UNIVERSITAS INDONESIA

PENINGKATAN KINERJA MESIN *RAPID PROTOTYPING* BERBASIS *FUSED DEPOSITION MODELLING*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

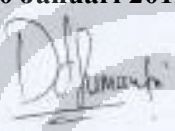
Dede Sumantri
0706266954

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK MESIN
DEPOK
JANUARI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama	**: Dede Sumantri**
NPM	**: 0706266954**
Tanda Tangan	**: 20 Januari 2012**
Tanggal	**:**



HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Dede Sumantri
NPM : 0706266954
Program Studi : Teknik Mesin
Judul Skripsi : Peningkatan Kinerja Mesin *Rapid Prototyping* Berbasis *Fused Deposition Modelling*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian 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. Danardhono A.S, DEA PE ( )

Penguji : .Ir. Henky S. Nugroho, MT. ( )

Penguji : Ir. Bambang P. Prianto, MIKomp ( )

Ditetapkan di : Depok
Tanggal : 20 Januari 2012

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, yang telah memberikan kekuatan, kesabaran dan ketenangan yang lebih dan atas rahmat-Nya sehingga 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) Keluarga terutama orang tua yang telah memberikan dukungan moril dan materiil yang begitu besar
2) Dr. Ir. Gandjar Kiswanto, M.Eng selaku dosen pembimbing
3) Dewan penguji yang telah memberikan evaluasi dan saran dalam skripsi ini
4) Dr. Ir.Harinaldi, M.Eng selaku kepala Departemen Teknik Mesin;
5) Teman-teman di Lab Manufaktur, Rendi Kurniawan, Andri Sulaiman, Jediel Billy R, Teguh Santoso, Anton Royanto, Agus Siswanto, Ferdi Bastian, Alvis yang telah membantu dan memberi saran kepada penulis dalam mengerjakan skripsi ini;
6) Setiani Anjarwirasti, S.Psi yang selalu memberi semangat untuk menyelesaikan tugas akhir ini secepatnya
7) Mas Yasin selaku karyawan Departemen Teknik Mesin yang telah membantu dalam skripsi ini
8) Teman-teman yang telah menemani penulis menghabiskan waktu luang dalam kesempitan

Akhir kata, semoga Allah S.W.T membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Depok, Januari 2012

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Dede Sumantri
NPM : 0706266954
Program Studi : Teknik Mesin
Departemen : Teknik Mesin
Fakultas : Teknik
Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

PENINGKATAN KINERJA MESIN *RAPID PROTOTYPING* BERBASIS *FUSED DEPOSITION MODELLING*

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 20 Januari 2012

Yang menyatakan

(Dede Sumantri)

ABSTRAK

Nama : Dede Sumantri

Program Studi : Teknik Mesin

Judul : Peningkatan Kinerja Mesin *Rapid Prototyping* Berbasis *Fused Deposition Modelling*

Mesin *rapid prototyping* berbasis *fused deposition modeling* (FDM) merupakan mesin yang berfungsi untuk membuat suatu *prototype* dengan cara memasukkan material *thermoplastic* ke dalam *heater* hingga terjadi perubahan fase dari *solid* menjadi *semisolid* dan mendepositkan material *semisolid* tersebut *layer by layer* hingga *prototype* tersebut jadi secara utuh. Penelitian ini bertujuan untuk meningkatkan kinerja mesin *rapid prototyping* berbasisi FDM yang telah dikembang oleh Departemen Teknik Mesin Universitas Indonesia dengan menggunakan metode peningkatan kinerja yang dilakukan. Metode peningkatan kinerja yang dilakukan adalah dengan menggurangi diameter *output* filamen menjadi 0.5 mm, meminimalisir defleksi yang terjadi pada produk akhir yang disebabkan adanya perbedaan temperatur yang tinggi pada setiap layer dan mengembangkan model pengisian dalam dengan menggunakan metode *slicing* yang di integrasikan dengan *CAM system.* Untuk mengurangi diameter *output* filamen dilakukan dengan mendesain *heater barrel* dengan diameter output 0.5 mm dan untuk mengurangi defleksi yang terjadi pada hasil *prototype* digunakan alas pemanas (*heatbed*)

Kata kunci: *rapid prototyping, Fused Deposition Modelling, FDM, prototype, slicing,* defleksi

ABSTRACT

Name : Dede Sumantri

Major : Mechanical Engineering

Title : *Improving Performance of Rapid Prototyping Machine Based on Fused Deposition Modelling*

Rapid prototyping machine based on fused deposition modeling (FDM) is a machine that used to create a prototype. Material thermoplastic was entered into a heater until change from solid to semisolid material and after that, semisolid material was deposited layer by layer until the prototype was finished in their entirety. This study aims to improve the performance of rapid prototyping machine based on FDM that has been developed by the Department of Mechanical Engineering University of Indonesia by using the method of performance enhancement. There are some Methods to improve performance of machine, they are reduce diameter of output filament to 0.5 mm, minimize deflection which occurs in the final product caused high temperature difference on each layer and developed filling models using the slicing algoritm that integrated with CAM systems. To reduce the diameter of output filament by designing the heater barrel with a diameter output 0.5 mm and to reduce the deflection that occurs on the prototype used heatbed.

Keywords : rapid prototyping, Fused Deposition Modelling, FDM, prototype, slicing, deflection

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR ISTILAH

LCD **:** Liquid Crystal Display
FDM : Fused Deposition Modelling
USART **:** Universal Serial Asynchronous Receiver-Transmitter
CAD **:** Computer Aided Design
STL **:** Stereolithography
ADC : Analog Digital Converter
MCU : Microcontroller Unit
MC **:** Mikrokontroler
MCS : Machine Coordinate System
CMM : Coordinate Machine Measurement
PC **:** Personal Computer

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Dewasa ini dunia industri terus mengalami perkembangan, terutama industri yang bergerak di bidang manufaktur. Dalam industri manufaktur desain suatu produk menjadi bagian yang sangat penting mengingat begitu ketatnya persaing dan cepatnya inovasi-inovasi yang dikeluarkan oleh produsen untuk mendapatkan pasar penjualan. Dalam kondisi yang demikian produsen yang dapat merespon kondisi pasar lebih cepat dan merealisasikan suatu konsep menjadi produk yang diinginkan oleh pasar akan menjadi *leader* di pasar tersebut. Hal tersebut tidak lepas dari suatu tahapan dalam mendesain suatu produk yaitu *prototyping*. Kenyataannya tidak mungkin suatu produsen berani untuk memproduksi secara massal suatu produk tanpa melihat *prototype* dari produk tersebut sehingga *prototyping* menjadi kunci utama untuk menilai apakah suatu produk desain telah memenuhi kriteria yang diinginkan dan siap untuk diproduksi secara massal. *Prototyping* akan sangat membantu menentukan proses produksi selanjutnya dan nilai investasi yang harus dikeluarkan.

Pada dasarnya *prototyping* adalah proses mereka bentuk dan tampilan suatu produk sebelum produk akhir sesuai dengan spesifikasi yang dibuat. Banyak cara dapat dilakukan untuk menghasilkan suatu *prototype* salah satunya adalah dengan mentransformasi CAD 3D tersebut ke dalam gambar kerja 2D yang berisi informasi mendetail tentang produk tersebut baik proyeksi dan isometri lalu memproduksi dengan teknik permesinan pada umumnya. Cara tersebut sangat tidak efektif karena akan membutuhkan waktu yang lama dan biaya yang besar. Terlebih bila hasilnya tidak memuaskan maka harus dibuat *prototype* baru dengan mengorbankan waktu dan biaya lebih banyak lagi.

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 menjadi alat vital dalam dunia industri. Namun untuk industri di Indonesia belum banyak digunakan dikarenakan harga mesin tersebut relatif mahal untuk industri-industri berkembang di Indonesia, maka dari itu Laboratorium Manufaktur dan Otomasi DTM FTUI mengembangkan mesin *rapid prototyping* berbasis *Fused Deposition Modelling* dengan fokus menciptakan suatu mesin *rapid prototyping* dengan harga yang jauh lebih terjangkau dan material yang lebih murah.

Gambar 1.1 Contoh produk *rapid prototyping*

Penelitian ini merupakan penelitian lanjutan untuk terus mengembangkan mesin *rapid prototyping* berbasisi *fused deposition modelling*. Penelitian ini bertujuan untuk meningkatkan kinerja dari mesin *rapid prototyping*.

1.2 Tujuan Penelitian

Secara umum 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. Penelitian ini adalah lanjutan dari peneliti sebelumnya yang telah mengembangkan mesin *rapid prototyping*. Pengembangan yang dilakukan meliputi pengembangkan perangkat lunak yang dapat mengontrol mesin *rapid prototyping* dan pengembangan perangkat keras untuk menghasilkan produk yang lebih baik.

Secara khusus penelitian ini bertujuan untuk meningkatkan kinerja mesin *rapid prototyping* untuk menghasilkan produk dengan akurasi yang lebih baik dari penelitian sebelumnya dengan menggunakan metode peningkatan kinerja yang dilakukan.

1.3 Perumusan Masalah

Beberapa masalah yang diangkat dalam penelitian ini, diantaranya adalah

1. Akurasi dari *output* filamen masih rendah
2. Proses pembentukan produk masih berbasis pada pembentukan permukaan.
3. Belum terintegrasi proses *slicing* dengan sebuah CAM sistem dan komunikasi dengan mikrokontroler.

1.4 Pembatasan Masalah

Penelitlian ini membatasi masalah pada :

1. Menggunakan mesin FDM yang telah dikembangkan sebelumnya.
2. Material yang digunakan adalah nylon.

1.5 Metodologi Penelitian

Metodologi penelitian yang diterapkan adalah sebagai berikut :

1. Melakukan studi literatur mengenai *rapid prototyping,* pemprograman dan sistem control dengan mikrokontroler.
2. Mengembangkan metode peningkatan akurasi.
3. Pengembangan perangkat lunak dan perangkat keras untuk menunjuang peningkatan akurasi.
4. Pengujian dan analisa perangkat lunak dan perangkat keras.

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. RAPID PROTOTYPING

Pada bab ini dijelaskan berbagai metode *rapid prototyping* yang digunakan dalam dunia industri, penggunaan algoritma slicing dan material thermoplastik.

BAB 3. METODE PENINGKATAN KINERJA MESIN RAPID PROTOTYPING

Pada bab ini dijelaskan beberapa analisis permasalahan yang diangkat dan metode yang digunakan untuk meningkatkan kinerja dari mesin *rapid prototyping* meliputi pengembangan *nozzle*, pengembangan *heatbed*, pengembangan perangkat lunak berbasis JAVA, dan pengembangan model *filling*.

BAB 4. ANALISIS DAN PENGUJIAN

Pada bab ini dilakukan analisis pengaruh parameter pengujian pada defleksi dan pengaplikasian hasil pengujian pada produk *rapid prototyping*.

BAB 5. KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan dari hasil penelitian lebih lanjut dengan meninjau terhadap tujuan penelitian dan saran untuk penelitian selanjutnya dari hasil dari apa yang telah dicapai pada penelitian .

BAB 2
RAPID PROTOTYPING

2.1 Klasifikasi *Rapid Prototyping*

Rapid Prototyping (RP) merupakan suatu proses pembuatan produk dengan cara solidifikasi material *layer by layer*. Prinsipnya adalah dengan melakukan penambahan *raw material* disetiap *layer* dimulai dari dari *layer* terbawah sampai dengan terakhir secara berturut-turut sampai terbentuk produk yang diinginkan.

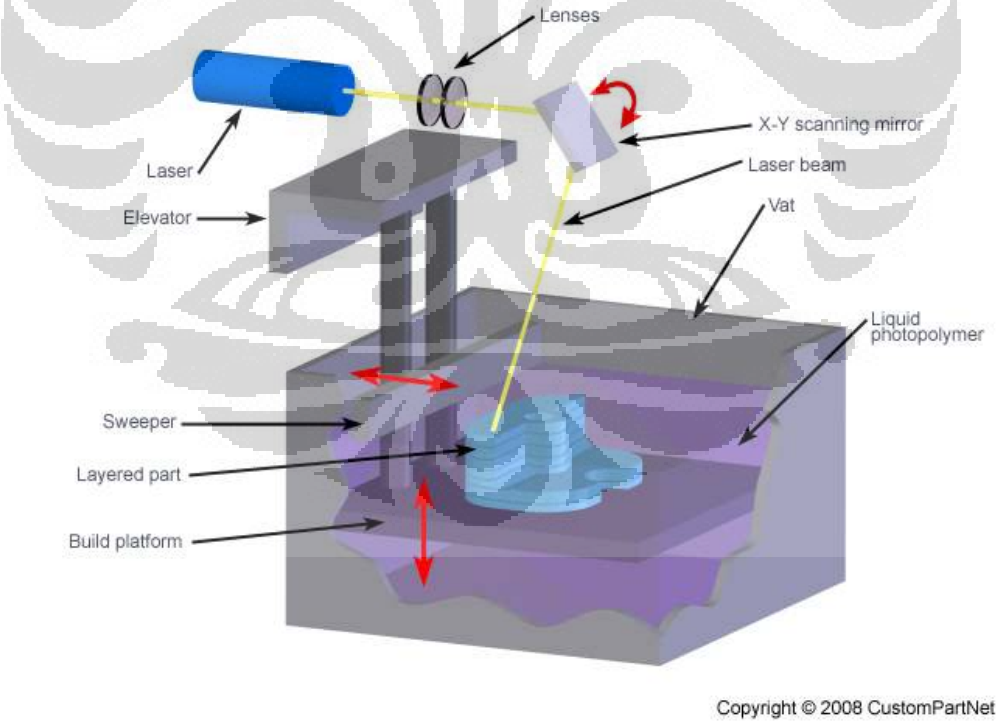
Rapid prototyping banyak digunakan di dunia industri karena memberikan keuntungan kepada industri untuk mengembangkan produknya dengan cepat. *Rapid Prototyping* mengurangi waktu pengembangan produk dengan memberikan kesempatan-kesempatan untuk koreksi terlebih dahulu terhadap produk yang dibuat (*prototype*). Dengan menganalisa *prototype*, 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)

2.1.1 Stereolithography

*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 SLA[11]

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.

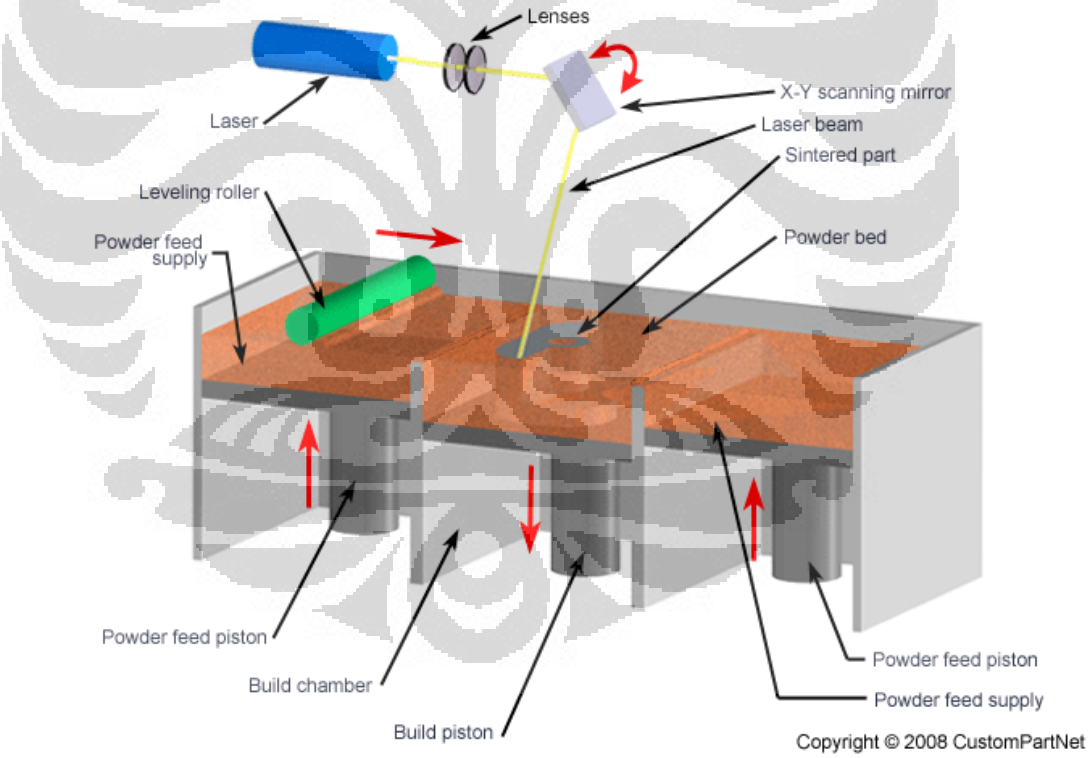
Table 1 Spesifikasi SLA[11]

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

2.1.2 Selective Laser Sintering

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) [11]

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.

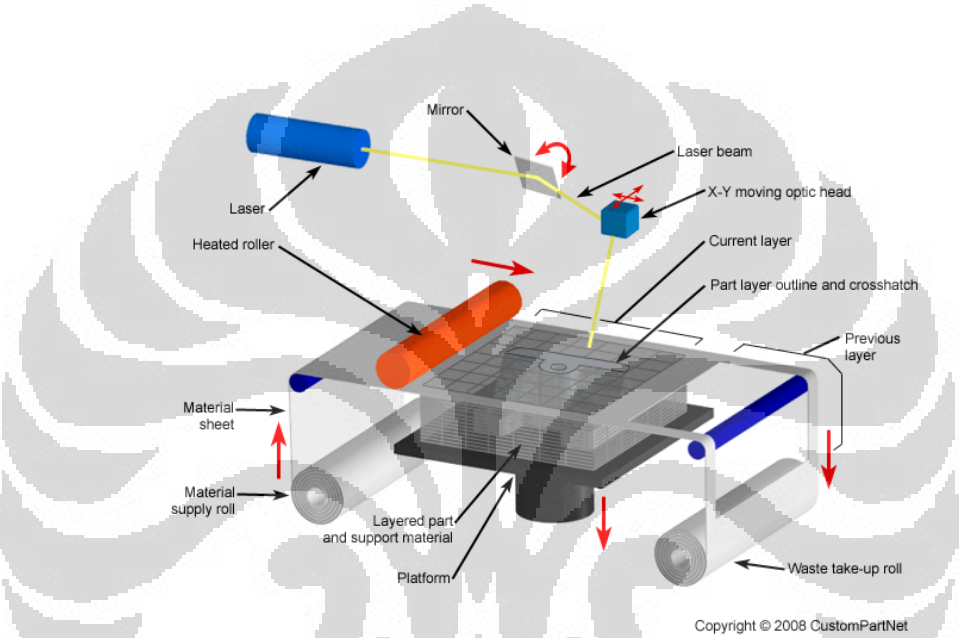
Table 2 Spesifikasi SLS[11]

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

2.1.3 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) [11]

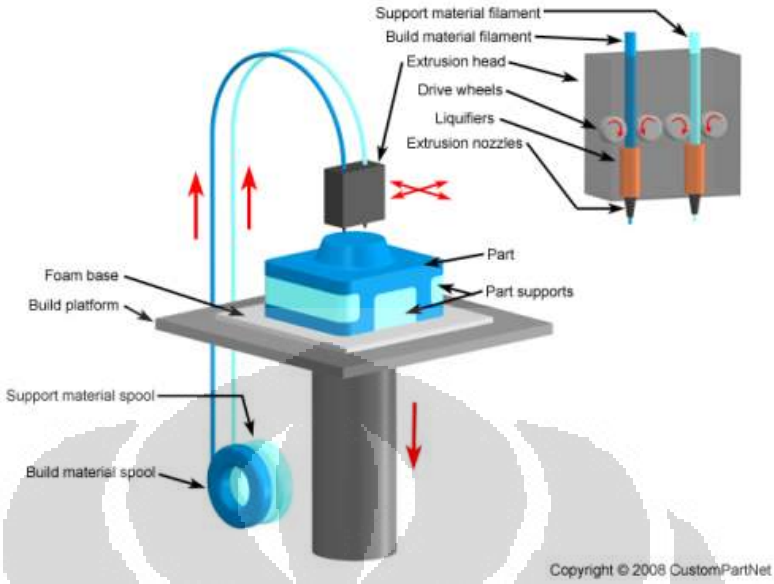
Terlihat pada gambar diatas, sinar laser ditembakkan dan dibiaskan oleh cermin khusus mengarahkan sinar laser ke lembaran material diatas *platform*. Sinar laser memotong outline pada material sesuai desain per-*layer*. Setelah terbentuk *layer*, platform bergerak turun kebawah, *waste roller* menggulung 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.

Table 3 Contoh Spesifikasi LOM[11]

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

2.1.4 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) [11]

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.

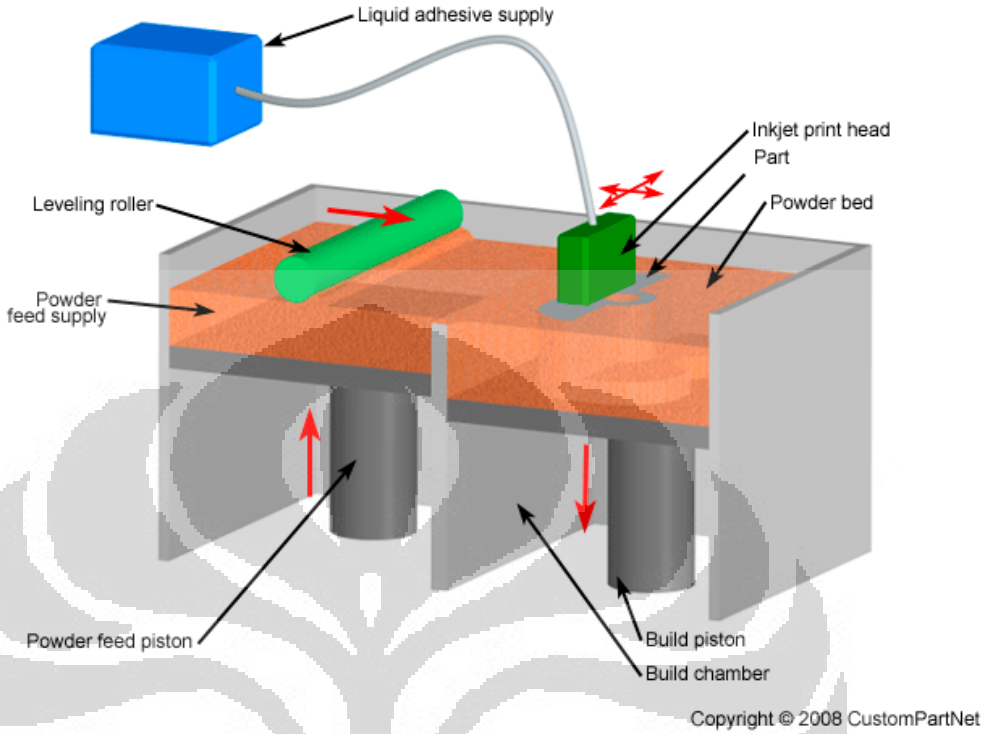
Ada beberapa variasi dari mesin FDM yang berkembang. Beberapa misalnya posisi *nozzle* yang tetap sedangkan *platform* yang bergerak mengikuti program *trajectory* kearah sumbu xyz. Beberapa pengembangan telah memungkinkan *support* material yang digunakan untuk mendukung pembentukan struktur selama 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.

Table 4 Contoh Spesifikasi FDM[11]

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

2.1.5 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) [11]

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 adhesif berikatan sehingga terbentuk sebuah *layer*. Selanjutnya *platform* bergerak ke bawah dan proses berulang sampai terbentuk *layer* terakhir. Setelah selesai, bubuk sisa yang tidak termasuk dalam struktur dapat dibuang dan dibersihkan.

Table 5 Contoh Spesifikasi 3DP[11]

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

2.2 Proses *Rapid Prototyping*

Proses *rapid prototyping* diawali dengan pemodelan CAD 3D suatu produk dengan menggunakan software CAD dengan mempertimbangkan orientasi terhadap ruang pembuatan dalam mesin *rapid prototyping.* Orientasi tersebut dilakukan untuk mempermudah proses pembuatan dan waktu yang diperlukan untuk pembuatan produk. Suatu software CAD memiliki sistem koordinat masing-masing yang biasanya dalam bentuk koordinat kartesian. Koordinat kartesian memiliki 3 sumbu aksis yaitu sumbu x, sumbu y dan sumbu z.Begitu pula dengan perangkat lunak yang dikembangkan dalam mesin *rapid prototyping* ini memiliki koordinat tersendiri. Walaupun dalam sistem koordinat yang sama yaitu koordinat kartesian namun bisa jadi ada perbedaan pendefinisian dari setiap sumbu tersebut sehingga dalam melakukan orientasi produk harus terlebih dahulu melihat kesesuaian definisi dari ketiga sumbu kartesian tersebut. Orientasi produk dalam proses *rapid prototyping* juga harus mempertimbangkan kemampuan mesin tersebut dalam membuat suatu *prototype.*

Setelah model dan orientasi selesai kemudian model dipotong dengan bidang horizontal. Banyaknya bidang potong sebanding dengan banyaknya *layer* yang dibuat. Tiap bidang horizontal akan menghasilkan bidang potong yang berisi informasi titik-titik perpotongan terhadap bidang tersebut dan kemudian dibuat jalur lintasan *nozzle* yang menghubungkan titik-titik perpotongan dalam bidang potong tersebut dan menghubungkan antara suatu titik di *layer* ke n dengan titik lain di *layer* n + 1 sampai akhirnya membentuk suatu produk sampai menjadi kondisi utuh.

2.3 Parameter *Rapid Prototyping*

Dalam *rapid prototyping* terdapat parameter-parameter yang menentukan kualitas produk. Beberapa parameter yang harus dipertimbangkan adalah *layer thickness, hatch space* dan orientasi produk. Waktu pembuatan dan keakuratan permukaan sangat dipengaruhi oleh orientasi, *layer thickness* dan parameter *hatch space*. Orientasi produk *rapid prototyping* mempengaruhi keakuratan dan waktu pembuatan. Pemilihan orientasi produk dalam kondisi yang optimal akan mempermudah proses produksi dan menghasilkan keakuratan permukaan yang tinggi. Waktu pembuatan produk sebanding dengan ketinggian z dalam arah pembuatan. Orientasi produk dengan tinggi z minimum akan menghasilkan *layer* yang sedikit dan akan mengurangi waktu pembuatan.

