Open-Source Conversion of

Stratasys FDM Vantage X

Rapid-Prototyping Machine

A thesis presented in partial fulfilment of the requirements for the degree of

Master of Engineering

in

Mechatronics

at Massey University, Albany,

New Zealand

George Graham Hare

2014

Abstract

3D Printing or Additive Manufacturing is quickly redefining how companies design and develop products (Maxey, 2014). Increasingly, with advances in technology and materials, end-use parts are now being manufactured.

Wellington Drive Technologies Limited (WDTL), a world leading supplier of energy saving, electronically commutated motors (ECM) are based in Albany, Auckland. They use fused deposition modelling (FDM) to prototype new products and concepts. WDTL suffered a failure of a Stratasys FDM Vantage X machine used to 3D print prototype parts. The cost of rectifying the problem put the machine beyond economic repair. The machine was therefore gifted to Massey University, Albany.

Mechanically the machine was in good working order, it made sense therefore to attempt to resurrect the machine for research purposes. However, due to the cost of OEM repairs, and the associated research limitations, it was decided that where possible, open-source solutions should be sought.

The purpose of this dissertation is to prove the viability of replacing closed-source proprietary hardware and software solutions with open-source. The electronics and firmware were designed around the MakerBot Thing-O-Matic while ReplicatorG was used for the front end, all of which are open-source.

Ironically on the 19th June 2013, almost a year after starting this project, Stratasys bought MakerBot, taking a big stake in the consumer based 3D-Printer market. Subsequent releases of Makerware (MakerBot's successor to ReplicatorG) have been made closed-source.

Acknowledgements

Professor Olaf Diegel and Associate Professor Johan Potgieter of the School of Engineering and Advanced Technology (SEAT), Massey University, Albany Campus without whom this project would never have happened.

Garry Armstrong and Ashley Reid for their support during my studies. Ryan Brookes and Brian Young of the Defence Technology Agency (DTA) for allowing me to dedicate my time and efforts to further my knowledge of additive manufacturing in general for the benefit of the New Zealand Defence Force.

My wife, Alexandra for her constant support and re-assurance and not least, my sons Dominic and Thomas, for their understanding and patience over the past two years.

Also to the too numerous to mention friends and colleagues who have come through my lab over the past two years to see what it is I'm going on about at the lunch table.

Table of Contents

List of Figures

List of Tables

1 Introduction

3D Printing or Additive Manufacturing is quickly redefining how companies design and develop products (Maxey, 2014). In recent years, with advances in materials and process technologies, there has also been a push towards manufacture of 'end-use' parts. For example, CFM International (a partnership between GE Aviation and Snecma) has begun manufacturing the nineteen individual fuel nozzles used in each Leading Edge Aviation Propulsion (LEAP) turbofan engine, using Direct Metal Laser Sintering (DMLS) technology (LaMonica, 2013).

Wellington Drive Technologies Limited (WDTL) based in Albany, on Auckland's North Shore is a world leading supplier of energy saving, electronically commutated motors (ECM's). They use fused deposition modelling (FDM) to manufacture prototype products and concepts. The Stratasys FDM Vantage X machine used by WDTL to 3D-print prototype parts suffered a firmware failure. The projected cost of rectifying the problem through normal servicing channels put the machine beyond economic repair and as a result, the machine was gifted to Massey University's School of Engineering and Advanced Technology (SEAT) for research purposes. Table 1 shows the OEM specifications / capabilities of the Stratasys FDM Vantage X.

Parameter	Value
Build Envelope	14 x 10 x 10 inches (355 x 254 x 254 mm)
Build Materials	ABS, ABSi and PC-ABS or PC and PC-ISO
Layer Thickness	ABS, ABSi and PC-ABS: .010 inch (.254 mm) .007 inch (.178 mm) .005 inch (.127 mm) PC and PC-ISO: .010 inch (.254 mm) .007 inch (.178 mm)
Material Canister Capacity	One autoload canister with 92 cubic inches (1510 cubic cm) modeling material. One autoload canister with 92 cubic inches (1510 cubic cm) support material.
Build Software	Insight software imports STL files, automatically slices the file and generates the FDM Vantage extrusion paths and necessary support structures.
Accuracy	Models are produced within an accuracy of +/- .005 inch (+/- .127 mm) up to 5 inches (127 mm). Accuracy on models greater than 5 inches (127mm) is plus +/- .0015 inch per inch (+/- .0015 mm per mm).

Table 1: Stratasys FDM Vantage X Specifications (Stratasys, 2006)

1.1 OEM Machine Limitations

The donated Stratasys FDM Vantage X machine was mechanically in good working order, it therefore made sense to attempt to repair the machine. However, while ignoring the cost of OEM repairs, a number of limitations associated with using the machine for research purposes in its' original state would remain. Some of these limitation are discussed further below.

1.1.1 Materials

The Stratasys FDM Vantage X was designed to use material canisters that are chipped to identify the material and also to track how much filament had been used. Once used, the canisters cannot be re-filled without significant effort. To successfully re-use a canister, both the canister and the machine would have to be manipulated as follows:

Step 1 - Prior to using the canister, the eeprom on the canister (used to store the serial number and usage data) must be copied to a backup file.

Step 2 - The hard disk drive in the industiral computer inside the machine would also need to be 'ghosted' to a backup drive prior to using the canister. The canister can then be inserted in to the machine and used as normal.

Step 3 - Once empty, the canister would need to be remove, re-filled as required and the eeprom image replaced with the pre-used copy taken in step 1, effectively returning the canister to its full, unused state.

Step 4 - However, the machine also makes a log of the serial number of each canister inserted into it, to prevent the same canister being re-used. The 'ghosted' backup image, from step 2, would have to be copied back to the machines hard disk drive, returning the machine to its' earlier state. This would allow the canister to be used again, as no log of the serial number would exist on the hard disk drive. (Szczys, 2012)

The above process would only solve one limitation associated with use of chipped canisters. Stratasys also used this chip to control what materials the machine would be capable of running. By reading the data from the chip, and checking against the firmware licenses loaded on the printers' hard disk drive, the machine would determine if the canister of material was licensed for use.

From a Stratasys perspective, this served two functions; firstly, it allowed them

to charge on a 'per material' basis (the more materials you wished to use, the more you paid in licencing fees) and secondly, allowed them to keep a level of quality control. By limiting materials used to those specified by Stratasys, they could control the process parameters within the firmware and with a reasonable level of confidence, know that the resulting output would be consistent.

At the conclusion of this project, the repaired machine should be free from limitations with respect to material selection and licencing.

1.1.2 Process Parameter Control

By using pre-set parameters driven by Stratasys and identified by the canister chip, firmware licensing and the Insight software used to run the machine, the only process parameters that can be altered at a user level are the layer thickness and in-fill density.

To enable the use of alternative materials, Massey University would require the ability to control a wide range of process parameters including (but not limited to) extrusion speed, extrusion temperature, layer thickness, back-off time, in-fill density etc.

Another important process parameter controlled by the Insight software is the slicing algorithm. Conventional 3D-Printers can only print 2-dimensional layers. The Z height for each individual layer does not change. As a result, curved surfaces are made up of stepped layers which are not only aesthetically poor but also, where those surfaces are thin, they are physically weak due to the potential for delamination. Massey University would like the ability to generate non-linear layers i.e. contiguous curved surfaces rather than stepped surfaces. To achieve this, the machine would have to be able to process 3-dimensional layer information.

In-fill density can be changed but generally only on a whole part basis. Massey University would like the ability to investigate variable density in-fill, i.e. within a

part, the ability to specify where a high density in-fill should be used for strength and a lower density in-fill for weight reduction and/or material optimisation.

At the conclusion of this project, the repaired machine and software for controlling it should be free from limitations with respect to the process parameters required for Massey University research purposes.

The purpose of this research has been to investigate the issues related to using proprietary commercial 3D printing technology for research purposes with the aim of returning the Stratasys FDM Vantage X (gifted to Massey University) to working order and free from the discussed limitations. A literature review was conducted, looking into 3D printing in general, with a more in-depth investigation into fused deposition modelling in particular and the adoption of open-source hardware and software.

This dissertation comprises seven chapters: Chapter 1, Introduction, discusses the origins of the Stratasys FDM Vantage X at Massey University, its' research limitations and purpose of this research; Chapter 2, Literature Review, presents a background to 3D-printing, discusses FDM in detail, the open-source community and its' application to 3D-printing; Chapter 3, Mechanical Design, introduces discussion of the hardware modifications and re-design; Chapter 4, Electrical Design, details the new elctronics developed for the machine; Chapter 5, Firmware Design, details the baseline firmware and the modifications required to ensure compatibility with the new hardware; Chapter 6, Software Design, discusses the baseline software build and the modifications required to add the features required to work with the new hardware and firmware; Chapter 7, Calibration, discusses the testing conducted to setup the 3 axes, the build envelope and also extruder PID settings; Chapter 8 discusses test printing and results; Chapter 9, Conclusion, discusses the outcomes and areas for further work.

2 Literature Review

2.1 Definition

Additive Manufacturing (AM) was defined by the American Society for Testing and Materials International (ASTM) in 2012, in standard F2792 -12a (developed by subcommittee F42.91) as:

The process of joining materials to make objects from 3D model data, usually layer upon layer (ASTM, 2012).

2.2 Additive Manufacturing Technologies

Additive Manufacturing has gone through an evolution. Initially 3D-Printing or Rapid Prototyping, later Rapid Manufacturing and now Additive Manufacturing. AM is not a single technology but rather a suite of technologies that use a common approach to building parts without the need for expensive tooling or waste. Many have touted AM as the next industrial revolution (Markillie, 2012). However the fact remains that AM technologies are not well designed for scaling and will therefore never replace conventional mass manufacturing techniques. Where AM does fit well is in markets that require short-run or bespoke parts or customisation and parts that, due to geometry constraints, cannot be manufactured using conventional methods. This is where AM gets 'complexity for free', conventionally as parts grow in complexity, this is usually reflected in the price of the part, due to the number of machining processes required etc. AM has no such constraint and in fact, as the complexity of the part increases, the amount of material is reduced and therefore the material cost of the part comes down. However, time in the machine is the biggest driver for cost, so although increased complexity reduces material cost, the length of time spent in the machine is largely unchanged and therefore a simple geometry part is likely to be similar in cost to a complex part. It should be

considered that if a part is simple, it should not be manufactured using AM, but rather conventional techniques would be better suited. AM techniques should be seen as complementary to existing manufacturing methods rather than a replacement for them.

ASTM F2792-12a (ASTM, 2012) provides a standardised terminology for AM technologies and defines seven categories as shown in Table 2 along with a brief description of the implementation of the technology and examples of machines using the technology.MakerBot would be used as the baseline for this build.

Category	Technology Implemetation	Machine Example
Vat Photo-polymerisation	Stereo Lithography (SLA) – uses an ultra violet (UV) laser to trace the layer geometry on to the surface of a vat of resin, selectively hardening the resin. The part then drops down on a build plate into the vat, by the selected layer thickness. A fresh layer of resin is drawn over the top of the part and the process continues. This process requires a significant volume of resin to be kept in the vat for large parts.	Stratasys Viper Si2 (http://www.stratasys.com)
	Digital Light Processing (DLP) – is a relative new-comer, instead of using a laser to trace the geometry, a complete image of the layer is projected onto the resin surface from beneath using the same technology as a digital overhead projector. This method is usually much faster than SLA and also requires far less resin to be held in the vat as the part is lifted out of the vat as it is built.	Formlabs Form1+ (http://formlabs.com)
Material Jetting	This method is primarily aimed at the investment casting market. Parts are jetted using similar technology to inkjet printing but using wax or resin to built very detailed parts, which can be cast. This is currently the only technology capable of producing parts with variable material properties. For example, by mixing resins together in different ratios within a part. Parts can be printed with hard ABS-like plastic in some places and blend or step to soft rubber like material.	Stratasys CrownWorx (http://www.stratasys.com) Objet500 Connex (http://www.stratasys.com) Objet was acquired by Stratasys in December 2012

Category	Technology Implemetation	Machine Example
Binder Jetting	This method is similar to material jetting in that a resin is deposited using similar technology to inkjet printing. However, the resin is jetted into a powder bed (usually a plaster of paris type material), fusing the powder together. This technology is capable of printing in full colour by mixing resins in exactly the same way as colour inkjet printing. The parts made tend to be very fragile and as a result are only usually used for form and fit prototyping or display models.	ZCorp ZPrinter 650 (http://www.zcorp.com) Zcorp was acquired by 3D Systems in January 2012
	Alternative materials include sand for producing casting moulds and cores. Metal powders can also be used, after post-processing to sinter the metal powder and remove the resin, a usable metal part is produced, although not usually fully dense.	ExOne S-Max Furan (http://www.exone.com) ExOne M-Print (http://www.exone.com)
Material Extrusion	Fused Deposition Modeling (FDM), also known as Fused Filament Fabrication (FFF) is a process using thermoplastic, usually in the form of a filament, heated until molten and then extruded through a fine nozzle.	Stratasys FDM Fortus (http://www.stratasys.com)
Powder Bed Fusion	Selective Laser Sintering (SLS) and Selective Laser Melting (SLM) are essentially the same process. A laser is used to trace the layer geometry in the same way as SLA but on a powder bed. Materials range from plasitics through to ceramics and even metals.	SLS 3D Systems sPro 60 HD (http://www.3dsystems.com)

Category	Technology Implemetation	Machine Example
	The processing of metals requires an inert atmosphere within the build envelope to prevent formation of metal oxides and impurities. The only real difference between SLS and SLM is amount of enegy imparted to the powder by the laser beam. SLS is low power (typically <200W) as a result the particles are only sintered together resulting in porous parts. SLM is much higher power (typically 200-1000W) and imparts sufficient energy to the powder to melt the particles and produce fully dense parts.	SLM Renishaw AM250 (http://www.renishaw.com)
	Electron Beam Melting (EBM) is the latest entrant in the powder bed fusion market, claiming better material properties than SLS/SLM due to being processed at high temperature and in a vacuum. EBM is also generally faster as there are no moving parts. I contrast to SLS/SLM, where galvanometers are used to direct the laser beam (upto 2M/s), EBM uses magnetic fields to direct the electron beam. As a result, the beam can be driven round at much higher speed (upto 8000M/s). This gives EBM the capability of maintaining multiple melt pools.	Arcam Q10 (http:\\www.arcam.com)
Sheet Lamination	This process usually uses sheets of paper but can also be plastic or thin metal sheets. Each sheet is cut out to the precise shape of the layer and then laminated to the previous one. There is a significant amount of waste from this process. It is	Mcor IRIS (http://www.mcortechnologies.com)

Category	Technology Implemetation	Machine Example
	generally used for topographical terrain models. I don't agree with this category being in an additive manufacturing standard given that each layer is treated subtractively prior to lamination. This would be equivalent to milling a block of material.	
Directed Energy Deposition	Directed energy deposition is similar to SLM in that a laser beam is used to melt powder or filament. However the process melts the material as it is deposited onto the part, there is no powder bed. Analagous to metal extrusion. The process is generally used with a 5/6 axis robot arm rather than a cartesian machine. The machines can be mobile and are used for doing repair/re-work on very large tooling which would be too expensive to remove and transport for repair. This process is done on 'cold' parts and therefore usually requires post-process heat treatment and machining for surface finish.	Optomec LENS (Laser Engineered Net Shaping) (http://www.optomec.com)

Table 2: ASTM Additive Manufacturing Technology Catagories

For the purposes of this research, the Material Extrusion category and Fused Deposition Modeling were the focus.

2.3 Fused Deposition Modeling (FDM)

The FDM process was invented by S. Scott Crump, with the initial patent filed on 30 October 1989 and approved on 9 June 1992 (Crump, 1992). On filing the patent, Crump co-founded Stratasys with his wife Lisa Crump in Eden Prairie, Minnesota (Stratasys).

The patent described a method whereby, an object (40) could be created as a digital computer aided design (CAD) model (36). The model would then processed by 'layering' software (42), the output of which was then fed into the modeler controller (44). The modeler controller (44) was responsible for converting the layers into a physical object (40a) by controlling motors on the X (24), Y (20) and Z (32) axes of a machine as well as controlling the temperature and feed rate of an extruder head (2). The modeler controller (44) would maintain the thermoplastic in the extruder melt chamber at a temperature just above the melting point, such that it would solidify almost instantly as it was extruded and cooled.

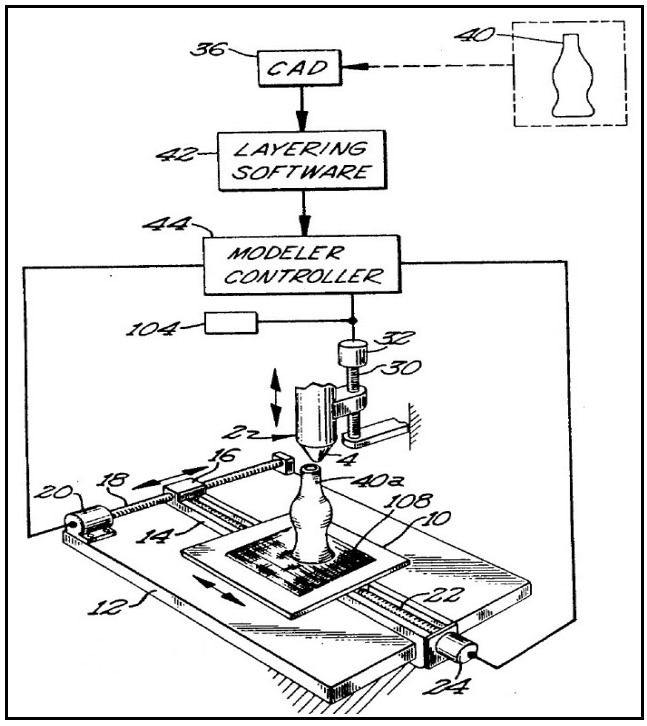


Figure 1: FDM Machine General Design (Crump, 1992)

Since the initial patent in 1989 there has only been one other patent, filed on 29 May 1997 (Crump, 1999), that has had a significant impact on the design of FDM machines. This patent related to the temperature control of the build volume during and after the build process to prevent 'curling' and delamination caused by thermal stresses within the part.

Several different designs for the extruder head were discussed in the patent (Crump, 1992). The most popular design to date has been the implementation of a flexible filament driven into a melt chamber using a motor mounted just above the melt chamber.

Early designs used DC motors to drive the filament into the melt chamber. The Stratasys FDM Vantage X that is the subject of this research also uses DC motors in the extruder head. DC motors have significant inertia and as a result, even after stopping the motor, filament would continue to be fed into the melt chamber, creating excess extrusion. Recent designs have largely solved this problem with the use of stepping motors, which allow for far more accurate control of the extruder.

2.4 The Open-Source Model

The Open-Source (OS) model is not a new concept, the Motor Vehicle Manufacturers Association in the United States (US), came to an agreement on sharing of intelectual property (IP) and patents back in 1911. By the time the US entered the second world war, 607 patents (92 belonging to the Ford Motor Company and 515 others held by other US manufacturers) were being openly shared between US motor manufacturers without exchange of money or lawsuits (Flink, 1977).

Open standards were adopted in the 1950's & 60's when the Advanced Research Projects Agency Network (ARPANET) used Request For Comments (RFC's) to develop telecommunication protocols, leading to the development of the Internet in 1969 (Crocker, 2009).

Bulletin Boards (BBS) were used as the early means for developers to share and modify source code and Linus Torvald was one of the first to adopt this process when he began devolpment of the Linux operating system, using comp.os.linux on USENET (Open Source, 2014).

Open Source was not used until the Free Software Movement adopted the term following a strategy meeting in January 1998, in Palo Alto, California (Tiemann, Michael, 2006).

As a development model, Open Source aims to promote universal access, via free license, to a product's design or blueprint, and universal redistribution of that design or blueprint, including subsequent improvements to it by anyone.

There have been numerous licences which have been used to enable free-access, starting with the Netscape Public License and later the Mozilla Public License (Netscape Public License, 1998). However, these licenses were asymetric with respect to rights. Netscape/Mozilla had the right to modify, distribute and even close the source on any contributions made, without reciprocal agreements for the contributors, it was therefore considered one to be avoided.

The Open Source Initiative published a list of almost 80 OS licenses (Open Source Initiative) approved by their License Review Process. However with so many different licenses, it is difficult to know which ones can or can't be used in conjunction with each other!

Although simple in it's concept, the OS model is difficult to implement as is evident by the number of licenses available. There are numerous examples where ideas that were initially developed in an OS model have subsequently been closed, as private individuals or enterprises modified/updated the code for their own commercial benefit. One of the most recent and contraversial was the closing of Makerbot Industries' Replicator design, the bulk of which had been based on OS hardware and software from the RepRap and later, Makerbot communities. Both hardware and software were developed over many years,

with input from countless owner/developers, improving the design. These modifications were incorporated into the Replicator and it successors without maintaining open access to the developments.

2.5 Development of Open-Source FDM Machines

Development of OS 3D Printers began in 2005 at Bath University. The project was started by Dr Adrian Bowyer, a senior lecturer in mechanical engineering. The goal of the project was to develop a machine capable of 'replicating' as much of itself as possible, with the aim of spreading the technology as far and wide, as cheaply as possible. Hence the name Replicating Rapid Prototyper, shortened to RepRap. The first RepRap was release in March 2007 and named 'Darwin' after Charles Darwin, 'the father' of evolutionary theory.

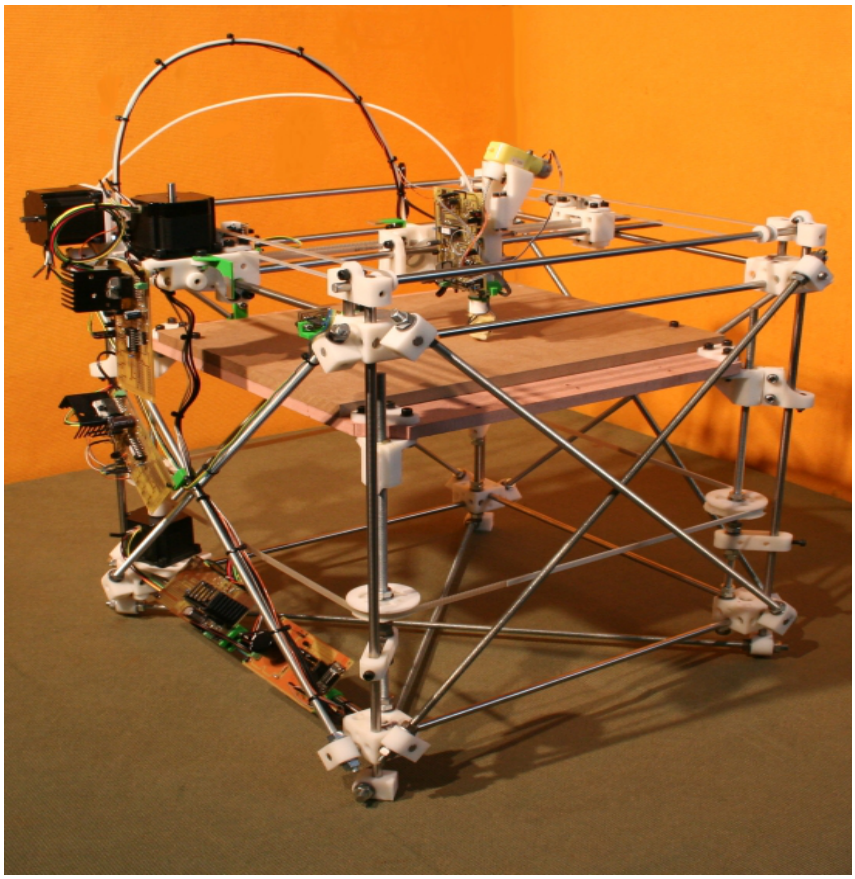


Figure 2: RepRap Darwin
(http://en.wikipedia.org/wiki/File:Reprap_Darwin.jpg)

Development continued and in the October 2009, version 2.0 the 'Mendel' was released. The biggest change was a move from a cube design, to a triangular prism. Generation 3, released in August 2010, was named 'Huxley', it was essentially a miniaturised version of Mendel (Gilloz, 2012).

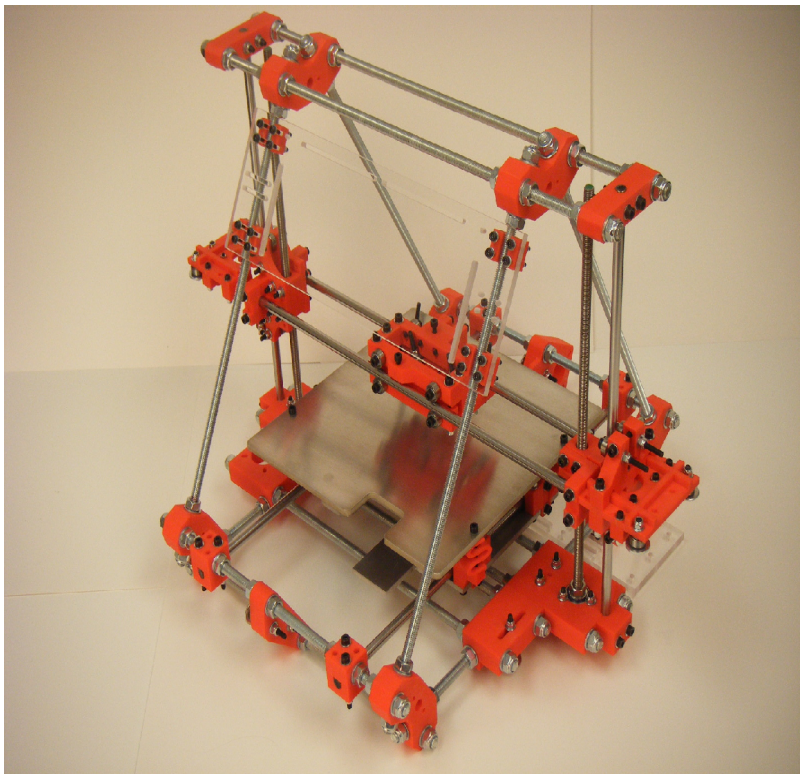


Figure 3: RepRap Mendel (http://www.cnccookbook.com/img/OthersProjects/3DPrinting/reprap.jpg)

The RepRap project used the term 'Fused Filament Fabrication' in order to avoid copyright issues with the term 'Fused Deposition Modeling' used by Stratasys however, the principal would have been in breach of the Stratasys patent (Crump, 1992) but this was never challenged, possibly due to the fact that a patent is only binding for 20 years and would have therefore been due to expire in 2009.

As of the beginning of this project in 2012, there were in excess of 200 designs either directly or indirectly related to Darwin (Gilloz, 2012).

Most RepRap designs are open-frame designs. One of the draw-backs of these designs is that they can be susceptible to drafts, which cause un-even cooling of the part. The result can be delamiation of the part where the thermal stress, from the contraction of the plastic as it cools, exceeds the shear stress of the inter-layer bond. Some of the later designs based on the RepRap, such as the MakerBot family (Cupcake CNC, Thing-o-Matic and the Replicators) or the Ultimaker use a 'Boxbot' approach, enclosing the print volume to retain heat and exclude drafts.

The Stratasys machine that I will be modifying extends this 'Boxbot' approach by having a fully heated build envelope rather than just the build table as many of the open-source bots do.

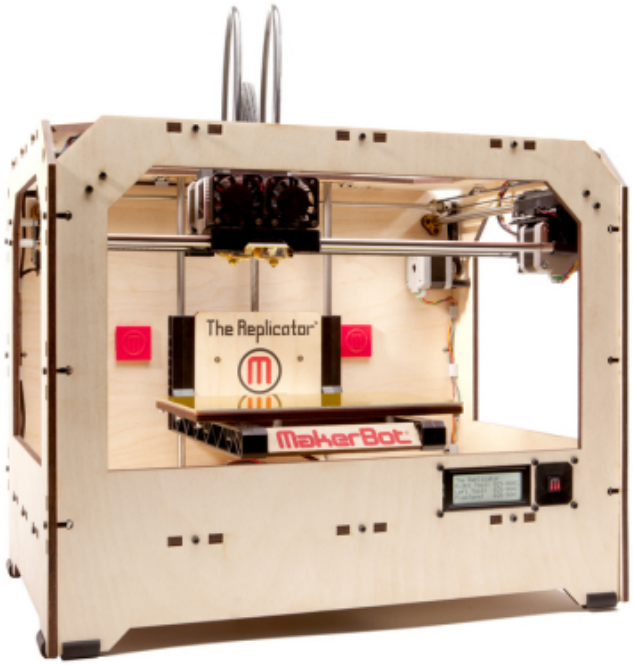


Figure 4: MakerBot Replicator (http://www.zahncenternyc.com/wp-content/uploads/2014/08/MakerBot-Replicator.jpg)

The hardware for the RepRap was designed to be simple to construct, using threaded rods for the frame which allowed for accurate adjustment. The corner knuckles were made of plastic and able to be 3D printed to allow it to replicate its own parts. The electronics used were based around the Arduino platform, an open source single board microcontroller with an Atmel 8-Bit AVR at the core.

Arduino was the brain-child of Massimo Banzi, who taught at the Interaction Design Institute Ivrea in Ivrea, Italy (Kushner, 2011). It was developed in response to the high cost of the 'BASIC Stamp' used for teaching at the time in 2005. As was the design philosophy for Arduino, the RepRap project used custom add-on boards (shields in Arduino terms) to control the stepper motors for each axis. Arduino was also used as the basis of the design for the extruder controller, but rather than having additonal shields to control the heater and motor control, a board was designed incorporating all of the necessary features.

The cross-platform Arduino Integrated Development Environment (IDE) was written in Java and derived from the Processing programming language and Wiring project (Kushner, 2011). Programming in C/C++ with a 'Wiring' wrapper, the IDE allowed the novice programmer to compile and upload to the Arduino hardware with a single button press.

Although the RepRap hardware used the Arduino platform as a basis, the RepRap firmware was written in native C++ rather than the Arduino 'sketch' based IDE. It used the GNU AVR-GCC toolchain and AVR-DUDE to upload the initial code to the Arduino via the In-Circuit Serial Programming (ICSP) port. This allowed for faster, more comprehensive control at a lower level, with some registers being set directly rather than through high-level function calls. RepRap used a modified Arduino bootstrap to allow firmware updates to be done over the USB connection.

G-code or RS-274-D (Electronic Industries Association, 1969), used to program Computer Numerical Control (CNC) equipment was seen as the most sensible 'language' for controlling the printer. However, directly interpreting the native ascii g-code places a heavy burden on the microcontroller and the serial link from the PC to the printer. As a result, the g-code is transcoded at the computer into a low-level hexadecimal format, Sanguino 3G or S3G (Metts), Sanguino being the particular implementation of the Arduino platform being used.

The software used for controlling the RepRap and most 3D printers derived from RepRap is called ReplicatorG. It is also open source and written in Java to

be cross-platform (ReplicatorG). ReplicatorG is written to work with either STereoLithography (STL) files or a sub-set of G-code (Bowyer, 2014).

The STL file format was developed for 3D Systems, by the Albert Consulting Group, for their stereolithography process (STL, 2014). Surface objects are represented using a triangulated mesh much as finite elements analysis packages currently use. When a CAD model is converted to STL format, the surfaces are represented with triangles, the size of the triangles is determined by the required resolution and the the maximum deviation allowed from the surface. The higher the resolution, the smaller the triangles but the bigger the file. There is a trade-off between the file size/resolution and the capabilities of the machine processing the file, there is no point generating STL files at a resolution higher than the positional step size of the machine.

