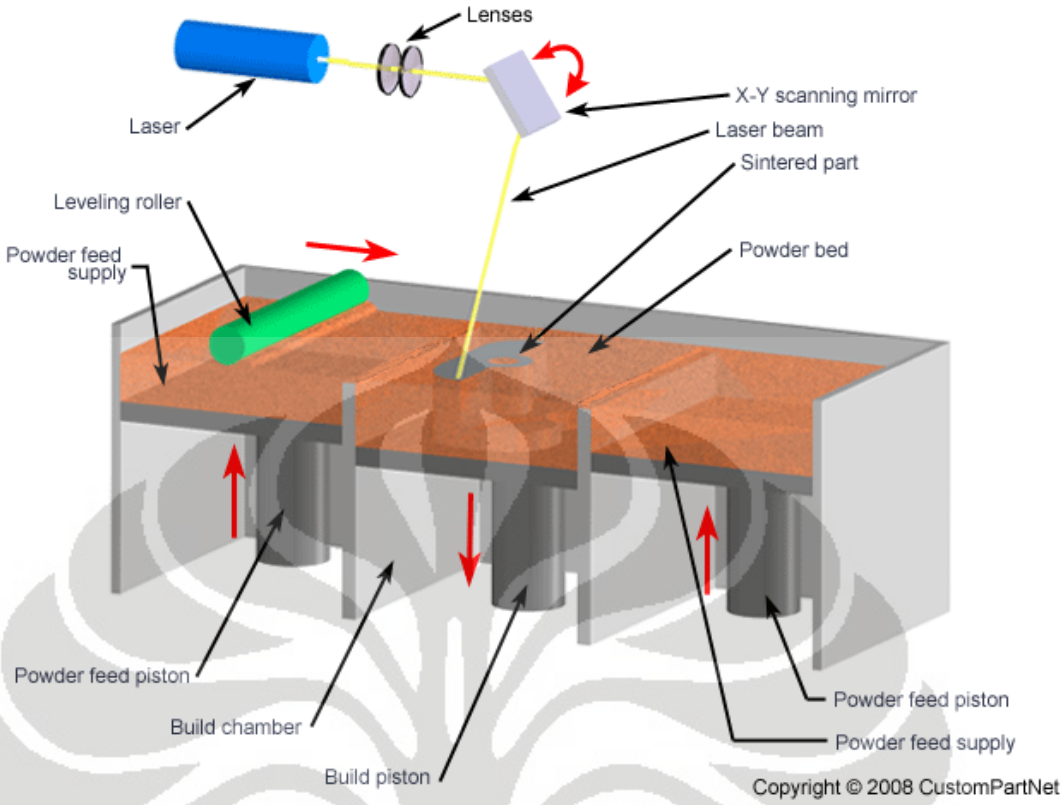
Pada gambar diatas dapat dilihat proses dari metode SLA. Laser UV ditembakkan melalui lensa dan X-Y scanning mirror yang bergerak sesuai kontrol dan pembacaan data 3D CAD membentuk *layer* per-*layer* ke resin fotosensitif dalam wadah. Setelah resin yang ditembakkan laser tadi mengeras, *Sweeper* bergerak diatas *layer* tersebut untuk membersihkan bagian atas *layer* (*finishing layer*). Kemudian *platform* (*elevator*) bergerak kebawah yang menandakan proses pembuatan *layer* pertama telah selesai. Selanjutnya adalah proses yang sama sampai dengan *layer* terakhir terbentuk. Setelah selesai dalam tahap ini, *platform* bergerak keatas. Benda yang disusun dari *layer* per-*layer* tadi menjadi satu benda yang utuh. Benda kemudian dilepaskan dari *platform*. Kebanyakan kasus dalam SLA, *finishing* terakhir benda ditaruh dalam UV oven kemudian dipoles.

Tabel 2.1. Contoh Spesifikasi SLA[4]

Abbreviation:	SLA
Material type:	Liquid (Photopolymer)
Materials:	Thermoplastics (Elastomers)
Max part size:	59.00 x 29.50 x 19.70 in.
Min feature size:	0.004 in.
Min layer thickness:	0.0010 in.
Tolerance:	0.0050 in.
Surface finish:	Smooth
Build speed:	Average
Applications:	Form/fit testing, Functional testing, Rapid tooling patterns, Snap fits, Very detailed parts, Presentation models, High heat applications

2.2. *Selective Laser Sintering* (SLS)

Selective Laser Sintering atau biasa disebut SLS pertama dikembangkan oleh Carl Deckard di University of Texas pada tahun 1989. Konsep dasar dari SLS pada dasarnya sama dengan *Stereolithography* (SLA) yaitu dengan menggunakan tembakan sinar laser yang bergerak untuk membentuk layer pada material bahan baku sehingga terbentuk benda tiga dimensi. Seperti metode RP lainnya, part dibuat diatas sebuah *platform* dimana *platform* tersebut dapat bergerak untuk menyesuaikan pembentukan *layer* per-*layer* sesuai kepresisian gerak *platform* tersebut. Perbedaan dengan SLA adalah pada tipe materialnya. Jika SLA menggunakan resin (liquid), SLS menggunakan bubuk (powder) yang jenisnya lebih beragam seperti termoplastik, elastomer, dan komposit. Tidak seperti SLA, SLS tidak membutuhkan material pendukung untuk menopang struktur yang dibentuk karena tiap *layer* yang dibuat merupakan pendukung pembentuk struktur pada saat dibuat. Dengan material metal composite, proses SLS mengeraskan material polimer disekitar bubuk metal (100 mikron diameter) membentuk *part*. *Part* tersebut kemudian dimasukkan dalam tungku dengan temperatur lebih dari 900°C dimana binder polimer dibakar dan disusupi dengan serbuk perunggu untuk meningkatkan densitas. Biasanya pembakaran memerlukan waktu kurang lebih satu hari yang setelah itu proses *machining* dan *finishing* dilakukan.



Gambar 2.2. *Selective Laser Sintering* (SLS) [4]

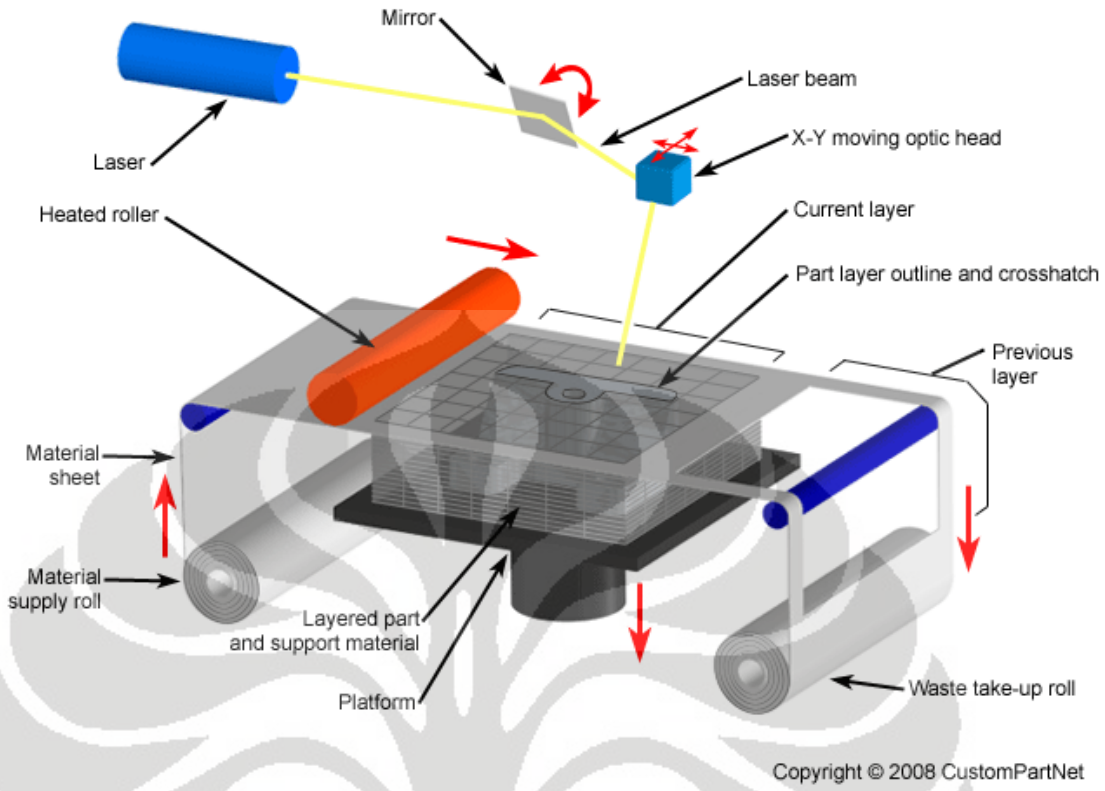
Laser ditembakkan melalui lensa dan dibiaskan oleh cermin yang bergerak mengarahkan sinar pada aksis X-Y sesuai sistem kontrol dan 3D CAD yang diinginkan. *Build piston* bergerak kebawah membentuk *layer* per-*layer*. Material bahan baku berupa bubuk (*powder*) diumpan dari *Powder feed piston* disampingnya dengan bantuan *roller* yang bergerak melintasi *build piston* bolak-balik selama proses. *Powder piston* bergerak keatas sedangkan *build piston* bergerak kebawah sesuai *level layer* yang dibentuk setelah *layer* pertama terbentuk dan begitu seterusnya.

Tabel 2.2. Contoh Spesifikasi SLS[4]

Abbreviation:	SLS
Material type:	Powder (Polymer)
Materials:	Thermoplastics such as Nylon, Polyamide, and Polystyrene; Elastomers; Composites
Max part size:	22.00 x 22.00 x 30.00 in.
Min feature size:	0.005 in.
Min layer thickness:	0.0040 in.
Tolerance:	0.0100 in.
Surface finish:	Average
Build speed:	Fast
Applications:	Form/fit testing, Functional testing, Rapid tooling patterns, Less detailed parts, Parts with snap-fits & living hinges, High heat applications

2.3. *Laminated Object Manufacturing* (LOM)

Laminated Object Manufacturing atau biasa disebut LOM pertama dikembangkan pada tahun 1991. Komponen utama dari sistem LOM ini adalah sebuah material metal lembaran yang diumpan diatas *platform*, *roller* yang dipanaskan untuk memberikan tekanan untuk *layer* dibawahnya dan sinar laser yang ditembakkan pada material tersebut untuk memotong material lembaran pada tiap *layer* tersebut dan seterusnya sehingga terbentuk sebuah *part* tiga dimensi. *Part* terbentuk dengan menumpuk, mengikat, dan memotong *layer* atau lapisan dari lembaran yang dibalut material adesif diatas *layer* sebelumnya. Sinar laser memotong outline part tersebut pada tiap *layer*. Setelah *layer* selesai terbentuk, *platform* bergerak turun (umumnya 0.002 – 0.02 in) dan kemudian lembaran yang lainnya ditempatkan diatas struktur yang telah terbentuk sebelumnya.



Gambar 2.3. *Laminated Object Manufacturing* (LOM) [4]

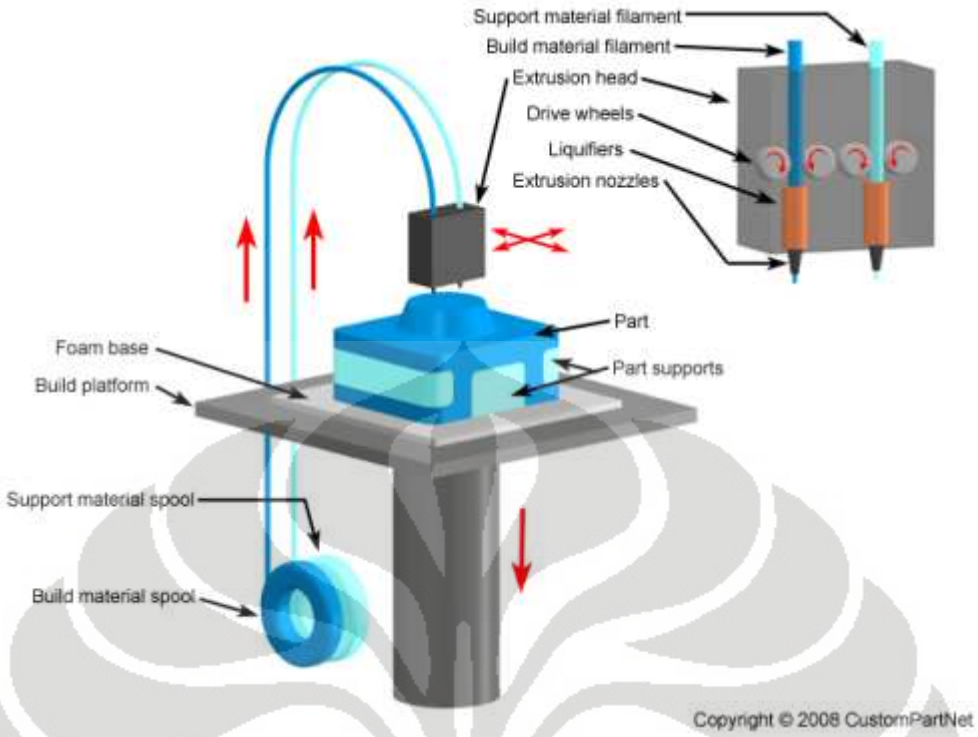
Terlihat pada gambar diatas, sinar laser ditembakkan dan dibiaskan oleh cermin khusus mengarahkan sinar laser ke lembaran material diatas *platform*. Sinar laser memotong outline pada material sesuai desain per-*layer*. Setelah terbentuk *layer*, platform bergerak turun kebawah, *waste roller* menggulung dan *supply roller* yang dipanaskan mensuplai lembaran baru diatas *layer* yang terbentuk sebelumnya. Pada saat proses berlangsung, support material tetap pada tempatnya untuk membantu pembentukan *layer* selanjutnya. Begitu seterusnya sampai dengan *layer* terakhir dan terbentuklah benda tiga dimensi.

Tabel 2.3. Contoh Spesifikasi LOM[4]

Abbreviation:	LOM
Material type:	Solid (Sheets)
Materials:	Thermoplastics such as PVC; Paper; Composites (Ferrous metals; Non-ferrous metals; Ceramics)
Max part size:	32.00 x 22.00 x 20.00 in.
Min feature size:	0.008 in.
Min layer thickness:	0.0020 in.
Tolerance:	0.0040 in.
Surface finish:	Rough
Build speed:	Fast
Applications:	Form/fit testing, Less detailed parts, Rapid tooling patterns

2.4. *Fused Deposition Modelling* (FDM)

*Fused Deposition Model*ling (FDM) adalah metode *Rapid Prototyping* yang sedikit berbeda dengan metode RP lain yang menggunakan sinar laser dalam proses utamanya. FDM memanfaatkan material yang diekstrusi dari sebuah *nozzle* yang kemudian digerakkan oleh motor. Material adalah termoplastik berbentuk benang (koil) yang dipanaskan diatas *melting point* oleh *heater* kemudian diekstrusi lewat lubang extruder *nozzle*. *Heater* mempertahankan temperatur tersebut dan mendeformasi material dari padatan menjadi semi-solid (liquid) agar mudah untuk diekstrusi. *Nozzle* yang bergerak dan mengeluarkan cairan ekstrusi membentuk *layer*. Material plastik ekstrusi akan mengeras secara cepat setelah dikeluarkan melewati *nozzle*. Setelah *layer* pertama terbentuk, *platform* bergerak kebawah dan kemudian adalah proses pembentukan *layer* selanjutnya. Ketebalan layer berkisar antara 0.013 – 0.005 inch. Pada aksis xy resolusi 0.001 inch dapat dicapai. Beberapa material yang dapat digunakan untuk bahan baku adalah ABS, Polyamide (PA), Polycarbonate (PC), Polyethilene (PE), Polypropilene (PP), dan Investment Casting Wax.



Gambar 2.4. *Fused Deposition Modelling* (FDM) [4]

Ketika mesin *Rapid Prototyping* ini akan beroperasi, material bahan untuk membangun keluar dari *nozzle* akibat pemanasan filament (liquefier) pada *heating system* dengan pengaturan laju *feeder* oleh *drive wheel* yang digerakkan oleh motor DC. Setelah dicapai temperature yang sesuai *drive wheel* berputar (saklar *feeder* akan on pada saat program *trajectory* berjalan) untuk menyuplai dan menekan keluar *nozzle* dalam bentuk semi-solid (fase antara *solid* dan *liquid*). Gerakan *nozzle* akan diarahkan sesuai dengan program *trajectory*. Setelah *layer* pertama terbentuk, *platform* bergerak kebawah dan proses sebelumnya berulang sampai dengan *layer* terakhir dan terbentuk benda tiga dimensi.

Ada beberapa variasi dari mesin FDM yang berkembang. Beberapa misalnya posisi *nozzle* yang tetap sedangkan *platform* yang bergerak mengikuti program *trajectory* kearah sumbu xyz. Beberapa pengembangan telah memungkinkan *support* material yang digunakan untuk mendukung pembentukan struktur selama proses. *Support* material adalah material yang heterogen dengan material utama. Umumnya diatas *platform* juga diletakkan beberapa material *platform* tambahan seperti gabus yang

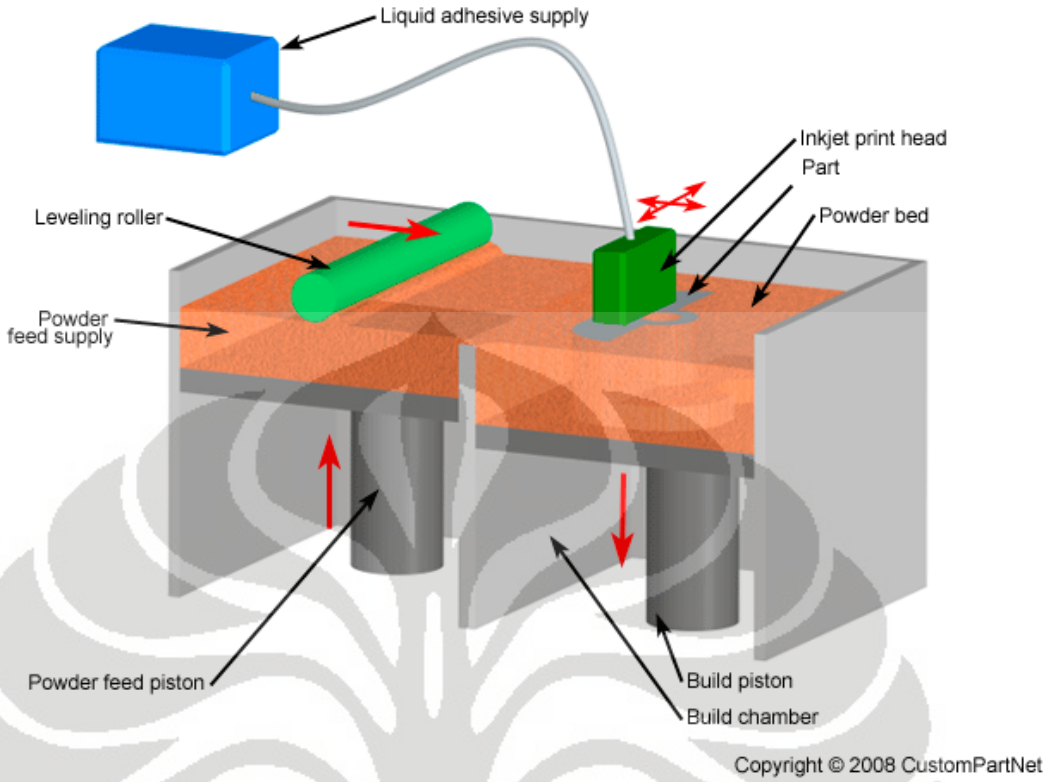
memungkinkan perekatan material pada *layer* pertama. Sumbu xyz digerakkan oleh motor *stepper* yang bergerak perpulsa.