Selain itu, *layer thickness* juga mempengaruhi keakuratan dan waktu pembuatan. Keakuratan permukaan akan semakin baik ketika produk dibuat dengan ketebalan yang sangat kecil akan tetapi waktu pembuatan akan semakin lama dan begitupun sebaliknya. Dengan kata lain produk akan dibuat lebih cepat dengan ketebalan yang lebih besar yang menghasilkan penurunan yang akurat. Terutama pada daerah kulvatur yang tinggi. *Hatch space* merupakan jarak antara vektor parallel yang digunakan untuk proses selanjutnya yaitu pembuatan jalur *nozzle* pada setiap permukaan layer. *Hatch space* yang besar mengurangi waktu pembuatan namun apabila *hatch space* terlalu besar akan menciptakan *gap* material. Penentuan parameter *hatch space* harus mempertimbangkan ketebalan dari keluaran material dari *nozzle* yang artinya harus juga melihat diameter *nozzle*

tersebut. Penentuan parameter *hatchspace* yang tepat akan menciptakan waktu pembuatan menjadi lebih minimum dan produk menjadi lebih *solid.*

Keakuratan permukaan dapat dijelaskan sebagai deviasi geometri dari model CAD sebelumnya terhadap produk yang menyebabkan kerugian keakuratan. Kerugian kedua pada tahap proses perencanaan dimana pada bagian produk berkontur akan terbentuk efek tangga bertingkat (*stair step*) yang tampak jelas akibat *layer thickness* yang besar. Kerugian ketiga pada saat proses, terutama pada mekanisme pergerakan *nozzle.*

2.4 *Slicing* dan Pembuatan Lintasan

Proses selanjutnya setelah melakukan pemodelan CAD dengan pertimbangan orientasi adalah melakukan proses *slicing*. (Kholil, 2008) telah melakukan penelitian sebelumnya dengan membuat algoritma *slicing* yang memungkinkan program mengenal dan memilah-milah informasi untuk membuat lintasan sampai akhirnya koordinat mesin keluar dari program tersebut. Pada penelitian ini dilakukan integrasi algoritma *slicing* tersebut dengan mesin *rapid prototyping*.

Ada beberapa metode yang digunakan dalam proses pembuatan lintasan. Ada yang berbentuk pembentukkan dinding terluar, motode pengisian, dan pemisahan produk dari material pelingkup yang menentukan pola lintasan mesin *protyping.* Pembentukkan dinding terluar akan membuat proses pembuatan menjadi lebih cepat dan sedikit membutuhkan material untuk mereka bentuk luar dari produk akhir. Namun, *prototype* yang dihasilkan sangat lemah dari segi kekuatan sehingga tidak cocok untuk diberi pembebanan. Pembentukkan dengan metode pengisian akan menghasilkan produk yang sesungguhnya. Pembentukkannya ini memiliki waktu proses yang lebih lama dibandingkan dengan formasi dinding terluar. Pembentukkan ini bisa dipilih untuk menghasilkan produk yang kuat.

Pola lintasan mesin *rapid prototyping* dibuat agar dapat digerakkan secara robotik pada bidang XY untuk mesin FDM. Proses-proses ini membutuhkan strategi pembuatan lintasan (*nozzle path*) yang berbeda. Beberapa pendekatan

pembuatan proses *slicing* dan NC *path* sudah diusulkan dan diimplementasikan ke karakteristik khusus dan kebutuhan-kebutuhan berbagai proses *rapid prototyping*. Pendekatan-pendekatan proses slicing dikategorikan ke dalam empat kelompok yaitu :

1. Metode *slicing* model STL dengan ketebalan yang seragam (*uniform*)
2. Metode *slicing* model STL dengan ketebalan *layer* adaptive
3. Metode *slicing* model CAD dengan ketebalan adaptive
4. Metode *slicing* dengan perhitungan kontur yang tepat

Model *slicing* ketebalan *layer* seragam dimana semua *layer* memiliki ketebalan yang sama sedangkan metode *slicing* ketebalan *layer adaptive*, ketebalan layer bervariasi menurut kompleksitas geometri. Proses *pembuatan tool path* dapat mempengaruhi kualitas permukaan, kekuatan, kekakuan, dan waktu pembuatan produk dalam proses *rapid prototyping*. Perencanaan lintasan termasuk perencanaan lintasan bagian pengisian material pada bagian dalam *layer*. Sedangkan untuk bagian luarnya hanya digunakan pada mesin LOM karena lintasan luar dilakukan untuk memotong lembaran *raw material*.

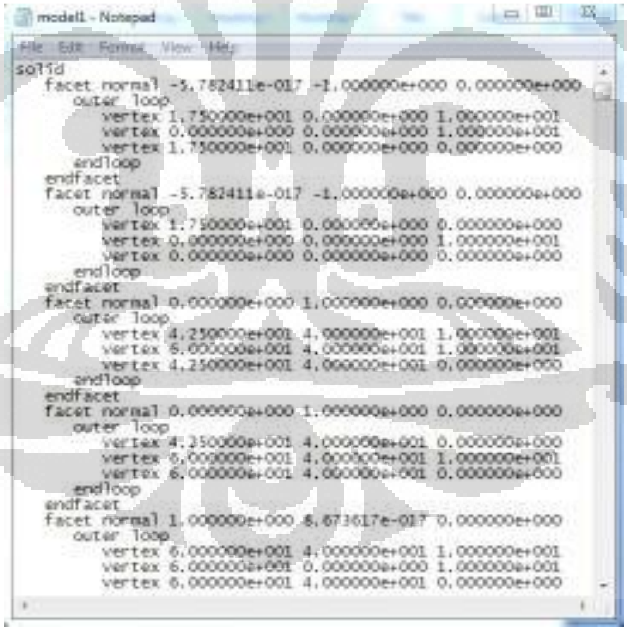
2.5 Model Facet

Sistem *rapid prototyping* menerima input sebuah objek produk 3 dimensi dalam format file STL sehingga model CAD harus diubah kedalam format STL. Perubahan format dari CAD ke STL adalah proses diskritisasi objek tersebut. Dalam format STL objek dipresentasikan dalam kumpulan segitiga yang membentuk objek tersebut secara utuh.

File STL merupakan kependekan dari *Streolithography*. File yang berekstansi STL terdiri dari 2 jenis format yaitu dalam format *binary* dan ASCII. Dalam format *binary* model *surface* yang tersusun atas segitiga-segitiga tersimpan dalam bentuk binary yang tidak dapat dibaca dengan menggunakan *text editor*. Dalam format ASCII (*American Standard for Information Interchange*) File STL dapat terbaca dengan *text editor* mudah dimengerti dan dibaca. Berikut ini adalah STL file berformat ASCII yang dibaca dalam *text editor.*

File STL menyimpan informasi objek dalam bentuk model facet 3 dimensi. Model facet adalah suatu model atau bentuk permukaan luar bidang yang tersusun

dari satu atau lebih segitiga. Segitiga tersebut disusun oleh sejumlah titik (*vertex*) yang menyusun model dihubungkan dengan garis yang menjadi sisi (*edge*) segitiga sehingga terbentuk segitiga yang saling berhubungan dan membentuk suatu permukaan bidang yang dikenal sebagai model facet. Ada dua fungsi utama dari pembentukan segitiga ini. Yang pertama adalah sebagai penghubung antar *vertex* untuk membuat sebuah permukaan dalam hal ini yang menjadi permukaan adalah bidang segitiga (*face*). Fungsi kedua adalah untuk menentukan vektor normal bidang pada wilayah tertentu. Vektor normal tersebut merupakan vektor normal segitiga yang didapat melalui proses *cross product* antara 2 vektor pembentuk segitiga. Arah dari vektor normal tersebut bergantung pada arah putaran vektor dari ketiga vertex yang digunakan. Arah vektor normal terhadap putaran vektor pembentuk segitiga dapat ditentukan mengikuti kaidah tangan kanan. Jika putaran searah dengan jarum jam (*clockwise*) maka vektor normal akan menuju bidang. Sebaliknya jika putaran berlawanan arah jarum jam (*counter clock wise*) maka vektor normal ke luar bidang.



Gambar 2.6 STL File

File yang berformat .stl mempresentasikan sebuah model facet dengan menyimpan informasi sesuai standar tertentu. Informasi yang disimpan dalam file tersebut adalah segitiga-segitiga yang memiliki beberapa *property* berupa posisi

ketiga buah *vertex* dalam bidang 3D. berikut ini adalah penjelasan mengenai file STL

1. Kata *solid* menandakan dimulainya penggambaran atau penyimpanan model facet sampai dengan ditutup dengan kata *endsolid*
2. Kata facet normal menandakan bahwa akan dibangun suatu permukaan yang berbentuk segitiga dengan nilai vektor normal berada pada kata setelah kata facet normal sampai dengan bertemu dengan kata *endfacet* yang berarti sebuah segitiga telah terbentuk
3. Kata *outerloop* menandakan dimulainya *loop* dari koordinat *vertex-vertex* yang membangun segitiga sampai dengan bertemu dengan kata *endloop*
4. Kata *vertex* merupakan *vertex* penyusun sebuah segitiga yang sebelumnya telah didefinisikan dengan *outerloop*. Informasi yang berada setelah kata *vertex* adalah posisi *vertex* pada sistem koordinat 3D

Dalam sistem *rapid prototyping* yang sedang dikembangkan informasi yang diberikan oleh fiel STL disimpan dalam dua buah vektor yaitu vektor segitiga dan vektor *vertex*. Setiap objek segitiga menyimpan informasi berupa nilai vektor normal segitiga tersebut dan indeks - indeks *vertex* penyusunnya. Dalam setiap objek *vertex* menyimpan posisi *vertex* tersebut serta normal *vertex* tersebut (jika ada). Penggunaan dua vektor yang menyimpan objek segitiga dan vertex dilakukan untuk menghindari redudansi mengingat *vertex* dapat dimiliki lebih dari satu segitiga. Dengan digunakannya dua buah vektor yang berbeda untuk menyimpan objek segitiga dan *vertex* maka beberapa segitiga bisa memiliki *vertex* yang sama.

2.6 Material Thermoplastik

Material Thermoplastik merupakan material yang sudah secara luas digunakan untuk *rapid prortotyping*. Tabel dibawah ini memeberikan informasi bebagai karakteristik polimer yang telah digunakan di dalam teknologi *rapid prototyping*

Table 6 Karakteristik polymer[12]

		Specific	Rockwell	UTS	Elongation	Modulus		Impact Notched Izod (J/cm)	Temperature (°C)		
						Tensile	Flexture		Deflection	Glass	Melting
Material	Structure	Gravity	Hardness	Mpa	of Fracture	Gpa	Gpa		at 460 kPa		
ABS	A	1.05	110 R	42	27%	2.4	2.4	2.5	68-140	107-115	
Acrylic	A	1.18	91 M	68.7	6%	3.1	3.3	0.16	60-103	100-105	130
Nylon	S	1.12	110 R	73.1	83%	2.1	2.4	1.5	85-245		250
Polycarbonate	A	1.21	120 R	69.4	96%	2.6	2.4	6.8	128-174	145-148	
Polyester											
PBT	A,S	1.42	120 R	57.1	36%	2.7	2.9	2.1	95-225		220
PET	A,S	1.32	110 R	55	130%	2.7		1.4	68-72	73-78	250
Polyethylene											
LDPE	S (-55% C)	0.92	60 R	11	190%	0.21	0.27	3.9	40-67		110
HDPE	S (-90% C)	0.96	63 R	20.3	380%	0.91	1.1	1.9	60-104		130
Polypropylene	S	0.94	96 R	36.8	120%	1.9	1.4	0.98	13-238		160
Polyvinylchloride											
Rigid	A	1.4		40	60%		3	2.75	62	75-105	200
Flexible	A			13	320%				62	75-105	

Polimer merupakan struktur *amorphous* dan terbentuk dari tiga buah monomer : *Acrylonitrile* (C3H3N), *Butadine* (C4H6) dan *Styrene* (C8H8). Kombinasi dari monomer-monomer tersebut terdiri dari formasi dua fasa co-polimer yang berbeda untuk membentuk ABS polimer. Fasa pertama adalah *hard sstyrene butadiene co polimer* dan yang kedua adalah *subbery styrenen acrylonitrile co polymer*. ABS polimer sudah digunakan di beberapa aplikasi seperti automotif, peralatan elektronik dan aplikasi lainnya.

Polimer mempunyai beberapa karakteristik yang sangat diperlukan seperti kekuatan (*strength*) yang baik dan ketahanan (*toughness*) yang relatif tinggi. Karakteristiknya dapat dimanipulasi dengan mengubah komposisi dari salah satu monomer. ABS digunakan pada teknologi *rapid ptorotyping* seperti *streolithography* (STL) , *fused Deposition Modeling* (FDM) dan *Laser sintering* (SLS).

2.6.1 Acrylic

Acrylic adalah polimer dengan struktur *amporphous* yang didapatkan dari *acrylic acid*. Material ini memeiliki transparansi yang baik yang mana dapat meneruskan 90 % cahaya. Hal ini membuat mereka menjadi kandidat utama untuk menggantikan kaca. Memiliki ketahan yang rendah terhadap goresan daripada kaca. *Acrylic* sudah tersedia dengan berbagai macam warna dan salah satunya

adalah *plexiglass*. Digunakan di automotif dan aplikasi peralatan optik. *Acrylic* digunakan untuk membuat prototype dengan menggunakan *laser sintering.*

2.6.2 Cellulose

Cellulose adalah polimer alami dengan struktur *amorphous* dan komposisi kimianya adalah C6H10O5, kayu yang mana komposisinya terdiri dari 50 % *cellulose* terdisintegrasi terlebih dahulu untuk mencapai temperatur lelehnya. Oleh karena itu, *cellulose* membutuhkan kombinasi dengan material lain agar memproduksi termoplastik dengan kekakuan panas yang diinginkan.

2.6.3 Nylon

Nylon adalah anggota dari keluarga *polyamide* (PA) yang kebanyakan memiliki struktur *crystalline*. Kebanyakan *nylon* yang digunakan adalah Nylon6 (PA6) dan, Nylon6.6 (PA6.6). nylon memiliki *wear resistance* yang baik dan kekuatannya dapat ditingkatkan dengan *reinforcing* dengan *fiberglass*. *Nylon* adalah material *rapid prototyping* utama yang digunakan pada *laser sintering* (LS) dan FDM.

2.6.4 Polycarbonate

Polycarbonate adalah polimer dengan komposisi kimia [C3H6(C6H4)2CO3]n. Mereka memiliki struktur *amorphous* dan memiliki karakteristik *creep resistance* yang baik dan ketahanan yang baik. Mereka memiliki hambatan terhadap panas yang sangat baik jika dibandingkan polimer yang lain. Mereka digunakan di aplikasi *automotive windshield* sama juga seperti produksi *housing*. Mereka digunakan untuk aplikasi *rapid prototyping* STL.

2.6.5 Termoplastic Polyester

Polyester adalah polimer yang terdiri dari struktur *semi-crystalline* dan juga terdiri dari dua tipe : *thermoplastic polyester* dan *thermosetting polyester*. Dua tipe *polyester* yang kebanyakan digunakan adalah *polybutylene terephthalate* (PBT) dan *polyethylene terephthalate* (PET). PBT digunakan di bidang manufaktur *automotive luggage rack* dan komponen *headlight*. PET digunakan dibidang pengepakan, automotif, dan industri

elektronik disebabkan sifat ketahanan dan hambatan temperatur yang tinggi. *Polyester* digunakan di teknologi *rapid prototyping* SLS.

2.6.6 Polyethylene (PE)

Polyethylene memiliki struktur *semi-crystalline* dan komposisi kimia-nya adalah [C2H4]n. *Polyethylene* memiliki ketahanan yang baik dan daya tahan terhadap reaksi kimia yang sangat baik. PE dapat diproses dengan metode produksi termoplastik apa saja, sehingga dapat menggunakan berbagai macam tipe termoplastik. mempunyai dua tipe : *low density polyethylene* (LDPE) dan *high density polyethylene* (HDPE). Perbedaan karakteristik dari kedua tipe tersebut disebabkan dari perbedaaan struktur karena massa jenis secara langsung berbanding lurus terhadap derajat kekristalan (% *crystalline*) dari struktur material. *Polyethylene* digunakan di teknologi FDM.

2.6.7 Polypropylene (PP)

Polypropylene memiliki struktur *semi-crystalline* dengan derajat kekristallan yang tinggi. PP, yang mana memiliki komposisi kimia [C3H6]n, merupakan plastik yang tersedia saat ini. PP memiliki daya tahan terhadap reaksi kimia yang baik dan karakteristik-nya sebanding dengan HDPE. PP digunakan di dalam manufaktur pembuatan plastik *hinges*. PP digunakan di teknologi *rapid prototyping* FDM.

2.6.8 Polyvinylchloride (PVC)

Polyvinylchloride memiliki struktur *amorphous* dengan komposisi kimia [C2H3C]n. Kekakuan dari PVC berbanding terbalik terhadap banyaknya *plasticiezer* yang terkandung di dalam. Ditambah lagi, PVC mengandung penstabil (stabilisator) agar supaya mengontrol ketidakstabilan-nya ketika terkena cahaya dan panas. PVC digunakan di teknologi *rapid prototyping* SLS.

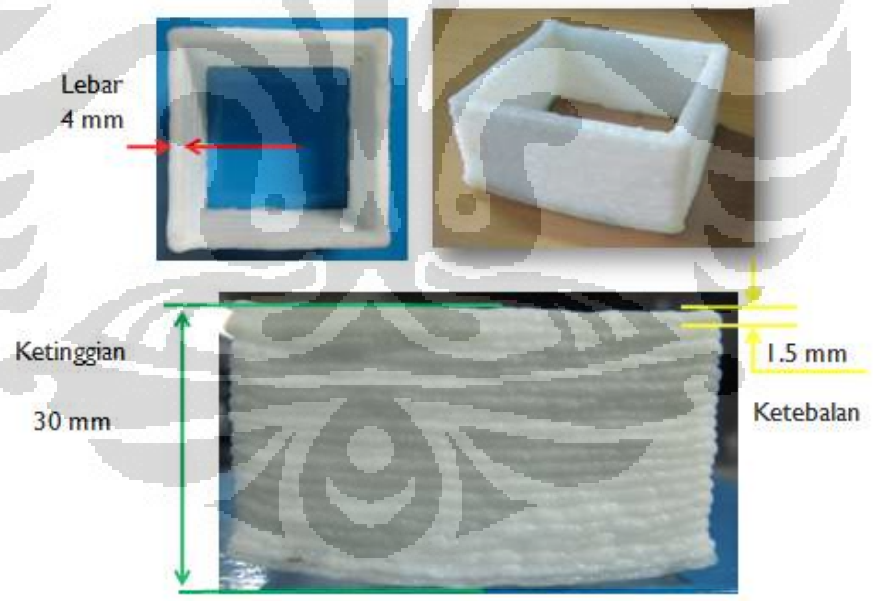
BAB 3
PENINGKATAN KINERJA MESIN RAPID PROTOTYPING

Untuk meningkatkan kinerja dari mesin *rapid prototyping* berbasis *fused deposition modelling* dilakukan analisis terhadap permasalahan yang terjadi dan menerapkan beberapa metode berdasarkan analisis yang telah dilakukan. Berikut ini adalah beberapa metode yang digunakan untuk meningkatkakn kinerja dari mesin *rapid prototyping*

3.1 Pengurangan Diameter *Output* Filamen

3.1.1 Analisis Pengurangan Diameter *Output* Filamen

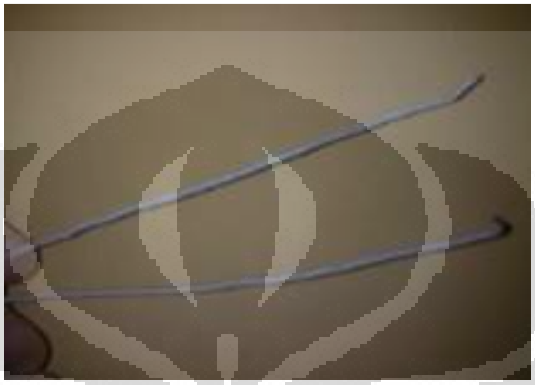
Pada penelitian sebelumnya produk akhir yang dihasilkan dari mesin *rapid prototyping* masih memiliki tingkat akurasi yang rendah, hal tersebut terlihat dari dimensi produk yang masih besar.



Gambar 3.1 Akurasi produk lama

Akurasi yang rendah dari produk akhir *rapid prototyping* disebabkan karena dimensi dari *output filament* yang dihasilkan oleh *nozzle* masih terlalu besar. Berdasarkan studi literatur yang dilakukan Tian Ming Wan [8]

menggunakan mesin *rapid prototyping* berbasis *fused deposition modeling* dengan menggunakan diameter *nozzle* 0.25 mm dan menghasilkan produk dengan akurasi yang lebih baik. Hal tersebut membuktikan bahwa dengan pengurangan diameter *nozzle* akan mengurangi diameter *output filament* yang akan meningkatkan akurasi dari produk akhir *rapid protortping* berbasis FDM.



Gambar 3.2 output filament produk lama

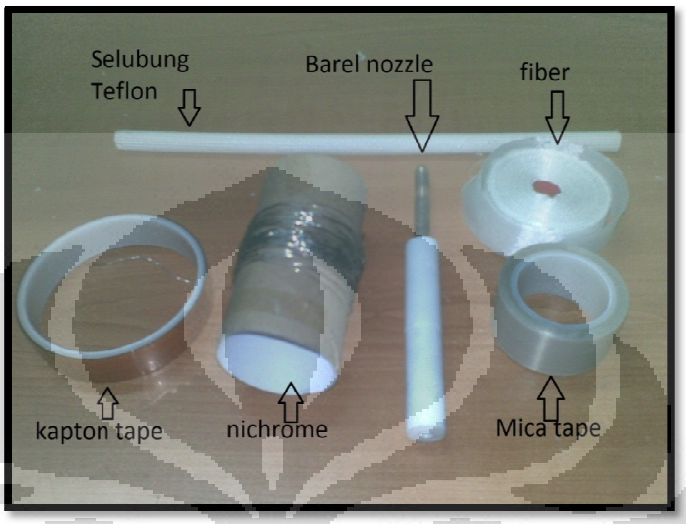
Pada gambar 3.2 diatas terlihat bahwa diameter *output filament* yang dihasilkan oleh mesin *rapid prototyping* masih rendah. Besarnya diameter *ouput filament* tersebut adalah 1 mm, *output filament* tersebut dihasilkan oleh diameter berukuran 1 mm. Berdasarkan pada penelitian sebelumnya, untuk meningkatkan kinerja dari mesin *rapid prototyping* dengan mengurangi dimensi *output filament* dilakukan pengembangan dengan cara mendisain *nozzle* dengan output 0.5 mm dengan target *output filament* yang dihasilkan adalah 0.5 mm.

3.1.2 Pengembangan Nozzle

Untuk dapat meningkatkan kinerja dari mesin *rapid prototyping* yaitu dengan meningkatkan akurasi dari produk akhir RP. Perlu adanya pengurangan dimensi output filament yang sangat dipengaruhi oleh diameter *nozzle* yang digunakann. Untuk itu, dikembangkan desain *nozzle* yang memiliki diameter *nozzle* 0.5 mm.

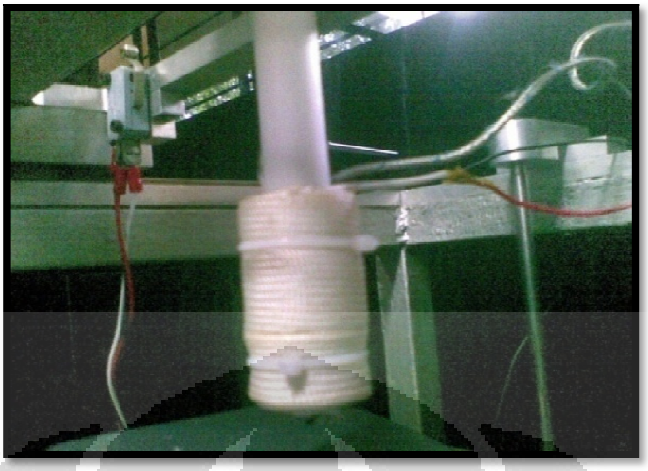
Material yang digunakan untuk pengembangan *nozzle i*ni masih menggunakan kuningan (*brass*) yang konduktifitas thermalnya tidak terlalu tinggi untuk menjaga agar laju pelepasan kalor yang terjadi tidak terlalu besar. Untuk

menjaga agar suhu *heater* tetap stabil maka diperlukan isolator untuk mengisolasi panas agar tidak keluar dari sistem. Berikut ini adalah beberapa isolator yang digunakan untuk mengurung panas pada komponen *heater.*

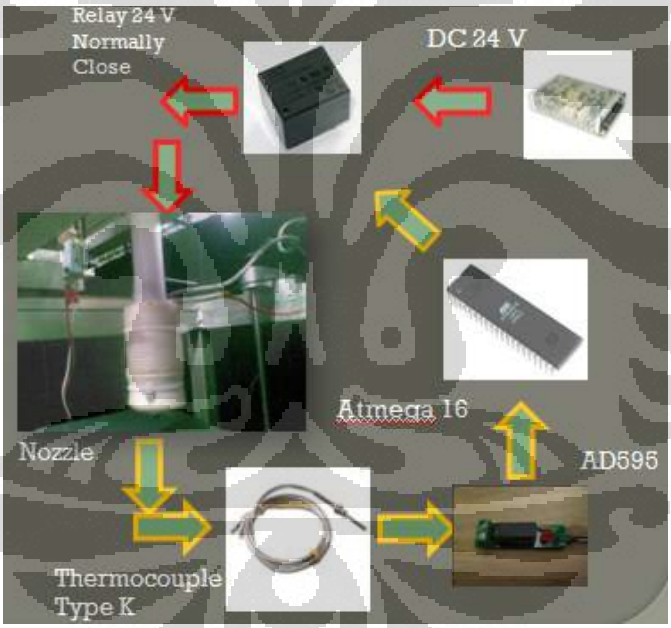


Gambar 3.3 Komponen *nozzle*

Setelah itu dilakukan proses *assembly* setiap komponen yang diatas, kawat *nichrome* masih digunakan pada penelitian ini sebagai elemen pemanas karena masih efektif untuk meningkatkan temperatur pada *heater*. Penggunaan selubung teflon pada isolator menurunkan temperatur kerja secara signifikan, pada penelitian sebelumnya temperatur kerja yang harus dicapai untuk merubah fase material berkisar antara 330-340 $°C$. Pada panggunaan *heater* yang baru temperatur kerja menurun pada *range* 250-260 $°C$. Hal tersebut membuktikan bahwa isolator yang digunakan lebih baik dari yang sebelumnya.



Gambar 3.4 ***Assembly nozzle***



Gambar 3.5 Skema kontrol suhu ***nozzle***

Tidak ada perbedaan antara sistem kontrol suhu *nozzle* yang lama dengan yang baru. Namun, untuk penelitian ini mikrokontroler atmega 16 mengontrol 2 suhu sekaligus yaitu temperatur *heatbed* dan temperatur *nozzle* dan menampilkannya dalam satu LCD yang sama.