Skeinforge is a 'plug-in' used with ReplicatorG to perform the slicing of STL files and generation of G-code for each individual layer. The power of Skeinforge comes from the fact that it is massively configurable, this is probably also a draw-back as an in-depth knowledge of the configuration settings is needed to produce a really good result. Later versions of ReplicatorG included other open source slicing engine options like Slic3r by Alessandro Ranellucci (http://slic3r.org) and Miracle Grue by Makerbot Industries (http://makerbot.github.io/Miracle-Grue).

The principle behind the slicing software is essentially simple. A perimeter is generated by joining up all of the points at which the layer cuts each triangle edge. The interior of the model is then 'filled' according to the selected fill pattern and density. The complexity comes from all of the parameters that can be modified relating to the extrusion rate, speed, how much to 'back-off' at the end of a run, where to place support material etc. If an object is to be printed on a 'raft', the slicing engine is also responsible for determining the layout of the raft depenent upon the operator setting for number of layers, distance the raft extends beyond the object perimeter etc.

STL as a file format has is limitations. STL files contain no scale information,

units are arbitrary. There is also no colour or material information. Newer formats such as the Additive Manufacturing File (AMF) format, an Extensible Markup Language (XML) based format, developed by ASTM, have been introduced to address some of these issues (ASTM, 2013).

Due to past experience with the MakerBot Thing-O-Matic, it was decided that the MakerBot would be used as the baseline for this build.

3 Mechanical Design

3.1 Motors

The FDM Vantage X uses 3-phase, brushless AC servo motors (ElectroCraft XBR-2910) to drive X, Y and Z axes.

3.1.1 Servo Motor Theory

AC servo motors are an induction motor. The rotor is made of laminated steel with conductive bars embedded in them, usually copper or aluminium. The conductive bars are short circuited together at the ends, forming a 'squirrel cage'. Laminated steel is used to reduce eddy current losses.

The stator is also made of laminated steel. Coils are wound around the laminates in slots. The coils are connected to a variable frequency power supply. When power flows through the windings, a rotating magnetic field is setup. A voltage is induced in the conductive bars which causes a current to flow in them. The resulting electromotive force causes the rotor to accelerate. As the speed of the rotor increases, the voltage induced decreases. As the speed of the rotor approaches that of the rotating field (synchronous speed), the resulting torque reduces to zero. In practice, due to drag and friction forces, the speed of the rotor is always lower than the synchronous speed. The speed of rotation can be calculated based upon the frequency of the supply and the number of pole-pairs on the stator.

$$\omega_f = 2 \cdot \pi \cdot f / p$$

where :

ω_f is the synchronous rotating speed in radians/sec

f is the supply frequency (Hz)

p is the number of pole-pairs

With the decrease in the cost of solid-state electronic switching devices, control of AC servo motors has become relatively cheap. In the Stratasys FDM Vantage X, power for the motors is a regulated 48V, 16.5A DC supply from an Astec MP8-3W-1W-1Q-30 modular power supply. The DC supply is inverted using thyristor or Insulated-Gate Bipolar Transistor (IGBT) switches in a 3-phase bridge configuration.

By turning on/off the thyristors in sequence, the current can be directed through each of the windings of the motor. Thyristors continue to conduct until the current returns to zero, the frequency controller must electronically force conduction to stop. Cummutation control is generally achieved using hall effect sensors on the back of the motor.

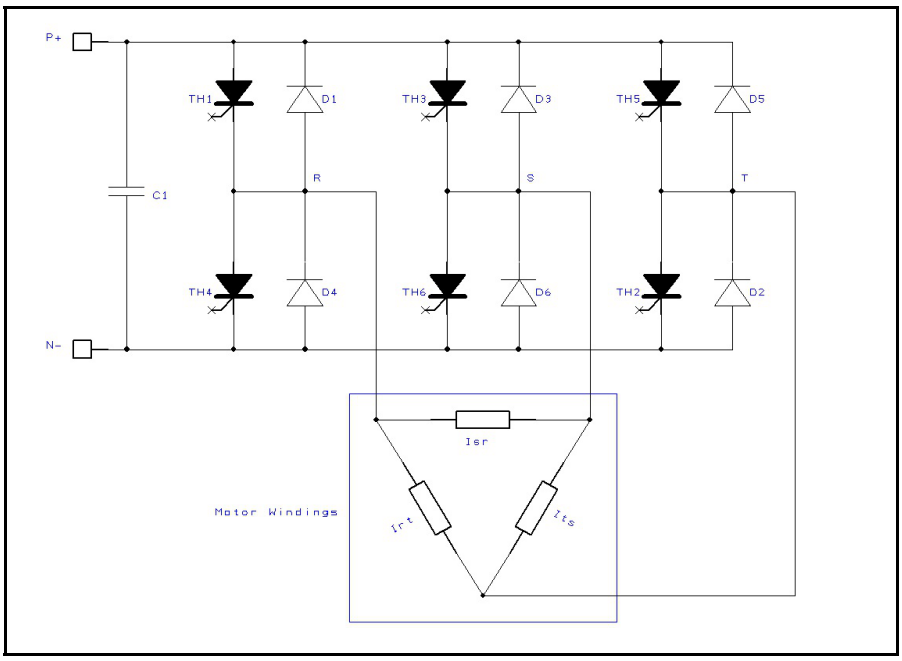


Figure 5: Simplified AC Motor Control

Since the motor tries to match the synchronous speed of the supply, by varying the frequency of the switching, the speed of the motor can be controlled. In the case of the motors used in the Stratasys FDM Vantage X, the control signals are Pulse Width Modulated (PWM) using Advanced Motion Controls B12A6 PWM Servo Amplifiers, providing a variable frequency 3-phase AC motor drive.

The input to the B12A6 PWM amplifiers is analogue, ±10VDC, from an ACROLOOP ACR8000 in the industrial PC.

On its own, an AC motor has only coarse positional control, provided by the commutation hall effect sensors. However, servo motors are fitted with a positional encoder attached to the shaft on the back of the motor and used to close the control loop. The encoder can be either absolute or incremental.

An absolute rotational encoder gives the position of the shaft regardless of any power cycling. Historically absolute encoders have been optical, providing a position in either Binary Coded Decimal (BCD) or more commonly Gray Coded (GC). The benefit of GC over BCD is that only 1 bit ever changes at a time, whereas BCD changes multiple bits at a time. By changing only a single bit, errors can more easily be detected and corrected. Modern high resolution magnetic encoders are now being used for improved reliability.

Decimal	**Binary Coded Decimal**	**Gray Code**
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Table 3: Gray Code vs Binary Coded Decimal

Incremental encoders such as the standard quadrature encoders used on many servo motors provide position relative to a reference point. They require a 'zero' calibration prior to use. They usually have three outputs, A, B and Z or Index. The A and B outputs provide angular position and direction, whilst the Z or Index provides a single pulse per full rotation for example each time 180º is passed on the shaft.

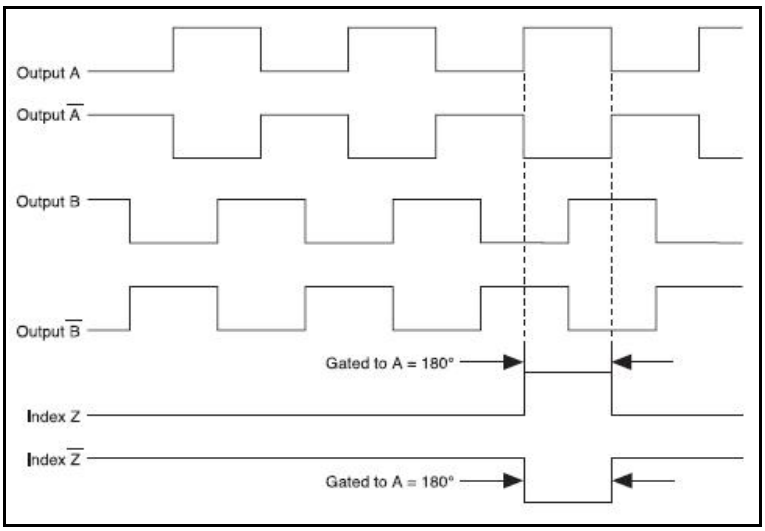


Figure 6: Quadrature Incremental Encoder Output Waveform (http://sine.ni.com)

The ACROLOOP in the industrial PC uses the feedback from the incremental encoders integrated into the XBR-2910 servo motors, giving full closed-loop speed and positional control after an initial 'zero'.

3.1.2 Stepping Motor Theory

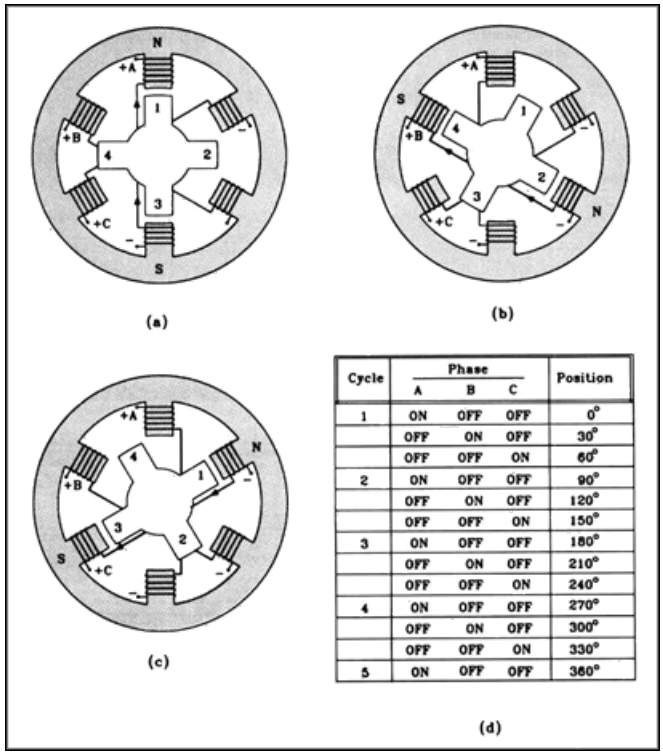
An alternative to servo motors is to use stepping motors. There are essentially two different types of stepping motor, Variable Reluctance (VR) stepping motors and Hybrid stepping motors. Stepping motors are open-loop, there is usually no positional feedback in the form of an encoder. Position is determined by the number of steps moved, giving angular displacement. Like incremental encoders, stepping motors require to be 'zeroed' at startup.

VR stepping motors have a laminated iron rotor and stator. All of the magnetic flux required to move the stator or hold the stator in place is generated by the stator windings.

With a single stack VR motor, the number of rotor and stator teeth determine the step angle generated.

Step Angle = 360 / (Poles x Teeth)

In the example shown in Figure 7, there are four rotor teeth and 3 poles (1 pole is made up of 2 teeth, creating a north and a south pole pair), giving a step angle of 30°.



Cycle	Phase A	Phase B	Phase C	Position
1	ON	OFF	OFF	0°
	OFF	ON	OFF	30°
	OFF	OFF	ON	60°
2	ON	OFF	OFF	90°
	OFF	ON	OFF	120°
	OFF	OFF	ON	150°
3	ON	OFF	OFF	180°
	OFF	ON	OFF	210°
	OFF	OFF	ON	240°
4	ON	OFF	OFF	270°
	OFF	ON	OFF	300°
	OFF	OFF	ON	330°
5	ON	OFF	OFF	360°

Figure 7: Cross-section through Single-Stack VR Stepping Motor

A three-stack VR stepping motor has the three phases connected to individual poles with an individual stator, see Figure 8 below. Each stator is offset by the step angle such that each phase is excited in sequence causing the shaft with all three stators to rotate. In contrast to the single stack VR, the multi-stack VR has the same number of teeth on the rotor as the stator, when each phase is energised the magnetic flux causes all the teeth on the stator to become aligned with the rotor teeth.

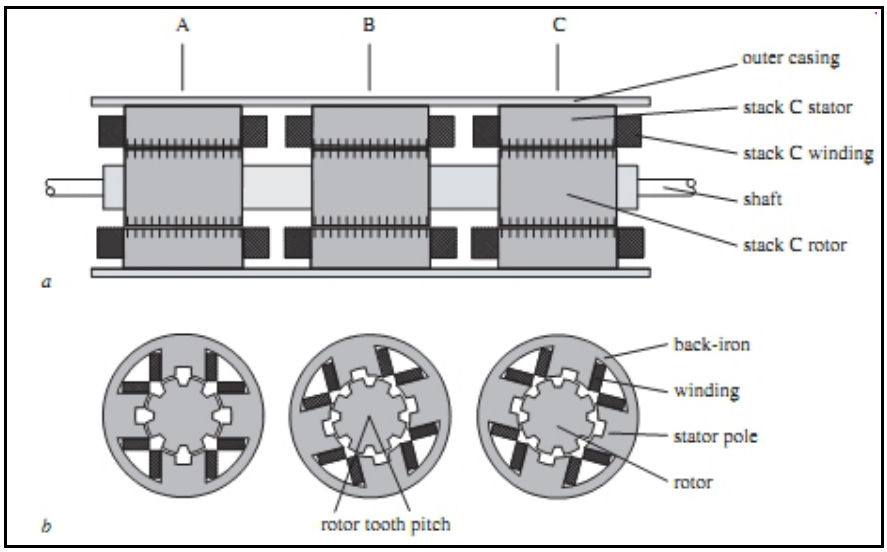


Figure 8: Cross-section of three-stack variable reluctance stepper motor (http://www.industrial-electronics.com)

Hybrid stepping motors have a laminated iron stator. On the rotor, a permanent magnet is mounted between laminated iron end-caps. The magnetic flux runs from the north-pole of the permanent magnet, into the iron end-cap, across to the stator, through the back iron of the stator and back across to the south-pole end-cap, completing the magnetic circuit. The stator windings are used to either encourage or discourage the magnetic flux through certainn poles. There are typically eight stator poles, and two windings (A & B).

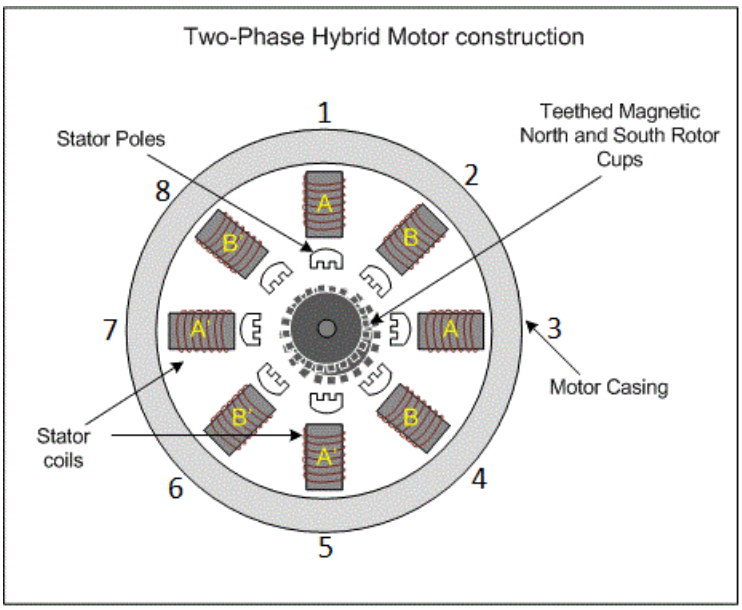


Figure 9: 2-Phase Hybrid Stepper Motor (http://www.engineersgarage.com)

All of the 'A' windings are connected together and all of the 'B' windings are connected together, however opposing pairs of poles are wound in the opposite direction, so each winding pair has a positive and a negative current direction.

Winding	Current direction	Direction of pole magnetic field	
		Radially outward	Radially inward
A	Positive	3,7	1,5
A	Negative	1,5	3,7
B	Positive	4,8	2,6
B	Negative	2,6	4,8

Table 4: Magnetic Field Direction in a 2-Pole Hybrid Stepper Motor

The step length of a hybrid motor is determined by the number of teeth on the rotor. A full excitation cycle (A+, B+, A-, B- for clockwise or A+, B-, A-, B+ for anti-clockwise) produces 4 steps, and since the next step requires the next tooth to be in the same place as the first, 4 steps equates to 1 tooth pitch. So, from the above is can be shown that the step length can be calculated using the following:

Step Angle = 360 / 4 p

Where:
p = number of teeth on the stator

A common step length for a hybrid motor is 1.8º, by transposing the above, p = (360 / 1.8) / 4 = 50 teeth.

Current open source 3D printers all use stepping motors for X, Y and Z axes, so rather than designing an interface to convert the stepping motor signals to servo motor signals, it was decided that the servo motors should be changed to stepping motors.

The OEM motors were identified as ElectroCraft XBR-2910, NEMA34 Brushless AC Servo Motors with integrated Encoders.

Parameter	Value
Torque Constant (Kt)	1.8 in-lbs/A
Voltage Constant (Ke)	19.9 – 23.1 V/kRPM
Rated Voltage	230 V
Continuous Speed (no load)	6000 rpm
Peak Torque	18.5 in-lbs / 2.1 Nm
Continuous Torque	6.1 in-lbs / 0.7 Nm
Inductance	5.8mH
Encoder Count	2000 ppr
Encoder Voltage	5 Vdc
Encoder Current	225 mA

Table 5: XBR-2910 Specifications

A set of 3 motors, with matching micro-stepping drives and power supplies, were sourced on Ebay.com from the Longs Motor Company in China.

Parameter	Value
Part Number	34HS1456
Step Angle	1.8º
Current	5.6A/phase
Resistance	0.55Ω/phase
Inductance	5.5mH/phase
Holding Torque	1232 oz-in / 8.7Nm
Rotor Inertia	2700g-cm^2
Weight	3.8Kg
Length	118mm

Table 6: Longs 34HS1456 Stepping Motor Specifications

Higher torque rated stepping motors were selected as stepping motors not having any positional feedback must be run well below their rated capacity in order to minimise the risk of missed steps due to acceleration. The motors selected are 1232 oz-in or 8.7 Nm, 5.6A.

The micro-stepping drives are DM860A, 24-80VDC up to 8.0A/phase and capable of 256 pulses/step, giving up to 51200 pulses/revolution with a 1.8º stepping motor (200 steps/revolution). However, for the purposes of this project, the MakerBot Thing-O-Matic is designed around 1/8 steps, so a setting of 1600 pulses/revolution was selected on the dip switches.

The micro stepping drives and power supplies were installed in the bottom of the machine after removing the redundant Astec MP8 power supply. As there was an individual power supply and drive per axis, each was put on a separate phase to help balance the electrical loads.

The existing servo motors were unbolted, but before the replacement motors could be fitted to the printer, the pulleys had to be transferred from the the motor spindles. On removing the pulleys, it was discovered that the shaft diameters were different. The original motors had a shaft diameter of 1/2”, while the new stepping motors had a shaft diameter of 14mm. There was sufficient shoulder on the pulleys to be able to hold them in a lathe chuck and ream the shaft hole

to 14mm whilst retaining sufficient material for the keyed shaft.

3.2 Front Panel Controller

The Stratasys FDM Vantage X Front Panel Controller (FPC) used a 4x40 LCD display along with a numeric keypad and a number of navigation buttons. A large green illuminated push-button and smaller red push-button start and stop the machine.

Figure 10: Stratasys FDM Vantage X Front Panel Controller

There was a single 9-way D-type connection to the FPC. The start/stop buttons were connected through the D-type to the Power Supply PCB. The start button was normally-open, while the stop button was normally-closed. The remaining connections on the D-type were a 12Vdc supply from the industrial PC and an RS-232 serial connection to the industrial PC.

The initial plan was to re-use the FPC so it was necessary to reverse engineer the serial interface to the industrial PC. Using a pair of RS-232 to USB adapter cables, the TX and RX lines were monitored using a terminal client. Initial attempts using HyperTerm were unsuccessful as HyperTerm was not capable of properly decoding ascii control characters like 'Escape' (ESC). Realterm was used instead as it was capable of capturing the data in hexadecimal format. This allowed full capture and decoding of all of the messages between the FPC and the PC.

The transmit pin was connected to the receive pin on adapter 1 and the receive pin was connected to the receive pin on adapter 2 as shown below.

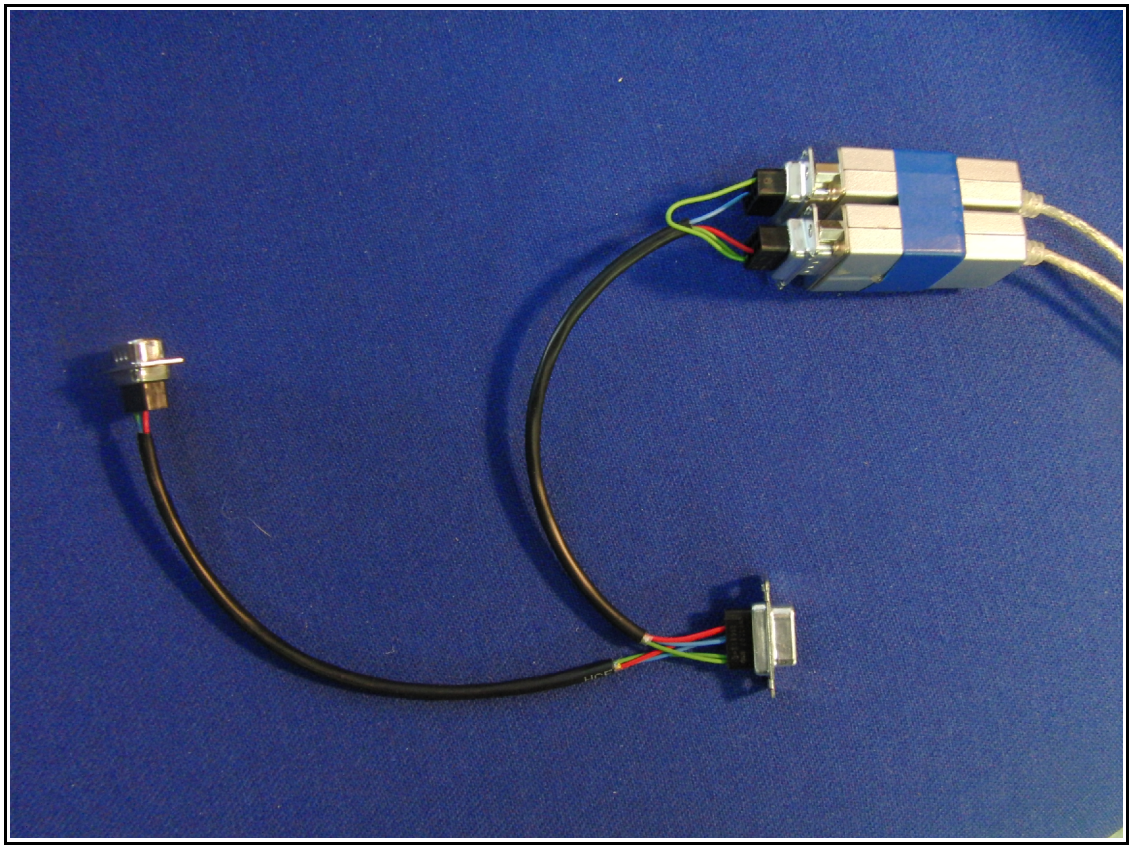


Figure 11: Message Snooping using Dick Smith Electronic RS-232 to USB Adapters

Table 7 shows the serial output codes used to represent each button on the FPC.

Button	ASCII	HEX
Menu	“M”	0x4D
Up Arrow	“+”	0x2B
Down Arrow	“-”	0x2D
Light	“L”	0x4C
Pause	“P”	0x50
Help	“?”	0x3F
Enter	CR	0x0A
Delete	DEL	0x7F
Escape	ESC	0x1B
.	“.”	0x2E
0-9	“0”-”9”	0x30 - 0x39

Table 7: Serial Output Data from FPC

An example of the format for sending data to the FPC is shown below.

[0x1B] [0x5B] [0x30][0x3B][0x37][0x48][0x4A]

Byte 1 – 0x1B, ASCII ESC
Byte 2 – 0x5B, ASCII [
Byte 3 – The row number (0:3), 0x30 – 0x33, ASCII 0-3
Byte 4 – 0x3B, ASCII ;
Byte 5 – The column number (0:39), 0x30 – 0x39, ASCII 0-9
Where the column number is greater than 9, an extra byte is sent.
Byte 6/7 - 0x48, ASCII H
Byte 7/8 – The ASCII character to be displayed in HEX, e.g. 0x4A, ASCII J

As can be seen from the example above, the overhead for sending messages to the FPC was significant. It was determined that the added complication of the changes to the hardware and firmware to handle the serial interface for the FPC was unwarranted.

It was therefore decided that the FPC should be redesigned using the physical form of the original panel but the interface and basic button structure from the

MakerBot design. The replacement FPC was designed in Solidworks, re-using the 40x4 LCD screen size but using a button layout more suited to the MakerBot design. The SD card interface was also moved to the FPC to allow for easy access. The panel and buttons were then 3D-printed in Nylon at Massey University, Albany on a TPM3000 Selective Laser Sintering machine. The replacement FPC is shown in Figure 12.

Figure 12: Replacement FPC

3.3 Extruder Head Solenoid

During testing, due to a design error, the solenoid used for selecting between the two extruders overheated and burned out. When approached with regard to replacing the solenoid, Tasman Machinery would only quote for a complete replacement extruder head at a cost of $20,000. The damaged solenoid was removed from the machine and OEM markings (LISK S-2458-48) found, indicating the device was made by G.W. Lisk in the United States. Lisk were

contacted and supplied a quote, by email, to replace the solenoid at a cost of $900 USD + shipping. Lisk supplied a specification sheet (G. W. Lisk Company, 2011) for the device and it was decided that the most cost effective and quickest solution to the problem was to repair the original solenoid.

Using a hydraulic press in the DTA workshop, the coil was pushed out of the housing. The coil was taken to Transformer Specialties in Kumeu and re-wound by hand for $345 and returned the following day. In the process of removing the coil, the housing was damaged beyond repair, so while the coil was being rewound, a new housing was machined in the DTA workshop using a lathe. Once the coil was returned, the unit was reassembled and crimped back together.

The cause of the overheating solenoid was found and is discussed further in the Chapter 4 – Electrical Design.

3.4 Other Hardware Modifications

Other modifications included the removal of the Canister Controllers from the front of the machine. This was to allow the use of alternate materials, not contained in canisters but loaded onto drums. There was a small compressor and associated receiver and pressure regulations devices which were no longer required. These were also removed.

In the base of the machine, a large UPS was located, that provided backup power to the industrial PC. This was unserviceable, so was removed along with the industrial PC which was no longer required. The space left from removing the UPS was used to house the power supplies for the individual axis stepping motor drives. The cover over this was modified, reducing it's width, to allow a smaller UPS (Blazer 800, acquired from TradeMe) to be fitted along side. This was used to provide backup power to the micro-controllers.

The OEM printed circuit boards (PCB's) were also removed and replaced with new hardware, to be discussed further in Chapter 4 – Electrical Design.

4 Electrical Design

The electrical design was split into three main components, the Power Supply, the Extruder Controllers and the Motherboard. The design for each is discussed further below, with complete design files (including gerber files for PCB manufacture) contained in Appendix A (See attached CD-ROM)

4.1 Power Supply

The OEM power supply was a delta configuration with a working voltage of 230v phase-phase. However, the standard 3 phase voltage in New Zealand is 230v phase-neutral or 415v phase-phase. As a result, when the machine was originally delivered, it utilised a heavy duty transformer on a trolley for converting the New Zealand supply voltage to a suitable level. A 4-core cable was used to connect the 3 phases and earth to the transformer and then, from the transformer to a gland on the back of the machine. Inside the machine, the 3 phases were connected to terminal blocks and then to a 3-phase circuit-breaker. The output of the circuit-breaker was connected to a filter bolted directly to the side of the machine housing.

The Power Supply PCB was connected directly to the output of the filter by studs with nuts clamping either side of the PCB.

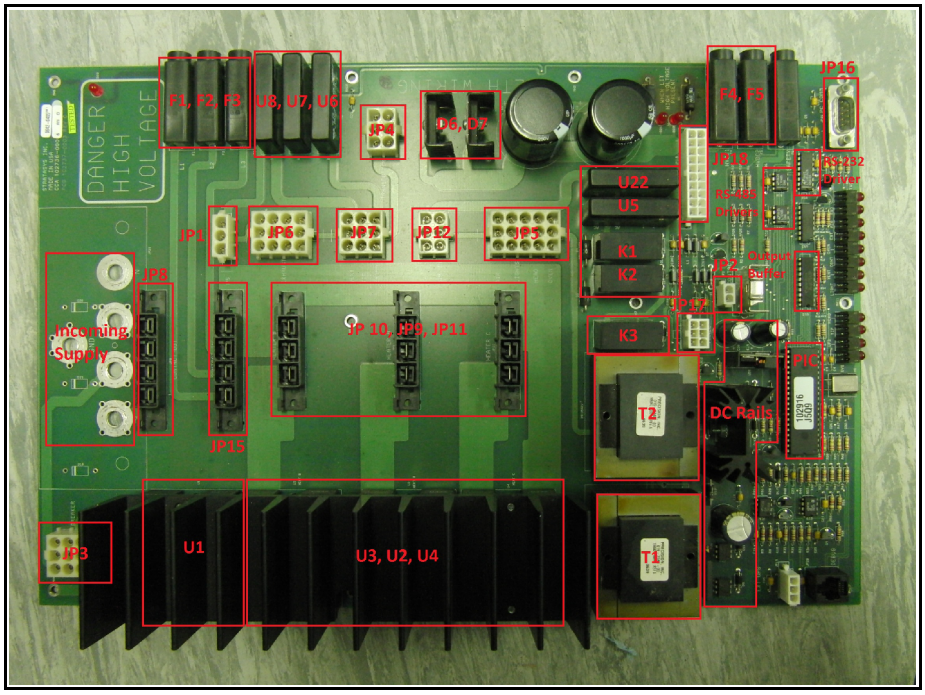


Figure 13: OEM Power Supply PCB

Phase A-B was connected off-board, via part of JP3, to the main power switch on the back of the machine. This connected power to the primary side of transformer T1. The secondary side of T1 was connected to a bridge rectifier (D10) and regulated DC power supplies (U20 and U21) for the PIC16F877 micro-controller (U9). The PIC supplied a control signal to U1 (Solid State Relay Crydom D2425), switching phase A-B to the UPS and the vacuum pump on the remainder of JP3.

The UPS supplied power to JP12, and back out to the industrial PC. JP12 was connected to the primary side of transformer T2. The secondary side of T2 supplied a bridge rectifier (D12), which was connected to the same power rails as D10. Opto-coupled outputs provided an indication to the PIC if the supplies were on.

The incoming power from the UPS was connected to U22, U5, K1 and K2, which control the AIR, LIGHTS, HEAD and OVEN circuits respectively on JP5 all driven by the PIC either via the buffer U15 or transistor driver circuits for the relays K1 and K2.

The PIC controls a DC signal to JP2, which is connected to the off-board contactor located underneath next to the incoming power filter. Phases A, B and C are connected from the filter to JP8. JP8 connects to the input side of the off-board contactor, with the output side of the contactor connecting back on to the PS at JP15 as L1, L2 and L3.