Tabel 2.4. Contoh Spesifikasi FDM[4]

Abbreviation:	FDM
Material type:	Solid (Filaments)
Materials:	Thermoplastics such as ABS, Polycarbonate, and Polyphenylsulfone; Elastomers
Max part size:	36.00 x 24.00 x 36.00 in.
Min feature size:	0.005 in.
Min layer thickness:	0.0050 in.
Tolerance:	0.0050 in.
Surface finish:	Rough
Build speed:	Slow
Applications:	Form/fit testing, Functional testing, Rapid tooling patterns, Small detailed parts, Presentation models, Patient and food applications, High heat applications

2.5. *Three Dimensional Printing* (3DP)

Three Dimensional Printing (3DP) adalah merupakan metode yang mirip dengan metode SLS yaitu dengan menggunakan *powder bed* namun tidak menggunakan sinar laser seperti FDM. Jadi bisa dikatakan 3DP merupakan perpaduan antara keduanya. 3DP menggunakan *ink jet* untuk memberikan perekat cair sebagai pengikat powder bed pada tiap layernya. Beberapa pilihan material yang dapat digunakan untuk metode ini adalah bubuk keramik yang sangat terbatas namun lebih murah dibandingkan dengan material lainnya. Metode ini merupakan salah satu metode RP yang cepat. Umumnya dapat mencapai 2 – 4 *layer* dalam satu menit. Akan tetapi dalam hal keakuratan, *surface finish*, dan kekuatan tidak sebaik metode lainnya.



Gambar 2.5. *Three Dimensional Printing* (3DP) [4]

Proses 3DP dimulai dengan suplai *powder bed* dari *feed piston* yang dibantu oleh *roller* yang mendistribusikan sehingga terbentuk lapisan tipis diatas *platform*. *Ink-jet print head* mendistribusikan perekat cair diatas *layer powder* tersebut sesuai kontrol gerakan dan desain *layer* per-*layer*. *Powder bed* yang direkatkan oleh cairan adesif berikatan sehingga terbentuk sebuah *layer*. Selanjutnya *platform* bergerak kebawah dan proses berulang sampai terbentuk *layer* terakhir. Setelah selesai, bubuk sisa yang tidak termasuk dalam struktur dapat dibuang dan dibersihkan.

Tabel 2.5. Contoh Spesifikasi 3DP[4]

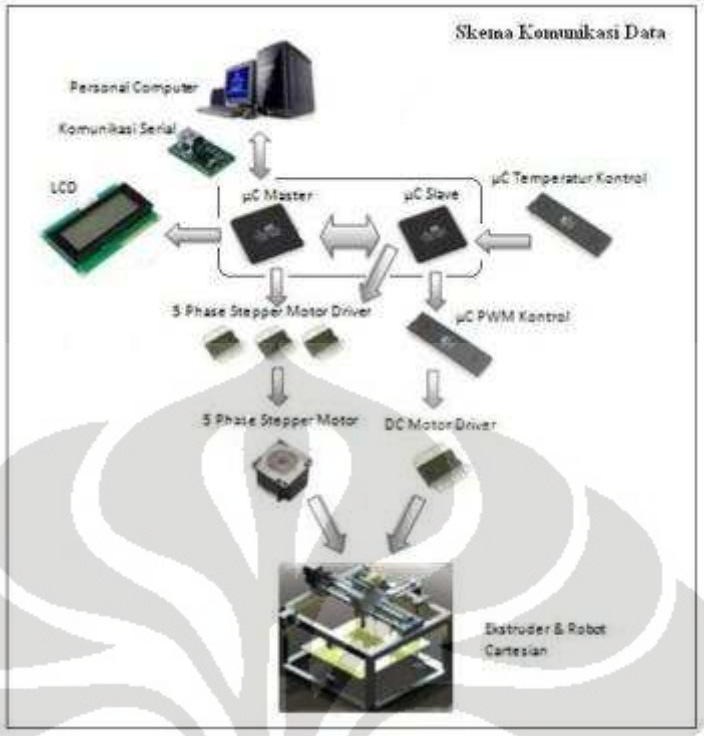
Abbreviation:	3DP
Material type:	Powder
Materials:	Ferrous metals such as Stainless steel; Non-ferrous metals such as Bronze; Elastomers; Composites; Ceramics
Max part size:	59.00 x 29.50 x 27.60 in.
Min feature size:	0.008 in.
Min layer thickness:	0.0020 in.
Tolerance:	0.0040 in.
Surface finish:	Rough
Build speed:	Very Fast
Applications:	Concept models, Limited functional testing, Architectural & landscape models, Color industrial design models, Consumer goods & packaging

BAB 3

PENGEMBANGAN PERANGKAT LUNAK KOMUNIKASI MIKROKONTROLER DENGAN PERSONAL COMPUTER

Perangkat lunak atau program berfungsi untuk mengendalikan sebuah sistem yang bekerja mengirim dan menerima data, program tersebut di-*compile* ke dalam mikrokontroler yang berfungsi sebagai perangkat keras pengendali system. Mikrokontroler ini yang nantinya akan mengirim dan menerima data berisi nilai atau *value*, nilai ini sudah harus dikenali oleh mikrokontroler sehingga pada saat nilai tersebut diterima, mikrokontroler langsung mengetahui *task* atau pekerjaan apa yang harus dilaksanakan. Dalam hal mengenali nilai dan pekerjaan apa yang harus dilakukan oleh mikrokontroler inilah perangkat lunak atau program berfungsi. Untuk meng-*compile* program ke dalam mikrokontroler digunakan *compiler* Codevision AVR.

Mikrokontroler dalam suatu sistem dapat berkomunikasi dengan mikrokontroler pada sistem lainnya dan saling mengirimkan nilai, atau dapat juga mikrokontroler dengan *personal computer*, namun komunikasi antara PC dan mikrokontroler harus dijembatani oleh sebuah konverter dalam hal ini digunakan pololu USB to serial. Pololu USB to serial ini berfungsi menyamakan kecepatan penerimaan dan pengiriman nilai yang dilakukan antara PC dan mikrokontroler sehingga nilai yang dikirim dapat dengan sempurna di terima. Program ini di *desain* khusus untuk menggerakan mesin *rapid prototyping* dengan menggunakan dua buah mikrokontroler yang berkomunikasi dengan PC, menggerakan tiga buah motor untuk tiga sumbu X,Y,dan Z untuk mengendalikan posisi *nozzle*.



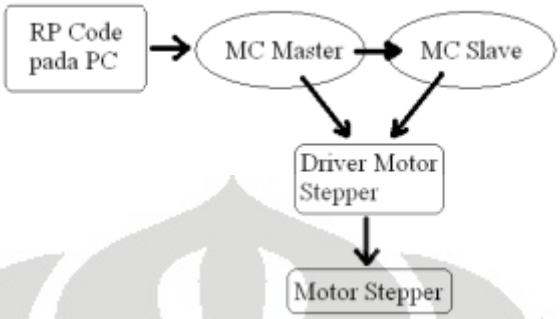
Gambar 3.1 Skema Komunikasi Data

Berikut adalah algoritma umum yang dirancang untuk sistem komunikasi dua mikrokontroler dan PC:

```
Algoritma :

    START

        1) PC menunggu mikrokontroler mengirim signal bahwa heater sudah siap.
        2) PC mengirim RP code berupa koordinat ke mikrokontroler master.
        3) Mikrokontroler master menampilkan data pada LCD.
        4) Mikrokontroler master mengirim koordinat X,Y ke mikrokontroler
           slave.
        5) Mikrokontroler master menjalankan motor sumbu X kemudian Z sesuai
           RP code.
        6) Mikrokontroler slave menjalankan motor sumbu Y sesuai RP code.
        7) Mikrokontroler slave memberi signal kepada mikrokontroler master
           bahwa motor telah selesai dijalankan.
        8) Mikrokontroler master memberi signal pada PC bahwa motor telah
           selesai dijalankan.
        9) PC mengirim koordinat berikutnya.
        10)Program berulang sampai koordinat terakhir.

    END
```

Pengaturan awal yang digunakan pada dua buah mikrokontroler yang digunakan adalah menjadikan salah satu mikrokontroler menjadi *master* dan mikrokontroler lainnya

sebagai *slave*, mikrokontroler *master* inilah yang berhubungan langsung dengan PC dan menerima data dari PC yang kemudian di kirimkan lagi kepada mikrokontroler *slave*.



Gambar 3.2 Bagan Pengiriman Data

Untuk mengirim data antar sesama mikrokontroler atau dari mikrokontroler ke PC dan sebaliknya terdapat kemungkinan terjadi *error* dalam proses pengiriman. Error ini dapat terjadi dikarenakan penerima belum siap menerima data namun pengirim sudah mengirimkan data sehingga data yang seharusnya diterima hilang karena ketidak-siapan dari penerima data. Untuk menanggulangi hal ini dibuatlah sebuah *communication protocol* yang berfungsi untuk mengatur agar data dikirim secara berurutan dan teratur, sehingga dapat mengurangi terjadinya *error* dalam pengiriman data.

3.1 Protokol Pengiriman Data Antara PC dan Mikrokontroler *Master*

Pengiriman data dari PC ke mikrokontroler *master* merupakan bagian terpenting, jika terjadi kesalahan atau *error* maka seluruh proses akan kacau. Karena data yang dikirim oleh PC ke mikrokontroler *master* masih akan diteruskan ke mikrokontoler *slave*. Sehingga jika terjadi *error* mikrokontroler *slave* pun akan mengalami error yang sama.

Protokol yang digunakan adalah sebagai berikut:

```
Protokol :

    START

       1) Membuka port komunikasi serial.
```

```
    2) PC mengunggu mikrokontroler master mengirim karakter ‘z’ yang
       menandakan bahwa heater siap.
    3) If karakter sudah di terima oleh PC tunggu dua setengah detik,
       kemudian tampilkan pilihan mode
    4) Switch, menunggu perintah user.
       1. Case 1: posisi awal.
       2. Case 2: kirim data.
       3. Case 3: exit program.
    5) Mikrokontroler menunggu PC mengirim karakter
    6) If case 1 yang di pilih, PC looping mengirim karakter ‘A’ ke
       mikrokontroler hingga mikrokontoler mengirim kembali karakter ‘A’.
    7) Mikrokontroler menjalankan perintah posisi awal.
    8) If case 2 yang di pilih, PC mengirimkan karakter ‘B’
    9) Mikrokontroler menjalankan perintah jalan program.
    10) Mikrokontroler menunggu PC mengirim satu karakter dari satu string
        data.
    11) Setelah data diterima dan di simpan mikrokontroler mengirim karakter
        ‘k’ ke PC
    12) PC menunggu hingga karakter ‘k’ dikirim.
    13) Setelah karakter ‘k’ diterima, PC melanjutkan mengirim karakter
        selanjutnya dari satu string data
    14) Pada akhir dari string mikrokontroler master mengirim karakter ‘x’
        yang menandakan akhir string.
    15) Mikrokontroler master mengirimkan satu karakter dari string
        berikutnya.
    16) Setelah 4 string data terkumpul, motor di jalankan.
    17) Proses berulang hinga seluruh data selesai di kirim.

END
```

3.2 Protokol Pengiriman Data Antar Dua Mikrokontroler

Seperti halnya pada pengiriman data antara mikrokontroler dan PC, pengiriman data antar dua mikrokontroler juga merupakan hal yang penting karena jika terjadi kesalahan maka sebagian dari sistem akan kacau, untuk itu dirancang protokol dengan sedemikian rupa sehingga kemungkinan *error* yang terjadi dapat ditiadakan.

Protokol yang digunakan adalah sebagai berikut:

```
Protokol :

    START

    1) Mikrokontroler slave menunggu mikrokontroler master mengirim
       karakter.
    2) If user memilih perintah posisi awal mikrokontroler master mengirim
       karakter ‘A’ ke mikrokontroler slave.
    3) Mikrokontroler master dan mikrokontroler slave menjalankan perintah
       posisi awal.
    4) If user memilih perintah jalan program mikrokontroler master
       mengirim karakter ‘B’ ke mikrokontroler slave.
    5) Mikrokontroler slave menjalankan perintah terima data.
    6) Mikrokontroler slave menunggu mikrokontroler master mengirim satu
       string data
```

```
    7) Stelah mikrokontroler master mengirim satu string data, karakter 'x'
       dikirim untuk menandakan akhir dari string.
    8) Mikrokontroler slave menyimpan data.
    9) Setelah 3 string data terkumpul, motor dijalankan.
    10)Proses berulang hingga seluruh data selesai di kirim.

END
```

3.3 Fungsi Utama Pada PC

PC berperan sebagai pengirim RP *code* yang berisi koordinat tujuan motor harus bergerak.RP *code* ini akan di terangkan lebih lanjut pada sub bab RP *code*

Pada fungsi utama PC ini terdapat tiga pilihan, yaitu:

1. Posisi awal : menggerakan *nozzle* ke posisi awal
2. Kirim data : mengirimkan RP *code*
3. Exit program : keluar dari program

```
Algoritma :

  START

    1) Membuka port komunikasi serial.
    2) Menunggu mikrokontroler master mengirim signal bahwa heater siap
       printf("menunggu heater siap").
    3) IF mikrokontroler master sudah mengirim signal printf("heater
       siap").
    4) Tunggu dua setengah detik.
    5) Tampilkan pilihan mode.
    6) Switch, menunggu perintah user.
       4. Case 1: posisi awal.
       5. Case 2: kirim data.
       6. Case 3: exit program.
    7) IF perintah salah, printf("Invalid Mode Selection").

  END
```

Program dapat di lihat pada Lampiran 3 baris 170-210.

3.3.1 Fungsi Membuka PORT Komunikasi

Port komunikasi adalah bagian penting yang berfungsi untuk mengirimkan data dari PC ke mikrokontroler. Namun tidak semua *compiler* C memiliki akses luas untuk mengirim dan menerima data dari luar *operating system* windowsXP, oleh karena itu di

pilihlah *compiler* Turbo-C yang memiliki fungsi khusus untuk mengirim dan menerima data. Berikut adalah fungsi untuk membuka PORT komunikasi, PORT yang dipakai adalah COM1.

```
Algoritma :

       START

          1) Matikan interrupt COM1.
          2) Pengaturan baud rate, 9200BPS.
          3) Pengaturan tipe serial, 8 Bits, No Parity, 1 Stop Bit.
          4) Menyalakan DTR, RTS, and OUT2.

       END
```

Program dapat di lihat pada Lampiran 3 baris 21-44.

3.3.2 Perintah Posisi Awal

Fungsi mengirim data untuk posisi awal, mikrokontroler akan melakukan perintah posisi awal jika mendapatkan karakter yang telah ditentukan. Setelah selesai mikrokontroler akan mengirim kembali karakter sebagai konfirmasi bahwa perintah sudah selesai dilaksanakan.

```
Algoritma :

       START

          1) Membuka PORT serial
          2) Memerintahkan mikrokontroler melakukan fungsi posisi awal.
          3) Menunggu verifikasi dari microcontroller.

       END
```

Program dapat di lihat pada Lampiran 3 baris 146-160.

3.3.3 Perintah Mengirim Data

Perintah mengirim data RP code yang telah disimpan didalam notepad untuk menggerakan tiga buah motor sumbu dan satu buah motor extruder.

```
Algoritma :

       START

          1) Mengosongkan tampilan layar.
```

```
    2) Mengosongkan buffer.
    3) Membuka PORT serial.
    4) Memerintahkan mikrokontroler melakukan fungsi terima data.
    5) Membuka file “input.txt”.
    6) IF file tidak bisa dibuka printf(“file tidak bisa dibuka”).
    7) Loop hingga end of file.
    8) Printf (“data ke  “).
    9) Kirim file input per baris ke mikrokontroler.
    10) Tampilkan data yang telah di kirim.
    11) Menunggu verifikasi mikrokontroler.
    12) Menutup file “input.txt”.
    13) Printf(“reading finish”).

END
```

Program dapat di lihat pada Lampiran 3 baris 64-143.

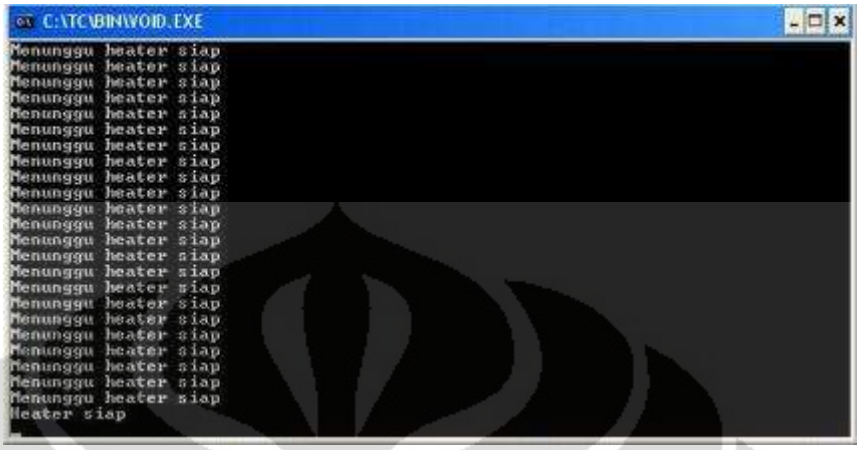
3.3.4 Tampilan *User Interface*

Pada *user interface*, pengguna dapat memilih perintah apa yang ingin diberikan kepada mikrokontroler, *user interface* ini juga berguna sebagai alat agar pengguna mengetahui sudah sampai dimana program dijalankan, dapat juga sebagai media untuk mengetahui dimana kesalahan terjadi.

Pada saat program dijalankan, program akan melakukan looping menunggu heater siap

Gambar 3.3 Menunggu Heater Siap

Ketika heater siap user interface akan menunggu 2.5 detik ini diperuntukan agar panas heater lebih menyebar dan merata.

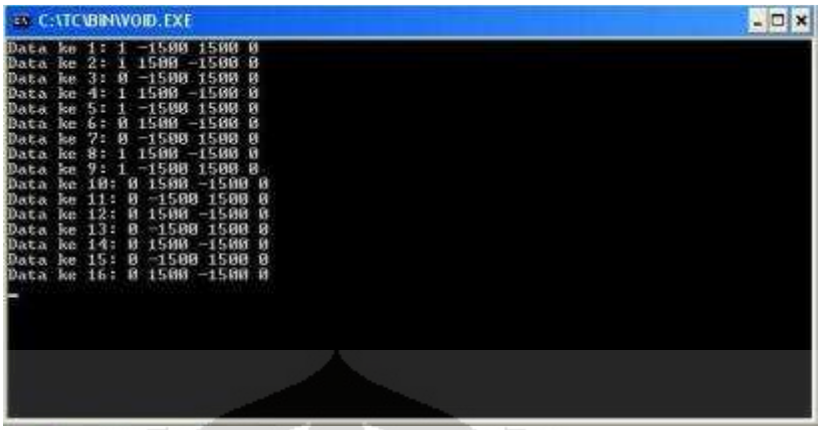


Gambar 3.4 Indikasi Heater Siap

Setelah menunggu 2.5 detik baru akan muncul pilihan perintah yang dapat di pilih oleh pengguna

Gambar 3.5 Pilihan Mode yang Disediakan

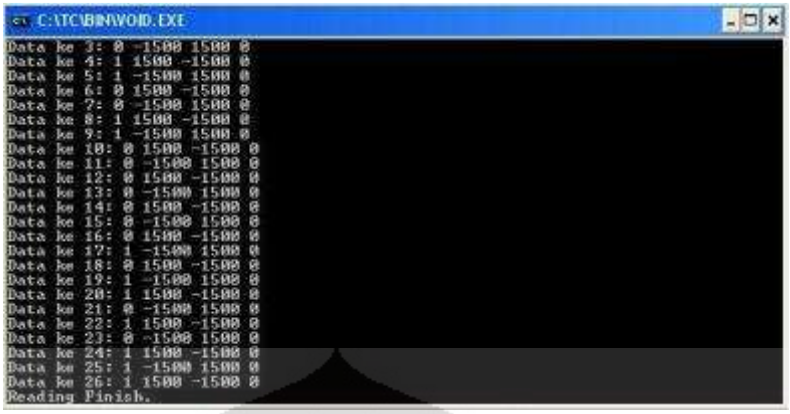
Pengguna hanya perlu memasukan angka perintah yang diinginkan kemudian menekan enter. Jika pengguna memilih posisi awal maka mikrokontroler akan menggerakan *nozzle* ke posisi awal, jika user memilih kirim data maka program akan mulai mengirim data dan user interface akan menampilkan data data yang telah di kirim. Perintah exit digunakan untuk keluar dari program.