3.2 Pengurangan Defleksi

3.2.1 Analisis Pengurangan Defleksi

Defeksi terjadi pada produk akhir dari *rapid prototyping* yang mneyebabkan produk menjadi tidak sesuai dengan yang diharapkan. Defleksi yang terjadi dapat mengganggu proses pembuatan *prototype.*

Gambar 3.6 Defleksi pada produk *rapid prototyping*

Defleksi terjadi pada beberapa layer terbawah dan tren nya menurun (defleksi menurun) seiring dengan peningkatan jumlah *layer*. Defleksi ini mungkin terjadi diakibatkan karena adanya perbedaan temperatur pada setiap *layer*, ketika satu *layer* pertama selesai didepositkan maka secara cepat temperatur material akan menuju temperatur ruang, kemudian *layer* kedua menimpa, dan terjadi perbedaan temperatur antara *layer* n dan *layer* n+1, sampai dengan beberapa *layer* ke atas terjadi penumpukan kalor yang menyebabkan semakin adanya perbedaan temperatur antara *layer* terbawah dengan layer atasnya.

Untuk menguatkan dugaan, kemudian dilakukan studi literatur dan ditemukan bahwa ada permasalahan yang sama pada kasus *injection molding*, *rapid prototyping* berbasis FDM dan *injection molding* memiliki beberapa persamaan dalam mekanisme pengekstrusian material ke dalam suatu *heater,* di dalam *injection molding* salah satu penyebab cacat produk adalah karena adanya perbedaan temperatur pada proses pendinginan, plastik yang mendingin lebih cepat pada bagian terluar *mold* tertarik kearah plastik yang temperaturnya lebih tinggi yang berada di dalam *mold.*

Berdasarkan hipotesa tersebut, perlu adanya perlakuan khusus untuk mengatasi defleksi yang terjadi pada beberapa *layer* terbawah tesebut. Untuk itu,

perlu adanya sesuatu alat yang berfungsi untuk mempertahankan kondisi *layer* terbawah agar tetap terjaga pada temperatur tertentu dan menahan penarikan material yang terjadi akibat perbedaan temperatur.

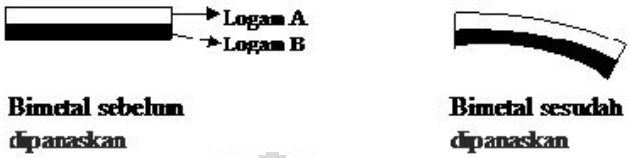
3.2.2 Pengembangan Heatbed

Pada Penelitian sebelumnya belum digunakan alas pemanas pada system. *heatbed* berfungsi untuk menjaga temperatur material yang keluar dari *heater* tetap berada diatas *glass point*. Material yang suhunya telah berada dibawah suhu glass pointnya akan menjadi licin dan sulit untuk mempertahankan posisinya pada saat proses pembuatan *prototype layer by layer*. Untuk itu, perlu adanya alat yang dapat mempertahankan fase *semisolid* dari material tersebut. Material *semisolid* tersebut akan memiliki daya *adhesive* dengan alas pemanas. Alas pemanas yang digunakan adalah setrika yang telah dimodifikasi sedemikian rupa agar suhu nya dapat dikontrol dengan mikrokontroler.

Gambar 3.7 *Heatbed*

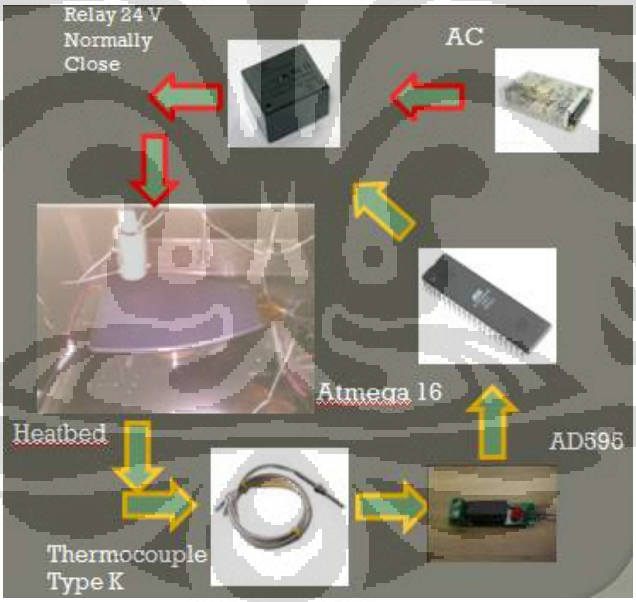
Setrika pada umumnya memiliki sensor suhu mekanik yaitu bimetal. Bimetal adalah sensor suhu yang terbuat dari dua lempengan logam yang berbeda koefisien muainya. Bila suatu logam dipanaskan maka akan terjadi pemuaian, besarnya pemuainya tergantung dari jenis logam dan temperatur kerja dari logam tersebut. Bila dua lempeng logam tersebut saling direkatkan dan dipanaskan maka logam yang memiliki koefisien muai lebih tinggi akan memuai lebih panjang sedangkan yang memiliki koefisien muai lebih rendah akan memuai lebih pendek

karena perbedaan muai tersebut maka bimetal akan melengkung kearah logam yang memiliki koefisien muai lebih rendah. Bimetal adalah saklar alami yang bersifat *normally close* (NC).



Gambar 3.8 bimetal

Untuk keperluan penelitian ini penulis membuang sensor bimetal tersebut dan menggantinya dengan relay 24 v yang telah dibahas diatas. Dengan mengganti sensor dari bimetal menjadi *relay* akan lebih keleluasan dan mudah untuk mengontrol suhu yang diinginkan.



Gambar 3.9 Skema kontrol suhu *heatbed*

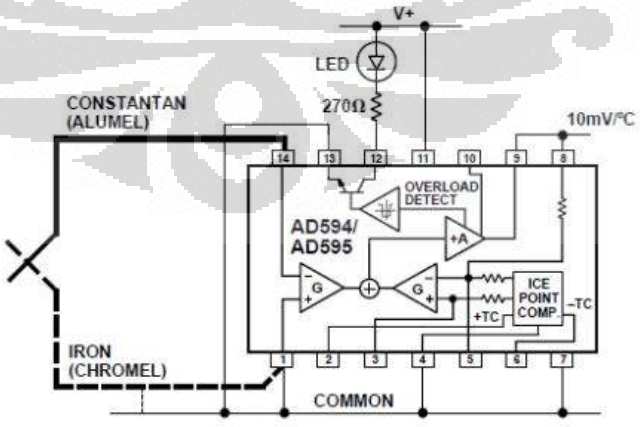
3.2.3 Thermocouple Amplifier

Temperatur kontrol berfungsi untuk mengatur temperatur pencairan material agar tetap konstan. Metode pengontrolan yang digunakan adalah dengan menggunakan mikrokontroller ATmega 16 melalui fitur ADC 10 bit yang sudah

terintegrasi didalam *chip*. Sensor yang digunakan untuk mengukur temperatur tersebut adalah *thermcouple* tipe-K, dan *thermocouple* ini sudah cukup linier antara temperatur dengan tegangan, akan tetapi tegangan yang dihasilkan dalam satuan mV sehingga dibutuhkan *thermocouple amplifier* (penguat tegangan). Setelah melalui *thermocouple amplifier* ini, barulah sinyal masuk ke ADC (*Analog Digital Converter*) pada mikrokontroller. *Thermokopel amplifier* yang digunakan adalah AD595 yang mampu menguatkan tegangan setiap kenaikan 10mV per 1 derajat Celcius. Jangkauan temperatur yang akan diukur adalah dari 0 sampai 500 derajat Celcius. Pada mikrokontroler fitur ADC memiliki 10 bit setiap *channel*-nya, sehingga dibutuhkan konversi hitungan sinyal yang masuk ke ADC, karena dengan 10 bit berarti angka karakter mampu mencapai 10 bit = 210 = 1024. Pada ADC menggunakan tegangan referensi AREF 5V, sehingga :

$$\frac{1024 bit}{5V} \times 10\, {}^{mV}\!/_{{}^{\circ}C} = 2{,}048\, {}^{bit}\!/_{{}^{\circ}C}$$

Pada perhitungan di atas berarti setiap kenaikkan angka 2,048 bit, berarti kenaikkan setiap 1 0C, oleh karena itu angka karakter yang didapat melalui AD dibagi dengan angka 2,048 sehingga jangkauan temperatur yang terukur dari 0 mencapai 500 derajat celcius. Desain PCB yang dikembangkan menggunakan alarm, berupa lampu LED sebagai tanda jika *thermocouple* tidak terpasang dengan baik maka LED akan menyala. Skematik yang dikembangkan mengikuti skematik berdasarkan datasheet AD595.



Gambar 3.10 Skematik AD595[14]

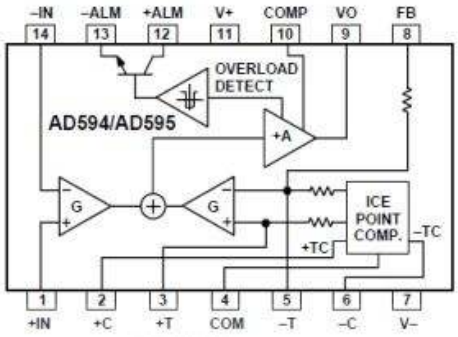
Table 7 Temperatur dan tegangan terukur AD595[14]

Thermocouple Temperature °C	Type J Voltage mV	AD594 Output mV	Type K Voltage mV	AD595 Output mV
-200	-7.890	-1523	-5.891	-1454
-180	-7.402	-1428	-5.550	-1370
-160	-6.821	-1316	-5.141	-1269
-140	-6.159	-1188	-4.669	-1152
-120	-5.426	-1046	-4.138	-1021
-100	-4.632	-893	-3.553	-876
-80	-3.785	-729	-2.920	-719
-60	-2.892	-556	-2.243	-552
-40	-1.960	-376	-1.527	-375
-20	-.995	-189	-.777	-189
-10	-.501	-94	-.392	-94
0	0	3.1	0	2.7
10	.507	101	.397	101
20	1.019	200	.798	200
25	1.277	250	1.000	250
30	1.536	300	1.203	300
40	2.058	401	1.611	401
50	2.585	503	2.022	503
60	3.115	606	2.436	605
80	4.186	813	3.266	810
100	5.268	1022	4.095	1015
120	6.359	1233	4.919	1219
140	7.457	1445	5.733	1420
160	8.560	1659	6.539	1620
180	9.667	1873	7.338	1817
200	10.777	2087	8.137	2015
220	11.887	2302	8.938	2213
240	12.998	2517	9.745	2413
260	14.108	2732	10.560	2614
280	15.217	2946	11.381	2817
300	16.325	3160	12.207	3022
320	17.432	3374	13.039	3227
340	18.537	3588	13.874	3434
360	19.640	3801	14.712	3641
380	20.743	4015	15.552	3849
400	21.846	4228	16.395	4057
420	22.949	4441	17.241	4266
440	24.054	4655	18.088	4476
460	25.161	4869	18.938	4686
480	26.272	5084	19.788	4896
500	27.388	5300	20.640	5107
520	28.511	5517	21.493	5318
540	29.642	5736	22.346	5529
560	30.782	5956	23.198	5740
580	31.933	6179	24.050	5950
600	33.096	6404	24.902	6161
620	34.273	6632	25.751	6371
640	35.464	6862	26.599	6581
660	36.671	7095	27.445	6790
680	37.893	7332	28.288	6998
700	39.130	7571	29.128	7206
720	40.382	7813	29.965	7413
740	41.647	8058	30.799	7619
750	42.283	8181	31.214	7722
760	–	–	31.629	7825
780	–	–	32.455	8029
800	–	–	33.277	8232
820	–	–	34.095	8434
840	–	–	34.909	8636
860	–	–	35.718	8836
880	–	–	36.524	9035
900	–	–	37.325	9233
920	–	–	38.122	9430
940	–	–	38.915	9626
960	–	–	39.703	9821
980	–	–	40.488	10015
1000	–	–	41.269	10209
1020	–	–	42.045	10400
1040	–	–	42.817	10591
1060	–	–	43.585	10781
1080	–	–	44.439	10970
1100	–	–	45.108	11158
1120	–	–	45.863	11345
1140	–	–	46.612	11530
1160	–	–	47.356	11714
1180	–	–	48.095	11897
1200	–	–	48.828	12078
1220	–	–	49.555	12258
1240	–	–	50.276	12436
1250	–	–	50.633	12524

Hubungan antara temperatur dengan tegangan pada tabel 7 sudah mencapai *linier*. AD595 merupakan *thermocouple amplifier* dengan *cold junction compensator* yang sudah terintegrasi di dalam IC. Compensator tersebut sudah digabungkan dengan referensi titik beku es dengan amplifier yang sudah terkalibrasi sehingga menghasilkan sinyal keluaran 10mV/0 C secara langsung dari sinyal masukan dari *thermocouple*. Tegangan keluaran dari *thermocouple* tidak linier terhadap temperatur, oleh karena itu pada AD595 hubungan tersebut dibuatlah *transfer function* agar keluaran tegangan keluaran aktual pada AD595 linier terhadap temperatur.

$$AD595output = (TypeK\_Voltage + 11\mu V) \times 247.3$$

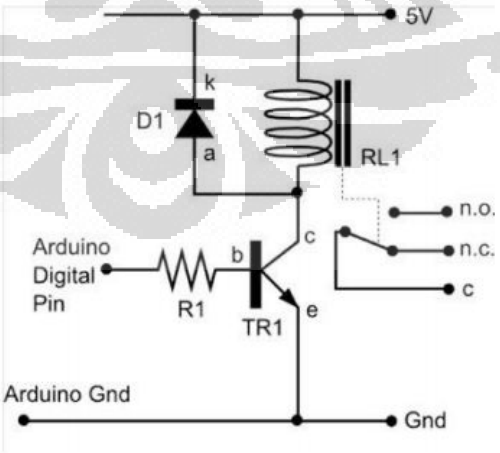
Persamaan di atas adalah hubungan tegangan keluaran thermocouple tipe-K dengan tegangan keluaran aktual pada AD595.



Gambar 3.11 IC AD595[14]

3.2.4 Relay

Relay digunakan pada penelitian ini berfungsi sebagai saklar elektromagnetis agar arus yang mengalir ke kawat *nichrome* dapat diputus jika temperatur yang diharapkan dicapai, sehingga diharapkan temperatur tetap konstan. Tipe relay yang digunakan adalah *solid state relay*, bekerja pada jangkauan 24VDC,10A. Relay dalam kondisi *normally closed*, sehingga setelah mendapat sinyal *trigger* dari mikrokontroller ATmega 16, kondisi relay menjadi *normally open* dan arus berhenti mengalir atau terputus. Oleh karena itu, temperatur yang dicapai tidak pernah konstan pada temperatur tertentu, akan tetapi temperatur akan naik dan turun tidak jauh dari jangkauan temperatur yang dibuat konstan, misalnya temperatur konstan yang diinginkan 250 C, maka temperatur yang dicapai bisa pada jangkauan ±100 C.



Gambar 3.12 Skematik relay[15]

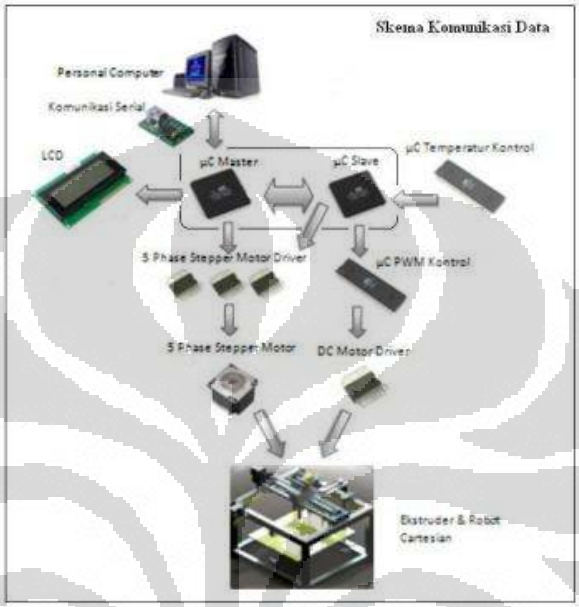
Illustrasi pada gambar 3.12 adalah prinsip kerja dari relay secara umum. Pada koil atau lilitan dipasangkan diode secara parallel dengan koil, bertujuan agar koil terproteksi terhadap arus balik ketika melakukan proses *switching*. Pada gambar 3.12, kondisi saklar dalam keadaan *normally closed.* Ketika pin yang terhubung ke *controller* (MCU) memberikan sinyal logika terhadap pin basis pada transistor TR1, maka arus dari pin *collector* akan mengalir ke pin emitter yang telah terhubung dengan ground (GND). Maka dengan mengalirnya arus tersebut maka koil RL1 akan teraktifasi dan merubah keadaan saklar dari *normally open* ke *normally closed.* Jangkauan tegangan pada koil relay pada umumnya antara 5 sampai 10 Volt, sedangkan untuk jangkauan tegangan saklar bisa mencapai 24V, jangkauan tegangan tersebut tergantung dari produsen yang membuat relay tersebut.

Gambar 3.13 Relay 24 V[16]

3.3 Pengembangan Komunikasi Mikrokontroler

Sebuah mikrokontroler dalam suatu sistem dapat berkomunikasi dengan mikrokontroler pada sistem lainnya dan saling mengirimkan nilai dimana setiap nilai dikonversi menjadi sebuah tugas yang harus diselesaikan untuk kemudian kembali memberikan *feedback* kepada pemberi value yang menandakan bahwa suatu tugas telah dijalankan dan menunggu tugas berikutnya diberikan. Dalam berkomunikasi baik antara mikrokontorler dengan mikrokontroler dan mikrokontroler dengan PC dilakukan dengan menggunakan komunikasi serial. Khusus komunikasi antara PC dengan mikrokontroler diperlukan suatu divais interface yang sebagai penghubung diantara keduanya secara serial. Pada penelitian ini digunakan divais serial yaitu USB to serial. 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*.



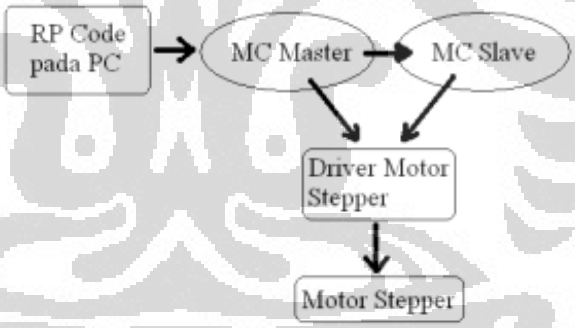
Gambar 3.14 Skema Komunikasi Data[6]

Pada penelitian sebelumnya Sulaiman [6] telah melakukan pengembangan perangkat lunak mesin *rapid prototyping* dan menciptakan suatu protokol khusus yang memungkin adanya komunikasi yang baik antara mikrokontroler dengan mikrokontroler dan mikrokontroler dengan PC. Keseluruhan protokol tersebut dijalankan dengan C *Compiler*. Keberlanjutan dari pengembangan perangkat lunak mesin *rapid prototyping* ini adalah bagaimana menjalankan protokol yang sudah dirancang tersebut dalam suatu *platform* baru yaitu dalam *platform* JAVA. Selain dari perubahan *platform* tersebut juga dilakukan pengembangan untuk menyesuaikan antara input yang diberikan dengan protokol yang ada. Input pada penelitian ini memiliki karakteristik yang berbeda dengan penelitian sebelumnya. Berikut adalah gambaran umum algoritma yang digunakan dan masih diterapkan pada penelitian ini.

```
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*, PC hanya melakukan hubungan komunikasi dengan mikrokontroler *master* setelah itu mikrokontroler master melakukan komunikasi dengan mikrokntroler *slave* untuk melakukan pembagian tugas menjalankan motor *stepper* sesuai dengan *protocol* yang berlaku.



Gambar 3.15 Bagan Pengiriman Data[6]

Pengiriman 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.

3.3.1 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 tidak akan berjalan dengan baik. Pada dasarnya aliran data dari PC ke mikrokontroler *master* akan langsung diteruskan oleh mikrokontroler master kepada mikrokontroler *slave* apapun nilai yang dikeluarkan oleh mikrokontroler *master* selama memenuhi *protocol* yang berlaku. Hal tersebut yang mnejadikan apabila terjadi kesalahan pembacaan atau penerimaan data dari PC akan menyebabkan seluruh sistem menjadi tidak berjalan sebagaimana mestinya.

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 menunggu perintah user.
         1. Case 1: posisi awal.
         2. Case 2: Start point
         3. Case 3: 3D Print
         4. Case 4: disconnected connection
         5. Case 5: exit program.
      4) Mikrokontroler master menunggu PC mengirim karakter
      5) If case 1 yang di pilih, PC looping mengirim karakter 'A' ke
         mikrokontroler master hingga mikrokontoler master mengirim
         kembali karakter 'A'.
      6) Mikrokontroler master menjalankan perintah posisi awal.
      7) If case 2 yang dipilih, PC akan mengirim karakter 'S' hingga
         mikrokontroler master mengirim kembali karakter 'S'.
      8) Mikrokontroler menjalankan perintah start point
      9) PC mengirim satu karakter dari satu string data diakhir oleh
         karakter 'x'
     10) PC menunggu karakter 'k'
     11) Setelah data di terima dan disimpan, mikrokontoler master
         mengirim karakter 'k' ke PC
     12) Setelah karaker 'k' diterima, PC melanjutkan mengirim karakter
         selanjutnya dari satu string data dan diakhiri oleh karakter
         'x'
     13) Setelah PC mengirim 3 kali karakter 'x', PC menunggu karakter
         'y'
     14) Setelah mikrokontroler master mengirim 3 kali karakter 'k',
         mikrokontroler master mengirim karakter 'y'
     15) Mikrokontroler master mengeksekusi koordinat
     16) Jika posisi koordinat belum seperti yang diinginkan maka
         mengulang langkah  7 – 18
     17) If case 3 yang di pilih, PC mengirimkan karakter 'B'
```

```
    18) Mikrokontroler master menjalankan perintah jalan program.
    19) Mikrokontroler master menunggu PC mengirim satu karakter dari
        satu string data dan diakhiri oleh karakter 'x'
    20) Setelah data diterima dan di simpan, mikrokontroler mengirim
        karakter 'k' ke PC
    21) PC menunggu hingga karakter 'k' dikirim.
    22) Setelah karakter 'k' diterima, PC melanjutkan mengirim
        karakter selanjutnya dari satu string data dimana setiap
        string data diakhiri oleh karakter 'x'
    23) Setelah PC mengirim 3 kali karakter 'x', PC menunggu karakter
        'y'
    24) Setelah mikrokontroller master mengirim 3 kali karakter 'k',
        mikrokontroler master mengirim karakter 'y'
    25) Proses berulang hinga seluruh data selesai di kirim.
    26) If case 4 dipilih maka PC menutup serial komunikasi
    27) If case 5 dipilih PC menutup program rapid prototyping
END
```

3.3.2 Protokol Pengiriman Data Antar Dua Mikrokontroler

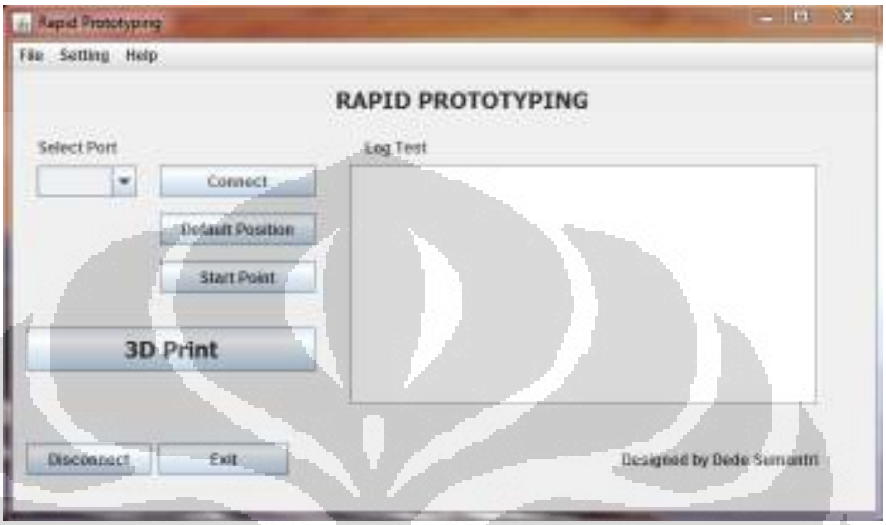
Seperti halnya pada pengiriman data antara mikrokontroler dan PC, pengiriman data antar dua mikrokontroler juga merupakan hal yang penting karena jika terjadi kesalahan maka sebagian dari sistem akan kacau, untuk itu dirancang protokol dengan sedemikian rupa sehingga kemungkinan *error* yang terjadi 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.
      11) If user memilih perintah start point, mikrokontroler master
          akan mngirim karakter 'S' kepada mikrokontroler slave
      12) Mikrokontroler slave dan mikrokontroler master menjalankan
          program start point
      13) Mikrokontroler slave menunggu mikrokontroler master mengirim
          satu string data
      14) Setelah mikrokontorler master mengirim satu string data,
          karakter 'x' dikirim untuk menandakan akhir dari string data
      15) Mikrokontroler slave menyimpan data
      16) Setelah 3 string terkumpul, motor dijalankan
      17) Proses berulang hingga seluruh data selesai di kirim.
```