Three solid state relays (U2, 3 and 4) controlled power to the heater elements A, B and C. Heater A was connected to phases L1 and L2, Heater B was connected to phases L2 and L3 and Heater C was connected to phases L1 and L3. The control signal for the relays was connected to JP18 which in turn was connected to the I/O Control PCB.

L1, L2 and L3 were fused, via F1, F2 and F3. L1-L2 was connected via JP6 to the four exhaust fans located in the top of the machine. L2-L3 was connected via JP1 to an external power supply (Astec MP8-3W-1W-1Q-30) which provided 48Vdc and 24Vdc to the I/O Control PCB. L1-L3 was connected to solid state relays U6, U7 and U8. U6 controlled power to the EXTRA circuit (connected to the compressor) on JP7. U7 controlled power to the TIP circuit on JP7. U8 controlled power to the VENT circuit on JP7. U6, U7 and U8 were driven from the PIC, through buffers (U15).

L1-L3 was also connected via JP4 to an off-board transformer (located underneath, with the contactor and power supply filter) the output from the transformer was connected back to JP4 and supplies two bridge rectifiers (D6 and D7). The outputs of the rectifiers were fused through F4 and F5 respectively and connected to JP18, providing High Voltage (150V) DC supplies to the I/O Control PCB.

The PIC also provided an RS232 serial connection via U12 to the industrial PC on JP16 and two RS485 serial connections via U10 and U11 to the I/O Control PCB on JP18.

With the opportunity to re-design the power supply, it was decided to convert

the machine to a 'star' configuration, meaning that it could be plugged directly into a standard New Zealand 3-phase supply. The change to a 'star' configuration required a change of connector and cable from 4-core (3P&E) to 5-core (3P,N&E). Internally, it required an additional terminal block to connect the neutral to the filter.

The PCB also had to be re-designed to switch phase-neutral rather than phase-phase. To save money and given that it was not planned to re-use the power supply PCB, it was stripped to re-use the parts, saving a significant cost in components (mainly connectors and solid-state relays SSR's).

Initially the plan was to make the replacement PCB the same foot-print as the original. However, when the cost of manufacturing the board was investigated, it was prohibitively expensive. Accordingly, it was decided that the power supply PCB be split into two parts, a Power Supply Board and a Power Supply Control Board. This enabled both boards to be manufactured using the LPKF PCB mill at DTA.

The design philosophy used was essentially the same as the OEM, the incoming supply from the filter was connected to the contactor. L3 was connected to the main switch on the back panel, controlling power to the Power Supply Control Board, which in turn was used to switched using a Crydom MCX240D5 solid-state relay (SRR) to power the UPS and a small Point of Load (PoL) 24VDC power supply for the contactor. The signal for the contactor was from the Power Supply Control Board and provided the gate voltage to an FET to switch the contactor. Power supplies for the stepping motor drivers were taken from the fused side of the contactor return side as shown in Figure 2. The heater loads were distributed across the 3 phases and also switched using Crydom D2425 SSR's as shown in Figure 3. The remaining loads were distributed as evenly as possible to minimise the neutral current flowing. See Table 1 and Figures 4 and 5 for High Voltage DC (HVDC) rectification and Auxilliary Power Control respectively.

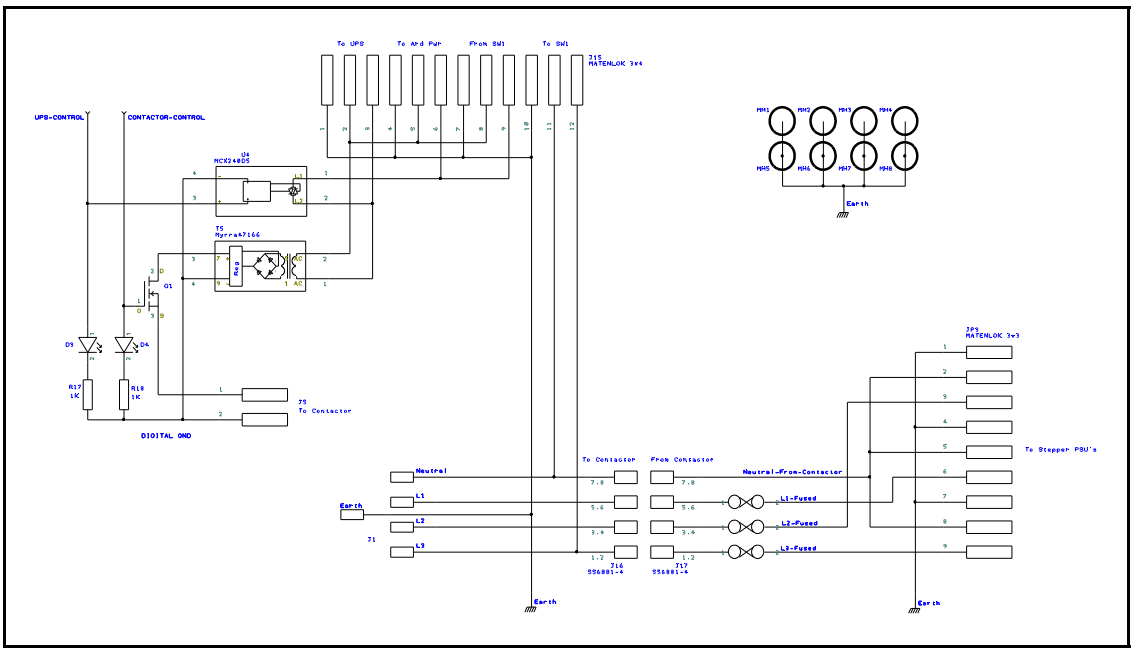


Figure 14: Incoming Supply Distribution

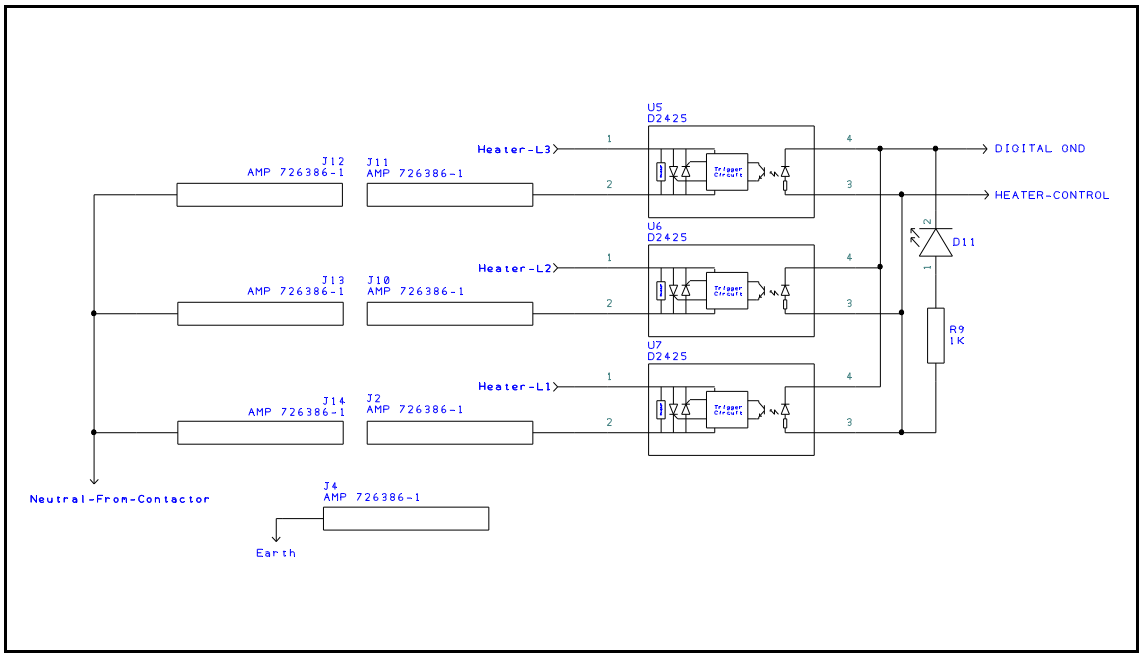


Figure 15: Heater Control

Phase	Load	Power
L1	Heater A	1.8kW
	X-Axis Stepper PSU	350W
	Other Ancillaries	~400W
L2	Heater B	1.8kW
	Y-Axis Stepper PSU	350W
	HVDC	~900W
L3	Heater C	1.8kW
	Z-Axis Stepper PSU	350W
	UPS + 24VDC	~400W

Table 8: 3-Phase Power Supply Loading

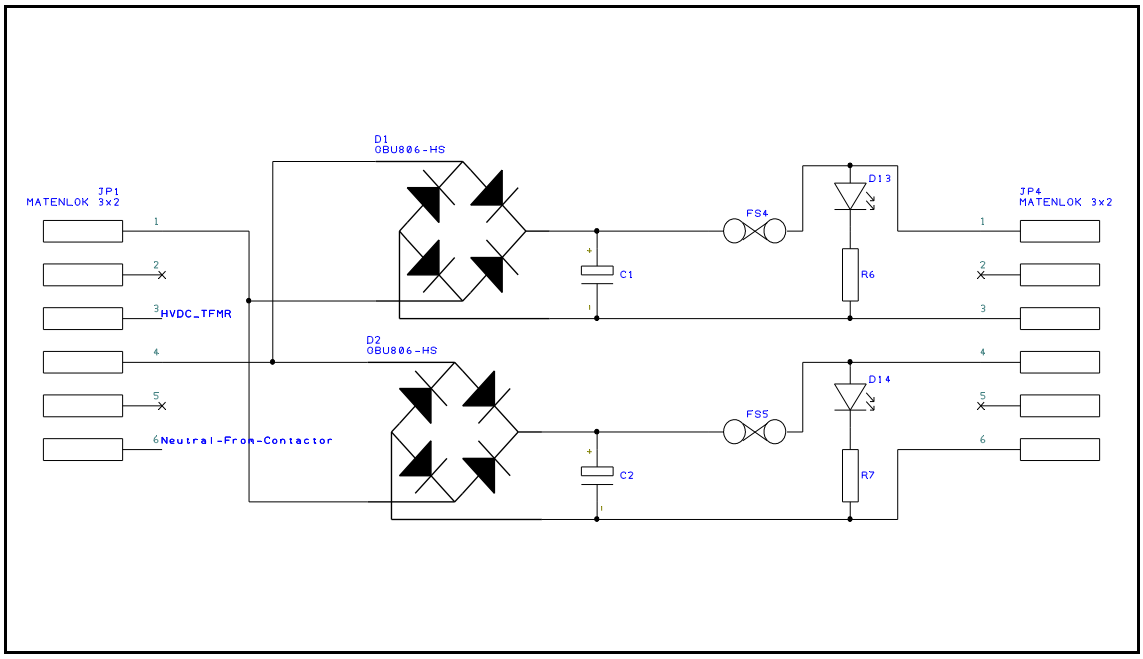


Figure 16: High Voltage DC Rectifier

The two HVDC supplies for the extruder head were developed from the secondary coils of a power transformer enclosed in the bottom of the machine. The outputs were rectified using GBU806-HB bridge rectifiers and smoothing capacitors. Each output was fused and an indicator LED connected using a high power resistor as shown in Figure 16.

Figure 17 shows how the remaining Auxilliary loads were switched using Crydom MCX240D5 SSR's along with indicator LED's.

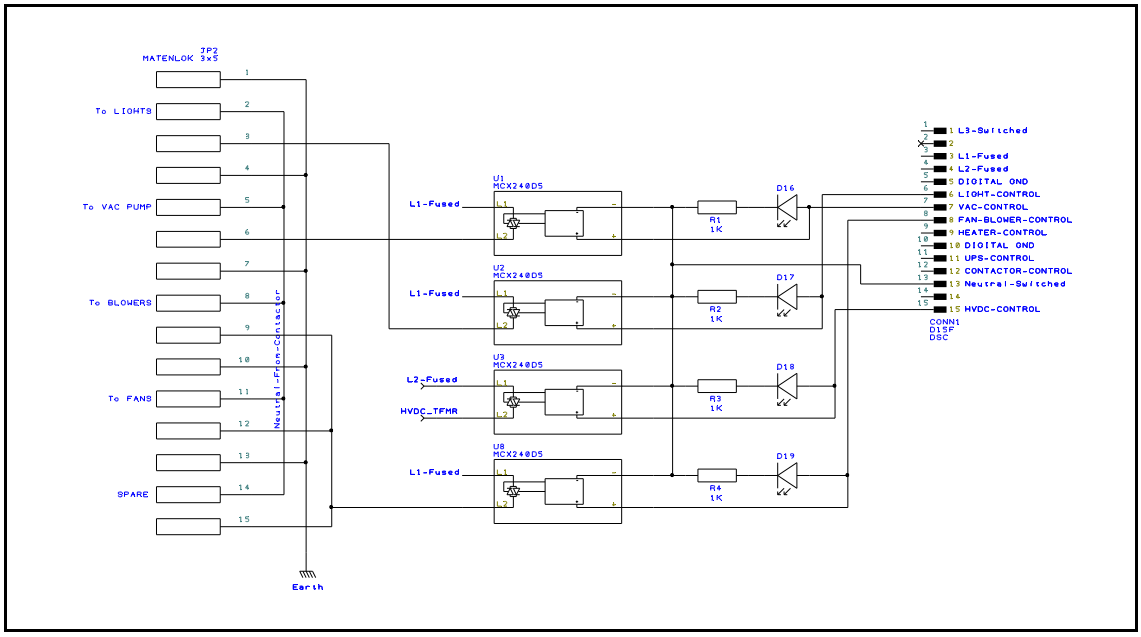


Figure 17: Auxilliary Power Control

The Power Supply Board was drawn up in DesignSpark PCB and a PCB generated, see Figures 18 and 19.

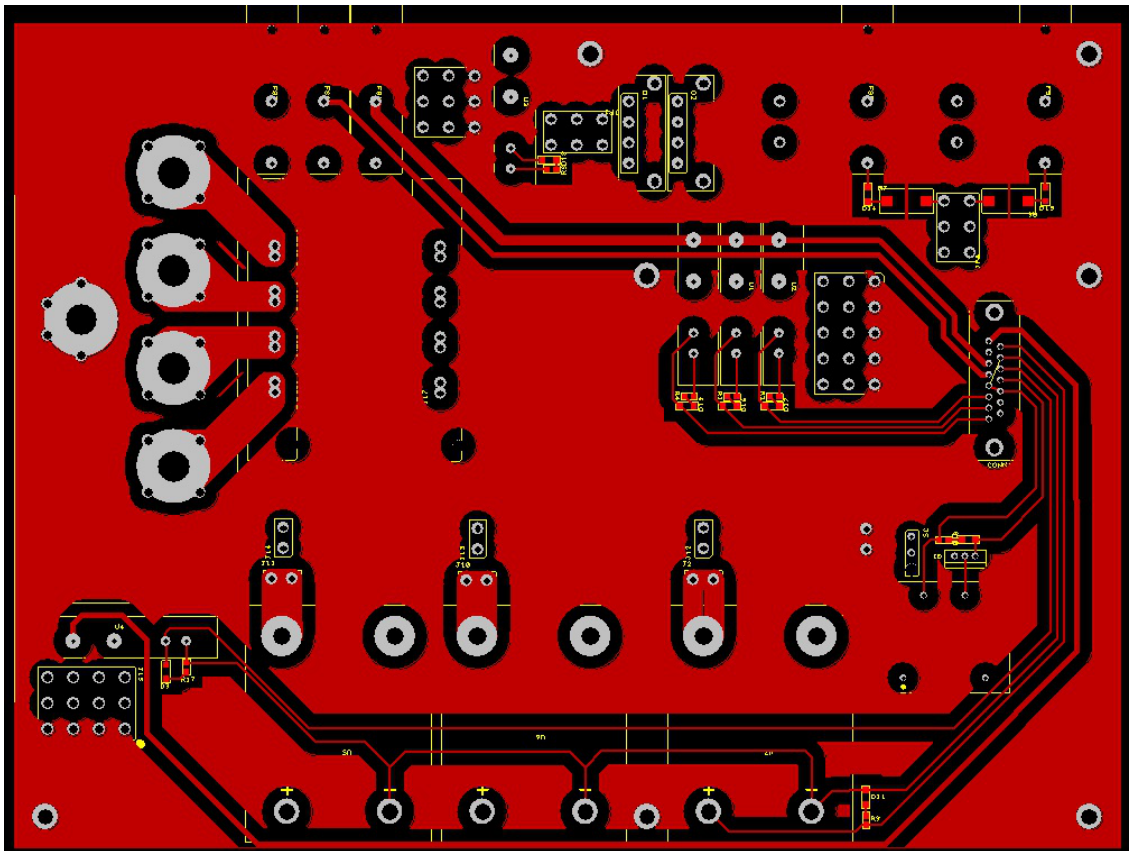


Figure 18: Power Supply Board - Top Copper

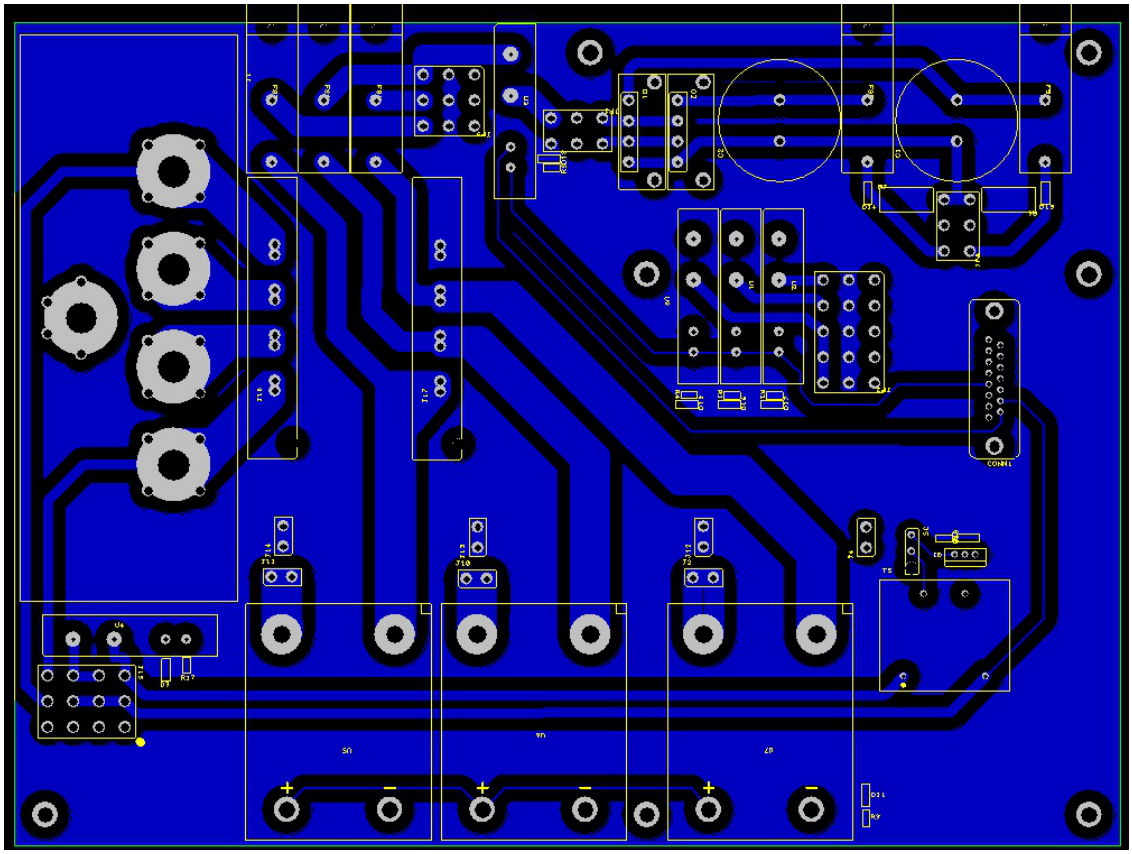


Figure 19: Power Supply Board - Bottom Copper

A fault was found with this design during testing. Switching the 24V to the contactor would not work. After some investigation, it was discovered that using a 5V control signal to high-side switch the 24V would not work as there was insufficient potential difference across the gate to switch the device on. The PCB was modified to switch the 24V to the contactor on the low-side.

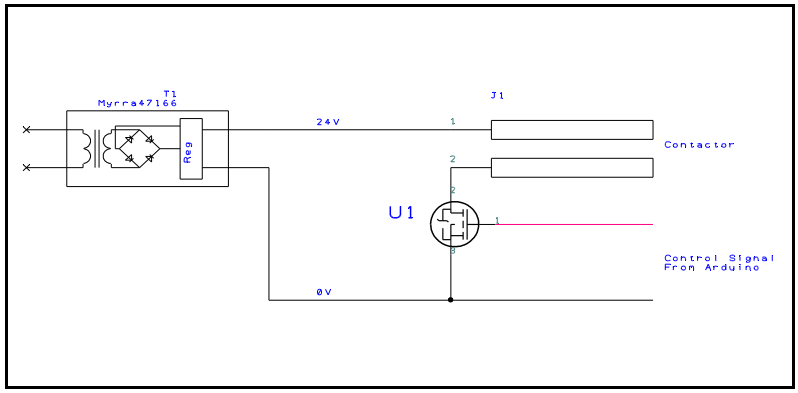


Figure 20: Low-Side Switching Modification

The digital control board was designed around an Arduino Mega. Although an Arduino Uno would have satisfied the number of I/O's required, the initial plan call for at least three hardware UART's, RS-232 port to communicate with the Web Server, an RS-232 port to communicate with the UPS and a logic level serial port to connect directly to the Motherboard.

The Arduino was powered from a redundant 12V supply, generated by a pair of Myrra 47154 regulated 12V PoL power bricks. The primary power supply was connected to L3 from the Power Supply Board and the secondary was connected to the UPS return from the Power Supply Board. The outputs of the two power supllies were tied together through a pair of 1N4001 rectifier diodes. The output of each power supply was connected to the input of a 4N35 opto-isolator. The output side of the opto-isolator was connected to the Arduino and also an LED to indicated the presence of power on each as shown in Figure 21.

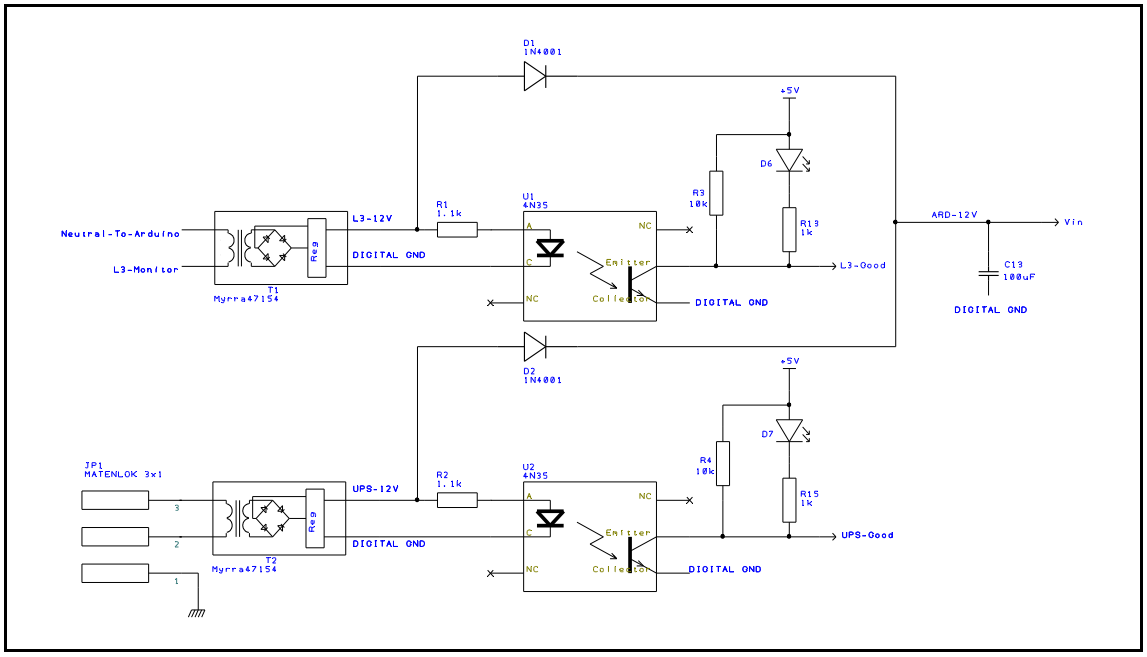


Figure 21: Arduino Mega Redundant Power Supply

Two additional Myrra 47154 PoL power bricks were used in the same configuration as shown in Figure 21, for monitoring the L1 and L2 from the Power Supply Board. This allowed the Arduino Mega to determine if all 3

phases were connected and healthy.

Initial testing of the UPS was done using the USB port connected to a PC running WinPower, downloaded from http://www.ups-software-download.com/winpower/windows.htm. This was successful, however, when re-configured to use the COM port WinPower was unable to communicate with the UPS. Using Google (https://www.google.co.nz/search?q=blazer+UPS+protocol) it was determined that the protocol used by the Blazer UPS was the Megatec/Q1 protocol (Chiou, 1996). According to the protocol specification, the serial connection to the UPS had to not only provide a comms link, but also supply power to the UPS serial interface board. For testing, power was connected directly to pin 7. Although power was being provided to the serial interface board, no communications could be achieved. After further investigation it was decided to open up the UPS to verify that the serial interface was indeed implemented. On opening the UPS, it was discovered that a diode (D029) was missing from serial interface board, see Figure 22. A 1N4148 diode was fitted to the board and the board re-fitted to the UPS and serial comms was restored.

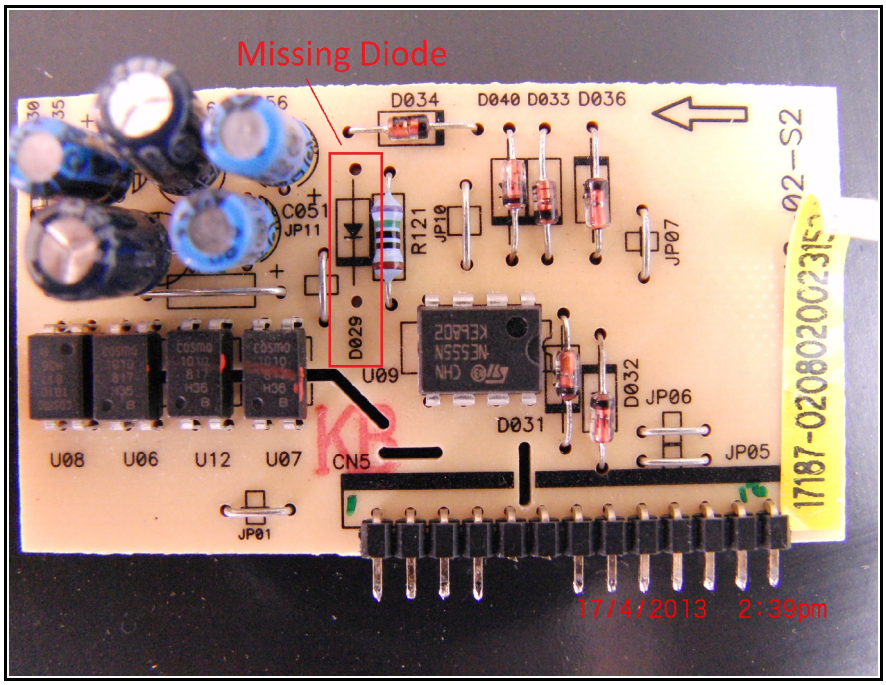


Figure 22: Serial Interface Board in UPS

As discussed above, the initial plan was to have three hardware UART's for communications as shown in Figure 23.

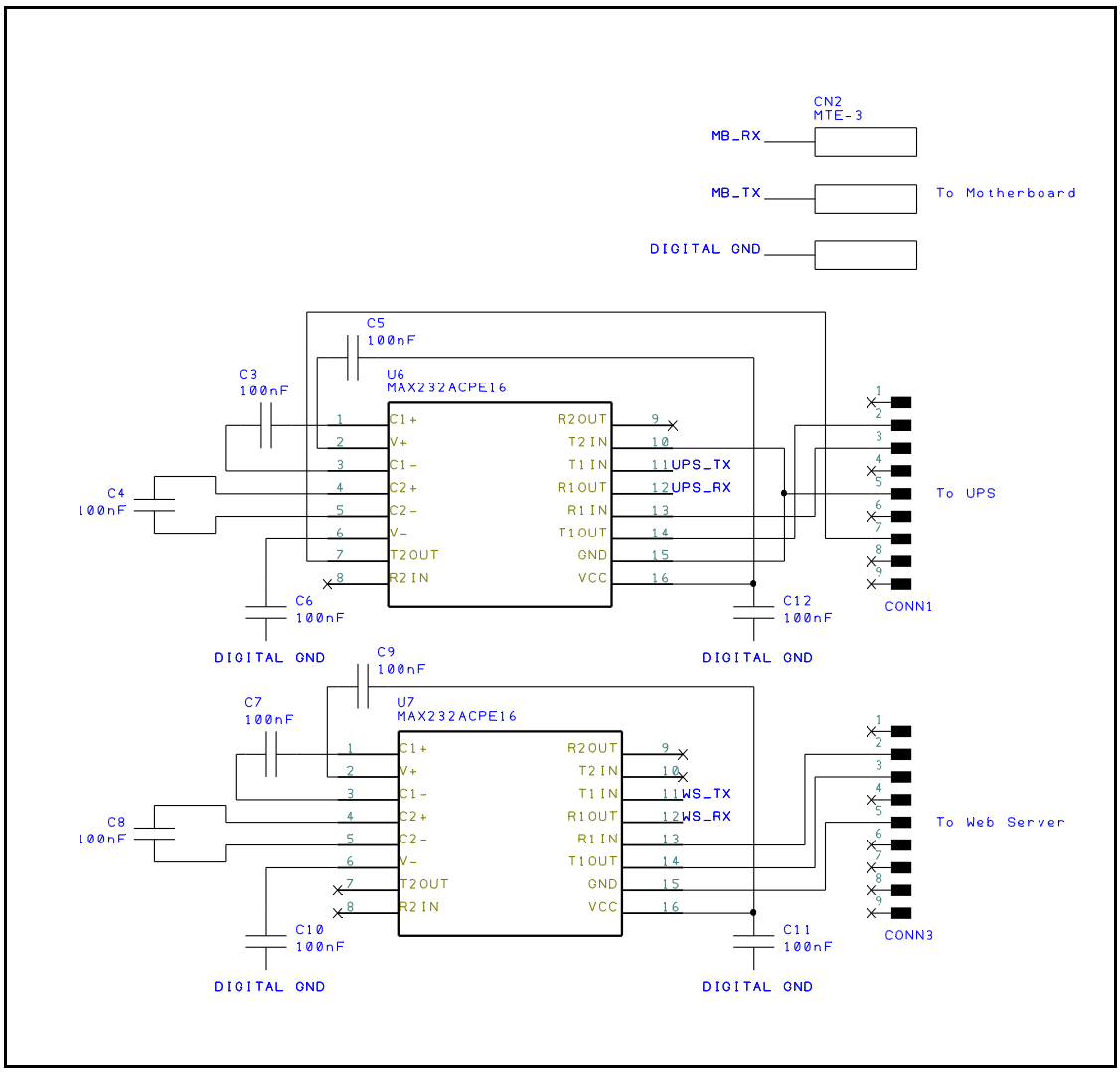


Figure 23: Serial Communications

To implement to UPS serial port, T2IN was tied to GND, resulting in 10Vdc at T2OUT. This was connected to pin 7, to provide power for the UPS serial interface.