Gambar 3.6 Pengiriman Data

User interface akan menampilkan data keberapa yang telah dikirim dan sedang dijalankan oleh mikrokontroler, LCD pada mikrokontroler juga akan menampilkan data yang diterima mikrokontroler

Gambar 3.7 Tampilan LCD

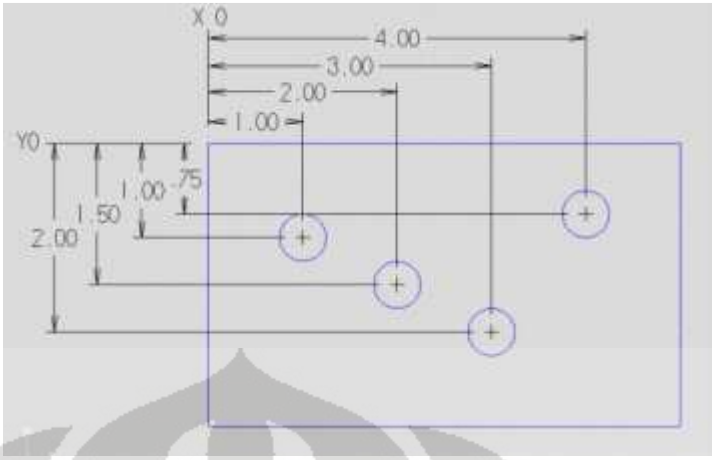


Gambar 3.8 Pengiriman Data Selesai

Setelah seluruh data selesai di kirim dan di jalankan oleh mikrokontroler user interface akan member tahu pengguna bahwa pembacaan data telah selesai.

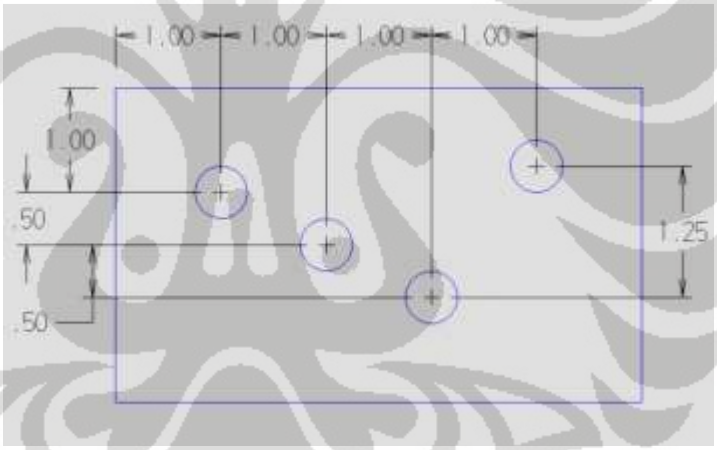
3.3.5 RP *Code*

RP *code* adalah koordinat pergerakan yang akan dikirim ke mikrokontroler, sehingga motor bergerak sesuai koordinat yang diberikan, RP *code* ini memiliki empat buah data, data pertama berisi *code* 1 dan 0 yang menandakan hidup atau matinya motor extruder. Jika data pertama berisikan angka 1 maka motor extruder menyala dan sebaliknya jika berisikan angka 0 motor extruder mati, data kedua berisi koordinat sumbu X, data ke tiga berisi koordinat sumbu Y, data ke 4 berisi koordinat sumbu Z. Metode yang digunakan untuk penulisan koordinat adalah metode *increment*. Perbedaan metode *increment* dan *absolute* terdapat pada titik awal koordinat berikut setelah satu koordinat dijalankan. Pada metode *absolute* titik awal akan kembali ke nol pada penulisan koordinat.



Gambar 3.9 *Absolut*[7]

Sedangkan pada metode *increment* titik awal tidak kembali ke nol melainkan dimulai dari koordinat terakhir extruder berada.

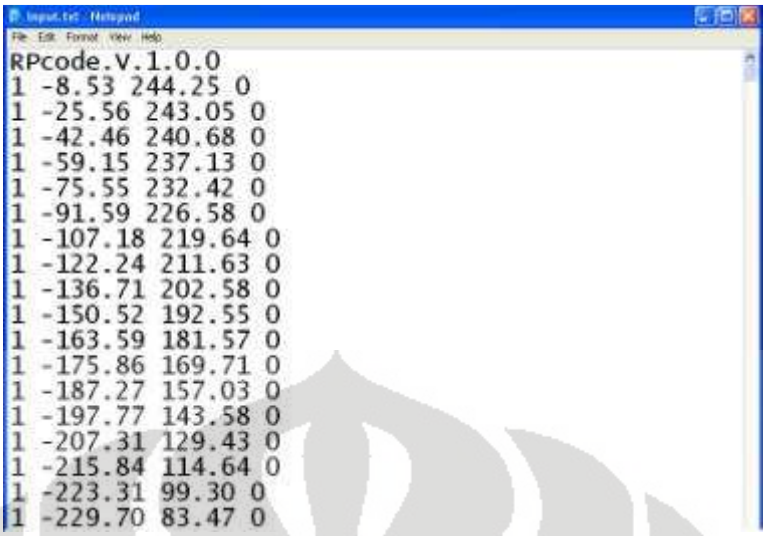


Gambar 3.10 *Increment*[4]

File RP *code* memiliki format data sebagai berikut:

<Header file> </n>

<char:num><space><stringx:num><space><stringy:num><space><stringz:num></n>



Gambar 3.11 Format Data

Susunan data seperti ini digunakan untuk mempermudah pengiriman data kedalam mikrokontroler. Keterangan format data adalah sebagai berikut:

<Header file> = Berfungsi untuk menunjukan versi dari format data.

<char:num> = Data dengan jenis karakter yang berisi angka 1 atau 0.

<stringx:num> = Data dengan jenis string, berisi angka yang menandakan koordinat sumbu X dengan ketelitian 0.01 mm.

<stringy:num> = Data dengan jenis string, berisi angka yang menandakan koordinat sumbu Y dengan ketelitian 0.01 mm.

<stringz:num> = Data dengan jenis string, berisi angka yang menandakan koordinat sumbu Z dengan ketelitian 0.01 mm.

Metode pengiriman data yang dilakukan adalah pengiriman per karakter setelah itu baru di kelompokan kembali dalam bentuk string dan dimasukan ke dalam *array*.

3.4 Fungsi Utama Pada Mikrokontroler *Master*

Fungsi utama mikrokontroler master yang menerima data dari PC terdapat 2 perintah yaitu:

1. Posisi awal : mikrokontroler menggerakan *nozzle* ke posisi awal
2. Jalan program : mikrokontroler menjalankan *nozzle* sesuai koordinat

algoritma yang digunakan adalah sebagai berikut:

```
Algoritma :

    START

       1) Menunggu mikrokontroler slave mengirim signal temperature.
       2) Mengirim signal ke PC.
       3) Switch, menunggu perintah user.
          1. Case 1: posisi awal.
          2. Case 2: jalan program.

    END
```

Program dapat di lihat pada Lampiran 1 baris 886-916.

3.4.1 Pengaturan I/O Mikrokontroler *Master*

Pengaturan awal pada mikrokontroler untuk I/O sudah di atur secara otomatis oleh *compiler* CodeVision AVR, namun untuk pengaplikasian lebih lanjut pengaturan ini dapat dirubah. Misalnya untuk merubah pengaturan port yang tidak aktif menjadi aktif selama program berjalan. Berikut adalah fungsi pengaturan setiap PORT pada mikrokontroler

```
Algoritma :

    START

       1) Setting PORT F
       2) Setting PORT A
       3) Setting PORT B
       4) Setting PORT D
       5) Setting PORT E
       6) Setting PORT H
       7) Setting PORT L

    END
```

Program dapat di lihat pada Lampiran 1 baris 774-845.

3.4.2 Pengaturan LCD Pada Mikrokontroler *Master*

LCD berfungsi untuk menampilkan koordinat yang dituju oleh *nozzle*. LCD ini juga berfungsi sebagai media verifikasi untuk mengetahui apakah data yang di kirim akurat. LCD yang di gunakan memiliki 20 karakter dan diseting berada pada port C

```
Algoritma :

    START

       1) Mulai penulisan in-line assembly language.
       2) Setting LCD pada PORT C.
       3) Tutup penulisan in-line assembly languange.

    END
```

Program dapat di lihat pada Lampiran 1 baris 9-13.

3.4.3 Pengaturan *Clock Speed* Pada Mikrokontroler *Master*

Clock speed adalah kecepatan suatu mikrokontroler melakukan suatu proses per-*cycle*. Clock speed diatur oleh besarnya oscillator yang dipakai yaitu 16Mhz. selain untuk menentukan kecepatan proses mikrokontroler, clock speed yang digunakan juga sangat mempengaruhi error yang terjadi pada saat proses berlangsung

```
Algoritma :

    START

       1) Setting oscillator.
       2) Setting factor pembagi.

    END
```

Program dapat di lihat pada Lampiran 1 baris 766-772.

3.4.4 Pengaturan Komunikasi Serial Pada Mikrokontroler *Master*

Untuk mengirim dan menerima data pada mikrokontroler digunakan port komunikasi serial. *Baud rate* yang digunakan adalah 9200bps *baud rate* ini yang

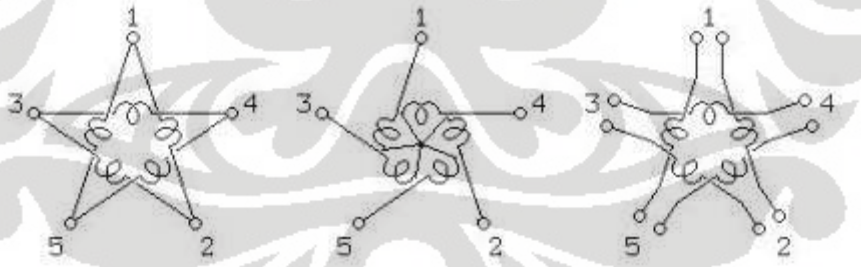
menentukan kecepatan transfer oleh USART. Port komunikasi serial yang digunakan adalah USART0 dan USART2.

```
Algoritma :

    START

       1) Setting tipe data serial communication, 8 Data, 1 Stop, No Parity
       2) Aktifasi transmitter receiver
       3) Setting mode serial asynchronous
       4) Setting baud rate 9200 BPS.

    END
```

Program dapat di lihat pada Lampiran 1 baris 849-871.

3.4.5 Fungsi Menggerakan Motor Pada Mikrokontroler *Master*

Motor yang digunakan ada motor *stepper bipolar* 5 fasa, berbeda dengan motor *stepper bipolar* pada umumnya motor *stepper bipolar* 5 fasa ini memiliki 5 buah input dengan 10 buah kutub magnet yang harus di aktifkan dengan urutan tertentu untuk menggerakan motor tersebut. Karena memiliki 10 kutub magnet maka pada motor *stepper bipolar* 5 fasa ini untuk melakukan 1 gerakan sebesar $0{,}72^0$ diperlukan 10 step. Kombinasi input yang harus diberikan pada motor dapat dilihat seperti gambar 3.11.



Gambar 3.12 Skema Motor Stepper[6]

```
Terminal 1  +++-----++++-----++
Terminal 2  --+++++-----+++++---
Terminal 3  +-----++++-----++++
Terminal 4  +++++-----++++-----
Terminal 5  ----+++++-----+++++-
            time --->
```

pada motor *stepper* sumbu X+ atau putaran motor searah jarum jam, untuk mengatur gerakan digunakan PORTA.1, PORTA.2, PORTA.3, PORTA.4, PORTA.5 dengan algoritma:

```
Algoritma :

    START

       1) Jalankan step 1 motor stepper sesuai delay.
       2) Jalankan step 2 motor stepper sesuai delay.
       3) Jalankan step 3 motor stepper sesuai delay.
       4) Jalankan step 4 motor stepper sesuai delay.
       5) Jalankan step 5 motor stepper sesuai delay.
       6) Jalankan step 6 motor stepper sesuai delay.
       7) Jalankan step 7 motor stepper sesuai delay.
       8) Jalankan step 8 motor stepper sesuai delay.
       9) Jalankan step 9 motor stepper sesuai delay.
       10)Jalankan step 10 motor stepper sesuai delay.

    END
```

Program dapat di lihat pada Lampiran 1 baris 59-184.

pada motor *stepper* sumbu X- atau putaran motor berlawanan arah jarum jam untuk mengatur gerakan digunakan PORTA.1, PORTA.2, PORTA.3, PORTA.4, PORTA.5 dengan algoritma:

```
Algoritma :

    START

       1) Jalankan step 1 motor stepper sesuai delay.
       2) Jalankan step 2 motor stepper sesuai delay.
       3) Jalankan step 3 motor stepper sesuai delay.
       4) Jalankan step 4 motor stepper sesuai delay.
       5) Jalankan step 5 motor stepper sesuai delay.
       6) Jalankan step 6 motor stepper sesuai delay.
       7) Jalankan step 7 motor stepper sesuai delay.
       8) Jalankan step 8 motor stepper sesuai delay.
       9) Jalankan step 9 motor stepper sesuai delay.
       10)Jalankan step 10 motor stepper sesuai delay.

    END
```

Program dapat di lihat pada Lampiran 1 baris 189-317.

Putaran motor berlawanan arah jarum jam adalah kebalikan dari putaran motor searah jarum jam, step 1 pada putaran motor searah jarum jam sama dengan step 10

pada putaran motor berlawanan arah jarum jam. Selain menggerakan motor sumbu X mikrokontroler master juga menggerakan motor sumbu Z.

Pada motor stepper sumbu Z+ atau putaran motor searah jarum jam, untuk mengatur gerakan digunakan PORTE.2, PORTE.4, PORTE.6, PORTE.3, PORTE.5 dengan algoritma:

```
Algoritma :

    START

       1) Jalankan step 1 motor stepper sesuai delay.
       2) Jalankan step 2 motor stepper sesuai delay.
       3) Jalankan step 3 motor stepper sesuai delay.
       4) Jalankan step 4 motor stepper sesuai delay.
       5) Jalankan step 5 motor stepper sesuai delay.
       6) Jalankan step 6 motor stepper sesuai delay.
       7) Jalankan step 7 motor stepper sesuai delay.
       8) Jalankan step 8 motor stepper sesuai delay.
       9) Jalankan step 9 motor stepper sesuai delay.
      10) Jalankan step 10 motor stepper sesuai delay.

    END
```

Program dapat di lihat pada Lampiran 1 baris 319-416..

pada motor *stepper* sumbu X- atau putaran motor berlawanan arah jarum jam untuk mengatur gerakan digunakan PORTE.2, PORTE.4, PORTE.6, PORTE.3, PORTE.5 dengan algoritma:

```
Algoritma :

    START

       1) Jalankan step 1 motor stepper sesuai delay.
       2) Jalankan step 2 motor stepper sesuai delay.
       3) Jalankan step 3 motor stepper sesuai delay.
       4) Jalankan step 4 motor stepper sesuai delay.
       5) Jalankan step 5 motor stepper sesuai delay.
       6) Jalankan step 6 motor stepper sesuai delay.
       7) Jalankan step 7 motor stepper sesuai delay.
       8) Jalankan step 8 motor stepper sesuai delay.
       9) Jalankan step 9 motor stepper sesuai delay.
      10) Jalankan step 10 motor stepper sesuai delay.

    END
```

Program dapat di lihat pada Lampiran 1 baris 418-519.

Untuk pengontrolan dua buah motor *stepper* menggunakan satu mikrokontroler, motor *stepper* tersebut tidak bisa dijalankan bersamaan secara lancar, karena mikrokontroler tidak dapat mengirimkan sinyal ke dua motor secara bersamaan melainkan bergantian dari satu motor ke motor yang lainya, sehingga hasil gerakan motor tidak lancar atau agak tersendat. Motor stepper sumbu X dan Z tidak perlu digerakan bersamaan, karena motor sumbu Z hanya bergerak ketika motor sumbu X dan Y sudah berhenti untuk berpindah ke *layer* berikutnya, sehingga motor sumbu Z dapat diletakan pada mikrokontroler master bersama motor sumbu X. Berbeda halnya dengan motor sumbu Y yang harus dapat bergerak bersamaan dengan motor sumbu X sehingga motor sumbu Y diletakan pada mikrokontroler slave.

3.4.6 Perintah Posisi Awal

Sebelum memulai membentuk benda mesin RP FDM harus diatur terlebih dahulu koordinat awalnya atau titik (0,0). Posisi awal ini berperan penting untuk menentukan MCS atau *machine coordinate system*

```
Algoritma :

    START

        1) Memerintahkan mikrokontroler slave melakukan fungsi posisi awal
        2) Menampilkan pada lcd (“Default posisition”)
        3) Menjalankan motor X ke arah X+ hingga menekan limit switch
        4) Menjalankan motor Z ke arah Z- hingga menekan limit switch

    END
```

Program dapat di lihat pada Lampiran 1 baris 521-562.

3.4.7 Fungsi Menampilkan Data Pada LCD Mikrokontroler *Master*

Setelah pengaturan awal LCD untuk menampilkan data pada LCD perlu di buat fungsi tersendiri.

```
Algoritma :

   START
```

```
    1) Menuju posisi LCD 0,0
    2) Konversi data int menjadi ASCII
    3) Tampilkan data di LCD

END
```

Program dapat di lihat pada Lampiran 1 baris 889-891

3.4.8 Perintah Jalan Program

Fungsi jalan program merupakan fungsi yang mengatur penerimaan data koordinat dari PC kemudian mengirimkanya ke mikrokontroler *slave* dan menjalankan motor sesuai koordinat yang di terima.

```
Algoritma :

  START

    1) Memerintahkan mikrokontroler slave melakukan fungsi jalan program.
    2) Memulai loop menerima data per karakter.
    3) Merubah karakter menjadi float.
    4) Memasukan nilai float ke dalam array.
    5) Memberi feed back ke PC.
    6) Menampilkan data yang diterima pada lcd.
    7) Mengirim koordinat X dan Y ke mikrokontroler slave
    8) Menunggu konfirmasi dari mikrokontroler slave.
    9) Menjalankan motor sumbu X
    10)Menjalankan motor sumbu Z
    11)Memberi konfirmasi ke PC.
    12)Menerima data berikutnya.

  END
```

Program dapat di lihat pada Lampiran 1 baris 564-757.

3.5 Fungsi Utama Pada Mikrokontroler *Slave*

Mikrokontroler *slave* menerima data dari mikrokontroler heater, menerima dan mengirim data ke mikrokontroler master dan berfungsi untuk menggerakan motor extruder. Perintah yang ada pada mikrokontroler *slave* adalah sebagai berikut:

1. Posisi awal : mikrokontroler menggerakan *nozzle* ke posisi awal
2. Terima data : mikrokontroler menjalankan *nozzle* sesuai koordinat

```
Algoritma :

    START

       1) Menunggu signal dari mikrokontroler tempratur.
       2) Mengirim signal ke mikrokontroler master.
       3) Switch, menunggu perintah user.
          1. Case 1: posisi awal.
          2. Case 2: terima data.

    END
```

Program dapat di lihat pada Lampiran 2 baris 625-649.

3.5.1 Pengaturan I/O Mikrokontroler *Slave*

Seperti halnya pada mikrokontroler *master*, pada mikrokontroler *slave* juga perlu dilakukan pengaturan port I/O, sebagai pengaturan awal pengaktifan port.

```
Algoritma :

    START

       1) Setting PORT F
       2) Setting PORT A
       3) Setting PORT B
       4) Setting PORT D
       5) Setting PORT E
       6) Setting PORT H
       7) Setting PORT L

    END
```

Program dapat di lihat pada Lampiran 2 baris 519-584.

3.5.2 Pengaturan LCD Pada Mikrokontroler *Slave*

Pada mikrokontroler *slave* LCD diseting berada pada port A dan berfungsi untuk menampilakn data yang di kirim dari mikrokontroler *master*.

```
Algoritma :

    START

       1) Mulai penulisan in-line assembly language.
       2) Setting LCD pada PORT C.
       3) Tutup penulisan in-line assembly languange.

    END
```

Program dapat di lihat pada Lampiran 2 baris 7-11.

3.5.3 Pengaturan Clock Speed Pada Mikrokontroler *Slave*

Oscillator berfungsi untuk menentukan kecepatan mikrokontroler bekerja atau *clock speed* yang digunakan, oscillator yang dipakai adalah 16Mhz

```
Algoritma :

    START

       1) Setting oscillator.
       2) Setting factor pembagi.

    END
```

Program dapat di lihat pada Lampiran 2 baris 511-517.