```
END
```

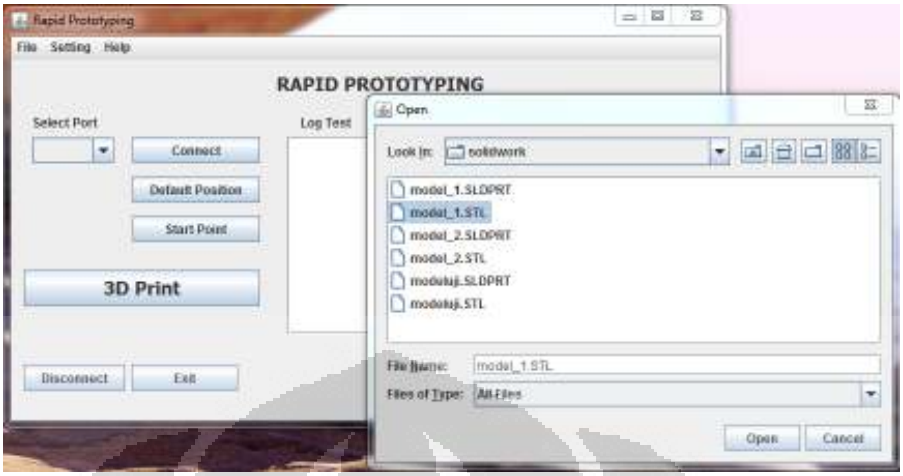
3.3.3 Tampilan User Interface



Gambar 3.16 Tampilan awal

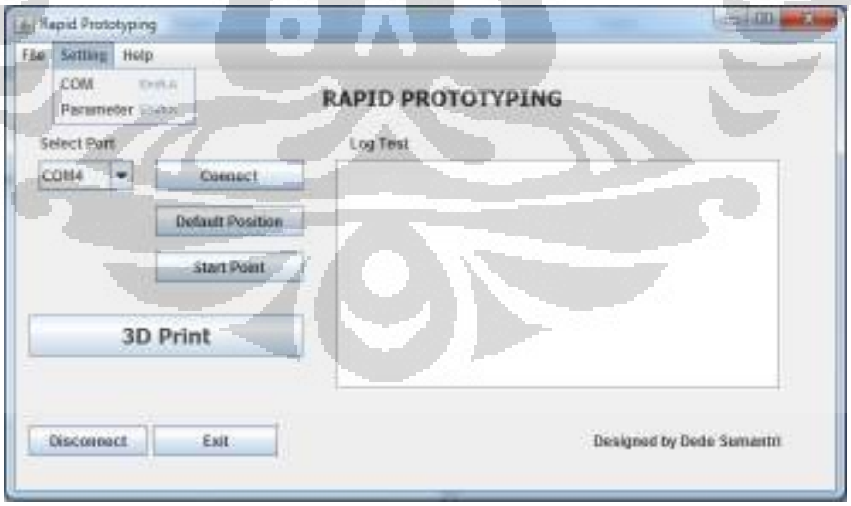
Pada *user interface*, pengguna dapat memilih perintah apa yang ingin diberikan kepada mikrokontroler, *user interface* ini juga berguna sebagai alat agar pengguna mengetahui sudah sampai dimana program dijalankan, dapat juga sebagai media untuk mengetahui dimana kesalahan terjadi. Program ini dijalankan pada JAVA *compiler* yaitu Netbeans 6.9.

Pada saat program dijalankan, program akan melakukan *scanning port* yang sedang digunakan di PC dan memasukkannya ke daftar *port*. Pada dasarnya ketika suatu divais di hubungkan dengan PC, *port* untuk divais tersebut tidak selalu sama antara computer yang satu dengan yang lain sehingga perlu adanya penyesuaian. *User* dapat memilih port mana yang akan digunakan pada program ini.



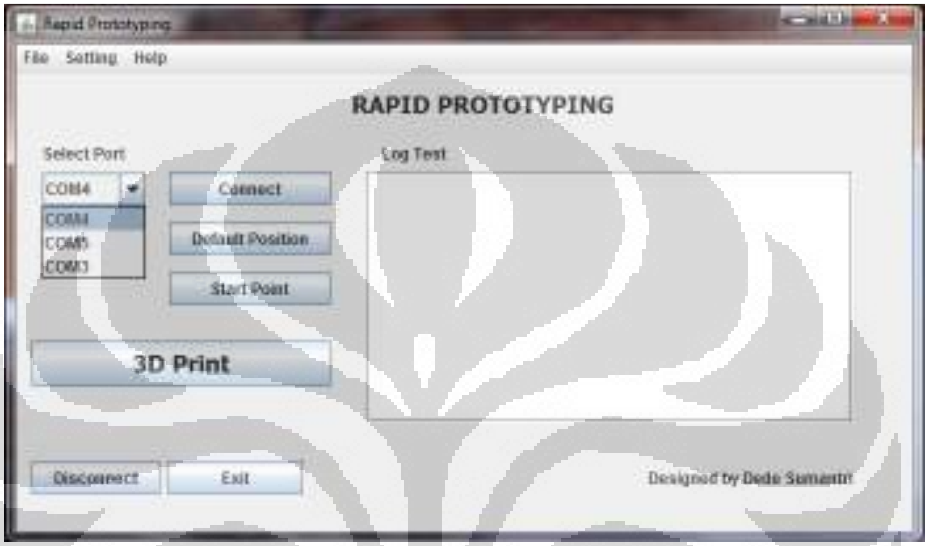
Gambar 3.17 Open file STL

Pada software ini telah *included* algoritma untuk meng-*slicing* STL file sampai akhirnya didapat RP *code* yang akan dikirim ke mesin *rapid prototyping.* Sebelum melakukan koneksi dengan mesin, *user* melakukan *slicing* produk dengan cara meng input file STL yang akan di buat *prototype* nya. Setelah meng input file STL *user* melakukan konfigurasi parameter yang dibutuhkan dalam proses *slicing* yaitu *layerthickness* dan *hatch space*. *Setting* parameter dapat dilakukan di *menu item* parameter pada *menubar setting*.



Gambar 3.18 Konfigurasi parameter *slicing*

Setelah proses *slicing* selesai dan RP *code* di dapat proses selanjutnya adalah melakukan komunikasi dengan mikrokontroler. Program telah mendapatkan *port – port* yang aktif pada PC, *user* akan memilih satu *port* yang terhubung dengan serial divais yaitu polulu serial to USB. Untuk mengetahui *port* mana terhubung dengan polulu serial to USB dapat dilihat pada *device manager*.



Gambar 3.19 Pemilihan *Com Port*

Setelah melakukan pemilihan port serial langkah selanjutnya adalah *open port serial* dengan mengklik tombol *connect.* Jika koneksi berhasil maka sampai tahap ini program sudah dapat berkomunikasi dengan mikrokontroler dan program yang dijalankan adalah menunggu mikrokontroler *master* mengirim karakter 'z' yang menandakan heater sudah siap. Ketika *heater* telah siap *user* dapat memilih program selanjutnya yang akan dijalankan seperti *default position* dan *start point* sebelum akhirnya melakukan pengiriman data dengan menekan tombol 3D Print.

3.3.4 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. *Connect* : mencari dan membuka port serial
2. *Default Position* :menggerakan *nozzle* ke posisi awal
3. *Start Point* :menggerakan *nozzle* ke posisi yang diinginkan untuk memulai membuat produk
4. *3D Print* :mengirimkan RP *code*
5. *Disconnected* :menutup *serial port*
6. *Exit* :keluar dari program

```
Algoritma :
    START
       1) Membuka port komunikasi serial.
       2) Menunggu mikrokontroler master mengirim signal bahwa heater
          siap
       3) menunggu perintah user.
          1. Case 1: Default Position
          2. Case 2: start point.
          3. Case 3: 3D Print
          4. Case 4: Disconnected
          5. Case 5: exit program.
    END
```

3.3.5 Fungsi Membuka PORT Komunikasi

Port komunikasi adalah bagian penting yang berfungsi untuk mengirimkan data dari PC ke mikrokontroler tanpa terlebih dahulu membuka PORT komunikasi pengiriman data tidak akan berhasil. Setiap bahasa pemprograman memiliki aturan tersendiri untuk membuka PORT komunikasi tersebut. Dalam penelitian ini bahasa pemprograman yang digunakan adalah JAVA dan program yang digunakan Netbeans 6.9

```
Algoritma :
       START
          1) Melakukan pencarian port yang aktif
          2) If port yang di dapat adalah port serial masukkan ke daftar
             listport
          3) Pengaturan baud rate, 9600 BPS.
          4) Pengaturan tipe serial, 8 Bits, No Parity, 1 Stop Bit.
          5) Setting DTR dan RTS
          6) Setting flow control
          7) Memilih port yang akan digunakan
          8) If terhubung dengan port serial yang diinginkan printf "Com
             X opened successfully"
          9) If port yang dipilih dalam kondisi sedang digunakan printf
             "Com X is in use"
          10) If port yang dipilih tidak terhubung printf "failed to open
             com X"
       END
```

3.3.6 Perintah Posisi Awal pada PC

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
```

3.3.7 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
```

3.3.8 Fungsi Utama Pada Mikrokontroler *Master*

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

1. Posisi awal :mikrokontroler menggerakan *nozzle* ke posisi awal
2. Jalan program :mikrokontroler menjalankan *nozzle* sesuai koordinat yang bertujuan untuk membuat produk
3. *Start point* :mikrokontroler menjalankan *nozzle* sesuai koordinat yang bertujuan untuk mencari titik awal pembuatan produk

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.
          3. Case 3: start point
    END
```

3.3.9 Perintah Posisi Awal pada Mikrokontroler Master

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 position")
       3) Menjalankan motor X ke arah X+ hingga menekan limit switch
       4) Menjalankan motor Z ke arah Z- hingga menekan limit switch
    END
```

3.3.10 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
```

3.3.11 Perintah *Start Point* pada Mikrokontroler *Master*

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 start
          point.
       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
```

3.3.12 Perintah Jalan Program pada Mikrokontrol *Master*

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
      13) Memerintahkan mikrokontroler slave melakukan fungsi jalan
          program.
      14) Memulai loop menerima data per karakter.
      15) Merubah karakter menjadi float.
      16) Memasukan nilai float ke dalam array.
      17) Memberi feed back ke PC.
      18) Menampilkan data yang diterima pada lcd.
      19) Mengirim koordinat X dan Y ke mikrokontroler slave
      20) Menunggu konfirmasi dari mikrokontroler slave.
      21) Menjalankan motor sumbu X
      22) Menjalankan motor sumbu Z
      23) Memberi konfirmasi ke PC.
      24) Menerima data berikutnya.
    END
```

3.3.13 Fungsi Utama Pada Mikrokontroler *Slave*

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

1. Posisi awal :mikrokontroler menggerakan *nozzle* ke posisi awal.
2. Jalan program :mikrokontroler menjalankan *nozzle* sesuai koordinat yang bertujuan untuk membuat produk.
3. *Start point* :mikrokontroler menjalankan *nozzle* sesuai koordinat yang bertujuan untuk mencari titik awal pembuatan produk.

```
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.
          3. Case 3: start point
    END
```

3.3.14 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
```

3.3.15 Perintah Posisi Awal pada Mikrokontroler Slave

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
```

3.3.16 Perintah *Start Point* pada Mikrokontroler Slave

Perintah *start point* 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
```

3.3.17 Perintah Terima Data pada Mikrokontroler Slave

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