During firmware development, discussed further in Chapter 5 – Firmware Design, it was decided that implementing another serial UART on the Motherboard was going to be problematic. It was therefore decided that a better solution to using a logic level UART would be to use the logic level UART

to drive an RS-485 interface (see Figure 12) to communicate with the Motherboard using the existing RS-485 bus. This required two extra pins to control the TX and RX enable pins however, this was not an issue as the Arduino Mega had more than enough spare I/O's to accommodate the change.

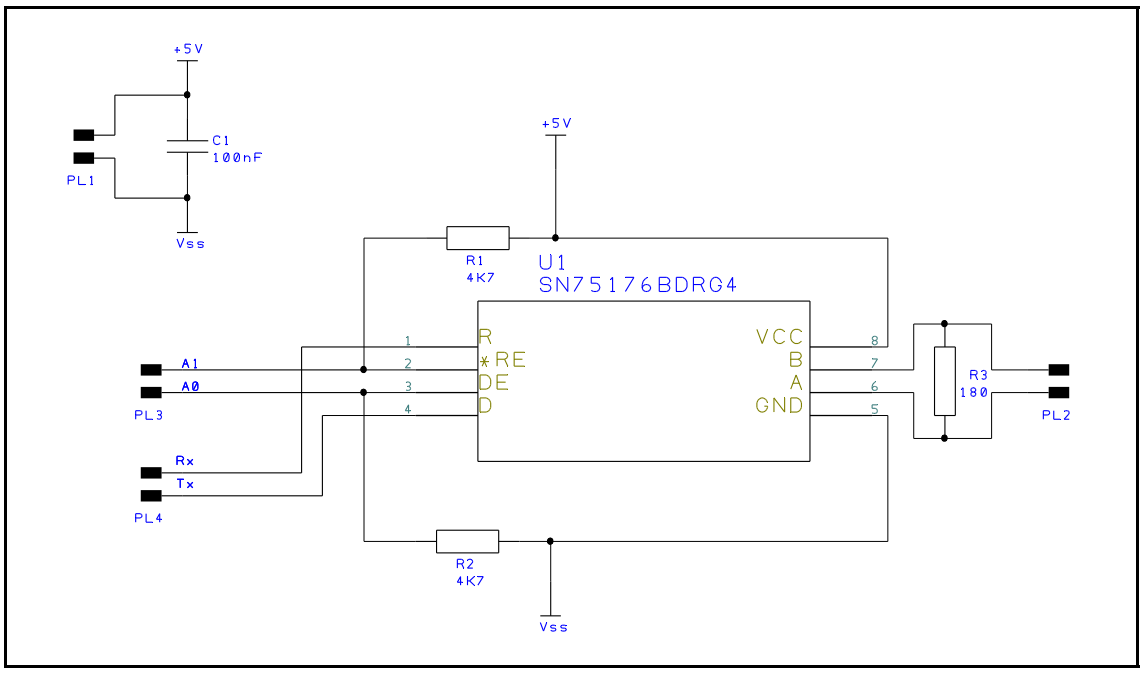


Figure 24: RS-485 Serial Interface

The small board was manufactured single sided (See Figure 25) and attached underneath the Arduino Mega.

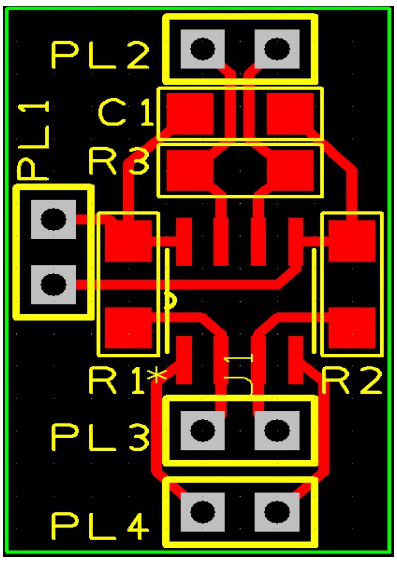


Figure 25: RS-485 Interface PCB

It was also decided at this point that the web server would not be used and therefore this RS-232 UART would no longer be required.

Given that the Power Supply Board was used for controlling the oven heater SSR's, it made sense for the Power Supply Control Board to monitor the temperature of the oven and provide the control signal to the Power Supply Board. The plan was to use the same thermocouple interface as used on the MakerBot Extruder Controller v3.6, however when attempting to source the MAX6675 thermocouple I/C, it was discovered that the device was no longer being manufactured. However, the Maxim Integrated website (http://www.maximintegrated.com) recommended the MAX31855 (Maxim Integrated, 2012) as a pin-pin replacement. The MAX31855, an I^2C cold juntion compensated thermocouple interface was only available in a 'K' type from local suppliers, however, the existing thermocouple in the oven was an E-type thermocouple. The MAX31855-E device was sourced from the Maxim Integrated website as samples. A total of three devices were requested, one for the oven and one for each extruder. Unfortunately, the matching E-Type thermocouple connectors (CN1 in Figure 26) could not be sourced locally and had to be ordered in from Omega in the UK.

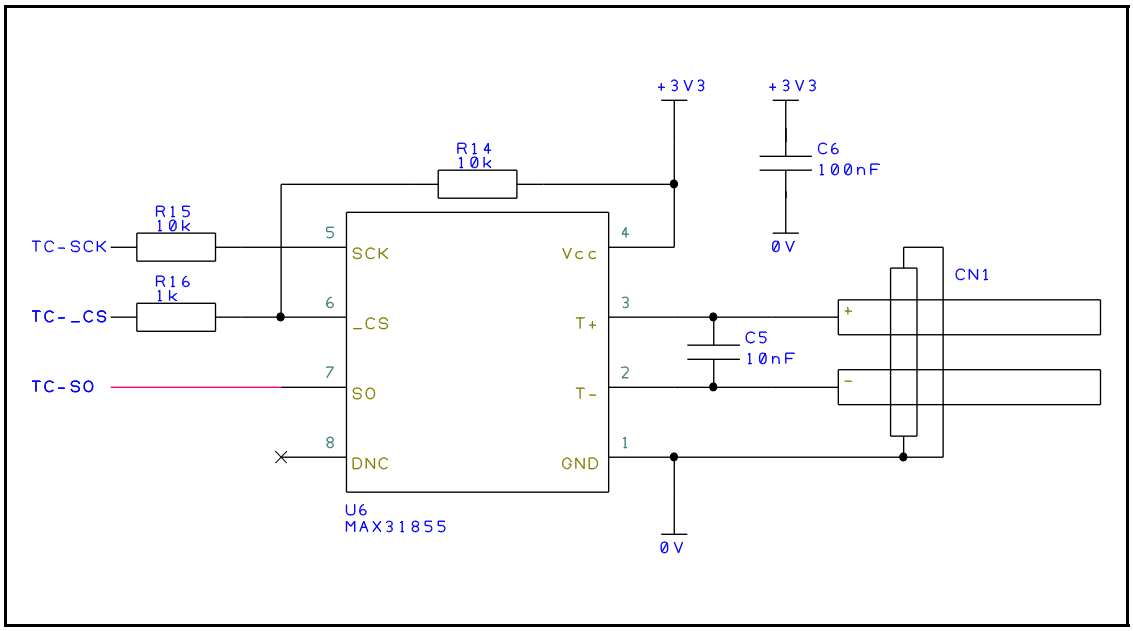


Figure 26: MAX31855 Thermocouple Interface

The Power Supply Control Board was also responsible for the contactor control signal. It was decided that it should be possible to shut off the contactor in an emergency, without having to rely on the Arduino. The E-Stop was implemented using the existing 'normally closed' button on the FPC and also a thermal cut-out connected to the top of the oven. A pair of relays, one normally open the other normally closed, were used, to enable control of the contactor by the Arduino with the E-Stops in series, such that if any one of the controls could open the contactor, isolating the 3 phase from the rest of the machine as shown in Figure 27.

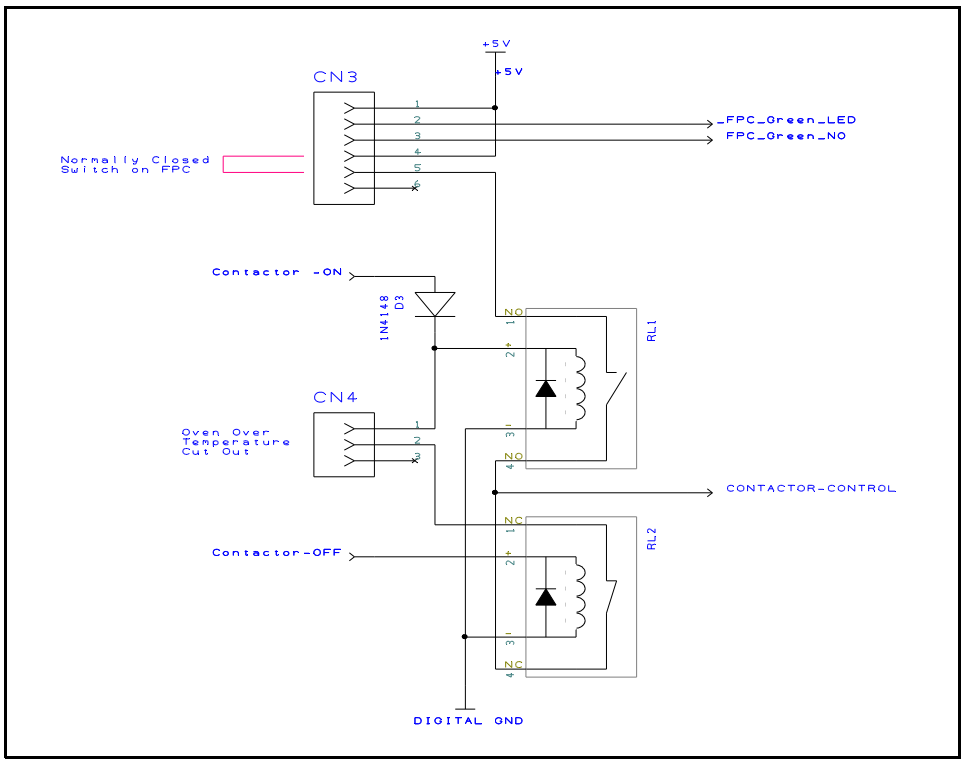


Figure 27: E-Stop Protected Contactor Control

It was also decided that the Arduino should have a method of checking to see if the DC power supplies were on-line. Four DC supplies were monitored, the 24V supply to the Motherboard and Extruders and the three 60V supplies to the Stepping Motor Drives. These supplies were monitored using opto-isolators in the same manner as the outputs from the PoL bricks used to monitor the UPS, L1, L2 and L3.

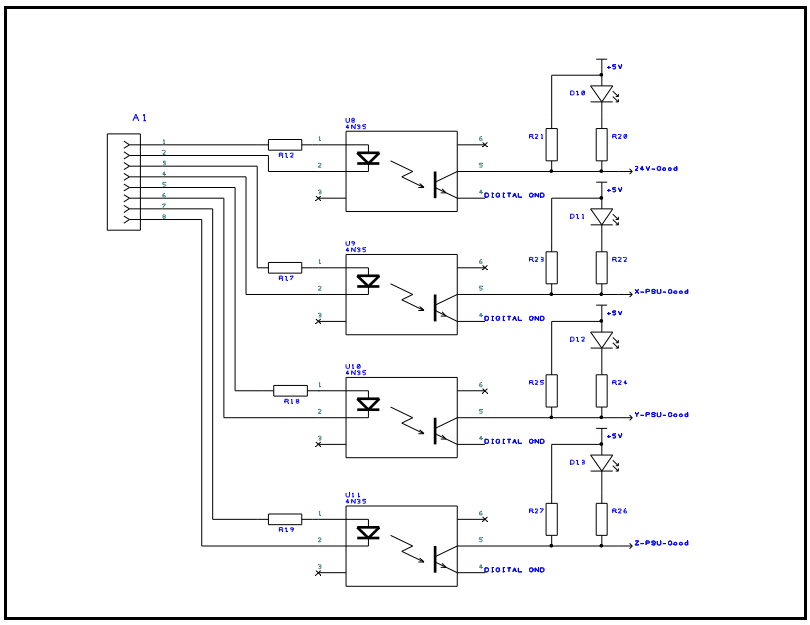


Figure 28: DC Power Rail Monitoring

All of these interfaces were tied together using an Arduino Mega 2560 as described earlier in this chapter.

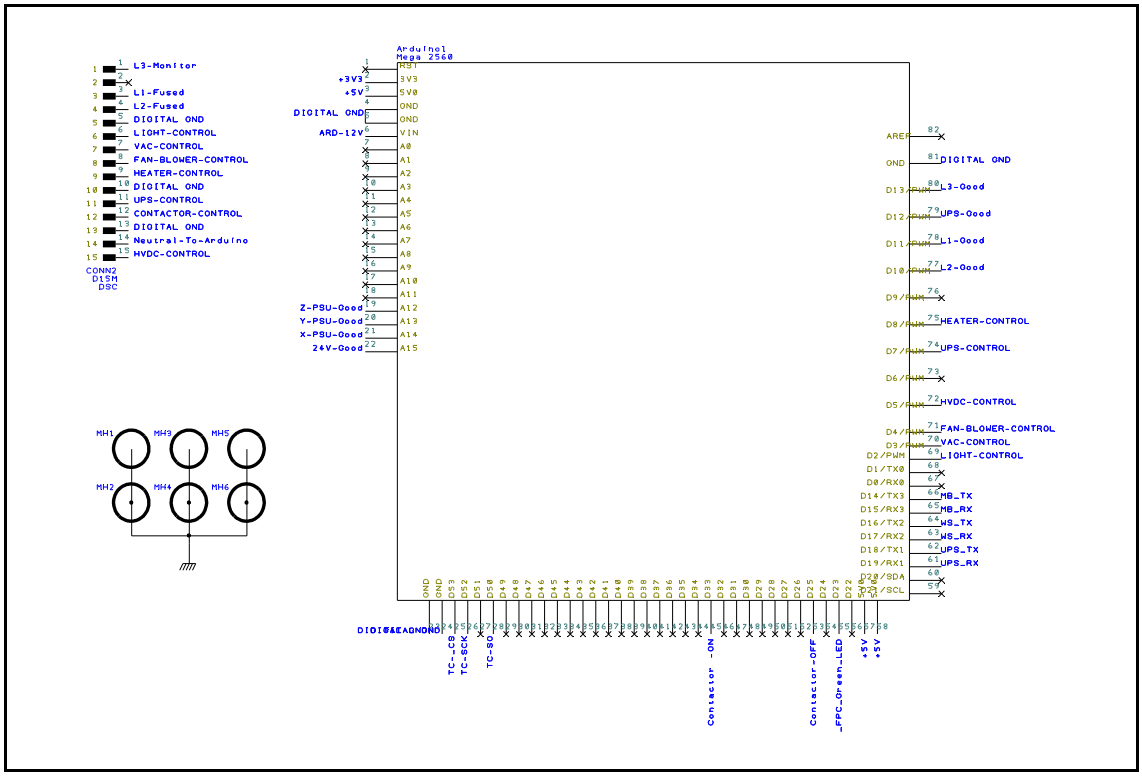


Figure 29: Arduino Mega 2560 Power Supply Controller

The Power Supply Control Board was drawn up in DesignSpark PCB and a PCB generated, see Figures 30 and 31.

During testing a fault was found. A track had been omitted from connector CN3, Pin 3 (high-lighted in blue on Figures 30 and 31) to the Arduino. This connection was for the 'normally open' button on the FPC, which was used for confirming the power up sequence. Without it, the contactor could not be closed. Once the fault had been located, a jumper wire was inserted and the problem resolved.

Figure 32 shows both the Power Supply and the Power Control PCB's fitted to the machine prior to re-fitting the guard cover.

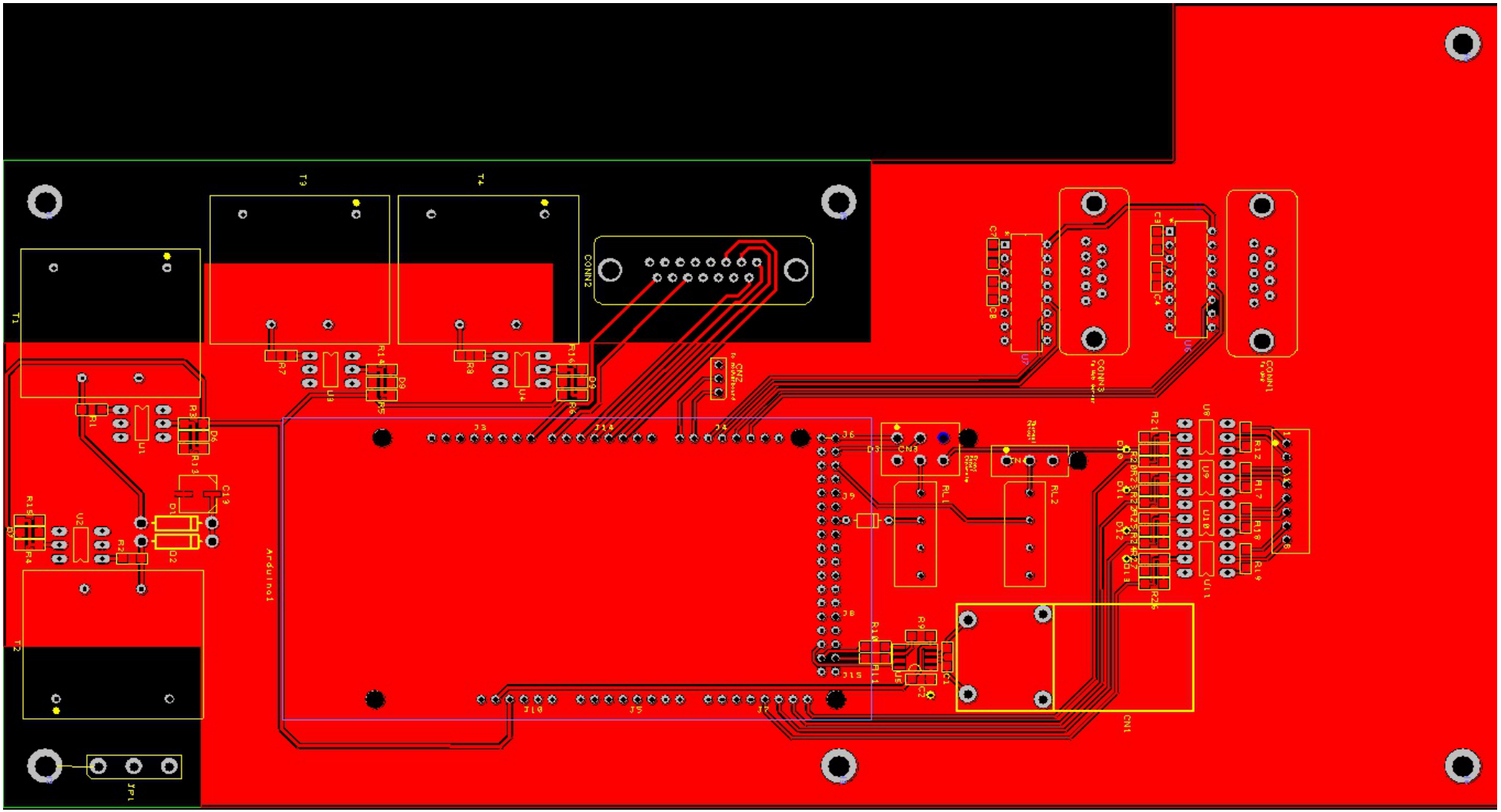


Figure 30: Power Control PCB - Top Copper

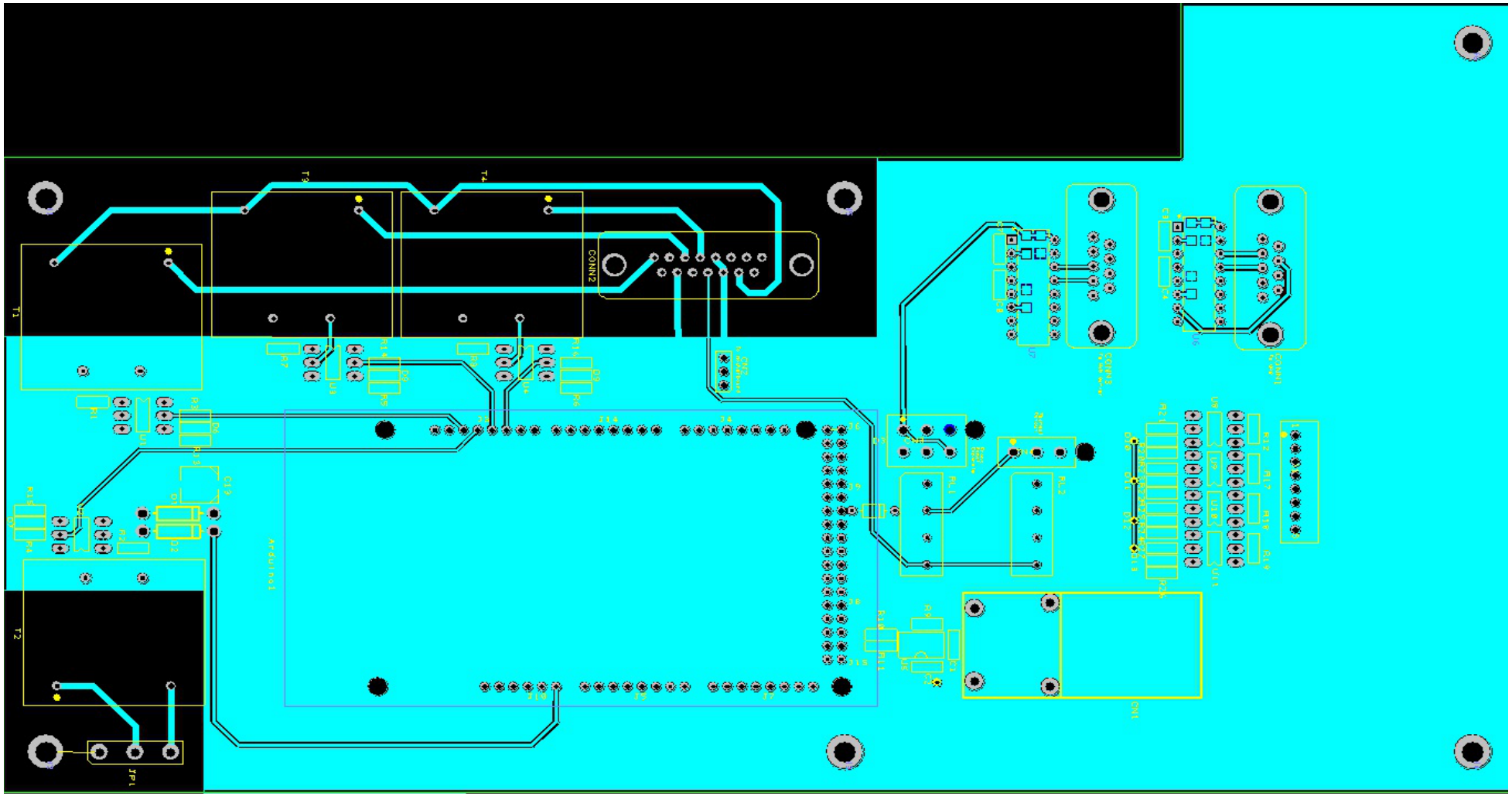


Figure 31: Power Control PCB - Bottom Copper

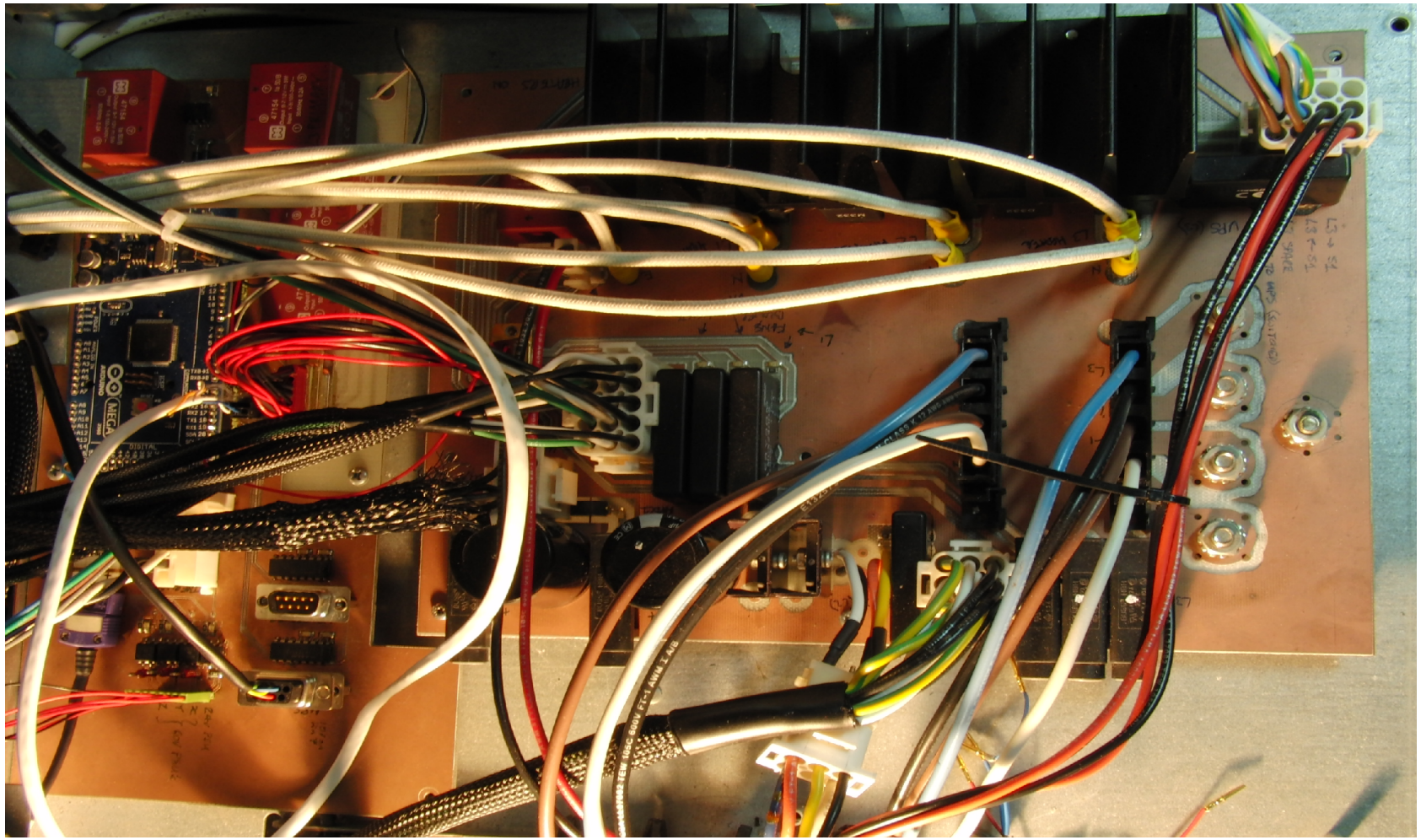


Figure 32: Power Supply & Power Control PCB's Fitted

4.2 Extruder Head

The OEM extruder controller was designed to control both extrudes on a single PCB. It used a PIC micro-controller and a Xilinx FPGA. The heating elements were set up in pairs, connected in parallel. Each element was 100 ohms, giving a total resistance for each extruder of 50 ohms. These were switched using IRF644 MOSFET's. The gate drive used xxxx to interface between the PIC and the MOSFET's. The drain on the MOSFET's were connected to the 150Vdc from the HVDC supply. Each extruder heater would consume 450W. Thermocouples were embedded into the heating elements to monitor the temperature, using Burr-Brown Instrumentation Amplifier devices. The same MOSFET setup was used to drive a 100 ohm solenoid. The solenoid was used to move the secondary extruder down approximately 3mm from its resting position (approximately 1.5mm above the main extruder position). The extruders were driven using Faulhaber 2342L036, 36V DC micromotors (Faulhaber, 2012) attached to the front of each motors is a Faulhaber 23/1 planetary reduction gearbox (Faulhaber, 2011), giving a 66:1 reduction ratio. On the back of each motor is a Faulhaber IE2-512 encoder (Faulhaber, 2012), providing 512 pulses per revolution. The motors are connected to the extruder PCB via JP3 and JP5 with pins 1 & 2 being motor power, pins 3 & 4 being power to the encoder and pins 5 & 6 being the A & B output from the encoder for each.

The initial plan was to design a new extruder board with the same layout as the OEM board. However, the MakerBot v3.6 Extruder Controller (MakerBot Industries, 2010) upon which the design was based was a single extruder controller. It was decided, for modularity, that it would be better to design individual extruder boards as 'daughter boards' and have a separate 'backplane' on which they would be fitted. The 'backplane' would also house the HVDC components, rather than having the HDVC on the same board as the micro-controllers.

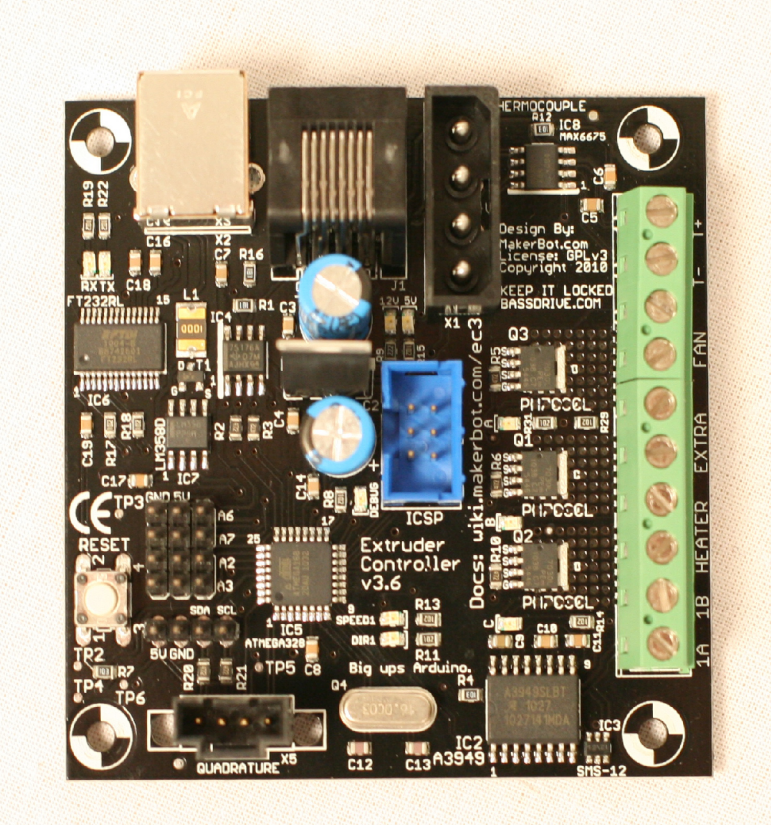


Figure 33: MakerBot Extruder Controller v3.6

The extruder controller is built around an ATMega328P running at 16MHz, using the same settings as the Arduino Uno, with a number of peripherals attached to perform the functions required.