3.5.4 Pengaturan Komunikasi Serial Pada Mikrokontroler *Slave*

Untuk menerima data dari mikrokontroler master, mikrokontroler *slave* juga menggunakan port komunikasi serial. PORT komunikasi serial yang digunakan adalah USART0, USART2, dan USART3

```
Algoritma :

    START

       1) Setting tipe data serial communication, 8 Data, 1 Stop, No Parity
       2) Aktifasi transmitter receiver
       3) Setting mode serial asynchronous
       4) Setting baud rate 9200 BPS.

    END
```

Program dapat di lihat pada Lampiran 2 baris 586-620.

3.5.5 Fungsi Menggerakan Motor Pada Mikrokontroler *Slave*

Seperti telah dibahas sebelumnya motor yang dijalankan oleh mikrokontroler slave ini adalah motor sumbu Y agar motor sumbu X dan sumbu Y dapat digerakan secara bersamaan dengan lancar tanpa tersendat.

Pada motor *stepper* sumbu Y+ atau putaran motor searah jarum jam, untuk mengatur gerakan digunakan PORTC.1, PORTC.2, PORTC.3, PORTC.4, PORTC.5 dengan alogaritma:

```
Algoritma :

    START

       1) Jalankan step 1 motor stepper sesuai delay.
       2) Jalankan step 2 motor stepper sesuai delay.
       3) Jalankan step 3 motor stepper sesuai delay.
       4) Jalankan step 4 motor stepper sesuai delay.
       5) Jalankan step 5 motor stepper sesuai delay.
       6) Jalankan step 6 motor stepper sesuai delay.
       7) Jalankan step 7 motor stepper sesuai delay.
       8) Jalankan step 8 motor stepper sesuai delay.
       9) Jalankan step 9 motor stepper sesuai delay.
       10)Jalankan step 10 motor stepper sesuai delay.

    END
```

Program dapat di lihat pada Lampiran 2 baris 80-207.

Pada motor *stepper* sumbu Y- atau putaran motor berlawanan arah jarum jam, untuk mengatur gerakan digunakan PORTC.1, PORTC.2, PORTC.3, PORTC.4, PORTC.5 dengan alogaritma:

```
Algoritma :

    START

       1) Jalankan step 1 motor stepper sesuai delay.
       2) Jalankan step 2 motor stepper sesuai delay.
       3) Jalankan step 3 motor stepper sesuai delay.
       4) Jalankan step 4 motor stepper sesuai delay.
       5) Jalankan step 5 motor stepper sesuai delay.
       6) Jalankan step 6 motor stepper sesuai delay.
       7) Jalankan step 7 motor stepper sesuai delay.
       8) Jalankan step 8 motor stepper sesuai delay.
       9) Jalankan step 9 motor stepper sesuai delay.
       10)Jalankan step 10 motor stepper sesuai delay.

    END
```

Program dapat di lihat pada Lampiran 2 baris 209-337.

3.5.6 Fungsi Menampilkan Data Pada LCD Mikrokontroler *Slave*

Untuk menampilkan data pada LCD di perlukan fungsi tersendiri. Fungsi tersebut adalah sebagai berikut:

```
Algoritma :

   START

      1) Menuju posisi LCD 0,0
```

```
    2) Konversi data int menjadi ASCII
    3) Tampilkan data di LCD

END
```

Program dapat di lihat pada Lampiran 2 baris 628-630.

3.5.7 Perintah Posisi Awal

Mikrokontroler slave menjalankan perintah posisi awal sesuai dengan perintah dari mikrokontroler master

```
Algoritma :

    START

        1) Menampilkan pada lcd (“Default posisition”)
        2) Menjalankan motor Y ke arah Y- hingga menekan limit switch

    END
```

Program dapat di lihat pada Lampiran 2 baris 346-371.

3.5.8 Perintah Terima Data

Perintah terima data pada mikrokontroler slave berfungsi menerima data koordinat yang sudah di terima oleh mikrokontroler master dari PC

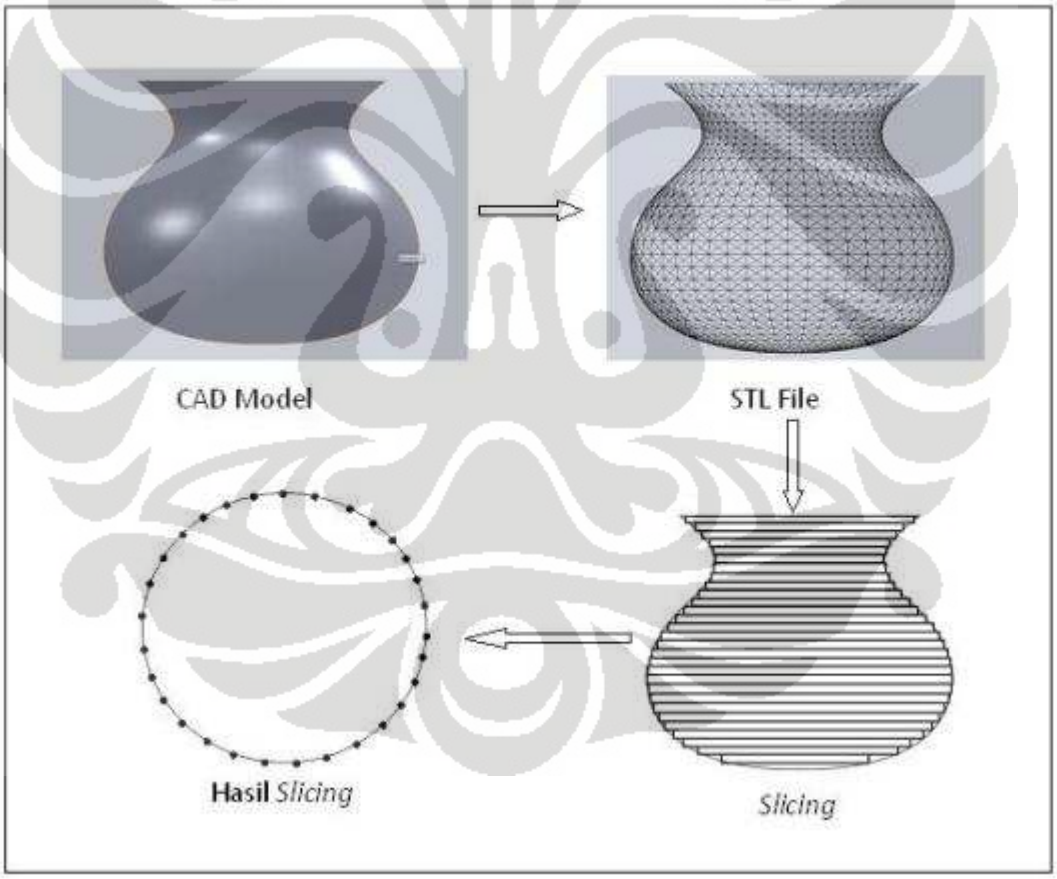
```
Algoritma :

    START

        1) Memulai loop menerima data per karakter.
        2) Merubah karakter menjadi float.
        3) Memasukan nilai float ke dalam array.
        4) Memberi feed back pada mikrokontroler master.
        5) Menampilkan koordinat Y pada lcd.
        6) Menjalankan motor sumbu Y.
        7) Memberi konfirmasi ke mikrokontroler master.
        8) Menerima data berikutnya.

    END
```

Program dapat di lihat pada Lampiran 2 baris 373-501.

BAB 4

INTERPOLASI LINGKARAN UNTUK MEMBUAT BENDA BERKONTUR

Pada prakteknya untuk menjalankan mesin *rapid prototyping* tidaklah diperlukan program untuk membuat interpolasi lingkaran karena CAD *model* pada PC langsung di ubah menjadi STL *file* kemudian *model* dalam bentuk STL *file* tersebut di-*slice* menjadi beberapa lapisan. Dari hasil perpotongan ini akan di dapat titik-titik koordinat yang nantinya akan di kirim ke mesin *rapid prototyping* sebagai koordinat tujuan.

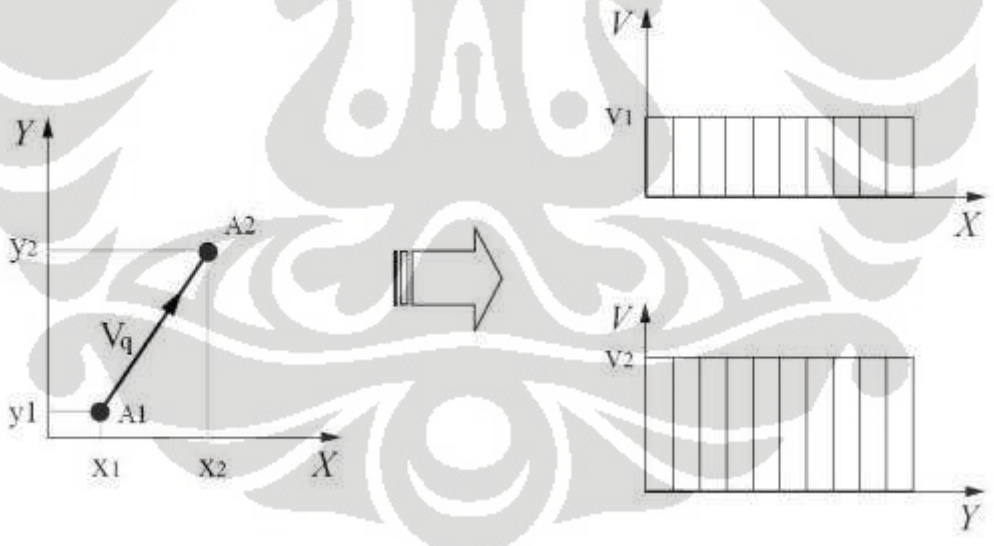


Gambar 4.1 Proses *Slicing*

Namun karena dalam penelitian ini belum digunakan program *slicer* maka untuk melakukan ujicoba membuat benda berkontur digunakan program interpolasi lingkaran.

4.1 Interpolator Garis

Untuk menjalankan dua sumbu secara bersamaan menuju suatu koordinat, pada mesin *rapid prototyping* diperlukan pengontrolan pada kecepatan motor stepper yang harus saling menyesuaikan sehingga dapat berpindah ke koordinat yang diinginkan secara akurat. Sebagai contoh, jika ada dua buah koordinat A1 dan A2, dan ingin digerakan dari koordinat A1 sebuah *extruder nozzle* mesin *rapid prototyping* ke koordinat A2 dengan kecepatan Vq maka digunakan sebuah interpolator, dimana interpolator tersebut berperan menghitung kecepatan masing-masing motor pada kedua axis, sehingga dari kecepatan kedua motor yang bergerak bersamaan didapat Vq yang diinginkan.



Gambar 4.2 Konsep Dasar Interpolator

Koordinat A1 bergerak ke koordinat A2 maka koordinat X dan Y harus bergerak bersamaan, X1 ke X2 dan Y1 ke Y2 maka motor yang menggerakan sumbu X harus memiliki kecepatan yang berkesinambungan dengan motor yang menggerakan sumbu

Y, disinilah interpolator berperan untuk mencari kecepatan motor penggerak untuk sumbu X dan Y hingga didapat Vx dan Vy

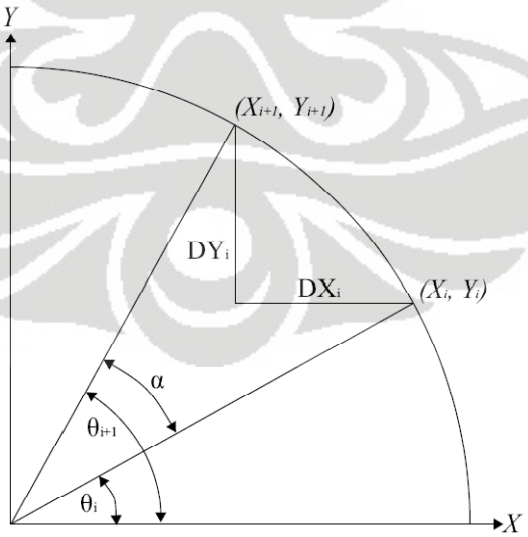
4.2 Perhitungan Interpolator Lingkaran

Pada interpolator lingkaran kecepatan yang diinterpolasi adalah kecepatan tangensial pada lingkaran, kecepatan tangensial lingkaran tersebut dikonversi menjadi kecepatan tangensial pada sumbu X dan Y.

$$V_x(t) = V \sin\theta(t) \qquad V_y(t) = V \cos\theta(t)$$

dimana $\theta(t) = (Vt/R)$

Kecepatan motor pada sumbu X dan sumbu Y dihitung dengan menggunakan interpolator lingkaran. Interpolator lingkaran membagi lingkaran menjadi segmen-segmen sesuai dengan sudut pembagi α semakin kecil α, semakin banyak segmen yang didapat maka error pada lingkaran akan semakin kecil atau lingkaran semakin sempurna.



Gambar 4.3 Interpolasi Lingkaran

Kecepatan motor pada sumbu X dan Y, untuk mencari Vx dan Vy kita ketahui

$$\cos\theta(i+1) = A\cos\theta(i) - B\sin\theta(i)$$
$$\sin\theta(i+1) = A\sin\theta(i) + B\cos\theta(i)$$

dimana

$$A = \cos\alpha \qquad B = \sin\alpha$$
$$\theta(i+1) = \theta(i) + \alpha$$

sedangkan untuk koordinat X_{i+1} dan Y_{i+1}

$$X(i+1) = R(i)\cos\theta(i+1) \qquad Y(i+1) = R(i)\sin\theta(i+1)$$

maka didapat

$$X(i+1) = AX(i) - BY(i) \qquad Y(i+1) = AY(i) + BX(i)$$

untuk menghitung DX dan DY

$$DX(i) = X(i+1) - X(i) = (A-1)X(i) - BY(i)$$
$$DY(i) = Y(i+1) - Y(i) = (A-1)Y(i) + BX(i)$$

setelah mendapatkan DX dan DY makan V_X dan V_Y dapat di hitung

$$V_x(i) = \frac{VDX(i)}{DS(i)}$$
$$V_y(i) = \frac{VDY(i)}{DS(i)}$$

dimana $DS(i) = \sqrt{DX^2(i) + DY^2(i)}$

4.3 Program Interpolasi Lingkaran Untuk Benda Berkontur

Program interpolasi lingkaran untuk benda berkontur ini dibuat dua proses yang pertama adalah program langsung di-*compile* ke dalam mikrokontroler sehingga seluruh proses perhitungan dilakukan oleh mikrokontroler. Proses yang kedua perhitungan di lakukan pada PC sehingga mikrokontroler hanya menerima data dan menjalankan motor penggerak. Untuk penggunaan sesungguhnya mikrokontroler tidak boleh diberi beban berlebih sehingga seluruh proses perhitungan haruslah dilakukan pada PC. Namun percobaan pembebanan pada mikrokontroler dilakukan sebagai pembanding untuk mengetahui perbedaan yang terjadi jika semua perhitungan dilakukan pada mikrokontroler atau dilakukan pada PC

4.3.1 Perhitungan Interpolasi Pada Mikrokontroler

Untuk interpolasi lingkaran yang langsung di *compile* ke dalam mikrokontroler, program interpolator diintegrasikan ke dalam program untuk menggerakan motor dan di-*compile* kedalam mikrokontroler sehingga motor dapat bergerak melingkar membentuk lingkaran. Karena satu motor menggunakan satu mikrokontroler maka digunakan dua mikrokontroler untuk dua motor penggerak pada sumbu X dan sumbu Y. Untuk pembuatan benda berkontur interpolator ini digabungkan lagi dengan kurva Bezier untuk menghitung jari jari lingkaran yang digunakan pada interpolator. Berikut adalah alogaritma umum yang dirancang untuk interaksi 2 mikrokontroler dalam melakukan interpolasi lingkaran untuk membuat benda berkontur:

```
Algoritma :

    START

        1) Menunggu mikrokontroler heater mengirim signal.
        2) Menjalankan extruder motor.
        3) Membuang material yang tersisa di dalam heater.
        4) Mikrokontroler master melakukan perhitungan interpolasi dan
           menghitung kurva bezier.
        5) Mengirim hasil perhitungan ke mikrokontroler slave.
        6) Mikrokontroler master menjalankan motor sumbu X.
        7) Mikrokontroler slave menjalankan motor sumbu Y.

    END
```

Perhitungan kurva Bezier digunakan untuk mencari jari jari setiap layer pada benda berkontur dan jari jari yang didapat dari perhitungan kurva Bezier akan digunakan oleh interpolator lingkaran untuk menghitung semua koordinat X dan Y untuk membuat sebuah lingkaran.

4.3.1.1 Protokol Pengiriman Data Antar Dua Mikrokontroler

Karena digunakan dua buah mikrokontroler maka proses pengiriman data adalah proses yang tidak terhindari. Maka dari itu digunakan protokol untuk menjaga agar tidak terjadi *error* dalam pengiriman data. Protokol yang digunakan adalah sebagai berikut:

```
Protokol :

    START

        1) Menunggu heater siap
        2) Mikrokontroler master menghitung koordinat motor X dan Y dengan
           interpolator.
        3) Mikrokontroler slave menunggu kiriman data sumbu X.
        4) Mikrokontroler master mengirim koordinat X kemudian mengirim
           karakter ‘x’ menandakan akhir dari data koordinat X yang di kirim.
        5) Mikrokontoler slave setelah menerima karakter ‘x’ menyimpan data
           yang telah di terima ke dalam array
        6) Mikrokontroler slave menunggu kiriman data sumbu Y
        7) Mikrokontroler master mengirim koordinat Y kemudian mengirim
           karakter ‘x’ menandakan akhir dari data koordinat Y yang di kirim.
        8) Mikrokontoler slave setelah menerima karakter ‘x’ menyimpan data
           yang telah di terima ke dalam array
        9) Mikrokontroler master menunggu konfirmasi dari mikrokontroler slave
           bahwa data siap di eksekusi
        10) Mikrokontroler slave mengirim karakter ‘y’ ke mikrokontroler master
           menandakan data siap di eksekusi
        11) Motor sumbu X dan sumbu Y dijalankan.
        12) Mikrokontroler master mengirim karakter ‘y’ ke mikrokontroler slave
           menandakan motor X selesai di jalankan kemudian menunggu
           mikrokontroler slave mengirim kembali karakter ‘y’.
        13) Mikrokontroler slave mengirim karakter ‘y’ ke mikrokontroler master
           menandakan motor Y telah selesai di jalankan dan siap menerima data
           berikutnya.
        14) Looping proses 2-7 hingga benda selesai dibentuk

    END
```

4.3.1.2 Program Interpolasi Pada Mikrokontroler Master

Mikrokontroler master melakukan semua perhitungan interpolasi kemudian mengirimkan hasil perhitungan ke mikrokontroler slave.

```
Algoritma :

    START

        1) Menunggu heater siap
        2) Menghitung koordinat motor X dan Y dengan interpolator
        3) Mengirim koordinat X dan Y ke mikrokontroler slave
        4) Melakukan perhitungan delay motor X
        5) Menunggu respon dari mikrokontroler slave bahwa motor siap
           dijalankan
        6) Menjalankan motor sumbu X.
        7) Menunggu respon dari mikrokontroler slave bahwa motor Y selesai
           dijalankan
        8) Looping proses 2-7 hingga benda selesai dibentuk

    END
```

4.3.1.3 Program Interpolasi Pada Mikrokontroler Slave

Data yang dihitung oleh mikrokontroler master diterima oleh mikrokontroler slave kemudian diproses menjadi gerakan motor.

```
Algoritma :

    START

        1) Menerima koordinat motor X dan Y dari mikrokontroler master
        2) Menghitung delay motor Y
        3) Menjalankan motor sumbu Y.
        4) Mengirim respon ke mikrokontroler master bahwa motor Y telah
           selesai dijalankan
        5) Looping proses 1-4 hingga benda selesai dibentuk

    END
```

4.3.2 Perhitungan Interpolasi Pada PC

Untuk perhitungan interpolasi pada PC program komunikasi yang digunakan sama seperti pada bab 3 namun dibuat program terpisah untuk men-*generate* koordinat lingkaran menggunakan program interpolasi yang kemudian di simpan dalam *text file*. Dari text file ini nantinya akan dibaca oleh program utama komunikasi PC-Mikrokontroler kemudian di kirim ke mikrokontroler untuk di eksekusi.