```
Algoritma :
    START
       9) Memulai loop menerima data per karakter.
      10) Merubah karakter menjadi float.
      11) Memasukan nilai float ke dalam array.
      12) Memberi feed back pada mikrokontroler master.
      13) Menampilkan koordinat Y pada lcd.
      14) Menjalankan motor sumbu Y.
      15) Memberi konfirmasi ke mikrokontroler master.
      16) Menerima data berikutnya.
    END
```

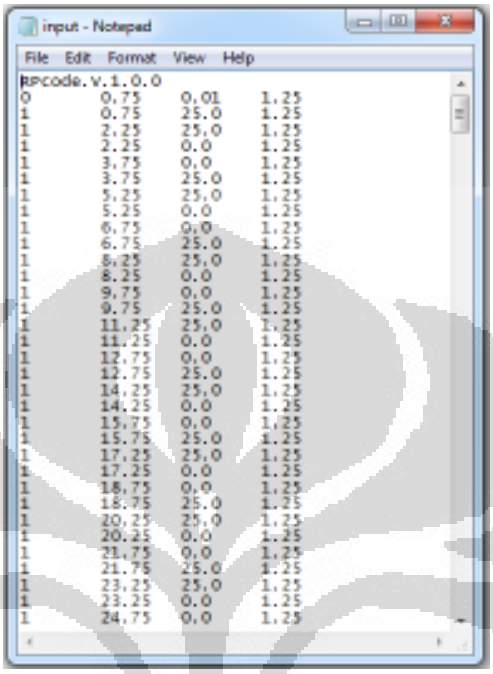
3.3.18 RP Code

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.

File RP *code* memiliki format data sebagai berikut:

<Header file> </n>

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



Gambar 3.20 Format data

Susunan data seperti ini digunakan untuk mempermudah pengiriman data ke dalam 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.

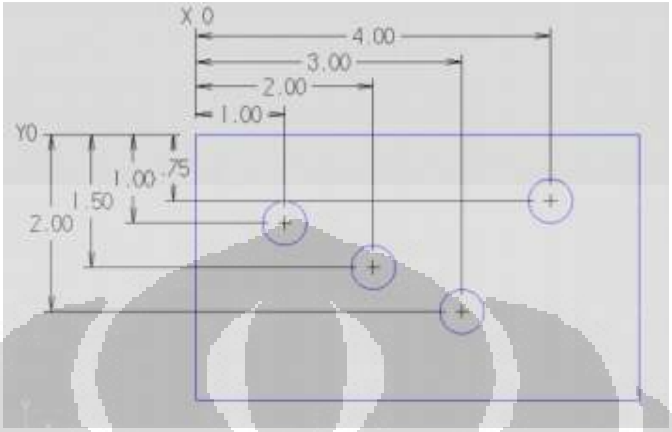
<stringx:num> = Data dengan jenis string, berisi angka yang menandakan koordinat sumbu X dengan ketelitian 0.01 mm.

<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.

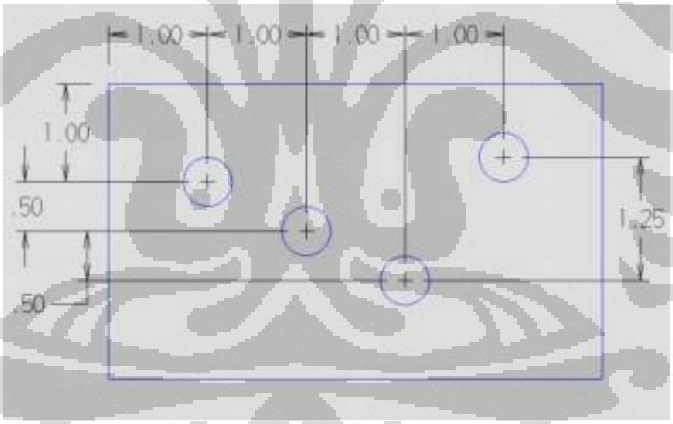
Metode yang digunakan untuk penulisan koordinat adalah metode *absolute*. Perbedaan metode *increment* dan *absolute* terdapat pada titik awal koordinat

berikut setelah satu koordinat dijalankan. Pada metode *absolute* titik awal akan kembali ke nol pada penulisan koordinat.



Gambar 3.21 *Absolut*[13]

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



Gambar 3.22 *Increment*[11]

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

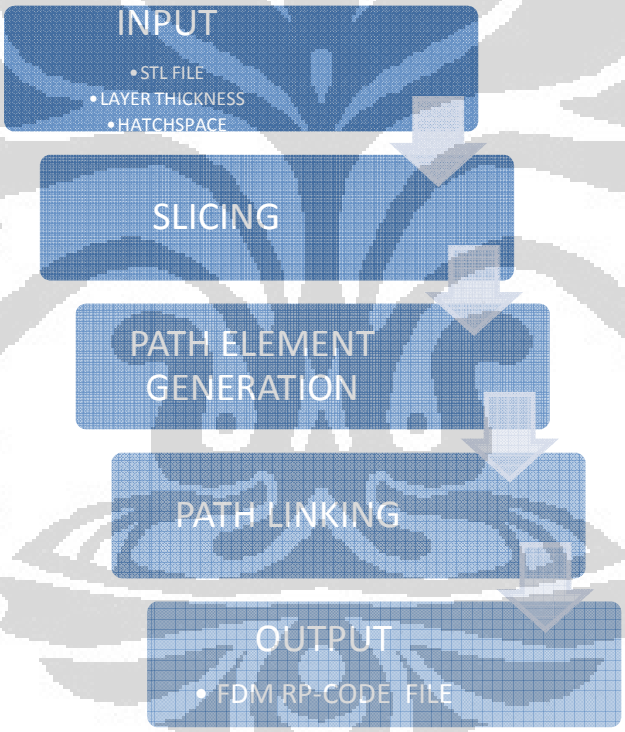
3.4 Pengembangan Model Filling

Pada penelitian sebelumnya, data yang akan dikirim ke mesin rapid prototyping tidak melalui proses *slicing* data tersebut adalah hasil dari interpolasi

yang dilakukan terhadap beberapa titik yang diolah dalam C compiler kemudian di print ke dalam .txt file (Sulaiman, 2010). Hasil dari proses interpolasi titik tersebut adalah kumpulan koordinat lintasan dalam format RP code. Hasildari interpolasi tersebut hanya bisa membuat benda dengan model permukaan dengan ketebalan yang mengikuti lebar dari *output filament*.

3.4.1 Pengembangan *Slicing Algorithm*

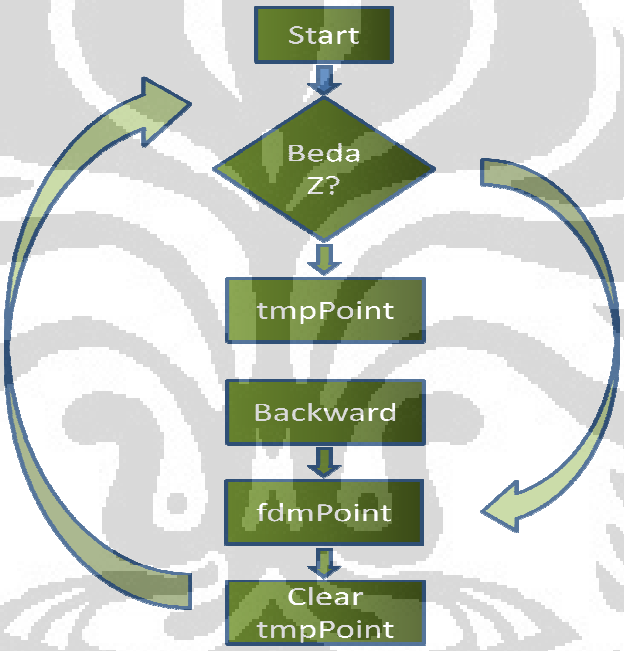
Pada penelitian ini RP *code* dihasilkan dari proses *slicing* dengan menggunakan algoritma pembuatan *laser trajectory* yang telah dikembangkan sebelumnya (Kholil, 2008). Berikut ini adalah gambaran umum pembuatan RP code dengan dengan menggunakan algortima *slicing*.



Gambar 3.23 Alur algoritma *slicing*

Hasil *code* dari penelitian sebelumnya (Kholil, 2008) belum cocok untuk diaplikasikan ke dalam mesin *rapid prototyping*. Program *slicing* tersebut masih diperuntukan bagi *rapid prototyping* berbasiskan *laser sintering* dimana material dan proses *solidifier* materialnya berbeda. Hal fundamental yang menyebabkan tidak bisanya slicing program tersebut diterapkan ke dalam *rapid prototyping*

berbasis FDM adalah karena algoritma jalan program yang digunakan pada mikrokontroler tidak dapat menjalankan ketiga koordinat (X, Y dan Z) sekaligus dalam waktu yang bersamaan. Penggunaan mekanisme *master* dan *slave* pada mikrokontroler hanya dapat menggerakkan koordinat X dan Y secara bersamaan dan koordinat Z dijalankan setelah koordinat X selesai dieksekusi ke motor sehingga perlu adanya perubahan dan penambahan pada *slicing algorithm* tersebut agar dapat diapliksikan ke dalam mesin *rapid prototyping*. Perubahan dan penambahan *algorithm* adalah dengan menerapkan *backward algorithm. Backward algorithm* diterapkan pada tahapan *path lingking* dari program *slicing.*



Gambar 3.24 *Backward algorithm*

Pada tahap *path linking* terjadi proses penghubungan titik hasil dari *slicing* terhadap bidang Z dan bidang X untuk membentuk suatu lintasan benda di setiap *layer*. Titik-titik tersebut dihubungkan setelah menemukan adanya kesejajaran atau kesearahan antara vektor. Ketika telah ditemukan bahwa suatu titik memiliki kesejajaran atau kesearahan terhadap suatu vektor maka titik tersebut langsung di *print* kedalam .txt file.

Untuk dapat menerapkan *backward algorithm* ini, setiap titik (vertex) yang telah di cek kesejajaran atau kesearahannya terhadap suatu vektor dikumpulkan dalam suatu *array vertex*. *Vertex* yang terdapat dalam *array vertex* tersebut adalah kumpulan *vertex* yang urutannya telah membentuk suatu jalur lintasan dan merupakan keseluruhan *vertex* pembentuk dari suatu model. Ketika keseluruhan vertex tersebut telah didapatkan maka *backward algorithm* dijalankan.

BAB 4
ANALISIS DAN PENGUJIAN

Pada hasil dari produk akhir *rapid prototyping* ini terjadi defleksi pada bagian bawah dimana terjadi pengangkatan layer. Besarnya defleksi sangat beragam dan sangat dipengaruhi oleh perbedaan temperatur dan beberapa variabel lain. Karena hal tersebut, dilakukan pengujian terhadap produk untuk melihat seberapa besar pengaruh parameter uji terhadap defleksi yang terjadi. Dalam pengujian ini ada beberapa parameter yang divariasikan, diantaranya adalah

1. Temperatur *nozzle*
2. Temperatur *heatbed*
3. Jumlah *layer* /lapisan
4. Panjang dari benda uji

4.1 Tujuan Pengujian

Pengujian ini bertujuan untuk melihat pengaruh parameter uji dan diameter *nozzle* terhadap defleksi yang terjadi produk *rapid prototyping*

4.2 Metode Pengujian

Pangujian ini dilakukan dengan mengirim data ke mesin *rapid prototyping* yang berisi koordinat yang membentuk suatu lintasan lurus dengan variasi panjang 25, 35, 50, 60 dan 70 mm. proses pembuatan benda uji tersebut dilakukan di atas *heatbed* dengan memvariasikan suhu dari *heatbed* dan *nozzle*. untuk setiap variasi suhu *heatbed* dan *nozzle* dilakukan juga variasi jumlah *layer* pada produk uji. Sebagai contoh, pada suhu *nozzle* 250 $°C$ dan *heatbed* 200 $°C$ dibuat benda uji yang memiliki *layer* sebanyak 5 *layer* dengan panjang 35 mm setelah itu *heatbed* dimatikan dan didiamkan sampai temperaturnya turun (selama 30 detik) setelah itu dilakukan pengukuran. Proses tersebut berulang untuk jumlah *layer* 10, 15 dan 20 dan variasi panjang tersebut.

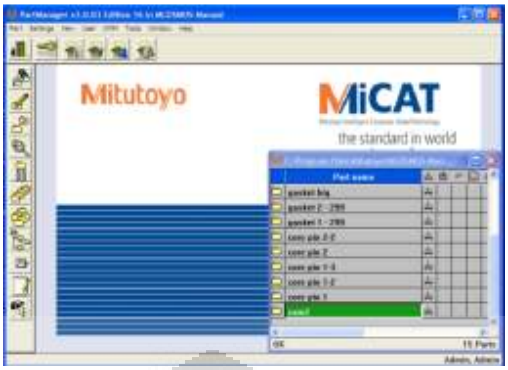
Gambar 4.1 Contoh benda uji

4.3 Metode Pengukuran

Gambar 4.2 CMM probe

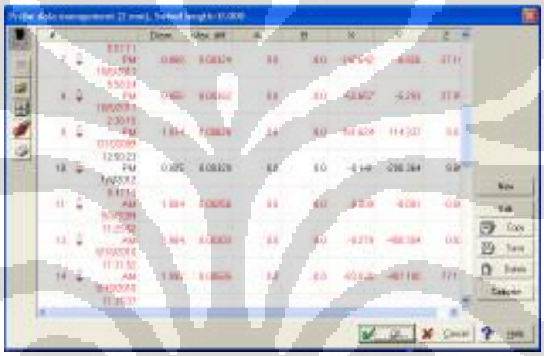
Alat ukur yang digunakan pada penelitian ini adalah CMM *Probe*. Berikut ini adalah langkah-langkah pengukuran untuk mendapatkan data penelitian mengenai defleksi yang terjadi pada benda uji.

1. Setelah benda uji siap di tempat pengukuran, hidupkan mesin dan siapkan jarum *probe* yang akan digunakan. Pilih *probe* yang akan digunakan. Pada penelitian kali ini *probe* yang digunakan adalah *probe* no 10 berdiameter 0.98 mm.
2. Buka software CMM.



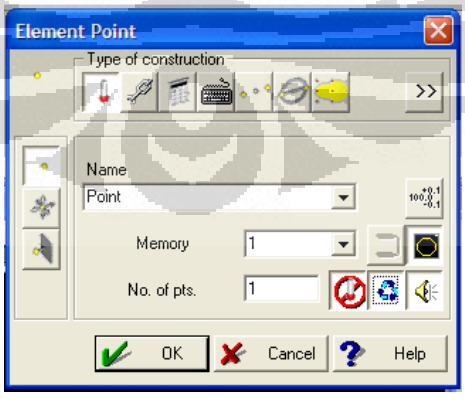
Gambar 4.3 Tampilan awal Mitutoyo Software

3. Pilih *probe* no 10 dengan diameter 0.98 mm.



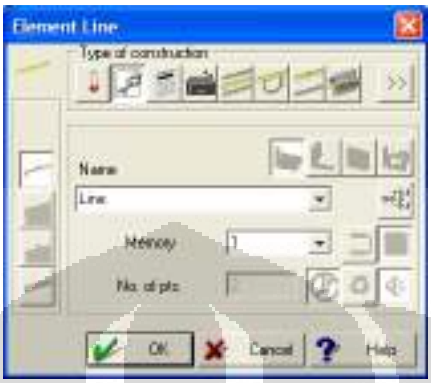
Gambar 4.4 Pemilihan *probe*

4. Setelah memilih *probe* yang digunakan, lakukan pengambilan titik selama 2 kali yaitu titik terujung dari benda uji.



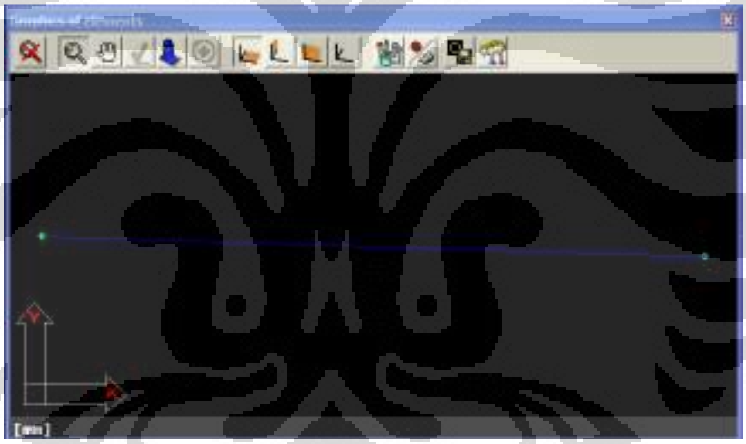
Gambar 4.5 Pengambilan titik

5. Setelah kedua titik di dapat, langkah selanjutnya adalah menghubungkan kedua titik tersebut membentuk suatu garis.



Gambar 4.6 Pembuatan garis

Gambar di bawah ini merupakan hasil dari pembuatan garis dari 2 titik.



Gambar 4.7 Hasil pembuatan garis

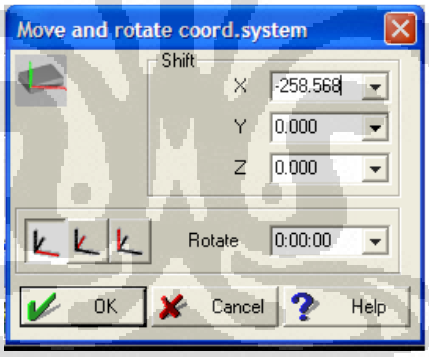
6. *Align* garis yang telah dibuat dengan bidang x-z.

Gambar 4.8 Aligment bidang X-Z

Gambar dibawah ini merupakan hasil dari *aligment* garis pada bidang X-Z, dapat dilihat bahwa koordinat Y dan Z sudah bernilai 0. Selisih antara koordinat X titik pertama dan kordinat X pada titik kedua mempresentasikan panjang dari benda uji.

Gambar 4.9 Hasil aligment bidang X-Z

7. Pindahkan titik pertama (sebelah kiri) agar koordinat X juga bernilai 0

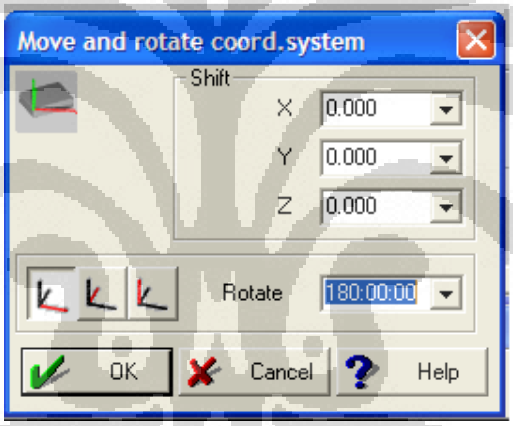


Gambar 4.10 Traslasi garis

Gambar dibawah ini merupakan hasil dari translasi titik kedua searah sumbu X, sekaran titik kedua berkoordinat (0,0,0) sedangkan titik pertama berkoordinat (50.475,0,0) yang artinya benda uji tersebut memiliki panjang 50.475 dan salah satu titiknya sekarang menjadi koordinat pusat.

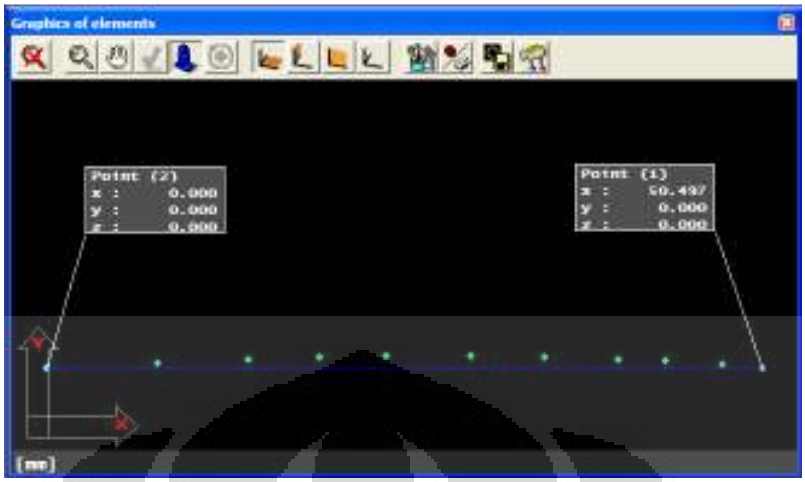
Gambar 4.11 Hasil dari tranlasi garis

8. Rotasikan agar nilai dari defleksi yang tercatat tidak bernilai negatif.



Gambar 4.12 Rotasi garis

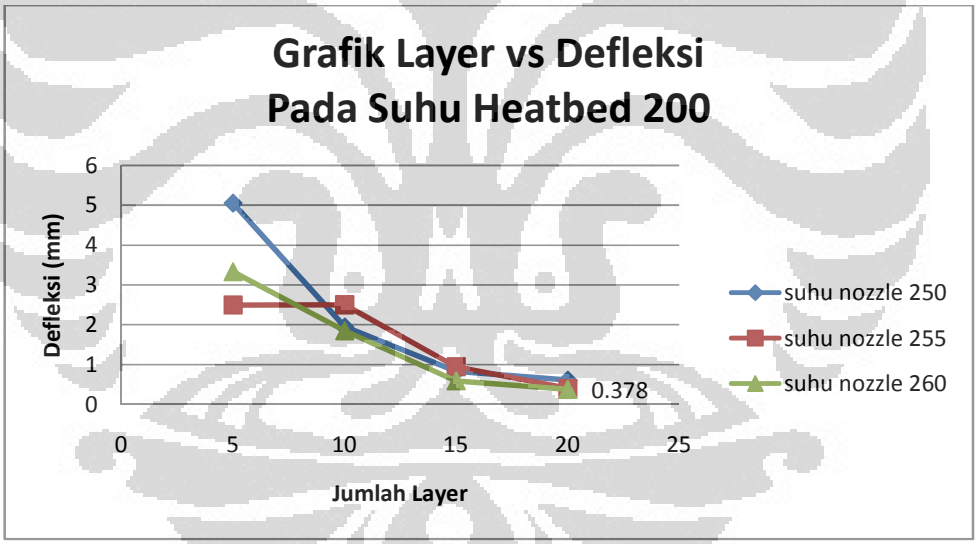
9. Setelah semua selesai, lakukan pengambilan titik selanjutnya menggunakan mode *repeated* sepanjang dimensi benda uji untuk melihat kelengkungan yang terjadi dan maksimum defleksi yang tejadi.



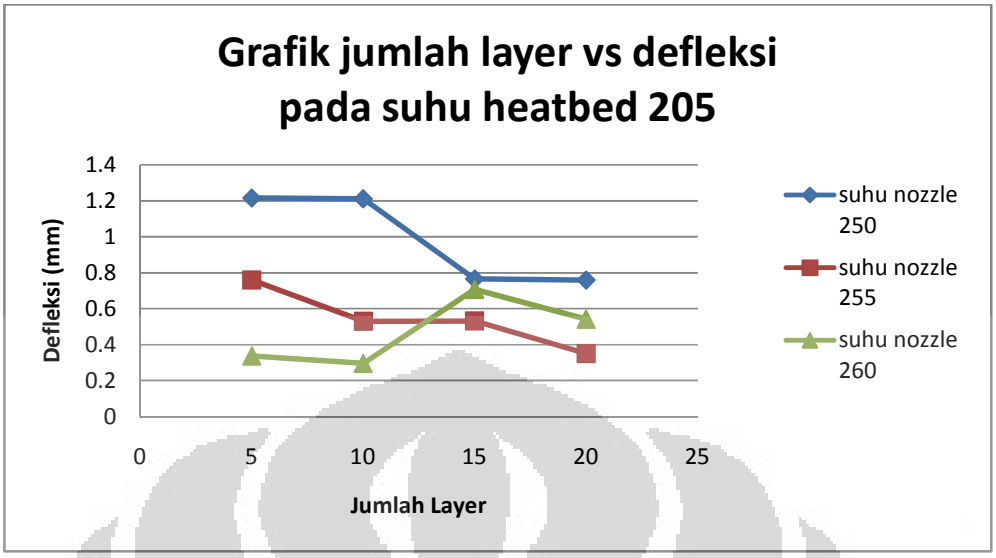
Gambar 4.13 Grafik kelengkungan pada CMM

4.4 Hasil Pengujian

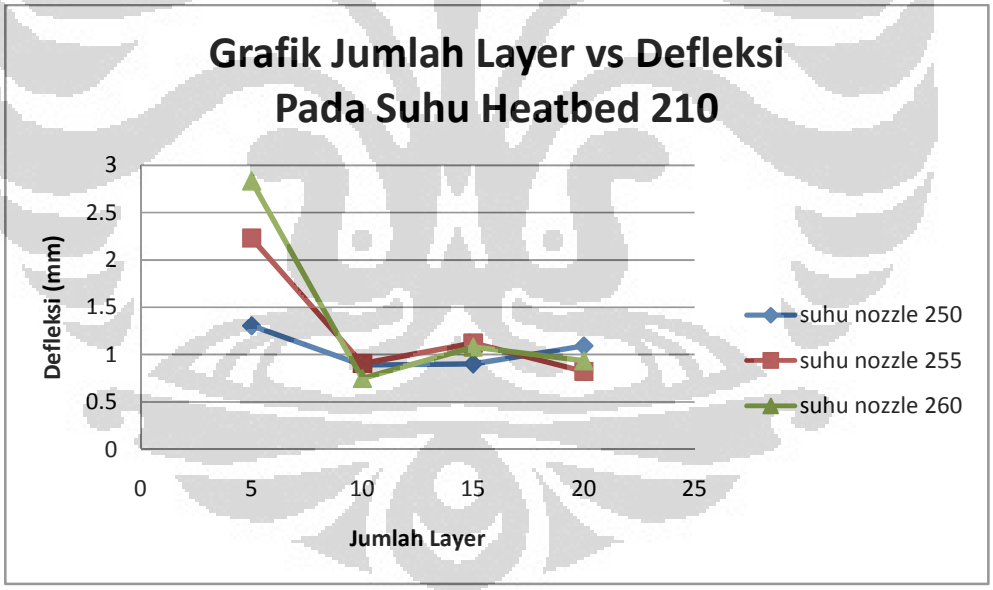
Berikut ini adalah hasil pengujian yang telah dilakukan



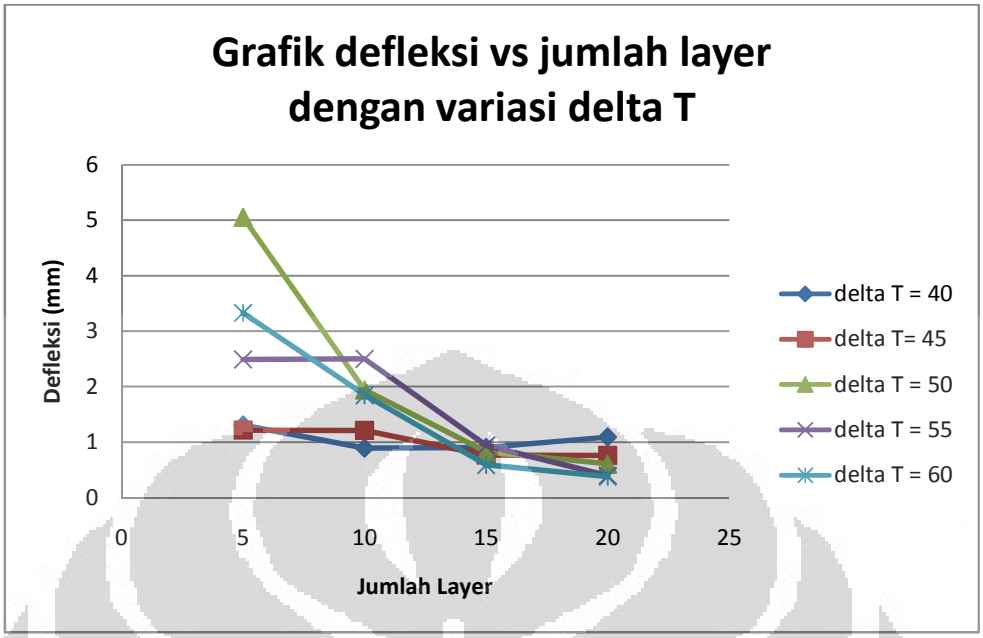
Gambar 4.14 Hubungan jumlah *layer* terhadap defleksi pada suhu *heatbed* 200 °C dan beberapa variasi suhu *nozzle*



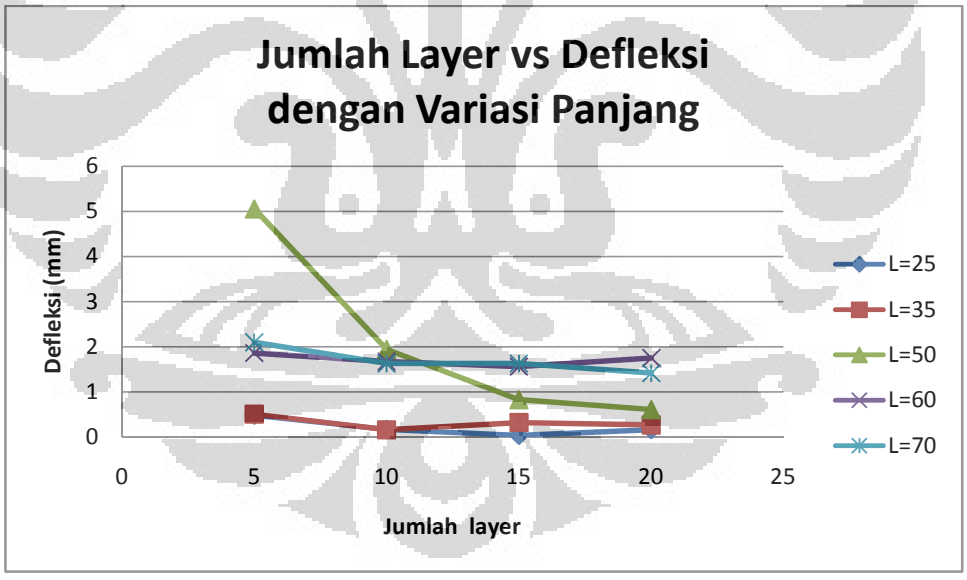
Gambar 4.15 Hubungan jumlah *layer* terhadap defleksi pada suhu *heatbed* 205 °C dan beberapa variasi suhu *nozzle*



Gambar 4.16 Hubungan jumlah *layer* terhadap defleksi pada suhu *heatbed* 210 °C dan beberapa variasi suhu *nozzle*



Gambar 4.17 Hubungan jumlah *layer* terhadap defleksi pada variasi suhu



Gambar 4.18 Hubungan jumlah *layer* terhadap defleksi pada variasi panjang benda uji

4.5 Analisis

4.5.1 Hubungan Jumlah *Layer* Terhadap Defleksi Pada Suhu *Heatbed* 200 °C

Berdasarkan grafik 4.14 diatas, besarnya defleksi menurun seiring dengan bertambahnya jumlah layer, perbedaan yang signifikan terjadi pada jumlah lapisan (n) = 5 sedangkan pada jumlah layer yang lain 10,25 dan 20 besarnya defleksi yang terjadi menunjukan besaran yang mendekati diantara setiap variasi suhu *nozzle*. Hal tersebut disebabkan pada jumlah layer yang rendah menghasilkan volum benda yang lebih kecil sehingga kapasitas kalor (Ct) yang ada di benda tersebut juga menjadi lebih kecil dibandingkan dengan laju pelepasan panas (Q) yang terjadi sehingga proses pendinginan (*cooling time*) menjadi lebih cepat. ketika laju pendinginan lebih cepat maka penyusutan menjadi lebih besar yang diikuti dengan nilai *strain* dan *stress* yang terjadi. Defleksi minimum terjadi pada *setting* suhu *heatbed* 200 °C dan suhu *nozzle* 260 °C, pada jumlah layer ke 20 dengan nilai defleksi 0.378.

$$C_t = \rho \times V \times c_p \ [J/°C]$$

Gambar 4.19 *Thermal capacity* (Ct)

4.5.2 Hubungan Jumlah *Layer* Terhadap Defleksi Pada Suhu *Heatbed* 205 °C

Berdasarkan grafik 4.15 di atas, besarnya defleksi menurun seiring dengan bertambahnya jumlah layer. Namun, pada suhu *nozzle* 260 °C dan jumlah lapisan 15 terjadi peningkatan defleksi dan kemudian turun kembali di *layer* ke 20 , hal tersebut bisa disebabkan karena temperatur *nozzle* yang tidak stabil sehingga yang seharusnya suhu *nozzle* berada pada 260 °C. Namun, pada saat pembuatan *layer* pertama suhu berfluktuatif pada *range* 250-255 °C. Peningkatan temperatur *heatbed* dan peningkatan temperature *nozzle* dapat menyebabkan temperatur menjadi lebih tinggi dan densitas material menjadi lebih tinggi sehingga pada *layer* dengan jumlah *layer* sedikit terjadi peleburan diantara layernya yang mengaburkan batas antara *layer*. Temperatur yang terlalu tinggi dapat membuat luas area pada permukaan bawah menjadi lebih besar. densitas yang tinggi

membuat material lebih cair dan meluber. Luas area yang lebih besar membuat daerah yang bersentuhan dengan pemukaan *heatbed* menjadi lebih besar dan lebih dapat menahan defleksi yang terjadi.

4.5.3 Hubungan Jumlah *Layer* Terhadap Defleksi Pada Suhu *Heatbed* 210 °C

Berdasarkan grafik 4.16 di atas, tren yang terjadi pada kondisi ini menunjukan tren yang sama yaitu terjadi penurunan defleksi seiring dengan peningkatan jumlah layer, namun terjadi fluktuasi pada tingkat lapisan yang lebih besar dan besarnya defleksi lebih besar dibandingkan dengan kondisi pada suhu *heatbed* yang lain (lihat grafik 4.14 dan grafik 4.15) peningkatan suhu *heatbed* tidak berbanding lurus dengan penurunan dari defleksi namun juga dipengaruhi oleh temperatur suhu *nozzle*. perbedaan tren yang terjadi pada kondisi ini dapat disebabkan karena adanya perbedaan temperatur. Kenaikan suhu *heatbed* yang tidak diimbangi dengan suhu keluaran yang tepat (perbedaan temperatur antara keduanya tinggi) maka densitas material ketika menyentuh *heatbed* tidak tinggi dan bisa terjadi suhu *heatbed* tidak dapat lagi mempertahankan suhu material akibat dari perubahan fase yang terlalu cepat ketika material yang keluar dari *nozzle* telah berubah fase maka material akan menjadi *glassy* (licin) dan membutuhkan energi yang lebih besar lagi untuk merubah fase material tersebut.

4.5.4 Hubungan Jumlah *Layer* Terhadap Defleksi Dengan Variasi Beda Temperatur

Berdasarkan grafik 4.17 di atas, penurunan besaran defleksi terjadi seiring dengan peningkatan perbedaan suhu antara suhu *heatbed* dengan suhu *nozzle* selama proses pembuatan karena terjadi konduksi panas pada material. Material yang baru saja keluar dari *nozzle* tidak bisa diasumsikan langsung mencapai suhu temperatur pointnya, namun mengalami transfer energi (joule) dari aliran panas yang berasal dari *heatbed*. aliran panas konduksi pada material menciptakan suatu *temperature profile*. semakin besar beda temperatur antara *heatbed* dengan *nozzle* maka semakin besar aliran energi pada material. ketika memasuki proses pendinginan terjadi perubahan *conductivitas thermal*. konduktivitas thermal adalah besaran yang bersifat *temperature dependent*, dimana besarnya bergantung pada besar dari perbedaan temperatur. semakin besar perbedaan temperatur maka

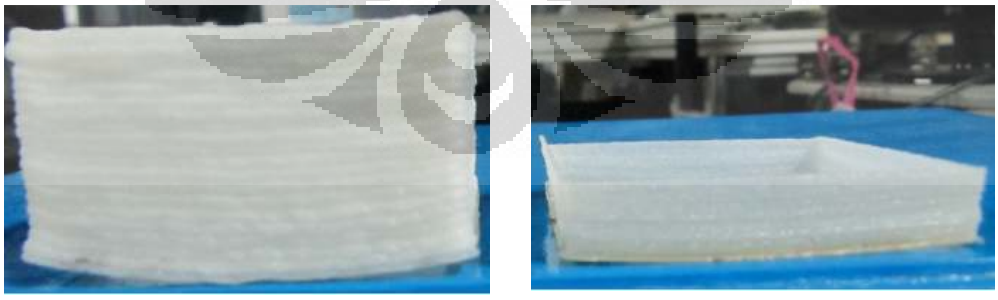
konduktivitas thermalnya akan semakin kecil. ketika konduktivitas thermalnya rendah laju pelepasan kalor yang terjadi akan rendah pula, ditambah energi yang tersimpan pada material (*thermal capacity*) peningkatan temperatur akan meningkat kan jumlah panas yang ada pada benda uji dan dengan laju pendinginan yang rendah waktu pendinginan akan semakin lama.

4.5.5 Hubungan Jumlah *Layer* Terhadap Defleksi Dengan Variasi Panjang Benda Uji

Berdasarkan grafik 4.18 di atas, penurunan besaran defleksi terjadi seiring dengan semakin pendeknya benda uji. berbedaan sangat signifikan terjadi jika dibandingkan dengan benda uji yang lebih panjang.hal ini disebabkan luas area permukaan yang menyentuh *heatbed* lebih kecil dibandingkan dengan benda uji yang lebih panjang. luas area yang lebih kecil mengurangi laju pelepasan kalor pada proses pendinginan. sehingga kalor.yang sejumlah kalor yang ada pada material menahan perubahan temperatur dari *glass temperature* (Tg) menuju *room temperature* (Tr) lebih lama. membuat laju pendinginan menjadi lebih lama juga. Benda uji yang lebih pendek meminimalisir *stress* yang terjadi akibat ada nya *pure bending* yang disebabkan oleh pengaruh thermal.

4.6 Aplikasi Hasil

Berdasarkan pengujian yang telah dilakukan, dapat diketahui bahwa *setting* terbaik untuk mereduksi defleksi adalah pada temperatur *nozzle* 260 $°C$ dan temperatur *heatbed* 200 $°C$ dengan defleksi maksimum yang terjadi sebesar 0.2 mm.

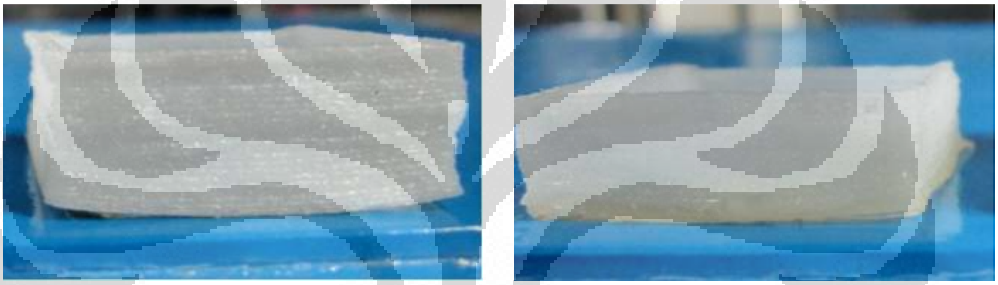


(a) (b)

Gambar 4.20 (a) Produk dengan diameter nozzle 1 mm, (b) Produk dengan diameter nozzle 0.5 mm

Pada gambar 4.20 (a) diatas adalah perbandingan produk yang dihasilkan dengan menggunakan *nozzle* berdiameter 1 mm dengan produk yang dihasillkan dengan menggunakan *nozzle* 0.5 mm. *Output* filamen yang dihasilkan oleh *nozzle* diameter 1 mm sebesar 1 mm , produk tersebut memiliki ketebalan 1.5 mm dan lebar filamen 4 mm, jumlah layer pada produk diatas adalah 20 lapisan dengan tinggi maksimum 30 mm.

Gambar 4.20 (b) adalah produk yang dihasilkan dari penggunaan *nozzle* dengan diameter 0.5 mm dihasilkan dari diameter *output* filamen 0.5 mm memiliki ketebalan 0.5 mm dan lebar filamen 1 mm . Jumlah *layer* pada produk di atas adalah 20 lapisan memiliki ketinggian 10 mm .

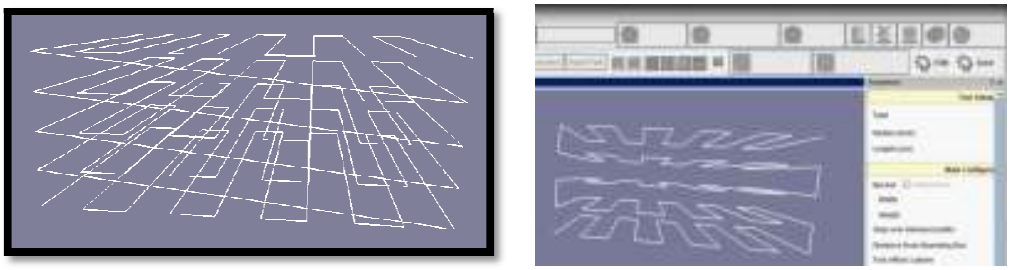


(a) (b)

Gambar 4.21 (a) Produk sebelum pemakaian *heatbed,* (b) Produk setelah pemakian *heatbed*

Gambar 4.22 (a) di atas adalah produk yang dihasilkan sebelum penggunaan *heatbed*, produk ini dihsailkan dengan menggunakan *nozzle* berdiameter 0.5 maksimum defleksi yang terjadi sebesar 1.45 mm. pengukuran defleksi pada produk ini menggunakan CMM *probe* dengan metode pengambilan data yang sama seperti pengambilan data benda uji.

Gambar 4.22 (b) diatas merupakan produk yang dihasilkan setelah penggunaan *heatbed* pasa suhu terukur 200 $°C$ menggunakan *nozzle* 0.5 mm dan memiliki defleksi maksimum 0.253 mm . pengukuran defleksi pada produk ini menggunakan CMM *probe* dengan metode pengambilan data yang sama seperti pengambilan data benda uji.

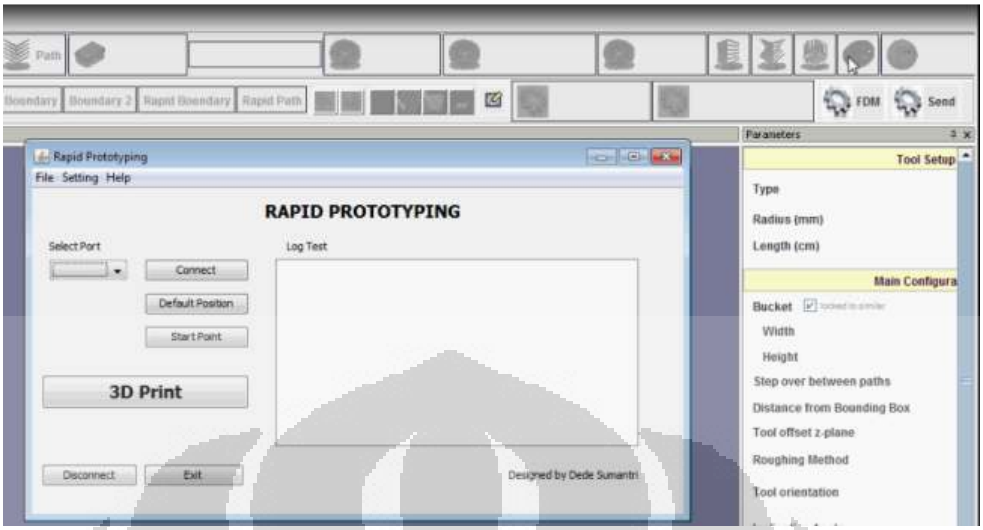


(a) (b)

Gambar 4.22 (a) Sebelum penerapan *backward algorithm, (b)* Setelah penerapan *backward algorithm*

Gambar 4.24 (a) diatas merupakan jalur lintasan sebelum diterapkannya *backward algorithm* , setelah *layer* n terbentuk lintasan n + 1 akan dimulai lagi pada titik awal pembentukan *layer* ke n, jalur lintasan seperti ini belum cocok diterapkan pada *rapid prototyping* berbasis *fused deposition modelling* dengan kondisi mesin yang masih mengeluarkan *output* secara kontinu dari awal sampai dengan akhir pembentukan. Lintasan ini juga tidak efektif dijalankan karena algoritma dari pengiriman data koordinat x, y dan z tidak dapat berjalan secara bersamaan. Kondisi saat ini mengharuskan koordinat z dijalankan setelah koordinat x selesai dieksekusi, sehingga pada saat pergantian *layer nozzle* akan menabrak *layer* yang baru selesai dibuat sebelum akhirnya naik ke l*ayer* selanjutnya.

Gambar 4.24 (b) merupakan visualisasi jalur lintasan yang telah menerapkan *backward algorithm* dimana setelah pembentukan *layer* ke n selesai, pembentukan layer n+1 tidak dmulai lagi dari titik awal pembentukan *layer* ke n namun diteruskan dengan gerakan berlawanan arah dengan pembentukan *layer* ke n. Jalur lintasan setelah diterapkannya *backward algorithm* lebih efektif diterapkan pada mesin *rapid prototyping* terutama dengan tipe pengeluaran material yang masih kontinu.



Gambar 4.23 Komunikasi mikrokontroler Hadiamill

Gambar 4.26 diatas, merupakan visualisasi pengintegrasian *software* yang digunakan untuk melakukan pengiriman data ke mikrokontroler dengan *software* hadiamill.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, beberapa hal yang dapat disimpulkan diantaranya adalah

1. Terjadi peningkatan kinerja mesin dengan hasil berkurangnya *output* filamen dari >1mm menjadi 0.5 mm.
2. Pengurangan diameter *output* berhasil meningkatkan akurasi produk *output* dengan lebar maksimal 1 mm dan ketebalan *layer* 0.5 mm.
3. Perbedaan temperatur antara heatbed dengan nozzle mempengaruhi defleksi yang terjadi, settingan terbaik adalah pada suhu *nozzle* 260 dan suhu *heatbed* 200 dengan maksimum defleksi 0.2 mm.
4. Penerapan *backward algorithm* pada jalur lintasan pada metode pengisian dalam dapat diterapkan untuk *output* material yang kontinu.
5. Baik pada lingkungan konveksi dan konduksi memiliki tren yang sama terhadap variabel jumlah *layer*, panjang benda uji sehingga *heatbed* (konduksi) dapat mensubstitusi ruang penghangat (konveksi).

5.2 Saran

1. Memvariasikan material *rapid prototyping* dengan perbedaan *thermal expansion coefisien*..
2. Meningkatkan ketangguhan algoritma *slicing* pada *platform* JAVA.
3. Memperbaiki sistem *nozzle extruder.*
4. Memvariasikan kecepatan *nozzle* dan kecepatan keluaran material.

DAFTAR REFERENSI

[1] Callister, W. D. (1999). *Materials Science and Engineering An Introduction 5th Edition* . New York: John Wiley and Sons .

[2] Incropera. (2000). *Fundamentals of Heat and Mass Transfer 5th Edition* . John Willey & Sons .

[3] Kholil, A. (2008). *Pengembangan Laser Tranjectory Proses Rapid Prototyping Untuk Produk Berkontur dan Perismatik.* Jakarta: Universitas Indonesia.

[4] Kurniawan, R. (2010). *Pengembangan Sistem Kontrol Mesin Rapid Prototyping Berbasis Fused Deposition Modelling.* Jakarta: Universitas Indonesia.

[5] Maryadi, H. (2010). *Pengembangan Feeder Material Mesin Rapid Prototyping Berbasis FDM (Fused Deposition Modelling).* Jakarta: Universitas Indonesia.

[6] Sulaiman, A. (2010). *Pengembangan Perangkat Lunak mesin Rapid Prototyping dengan Metode FDM (Fused Deposition Modelling).* Depok: Universitas Indonesia.

[7] Tata, S. (2000). *Pengetahuan Bahan Teknik.* Jakarta : Pradnya Paramita.

[8] Tian-Ming-Wang. (2006). *A Model Research for Prototype Warp Deformation in the FDM Proses.* London: Springer.

[9] Welty, J. R. (2001). *Fundamentals of Momentum, Heat and Mass Transfer* . Oregon : John Wlley & sons Inc .

[10] http://computerassistedsurgeryblog.com/2010/05/17/an-overview-of-rapid-prototyping-technology/, dilihat 2 Juli 2010, jam 11.00.

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

[12] Ali K. Kamrani, Emad Abouel Nasr. Rapid Prototyping Theory and Practice. Springer. 2006.

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

[14] AD594/AD595 Datasheet. Monolithic Thermocouple Amplifiers with Cold Junction Compensation. http://www.analog.com/static/imported-files/data_sheets/AD594_595.pdf, download 11 Juni 2010, jam 8.01.

[15] http://blog.makezine.com/arduinorelay.jpg, download 23 November 2010, jam 9.24.

[16] http://img1.topfreebiz.com/o2010-5/cache/Signal-Relays-HRS1--19114541345.jpg, download 9 September 2010, jam 18.36.

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          : 1.1.0
Date             : 10/12/2011
Developer        : Dede Sumantri
Company          : Universitas 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)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// Get a character from the USART2 Receiver
#pragma used+
char getchar2(void)
{
char status,data;
while (1)
     {
     while (((status=UCSR2A) & RX_COMPLETE)==0);
     data=UDR2;
     if     ((status     &     (FRAMING_ERROR     |     PARITY_ERROR     |
DATA_OVERRUN))==0)
        return data;
     };
}
#pragma used-

// Write a character to the USART2 Transmitter
#pragma used+
void putchar2(char c)
{
while ((UCSR2A & DATA_REGISTER_EMPTY)==0);
UDR2=c;
}
#pragma used-

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

// Declare your global variables here


void cw_motor_1(int x)
{
   //PORTA.1 = input 1
   //PORTA.2 = input 2
   //PORTA.3 = input 3
   //PORTA.4 = input 4
   //PORTA.5 = input 5
int us;
//step 1
```