Figure 34: ATMega328P Micro-controller

Although the plan was to load the firmware on the micro using the in-circuit serial port (ICSP), it was decided that for initial testing purposes, the Arduino Uno bootloader would be uploaded to the micro and the USB port used for debugging the hardware prior to starting on the MakerBot firmware.

The Arduino Uno was designed to be programmed over the USB interface using an FTDI I/C. The FT232RL (Future Technology Devices International Ltd, 2012) required no external crystal and very little external circuitry to convert USB to a TTL level UART interface. This was connected directly to the TX/RX pins on the ATMega328P. LED's connected to the GPIO 0&1 pins on the FT232RL provided a visual indication of transmit and receive data.

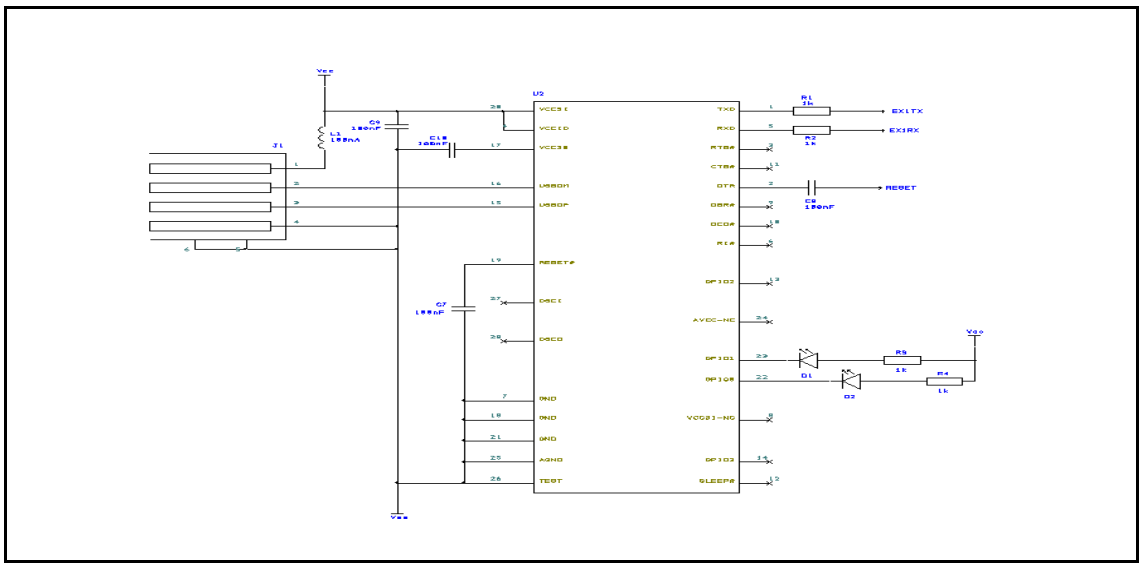


Figure 35: USB Interface to ATMega328P Micro-controller

With hindsight, the USB interface could have been implemented on a separate board and a cable connected to a header on the extruder board to connect it when necessary. This would have saved board space and complexity.

As discussed previously, the MakerBot is designed for a K-Type thermocouple, however, the thermocouples already used in the OEM machine are E-Type thermocouples. It made sense to re-use the existing thermocouples rather than having to source alternatives. 'E' type thermocouple interface IC's were

sourced from the Maxim Integrated website as samples. The only difference between the MAX6675 and the MAX31855-E was the output resolution. The MAX6675 was 12-bit plus sign, whereas the MAX31855-E was 14-bit including sign. The result was an extra bit which had to be accounted for in the firmware, discussed later in Chapter 5 – Firmware Design.

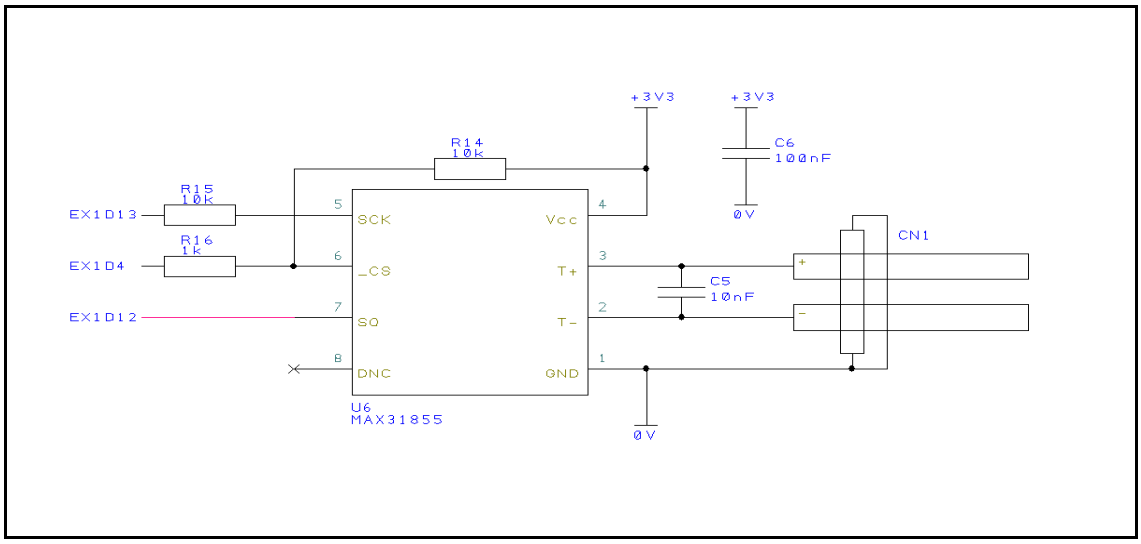


Figure 36: MAX31855 Thermocouple Interface

Serial communications between the extruder and the motherboard were done using an RS-485 multi-drop bus, which allowed a single host (the motherboard) to talk to multiple slave devices (extruders and power supply). RS-485 is a half-duplex differential protocol that requires only two wires. The SN75176 I/C is an RS-422/485 bus driver used to convert the logic-level serial data stream from the micro-controller to the line levels required for the RS-485 bus. It also used two additional pins which enable/disable the transmit and receive. The *RE (receive enable) pin was active low. When pulled down, it allows the micro-controller to 'listen' to the data stream on the RS-485 bus. The DE (data/transmit enable) pin was active high. When pulled high, data arriving at the D pin is sent out to the bus. When the *RE pin is held high and the DE pin is held low, the output of the driver is high impedance so other devices can continue to use the bus without interference.

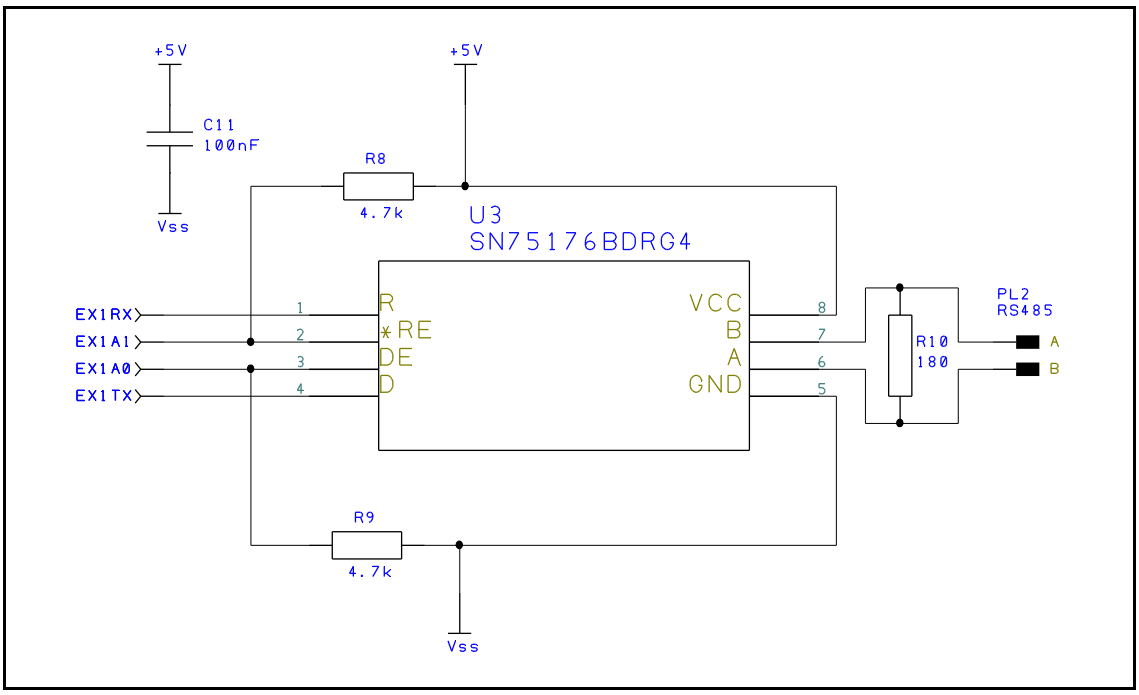


Figure 37: RS-485 Serial Interface

Since MakerBot has gone down the route of using stepping motors for powering their extruders, the DC motor controller implemented on the Extruder Controller v3.6 was re-used to control the speed of the cooling fan attached to the extruder. However, the OEM motors on the FDM Vantage extruder head were DC motors and perfectly suited to the design of the extruder head, so it was decided to re-use the existing motors rather than look for replacement stepping motors to provide the same function. The solution was to use the DC motor controller for its original purpose (driving the extruder). However, the A3949 used in the original MakerBot design is not suitable for driving the 36V Faulhaber motors. On the Allegro website the A3950 (Allegro Microsystems LLC, 2012) was found as a pin-pin replacement for the higher voltage motors, up to 36V 2.8A. The only change required was the working voltage of the capacitors used on the motor control side and the transient voltage suppressor (TVS diodes) attached across the motor outputs.

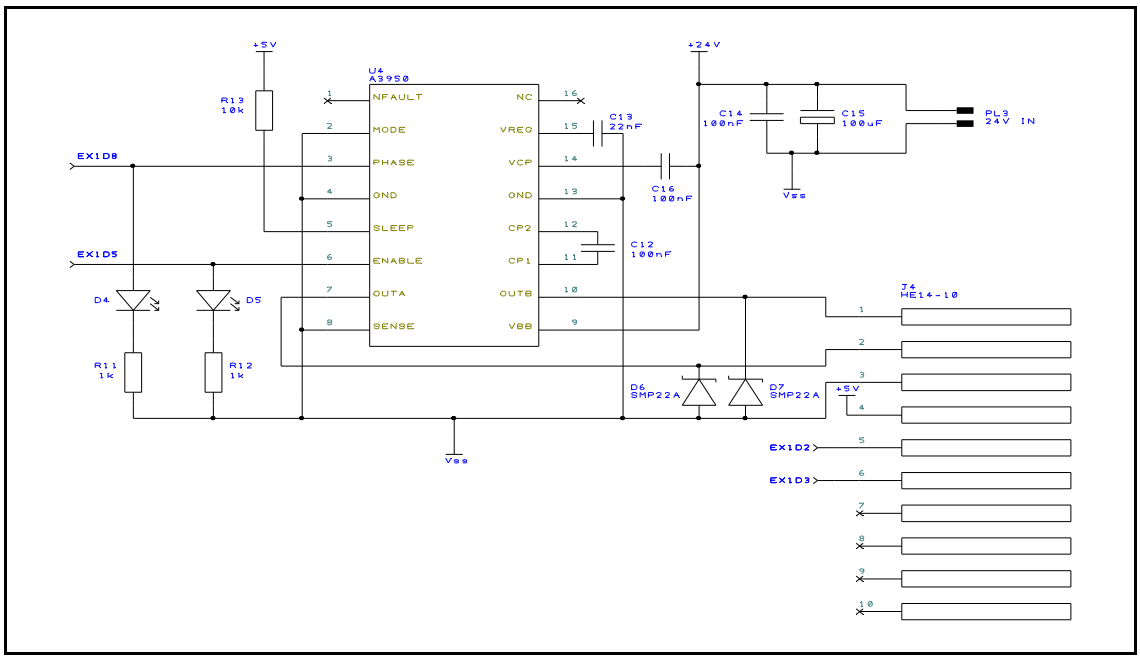


Figure 38: A3950 Full-Bridge DC Motor Control

The supply voltage for the extruder controller was taken from the 24VDC supply in the base of the machine. The 24V was used to the supply the DC motor controller and also regulated using a Murata 7805SR-C. This was a switch-mode version of the standard 7805 linear regulator, able to operate at input voltages up to 36V without the thermal losses associated with linear regulators. The MAX31855 is designed to work at 3.3V rather than 5V, so a 3.3V regulator (LD39015M33R) had to be included on the board, connected to the 5V rail. A comparator was used to automatically select between the 5V from the USB supply and the output of the 7805SR-C if available. This made testing easy as I was able to power all but the DC motor controller from the USB for development. LEDs were attached to each of the supply rails for visual indication of supply status.

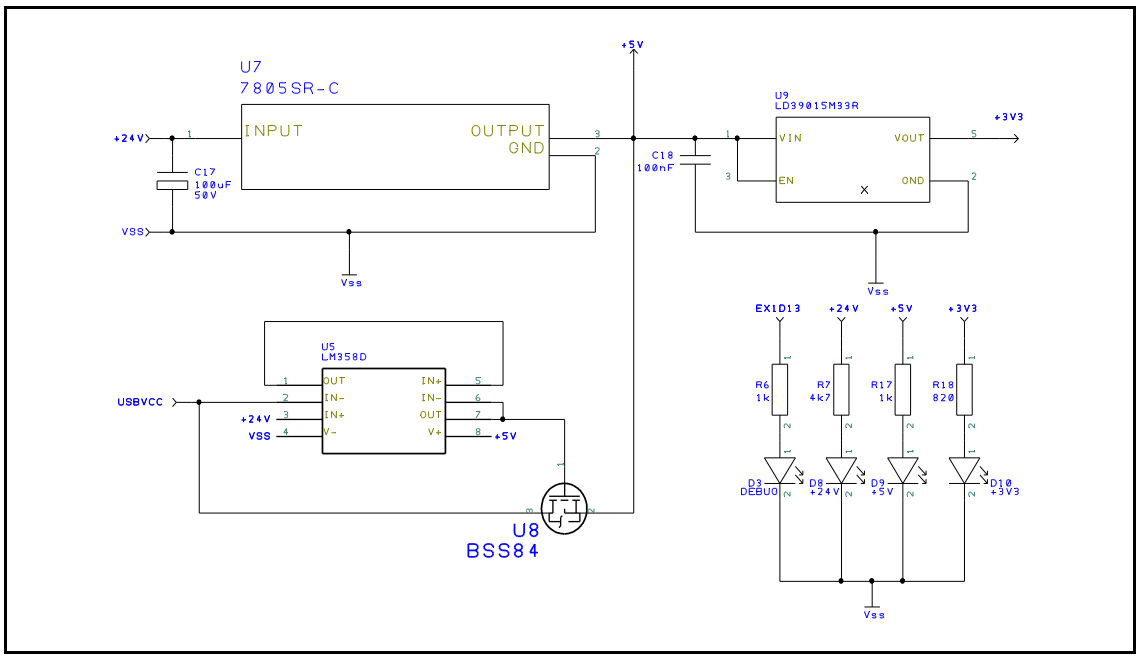


Figure 39: Extruder Power Supply

Due to the accuracy required by the complexity and number of via's etc, it was decided that rather than manufacturing the extruder PCB's on the board mill at DTA, the boards would be manufactured locally by PCBZone (http://www.pcbzone.net/). The boards were four layer boards, two signal layers (top and bottom), a ground plane and a 5v plane as shown in Figures 40-43.

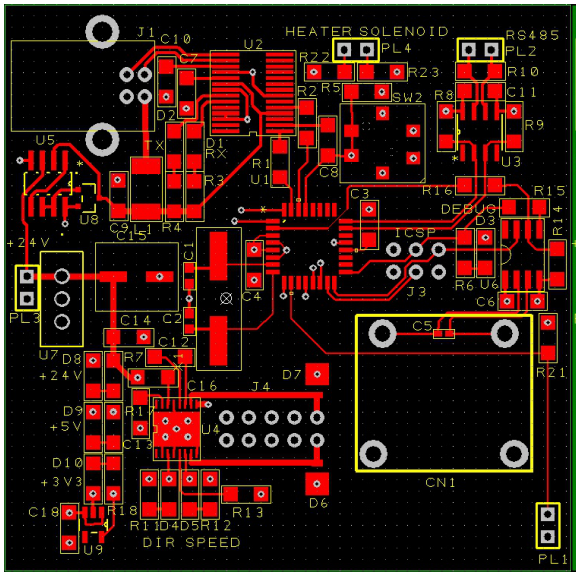


Figure 40: Extruder Controller Top Copper

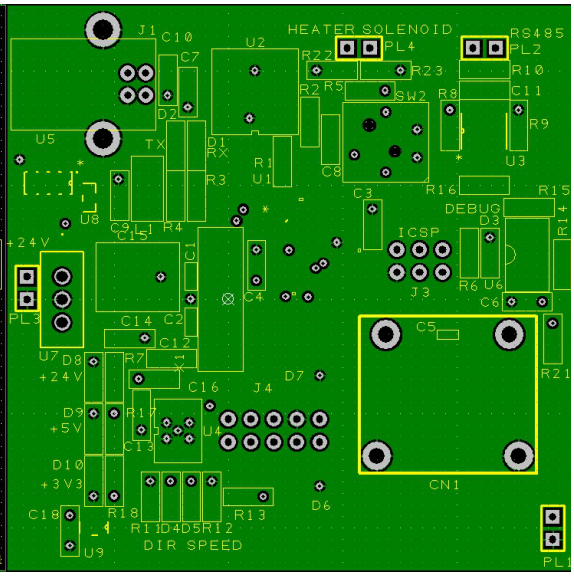


Figure 41: Extruder Controller Gnd Plane

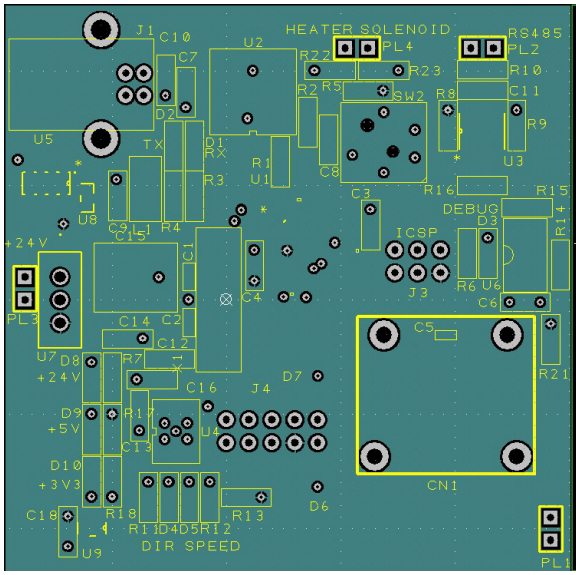


Figure 42: Extruder Controller Vcc +5v Plane

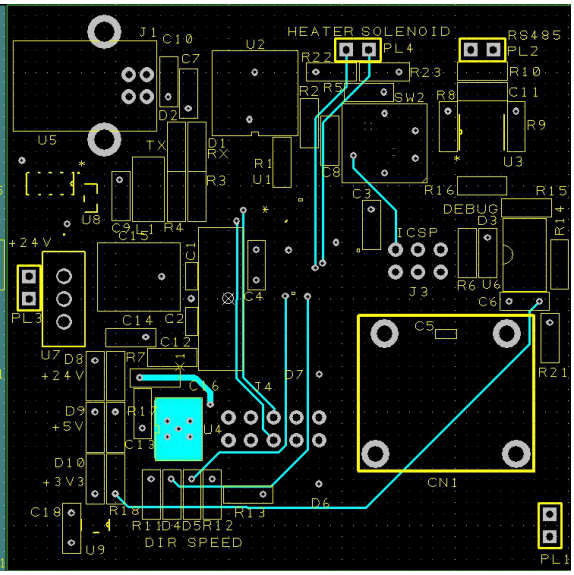


Figure 43: Extruder Controller Bottom Copper

A number of issues were discovered with the initial design after the boards had been manufactured and assembled. When the USB cable was connected, the board powered up as expected, however the Arduino IDE was unable to connect to the micro-controller. After some investigation, it was determined that

during the design process, the TX and RX lines had been transposed between the FT232RL and the ATMega328P. This was easily solved by cutting the two tracks and linking across them with jumper wires. Testing could then be conducted using the Arduino IDE and simple sketches to test each individual part of the design. No issues were found with thermocouple interface or the DC motor controller. However, when it came to testing the RS-485 interface, it was discovered that somehow, the TX and RX lines between the ATMega328P and the SN75176 had been omitted altogether. This was put down to inexperience in designing boards using multiple sheets with DesignSpark. Again, this problem was resolved with the addition of two jumper wires.

As mentioned previously, it was decided to mount two individual extruder controllers onto a backplane. The outputs from the extruder controllers switch the high voltage DC to the extruder heaters and the solenoid used to select between the extruders.

By searching for the part number of the 34-way, right angle connector (AMP 213289-1) on the OEM extruder board, I was able to source a sample part from TE Connectivity, which not only saved me purchasing a suitable connector, but also enabled me to re-used some of the wiring loom from the bottom of the machine up to the extruder head. Table 9 shows the pin-out used.

Signal	Pin Designation
HVDC1	LL
HVDC1-Return	DD
HVDC2	JJ
HVDC2-Return	NN
24VDC	C
0V	A
RS-485A	B
RS-485B	F

Table 9: Extruder Head Pin-Out

The initial design for the backplane switching is shown in Figure 44.

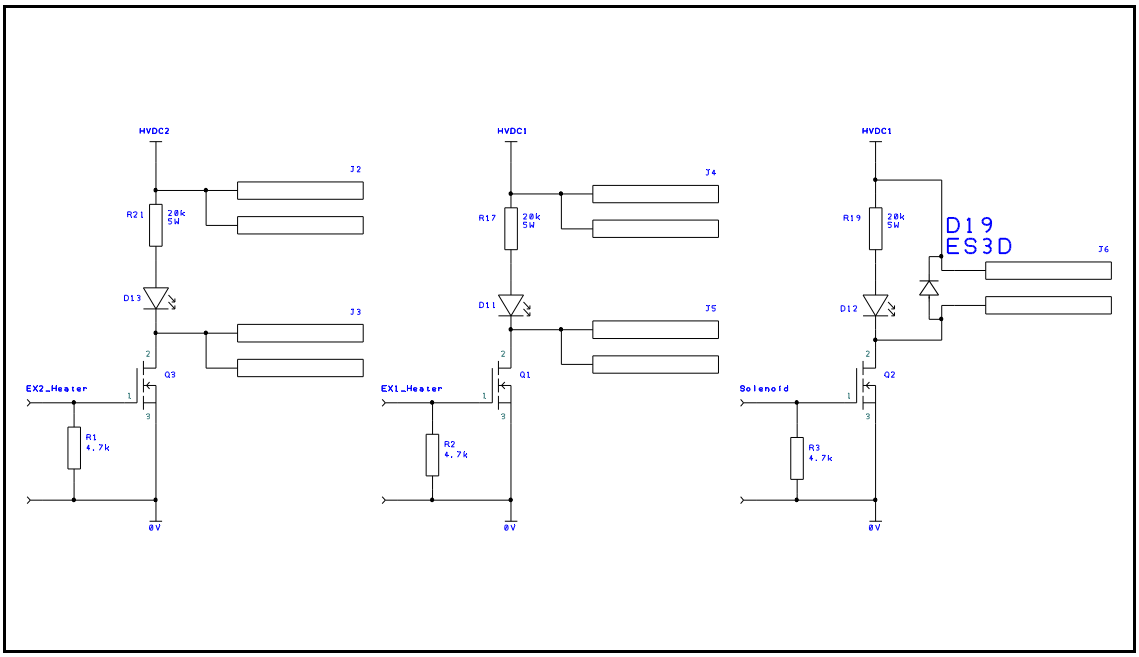


Figure 44: HVDC Switching on Backplane

As can be seen in Figure 44, the MOSFET's used to switch the HVDC were low-side switched, following lessons learned from the development of the Power Supply Board.

As the backplane was a relatively simple board, it was again machined on the board mill at DTA as shown in Figure 45.

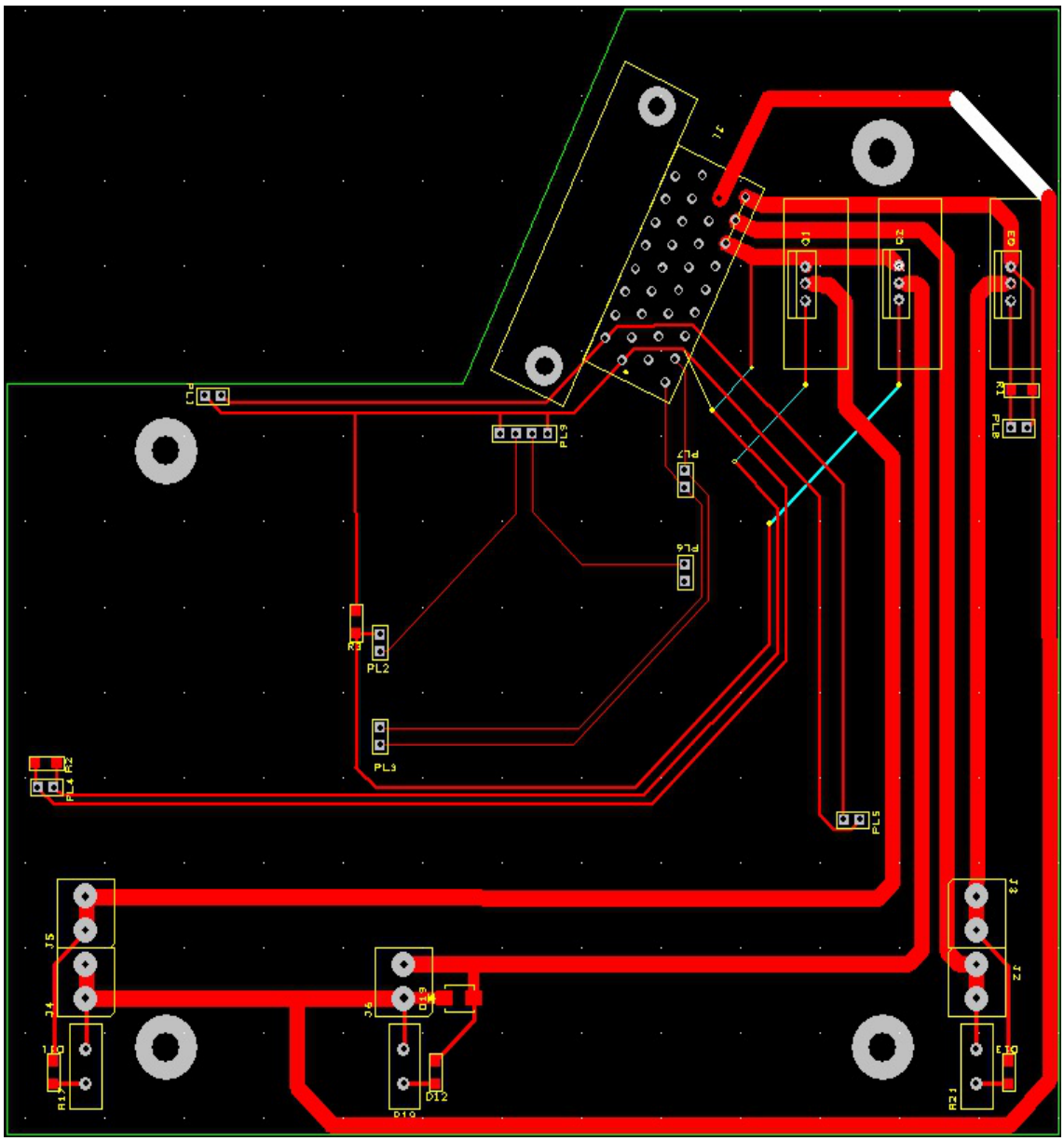


Figure 45: HVDC Backplane PCB Layout

The board was assembled with the two extruder controllers as shown in Figure 46 over the page.

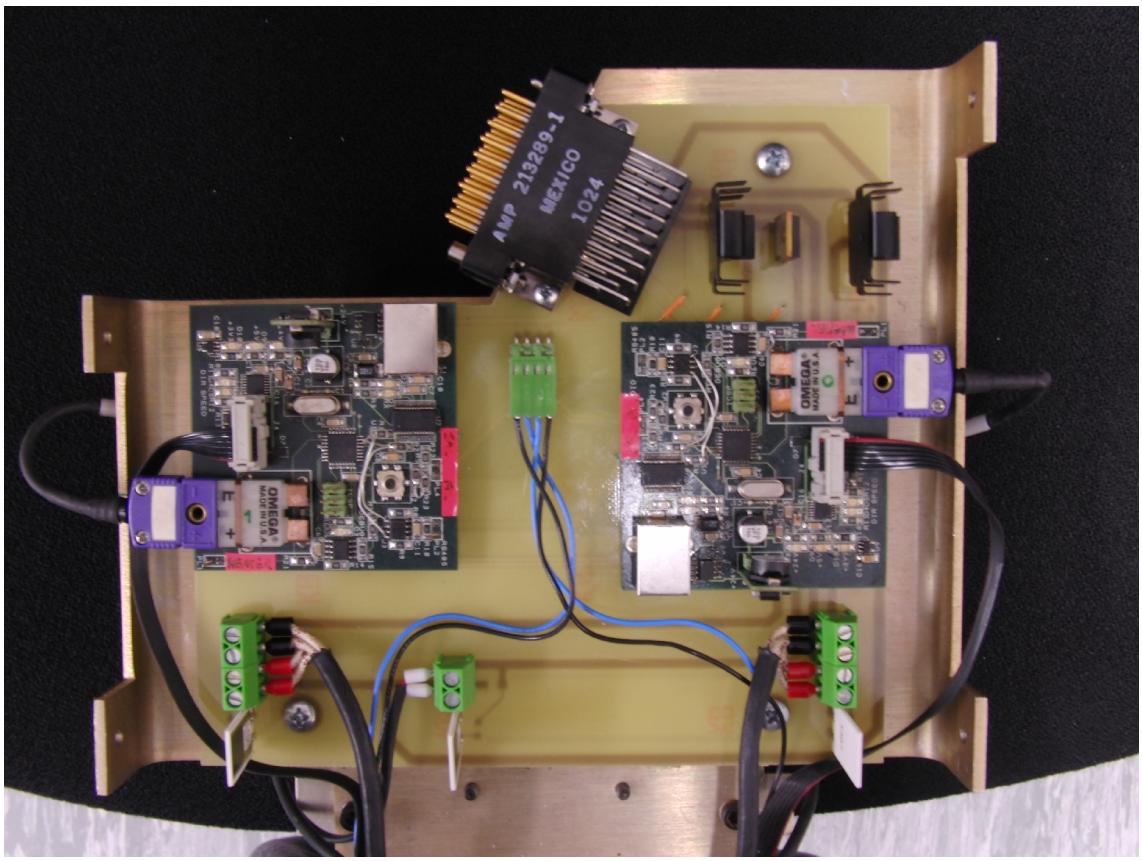


Figure 46: MotherBoard with Extruder Daughter Boards Fitted

The decision to manufacture separate extruder controller boards proved a fortuitous decision as during testing, a fault with one of the MOSFET's used for switching the heaters and solenoid caused 150Vdc to be connected directly to the ATMega328P which wrote the board off. If the design had been a monolithic board, both extruders would have been destroyed along with the switching fabric, resulting in a very expensive repair. Fortunately, only one extruder controller was destroyed and was replaced with minimal impact.