4.3.2.1 Program Interpolasi Pada PC

Program interpolasi pada PC sama halnya dengan program interpolasi yang langsung di *compile* pada mikrokontroler, hanya saja hasil perhitungan dari program interpolasi ini tidaklah langsung menjalankan motor *driver* tetapi di simpan terlebih

dahulu dalam bentuk *text file* yang kemudian akan di kirim ke mikrokontroler. Berikut adalah algoritma yang di gunakan:

```
Algoritma :

    START

        1) Menghitung koordinat X dan Y
        2) Menampilkan koordinat yang sudah dihitung.
        3) Menyimpan koordinat pada text file
        4) Menghitung koordinat Z
        5) Menghitung perbendaan jari jari lingkaran pertama dan lingkaran
           berikutnya
        6) Looping proses 1-5 hingga perhitungan selesai

    END
```

Setelah di lakukan kedua proses diatas ternyata tidak terdapat perbedaan hasil yang terlalu jelas terlihat, dikarenakan perhitungan yang dilakukan tidaklah terlalu intensif dan *spec* mikrokontroler yang digunakan cukup baik maka proses perhitungan tetap dapat dilakukan dengan baik dalam batas tertentu.

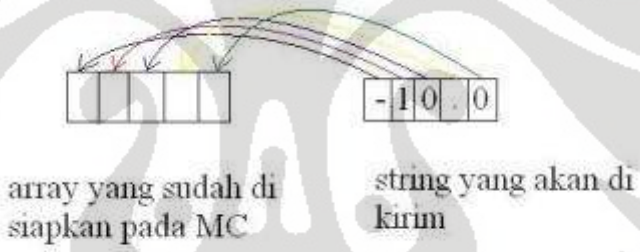
BAB 5

ANALISIS DAN PENGUJIAN PERANGKAT LUNAK

Analisa dan pengujian bertujuan untuk mengetahui keakurasian dari program yang telah di buat. Dimulai dari analisa proses pengiriman data hingga proses pengujian pada mesin *rapid prototyping*.

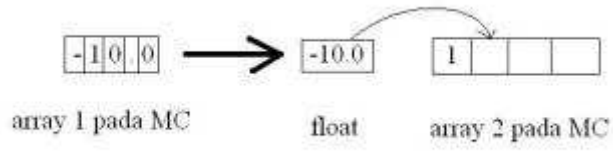
5.1 Analisa Proses Pengiriman Data

Pada proses pengiriman data, USART yang digunakan hanya dapat mengirim satu buah nilai dengan nilai maksimum 8 *bits*. Maka dari itu pengiriman yang dilakukan adalah mengirim *string* yang berisi tipe data *char* dengan nilai 1 *bytes* dengan cara seperti pada gambar 5.1.



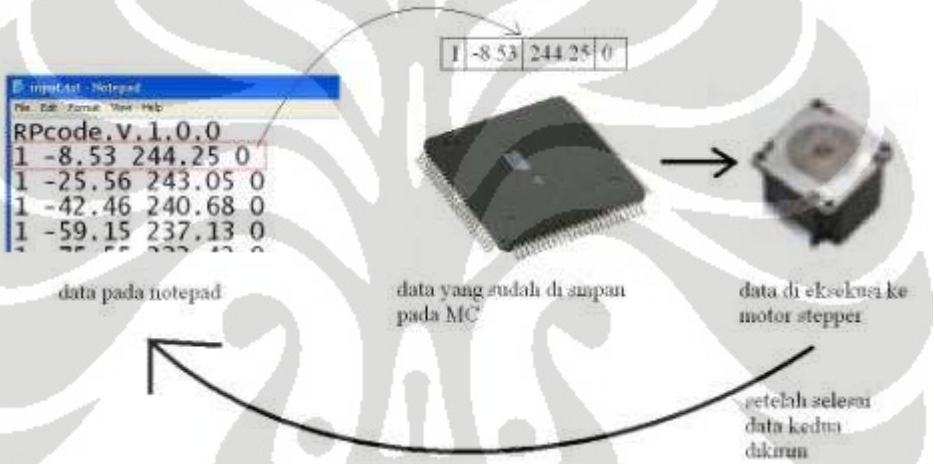
Gambar 5.1 Proses Pengiriman Data

Tipe data *char* yang berada didalam *string* dikirim satu per satu secara berurutan kemudian ditampung didalam *array* pada mikrokontroler. Setelah seluruhnya berhasil ditampung maka tipe data *string* tersebut dikonversi menjadi *float* kemudian dimasukan ke dalam *array* ke-2 sesuai format data.



Gambar 5.2 Penerimaan Data Pada Mikrokontroler

Setelah array kedua ini penuh maka data siap untuk dieksekusi ke motor penggerak. Setelah motor penggerak selesai dijalankan sesuai data, data selanjutnya siap diterima.



Gambar 5.3 Siklus Pengiriman Data

Pada proses pengiriman data seperti gambar 5.3 akan terdapat jedah setiap kali motor selesai dijalankan. Jedah yang terjadi dapat dihitung sesuai dengan banyaknya data utama yang dikirim ditambah data yang dikirim sebagai protokol dibagi *baud rate* yang digunakan pada USART ditambahkan lagi dengan waktu pengerjaan proses oleh mikrokontroler. Sebagai contoh data yang akan dikirim adalah 1 -8.53 244.25 0 banyak data adalah 13 buah karakter, pada setiap pengiriman satu karakter dikirim satu karakter lagi sebagai protokol maka ada 26 karakter yang dikirim melalui USART. Sedangkan *baud rate* yang digunakan adalah 9600 bps.

$$\frac{26}{9600} \times 1 \text{ sec} = 2.70833 \text{ ms}$$

Kemudian waktu yang digunakan oleh mikrokontroler untuk mengerjakan setiap proses adalah banyaknya proses dibagi dengan kecepatan crystal yang digunakan yaitu 16 Mhz.

$$\frac{26}{16.000.000} \times 1 \text{ sec} = 1.625 \text{ µs}$$

Maka total waktu jedah seluruhnya pada saat pengiriman data hingga eksekusi adalah:

$$2.70833 \text{ ms} + 1.625 \text{ µs} = 2.709955 \text{ ms}$$

Waktu jedah yang didapat tidaklah terlalu memiliki akibat yang berarti pada pergerakan ekstruder atau dengan kata lain tidak merusak hasil material yang dibentuk.

Metode pengiriman data, dengan mengirim data pada *notepad* sebaris demi sebaris ini dapat mengatasi data koordinat yang sangat besar, tergantung dari RAM yang dimiliki oleh PC yang digunakan, dan tidak membebani *memory* pada mikrokontroler. Hal ini dikarenakan media yang sebagai penampung data adalah PC kemudian PC mengirim sebaris data koordinat untuk dieksekusi oleh mikrokontroler, setelah mikrokontroler selesai barulah data berikutnya dikirim dan seterusnya.

5.2 Akurasi Data

Proses pengiriman data dari PC ke mikrokontroler dapat dimonitor dari *user interface* pada gambar 5.4 dan tampilan LCD pada mikrokontroler pada gambar 5.5

Gambar 5.4 Pengiriman Data Dilihat dari *User Interface* di PC

Gambar 5.5 Tampilan LCD

Data yang dikirim adalah baris pertama pada *user interface* dan data yang diterima oleh mikrokontroler sudah sesuai dengan data yang dikirim tersebut.

5.2.1 Pengujian Data

Pengiriman data dari PC ke mikrokontroler dapat dilihat akurasinya melalui tampilan LCD, namun pada prakteknya setelah data tersebut diolah oleh mikrokontroler ada factor-faktor lain yang dapat mempengaruhi data. Sehingga data tersebut harus diuji lagi keakurasiannya. Pengujian yang dilakukan adalah menyesuaikan ukuran yang diberikan dengan ukuran sesungguhnya pada mesin RP FDM, ukuran yang disesuaikan dibagi menjadi dua jenis yaitu ukuran sisi-sisi persegi dan ukuran diameter lingkaran.

Pada pengujian untuk menyesuaikan ukuran sisi-sisi persegi, data dibedakan menjadi sisi persegi pada sumbu X dan sisi persegi pada sumbu Y untuk menentukan *error* rata-rata pada kedua sumbu.

Tabel 5.1 Data Percobaan Sumbu X Dalam mm Satuan.

No	X yang Diberikan	X Aktual	*Error*	% *Error*
1	10	10	0	0
2	15	15	0	0
3	20	19.5	0.5	2.5
4	25	25	0	0
5	30	29	1	3.33

6	35	34.5	0.5	1.43
7	40	40	0	0
8	45	44.5	0.5	1.11
9	50	49.5	0.5	1
10	55	54.5	0.5	0.91
11	60	59	1	1.67
12	65	65	0	0
			Rata rata % *error*	0.99

Dari 12 data yang diambil didapat rata-rata persen *error* pada sumbu X adalah 0.99%

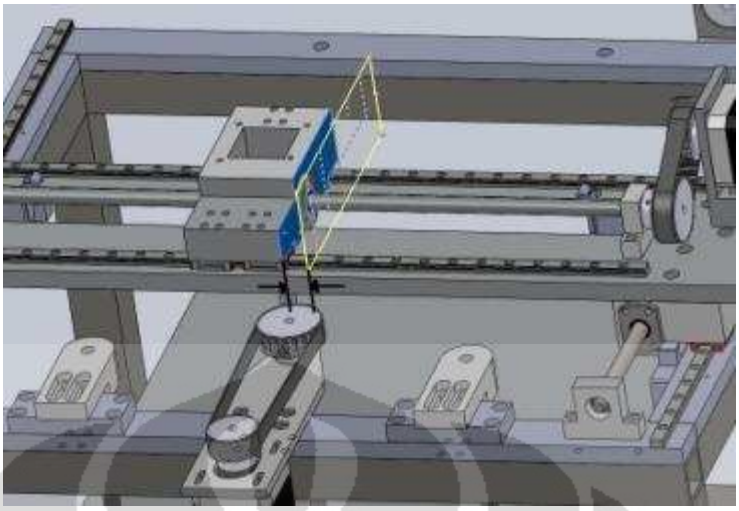
Tabel 5.2 Data Percobaan Sumbu Y Dalam mm Satuan.

No	Y yang Diberikan	Y Aktual	*Error*	% *Error*
1	10	10	0	0
2	15	15	0	0
3	20	20	0	0
4	25	25	0	0
5	30	29.5	0.5	1.67
6	35	35	0	0
7	40	39	1	2.5
8	45	44	1	2.22
9	50	49	1	2
10	55	55	0	0
11	60	60	0	0
12	65	65	0	0
			Rata rata % *error*	0.69

Dari 12 data yang diambil didapat rata-rata persen *error* pada sumbu Y adalah 0.69%

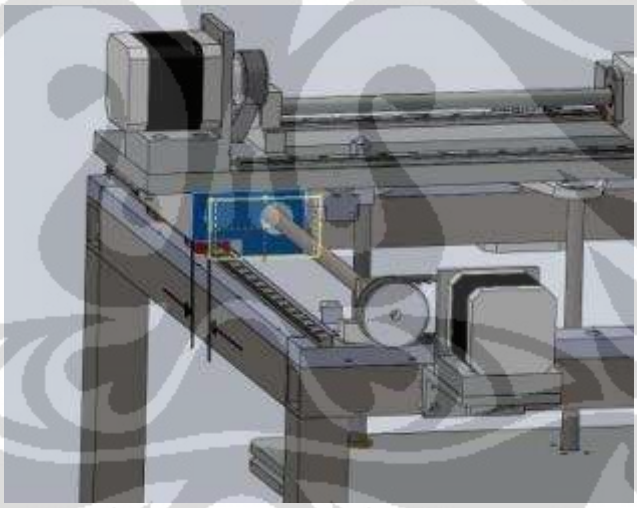
Pada kedua sumbu, yaitu sumbu X dan sumbu Y dapat dilihat *error* yang terjadi tidak melebihi 1 % dan error 1 % tersebut berkisar antara 0.5 – 1 mm.

Selain dari data di atas dilakukan juga pengukuran menggunakan CMM untuk mengukur *reapeatability* dari mesin RPFDM yaitu dengan cara menjalankan mesin sejauh 1 cm secara berulang sebanyak lima kali.



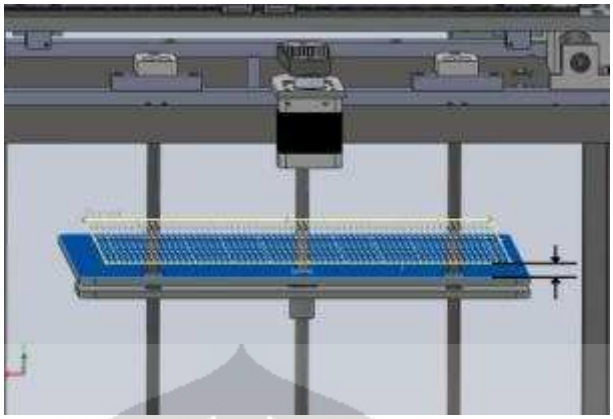
Gambar 5.6 Pengukuran Sumbu X

Untuk sumbu X diukur jarak perpindahan dari bidang yang diwarnai biru ke bidang kuning apakah tepat 1 cm seperti input yang diberikan.



Gambar 5.7 Pengukuran Sumbu Y

Untuk sumbu Y diukur jarak perpindahan dari bidang yang diwarnai biru ke bidang kuning apakah tepat 1 cm seperti input yang diberikan.



Gambar 5.8 Pengukuran Sumbu Z

Untuk sumbu Z diukur jarak perpindahan dari bidang yang diwarnai biru ke bidang kuning apakah tepat 1 cm seperti input yang diberikan.

Tabel 5.3 Hasil Pengukuran CMM

Input x	Aktual	Error	% Error
10	9.97	0.03	0.3
10	9.97	0.03	0.3
10	9.962	0.038	0.38
10	9.992	0.008	0.08
10	9.977	0.023	0.23
		Rata rata % error	0.258
Input y	**Aktual**	**Error**	**% Error**
10	9.957	0.043	0.43
10	9.996	0.004	0.04
10	9.984	0.016	0.16
10	9.952	0.048	0.48
10	9.917	0.083	0.83
		Rata rata % error	0.388
Input z	**Aktual**	**Error**	**% Error**
10	9.93	0.07	0.7
10	9.879	0.121	1.21
10	9.932	0.068	0.68
10	9.686	0.314	3.14
10	9.871	0.129	1.29
		Rata rata % error	1.404

Dari tabel dapat dilihat bahwa terdapat penyimpangan rata rata sebesar 0.258% pada sumbu X, 0.388% pada sumbu Y, dan 1.404% pada sumbu Z.

Pengujian penyesuaian ukuran diameter lingkaran bertujuan untuk mengetahui keakurasian sumbu X dan sumbu Y pada saat berjalan bersamaan menggunakan interpolasi.

Tabel 5.4 Data Percobaan Diameter Lingkaran Dalam mm Satuan.

No	Diameter yang Diberikan	Diameter Aktual	Error	% Error
1	10	9	1	10
2	20	19	1	5
3	30	29	1	3.33
4	40	39	1	2.5
5	45	44	1	2.22
6	50	49	1	2
7	55	54	1	1.82
8	60	59	1	1.67
9	65	64	1	1.54
10	70	69	1	1.43
			Rata rata % error	3.15

Dari 10 data yang diambil didapat rata-rata persen *error* pada diameter lingkaran yang dibentuk adalah 3.15%, *error* sebesar 3.15% ini memiliki nilai penyimpangan sebesar 1 mm.

Dari data diatas didapat penyimpangan yang terjadi pada mesin RP FDM adalah sebesar 0.5 – 1 mm. Besar penyimpangan ini lah yang menentukan keakurasian dari mesin RP FDM. Penyimpangan ini diakibatkan oleh dua hal yaitu:

1. Terdapat ketidaksesuaian nilai pada saat jenis data *string* di ubah menjadi *float* dikarenakan adanya ketidaksempurnaan alogaritma pengubah *string to float* pada *compiler*, hal ini dapat diatasi dengan merubah fungsi *string to float* tersebut dengan algoritma yang lebih baik.
2. Hal lain yang menyebabkan penyimpangan tersebut adalah pembulatan data *float* menjadi *integer* pada saat akan di konversi menjadi putaran motor *stepper* dikarenakan motor *stepper* tidak dapat melakukan pergerakan *half step*, hal ini dapat diatasi dengan membuat alogaritma *half step* untuk motor *stepper* yang digunakan. *Error* ini dapat dilihat pada subbab *error*.

5.2.2 *Error* Akibat Perubahan Nilai *Float* ke *Integer*

Pembulatan nilai *float* menjadi *integer* pada saat koordinat yang dikirim dikonversi menjadi putaran motor dapat mengakibatkan *error* yang terakumulasi. Untuk mengetahui seberapa besar *error* yang terjadi diambilah koordinat seperempat lingkaran dari hasil program interpolator, kemudian diambil koordinat salah satu sumbu, yaitu sumbu X. Kemudian koordinat sumbu X ini dirubah menjadi putaran motor. Dikarenakan motor *stepper* tidak dapat menerima nilai *float* maka nilai tersebut dibulatkan. Setelah itu, nilai koordinat yang dikirim, nilai perubahan ke putaran motor, dan nilai pembulatannya dijumlahkan, untuk membandingkan seberapa besar penyimpangan yang terjadi akibat pembulatan.

Tabel 5.5 Data Pembulatan Putaran Motor

Koordinat yang dikirim (0.01mm)	Jumlah teoritis step putaran motor	Pembulatan step putaran motor (aktual)		Koordinat yang dikirim (0.01mm)	Jumlah teoritis step putaran motor	Pembulatan step putaran motor (aktual)
-0.61	-0.19	0		-25.55	-7.98	-8
-1.83	-0.57	-1		-26.36	-8.24	-8
-3.04	-0.95	-1		-27.14	-8.48	-8
-4.26	-1.33	-1		-27.90	-8.72	-9
-5.46	-1.71	-2		-28.61	-8.94	-9
-6.67	-2.08	-2		-29.29	-9.15	-9
-7.86	-2.46	-2		-29.94	-9.36	-9
-9.04	-2.83	-3		-30.55	-9.55	-10
-10.21	-3.19	-3		-31.12	-9.72	-10
-11.37	-3.55	-4		-31.65	-9.89	-10
-12.52	-3.91	-4		-32.15	-10.05	-10
-13.65	-4.27	-4		-32.61	-10.19	-10
-14.76	-4.61	-5		-33.02	-10.32	-10
-15.86	-4.96	-5		-33.40	-10.44	-10
-16.94	-5.29	-5		-33.73	-10.54	-11
-17.99	-5.62	-6		-34.03	-10.63	-11
-19.02	-5.95	-6		-34.28	-10.71	-11
-20.04	-6.26	-6		-34.49	-10.78	-11
-21.02	-6.57	-7		-34.66	-10.83	-11
-21.98	-6.87	-7		-34.79	-10.87	-11
-22.92	-7.16	-7		-34.87	-10.90	-11
-23.82	-7.44	-7		-34.91	-10.91	-11
-24.70	-7.72	-8	Total	-1000.63	-312.70	-314

Setelah dijumlahkan didapat jumlah koordinat yang dikirim sebesar -1000.63 (0.01mm), jumlah putaran motor sebesar -312.70, dan jumlah putaran motor yang sudah dibulatkan sebesar -314 (tanda (–) menunjukan arah putaran motor). Maka dapat dihitung terdapat selisih 1.3 putaran pada saat sebelum dan sesudah dibulatkan, jika putaran motor yang sudah dibulatkan ini dikembalikan lagi menjadi jarak koordinat maka didapat jumlah koordinatnya adalah 1004.8 (0.01mm). Error yang terjadi sebesar 4.17 dalam 0.01 mm atau sebesar 0.0417 mm. Persen error yang terjadi adalah

$$^{4.17}/_{1000.63} \times 100\% = 0.42\%$$

5.2.3 Pengujian Benda Berkontur

Setelah dilakukan pengujian lingkaran dilakukan pembuatan benda berkontur.