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

//step 2

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

//step 3

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

//step 4

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

//step 5
```

```
    PORTA.1 = 0;
    PORTA.2 = 1;
    PORTA.3 = 1;
    PORTA.4 = 1;
    PORTA.5 = 0;
   for(us=0;us<=x;us++)
    {
    delay_us(1);
    }
//step 6

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

//step 7

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

//step 8

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

//step 9
```

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

//step 10

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


}

void ccw_motor_1(int x)
{
    //PORTA.1 = input 1
    //PORTA.2 = input 2
    //PORTA.3 = input 3
    //PORTA.4 = input 4
    //PORTA.5 = input 5
int us;
//step 1

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

//step 2
```

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

//step 3

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

//step 4

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

//step 5

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

```
//step 6

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

//step 7

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

//step 8

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

//step 9

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

```
//step 10

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

}

void cw_motor_3(int x) //mendekati motor
{
   //PORTE.1 = input 1 blue   PORTE.2
   //PORTE.2 = input 2 red    PORTE.3
   //PORTE.3 = input 3 orange P\ORTE.4
   //PORTE.4 = input 4 green  PORTE.5
   //PORTE.5 = input 5 black  PORTE.6

//step 1

   PORTE.2 = 1;
   PORTE.4 = 0;
   PORTE.6 = 1;
   PORTE.3 = 1;
   PORTE.5 = 0;
   delay_ms(x);

//step 2

   PORTE.2 = 1;
   PORTE.4 = 0;
   PORTE.6 = 0;
   PORTE.3 = 1;
   PORTE.5 = 0;
   delay_ms(x);

//step 3

   PORTE.2 = 1;
   PORTE.4 = 1;
   PORTE.6 = 0;
   PORTE.3 = 1;
```

```
    PORTE.5 = 0;
    delay_ms(x);

//step 4

    PORTE.2 = 0;
    PORTE.4 = 1;
    PORTE.6 = 0;
    PORTE.3 = 1;
    PORTE.5 = 0;
    delay_ms(x);

//step 5

    PORTE.2 = 0;
    PORTE.4 = 1;
    PORTE.6 = 0;
    PORTE.3 = 1;
    PORTE.5 = 1;
    delay_ms(x);

//step 6

    PORTE.2 = 0;
    PORTE.4 = 1;
    PORTE.6 = 0;
    PORTE.3 = 0;
    PORTE.5 = 1;
    delay_ms(x);

//step 7

    PORTE.2 = 0;
    PORTE.4 = 1;
    PORTE.6 = 1;
    PORTE.3 = 0;
    PORTE.5 = 1;
    delay_ms(x);

//step 8

    PORTE.2 = 0;
    PORTE.4 = 0;
    PORTE.6 = 1;
    PORTE.3 = 0;
    PORTE.5 = 1;
    delay_ms(x);
```

```
//step 9

   PORTE.2 = 1;
   PORTE.4 = 0;
   PORTE.6 = 1;
   PORTE.3 = 0;
   PORTE.5 = 1;
   delay_ms(x);

//step 10

   PORTE.2 = 1;
   PORTE.4 = 0;
   PORTE.6 = 1;
   PORTE.3 = 0;
   PORTE.5 = 0;
   delay_ms(x);
}

void ccw_motor_3(int x) //menjauhi motor
{
   //PORTE.1 = input 1  PORTE.2
   //PORTE.2 = input 2  PORTE.3
   //PORTE.3 = input 3  PORTE.4
   //PORTE.4 = input 4  PORTE.5
   //PORTE.5 = input 5  PORTE.6

//step 1

   PORTE.2 = 1;
   PORTE.4 = 0;
   PORTE.6 = 1;
   PORTE.3 = 0;
   PORTE.5 = 0;
   delay_ms(x);

//step 2

   PORTE.2 = 1;
   PORTE.4 = 0;
   PORTE.6 = 1;
   PORTE.3 = 0;
   PORTE.5 = 1;
   delay_ms(x);
//step 3
```

```
    PORTE.2 = 0;
    PORTE.4 = 0;
    PORTE.6 = 1;
    PORTE.3 = 0;
    PORTE.5 = 1;
    delay_ms(x);
//step 4

    PORTE.2 = 0;
    PORTE.4 = 1;
    PORTE.6 = 1;
    PORTE.3 = 0;
    PORTE.5 = 1;
    delay_ms(x);

//step 5

    PORTE.2 = 0;
    PORTE.4 = 1;
    PORTE.6 = 0;
    PORTE.3 = 0;
    PORTE.5 = 1;
    delay_ms(x);

//step 6

    PORTE.2 = 0;
    PORTE.4 = 1;
    PORTE.6 = 0;
    PORTE.3 = 1;
    PORTE.5 = 1;
    delay_ms(x);

//step 7

    PORTE.2 = 0;
    PORTE.4 = 1;
    PORTE.6 = 0;
    PORTE.3 = 1;
    PORTE.5 = 0;
    delay_ms(x);

//step 8

    PORTE.2 = 1;
    PORTE.4 = 1;
    PORTE.6 = 0;
```

```
   PORTE.3 = 1;
   PORTE.5 = 0;
   delay_ms(x);

//step 9

   PORTE.2 = 1;
   PORTE.4 = 0;
   PORTE.6 = 0;
   PORTE.3 = 1;
   PORTE.5 = 0;
   delay_ms(x);

//step 10

   PORTE.2 = 1;
   PORTE.4 = 0;
   PORTE.6 = 1;
   PORTE.3 = 1;
   PORTE.5 = 0;
   delay_ms(x);
}
void motor_stop()
{
   PORTA = 0x00;
   PORTC = 0x00;
   PORTE = 0x00;
}

void posisi_awal()
{
int i;

putchar2('A');

lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Default Position");


 //sumbu X+
   for (i=1;i<=10000;i++)
   {
      if (PIND.0==0)
      {
         motor_stop();
         break;
```

```
        }
        else
        {
           cw_motor_1(210); //menjauhi motor
           //ccw_motor_1(); //mendekati motor
        }
     }

//sumbu Z+
    for (i=1;i<=10000;i++)
    {
       if (PIND.4==0)
       {
          motor_stop();
          break;
       }
       else
       {
          cw_motor_3(2); //mendekati motor
       }
    }


}

/**

algoritma starting point untuk setting point dimana
akan memulai melakukan pengiriman data
*/

  void start_point(){


    int counter  ;
    int k        ;
    int a ;
    int b ;
    int kk ;
    int ii ;

    char input_data [16];
    float input_list [3] ;
    float koordinatx ;
    float koordinaty ;
    float koordinatz ;
    float positif ;
```

```
  float kuadrat ;
  float c ;
  float delay ;
  float delay_time ;
  float storage_start [3];

  char lcdx [16];
  char lcdy [16];
  char lcdz [16];
  char waiting ;

  putchar2('S') ;

  storage_start[0] = 0;
storage_start[1] = 0;
storage_start[2] = 0;
//storage_start[3] = 0;

while(1){

   for(counter = 0; counter <3; counter++) //untuk mengambil angka perbaris
    {
       lcd_clear();
       lcd_gotoxy(0,0);


       k = 0;
       input_data[k] = getchar();
       while(input_data[k] != 'x')
       {
         //lcd_putchar(input[i]);
         k++;
         input_data[k] = getchar();
       }

       input_data[k] = '\0';
       input_list[counter] = atof(input_data);
       putchar ('k');

    }

    lcd_clear();

    putchar('n');


    koordinatx=(input_list[0]-storage_start[0])/0.032;
```

```
koordinaty=(input_list[1]-storage_start[1])/0.032;
koordinatz=(input_list[2]-storage_start[2])/0.032;



kuadrat=(koordinatx*koordinatx)+(koordinaty*koordinaty);
c =sqrt (kuadrat);
delay_time =((c /koordinatx)*190) ;


ftoa(input_list[0],2,lcdx);
ftoa(input_list[1],2,lcdy);
ftoa(input_list[2],2,lcdz);

storage_start[0] = input_list[0];
storage_start[1] = input_list[1];
storage_start[2] = input_list[2];

for (a=0;a<strlen(lcdx);a++)
{
   lcd_gotoxy(a,0);
   lcd_putchar(lcdx[a]);
}

for (a=0;a<strlen(lcdy);a++)
{
   lcd_gotoxy(a,1);
   lcd_putchar(lcdy[a]);
}

for (a=0;a<strlen(lcdz);a++)
{
   lcd_gotoxy(a,2);
   lcd_putchar(lcdz[a]);
}



for(b=0 ; b<strlen(lcdx); b++) // mengirim koordinat x ke MC Slave
{
   putchar2  (lcdx[b]);

}

putchar2('x');

// delay_ms(10);
```

```
for(b=0 ; b<strlen(lcdy); b++)
{
   putchar2  (lcdy[b]); // mengirim koordinat y ke MC Slave
}

putchar2('x');


putchar('N');

waiting=getchar2();

while(waiting != 'y')
{
   lcd_putsf("Waiting...\n");
   waiting = getchar2();
}

if (delay_time<0)
{
   delay=delay_time*(-1);

      if(koordinatx < 0)
      {

         kk = koordinatx*(-1);


         for (ii=1;ii<=kk;ii++)
         {
            cw_motor_1(delay);

         }
      }
      else
      {
         for (ii=1;ii<=koordinatx;ii++)
         {
            ccw_motor_1(delay);
         }
      }
   }
else
{
    if(koordinatx < 0)
```

```
    {

       kk = koordinatx*(-1);


       for (ii=1;ii<=kk;ii++)
       {
          cw_motor_1(delay_time);

       }
    }
  else
    {
       for (ii=1;ii<=koordinatx;ii++)

       {
           ccw_motor_1(delay_time);
       }
    }
}


putchar2('y');
waiting=getchar2();

while(waiting != 'y')
{
    lcd_putsf("Waiting...\n");
    waiting = getchar2();
}

if(koordinatz>0)
{

   for (ii=0;ii<=koordinatz;ii++)
   {
      ccw_motor_3(1);
   }

}
 else
{
   positif = -koordinatz ;

   for (ii= 0;ii <= positif ;ii++ )
   {
```

```
            cw_motor_3 (1);

          }

        }

        putchar('y');

        break;

      }


}


void jalan_program()
{

int i;
int j;
int m;
//int counter;


float xy;
float xy1;
float k;
float ddx;
float kirimx;
float kirimy;
float kirimg;
float kirimz;
float InputList [4];
float storage [4];

float delay;
char kirimy1[16];
char kirimx1[16];
char kirimg1[16];
```

```
char kirimz1[16];
char response;
char input[16];

putchar2('B');

  // inisialisasi variabel storage
 storage[0] = 0;
 storage[1] = 0;
 storage[2] = 0;
 storage[3] = 0;

// counter = 0;
while (1)
 {

for(j = 0; j <4; j++) //untuk mengambil angka perbaris
      {
         lcd_clear();
         lcd_gotoxy(0,0);

         //Untuk MC2 dari sini
         i = 0;
         input[i] = getchar();

         while(input[i] != 'x')
         {
           //lcd_putchar(input[i]);
           i++;
           input[i] = getchar();
         }
         input[i] = '\0';


         InputList[j] = atof(input);

         putchar ('k');

      }

      lcd_clear();

      putchar('n');
      //lcd_putsf("Processing...");
```

```
      //delay_ms(5000); //Lakukan proses terhadap motor, kirim inputList-nya
(index ke-1 = x, 2 = y, 3 = z)
      kirimg=InputList[0] ;
      kirimx=(InputList[1]-storage[1])/0.032;
      kirimy=(InputList[2]-storage[2])/0.032;
      kirimz=(InputList[3]-storage[3])/0.032;

      xy=(kirimx*kirimx)+(kirimy*kirimy);
      xy1=sqrt (xy);
      ddx=((xy1/kirimx)*190) ;
    //  ddy=((xy1/kirimy)*1) ;
      ftoa(kirimg,2,kirimg1);
      ftoa(InputList[1],2,kirimx1);
      ftoa(InputList[2],2,kirimy1);
      ftoa(InputList[3],2,kirimz1);


      storage[1] = InputList[1];
      storage[2] = InputList[2];
      storage[3] = InputList[3];

      for (m=0;m<strlen(kirimx1);m++)
      {
         lcd_gotoxy(m,0);
         lcd_putchar(kirimx1[m]);
      }

      for (m=0;m<strlen(kirimy1);m++)
      {
         lcd_gotoxy(m,1);
         lcd_putchar(kirimy1[m]);
      }

      for (m=0;m<strlen(kirimz1);m++)
      {
         lcd_gotoxy(m,2);
         lcd_putchar(kirimz1[m]);
      }
      //delay_ms (5000);

      for(j=0 ; j<strlen(kirimg1); j++)
      {
         putchar2  (kirimg1[j]); //tergantung usart yang digunakan
      }
      putchar2('x');
```

```
for(j=0 ; j<strlen(kirimx1); j++)
{
   putchar2  (kirimx1[j]); //tergantung usart yang digunakan
}
putchar2('x');

// delay_ms(10);


for(j=0 ; j<strlen(kirimy1); j++)
{
   putchar2  (kirimy1[j]); //tergantung usart yang digunakan
}
putchar2('x');


response=getchar2();

while(response != 'y')
{
   lcd_putsf("Waiting...\n");
   response = getchar2();
}

if (ddx<0)     // ddx<0 ---> ccw
   {
      delay=ddx*(-1);

      if(kirimx < 0)
      {

         k = kirimx*(-1);


         for (i=1;i<=k;i++)
         {
            cw_motor_1(delay);  // menjauhi motor

         }
      }
      else
      {
         for (i=1;i<=kirimx;i++)
         {
            ccw_motor_1(delay);
         }
      }
```

```
    }
  else
    {
     if(kirimx < 0)
       {

          k = kirimx*(-1);


          for (i=1;i<=k;i++)
          {
             cw_motor_1(ddx);

          }
       }
       else
       {
          for (i=1;i<=kirimx;i++)

          {
             ccw_motor_1(ddx);          //mendekati motor , sudah dirubah
arahnya
          }
       }
       }

      putchar2('y');
      response=getchar2();

      while(response != 'y')
      {
        lcd_putsf("Waiting...\n");
        response = getchar2();
      }


   if(kirimz>0)
     {

        for (i=0;i<=kirimz;i++)
        {
          ccw_motor_3(1);
        }

     }
     putchar('y');
```

```
 }
}

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=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
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=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTC=0x00;
DDRC=0xFF;

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
// State7=1 State6=1 State5=1 State4=1 State3=1 State2=1 State1=1 State0=1
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=0x7C;

// Port F initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTF=0xFF;
DDRF=0xFF;

// Port G initialization
// Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State5=T State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;

// Port H 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
PORTH=0x00;
DDRH=0x00;

// Port J initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTJ=0x7C;
DDRJ=0x7C;

// Port K initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTK=0x00;
DDRK=0xFF;

// Port L 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
PORTL=0x00;
```

```
DDRL=0x00;


// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 9600
UCSR0A=0x00;
UCSR0B=0x18;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;

// USART2 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART2 Receiver: On
// USART2 Transmitter: On
// USART2 Mode: Asynchronous
// USART2 Baud Rate: 9600
UCSR2A=0x00;
UCSR2B=0x18;
UCSR2C=0x06;
UBRR2H=0x00;
UBRR2L=0x67;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
ADCSRB=0x00;

// LCD module initialization
lcd_init(20);


    // Place your code here

      temperatur=getchar2();
      while(temperatur != 'z')
      {
       lcd_clear();
      lcd_gotoxy(0,0);
      lcd_putsf("Waiting...\n");
```

```
        temperatur = getchar2();
        }
        putchar('z');

        while (1)
        {
        perintah = getchar ();

        switch(perintah)

        {
        case 'A' :
           putchar ('A');
           posisi_awal();
           break;
        case 'B' :
           jalan_program();
           break;
        case 'S'   :
           putchar ('S');
           start_point ();
           break;
        }

        putchar('z');

        }

}
```

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         : 1.1.0
Date            : 10/12/2011
Developer       : Dede Sumantri
Company         : Universitas 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)
#define RX_COMPLETE (1<<RXC)

// Get a character from the USART2 Receiver
#pragma used+
char getchar2(void)
{
char status,data;
while (1)
      {
      while (((status=UCSR2A) & RX_COMPLETE)==0);
      data=UDR2;
      if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
         return data;
      };
}
#pragma used-

// Write a character to the USART2 Transmitter
#pragma used+
void putchar2(char c)
{
while ((UCSR2A & DATA_REGISTER_EMPTY)==0);
UDR2=c;
}
#pragma used-

// Get a character from the USART3 Receiver
#pragma used+
char getchar3(void)
{
char status,data;
while (1)
      {
      while (((status=UCSR3A) & RX_COMPLETE)==0);
      data=UDR3;
      if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
         return data;
      };
}
#pragma used-

// Write a character to the USART3 Transmitter
#pragma used+
```

```
void putchar3(char c)
{
while ((UCSR3A & DATA_REGISTER_EMPTY)==0);
UDR3=c;
}
#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++)
    {
    delay_us(1);
    }
//step 4

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

//step 5

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

//step 6

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

//step 7

    PORTC.1 = 0;
```

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

//step 8

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

//step 9

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

//step 10

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

}
```

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

//step 3

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

//step 4

   PORTC.1 = 0;
```

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

//step 5

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

//step 6

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

//step 7

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

//step 8
```

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

//step 9

    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 10

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

}

void motor_stop()
{
    PORTA = 0x00;
    PORTC = 0x00;
    PORTE = 0x00;
}

void posisi_awal()
{

    int i;
```

```
    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_putsf("Default Position");



    //sumbu Y-
    for (i=1;i<=10000;i++)
    {
       if (PINE.3==0)
       {
          motor_stop();
          break;
       }
       else
       {
          ccw_motor_2(210); //mendekati motor
       }
    }

}

void start_point()
{
    int a;
    int b,c;


    char input[16];
    float xy;
    float xy1;
    float k;
    float ddy;
    float kirimx;
    float kirimy;


    float list [3];
    float storage [3];
    char waiting;
    char kirimy1[16];


    float delay;
    storage[0] = 0;
    storage[1] = 0;
```

```
while(1)
 {
      for(b = 0; b <2; b++)
      {
         lcd_clear();
         lcd_gotoxy(0,0);
         a = 0;
         input[a] = getchar2();

         while(input[a] != 'x')
         {
            lcd_putchar(input[a]);
            a++;
            input[a] = getchar2();
         }
         input[a] = '\0';

         list[b] = atof(input);


      }

      kirimx=(list[0]-storage[0])/0.032;
      kirimy=(list[1]-storage[1])/0.032;


      ftoa (kirimy,3,kirimy1);
      for (c=0;c<strlen(kirimy1);c++)
      {
         lcd_gotoxy(c,0);
         lcd_putchar(kirimy1[c]);
      }

      xy=(kirimx*kirimx)+(kirimy*kirimy);
      xy1=sqrt (xy);
      ddy=((xy1/kirimy)*190) ;


      storage[0]= list[0];
      storage[1]= list[1];

      putchar2('y');
```

```
if (ddy<0)
{
   delay=ddy *(-1);
   if(kirimy < 0)
   {

      k = kirimy *(-1);


      for (a=1;a<=k;a++)
      {
         ccw_motor_2(delay);     //ccw itu mendekati motor
      }
   }
   else
   {
      for (a=1;a<=kirimy;a++)

      {
         cw_motor_2(delay);
      }
   }
}
else
{
   if(kirimy < 0)
   {

      k = kirimy*(-1);


      for (a=1;a<=k;a++)
      {
         ccw_motor_2(ddy);     //menjauhi motor
      }
   }
   else
   {
      for (a=1;a<=kirimy;a++)

      {
         cw_motor_2(ddy);     //mendekati motor
      }
   }
}

waiting=getchar2();
```

```
        while(waiting != 'y')
        {
           lcd_putsf("Waiting...\n");
           waiting = getchar2();
        }

        putchar2('y');

        break ;
  }


}



void terima_data ()
{
   int i;
   int j,m;


   char input[16];
   float xy;
   float xy1;
   float k;
   float ddy;
   float kirimx;
   float kirimy;

   float list [3];
   float storage [3];
   char response;
   char kirimy1[16];


   float delay;
   storage[0] = 0;
   storage[1] = 0;
   storage[2] = 0;

while(1)
  {
        for(j = 0; j <3; j++)
        {
           lcd_clear();
           lcd_gotoxy(0,0);
```

```
    i = 0;
    input[i] = getchar2();
    while(input[i] != 'x')
    {
      //  lcd_putchar(input[i]);
        i++;
        input[i] = getchar2();
    }
    input[i] = '\0';

    list[j] = atof(input);


}

if (list[0]==1)
{
    putchar3('y');
}
else
{
    putchar3('u');
}

kirimx=(list[1]-storage[1])/0.032;
kirimy=(list[2]-storage[2])/0.032;
ftoa (kirimy,3,kirimy1);
for (m=0;m<strlen(kirimy1);m++)
{
    lcd_gotoxy(m,0);
    lcd_putchar(kirimy1[m]);
}
xy=(kirimx*kirimx)+(kirimy*kirimy);
xy1=sqrt (xy);
ddy=((xy1/kirimy)*190) ;


storage[1]= list[1];
storage[2]= list[2];

putchar2('y');

if (ddy<0)
{
    delay=ddy *(-1);
    if(kirimy < 0)
    {
```

```
        k = kirimy *(-1);

        for (i=1;i<=k;i++)
        {
           ccw_motor_2(delay);     //ccw itu mendekati motor
        }
     }
     else
     {
        for (i=1;i<=kirimy;i++)

        {
           cw_motor_2(delay);
        }
     }
  }
  else
  {
     if(kirimy < 0)
     {

        k = kirimy*(-1);


        for (i=1;i<=k;i++)
        {
           ccw_motor_2(ddy);     //menjauhi motor
        }
     }
     else
     {
        for (i=1;i<=kirimy;i++)

        {
           cw_motor_2(ddy);     //mendekati motor
        }
     }
  }

  response=getchar2();
  while(response != 'y')
  {
     lcd_putsf("Waiting...\n");
     response = getchar2();
  }
```

```
      putchar2('y');
  }
}


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
PORTF=0xFF;
DDRF=0xFF;

// Port G initialization
// Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State5=T State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;

// Port H 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
PORTH=0x00;
DDRH=0x00;

// Port J 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
PORTJ=0xFF;
DDRJ=0xFF;

// Port K 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
PORTK=0x00;
DDRK=0xFF;

// Port L 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
```

```
PORTL=0x00;
DDRL=0x00;

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 9600
UCSR0A=0x00;
UCSR0B=0x18;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;

// USART2 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// 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;
     case 'S' :
        start_point () ;
        break;

    }

    };
}
```