Additonally, during the process, the solenoid was left connected to the 150Vdc causing it to be overheated. On investigation, it was discovered that the solenoid was only rated for 48Vdc continous operation (G. W. Lisk Company, 2011). The possibility of powering the solenoid from a 48Vdc supply was tested. However, although 48Vdc was sufficient to hold the extruder in place once actuated, it was not sufficient the overcome the spring return mechanism.

150Vdc was therefore needed for initial the initial actuation. A method was investigated (Storr, 2014) for reducing the voltage across the solenoid after initial actuation. By placing a power resistor in series with the solenoid with an additional FET across the terminals, the resistor could be switched in and out enabling 150Vdc to be used to actuate the solenoid by shorting the resistor with the FET, then opening the FET to put the resistor in circuit, reducing the voltage across the solenoid to 48Vdc. A monostable (SN74LS121) was used to generate a short pulse for actuating the solenoid whilst a second FET was used to open the circuit, switching the solenoid off.

Having had an extruder controller destroyed, it was decided that the design should be modified to isolate the 150Vdc from the extruder controllers completely using opto-isolators. The resulting circuit is shown in Figure 47.

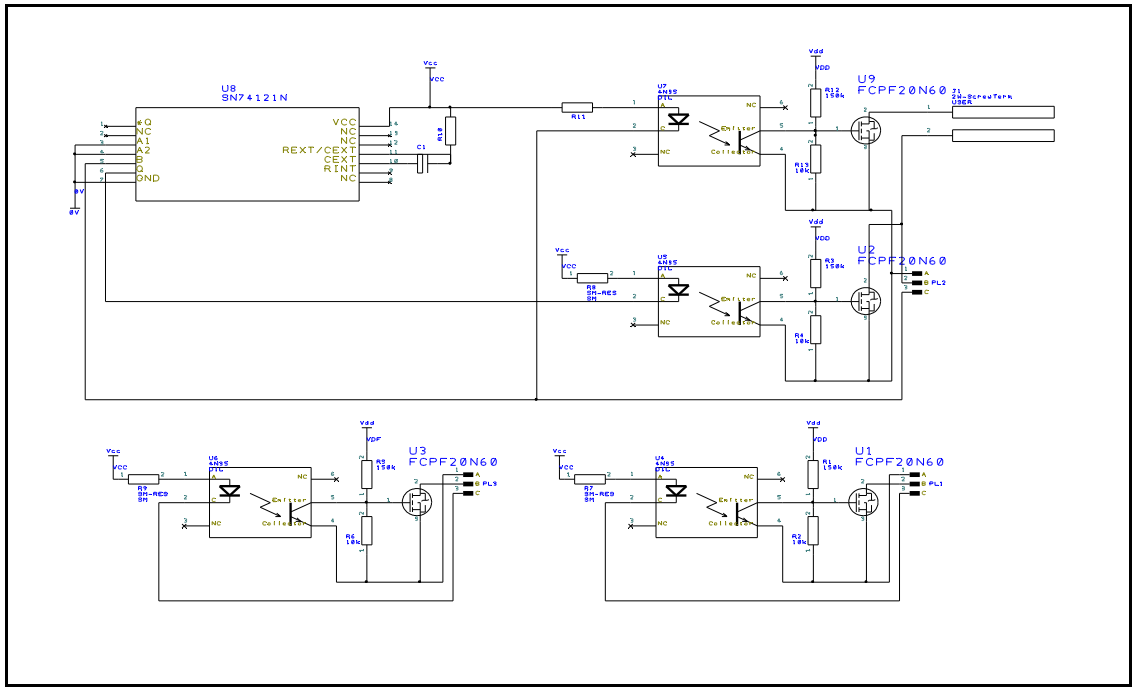


Figure 47: Opto-Isolator / Solenoid Voltage Switching Circuit

Rather than redesigning the backplane, it was decided that the opto-isolator board should be designed to pick-up on the existing FET connections. Power for the input of the opto-isolators and the monostable were taken from the ICSP header on an extruder controller. Additionally tracks were cut on the backplane

to isolate the digital 0V from the HVDC 0V. The result was the board shown in Figure 48.

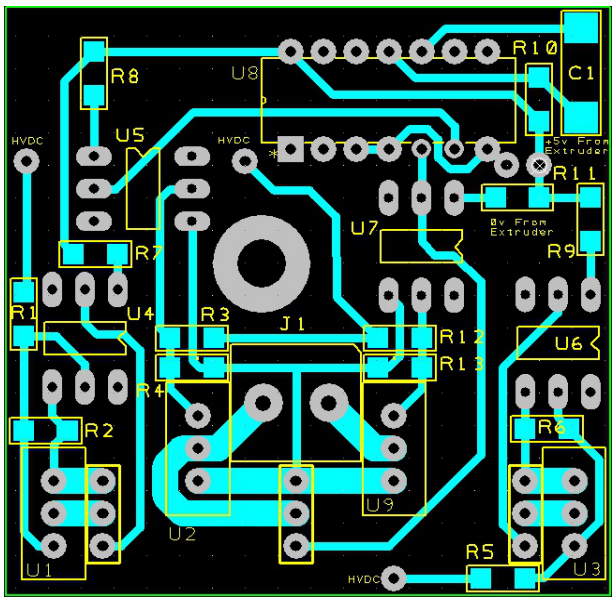


Figure 48: Opto-Isolator / Solenoid Voltage Switching Board

The board was fitted to the Extruder Motherboard as shown in Figure 49.

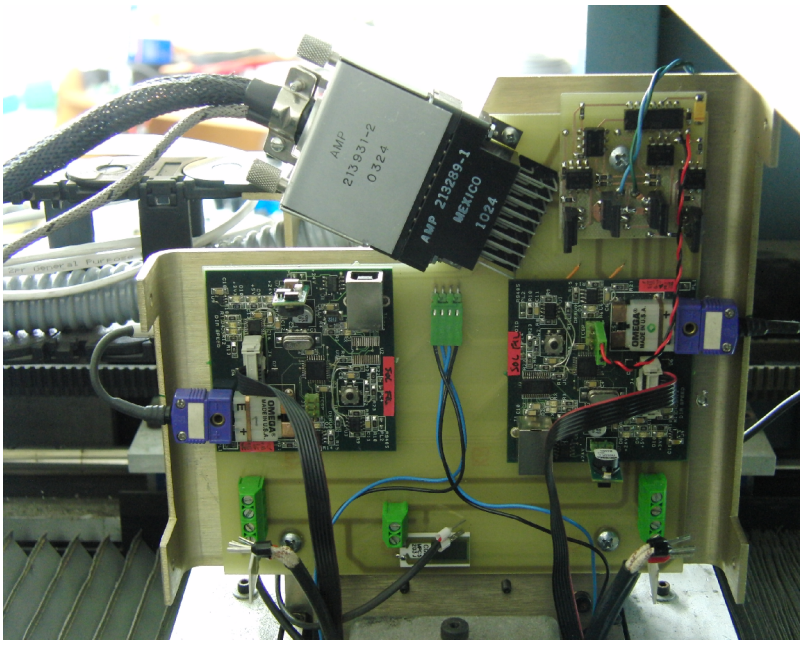


Figure 49: Extruder Motherboard with modified Opto-Isolators and Solenoid Switching

4.3 Motherboard

The Motherboard design was based around the MakerBot Motherboard v2.4 (MakerBot Industries, 2010). The basic MakerBot Thing-O-Matic, upon which most of this work is based, has no user interface. Control of the printer is all done through the PC using the ReplicatorG software, discussed further in Chapter 6 – Software Design. However, MakerBot also developed an optional external interface (MakerBot Industries, 2011), to allow the machine to be controlled without the need for a PC.

The initial plan was to locate the Motherboard in the base of the machine in place of the OEM I/O PCB. However, during the re-design of the FPC discussed previously in Chapter 3, it became evident that it would make far more sense to locate the Motherboard directly on the back of the FPC, this allowed easy access the SD card and minimised the amount of wiring needed between the FPC and the Motherboard.

The FPC/Motherboard combination required three PCB's to be developed. One for the keypad interface, the Motherboard itself, and a third board to allow the SD socket to be positioned such that a card could be inserted into the FPC.

As with the original MakerBot design, the Motherboard was designed around an Arduino Mega 2560. However, a number of modifications were required to the design to allow it to be used. There were a pair of LED's, 'FOO' and 'BAR' which were essentially surplus to requirement. By removing them from the design, an extra button could be implemented on the FPC, this allowed the lights inside the oven chamber to be controlled. The MakerBot designed interface was based around a 4x16 LCD display however, the Stratasys FPC was designed around a 4x40 LCD. An equivalent LCD display was found on the Element14 website (Midas Components Ltd, 2011) based on the same Hitatchi HD44870 (Hitatchi, 1999) chipset used by the MakerBot 4x16 display. To drive the much larger screen, an extra I/O was required to drive a second 'Enable' line.

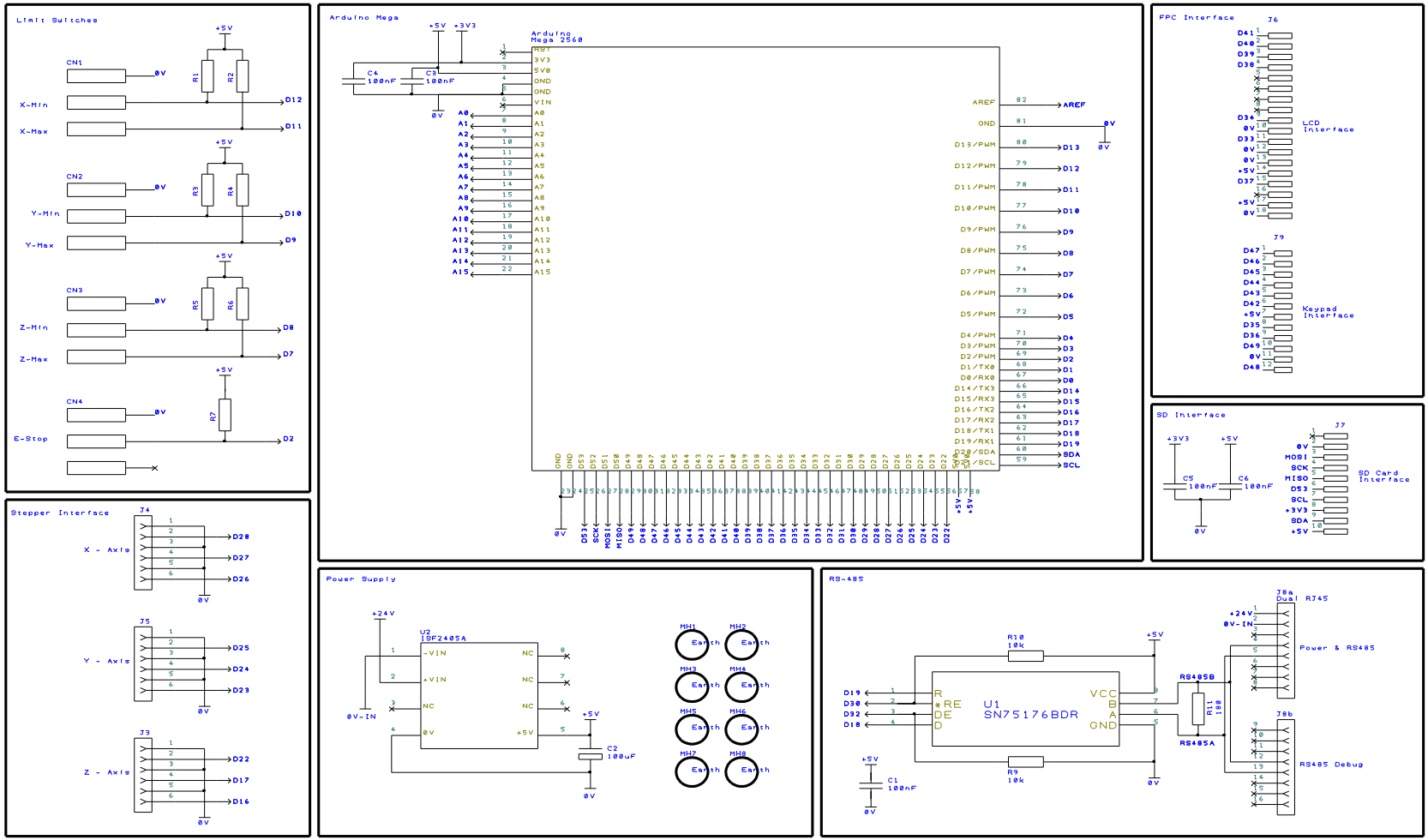


Figure 50: Motherboard Schematic Design

The design was manufactured on the board mill at DTA, see Figures 51 and 52.

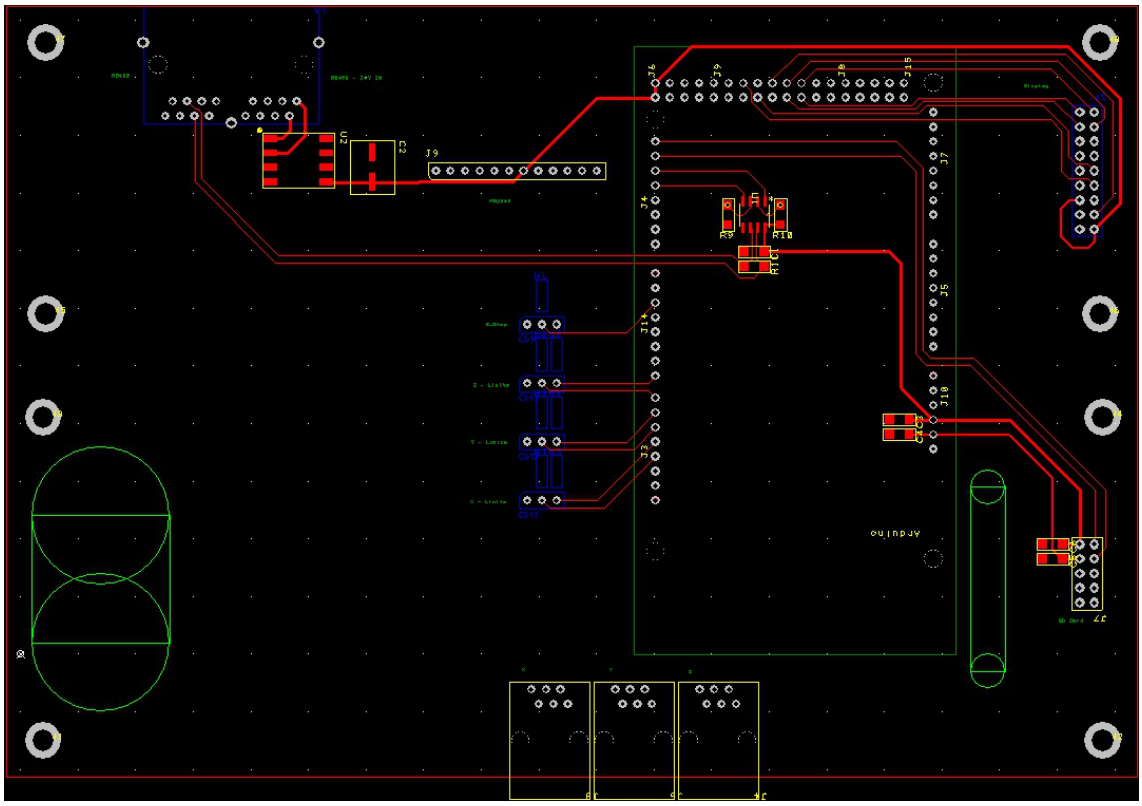


Figure 51: Motherboard - Top Copper

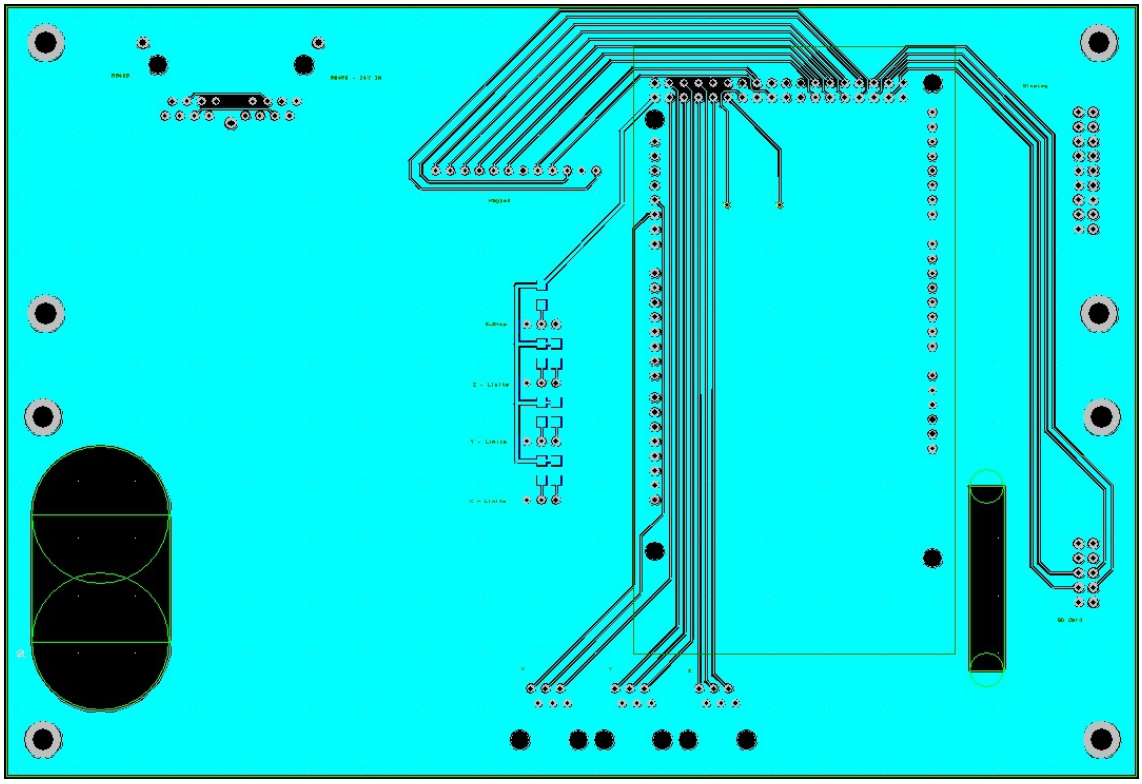


Figure 52: Motherboard - Bottom Copper

The keypad was very simple in design. Each button was connected via a pull-up resistor to an individual digital input on the Arduino.

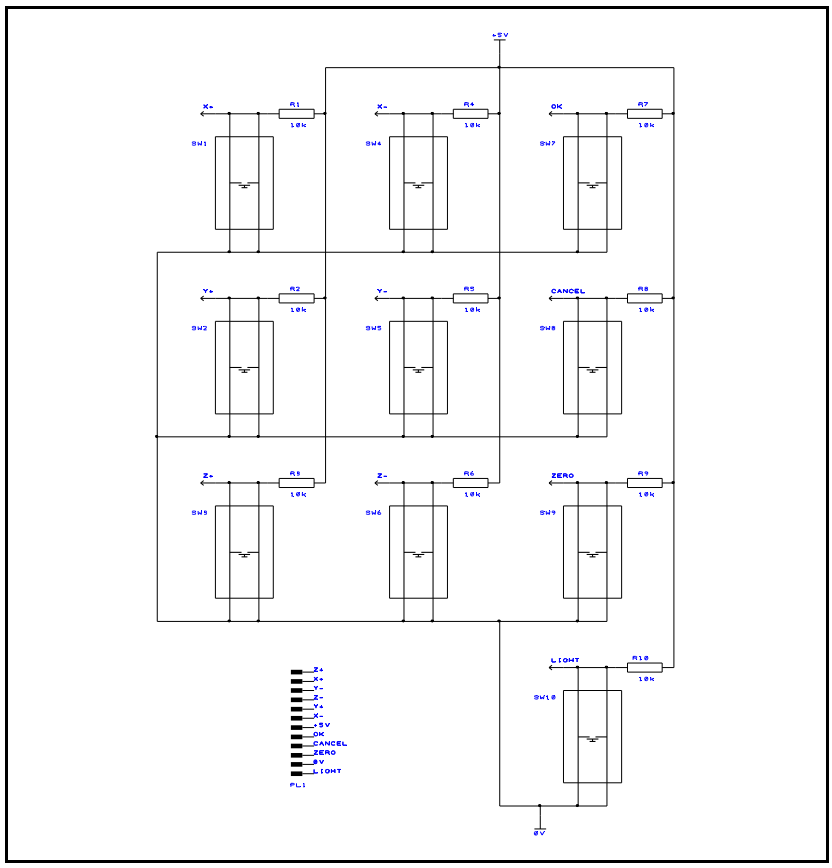


Figure 53: FPC Keypad Interface

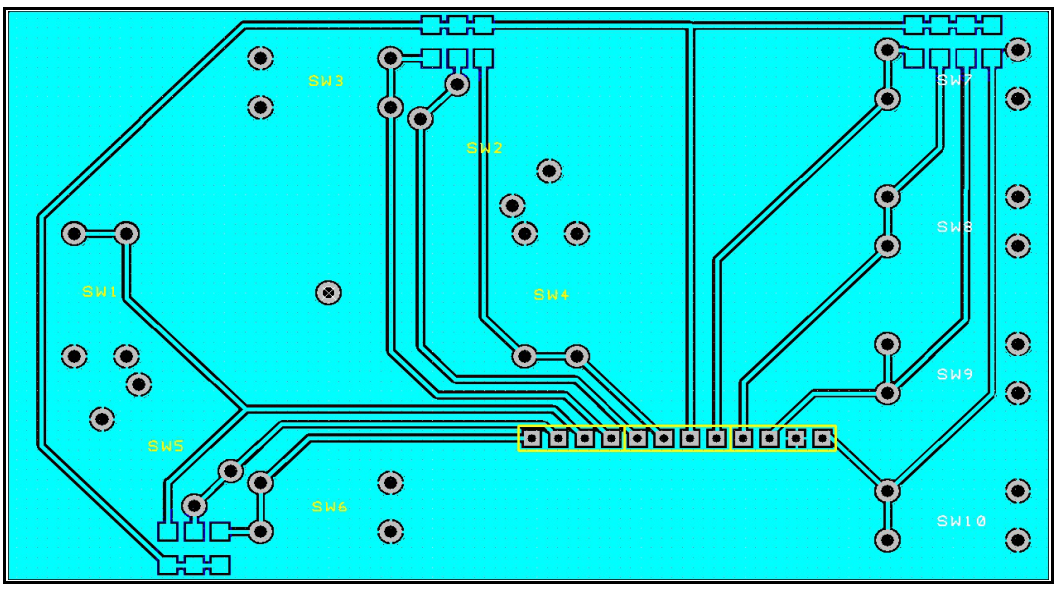


Figure 54: FPC Keypad PCB

The third board to make up the FPC/Motherboard unit was the SD Card interface.

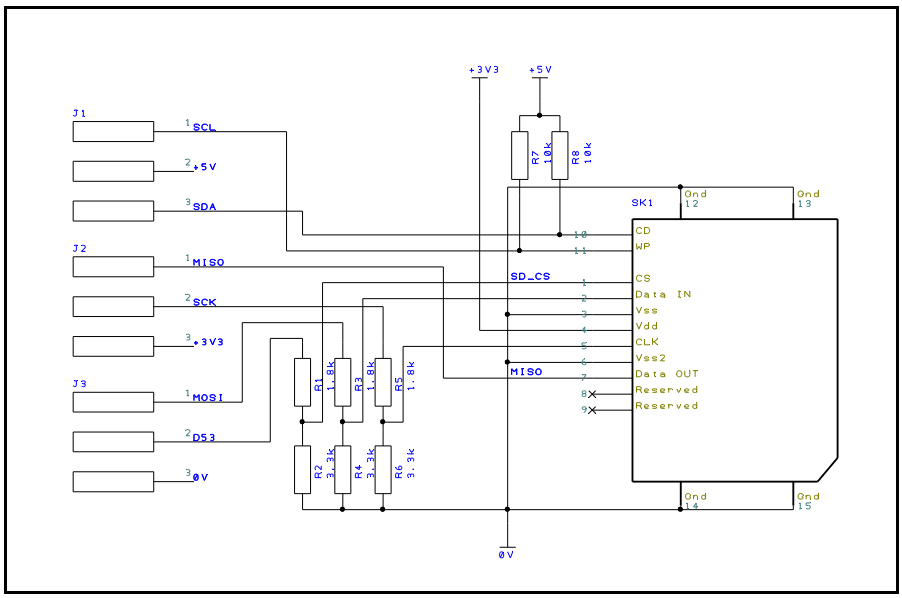


Figure 55: SD Card Interface

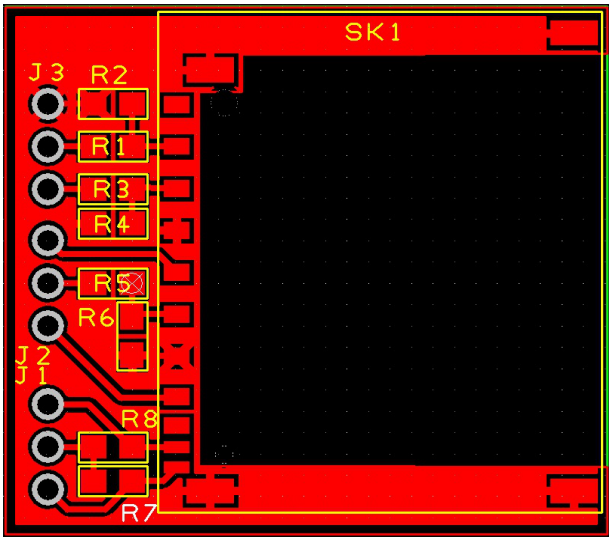


Figure 56: SD Card PCB

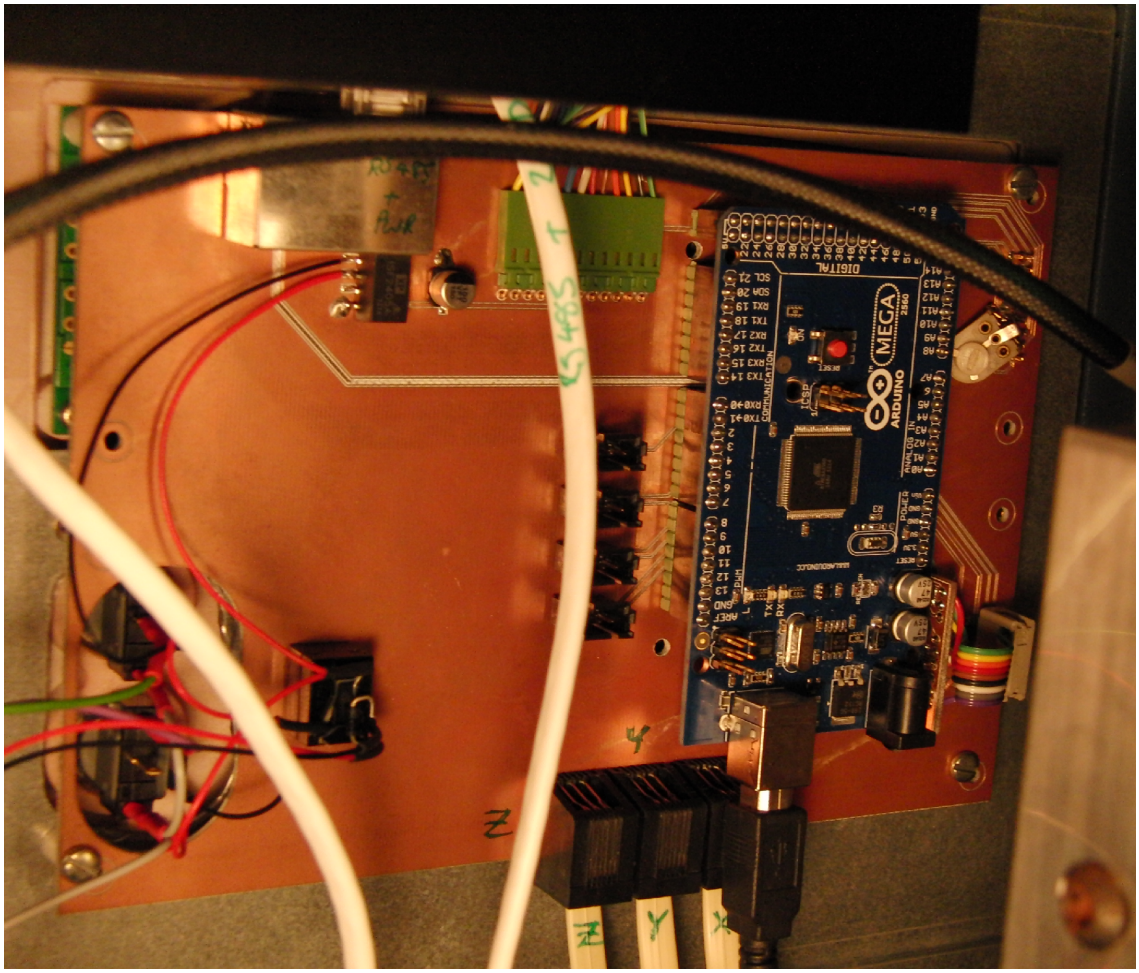


Figure 57: Motherboard fitted to back of FCP

During testing a couple of modifications were made to the Motherboard. A potentiometer was added to the contrast pin on the LCD to allow the screen to be set to a suitable contrast level. Also, a relay was attached to the back of the board. Originally, it was planned that the green and red pushbuttons on the FPC would be LED lit, the assumption was made that suitable switches would be available for 5V operation. However, it was discovered that although available from the OEM's, 5V LED lit switches were not stocked by the usual retail suppliers in New Zealand. It was therefore necessary to use 24V LED lit switches. The relay was used to convert the 5V signal from the Power Supply Control Board to 24V, using the incoming 24V supply.

5 Firmware Development

The firmware version used as a baseline was v3.1 of the MakerBot firmware. This version of the firmware was the last before the introduction of Jetty and Sailfish which use stepping motors for extruder control, so provided a good starting point.

The toolchain used for building the MakerBot firmware was as follows:

Installed with Arduino 1.0.5 were the avr tools, avr-gcc, avr-g++ and avrdude. These are required for cross-compiling the code to run on the Atmel ATmega micro-controllers.

The build is done using Scons, an open-source software construction tool. Version 2.3.1 was the latest version at the time and required Python version >2.4 & <3.0. Initially the 64-bit version of Python 2.7.6 was installed. Python ran perfectly however, Scons was unable to detect the installation and therefore would not install. After trying older versions and searching for a solution to the problem, it was discovered that Scons only appeared to work with 32-bit versions of Python.