Gambar 5.9 Kendi

Gambar 5.10 Cawan

Gambar 5.11 Vas

Gambar 5.12 Kubah

Dari Gambar 5.9 – Gambar 5.11 dapat dilihat bahwa terdapat ketidak-simetrisan antara kontur sisi kanan dan sisi kiri. Hal ini juga diakibatkan karena kitidaksesuaian nilai pada saat dilakukan konversi dari jenis data *string* menjadi *float*. Selain itu juga dikarenakan pembulatan data *float* menjadi *integer* pada saat akan di konversi menjadi putaran motor *stepper*. Hal yang paling mempengaruhi adalah penggunaan nilai *phi* pada program interpolator lingkaran, ketidaksimetrisan diatas dikarenakan konversi sudut derajat menjadi sudut radian. Dikarenakan perhitungan pada program tidak dapat menggunakan sudut derajat namun harus menggunakan sudut radian. Sedangkan sudut radian ini memerlukan nilai *phi* yang spesifik, ketidaksimetrisan diatas diakibatkan penggunaan *phi*=22/7 setelah nilai *phi* dirubah menjadi 3.1415926 didapat bentuk yang simetris, dapat di lihat pada Gambar 5.12.

BAB 6

KESIMPULAN DAN SARAN PENELITIAN LEBIH LANJUT

Kesimpulan dan saran lebih lanjut untuk penelitian berikutnya berdasarkan penelitian yang sudah di lakukan adalah sebagai berikut:

6.1 Kesimpulan

1. Program untuk mengendalikan mesin *rapid prototyping* dapat mengirim dan menerima data dengan cepat.
2. Keakurasian mesin RP FDM memiliki penyimpangan sebesar 0.5 – 1 mm
3. Program sudah dapat mengendalikan mesin *rapid prototyping* sesuai perintah yang di berikan pengguna, dan menjalankan motor sesuai koordinat yang di berikan.

6.2 Saran Penelitian Lebih Lanjut

Untuk pengembangan perangkat lunak *rapid prototyping* lebih lanjut disarankan untuk melakukan beberapa perubahan untuk meningkatkan kinerja sistem yaitu:

1. Mengganti algoritma motor *stepper* dengan alogaritma yang dapat mendukung *half step* sehingga motor *stepper* dapat bergerak lebih presisi, untuk keperluan pembuatan benda dengan akurasi yang tinggi.
2. Memaksimalkan penggunaan *interrupt* pada program sehingga limit switch dapat berfungsi sebagaimana seharusnya.
3. Memaksimalkan algoritma untuk mengirim dan menerima data sehingga data dalam jumlah banyak dapat sekaligus di kirim.
4. Membuat *user interface* dengan java atau bahasa pemrograman lainnya sehingga tampilan *user interface* dapat dibuat lebih menarik.

DAFTAR REFERENSI

[1] Allaboutcircuits. (2010). *Embedded Systems and Microcontrollers*. From http://forum.allaboutcircuits.com/forumdisplay.php?f=17, 27 November 2010.

[2] ATMEL. (2009). *ATMega2560 Microcontroller PDF*. From http://www.atmel.com/dyn/products/product_card.asp?family_id=607&family_name=AVR+8%2DBit+RISC+&part_id=2014, 3 April 2009.

[3] Codevision. (2009). *Codevision AVR C compilier*. From http://www.hpinfotech.ro/html/cvavr.htm, 10 Januari 2009.

[4] Custompartnet. (2010). *Additive Fabrication*. From http://www.custompartnet.com/, 22 November 2010

[5] Dietel & Dietel. (2003). *How to Program in C. 3rd edition*. New Jersey: Prentice Hall.

[6] Jones. (2010). *Stepper Motor Types*. From http://www.cs.uiowa.edu/~jones/step/types.html, 22 November 2010.

[7] Mmattera. (2010). *G Code: Increment, Absolute*. From http://www.mmattera.com/g-code/abs-inc.html, 22 November 2010.

[8] Pololu. (2009). *USB to Serial Adaptor*. From http://www.pololu.com/catalog/product/391, 27 Maret 2009.

[9] Stetz, Kyle. (2010). *Rapid Prototyping Study*. From http://kylestetzrp.wordpress.com/, 22 November 2010.

[10] Suh, Suk-Hwan. (2008). *Theory and Design of CNC Systems*. New York: Springer.

[11] Wankhede, Mahesh. (2007). *Switch On I/O Ports*. From http://www.freewebs.com/maheshwankhede/ports.html, 20 Augustus 2009.

[12] Wikipedia. (2010). *Communications Protocol*. From http://en.wikipedia.org/wiki/Communications_protocol, 22 November 2010.

[13] Kamrani, Ali K. , Nasr, Emad Abouel. (2005). *Rapid Prototyping: Theory and Practice*. New York: Springer.

LAMPIRAN 1

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.4 Standard
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Program Mikrokontroler Master
Version : 1.0.0
Date : 10/19/2010
Author : Andry Sulaiman
Company : University of Indonesia
Comments :


Chip type : ATmega2560
Program type : Application
Clock frequency : 16.000000 MHz
Memory model : Small
External RAM size : 0
Data Stack size : 2048
*****************************************************/

#include <mega2560.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <math.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x08 ;PORTC
#endasm
#include <lcd.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
```

```
25  #define DATA_OVERRUN (1<<OVR)
26  #define DATA_REGISTER_EMPTY (1<<UDRE)
27  #define RX_COMPLETE (1<<RXC)
28
29  // Get a character from the USART2 Receiver
30  #pragma used+
31  char getchar2(void)
32  {
33  char status,data;
34  while (1)
35        {
36        while (((status=UCSR2A) & RX_COMPLETE)==0);
37        data=UDR2;
38        if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
39          return data;
40        };
41  }
42  #pragma used-
43
44  // Write a character to the USART2 Transmitter
45  #pragma used+
46  void putchar2(char c)
47  {
48  while ((UCSR2A & DATA_REGISTER_EMPTY)==0);
49  UDR2=c;
50  }
51  #pragma used-
52
53  // Standard Input/Output functions
54  #include <stdio.h>
55
56  // Declare your global variables here
57
58
59  void cw_motor_1(int x)
60  {
61     //PORTA.1 = input 1
62     //PORTA.2 = input 2
63     //PORTA.3 = input 3
64     //PORTA.4 = input 4
65     //PORTA.5 = input 5
66  int us;
67  //step 1
68
69     PORTA.1 = 1;
70     PORTA.2 = 1;
71     PORTA.3 = 0;
72     PORTA.4 = 0;
```

```
73        PORTA.5 = 1;
74       for(us=0;us<=x;us++)
75        {
76        delay_us(1);
77        }
78
79     //step 2
80
81        PORTA.1 = 1;
82        PORTA.2 = 1;
83        PORTA.3 = 0;
84        PORTA.4 = 0;
85        PORTA.5 = 0;
86        for(us=0;us<=x;us++)
87        {
88        delay_us(1);
89        }
90
91     //step 3
92
93        PORTA.1 = 1;
94        PORTA.2 = 1;
95        PORTA.3 = 1;
96        PORTA.4 = 0;
97        PORTA.5 = 0;
98        for(us=0;us<=x;us++)
99        {
100       delay_us(1);
101       }
102
103    //step 4
104
105       PORTA.1 = 0;
106       PORTA.2 = 1;
107       PORTA.3 = 1;
108       PORTA.4 = 0;
109       PORTA.5 = 0;
110      for(us=0;us<=x;us++)
111       {
112       delay_us(1);
113       }
114
115    //step 5
116
117       PORTA.1 = 0;
118       PORTA.2 = 1;
119       PORTA.3 = 1;
120       PORTA.4 = 1;
```

```
121       PORTA.5 = 0;
122      for(us=0;us<=x;us++)
123       {
124       delay_us(1);
125       }
126     //step 6
127
128       PORTA.1 = 0;
129       PORTA.2 = 0;
130       PORTA.3 = 1;
131       PORTA.4 = 1;
132       PORTA.5 = 0;
133       for(us=0;us<=x;us++)
134       {
135       delay_us(1);
136       }
137
138     //step 7
139
140       PORTA.1 = 0;
141       PORTA.2 = 0;
142       PORTA.3 = 1;
143       PORTA.4 = 1;
144       PORTA.5 = 1;
145       for(us=0;us<=x;us++)
146       {
147       delay_us(1);
148       }
149
150     //step 8
151
152       PORTA.1 = 0;
153       PORTA.2 = 0;
154       PORTA.3 = 0;
155       PORTA.4 = 1;
156       PORTA.5 = 1;
157      for(us=0;us<=x;us++)
158       {
159       delay_us(1);
160       }
161
162     //step 9
163
164       PORTA.1 = 1;
165       PORTA.2 = 0;
166       PORTA.3 = 0;
167       PORTA.4 = 1;
168       PORTA.5 = 1;
```

```
169       for(us=0;us<=x;us++)
170       {
171       delay_us(1);
172       }
173
174    //step 10
175
176       PORTA.1 = 1;
177       PORTA.2 = 0;
178       PORTA.3 = 0;
179       PORTA.4 = 0;
180       PORTA.5 = 1;
181       for(us=0;us<=x;us++)
182       {
183       delay_us(1);
184       }
185
186
187    }
188
189    void ccw_motor_1(int x)
190    {
191       //PORTA.1 = input 1
192       //PORTA.2 = input 2
193       //PORTA.3 = input 3
194       //PORTA.4 = input 4
195       //PORTA.5 = input 5
196    int us;
197    //step 1
198
199       PORTA.1 = 1;
200       PORTA.2 = 0;
201       PORTA.3 = 0;
202       PORTA.4 = 0;
203       PORTA.5 = 1;
204       for(us=0;us<=x;us++)
205       {
206       delay_us(1);
207       }
208
209    //step 2
210
211       PORTA.1 = 1;
212       PORTA.2 = 0;
213       PORTA.3 = 0;
214       PORTA.4 = 1;
215       PORTA.5 = 1;
216       for(us=0;us<=x;us++)
```

```
217     {
218     delay_us(1);
219     }
220
221   //step 3
222
223     PORTA.1 = 0;
224     PORTA.2 = 0;
225     PORTA.3 = 0;
226     PORTA.4 = 1;
227     PORTA.5 = 1;
228     for(us=0;us<=x;us++)
229     {
230     delay_us(1);
231     }
232
233   //step 4
234
235     PORTA.1 = 0;
236     PORTA.2 = 0;
237     PORTA.3 = 1;
238     PORTA.4 = 1;
239     PORTA.5 = 1;
240     for(us=0;us<=x;us++)
241     {
242     delay_us(1);
243     }
244
245   //step 5
246
247     PORTA.1 = 0;
248     PORTA.2 = 0;
249     PORTA.3 = 1;
250     PORTA.4 = 1;
251     PORTA.5 = 0;
252     for(us=0;us<=x;us++)
253     {
254     delay_us(1);
255     }
256
257   //step 6
258
259     PORTA.1 = 0;
260     PORTA.2 = 1;
261     PORTA.3 = 1;
262     PORTA.4 = 1;
263     PORTA.5 = 0;
264     for(us=0;us<=x;us++)
```

```
265       {
266       delay_us(1);
267       }
268
269     //step 7
270
271       PORTA.1 = 0;
272       PORTA.2 = 1;
273       PORTA.3 = 1;
274       PORTA.4 = 0;
275       PORTA.5 = 0;
276       for(us=0;us<=x;us++)
277       {
278       delay_us(1);
279       }
280
281     //step 8
282
283       PORTA.1 = 1;
284       PORTA.2 = 1;
285       PORTA.3 = 1;
286       PORTA.4 = 0;
287       PORTA.5 = 0;
288       for(us=0;us<=x;us++)
289       {
290       delay_us(1);
291       }
292
293     //step 9
294
295       PORTA.1 = 1;
296       PORTA.2 = 1;
297       PORTA.3 = 0;
298       PORTA.4 = 0;
299       PORTA.5 = 0;
300      for(us=0;us<=x;us++)
301       {
302       delay_us(1);
303       }
304
305     //step 10
306
307       PORTA.1 = 1;
308       PORTA.2 = 1;
309       PORTA.3 = 0;
310       PORTA.4 = 0;
311       PORTA.5 = 1;
312       for(us=0;us<=x;us++)
```

```
313     {
314     delay_us(1);
315     }
316
317 }
318
319 void cw_motor_3(int x) //mendekati motor
320 {
321     //PORTE.1 = input 1 blue   PORTE.2
322     //PORTE.2 = input 2 red    PORTE.3
323     //PORTE.3 = input 3 orange P\ORTE.4
324     //PORTE.4 = input 4 green  PORTE.5
325     //PORTE.5 = input 5 black  PORTE.6
326
327 //step 1
328
329     PORTE.2 = 1;
330     PORTE.4 = 0;
331     PORTE.6 = 1;
332     PORTE.3 = 1;
333     PORTE.5 = 0;
334     delay_ms(x);
335
336 //step 2
337
338     PORTE.2 = 1;
339     PORTE.4 = 0;
340     PORTE.6 = 0;
341     PORTE.3 = 1;
342     PORTE.5 = 0;
343     delay_ms(x);
344
345 //step 3
346
347     PORTE.2 = 1;
348     PORTE.4 = 1;
349     PORTE.6 = 0;
350     PORTE.3 = 1;
351     PORTE.5 = 0;
352     delay_ms(x);
353
354 //step 4
355
356     PORTE.2 = 0;
357     PORTE.4 = 1;
358     PORTE.6 = 0;
359     PORTE.3 = 1;
360     PORTE.5 = 0;
```

```
361         delay_ms(x);
362
363     //step 5
364
365         PORTE.2 = 0;
366         PORTE.4 = 1;
367         PORTE.6 = 0;
368         PORTE.3 = 1;
369         PORTE.5 = 1;
370         delay_ms(x);
371
372     //step 6
373
374         PORTE.2 = 0;
375         PORTE.4 = 1;
376         PORTE.6 = 0;
377         PORTE.3 = 0;
378         PORTE.5 = 1;
379         delay_ms(x);
380
381     //step 7
382
383         PORTE.2 = 0;
384         PORTE.4 = 1;
385         PORTE.6 = 1;
386         PORTE.3 = 0;
387         PORTE.5 = 1;
388         delay_ms(x);
389
390     //step 8
391
392         PORTE.2 = 0;
393         PORTE.4 = 0;
394         PORTE.6 = 1;
395         PORTE.3 = 0;
396         PORTE.5 = 1;
397         delay_ms(x);
398
399     //step 9
400
401         PORTE.2 = 1;
402         PORTE.4 = 0;
403         PORTE.6 = 1;
404         PORTE.3 = 0;
405         PORTE.5 = 1;
406         delay_ms(x);
407
408     //step 10
```

```
409
410       PORTE.2 = 1;
411       PORTE.4 = 0;
412       PORTE.6 = 1;
413       PORTE.3 = 0;
414       PORTE.5 = 0;
415       delay_ms(x);
416    }
417
418    void ccw_motor_3(int x) //menjauhi motor
419    {
420       //PORTE.1 = input 1  PORTE.2
421       //PORTE.2 = input 2  PORTE.3
422       //PORTE.3 = input 3  PORTE.4
423       //PORTE.4 = input 4  PORTE.5
424       //PORTE.5 = input 5  PORTE.6
425
426    //step 1
427
428       PORTE.2 = 1;
429       PORTE.4 = 0;
430       PORTE.6 = 1;
431       PORTE.3 = 0;
432       PORTE.5 = 0;
433       delay_ms(x);
434
435    //step 2
436
437       PORTE.2 = 1;
438       PORTE.4 = 0;
439       PORTE.6 = 1;
440       PORTE.3 = 0;
441       PORTE.5 = 1;
442       delay_ms(x);
443    //step 3
444
445       PORTE.2 = 0;
446       PORTE.4 = 0;
447       PORTE.6 = 1;
448       PORTE.3 = 0;
449       PORTE.5 = 1;
450       delay_ms(x);
451    //step 4
452
453       PORTE.2 = 0;
454       PORTE.4 = 1;
455       PORTE.6 = 1;
456       PORTE.3 = 0;
```

```
457         PORTE.5 = 1;
458         delay_ms(x);
459
460     //step 5
461
462         PORTE.2 = 0;
463         PORTE.4 = 1;
464         PORTE.6 = 0;
465         PORTE.3 = 0;
466         PORTE.5 = 1;
467         delay_ms(x);
468
469     //step 6
470
471         PORTE.2 = 0;
472         PORTE.4 = 1;
473         PORTE.6 = 0;
474         PORTE.3 = 1;
475         PORTE.5 = 1;
476         delay_ms(x);
477
478     //step 7
479
480         PORTE.2 = 0;
481         PORTE.4 = 1;
482         PORTE.6 = 0;
483         PORTE.3 = 1;
484         PORTE.5 = 0;
485         delay_ms(x);
486
487     //step 8
488
489         PORTE.2 = 1;
490         PORTE.4 = 1;
491         PORTE.6 = 0;
492         PORTE.3 = 1;
493         PORTE.5 = 0;
494         delay_ms(x);
495
496     //step 9
497
498         PORTE.2 = 1;
499         PORTE.4 = 0;
500         PORTE.6 = 0;
501         PORTE.3 = 1;
502         PORTE.5 = 0;
503         delay_ms(x);
504
```

```
505    //step 10
506
507       PORTE.2 = 1;
508       PORTE.4 = 0;
509       PORTE.6 = 1;
510       PORTE.3 = 1;
511       PORTE.5 = 0;
512       delay_ms(x);
513    }
514    void motor_stop()
515    {
516       PORTA = 0x00;
517       PORTC = 0x00;
518       PORTE = 0x00;
519    }
520
521    void posisi_awal()
522    {
523    int i;
524
525    putchar2('A');
526
527    lcd_clear();
528    lcd_gotoxy(0,0);
529    lcd_putsf("Default Position");
530
531
532     //sumbu X+
533       for (i=1;i<=10000;i++)
534       {
535          if (PIND.0==0)
536          {
537             motor_stop();
538             break;
539          }
540          else
541          {
542             cw_motor_1(210); //menjauhi motor
543             //ccw_motor_1(); //mendekati motor
544          }
545       }
546
547    //sumbu Z+
548       for (i=1;i<=10000;i++)
549       {
550          if (PIND.4==0)
551          {
552             motor_stop();
```

```
553             break;
554         }
555         else
556         {
557             cw_motor_3(1); //mendekati motor
558         }
559     }
560
561
562 }
563
564 void jalan_program()
565 {
566
567 int i;
568 int j;
569 int m;
570
571
572 float xy;
573 float xy1;
574 float k;
575 float ddx;
576 float kirimx;
577 float kirimy;
578 float kirimg;
579 float kirimz;
580 float InputList [4];
581
582 float delay;
583 char kirimy1[16];
584 char kirimx1[16];
585 char kirimg1[16];
586 char kirimz1[16];
587 char response;
588 char input[16];
589
590 putchar2('B');
591
592 while (1)
593  {
594
595 for(j = 0; j <4; j++) //untuk mengambil angka perbaris
596         {
597             lcd_clear();
598             lcd_gotoxy(0,0);
599
600             //Untuk MC2 dari sini
```

```
601             i = 0;
602             input[i] = getchar();
603             while(input[i] != 'x')
604             {
605                //lcd_putchar(input[i]);
606                i++;
607                input[i] = getchar();
608             }
609             input[i] = '\0';
610
611
612             InputList[j] = atof(input);
613             //Kayanya sampe sini (yang atas), tapi ga pake inputList, cukup variabel int
614    biasa
615             putchar ('k');
616             //delay_ms(1000);
617
618             //delay_ms(1000);
619          }
620
621          lcd_clear();
622
623          putchar('n');
624          //lcd_putsf("Processing...");
625
626
627          //delay_ms(5000); //Lakukan proses terhadap motor, kirim inputList-nya (index ke-
628    1 = x, 2 = y, 3 = z)
629          kirimg=InputList[0];
630          kirimx=InputList[1]/3.2;
631          kirimy=InputList[2]/3.2;
632          kirimz=InputList[3]/3.2;
633
634          xy=(kirimx*kirimx)+(kirimy*kirimy);
635          xy1=sqrt (xy);
636          ddx=((xy1/kirimx)*190) ;
637    //       ddy=((xy1/kirimy)*1) ;
638          ftoa(kirimg,2,kirimg1);
639          ftoa(InputList[1],2,kirimy1);
640          ftoa(InputList[2],2,kirimx1);
641          ftoa(InputList[3],2,kirimz1);
642          for (m=0;m<strlen(kirimx1);m++)
643          {
644          lcd_gotoxy(m,0);
645          lcd_putchar(kirimx1[m]);
646          }
647          for (m=0;m<strlen(kirimy1);m++)
648          {
```