LAMPIRAN 3

```
/*****************************************************
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 Temperatur
Version          : 1.1.0
Date             : 10/12/2011
Developer        : Dede Sumantri
Company          : Universitas Indonesia
Comments         :


Chip type              : Atmega16
Program type           : Application
Clock frequency        : 16.000000 MHz
Memory model           : Small
External RAM size      : 0
Data Stack size        : 256
*****************************************************/

#include <mega16.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
#include <stdlib.h>
#include <delay.h>
#include <math.h>
#include <stdio.h>


#define ADC_VREF_TYPE 0x00

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
```

```
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}


void main(void)
{
// Declare your local variables here
float temp_nozzle[32];
float temp_heatbed [32];
float suhu_nozzle, suhu_heatbed,suhu1,suhu2,total_tempnozzle,
total_tempheatbed;
int i,j;
char LCD_nozzle[16];
char LCD_heatbed [16];

// Input/Output Ports initialization
// Port A initialization
PORTA=0x00;
DDRA=0x00;

// Port B initialization
PORTB=0x00;
DDRB=0x01;

// Port C initialization
PORTC=0x00;
DDRC=0x00;

// Port D initialization
PORTD=0x00;
DDRD=0x38;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;
```

```
// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: Free Running
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0xA7;
SFIOR&=0x1F;

// LCD module initialization
lcd_init(16);

while (1)
      {
      // Place your code here
      //Get data 32 times

        for(i=0;i<=31;i++)
          {
             temp_nozzle[i]= read_adc(1)*0.48828125 ;

          }

          for(i=0;i<=31;i++)
          {

             temp_heatbed[i]= read_adc(2)*0.48828125 ;
          }

     //Sum
       total_tempnozzle = 0;
       total_tempheatbed = 0 ;


          for(j=0;j<=31;j++)
          {
             total_tempnozzle += temp_nozzle[j];
          }
          for(j=0;j<=31;j++)
          {

             total_tempheatbed += temp_heatbed[j];
          }


     //Average
     suhu_nozzle = (total_tempnozzle/32);
```

```
    suhu_heatbed = (total_tempheatbed/32);
    suhu1 = ((suhu_nozzle*1.006) - 0.657);
    suhu2 = ((suhu_heatbed*1.006) - 0.657);

    //suhu = (read_adc(0)*0.48828125);
    ftoa (suhu1,1,LCD_nozzle);
    ftoa (suhu2,1,LCD_heatbed);


    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_puts(LCD_nozzle);
    lcd_gotoxy(6,0);
    lcd_putsf("Celcius");


    lcd_gotoxy(0,1);
    lcd_puts(LCD_heatbed);
    lcd_gotoxy(6,1);
    lcd_putsf("Celcius");


        //delay_ms(1500);

        if (suhu_nozzle >= 250){

           PORTB.0 = 1 ;
           putchar ('c');

                 }

        else {
           PORTB.0 = 0;
        }
        if (suhu_heatbed >= 200){

           PORTB.1 = 1 ;
                 }
        else {

           PORTB.1 = 0 ;

        }
             delay_ms(1000);
         };
}
```

LAMPIRAN 4

```
/**

Project          : Slicing Algoritm
Compiler         : Netbean 6.9 Java Compiler
Package          : id.ac.ui.eng.hadiamil.machining.prototyping
Java Class       : get_coordinate_model.java
Version          : 1.1.0
Date             : 27/12/2011
Developer        : Dede Sumantri
Company          : Universitas Indonesia
Comments         :

*/



package id.ac.ui.eng.hadiamil.machining.prototyping;

import id.ac.ui.eng.hadiamil.app.Setup;
import id.ac.ui.eng.hadiamil.app.controller.MyActionListener;
import id.ac.ui.eng.hadiamil.app.ui.MainFrame;

import java.io.*;

import java.util.*;
import java.text.* ;


/**
 *
 * @author soemantri
 */
public class get_coordinate_model {

private static int counter = 0 ;
private static int counter_vertex = 0;
private static int counter_index_vertex = 0;

private static int  row = 1;
public static int jumlah_segitiga = 1  ;
public                  static     double     tinggi_angkat     =
FdmPrototyping.layer_thickness ;


 // static int index = 0 ;


 /**
  *    menurut  saya,  bentuk  array  didefinisikan  sebagai  public
static saja., karena
  * apa yang ada dalam array tersebut akan terpakai terus sampai
dengan akhir program
```

```
  * sehingga array tersebut mudah untuk diakses dengan diolah
dikelas lain
  *
  */


static int [] index_vertex  = new int [1000];
static int [][] index_segitiga_vertex  = new int [10000][1000];
static int [][] array_normal = new int [1000][10];
static double koordinat_vertex  [][]= new double [10000][3];

static double  array_normalx []  = new double [10000];
static double  array_normaly []  = new double  [10000];
static double array_normalz [] = new double [10000];

static double  array_vertexx [] = new double [100000];
static double array_vertexy []  = new double [100000];
static double  array_vertexz [] = new double [100000];

static double  xminmax [] = new double [2];
static double yminmax [] = new double [2];
static double zminmax []  = new double[2];


private static    Scanner x ;

public static void open_file_stl (){
    try {

      // "D:/semester x/produk/model4/baut.stl"
      //     "D:/semester   x/java/stl/model_1.stl"   "D:/semester
x/produk/model3/Part3.stl"
      //MyActionListener.dd
         x    =    new    Scanner    (new    File    (    "D:/semester
x/produk/model3/Part2.stl"));
    }
        catch(Exception e){
            System.out.println ("couldnot find file");


        }


}


public static void get_coordinate_stl (){

    xminmax [0]= 1000;
    xminmax [1]= -1000;

    yminmax [0]= 1000;
    yminmax [1]= -1000;

    zminmax [0]= 1000;
    zminmax [1]= -1000;
```

```
    int index_segitiga = 1;


while (x.hasNext()){


    String a = x.next();

     if (a.equals ("facet")){


       String b = x.next(); // collecting "normal" String words

       String normalx = x.next();
       String normaly = x.next();
       String normalz = x.next();

        DecimalFormat df = new DecimalFormat("#.##");

       float normal_x = Float.parseFloat(normalx);
       array_normalx [counter]= normal_x ;
       float normal_y = Float.parseFloat(normaly);
       array_normaly [counter]= normal_y ;
       float normal_z = Float.parseFloat(normalz);
       array_normalz [counter]= normal_z ;

       counter ++ ;

        }

     else if (a.equals("vertex")) {

      String vertexx = x.next();
      String vertexy = x.next();
      String vertexz = x.next();

       // DecimalFormat df = new DecimalFormat("#.##");

      double vert_x = Double.parseDouble(vertexx);
      int vertex_x = (int)vert_x ;
      array_vertexx[counter_vertex] = vertex_x;
      double vert_y =Double.parseDouble(vertexy);
      int vertex_y = (int)vert_y ;
      array_vertexy[counter_vertex] = vertex_y;
      double vert_z = Double.parseDouble(vertexz);
      int vertex_z = (int)vert_z ;
      array_vertexz[counter_vertex] = vertex_z;


        //System.out.println(array_vertexx[counter_vertex]   +"\t"+
array_vertexy                [counter_vertex]                +"\t"+
array_vertexz[counter_vertex]);
        // System.out.println(row);

        koordinat_vertex[row][0]= 0.001 ;
        koordinat_vertex[row][1]= 0.001 ;
        koordinat_vertex[row][2]= 0.001 ;
```

```
      int index = 0 ;
        for (int index_sama = 1 ; index_sama <= row ;index_sama
++){

           // System.out.println (index_sama);

                      //koordinat yang sudah tersimpan          //
koordinat yang baru diambil

            if(          Math.abs(koordinat_vertex[index_sama][0]-
array_vertexx[counter_vertex])<0.0001&&
                Math.abs(koordinat_vertex[index_sama][1]-
array_vertexy[counter_vertex])<0.0001&&
                Math.abs(koordinat_vertex[index_sama][2]-
array_vertexz[counter_vertex])<0.0001){


               // koordinat ini sama dengan koordinat pada index
berapa di array kordinat_vertex
               // catat indexnya

                index = index_sama ;

                //System.out.println(index);

            }
         }
     //System.out.println();
        if (index == 0 ){

// koordinat yang masuk
//koordinat_vertex = v
// row = j
koordinat_vertex[row][0]=     array_vertexx[counter_vertex]      ;
//koordinat x
koordinat_vertex[row][1]=     array_vertexy[counter_vertex]      ;
//koordinat y
koordinat_vertex[row][2]=   array_vertexz[counter_vertex]   ;   //
koordinat z

            //  System.out.println(koordinat_vertex[row][0]+  "\t"+
koordinat_vertex[row][1]+ "\t"+koordinat_vertex[row][2]);
          // System.out.print(row);
           if(xminmax[0] > koordinat_vertex[row][0])
           {
                xminmax[0]= koordinat_vertex[row][0];

           }

           if(xminmax[1] < koordinat_vertex[row][0])
           {
                xminmax[1]= koordinat_vertex[row][0];

           }
           if(yminmax[0] > koordinat_vertex[row][1])
```

```
            {
                 yminmax[0]= koordinat_vertex[row][1];

            }

            if(yminmax[1] < koordinat_vertex[row][1])
            {
                 yminmax[1]= koordinat_vertex[row][1];

            }

            if(zminmax[0] > koordinat_vertex[row][2])
              {
                 zminmax[0]= koordinat_vertex[row][2];
              }

            if(zminmax[1] < koordinat_vertex[row][2])
              {
                 zminmax[1]= koordinat_vertex[row][2];
              }

             // index_vertex = arr
              // counter_index_vertex = count

              index_vertex[counter_index_vertex] = row;

            // System.out.println (counter_index_vertex) ;
            //System.out.println(index_vertex);

              }
          else {
            //index bukan sama dengan 0 (koordinat yang tidak masuk)
dicatat indexnya.,

            index_vertex[counter_index_vertex]= index ;
             row = row - 1 ;

             //row = j
          }


        row ++ ;
        counter_index_vertex ++ ;
         counter_vertex ++ ;

         }


      else if (a.equals("endsolid")){
      // row = j
      // jumlah_segitiga = i

      row = row - 1 ;
      jumlah_segitiga = jumlah_segitiga - 1 ;

      }
      else if (a.equals("endfacet")){
      // index_segitiga_vertex = T
```

```
         index_segitiga_vertex               [jumlah_segitiga][0]=
index_vertex[0];
         index_segitiga_vertex               [jumlah_segitiga][1]=
index_vertex[1];
         index_segitiga_vertex               [jumlah_segitiga][2]=
index_vertex[2];

        jumlah_segitiga = jumlah_segitiga + 1 ;
        counter_index_vertex = 0 ;
     }

}
   double delta_x [] = new double [1000];
   delta_x [1]= xminmax [0] - 0 ;
   double delta_y [] = new double [1000];
   delta_y [1]= yminmax [0] - 0 ;
   double delta_z [] = new double [1000];
   delta_z [1]= zminmax [0] - 0 ;


   //normalisasi
   xminmax [0]= xminmax [0]- delta_x [1] ;
   xminmax [1]= xminmax [1]- delta_x [1];
   yminmax [0]= yminmax [0]- delta_y [1] ;
   yminmax [1]= yminmax [1]- delta_y [1] ;
   zminmax [0]= zminmax [0]- delta_z [1] ;
   zminmax [1]= zminmax [1]- delta_z [1] ;


  // System.out.println(jumlah_segitiga);
     System.out.println ();
    // System.out.println ("index segitiga dan index vertex ");


   for (int a = 1 ;a <= jumlah_segitiga ; a ++){



       // jumlah_segitiga = i

    System.out.println (a + "\t" + index_segitiga_vertex [a][0] +
"\t" + index_segitiga_vertex[a][1]
       + "\t" + index_segitiga_vertex[a][2]);


   }

     System.out.println();
   //   System.out.println("koordinat vertex");
     for (int b = 1 ;  b <= row ; b ++){



      System.out.println (b + "\t" + koordinat_vertex [b][0] +
"\t" + koordinat_vertex [b][1]
       + "\t" + koordinat_vertex [b][2]);
```

```
      }
}

public static void close_file_stl (){
    x.close();


}

}
```

LAMPIRAN 5

```
/**

Project          : Slicing Algoritm
Compiler         : Netbean 6.9 Java Compiler
Package          : id.ac.ui.eng.hadiamil.machining.prototyping
Java Class       : brute_searching.java
Version          : 1.1.0
Date             : 27/12/2011
Developer        : Dede Sumantri
Company          : Universitas Indonesia
Comments         :

*/

package id.ac.ui.eng.hadiamil.machining.prototyping;

/**
 *
 * @author soemantri
 */
public class brute_searching {

   public static int counter_angkat = 0;

 static int [][] angkatan = new int [1000] [1000];

static int [] jumlah_angkatan = new int [1000] ;

public static int index1 ;
public static int index2 ;
public static int index_koordinat_vertex = 0 ;

public static double koordinatz_1 ;
public static double koordinatz_2 ;
static double layer_i = 0 ;


public static void brute_searching (){
    /**
     * asumsi_z [0] = zterendah
     * asumsi_z [1] = ztertinggi
     *
     * segitiga = j
     * indexv = index_koordinat_vertex
     * array T = index_segitiga_vertex
     * array v = index_koordinat_vertex
     */

    System.out.println ();


    double[] z_tertinggi_terendah = new double [2];
```

```
    z_tertinggi_terendah [0] = -1000 ; // nilai z rendah
    z_tertinggi_terendah [1] = 1000 ; // nilai z tertinggi


    for ( double i = get_coordinate_model.zminmax[0] +
(get_coordinate_model.tinggi_angkat/2); i <=
get_coordinate_model.zminmax[1] ; i = i +
get_coordinate_model.tinggi_angkat ){

       // layer_i = layer_i + i ;


       // System.out.println(zminmax[1]);
      // System.out.println (i);

    counter_angkat ++ ;
    jumlah_angkatan [counter_angkat] = 0 ;

    for (int segitiga = 1 ; segitiga <=
get_coordinate_model.jumlah_segitiga; segitiga++){

        //System.out.println(segitiga);

        for (int k = 0 ; k <= 2 ; k ++){

          //  System.out.println (k);

        index_koordinat_vertex = k ;
        index1 = get_coordinate_model.index_segitiga_vertex
[segitiga][index_koordinat_vertex];
        index_koordinat_vertex = k + 1 ;

            if( index_koordinat_vertex > 2){

                index_koordinat_vertex = index_koordinat_vertex -
2 ;

            }

        index2 = get_coordinate_model.index_segitiga_vertex
[segitiga][index_koordinat_vertex];
        koordinatz_1 = get_coordinate_model.koordinat_vertex
[index1][2];
        koordinatz_2 = get_coordinate_model.koordinat_vertex
[index2][2];
       // System.out.println (index1 + "\t" + index2 );
         //   System.out.println (koordinatz_1 + "\t" +
koordinatz_2 );



    if ((koordinatz_1 > i && i > koordinatz_2)||(koordinatz_1 < i
&& i < koordinatz_2 )|| koordinatz_1 == i || koordinatz_2 == i
                || (koordinatz_1 == i )&& (koordinatz_2 == i)) {

       // System.out.println (counter_angkat);
```

```
            jumlah_angkatan [counter_angkat] = jumlah_angkatan
[counter_angkat] + 1 ;

          //     System.out.println (counter_angkat + "\t" +
jumlah_angkatan [counter_angkat]);

            angkatan
[counter_angkat][jumlah_angkatan[counter_angkat]] = segitiga ;

       //                                          System.out.println
(angkatan[counter_angkat][jumlah_angkatan[counter_angkat]]);

           if (z_tertinggi_terendah [1] < i){

               z_tertinggi_terendah [1] = i ;

            }
            if (z_tertinggi_terendah [0] > i){

                z_tertinggi_terendah [1] = i ;

            }
          break ;

        }
        }

    }


    }


}

}
```

LAMPIRAN 6

/**

Project : Slicing Algoritm
Compiler : Netbean 6.9 Java Compiler
Package : id.ac.ui.eng.hadiamil.machining.prototyping
Java Class : slicing.java
Version : 1.1.0
Date : 27/12/2011
Developer : Dede Sumantri
Company : Universitas Indonesia
Comments :

```
*/
package rapid_prototyping;

/**
 *
 * @author soemantri
 */
public class slicing {
    static int p = 1 ;
    static int counter_angkat2 = 1;
static int counter_bagian = 0 ;
static int bagian = 0 ;
static int [] jumlah_bagian = new int [1000]  ;
static  double  tinggi_layer  =  get_coordinate_model.zminmax[0]  +
(get_coordinate_model.tinggi_angkat/2);
static int index_segitiga2 = 0 ;
static int counter_index_segitiga;
static int m ;

static int index_koordinat_vertex2 ;
static int index1_2 ;
static int index2_2 ;
static double koordinatz1_2 ;
static double koordinatz2_2 ;
static double koordinatz3_2 ;
static double titik_potong [][][] = new double [1000][110][3];


static  int s = 1;
static int index_pertama ;
static int index_kedua ;
static int index_v1 ;
static int index_v2 ;
static int index_ketiga ;
static int index3 ;
static int simpan1 ;
static int simpan2 ;



public static void segitiga_pertama (){
```

```
    while (p == 1){

function.masih_ada_bukan_nol(brute_searching.jumlah_angkatan[counter_angkat2],counter_angkat2);


/**
System.out.println(brute_searching.jumlah_angkatan[brute_searching.counter_angkat]);

System.out.println(function.ada_nol);
System.out.println (counter_angkat2);
*/


        if ( function.ada_nol <= 0){



  tinggi_layer            =            tinggi_layer            +
get_coordinate_model.tinggi_angkat ;
//System.out.println (tinggi_layer);
           s = 0 ;

           counter_angkat2 = counter_angkat2 + 1 ;

               if                (tinggi_layer                >
get_coordinate_model.zminmax[1]){

               break ;


               }

        }
/**
  System.out.println(tinggi_layer);
 segitiga pertama
 System.out.println (counter_angkat2);

System.out.println(brute_searching.jumlah_angkatan[counter_angkat2];
   */



    for (  counter_index_segitiga = 1 ; counter_index_segitiga <= brute_searching.jumlah_angkatan[counter_angkat2];counter_index_segitiga++){


       index_segitiga2          =          brute_searching.angkatan[counter_angkat2][counter_index_segitiga];


//System.out.println(brute_searching.angkatan[counter_angkat2][counter_index_segitiga]) ;
```

```
       if (index_segitiga2 > 0){

           counter_bagian = counter_bagian + 1 ;
          //  System.out.println(counter_bagian);
           jumlah_bagian [counter_bagian ] = 0 ;


           for ( m = 0 ; m<3; m++ ){

  index_koordinat_vertex2 = m ;


  index1_2      =      get_coordinate_model.index_segitiga_vertex
[index_segitiga2][index_koordinat_vertex2];

// System.out.println (index1_2);

               index_koordinat_vertex2 = m + 1 ;
                if (m >= 3){
            index_koordinat_vertex2  =  index_koordinat_vertex2  -
3 ;

           }
               //System.out.println (index_koordinat_vertex2);

            //   System.out.println(index_koordinat_vertex2);
         index2_2   =   get_coordinate_model.index_segitiga_vertex
[index_segitiga2][index_koordinat_vertex2];



koordinatz1_2        =       get_coordinate_model.koordinat_vertex
[index1_2 ][2];
koordinatz2_2        =       get_coordinate_model.koordinat_vertex
[index2_2 ][2];



int compare1 = Double.compare(koordinatz1_2, tinggi_layer);
int compare2 = Double.compare(koordinatz2_2, tinggi_layer);

if    ((koordinatz1_2    <    tinggi_layer    &&    tinggi_layer    <
koordinatz2_2)||
              (koordinatz1_2  >  tinggi_layer  &&  tinggi_layer  >
            koordinatz2_2)){


function.findz_intersection(index1_2,index2_2);
jumlah_bagian [counter_bagian] = jumlah_bagian [counter_bagian] +
1;

//System.out.println(jumlah_bagian[counter_bagian]);
//System.out.println (tinggi_layer);
// System.out.println (index1_2 +"\t"+ index2_2);
```

```
titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][0]   =
function.vertex_tengah[0];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]   =
function.vertex_tengah[1];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]   =
function. vertex_tengah[2];

/**
System.out.println(function.vertex_tengah[0]+"\t"+
function.vertex_tengah[1] +"\t"+ function.vertex_tengah[2]);

System.out.println         (counter_angkat2         +         "\t"
+counter_index_segitiga);
   */


 brute_searching.angkatan
[counter_angkat2][counter_index_segitiga] = 0 ;
            //
System.out.println(brute_searching.angkatan[counter_angkat2][count
er_index_segitiga]);
          break ;


         }

else         if         (Double.valueOf(koordinatz1_2)         ==
Double.valueOf(tinggi_layer)){

           index2_2 = -1 ;

 jumlah_bagian[counter_bagian]   =  jumlah_bagian[counter_bagian]+
1 ;



titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][0]   =
get_coordinate_model.koordinat_vertex[index1_2][0];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]   =
get_coordinate_model.koordinat_vertex [index1_2][1];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]   =
get_coordinate_model.koordinat_vertex [index1_2][2];

 brute_searching.angkatan
[counter_angkat2][counter_index_segitiga] = 0 ;
             break ;

          }

          else      if      (Double.valueOf(koordinatz2_2)      ==
Double.valueOf(tinggi_layer)){
            index1_2 = index2_2 ;
```

```
            index2_2 = -1 ;

            jumlah_bagian            [counter_bagian]            =
jumlah_bagian[counter_bagian] + 1 ;



titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][0]   =
get_coordinate_model.koordinat_vertex[index1_2][0];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]   =
get_coordinate_model.koordinat_vertex [index1_2][1];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]   =
get_coordinate_model.koordinat_vertex [index1_2][2];

              brute_searching.angkatan
[counter_angkat2][counter_index_segitiga] = 0 ;
              break ;


           }


            }


       if
(brute_searching.angkatan[counter_angkat2][counter_index_segitiga]
== 0){

       break ;


       }



           }

    }
      // function.dispay_angkatan();



        // segitiga selanjutnya


/**
 System.out.println
(brute_searching.jumlah_angkatan[counter_angkat2]    +    "\t"    +
counter_angkat2);

System.out.println               ("jumlah               angkatan="
+brute_searching.jumlah_angkatan[counter_angkat2]);
    while                   (                   s                   <=
brute_searching.jumlah_angkatan[counter_angkat2]){
```

```
 System.out.println(counter_angkat2 +"\t"+ s);
 System.out.println("nilai s ="+ s);
*/

 index_segitiga2 = brute_searching.angkatan[counter_angkat2][s] ;


    if (index_segitiga2 > 0){
        //masuk  kesini  ketika  s==2  caounter_angkat2  =  1  dan
index_segitiga == 2

       // System.out.println(index_segitiga2);

            for (int y = 0 ; y<3 ; y ++){
                index_pertama = y ;


                index_v1                                              =
get_coordinate_model.index_segitiga_vertex[index_segitiga2][index_
pertama];

                index_kedua = y +1 ;
              // System.out.println (index_v1);

                if (index_kedua >=3){

                    index_kedua = index_kedua - 3 ;

                }
                index_v2                                              =
get_coordinate_model.index_segitiga_vertex[index_segitiga2][index_
kedua];


//  System.out.println (index1_2 + "\t" + index2_2);

// System.out.println (index_v1 + "\t" + index_v2);

if  ((Integer.valueOf(index_v1)  ==  Integer.valueOf(index1_2)  &&
      Integer.valueOf(index_v2) == Integer.valueOf(index2_2 ))
                  ||(Integer.valueOf(index_v1)                        ==
      Integer.valueOf(index2_2)  &&  Integer.valueOf(index_v2)   ==
      Integer.valueOf(index1_2))){

                //  System.out.println  (index_pertama  +  "\t"  +
index_kedua);
             function.get_pasangan_ketiga(index_pertama,
index_kedua);

             index_ketiga = function.pasangan_ketiga ;

//System.out.println ("nilai index ke tiganya ="+index_ketiga);

index3        =         get_coordinate_model.index_segitiga_vertex
[index_segitiga2][index_ketiga] ;
// System.out.println (index3);
koordinatz1_2                                                         =
get_coordinate_model.koordinat_vertex[index1_2 ][2];
```

```
koordinatz2_2                                                    =
get_coordinate_model.koordinat_vertex[index2_2 ][2];
koordinatz3_2 = get_coordinate_model.koordinat_vertex[index3 ][2];

//  System.out.println (tinggi_layer);
// System.out.println (koordinatz1_2 + "\t"+ koordinatz2_2 +"\t" +
koordinatz3_2 + "\t" + tinggi_layer);

//cek 1 dan 3

       if  ((koordinatz1_2  <  tinggi_layer  &&  tinggi_layer  <
              koordinatz3_2  )||(koordinatz1_2  >  tinggi_layer  &&
              tinggi_layer > koordinatz3_2)){

              index2_2 = index1_2 ;
              index1_2 = index3 ;

           //  System.out.println(index1_2 + "\t" + index2_2 );

              function.findz_intersection(index1_2,index2_2);
              jumlah_bagian[counter_bagian]                      =
jumlah_bagian[counter_bagian] + 1 ;



titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][0]   =
function.vertex_tengah[0];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]   =
function.vertex_tengah[1];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]   =
function. vertex_tengah[2];

       /**
System.out.println(titik_potong[counter_bagian][jumlah_bagian[coun
ter_bagian]][0] +"\t" +

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]+"\t
"+titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]);
*//

              brute_searching.angkatan [counter_angkat2][s] = 0 ;
              s = 0 ;
              break ;

              }

              // cek 2 dan 3

    else  if  ((koordinatz2_2  <  tinggi_layer  &&  tinggi_layer  <
              koordinatz3_2)||  (koordinatz2_2  >  tinggi_layer  &&
              tinggi_layer > koordinatz3_2)){

              index1_2 = index2_2 ;
              index2_2 = index3 ;

 // System.out.println (index1_2 + "\t" + index2_2);
```

```
            function.findz_intersection(index1_2,index2_2);
            jumlah_bagian[counter_bagian]                          =
            jumlah_bagian[counter_bagian] + 1 ;


titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][0]   =
function.vertex_tengah[0];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]   =
function.vertex_tengah[1];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]   =
function. vertex_tengah[2];

              brute_searching.angkatan [counter_angkat2][s] = 0 ;
              s = 0 ;
              break ;

            }

            // cek 3

   else           if          (Double.valueOf(tinggi_layer)==
Double.valueOf(koordinatz3_2)){
               index1_2 = index3 ;
               index2_2 = -1 ;

jumlah_bagian[counter_bagian]  =  jumlah_bagian[counter_bagian]  +
1 ;

/**
 System.out.println(counter_bagian          +"\t"            +
jumlah_bagian[counter_bagian]);
  System.out.println (jumlah_bagian [counter_bagian]);
*/

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][0]   =
get_coordinate_model.koordinat_vertex[index1_2][0];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]   =
get_coordinate_model.koordinat_vertex [index1_2][1];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]   =
get_coordinate_model.koordinat_vertex [index1_2][2];

              brute_searching.angkatan [counter_angkat2][s] = 0 ;
              s = 0 ;
              break ;


            }

            else if (Integer.valueOf(index2_2) == -1 ){
                simpan1 = index1_2 ;
                simpan2 = index2_2 ;
```

```
                if          (Integer.valueOf(index1_2)          ==
Integer.valueOf(index_v1)){
                        index2_2 = index_v2 ;

                }
                else     if     (Integer.valueOf(index1_2)     ==
Integer.valueOf(index_v2) ){
                   index2_2 = index_v1 ;

                }
                else{
                   continue ;

                }

             }


             //ambil index ke 3

                function.get_pasangan_ketiga(index_pertama,
index_kedua);
      index_ketiga = function.pasangan_ketiga ;

       index3                                                    =
get_coordinate_model.index_segitiga_vertex[index_segitiga2][index_
kedua];

      koordinatz1_2                                              =
get_coordinate_model.koordinat_vertex[index1_2 ][2];
     koordinatz2_2                                               =
get_coordinate_model.koordinat_vertex[index2_2 ][2];
     koordinatz3_2                                               =
get_coordinate_model.koordinat_vertex[index3 ][2];


       //cari index ke 2
     if              (Double.valueOf(koordinatz2_2)             ==
Double.valueOf(tinggi_layer)){

        index1_2 = index2_2 ;
        index2_2 = -1 ;

        jumlah_bagian[counter_bagian]                            =
jumlah_bagian[counter_bagian] + 1 ;


titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][0]   =
get_coordinate_model.koordinat_vertex[index1_2][0];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]   =
get_coordinate_model.koordinat_vertex [index1_2][1];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]   =
get_coordinate_model.koordinat_vertex [index1_2][2];

             brute_searching.angkatan [counter_angkat2][s] = 0 ;
```

```
            s = 0 ;
            break ;

     }
     else        if        (Double.valueOf(koordinatz3_2)        ==
Double.valueOf(tinggi_layer) ){

           index1_2 = index3 ;
           index2_2 = -1 ;

          jumlah_bagian[counter_bagian]                          =
jumlah_bagian[counter_bagian] + 1 ;


titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][0]   =
get_coordinate_model.koordinat_vertex[index1_2][0];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]   =
get_coordinate_model.koordinat_vertex[index1_2][1];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]   =
get_coordinate_model.koordinat_vertex[index1_2][2];

            brute_searching.angkatan [counter_angkat2][s] = 0 ;
            s = 0 ;
            break ;


     }
     // cek garis 2 dan 3

     else if   ((koordinatz2_2 < tinggi_layer && tinggi_layer <
koordinatz3_2 )
         ||  (koordinatz2_2  >  tinggi_layer  &&  tinggi_layer  >
koordinatz3_2 )){

         index1_2 = index2_2 ;
         index2_2 = index3 ;
         function.findz_intersection(index1_2,index2_2);


         jumlah_bagian[counter_bagian]                           =
jumlah_bagian[counter_bagian] + 1 ;


titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][0]   =
function.vertex_tengah[0];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]   =
function.vertex_tengah[1];

titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]   =
function. vertex_tengah[2];

            brute_searching.angkatan [counter_angkat2][s] = 0 ;
            s = 0 ;
            break ;
```

```
          }
      else {

              index1_2 = simpan1 ;
              index2_2 = simpan2 ;

      }

               }
    }
              }

//System.out.println(brute_searching.angkatan[counter_angkat2 ][s]
);


    s = s + 1 ;



    }
      // System.out.println ();

//System.out.println    (counter_bagian    +"\t"+    jumlah_bagian
[counter_bagian]);


       //function.dispay_angkatan();
    }


  //
System.out.println(titik_potong[counter_bagian][jumlah_bagian[coun
ter_bagian]][0] +"\t" +
    //
titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][1]+"\t
"+titik_potong[counter_bagian][jumlah_bagian[counter_bagian]][2]);




    }

}
```