The firmware was split into sections. Under the firmware directory, there were sub-directories for the Extruder, Motherboard, PowerSupply and Shared. The 'Shared' directory contained common modules of code which are utilised by Motherboard, Extruder and PowerSupply firmwares. Code modules specific to the different boards were contained under the relevant sub-directory. Under each sub-directory, for example under 'Extruder', there were a further set of code modules common to all extruder boards and a 'Boards' sub-directory with code modules specific to a particular extruder board.

For the purposes of this development, the following were used as baseline builds:

Extruder : ECV34_328P – Based on the ATMega328P from Atmel.
Motherboard : MB24_2560 – Based on the ATMega2560 from Atmel.
PowerSupply: PS_2560 - Based on the ATMega2560 from Atmel.

The SConstruct and associated SConscript.extruder, Sconscript.motherboard and Sconscript.powersupply build scripts were modified to reflect the build environment.

The rest of this chapter describes the main changes (but not all) required to get the firmware funtional. A full copy of the source along with hex files for each component can be found in Appendix B, on the CD-ROM accompanying this thesis.

5.1 Extruder Controller

The core of the extruder controller code is unchanged. However, there were a number of areas where obsolete code was removed due to the change in architecture.

By having a separate power supply, and off-loading the oven temperature control to it, there was no longer any need for heated build platform code within the extruder controller.

The following is a list of areas where obsolete code was removed:

- Software Servos – Not used
- Cooling Fan – Now controlled by Power Supply
- Thermistor Tables – Only thermocouples are used
- Platform Heater & Thermistor – Now controlled by Power Supply
- Automated Build Platform – Not available

Channel A was also changed from an output controlling a MOSFET to an input

for detecting whether filament was loaded or not. Due to time constraints, the filament checking was not integrated into the code.

SetValve() was also changed to SetSolenoid(), although the functionality remains the same.

In the Shared sub-directory, the Thermocouple.cc module was modifed to reflect the change to the MAX31855. As described in Chapter 4, the MAX31855 was 14bit including sign, whereas the original MAX6675 was 12bit plus sign. The code had to be modified to read thirteen bits of temperature data rather than twelve. The mask also had to be changed to 0x7FF0 to return only the top eleven bits, returning whole degrees rather than the quarter degree the MAX31855 is capable of. This was due to the rest of the temperature control code being written for integer values rather than float/double.

5.2 Power Supply Controller

5.2.1 Additional UART

The initial plan was to implement the Power Supply firmware as an entirely seperate piece of code written in the Arduino IDE. However, it soon became evident that there would be far too much involved in implementing a completely new interface in the Motherboard code, there were also insufficient spare pins on the Motherboard Arduino Mega to handle another hardware UART interface. As a result, it was decided to implement the Power Supply Controller as an additional “tool” on the existing RS-485 bus already used by the Motherboard for talking to the Extruder Controllers.

The Power Supply Controller firmware was based upon the Extruder Controller firmware but with modifications to implement an additional RS-232 interface to monitor the Uninterrupted Power Supply (UPS).

In the Shared sub-directory, the UART.cc module was modified to add an additional UART (UART3) configured as an RS-232 port. The baud rate required was 2400. The value for the UART Baud Rate Register (UBRR3) was defined as 832. Additionally, the UART3 interrupt service routines were added for both transmit and receive.

5.2.2 UPSPacket Class

After adding the UART, communication with the UPS was still not working. On further investigation, it was discovered that the 'Packet' class being used by the 'UART' class automatically added the start byte (0xD5), length and the cyclic redundancy check (CRC). As a result, the UPS was not sending the appropriate query string “Q1<CR>” when polled. The problem was resolved by creating my own UPSPacket class based upon the original Packet class. By removing the header and CRC trailer and the equivalent checks on the received packet, communication with the UPS was successful.

When polled, the UPS would respond with a string as follows:

(MMM.M NNN.N PPP.P QQQ RR.R SS.S TT.T b7b6b5b4b3b2b1b0<cr>

Where,

(- Start character
MMM.M - Input Voltage
NNN.N - Input Fault Voltage
PPP.P - Output Voltage
QQQ - Output Current
RR.R - Input Frequency
SS.S - Battery Voltage
TT.T - Temperature

UPS Status

Bit	- Description
7	Line Supply Fail
6	Battery Low
5	Bypass/Boost or Buck Active
4	UPS Failed
3	UPS Type (1: Standby, 0: On_line)
2	Test in progress
1	Shutdown Active

0 Beeper On

From a control point of view, only 'Status' output was of concern. The following code block was used in runUPSSlice(), to collect the UPS status.

```
for(int i=0; i<8; i++)
        board.UPSStat[i] = in.read8(37+i);
setUPSStatus(StatusToInt());
processedUPS = true;
```

The UPSStatus read from the COM port was an ASCII representation of a binary byte. The following code block was therefore required to convert the ASCII string into its equivalent 8-bit unsigned integer :

```
uint8_t PowerSupplyBoard::StatusToInt()
{
        int Status = 0;

        if (UPSStat[0] == '1')  // Line-Fail
                Status |= 0x80;
        if (UPSStat[1] == '1')  // Battery Low
                Status |= 0x40;
        if (UPSStat[2] == '1')  // Bypass/Boost or Buck Active
                Status |= 0x20;
        if (UPSStat[3] == '1')  // UPS Failed
                Status |= 0x10;
        if (UPSStat[4] == '1')  // 1 = UPS Standby - 0 = On-Line
                Status |= 0x08;
        if (UPSStat[5] == '1')  // Test in progress
                Status |= 0x04;
        if (UPSStat[6] == '1')  // Shutdown Active
                Status |= 0x02;
        if (UPSStat[7] == '1')  // Beeper ON
                Status |= 0x01;

        return Status;
}
```

5.2.3 Heater Control

As the Power Supply Controller firmware was based on the Extruder Controller firmware, the code was already there to control a heater. However, the heater was an oven rather than an extruder. The difference was in the control algorithm, the heaters in the extruders were DC and can be turned on/off at any time. The heaters in the oven are AC, switched on/off by a solid state relay

(SSR) on each phase. SSR's are controlled, they allow the relay to be activated any time, however they only allow the relay to be switched off at a zero-crossing point, to minimise power dissipation within the semi-conductor device. There are two zero-crossing points for each cycle on each phase, therefore for full control with 8-bits, the window time for the oven pulse-width-modulation (PWM) had to be set to 2.55s.

The Heater control pin was Port H, Bit 5 – OCR4A. The following code was used to configure the Timer 4 for driving the PWM pin.

```
// Timer 4:
//  Mode: Fast PWM - OCR4A for TOP count
//  Prescaler: 1/1024
//  Chamber Heater
//   - uses OCR4A and OCR4Cto generate PWM

DDRH  |= 0b00100000; // Data Direction Register H set to output on OCR4C (bit 5)
TCCR4A = 0b00000011; // initially disable heater PWM by clearing bits 2&3 of
                                Timer/Compare Control Register 4A
TCCR4B = 0b00011101; // Set Waveform Generator Mode (WGM) and Clock Select
                                (CS) on Timer/Compare Control Register 4B
OCR4C  = 0x00; // initial value for heater PWM set to 0 in Output Compare Register 4C.
                        To set, value * 156 = 10mS increments (1/2 cycle at 50Hz)
OCR4A   = 0x9B93;       // Sets the top count of the Output Compare Register 4A to
                                0x9B93 (39827), sets PWM at 2.55s period.
```

The heater is then controlled using the following function calls:

```
void ChamberHeatingElement::setHeatingElement(uint8_t value)
{
        ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
        {
                if (value == 0 || value == 255)
                {
                        pwmHEATEROn(false);
                        HEATER_CONTROL.setValue(value == 255);
                }
                else
                {
                        OCR4C = value * 156; // Set PWM, value * 156 = value * 10ms
                        pwmHEATEROn(true);
                }
        }
}

void pwmHEATEROn(bool on)
{
        if (on)
                TCCR4A |= 0b00001011;
        else
                TCCR4A &= 0b11110011;
```

}

If the Heater is off (value = 0) then the PWM is turned off by clearing bits 2&3 of TCCR4A and the HEATER_CONTROL output pin is set to LOW.

If the Heater is fully on (value = 255) the PWM is still turned off by clearing bits 2&3 of TCCR4A, but the HEATER_CONTROL output pin is set to HIGH.

If the Heater is set to any value in between (value >0 & <255) then the Output Compare Register 4C is set to the output value multiplied by 156, to give 10ms increments on the PWM output pin and the PWM is enabled by setting bits 0,1 & 3 to HIGH and bit 2 to LOW.

During testing it was clear that the default settings used for the oven PID control were incorrect as the temperature did not reach a stable level, but oscillated about the set point +7ºC/-3ºC, a swing of 10ºC which was unacceptable.

A logger was constructed using and Arduino EtherDue, it used a MAX31855-K and a K type thermocouple purchased from Jaycar. The thermocouple was fed into the oven and attached in the same location as the oven thermocouple to collect oven temperature data. The logger was also connected to the heater control output on the Power Supply Control Board, and logged the temperature and output state to a microSD card every 100mS.

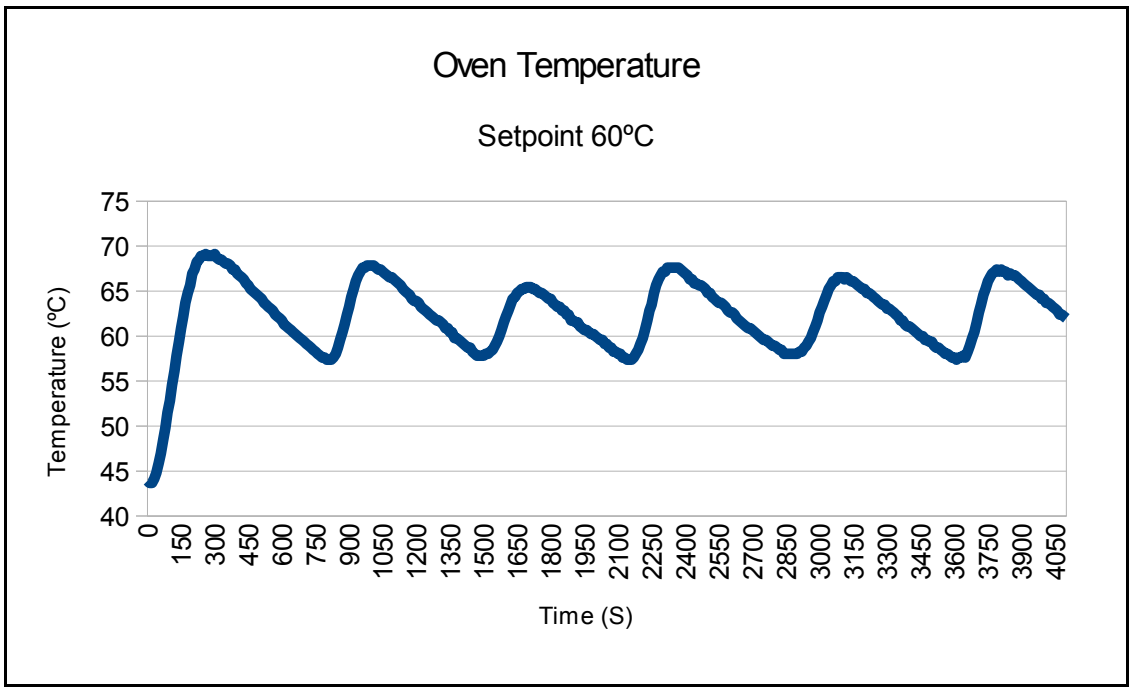


Figure 58: Oven Temperature Control Log, Default Parameters

The PID parameter were altered in an effort to correct for the overshoot but without success. Following some furthe investigation, it was noted that the PID appeared to go into saturation during both heat-up and cool-down phases. It was determined that the cause of the saturation was a function of the response time of the PID and the response time of the oven. It took approximately 12 minutes per temperature cycle of the oven, however, the PID was saturating in approximately 2 minutes. Looking at the code, it was found that the update rate for the PID was set to 0.5s. For the extruders, this is not a problem as they had a fast enough response time, for the oven however, this was much to fast. The problem was resolved by inserting an pre-processor #if statement into the header file for the heater (heater.cc). If the code being compiled was defined as IS_POWER_SUPPLY, the update interval was set to 5s, otherwise it was set to 0.5s, as originally defined. This resulted in, a small overshoot (4-5ºC) which then stabilised and kept the setpoint ±1ºC as can be seen in Figures 59, 60 & 61.

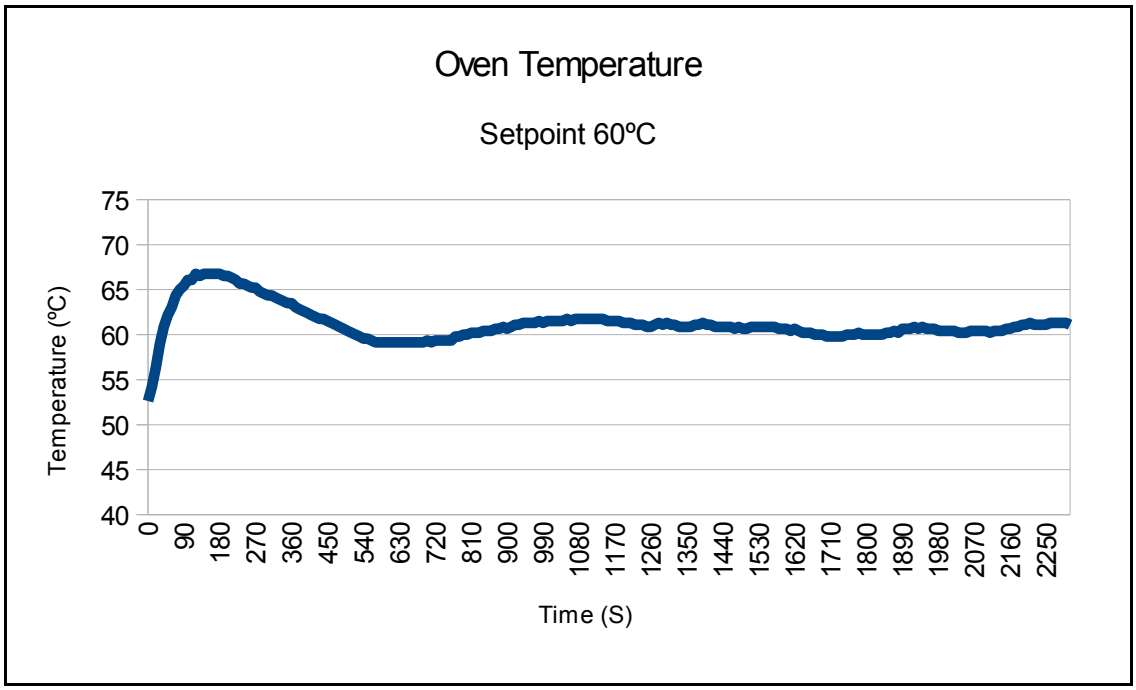


Figure 59: Oven Temperature Control, Update Interval 5s – 60 Degrees C

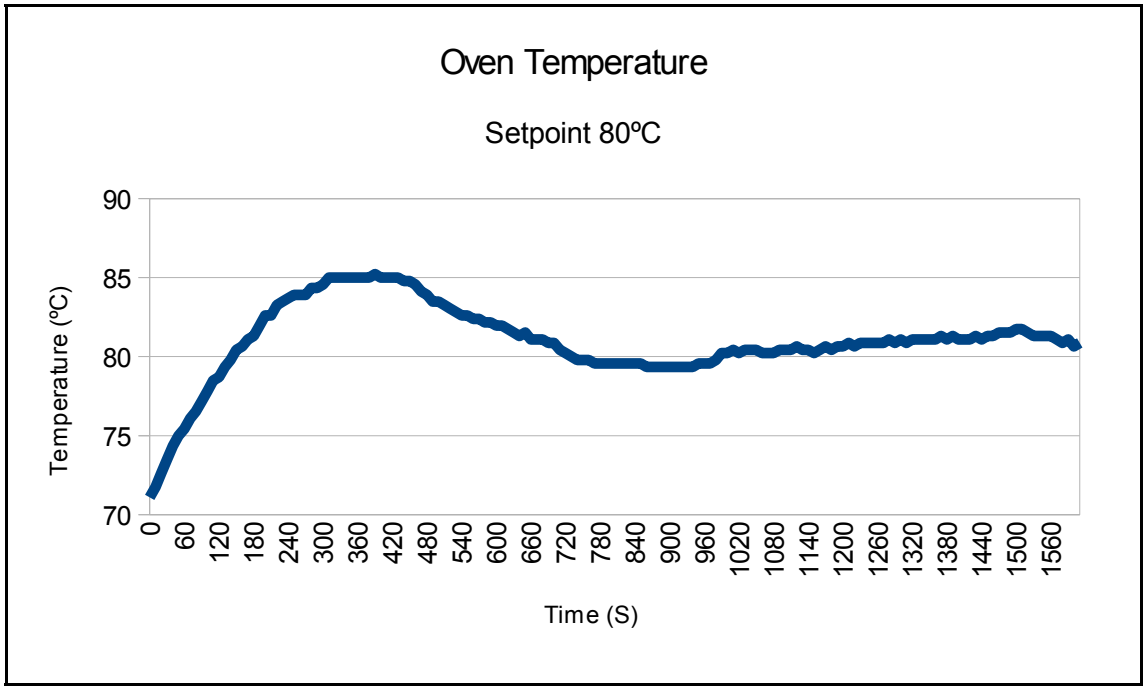


Figure 60: Oven Temperature Control, Update Interval 5s – 80 Degrees C

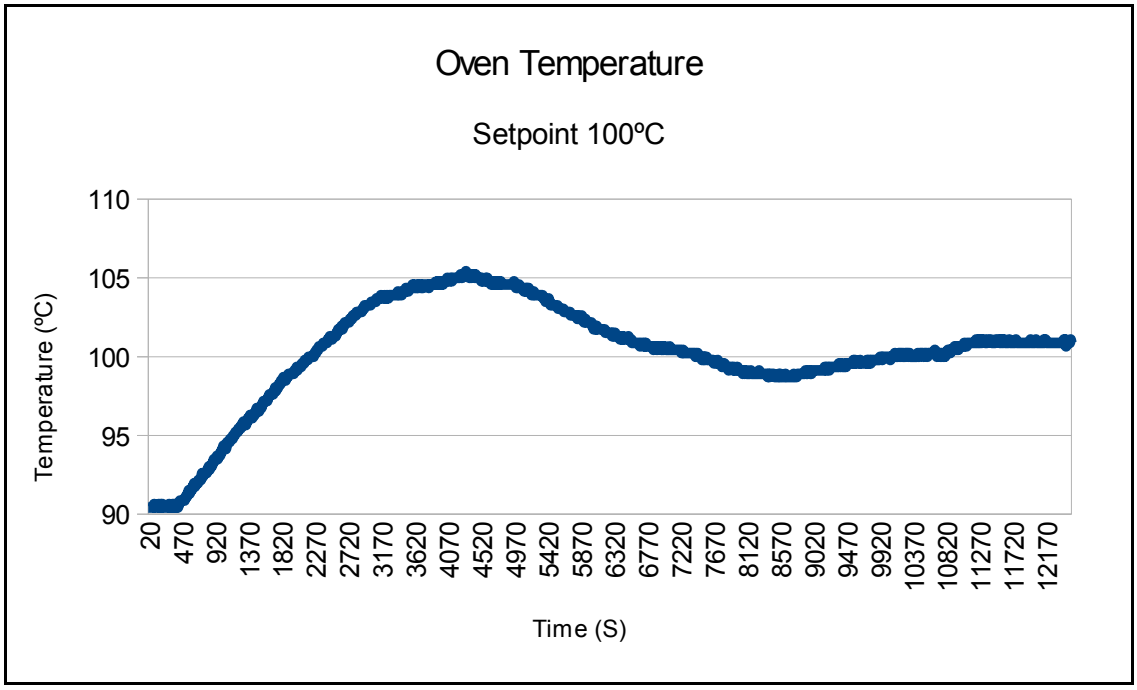


Figure 61: Oven Temperature Control, Update Interval 5s - 100 Degrees C

5.2.4 Digital Output Control Functions

A function block was added to toggle the oven lights:

```
void PowerSupplyBoard::setLight()
{
        ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
        {
                LIGHT_CONTROL.setValue(!LIGHT_CONTROL.getValue());
        }
}
```

Function blocks were also added for the UPS control relay, the fans and blowers, the table vacuum, the high voltage DC and the Front Panel Controller green Light Emitting Diode (LED). An example is shown below.

```
void PowerSupplyBoard::setUPS(bool on)
{
        ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
```

```
        {
                UPS_CONTROL.setValue(on);
        }
}
```

5.2.5 AC and DC PSU Status

The status of the AC and DC power supply rails was monitored through 8 input pins. The 3 AC phases and the UPS are each connected to a PCB mounted Myrra PoL power supply. As described in Chapter 4, the output of each of them is connected to the input side of an opto-isolator. The output of the opto-isolator pulls the relevant input pin on the Arduino down to ground when the supply is on. The AC inputs are connected to the top 4-bits of Port B and are checked using the following function call :

```
uint8_t PowerSupplyBoard::checkACInputs()
{
        return PINB & 0xF0;
}
```

The 60VDC power supplies for the stepper motor drives and the 24VDC power supply for the rest of the electronics are also connected to opto-isolators in the same way. The DC inputs are connected to the top 4-bits of Port K and are checked using the following function call :

```
uint8_t PowerSupplyBoard::checkDCInputs()
{
        return PINK & 0xF0;
}
```

5.2.6 Tool Commands

In order to allow the Motherboard to communicate with the Power Supply Controller as a “tool”, the following changes were made to the HOST.cc code module.

```
        case SLAVE_CMD_TOGGLE_SOLENOID:
                to_host.append8(RC_OK);
```

```
        board.setVac((from_host.read8(2) & 0x01) != 0);
        return true;
```

The SLAVE_CMD_TOGGLE_SOLENOID used in the extruder code was used to control the Vacuum pump output for holding the work-piece on the table.

```
case SLAVE_CMD_TOGGLE_ABP:
        to_host.append8(RC_OK);
        board.setLight();
        return true;
```

The SLAVE_CMD_TOGGLE_ABP (Automated Build Platform) was used to control the oven lights.

```
case SLAVE_CMD_GET_AC_STATUS:
        to_host.append8(RC_OK);
        to_host.append8(board.getACStatus());
        return true;
case SLAVE_CMD_GET_DC_STATUS:
        to_host.append8(RC_OK);
        to_host.append8(board.getDCStatus());
        return true;
case SLAVE_CMD_GET_UPS_STATUS:
        to_host.append8(RC_OK);
        to_host.append8(board.getUPSStatus());
        return true;
case SLAVE_CMD_GET_STARTUP:
        to_host.append8(RC_OK);
        to_host.append8(board.getPSUStart());
        return true;
```

The above commands were added to enable the Motherboard to query the state of the power supplies and also to control the startup process.

5.3 Motherboard

The main changes in the Motherboard firmware were around the interface. The original MakerBot Interface was used as the basis for my design, but an additional button was added to the allow the lights in the oven to be turned on/off at any time rather than having it as a function within a menu. The display was also changed from a 20x4 LCD to a 40x4 LCD, this enabled the new interface to fit in the same space envelope as the original Stratasys FPC. It also allowed far more information to be displayed on the screen at any one time, for instance the power supply status information.

To allow the use of the bigger screen, the LiquidCrystal.cc module was modified to add the E2 line used for the bottom lines of the display. The code also had to be modified to correct for the longer line length and additional code was added to select between the top and bottom lines as they are logically two different displays. As a result, I had to duplicate funtion calls during initialisation etc. for example:

```
void LiquidCrystal::clear()
{
 _currline = 0;  // Top half of screen
 command(LCD_CLEARDISPLAY);  // clear display, set cursor position to zero
 _currline = 2;  // Bottom half of screen
 command(LCD_CLEARDISPLAY);  // clear display, set cursor position to zero
 _delay_us(2000);  // this command takes a long time!
}
```

In the example code above, _currline is used to differentiate between the top and bottom halves of the screen. The same command had to be sent to both to complete the action required. When writing to specific locations on the screen, the line number was used to determine which half of the screen to write the character to.

The menu structure was also extended to include informations and functions relevant to the machine. As described above a power supply status menu was added as well as a pre-heat menu, to enable the machine to be warmed up prior to starting a print. The power supply status is shown below :

```
FDM OpenVantage - Power Supply Status
AC : L1-xxxx, L2-xxxx, L3-xxxx, UPS-xxxx
DC :  X-xxxx,  Y-xxxx,  Z-xxxx, 24V-xxxx
UPS Status : xxxx
```

The Motherboard polls the Power Supply Controller for the status of each supply sequentially between updates. This was to prevent slow responses to button presses.

```
OutPacket responsePacket;

// Redraw status info
switch (updatePhase) {
case 0:
        lcd.setCursor(8,1);
        if(queryExtruderParameter(2,SLAVE_CMD_GET_AC_STATUS,
                                                responsePacket)) {
                uint8_t data = (responsePacket.read8(1) & 0xF0);
                if((data & 0x20) == 0x20)
                        lcd.writeString("BAD!");
                else
                        lcd.writeString("Good");
        } else {
                lcd.writeString("XXXX");
        }
        break;
case 1:
        lcd.setCursor(17,1);
        if(queryExtruderParameter(2,SLAVE_CMD_GET_AC_STATUS,
                                                responsePacket)) {
                uint8_t data = (responsePacket.read8(1) & 0xF0);
                if((data & 0x10) == 0x10)
                        lcd.writeString("BAD!");
                else
                        lcd.writeString("Good");
        } else {
                lcd.writeString("XXXX");
        }
        break;
case 2:
        lcd.setCursor(26,1);
        if(queryExtruderParameter(2,SLAVE_CMD_GET_AC_STATUS,
                                                responsePacket)) {
                uint8_t data = (responsePacket.read8(1) & 0xF0);

                if((data & 0x80) == 0x80)
                        lcd.writeString("BAD!");
                else
                        lcd.writeString("Good");
        } else {
                lcd.writeString("XXXX");
        }
        break;
case 3:
        lcd.setCursor(36,1);
        if(queryExtruderParameter(2,SLAVE_CMD_GET_AC_STATUS,
                                                responsePacket)) {
                uint8_t data = (responsePacket.read8(1) & 0xF0);
                if((data & 0x40) == 0x40)
                        lcd.writeString("BAD!");
                else
                        lcd.writeString("Good");
        } else {
                lcd.writeString("XXXX");
        }
        break;
case 4:
        lcd.setCursor(8,2);
        if(queryExtruderParameter(2,SLAVE_CMD_GET_DC_STATUS,
```

```
                                                        responsePacket)) {
                uint8_t data = (responsePacket.read8(1) & 0xF0);
                if((data & 0x40) == 0x40)
                        lcd.writeString("BAD!");
                else
                        lcd.writeString("Good");
        } else {
                lcd.writeString("XXXX");
        }
        break;
case 5:
        lcd.setCursor(17,2);
        if(queryExtruderParameter(2,SLAVE_CMD_GET_DC_STATUS,
                                                        responsePacket)) {
                uint8_t data = (responsePacket.read8(1) & 0xF0);
                if((data & 0x20) == 0x20)
                        lcd.writeString("BAD!");
                else
                        lcd.writeString("Good");
        } else {
                lcd.writeString("XXXX");
        }
        break;
case 6:
        lcd.setCursor(26,2);
        if(queryExtruderParameter(2,SLAVE_CMD_GET_DC_STATUS,
                                                        responsePacket)) {
                uint8_t data = (responsePacket.read8(1) & 0xF0);
                if((data & 0x10) == 0x10)
                        lcd.writeString("BAD!");
                else
                        lcd.writeString("Good");
        } else {
                lcd.writeString("XXXX");
        }
        break;
case 7:
        lcd.setCursor(36,2);
        if(queryExtruderParameter(2,SLAVE_CMD_GET_DC_STATUS,
                                                        responsePacket)) {
                uint8_t data = (responsePacket.read8(1) & 0xF0);
                if((data & 0x80) == 0x80)
                        lcd.writeString("BAD!");
                else
                        lcd.writeString("Good");
        } else {
                lcd.writeString("XXXX");
        }
        break;
case 8:
        lcd.setCursor(13,3);
        if(queryExtruderParameter(2,SLAVE_CMD_GET_UPS_STATUS,
                                                        responsePacket)) {
                uint8_t data = responsePacket.read8(1);
                if((data & 0x80) == 0x80)
                        lcd.writeString("Line-Fail   ");
                if((data & 0x02) == 0x02)
                        lcd.writeString("Off-Line    ");
                if((data & 0x40) == 0x40)
                        lcd.writeString("Battery LOW ");
```

```
                    if((data & 0x10) == 0x10)
                        lcd.writeString("Failed      ");
                    if(data == 0x09)
                        lcd.writeString("Good                ");

                }else {
                    lcd.writeString("XXXX                ");
                }
                break;
        }
```

Since the Power Supply Controller is logically just another 'tool', the 'queryExtruderParameter' function was used to get the status of each of the AC, DC and UPS supplies. Additonal functions were added to allow the Motherboard to send other commands to the Power Supply Controller, such as setting the pre-heat temperatures and turning the oven lights on and off. These additional tool commands also had to be added to the Command.cc and Tool.cc code modules.

5.4 Print Reliability

During testing a problem was observed. The thermocouple in the extruders would randomly return a temperature in excess of 500ºC. This would cause the extruder heater to shutdown immediately due to an apparent overheat condition. However, since the extruders only heat up at around 10ºC/second, this was an erroneous reading. Looking into the extruder code, it was determined there was an error in Thermocouple.cc, whereby if an error occurred when reading the thermocouple, the function returned straight away, without first releasing the I^2C bus. This was modified as follows:

```
    if (i == 15) { // Safety check: Check for open thermocouple input
            if (so_pin.getValue()) {
                current_temp = BAD_TEMPERATURE; // Set the temperature to 1024
as an error condition
                cs_pin.setValue(true);   // GGH to ensure bus is released 5/11/14
                return SS_ERROR_UNPLUGGED;
            }
```

Through observation, this problem was determined to be a potential electro-magnetic compatibility (EMC) problem caused by the extruder motor turning

on/off, even though TVS diodes had been fitted. It was determined that solving the EMC problem at this late stage in the project would cause significant delays and since it was a transient problem, not continuous, an alternative solution could be found. Rather than just shutting the heater down straight away, the same approach would be taken to an timeout or other error, the error count would be incremented and if the MAX_ERROR_COUNT exceeded 5, the extruder would be shut down. Given that the heater temperature was sampled every 0.5 seconds, this would mean that an overheat could exist for upto 2.5 seconds, which at 10ºC/second would only amount to 25ºC rise.

With these fixes in place, there were no more instances of the extruder heater shutting down during a build.

6 Software Design

As identified in Chapter 2, ReplicatorG (RepG) is the de-facto software for controlling open source 3D printers, although there are a number of others that have been developed. It was decided that as the rest of the machine was based around MakerBot hardware and firmware, the software used to control it should also be kept with RepG. The source for RepG was held on GitHub. The toolchain required for the software development included Git, to extract the latest source, Java which RepG was written in and Ant, a command-line tool for building software.