```
649         lcd_gotoxy(m,1);
650         lcd_putchar(kirimy1[m]);
651         }
652         for (m=0;m<strlen(kirimz1);m++)
653         {
654         lcd_gotoxy(m,2);
655         lcd_putchar(kirimz1[m]);
656         }
657         //delay_ms (5000);
658
659         for(j=0 ; j<strlen(kirimg1); j++)
660         {
661         putchar2  (kirimg1[j]); //tergantung usart yang digunakan
662         }
663         putchar2('x');
664
665         for(j=0 ; j<strlen(kirimx1); j++)
666         {
667         putchar2  (kirimx1[j]); //tergantung usart yang digunakan
668         }
669         putchar2('x');
670
671        // delay_ms(10);
672
673
674         for(j=0 ; j<strlen(kirimy1); j++)
675         {
676         putchar2  (kirimy1[j]); //tergantung usart yang digunakan
677         }
678         putchar2('x');
679
680
681         response=getchar2();
682
683         while(response != 'y')
684         {
685         lcd_putsf("Waiting...\n");
686         response = getchar2();
687         }
688
689         if (ddx<0)
690            {
691            delay=ddx*(-1);
692
693               if(kirimx < 0)
694               {
695
696               k = kirimx*(-1);
```

```
697
698
699                 for (i=1;i<=k;i++)
700                 {
701                 ccw_motor_1(delay);
702
703                 }
704               }
705               else
706               {
707                 for (i=1;i<=kirimx;i++)
708
709                 {
710                 cw_motor_1(delay);
711                 }
712               }
713             }
714             else
715             {
716              if(kirimx < 0)
717               {
718
719               k = kirimx*(-1);
720
721
722                 for (i=1;i<=k;i++)
723                 {
724                 ccw_motor_1(ddx);
725
726                 }
727               }
728               else
729               {
730                 for (i=1;i<=kirimx;i++)
731
732                 {
733                 cw_motor_1(ddx);
734                 }
735               }
736             }
737
738             putchar2('y');
739             response=getchar2();
740             while(response != 'y')
741             {
742             lcd_putsf("Waiting...\n");
743             response = getchar2();
744             }
```

```
745             if(kirimz>0)
746             {
747
748             for (i=0;i<=kirimz;i++)
749             {
750             ccw_motor_3(1);
751             }
752
753             }
754             putchar('y');
755
756
757      }
758     }
759
760     void main(void)
761     {
762     char perintah;
763     char temperatur;
764     // Declare your local variables here
765
766
767     // Crystal Oscillator division factor: 1
768     #pragma optsize-
769     CLKPR=0x80;
770     CLKPR=0x00;
771     #ifdef _OPTIMIZE_SIZE_
772     #pragma optsize+
773     #endif
774
775     // Input/Output Ports initialization
776     // Port A initialization
777     // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
778     Func0=Out
779     // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
780     PORTA=0x00;
781     DDRA=0xFF;
782
783     // Port B initialization
784     // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
785     // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
786     PORTB=0x00;
787     DDRB=0x00;
788
789     // Port C initialization
790     // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
791     Func0=Out
792     // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
```

```
793   PORTC=0x00;
794   DDRC=0xFF;
795
796   // Port D initialization
797   // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
798   Func0=Out
799   // State7=1 State6=1 State5=1 State4=1 State3=1 State2=1 State1=1 State0=1
800   PORTD=0xFF;
801   DDRD=0xFF;
802
803   // Port E initialization
804   // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
805   // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
806   PORTE=0x00;
807   DDRE=0x7C;
808
809   // Port F initialization
810   // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
811   Func0=Out
812   // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
813   PORTF=0xFF;
814   DDRF=0xFF;
815
816   // Port G initialization
817   // Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
818   // State5=T State4=T State3=T State2=T State1=T State0=T
819   PORTG=0x00;
820   DDRG=0x00;
821
822   // Port H initialization
823   // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
824   // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
825   PORTH=0x00;
826   DDRH=0x00;
827
828   // Port J initialization
829   // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
830   Func0=Out
831   // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
832   PORTJ=0x7C;
833   DDRJ=0x7C;
834
835   // Port K initialization
836   // Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
837   Func0=Out
838   // State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
839   PORTK=0x00;
840   DDRK=0xFF;
```

```
841
842     // Port L initialization
843     // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
844     // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
845     PORTL=0x00;
846     DDRL=0x00;
847
848
849
850     // USART0 initialization
851     // Communication Parameters: 8 Data, 1 Stop, No Parity
852     // USART0 Receiver: On
853     // USART0 Transmitter: On
854     // USART0 Mode: Asynchronous
855     // USART0 Baud Rate: 9600
856     UCSR0A=0x00;
857     UCSR0B=0x18;
858     UCSR0C=0x06;
859     UBRR0H=0x00;
860     UBRR0L=0x67;
861
862     // USART2 initialization
863     // Communication Parameters: 8 Data, 1 Stop, No Parity
864     // USART2 Receiver: On
865     // USART2 Transmitter: On
866     // USART2 Mode: Asynchronous
867     // USART2 Baud Rate: 9600
868     UCSR2A=0x00;
869     UCSR2B=0x18;
870     UCSR2C=0x06;
871     UBRR2H=0x00;
872     UBRR2L=0x67;
873
874     // Analog Comparator initialization
875     // Analog Comparator: Off
876     // Analog Comparator Input Capture by Timer/Counter 1: Off
877     ACSR=0x80;
878     ADCSRB=0x00;
879
880     // LCD module initialization
881     lcd_init(20);
882
883
884
885         // Place your code here
886
887           temperatur=getchar2();
888           while(temperatur != 'z')
```

```
889         {
890          lcd_clear();
891         lcd_gotoxy(0,0);
892         lcd_putsf("Waiting...\n");
893
894
895         temperatur = getchar2();
896         }
897         putchar('z');
898
899         while (1)
900         {
901         perintah = getchar ();
902
903         switch(perintah)
904
905         {
906         case 'A' :
907         putchar ('A');
908         posisi_awal();
909         break;
910         case 'B' :
911         jalan_program();
912         break;
913         }
914         putchar('z');
915         }
916
917     }
```

LAMPIRAN 2

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.4 Standard
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Program Mikrokontroler Slave
Version : 1.0.0
Date : 10/19/2010
Author : Andry Sulaiman
Company : University of Indonesia
Comments :


Chip type : ATmega2560
Program type : Application
Clock frequency : 16.000000 MHz
Memory model : Small
External RAM size : 0
Data Stack size : 2048
*****************************************************/

#include <mega2560.h>
#include <stdlib.h>
#include <delay.h>
#include <string.h>
#include <math.h>

// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x02 ;PORTA
#endasm
#include <lcd.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
```

```
25  #define RX_COMPLETE (1<<RXC)
26
27  // Get a character from the USART2 Receiver
28  #pragma used+
29  char getchar2(void)
30  {
31  char status,data;
32  while (1)
33        {
34        while (((status=UCSR2A) & RX_COMPLETE)==0);
35        data=UDR2;
36        if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
37          return data;
38        };
39  }
40  #pragma used-
41
42  // Write a character to the USART2 Transmitter
43  #pragma used+
44  void putchar2(char c)
45  {
46  while ((UCSR2A & DATA_REGISTER_EMPTY)==0);
47  UDR2=c;
48  }
49  #pragma used-
50
51  // Get a character from the USART3 Receiver
52  #pragma used+
53  char getchar3(void)
54  {
55  char status,data;
56  while (1)
57        {
58        while (((status=UCSR3A) & RX_COMPLETE)==0);
59        data=UDR3;
60        if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
61          return data;
62        };
63  }
64  #pragma used-
65
66  // Write a character to the USART3 Transmitter
67  #pragma used+
68  void putchar3(char c)
69  {
70  while ((UCSR3A & DATA_REGISTER_EMPTY)==0);
71  UDR3=c;
72  }
```

```
#pragma used-

// Standard Input/Output functions
#include <stdio.h>

// Declare your global variables here

void cw_motor_2(int x)
{
   //PORTC.1 = input 1
   //PORTC.2 = input 2
   //PORTC.3 = input 3
   //PORTC.4 = input 4
   //PORTC.5 = input 5
int us;
//step 1

   PORTC.1 = 1;
   PORTC.2 = 1;
   PORTC.3 = 0;
   PORTC.4 = 0;
   PORTC.5 = 1;
    for(us=0;us<=x;us++)
   {
   delay_us(1);
   }

//step 2

   PORTC.1 = 1;
   PORTC.2 = 1;
   PORTC.3 = 0;
   PORTC.4 = 0;
   PORTC.5 = 0;
   for(us=0;us<=x;us++)
   {
   delay_us(1);
   }

//step 3

   PORTC.1 = 1;
   PORTC.2 = 1;
   PORTC.3 = 1;
   PORTC.4 = 0;
   PORTC.5 = 0;
  for(us=0;us<=x;us++)
   {
```

```
121       delay_us(1);
122       }
123     //step 4
124
125       PORTC.1 = 0;
126       PORTC.2 = 1;
127       PORTC.3 = 1;
128       PORTC.4 = 0;
129       PORTC.5 = 0;
130       for(us=0;us<=x;us++)
131       {
132       delay_us(1);
133       }
134
135     //step 5
136
137       PORTC.1 = 0;
138       PORTC.2 = 1;
139       PORTC.3 = 1;
140       PORTC.4 = 1;
141       PORTC.5 = 0;
142       for(us=0;us<=x;us++)
143       {
144       delay_us(1);
145       }
146
147     //step 6
148
149       PORTC.1 = 0;
150       PORTC.2 = 0;
151       PORTC.3 = 1;
152       PORTC.4 = 1;
153       PORTC.5 = 0;
154      for(us=0;us<=x;us++)
155       {
156       delay_us(1);
157       }
158
159     //step 7
160
161       PORTC.1 = 0;
162       PORTC.2 = 0;
163       PORTC.3 = 1;
164       PORTC.4 = 1;
165       PORTC.5 = 1;
166       for(us=0;us<=x;us++)
167       {
168       delay_us(1);
```

```
169     }
170
171  //step 8
172
173     PORTC.1 = 0;
174     PORTC.2 = 0;
175     PORTC.3 = 0;
176     PORTC.4 = 1;
177     PORTC.5 = 1;
178     for(us=0;us<=x;us++)
179     {
180     delay_us(1);
181     }
182
183  //step 9
184
185     PORTC.1 = 1;
186     PORTC.2 = 0;
187     PORTC.3 = 0;
188     PORTC.4 = 1;
189     PORTC.5 = 1;
190     for(us=0;us<=x;us++)
191     {
192     delay_us(1);
193     }
194
195  //step 10
196
197     PORTC.1 = 1;
198     PORTC.2 = 0;
199     PORTC.3 = 0;
200     PORTC.4 = 0;
201     PORTC.5 = 1;
202     for(us=0;us<=x;us++)
203     {
204     delay_us(1);
205     }
206
207  }
208
209  void ccw_motor_2(int x)
210  {
211     //PORTC.1 = input 1
212     //PORTC.2 = input 2
213     //PORTC.3 = input 3
214     //PORTC.4 = input 4
215     //PORTC.5 = input 5
216  int us;
```

```
217    //step 1
218
219        PORTC.1 = 1;
220        PORTC.2 = 0;
221        PORTC.3 = 0;
222        PORTC.4 = 0;
223        PORTC.5 = 1;
224        for(us=0;us<=x;us++)
225        {
226        delay_us(1);
227        }
228
229    //step 2
230
231        PORTC.1 = 1;
232        PORTC.2 = 0;
233        PORTC.3 = 0;
234        PORTC.4 = 1;
235        PORTC.5 = 1;
236        for(us=0;us<=x;us++)
237        {
238        delay_us(1);
239        }
240
241    //step 3
242
243        PORTC.1 = 0;
244        PORTC.2 = 0;
245        PORTC.3 = 0;
246        PORTC.4 = 1;
247        PORTC.5 = 1;
248         for(us=0;us<=x;us++)
249        {
250        delay_us(1);
251        }
252
253    //step 4
254
255        PORTC.1 = 0;
256        PORTC.2 = 0;
257        PORTC.3 = 1;
258        PORTC.4 = 1;
259        PORTC.5 = 1;
260        for(us=0;us<=x;us++)
261        {
262        delay_us(1);
263        }
264
```

```
265     //step 5
266
267         PORTC.1 = 0;
268         PORTC.2 = 0;
269         PORTC.3 = 1;
270         PORTC.4 = 1;
271         PORTC.5 = 0;
272         for(us=0;us<=x;us++)
273         {
274         delay_us(1);
275         }
276
277     //step 6
278
279         PORTC.1 = 0;
280         PORTC.2 = 1;
281         PORTC.3 = 1;
282         PORTC.4 = 1;
283         PORTC.5 = 0;
284         for(us=0;us<=x;us++)
285         {
286         delay_us(1);
287         }
288
289     //step 7
290
291         PORTC.1 = 0;
292         PORTC.2 = 1;
293         PORTC.3 = 1;
294         PORTC.4 = 0;
295         PORTC.5 = 0;
296         for(us=0;us<=x;us++)
297         {
298         delay_us(1);
299         }
300
301     //step 8
302
303         PORTC.1 = 1;
304         PORTC.2 = 1;
305         PORTC.3 = 1;
306         PORTC.4 = 0;
307         PORTC.5 = 0;
308          for(us=0;us<=x;us++)
309         {
310         delay_us(1);
311         }
312
```

```
313     //step 9
314
315        PORTC.1 = 1;
316        PORTC.2 = 1;
317        PORTC.3 = 0;
318        PORTC.4 = 0;
319        PORTC.5 = 0;
320        for(us=0;us<=x;us++)
321        {
322        delay_us(1);
323        }
324
325     //step 10
326
327        PORTC.1 = 1;
328        PORTC.2 = 1;
329        PORTC.3 = 0;
330        PORTC.4 = 0;
331        PORTC.5 = 1;
332        for(us=0;us<=x;us++)
333        {
334        delay_us(1);
335        }
336
337     }
338
339     void motor_stop()
340     {
341        PORTA = 0x00;
342        PORTC = 0x00;
343        PORTE = 0x00;
344     }
345
346     void posisi_awal()
347     {
348
349        int i;
350
351        lcd_clear();
352        lcd_gotoxy(0,0);
353        lcd_putsf("Default Position");
354
355
356
357        //sumbu Y-
358        for (i=1;i<=10000;i++)
359        {
360           if (PINE.3==0)
```

```
361         {
362            motor_stop();
363            break;
364         }
365         else
366         {
367            ccw_motor_2(210); //mendekati motor
368         }
369      }
370
371   }
372
373   void terima_data ()
374   {
375   int i;
376   int j,m;
377
378
379   char input[16];
380   float xy;
381   float xy1;
382   float k;
383   float ddy;
384   //float ddx;
385   float kirimx;
386   float kirimy;
387
388   float list [3];
389   char response;
390   char kirimy1[16];
391   //char kirimx1[16];
392
393   float delay;
394
395   while(1)
396     {
397    for(j = 0; j <3; j++)
398         {
399            lcd_clear();
400            lcd_gotoxy(0,0);
401            i = 0;
402            input[i] = getchar2();
403            while(input[i] != 'x')
404            {
405               lcd_putchar(input[i]);
406               i++;
407               input[i] = getchar2();
408            }
```

```
409             input[i] = '\0';
410             //delay_ms(1000);
411
412
413             //delay_ms(1000);
414             list[j] = atof(input);
415
416
417         }
418
419         if (list[0]==1)
420         {
421         putchar3('y');
422         }
423         else
424         {
425         putchar3('u');
426         }
427
428         kirimx=list[1];
429         kirimy=list[2];
430
431 //        ftoa (kirimx,3,kirimx1);
432         ftoa (kirimy,3,kirimy1);
433         for (m=0;m<strlen(kirimy1);m++)
434         {
435         lcd_gotoxy(m,0);
436         lcd_putchar(kirimy1[m]);
437         }
438         xy=(kirimx*kirimx)+(kirimy*kirimy);
439         xy1=sqrt (xy);
440         ddy=((xy1/kirimy)*190) ;
441
442
443         putchar2('y');
444
445
446
447         if (ddy<0)
448         {
449         delay=ddy *(-1);
450             if(kirimy < 0)
451             {
452
453             k = kirimy *(-1);
454
455
456                 for (i=1;i<=k;i++)
```

```
457                 {
458                 ccw_motor_2(delay);
459                 }
460             }
461             else
462             {
463                 for (i=1;i<=kirimy;i++)
464
465                 {
466                 cw_motor_2(delay);
467                 }
468             }
469         }
470         else
471         {
472             if(kirimy < 0)
473             {
474
475             k = kirimy*(-1);
476
477
478                 for (i=1;i<=k;i++)
479                 {
480                 ccw_motor_2(ddy);
481                 }
482             }
483             else
484             {
485                 for (i=1;i<=kirimy;i++)
486
487                 {
488                 cw_motor_2(ddy);
489                 }
490             }
491         }
492
493         response=getchar2();
494         while(response != 'y')
495                 {
496                 lcd_putsf("Waiting...\n");
497                 response = getchar2();
498                 }
499         putchar2('y');
500     }
501 }
502
503
504 void main(void)
```

```
{
char perintah;
char temperatur;

// Declare your local variables here

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0xFF;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0xFF;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0xFF;
DDRD=0xFF;

// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTE=0x00;
DDRE=0x00;

// Port F initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
```

```
553     PORTF=0xFF;
554     DDRF=0xFF;
555
556     // Port G initialization
557     // Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
558     // State5=T State4=T State3=T State2=T State1=T State0=T
559     PORTG=0x00;
560     DDRG=0x00;
561
562     // Port H initialization
563     // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
564     // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
565     PORTH=0x00;
566     DDRH=0x00;
567
568     // Port J initialization
569     // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
570     // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
571     PORTJ=0xFF;
572     DDRJ=0xFF;
573
574     // Port K initialization
575     // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
576     // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
577     PORTK=0x00;
578     DDRK=0xFF;
579
580     // Port L initialization
581     // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
582     // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
583     PORTL=0x00;
584     DDRL=0x00;
585
586     // USART0 initialization
587     // Communication Parameters: 8 Data, 1 Stop, No Parity
588     // USART0 Receiver: On
589     // USART0 Transmitter: On
590     // USART0 Mode: Asynchronous
591     // USART0 Baud Rate: 9600
592     UCSR0A=0x00;
593     UCSR0B=0x18;
594     UCSR0C=0x06;
595     UBRR0H=0x00;
596     UBRR0L=0x67;
597
598     // USART2 initialization
599     // Communication Parameters: 8 Data, 1 Stop, No Parity
600     // USART2 Receiver: On
```

```
// USART2 Transmitter: On
// USART2 Mode: Asynchronous
// USART2 Baud Rate: 9600
UCSR2A=0x00;
UCSR2B=0x18;
UCSR2C=0x06;
UBRR2H=0x00;
UBRR2L=0x67;

// USART3 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART3 Receiver: On
// USART3 Transmitter: On
// USART3 Mode: Asynchronous
// USART3 Baud Rate: 9600
UCSR3A=0x00;
UCSR3B=0x18;
UCSR3C=0x06;
UBRR3H=0x00;
UBRR3L=0x67;

// LCD module initialization
lcd_init(16);

temperatur=getchar();
    while(temperatur != 'c')
    {
    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_putsf("Waiting...\n");
    temperatur = getchar();
    }
    putchar2('z');

while (1)
    {
    // Place your code here
    perintah = getchar2();
    switch(perintah)
    {
    case 'A' :
    posisi_awal();
    break;
    case 'B' :
    terima_data();
    break;
    }
    };
```