LAMPIRAN 7

```
/**

Project         : Slicing Algoritm
Compiler        : Netbean 6.9 Java Compiler
Package         : id.ac.ui.eng.hadiamil.machining.prototyping
Java Class      : path_element_generation.java
Version         : 1.1.0
Date            : 27/12/2011
Developer       : Dede Sumantri
Company         : Universitas Indonesia
Comments        :

*/


package id.ac.ui.eng.hadiamil.machining.prototyping;

public class Path_element_generation {
static int [] vjumlah_lintasan = new int [10000]  ;
static int bagian_lintasan = 0;
static double v_lintasan [][][] = new double [10000][110][3];
static double vertex_satu [] = new double [3];
static double vertex_dua [] = new double [3];

 static    double       z    =    get_coordinate_model.xminmax[0]-
get_coordinate_model.tinggi_angkat/2;
static int h = 1 ;
static int y = 1 ;
static int q ;
static int rr ;
static double hatchspace ;

 static double elemen_y [] = new double [1000];


    public static void elementy (){
/**
 System.out.println(slicing.titik_potong[1][2][0]             +"\t"
slicing.titik_potong[1][2][1]+"\t"+slicing.titik_potong[1][2][2]);
 System.out.println(slicing.titik_potong[1][3][0]      +"\t"      +
slicing.titik_potong[1][3][1]+"\t"+slicing.titik_potong[1][3][2]);
 */

        while ( h == 1){
          //  System.out.println (z);
            if (z > get_coordinate_model.zminmax[1]){
                break ;

            }

        z = z + get_coordinate_model.tinggi_angkat ;


hatchspace        =        get_coordinate_model.xminmax[0]        -
(FdmPrototyping.step_over/2);
```

```
        while (y==1 ){

        // hatchspace = x
         hatchspace  = hatchspace + FdmPrototyping.step_over ;

        if (hatchspace >=get_coordinate_model.xminmax[1]){

             break ;

        }

        bagian_lintasan = bagian_lintasan + 1 ;
        vjumlah_lintasan [bagian_lintasan] = 0 ;

        // j adalah k di matlab!

        for (int j = 1 ;j <= slicing.counter_bagian; j++){
       // System.out.println (z);
         if (Math.abs(slicing.titik_potong[j][1][2]- z)>0.000001){

             continue ;

         }

       for (int t = 1 ; t <= slicing.jumlah_bagian[j];t++){
           //System.out.println (t);
           int satu = t ;
           int dua =  t + 1 ;
          // System.out.println (dua);
           // System.out.println (slicing.jumlah_bagian[j]);

   if (dua >slicing.jumlah_bagian[j]){

              dua = dua - slicing.jumlah_bagian[j];

           }

       if                    (slicing.titik_potong[j][satu][0]==
Double.valueOf(hatchspace)){
       vjumlah_lintasan[bagian_lintasan]     =     vjumlah_lintasan
[bagian_lintasan] + 1 ;

v_lintasan
[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][0]=
slicing.titik_potong [j][satu][0];
 v_lintasan
[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][1]=
slicing.titik_potong [j][satu][1];
 v_lintasan
[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][2]=
slicing.titik_potong [j][satu][2];



       }
```

```
      else          if          (slicing.titik_potong[j][dua][0]==
Double.valueOf(hatchspace)){

 v_lintasan
[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][0]=
slicing.titik_potong [j][satu][0];
 v_lintasan
[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][1]=
slicing.titik_potong [j][satu][1];
 v_lintasan
[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][2]=
slicing.titik_potong [j][satu][2];
                        }

      else if ((slicing.titik_potong[j][satu][0] < hatchspace &&
hatchspace < slicing.titik_potong[j][dua][0])
                  ||      (slicing.titik_potong[j][satu][0]      >
hatchspace && hatchspace > slicing.titik_potong[j][dua][0])){



       vjumlah_lintasan                              [bagian_lintasan]=
vjumlah_lintasan[bagian_lintasan] + 1 ;
  //  System.out.println(j  + "\t"  + satu  + "\t"  + dua  + "\t"  +
hatchspace);
     vertex_satu [0] = slicing.titik_potong[j][satu][0];
     vertex_satu [1] = slicing.titik_potong[j][satu][1];
     vertex_satu [2] = slicing.titik_potong[j][satu][2];

     //System.out.println(slicing.titik_potong[j][satu][0]);
     vertex_dua [0] = slicing.titik_potong[j][dua][0];
     vertex_dua [1] = slicing.titik_potong[j][dua][1];
     vertex_dua [2] = slicing.titik_potong[j][dua][2];

     //System.out.println
(vertex_satu[0]+"\t"+vertex_satu[1]+"\t"+ vertex_satu[2]);

     function.proses_cari_titikx(hatchspace);

 v_lintasan
[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][0]=function.v
ert[0];
 v_lintasan
[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][1]=function.v
ert[1];
 v_lintasan
[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][2]=function.v
ert[2];


ystem.out.println(v_lintasan[bagian_lintasan][vjumlah_lintasan[bag
ian_lintasan]][1]);

//System.out.println(v_lintasan[bagian_lintasan][vjumlah_lintasan[
bagian_lintasan]][0] +"\t" +
  //
v_lintasan[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][1]+
"\t"+
```

```
  //
v_lintasan[bagian_lintasan][vjumlah_lintasan[bagian_lintasan]][2])
;

  //System.out.println  (v_lintasan  [2][1][0]  +"\t"  +  v_lintasan
[2][1][1] +"\t"+ v_lintasan [2][1][2]);
      }

      }

       }
           // q sama dengan i ;
   //   System.out.println(bagian_lintasan);
      //  System.out.println(vjumlah_lintasan[bagian_lintasan]);

 for (q = 1 ; q<= bagian_lintasan ; q++){
 // System.out.println(vjumlah_lintasan[q]);
    if (vjumlah_lintasan[q]== 0){
        continue ;

    }
// System.out.println(vjumlah_lintasan[q]);

 for ( rr = 1 ; rr <= vjumlah_lintasan[q];rr++){

     //System.out.println(r);
     elemen_y [rr] = v_lintasan [q][rr][1];
  //  System.out.println(v_lintasan [q][rr][1]);

 }


 function.insertionsort(elemen_y[rr], 1, vjumlah_lintasan[q], 1);


 for (int g = 1 ; g <= vjumlah_lintasan [q];g++){
 elemen_y[g] = function.result_doinsertion [g];

 v_lintasan [q][g][1] = elemen_y [g]  ;

 //System.out.println (v_lintasan [q][g][1]);

//   System.out.println(v_lintasan[q][g][0] +"\t" +
  //          v_lintasan[q][g][1]+"\t"+
    //        v_lintasan[q][g][2]);


 }



 }



        }

    }
```

LAMPIRAN 8

```
/**

Project          : Slicing Algoritm
Compiler         : Netbean 6.9 Java Compiler
Package          : id.ac.ui.eng.hadiamil.machining.prototyping
Java Class       : path_linking.java
Version          : 1.1.0
Date             : 27/12/2011
Developer        : Dede Sumantri
Company          : Universitas Indonesia
Comments         :

*/
package id.ac.ui.eng.hadiamil.machining.prototyping;

import id.ac.ui.eng.hadiamil.base.Vertex;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Vector;

/**
 *
 * @author soemantri
 */
public class path_linking {

static int catat_persambungan = 0;
static int bagian_saat_ini = 1;
static double currentz = 1;
static float currentz_vertex = 1 ;
static int pindah_awal = -1;
static int mulai;
static double vertex_ini[] = new double[3];
static int loncat;
static int akhir;
static int tambah;
static int layer = 1;
static int oddcounter  =1;
static double hubx[] = new double[2];
static double huby[] = new double[2];
static double hubz[] = new double[2];

static double x1[] = new double[2];
static double y1[] = new double[2];
static double z1[] = new double[2];

static String[] stringx = new String[3];
static String[] stringy = new String[3];
static String[] stringz = new String[3];
```

```
public static List<Vertex> points = new ArrayList<Vertex>();

public static List<Vertex> fdmPoints = new ArrayList<Vertex>();
public static List<Vertex> tmpPoints = new ArrayList<Vertex>();
private Object array;

public static void method() {

      BufferedWriter gcode_file = null;
      FileWriter fwriter = null;

      try {

      gcode_file        =        new        BufferedWriter(new
FileWriter("D:/gcode_hadiamill.txt"));

      gcode_file.write("RPcode.V.1.0.0 ");
      gcode_file.newLine();

for (int n = 1; n <= Path_element_generation.bagian_lintasan; n++)
{

      // System.out.println (n);

      if (Path_element_generation.vjumlah_lintasan[n] == 0) {

            continue;

      }

if (currentz != Path_element_generation.v_lintasan[n][1][2]) {


currentz = Path_element_generation.v_lintasan[n][1][2];
      // System.out.println(currentz);
            pindah_awal = -1;

}
else {

      pindah_awal = -pindah_awal;
}



if (pindah_awal == 1) {
      mulai = Path_element_generation.vjumlah_lintasan[n];
      // System.out.println (mulai);
      loncat = -2;
      akhir = 2;
      tambah = -1;
      }
else {

      mulai = 1;
      loncat = 2;
      akhir = Path_element_generation.vjumlah_lintasan[n] - 1;
```

```
      tambah = 1;

      }


      //System.out.println (n);
if (n > 1) {
      //System.out.println ("kedua kesini");
vertex_ini[0] = Path_element_generation.v_lintasan[n][mulai][0];
vertex_ini[1] = Path_element_generation.v_lintasan[n][mulai][1];
vertex_ini[2] = Path_element_generation.v_lintasan[n][mulai][2];

//System.out.println    (vertex_ini[0]+"\t"+    vertex_ini[1]+"\t"+
vertex_ini[2]);

function.anggota_bagian(bagian_saat_ini)
//         System.out.println         ("result         ="         +
function.result_anggota_bagian);

if (function.result_anggota_bagian == 1) {

      hubx[1] = vertex_ini[0];
      huby[1] = vertex_ini[1];
      hubz[1] = vertex_ini[2];
// System.out.println(hubx[1] + "\t" + huby[1] + "\t" + hubz[1]);


stringx[2] = Double.toString(hubx[1]);
stringy[2] = Double.toString(huby[1]);
stringz[2] = Double.toString(hubz[1]);

/**
gcode_file.write("1" + "\t" + stringx[2] + "\t"+ stringy[2] + "\t"
+ stringz[2]);
gcode_file.newLine();
      */

Vertex v = new Vertex();
v.setX(Float.parseFloat(stringx[2]));
v.setY(Float.parseFloat(stringy[2]));
v.setZ(Float.parseFloat(stringz[2]));
points.add(v);


catat_persambungan = 1;
      }
else {
//System.out.println ("cari anggota");
function.cari_anggota_bagian(1);
bagian_saat_ini = function.result_cari_anggota_bagian;
// System.out.println
// (function.result_cari_anggota_bagian);
//System.out.println (bagian_saat_ini);
catat_persambungan = 0;

if (Math.abs(vertex_ini[2] -  Path_element_generation.v_lintasan[n
- 1][1][2]) > 0.0001) {
```

```
      layer = layer + 1;
      // System.out.println(layer);
      }
}


}


// System.out.println (mulai + "\t" +akhir +"\t"+ loncat);
for (int hh = mulai; hh == akhir; hh = loncat) {

      // System.out.println (mulai + "\t" +akhir +"\t"+ loncat);


x1[0] = Path_element_generation.v_lintasan[n][hh][0];
y1[0] = Path_element_generation.v_lintasan[n][hh][1];
z1[0] = Path_element_generation.v_lintasan[n][hh][2];

if (catat_persambungan == 0) {
      stringx[0] = Double.toString(x1[0]);
      stringy[0] = Double.toString(y1[0]);
      stringz[0] = Double.toString(z1[0]);


/**
gcode_file.write("0" + "\t" + stringx[0] + "\t"+ stringy[0] + "\t"
+ stringz[0]);
gcode_file.newLine();
*/

Vertex v = new Vertex();
v.setX(Float.parseFloat(stringx[0]));
v.setY(Float.parseFloat(stringy[0]));
v.setZ(Float.parseFloat(stringz[0]));
points.add(v);

}

x1[1] = Path_element_generation.v_lintasan[n][hh + tambah][0];
y1[1] = Path_element_generation.v_lintasan[n][hh + tambah][1];
z1[1] = Path_element_generation.v_lintasan[n][hh + tambah][2];

stringx[1] = Double.toString(x1[1]);
stringy[1] = Double.toString(y1[1]);
stringz[1] = Double.toString(z1[1]);

/**

gcode_file.write("1" + "\t" + stringx[1] + "\t"+ stringy[1] + "\t"
+ stringz[1]);
gcode_file.newLine();
*/

Vertex v = new Vertex();
v.setX(Float.parseFloat(stringx[1]));
v.setY(Float.parseFloat(stringy[1]));
```

```
v.setZ(Float.parseFloat(stringz[1]));
points.add(v);


catat_persambungan = 0;
      }

hubx[0] = 0;
hubx[1] = 0;
huby[0] = 0;
huby[1] = 0;
hubz[0] = 0;
hubz[1] = 0;

if (n < Path_element_generation.bagian_lintasan) {

      hubx[0] = x1[1];
      huby[0] = y1[1];
      hubz[0] = z1[1];

      }

}


      for (int i = 0; i < points.size(); i++) {

            Vertex vertex1 = points.get(i);
            //    Vertex vertex2 = points.get(i++);
            //System.out.println(z_vertex1 + "/t" + z_vertex2);



if (currentz_vertex != vertex1.getZ() ){

      for (int j = tmpPoints.size() - 1; j >= 0; j--) {
            fdmPoints.add(tmpPoints.get(j));



            }


      tmpPoints.clear();


if (oddcounter == 1 ){
      fdmPoints.add(vertex1);

      currentz_vertex = vertex1.getZ() ;

      oddcounter = 0 ;

}

else {
```

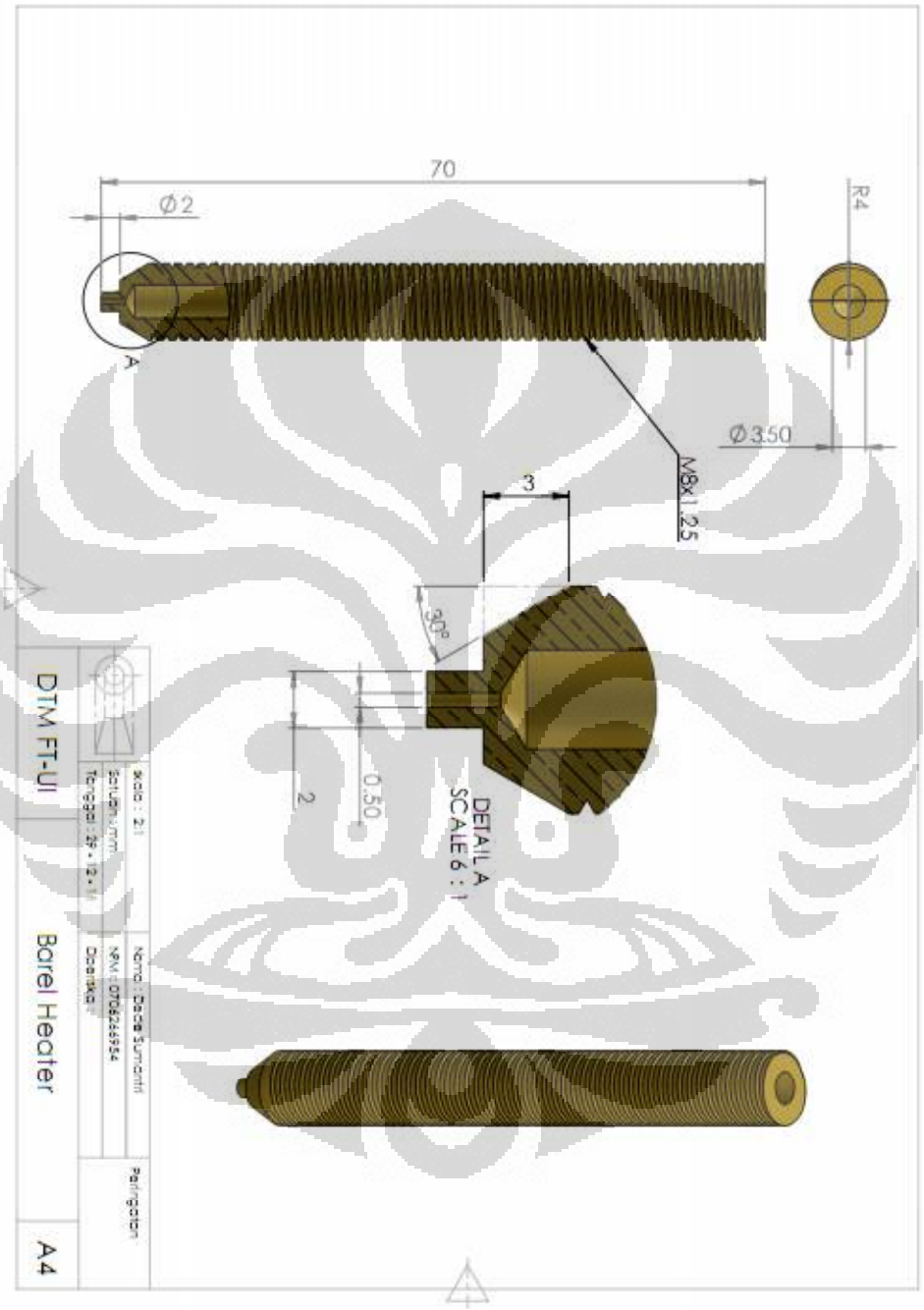
```
tmpPoints.add(vertex1);

currentz_vertex = vertex1.getZ();
oddcounter = 1 ;

}


      }
      else {


if (oddcounter == 0){

//ada kalanya i dan i+1 beda, masukkin 1 + 1 ke tmppoints
//beda z dengan kondisi oddcounter == 1
      fdmPoints.add(vertex1);

      oddcounter = 0 ;


      }

      else {

      tmpPoints.add(vertex1);
      // looping backward


      oddcounter = 1 ;

      }

      }

                  }

      for (int i = 0; i < fdmPoints.size(); i++) {

            Vertex point = fdmPoints.get(i);

gcode_file.write("1" + "\t" + point.getX() + "\t"+ point.getY() +
"\t" + point.getZ());

gcode_file.newLine();

}
                  // write to file
gcode_file.close();

} catch (IOException ioexecption) {

      System.out.println("file cacat");

}

}
}
```

LAMPIRAN 9

Data defleksi

NO	SUHU HEATBED c	SUHU NOZZLE C	dT C	JUMLAH LAYER	DEFLEKSI mm
1	200	250	50	5	5.053
2	200	250	50	10	1.942
3	200	250	50	15	0.826
4	200	250	50	20	0.606
5	200	255	55	5	2.49
6	200	255	55	10	2.499
7	200	255	55	15	0.94
8	200	255	55	20	0.397
9	200	260	60	5	3.331
10	200	260	60	10	1.849
11	200	260	60	15	0.59
12	200	260	60	20	0.378
13	205	250	45	5	1.216
14	205	250	45	10	1.211
15	205	250	45	15	0.765
16	205	250	45	20	0.759
17	205	255	50	5	0.53
18	205	255	50	10	0.532
19	205	255	50	15	0.352
20	205	255	50	20	0.6
21	205	260	55	5	0.337
22	205	260	55	10	0.296
23	205	260	55	15	0.707
24	205	260	55	20	0.541
25	210	250	40	5	1.305
26	210	250	40	10	0.89
27	210	250	40	15	0.902
28	210	250	40	20	1.092
29	210	255	45	5	2.23
30	210	255	45	10	0.904
31	210	255	45	15	1.121
32	210	255	45	20	0.822
33	210	260	50	5	2.833
34	210	260	50	10	0.747
35	210	260	50	15	1.081
36	210	260	50	20	0.933

LAMPIRAN 10


70
Ø2
R4
A
Ø3.50
M8x1.25
3
30°
2
0.50
DETAIL A
SCALE 6 : 1
DTM FT-UI
Barel Heater
A4