LAMPIRAN 3

```
/*****************************************************

Project      : Program Interface PC
Version      : 1.0.0
Date         : 11/8/2010
Author       : Andry Sulaiman
Company      : University of Indonesia
Comments     :

*****************************************************/
/*Header-Library*/
#include<dos.h>
#include<stdio.h>
#include<conio.h>
#include<string.h>

//Mendefinisikan COM1 sebagai default
#define PORT1 0x3F8

/*
Inisialisasi PORT
COM1 0x3F8
COM2 0x2F8
COM3 0x3E8
COM4 0x2E8
*/

//Variabel Declaration Global

//Function Open Port Pada Serial Data
void open_port_serial()
{
outportb(PORT1 + 1, 0);       //Matikan Fungsi Interupt pd PORT1

/*PORT1 - Setting Komunikasi pada PORT1*/
outportb(PORT1 + 3, 0x80);   //Set DLAB ON
outportb(PORT1 + 0, 0x0C);   /*Set Bautrate 9600bps - Divisior Latch Low Byte*/

/*
Default Bautrate Setting Registry
Default 0x03 = 38.400bps
0x01 = 115.000bps
0x02 = 56.700bps
0x06 = 19.200bps
0x0C = 9.600bps
0x18 = 4.800bps
```

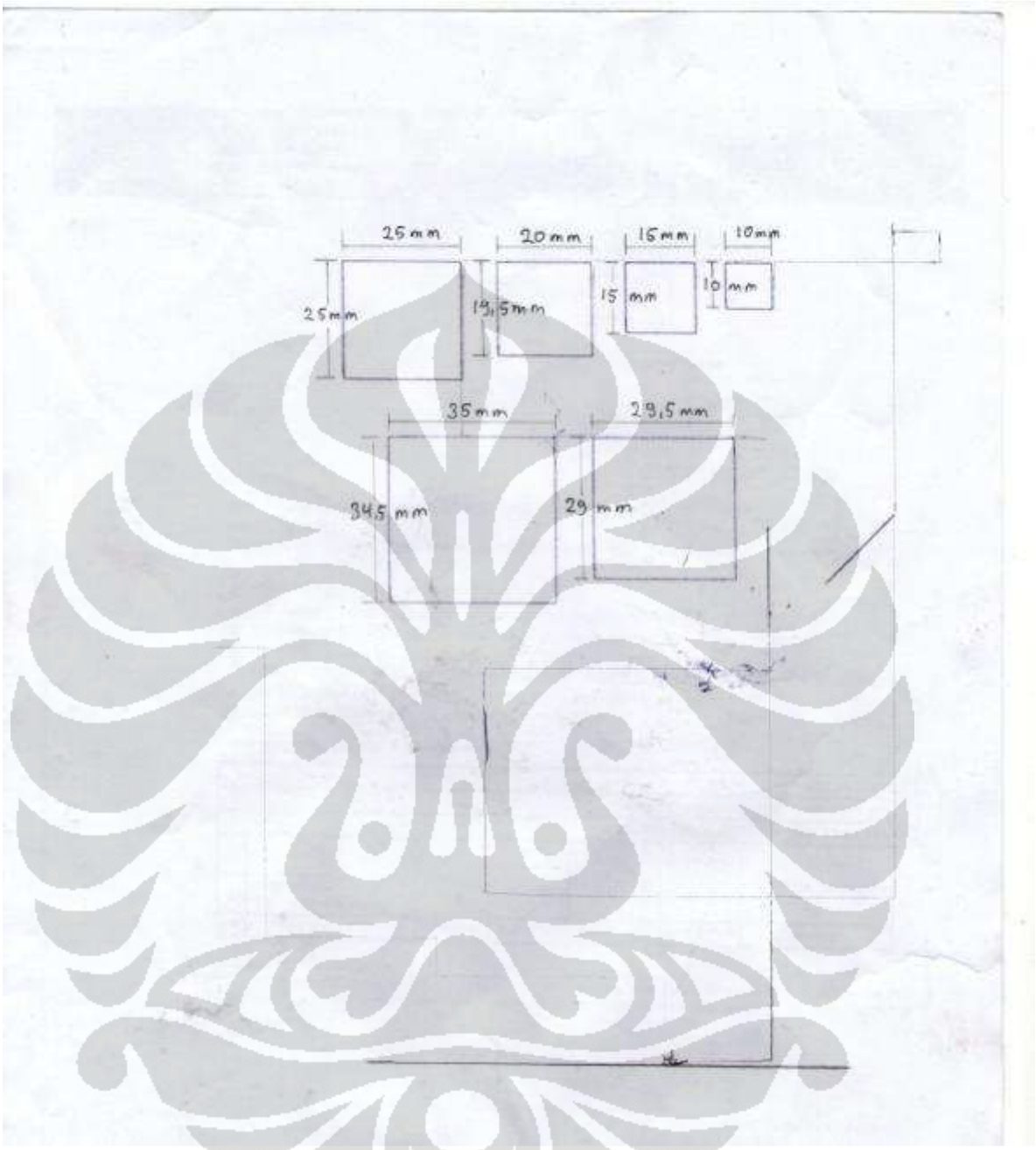
```
0x30 = 2.400bps
*/

outportb(PORT1 + 1, 0x00);  /*Set Bautrate - Divisior Latch High Byte*/
outportb(PORT1 + 3, 0x03);  //8bit,No Parity,1 Stop Bit
outportb(PORT1 + 2, 0xC7); //FIFO Control Registry
outportb(PORT1 + 4, 0x0B); //Turn on DTR, RTS,and OUT2
}

void get_serial_data()
 {
    int c;
    int counter;
    char buffer[100];
    while(1)
    {
            c = inportb(PORT1 + 5);
            if(c & 1)
            {
                    buffer[counter] = inport(PORT1);
                    printf("%c", buffer[counter]);

                    }
                    if(kbhit())
                    {
                                    break; } } }

void kirim_data()
 {
   int i, j, k;
   int v=1;
         char loop;
         char my_string[500];
         char response;
         char szKey[] = "RPcode.V.1.0.0";
         char temp[5];
         FILE *in;

         clrscr();
         flushall();
         open_port_serial();
         outportb(PORT1, 'B');

         in = fopen("input.txt", "r");
         if(in == NULL)
         {
                 printf("File tidak bisa dibuka!");
                 getch();
```

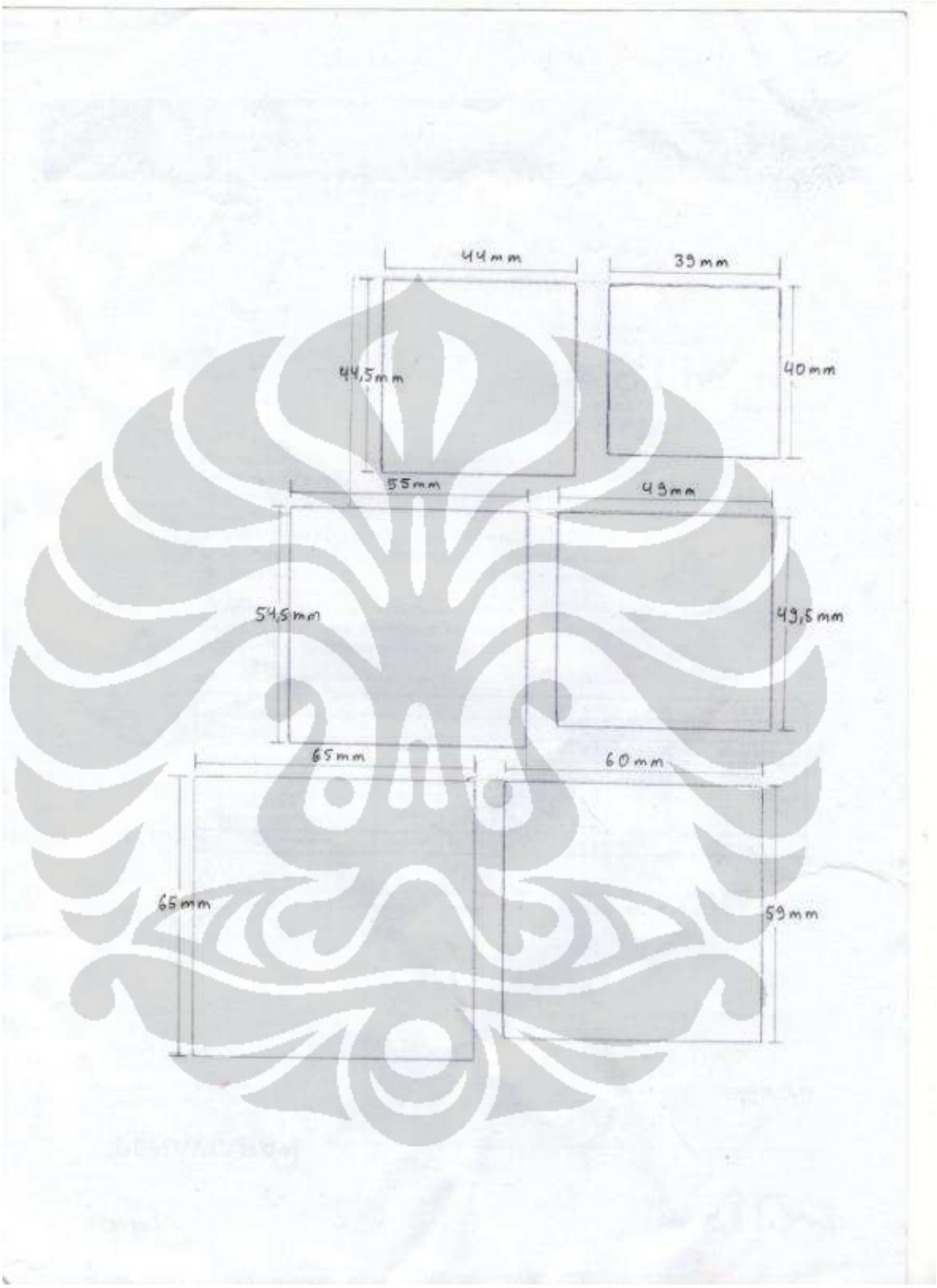
```
85            }
86            else
87            {
88          fscanf(in, "%s", &my_string);
89          printf("first line of : %s\n", my_string);
90          if(strcmp(szKey,my_string)==0)
91          {
92
93                        while(!feof(in)) //ini nanti while tidak feof
94                        {
95
96                                  printf("Data ke %d: ", v);
97                    v++;
98                                  for(i = 0; i <4; i++)
99                                  {
100                                 //printf("Turn %d: ", (i + 1));
101
102                                       fscanf(in, "%s", &temp);
103
104                                            for(j = 0; j < strlen(temp); j++)
105                                            {
106                                                 outportb(PORT1, temp[j]);
107                                                 putchar(temp[j]);
108                                            }
109                           outportb(PORT1, 'x');
110
111                     putchar(' ');
112                                       loop = inportb(PORT1);
113                                       while(loop != 'k')
114                                            {
115                                 //printf("Waiting...\n");
116                               loop = inportb(PORT1);
117                                            }
118                                 //delay(10);
119                       }
120                                  putchar ('\n');
121
122                                 response = inportb(PORT1);
123                       //   printf("reponse: %c\n", response);
124
125                           while(response != 'y')
126                                       {
127                                 //printf("Waiting...\n");
128                                            response = inportb(PORT1);
129                                       }
130               }
131         }
132         else
```

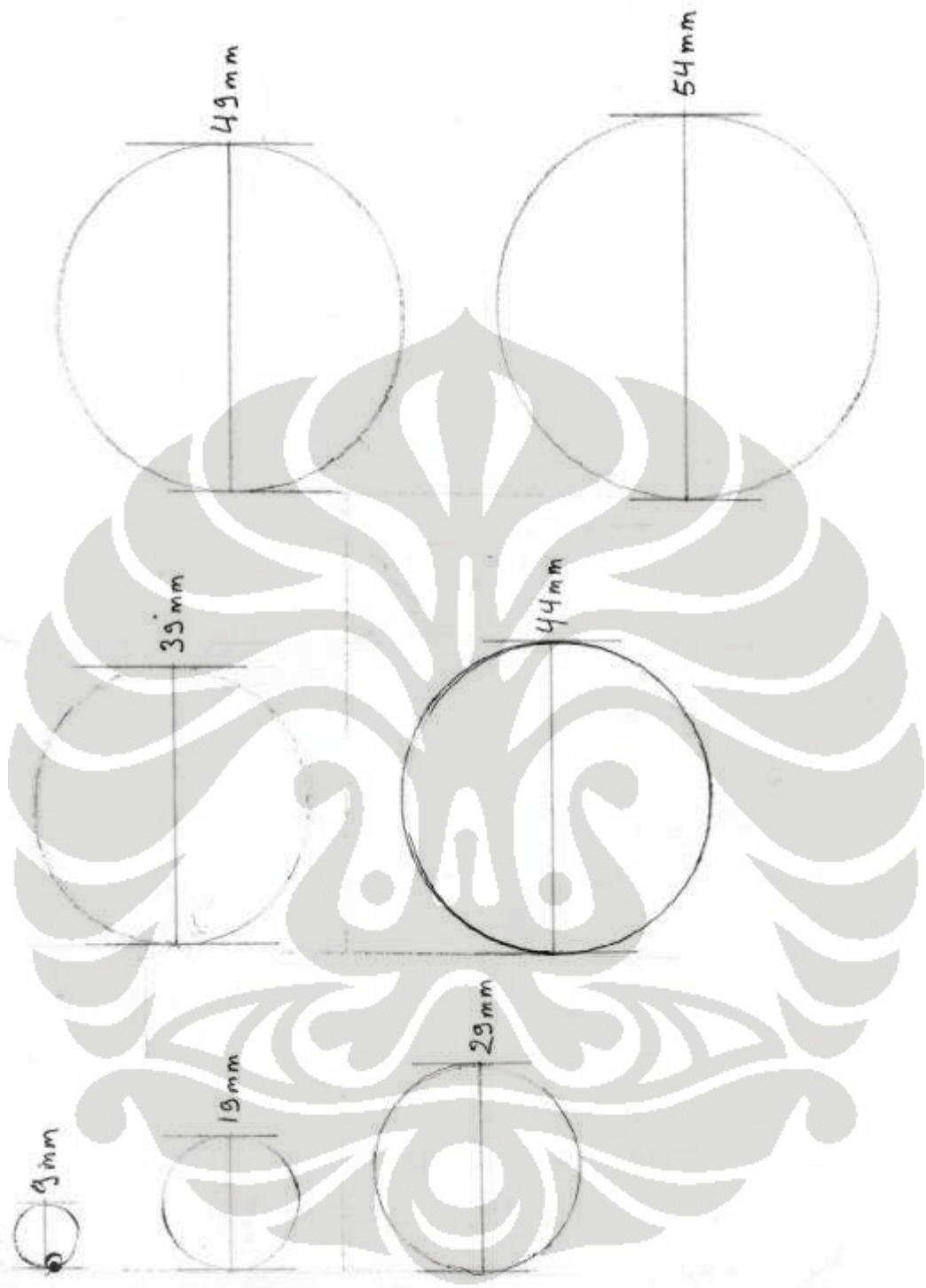
```
133         {
134            printf("Versi data tidak cocok");
135            getch();
136         }
137
138           }
139
140           fclose(in);
141           printf("Reading Finish.");
142           getch();
143    }
144
145
146   void posisi_awal ()
147
148   {
149   char terima;
150            while(1)
151           {
152           open_port_serial();
153           outportb(PORT1, 'A');
154           terima=inportb(PORT1);
155           if(terima=='A')
156           {
157           break;
158           }
159           }
160   }
161   void exit_program()
162       {
163       /*Fungsi untuk keluar dari program*/
164       exit(0);
165       }
166
167   //Main Function
168   void main ()
169   {
170    while(1)
171           {
172           int select_mode;
173           char suhu;
174           open_port_serial();
175           suhu = inportb(PORT1);
176
177
178                          while(suhu != 'z')
179                          {
180                                 printf("Menunggu heater siap\n");
```

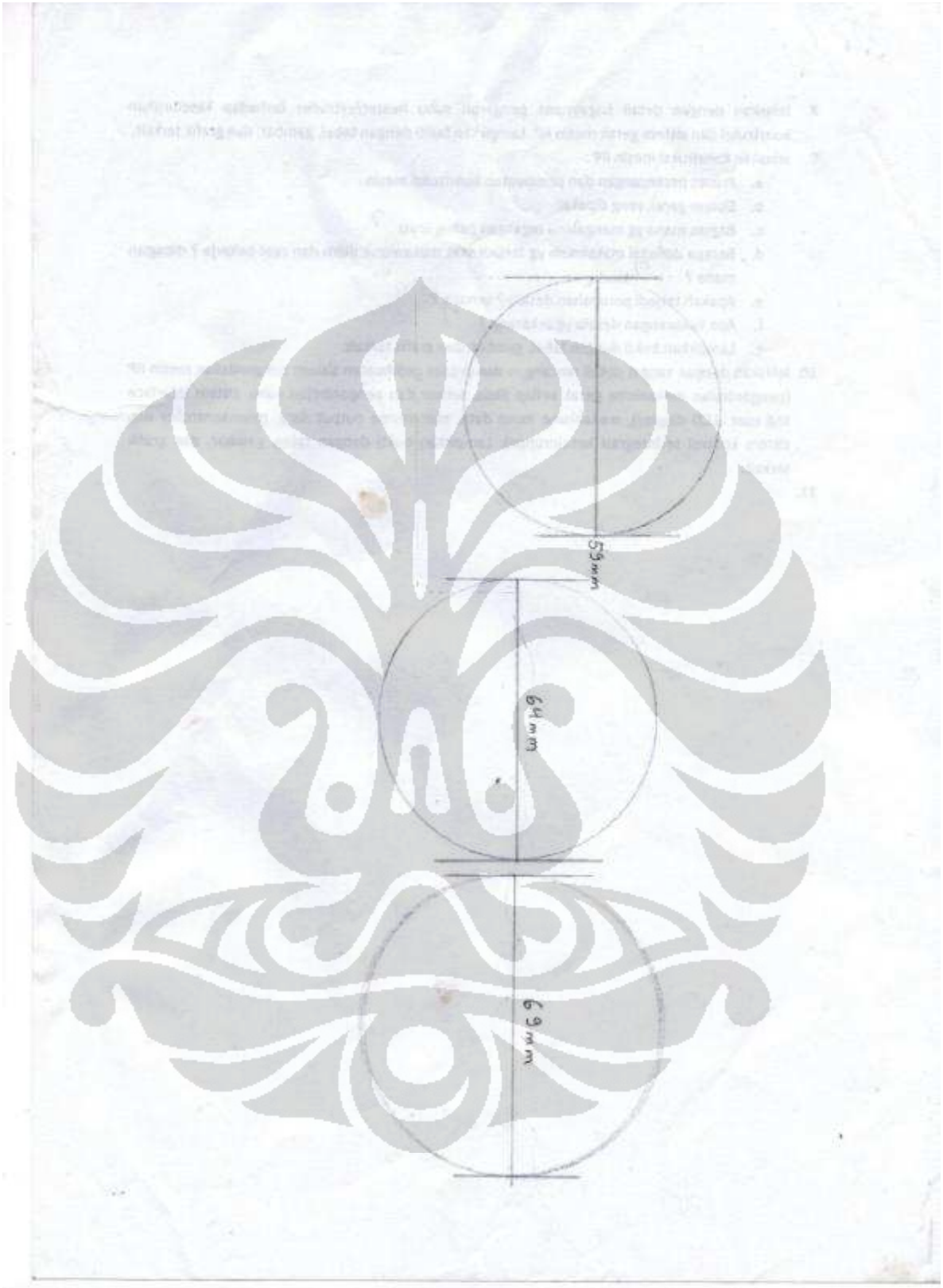
```
                                    suhu = inportb(PORT1);
                            }
        printf("Heater siap\n");
        delay (2500);



        clrscr();
        printf("Rapid Prototyping\n");
        printf("Select Mode\n");
        printf("1. Posisi Awal\n");
        printf("2. Kirim Data\n");
        printf("3. Exit\n");
        scanf("%d", &select_mode);
        switch(select_mode)
          {
            case 1 :
            posisi_awal();
            break;
            case 2 :
            kirim_data();
            break;
            case 3 :
            exit_program();
            break;
            default:
            printf("Invalid Mode Selection\n");
          }
        getch();
        }
```

LAMPIRAN 4


25 mm
20 mm
15 mm
10 mm
25 mm
19,5 mm
15 mm
10 mm
35 mm
29,5 mm
34,5 mm
29 mm


44 mm
39 mm
44,5 mm
40 mm
55 mm
49 mm
54,5 mm
49,5 mm
65 mm
60 mm
65 mm
59 mm


49mm
54mm
39mm
44mm
29mm
19mm
9mm


53mm
64mm
69mm