Git version 1.8.4 was installed from http://git-scm.com/download/win and the latest ReplicatorG repository from git://github.com/makerbot/ReplicatorG.git was cloned. Java SE 8 Update 5 was installed from http://download.oracle.com/otn-pub/java/jdk/8u5-b13/jdk-8u5-windows-x64.exe. Apache Ant version 1.9.4 from http://ant.apache.org/bindownload.cgi was installed, completing the tool chain.

The rest of this chapter describes the main (but not all) modifications to the RepG software required to get the machine functional. A full code listing for the modified RepG, along with a distributable .ZIP file for windows, can be found in Appendix C, on the CD accompanying this thesis.

6.1 ReplicatorG 0040

The latest version of RepG at the time of this project was Version 0040, and so made the logical starting point. As mentioned above, the latest RepG repository was cloned from github.

RepG was built from the command-line using “ant dist-windows” in the source directory.

Having built RepG, testing commenced against the hardware. Initial testing was done using the control panel within the RepG software. The control panel has the ability to jog the three axes, set and monitor the extruder temperatures and set the extruder motor direction and speed.

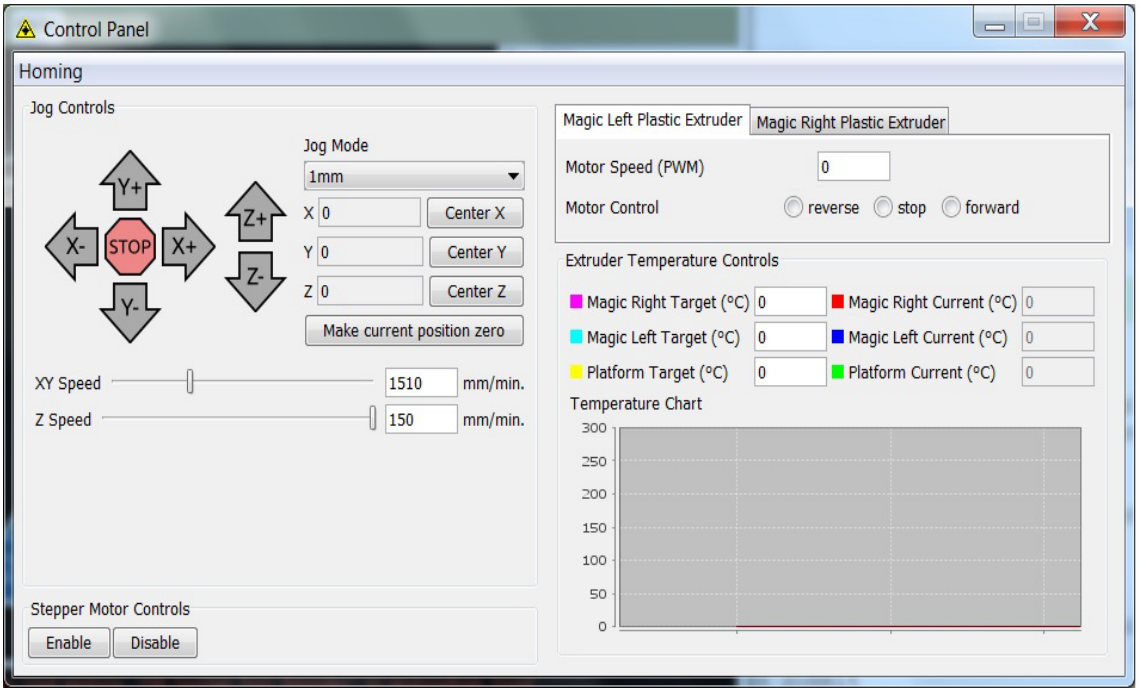


Figure 62: Original 0040 Control Panel

First tests were successful, the motor speed and direction were able to be set from the control panel. Additionally when the heater setpoint and temperature were checked, they worked as planned and with the default PID settings, the temperature was maintained within ±1ºC.

Confident that the hardware was working as expected, I closed the control panel and entered G-code “M108 T0 256” into the editor window. This should have set the motor on Tool 0 to full speed, going forward. However, when the build button was pressed to convert it to S3G code and it was sent to the machine, the extruder motor on Tool0 did not turn on as expected. To understand what was going on, a spare Arduino was used with an RS-485 interface and set it to listen to the traffic on the RS-485 bus between the Motherboard and Tool 0. When monitoring the RS-485, it was discovered that M108 T0 256 was being converted to the following S3G code by RepG :

“0x86 0x00” - Tool change to index 0

“0x89 0x1F” - disables all drives

With some further digging, it was discovered that the disable drives command is appended to the end of every build as specified in the machines.xml definition. So the M108 was being translated to a select Tool 0 rather than setting the motor speed.

The version of RepG being used to control my own Thing-O-Matic was still at version 0037, so out of interest the same G-code was entered into the editor window (“M108 T0 256”) and the code was built to an S3G file. When the file was opened in a hex editor, the following was observed:

“0x86 0x00” - Select Tool Index 0

“0x88 0x00 0x04 0x01 0xFF” - Pass through command to Tool0 setting PWM motor speed to 256

“0x89 0x1F” - Disable Drives

The generated S3G code was tried and found to work.

The MakerBot user-group on Google were consulted and a request for help/advice was posted, asking why the code might be broken between version 0037 and 0040. The response was that basically DC motor drives were no longer being used as stepping motor extruders were preferred, so the M108 code had been depricated and removed from the source tree.

With some helpful pointers from Dan Newman (Newman, 2014) of the MakerBot user-group, version 0037 was retrieved by tracking back through the archive of previous commits, then by comparing the date on the working version 0037 against the archive, the commit used to generate the code was selected and a zip file of the complete source at that point in time (22 June 2012) was downloaded.

6.2 ReplicatorG 0037

ReplicatorG 0037 was built in the same way described previously. Testing confirmed that M108 was converted to the correct S3G codes. Having solved the problem with M108 by reverting back to version 0037 of ReplicatorG, the next hurdle was how to incorperate an additional tool. When developing the firmware for the power supply, to keep the power supply seperate from the extruders, it was decided that the slave ID of the Power Supply Controller would be set to 127. However, when the power supply tool was added to the thingomatic.xml machine config file, RepG would only enumerate the extruders on ID 0 and 1. When the slave ID for the power supply was changed to 2 (see below), RepG was able to detect all three tools.

```
<tools>
        <tool name="Right (0)" index="0" type="extruder" material="abs" motor="true"
                fan="false" heatedplatform="false" heater="true" uses_relay="false"/>

        <tool name="Left (1)" index="1" type="extruder" material="support" motor="true"
                        fan="false" heatedplatform="false" heater="true" uses_relay="false"/>

        <tool name="Chamber" index="2" type="powersupply" material="" motor="false"
                        fan="false" heatedplatform="false" heater="true" uses_relay="false"/>
</tools>
```

Since the power supply was defined as a heater without a heated platform, I had to use the SetTemperature command rather than the SetPlatformTemperture to set the oven temperature. This was tested by manually creating an S3G file using a hex editor with the following:

“0x88 0x02” - Pass command through to Tool index 2

“0x1F” - Set Temperature Command

“0x02” - Command Payload Length

“0x00 0x64” - Temperature (100)

The S3G code above was uploaded to the machine and the heater output pin on the Power Supply Controller Board monitored with a scope and verify it's operation. This confirmed that it the code for RepG would not prevent the

sending of commands to a tool other than 0 or 1.

The layout of the ReplicatorG code modules was logical, making it easy to walk throught the code to find the modules used to define the Control Panel, even without any experience of programming in Java.

In the temperature monitoring window, the platform setpoint and monitoring were changed to control/monitor the oven temperature. This was fairly straight forward, rather than checking for references to either Tool 0 or Tool 1 having a “HeatedPlatform”, the “Heater” for Tool 2 (the power supply) was used. The same function calls to the heater were used for Tool 2 as for Tools 0 and 1.

The code used to control the motor direction and speed was located in ExtruderPanel.java. Since it should not be possible to operate Tool 0 and Tool 1 at the same time, because of the diffference in Z height, the code was modified to energise the solenoid when Tool 1 is turned on and turn off the Tool 0 motor and visa-versa when Tool 0 is turned on. This worked however, radio buttons would not reset as they were in two different button groups. The result was that when the Tool 0 motor was turned off by operating Tool 1, although the motor stopped, in the Tool 0 tab, the motor was still shown in either forward/reverse according to the radio buttons and the same for Tool 1 when operating Tool 0. This was not a functionality problem, but should be addressed in any further development.

Since the Stratasys machine does not have the capability of running an automated build platform (ABP) like the Thing-O-Matic. ControlPanelWindow.java was modified, the code previously used to control the ABP was changed to toggle the lights in the oven on/off instead.

```
JButton lightOnButton = new JButton("Light On/Off");
lightOnButton.addActionListener(new ActionListener()
{
        public void actionPerformed(ActionEvent e)
        {
                machine.runCommand(new replicatorg.drivers.commands.
                                                ToggleAutomatedBuildPlatform(true,2));
        }
```

```
});
activationPanel.add(lightOnButton);
```

An additional pair of buttons was added in the same panel to control the vacuum pump, used to hold the work piece on the table, in the oven.

```
JButton vacOnButton = new JButton("Vac On");
vacOnButton.addActionListener(new ActionListener()
{
        public void actionPerformed(ActionEvent e)
        {
                machine.runCommand(new                replicatorg.drivers.commands.
SelectTool(2));
                machine.runCommand(new                replicatorg.drivers.commands.
OpenValve());
        }
});
activationPanel.add(vacOnButton);
activationPanel.add(Box.createHorizontalGlue());
JButton vacOffButton = new JButton("Vac Off");
vacOffButton.addActionListener(new ActionListener()
{
        public void actionPerformed(ActionEvent e)
        {
                machine.runCommand(new                replicatorg.drivers.commands.
SelectTool(2));
                machine.runCommand(new                replicatorg.drivers.commands.
CloseValve());
        }
});
activationPanel.add(vacOffButton);
activationPanel.add(Box.createHorizontalGlue());
```

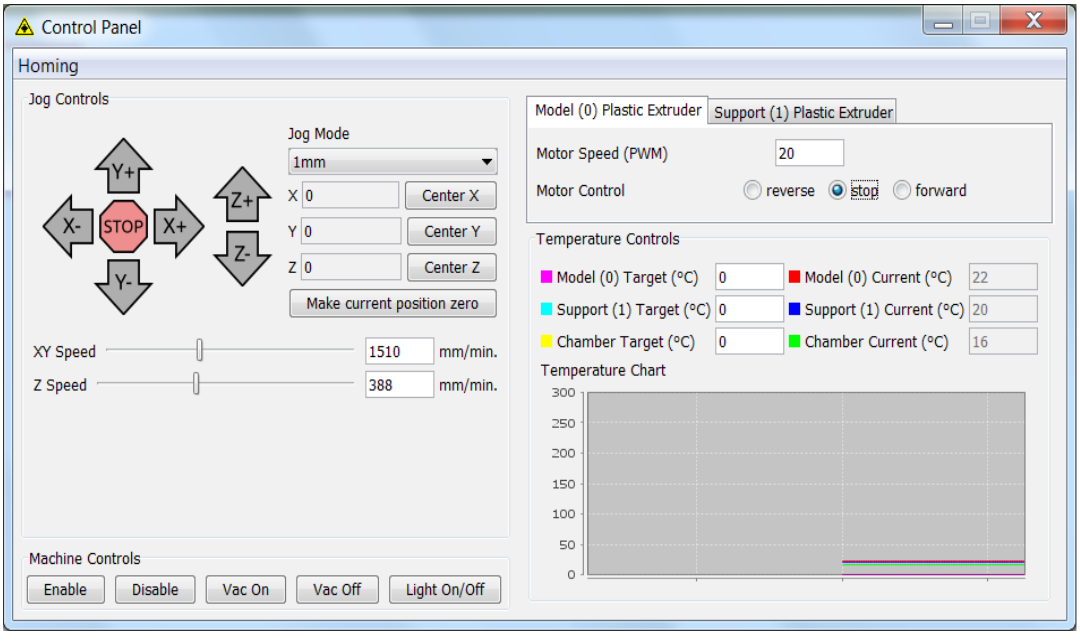


Figure 63: Modified 0037 Control Panel Layout

The OnboardParametersWindow.java code module was modified, adding the oven/chamber as another tool tab.

```
for(ToolModel t : tools)
{
        if (t.getIndex() == 2)
                paramsTabs.addTab("Chamber", new
                        ExtruderOnboardParameters(targetParams, t,(JFrame)this));
        else
                paramsTabs.addTab("Extruder"+t.getIndex(), new
                        ExtruderOnboardParameters(targetParams, t,(JFrame)this));
}
```

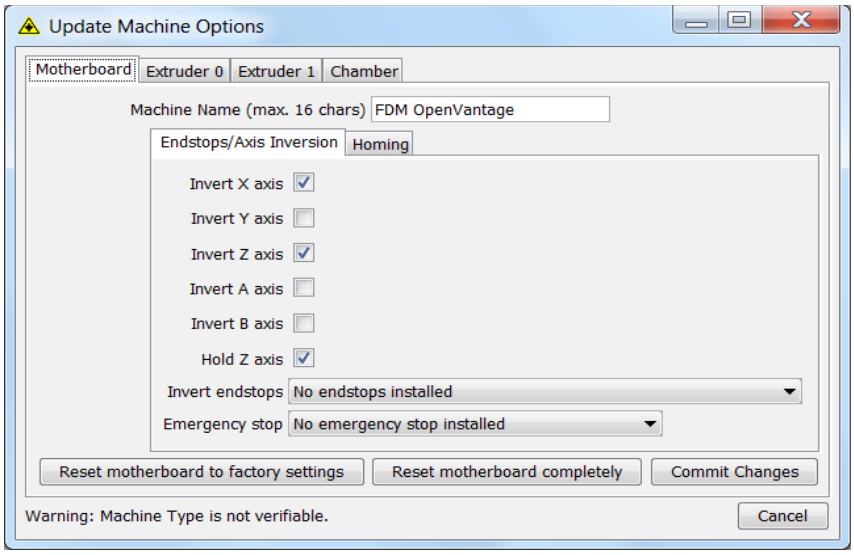


Figure 64: Machine Options - Chamber Tab Added

The ExtruderOnboardParameters.java code module, used for setting the PID parameters on the tools was also modified, adding PID parameters to the tab.

```
if(toolIndex == 2)
{
        PIDPanel pidPanel = new PIDPanel(OnboardParameters.EXTRUDER,
                                        "Chamber PID parameters", toolIndex);
        this.add(pidPanel,"growx");
        commitList.add(pidPanel);
}
```

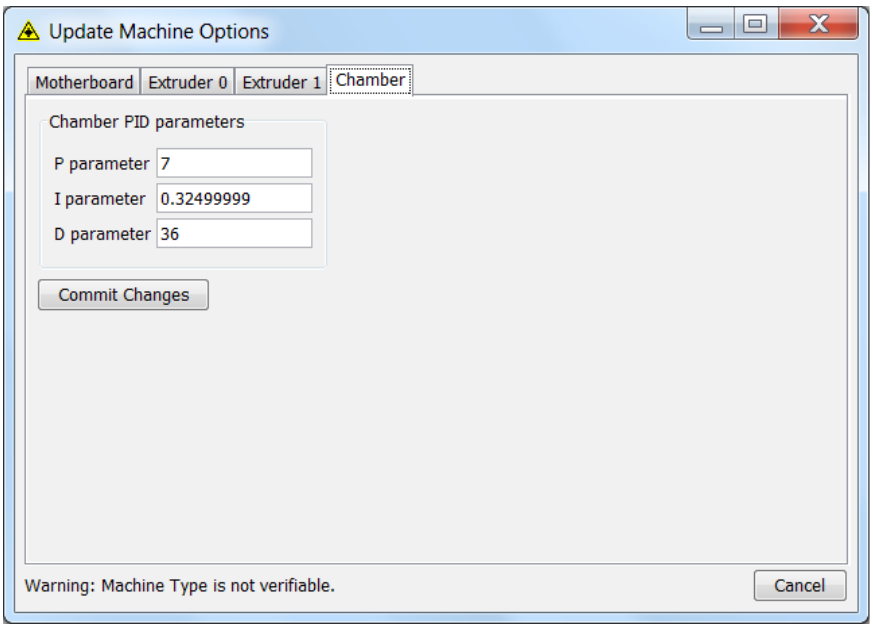


Figure 65: Chamber PID Parameter Entry Tab

7 Calibration

Before commencing printing, it was necessary to calibrate the machine. The default settings for the Thing-o-Matic were based upon steppers directly driving a belt using a small timing gear (10.82mm) on the X and Y axes and a direct drive lead-screw on the Z axis. All steppers were driven at 1/8th steps (1600 pulses/revolution). The X and Y axes required 47 steps/mm and the Z axis required 200 steps/mm. This equated to a maximum possible resolution of 0.02mm in X and Y and 0.005 in Z.

Initially the machine axes were recalibrated based upon the 1/8th steps (1600 pulses/revolution) used by the MakerBot design. Due to the belt design of the Stratasys machine, the Y axis moved half the distance of the X axis. Using a digital vernier caliper attached between the frame of the machine and the extruder head, the X axis required 70 steps/mm. The Y axis required 35 steps/mm. The Z axis was driven by a timing gear from the motor, to a belt. A larger timing gear steps the angular distance down further, before translating it to linear displacement at the lead-screws either side of the chamber. The result of all the gearing down was that at the standard 1600 pulses/revolution the Z axis required 1260 steps/mm. This gave a theoretical Z axis resolution of 0.0008mm, X was 0.014mm and Y was only 0.028mm.

The maximum resolution of the Stratasys FDM Vantage was quoted as 0.127mm in all axes, see Table 1. Using the settings above would have resulted in a higher resolution machine however, in the Y axis, the machine would have still been inferior to the standard MakerBot Thing-o-Matic. This was not acceptable and since the replacement stepper drives installed were capable of micro-stepping at upto 51200 micro-steps/revolution, it was decided that the individual axes should be re-calibrated using alternative settings to improve the overall performance.

The aim was to get each axis in the region of 5 micros resolution. Taking the initial calibration settings, the X axis was reset to operate at 5000 pulses/revolution and required 221.56 steps/mm, the Y axis was reset to operate at 10000 pulses/revolution and required 221.56 steps/mm and the Z axis was reset to operate at 400 pulses/revolution, and required 315.13 steps/mm.

This gave an overall theoretical resolution of 0.0045mm and X and Y and 0.0032mm in Z, far exceeding both the original Stratasys FDM Vantage X performance and that of the MakerBot Thing-o-Matic upon which this was based.

Once calibrated the midpoint between the two extruder heads was used as a reference for working out the maximum build envelope. The build envelope was set to 440mm in the X axis, 370mm in the Y axis and 415 in the Z axis.

Having determined the maximum dimensions of the build envelope, it was then necessary to get the offsets from the center position (0,0,0) and the home position, determined by the X,Y and Z limit switches. The X and Y axes had limit switches for both maximum and minimum but the home position was determined using just the minimum for each axis (the front left hand corner of the machine). The Z axis had only a maximum limit switch, with the minimum being the extruder height, determined in firmware by the calibration process. This was done by manually centering the extruder head to the build table (using the jog buttons on the control panel in RepG) and then manually driving the table up to the point where it just touches the extruder nozzle. This position was set to 0,0,0 using the “Make current position zero” button on the control panel. Each axis was then set to home using the “Homing” drop-down menu. The home position was saved to the motherboard eeprom using gcode (M131 X Y Z). The machine could then be reset by sending each axis to its home position and recalling the offsets stored above, the axes could then be driven to the 0,0,0 position to begin printing.

The gap between the extruder head and the build platform was measured with feeler gauges and the offset entered into the home parameters page.

The next step in preparing to print was the check the extruder parameters. The extruder had been tested previously, to confirm the operation of the individual interfaces, however this had all been carried out without having any filament loaded. The extruder was heated up to 260ºC and Stratasys ABS/PC filament loaded. The speed of the extruder motor was slowly increased until the drive wheels began to slip on the filament. The point at which this began to occur was not acceptable, the filament was not coming out of the nozzle fast enough. The temperature of the extruder was increased and the filament was extruded much faster, before stopping altogether. By looking at the extruder, it was determined that the temperature of the extruder was no longer being checked. When the extruder was reset, it could be seen that the extruder had indeed began cooling down, resulting in the filament stopping flowing. With some further testing, it was determined that the extruder heater shut down if the temperature exceeded 280ºC. Looking through the Extruder Controller firmware, it was determined that a hard limit of 280ºC had been coded into the extruder for safety. Given that the purpose of the machine was to investigate alternative materials, it was decided that this should be increased to 350ºC to allow for materials requiring higher melting temperatures. This resolved the extruder heater shut-down problem, however 260ºC should have been sufficient to melt the PC/ABS filament, so at this point it was decided to check the temperature of the extruder with an external, calibrated thermocouple, connected to a Fluke 52, dual input digital thermocouple meter. It was discovered that the temperature being reported by the embedded thermocoupe did not match the reading on the Fluke meter. At 150ºC, the extruder thermocouple was only reporting 146ºC, but at 270ºC, the extruder thermocouple was only reporting 256ºC, some 14ºC below temperature.

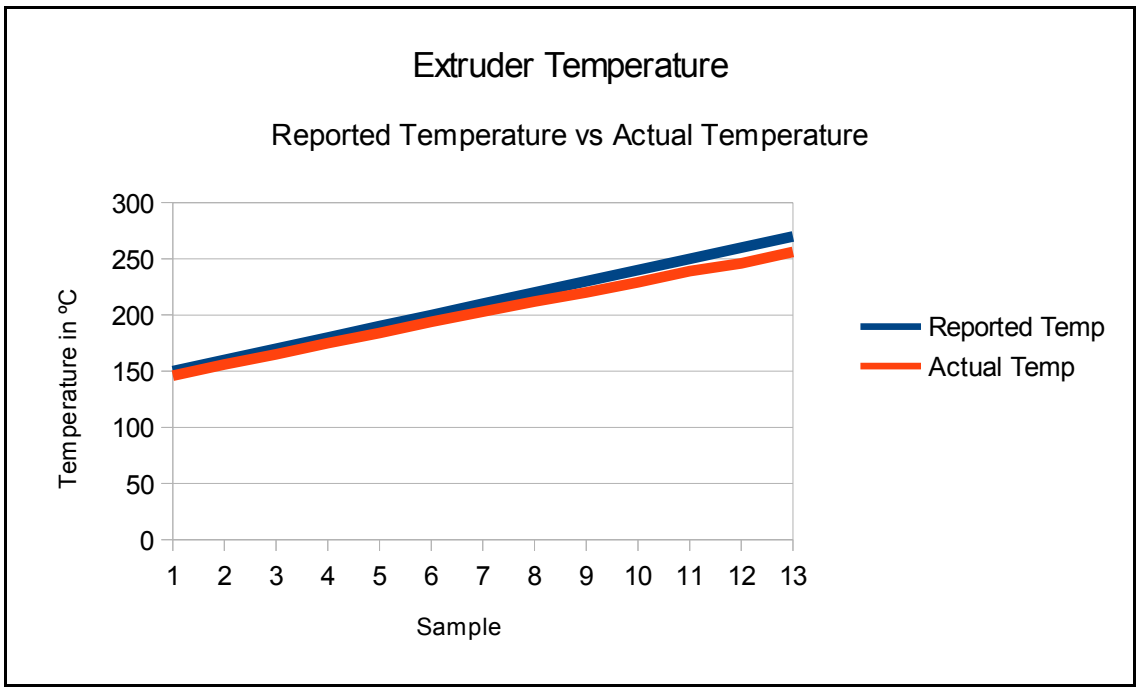


Figure 66: Extruder Temperature Error Measurement

The relationship was essentially linear, as can be seen in Figure 66.

The equations for the low and high temperature offsets are shown in Equations 1 & 2 respectively.

Equation 1 : $(150 + x) / y = 146$

Equation 2 : $(270 + x) / y = 256$

These equations were transposed for y (Equation 3) and used to calculate x (Equation 4).

Equation 3 : $(150 + x) / 146 = (270 + x) / 256$

Equation 4 : $x = 1020/100 = 9.27$

The value for x was then substituted back into Equation 1 to solve for y.

Equation 5 : $y = (150 + 9.27) / 146 = 1.09$

The correcting factors were applied to the measured thermocouple values in the thermocouple.cc code module.

```
cal_factor = (raw + 9.27)/1.09;
```

Subsequent testing with the Fluke 52 showed that the measured temperature matched the reported temperature, however, following the application of the correction factor, it was discovered that the default PID parameters were no longer correct, the temperature was no longer kept stable but began to oscillate and climbed well beyond the set-point.

Further investigation of the firmware revealed a number of 'fudge' factors that had been put in over the years to compensate for short-comings in the heater control code. These were removed and the code changed to reflect the original Arduino PID development done by Brett Beauregard [Beauregard, 2012]. The heater was then re-tuned. Since the heaters were much higher power, they were much more responsive than the heaters used in any open-source 3D printer. The following settings were found to produce a reasonable compromise between overshoot and settling time.

Parameter	Default MakerBot Settings	FDM Vantage Settings
k_P	7	5
k_I	0.325	4
k_D	36	1.5

Table 10: PID Paramenter Settings

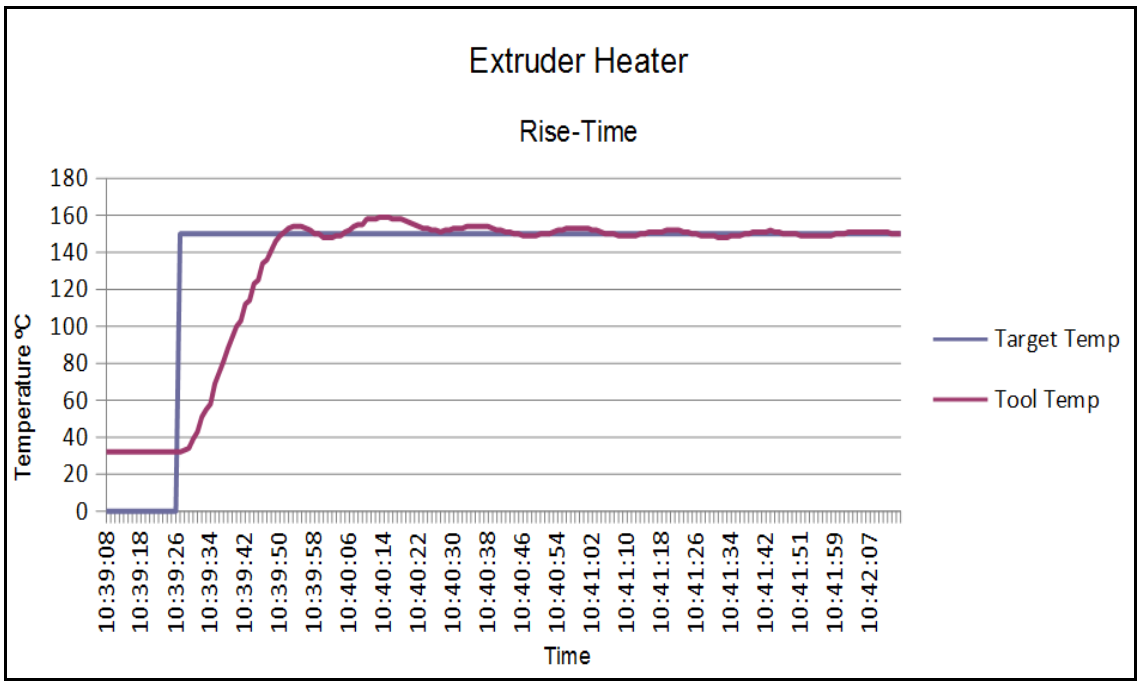


Figure 67: Extruder Heater Rise-Time

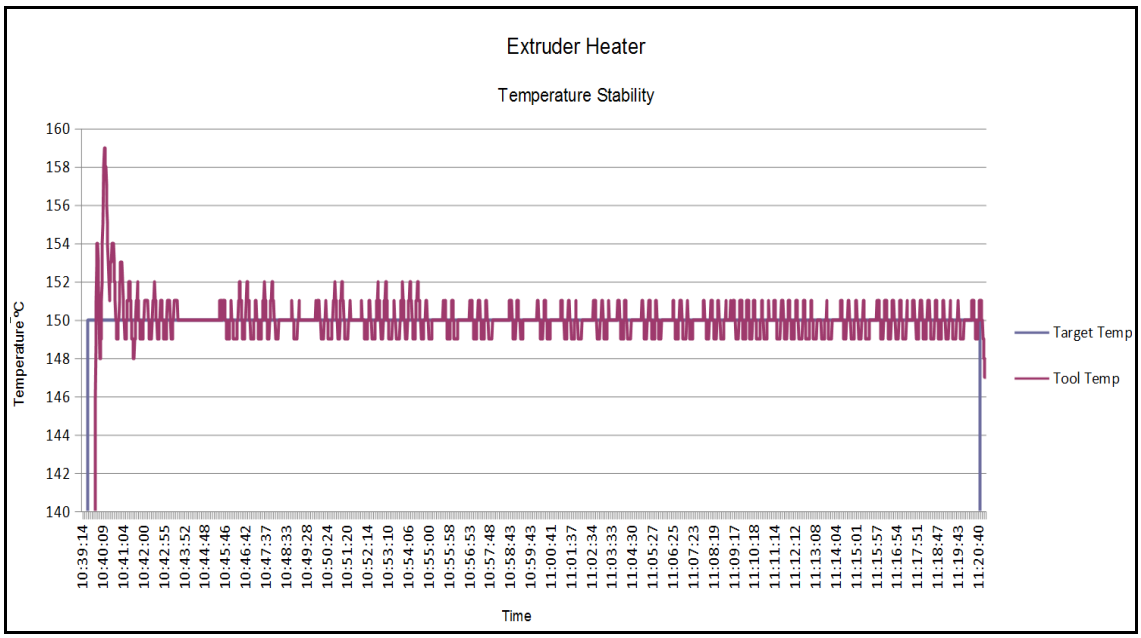


Figure 68: Extruder Heater Stability

As can be seen in Figures 67 and 68, the overshoot was limited to 9ºC and the temperature was stable to ±1ºC after stabilising. The temperature was stable to within ±2ºC within under 2 minutes for the 150ºC setpoint.

8 Testing

There were four main process variables that could be altered;

- Temperature
- Layer Thickness
- Flow Rate – Extruder Motor Speed
- Feed Rate – X/Y Axis Motor Speed

The settings for all of the above parameters were driven by the material being extruded, temperature being the main one, as differnet materials have different melting points. Different materials also have different viscosities, which drive the flow and feed rates. Layer thickness sets the Z-axis resolution and is driven by flow & feed rates possible for each material.

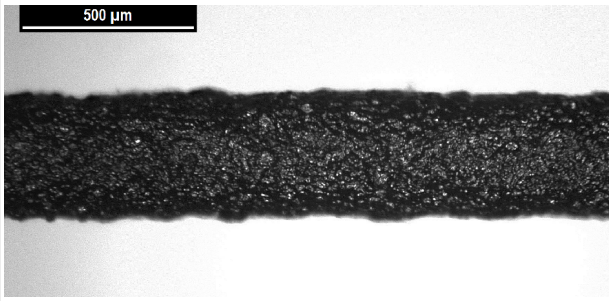
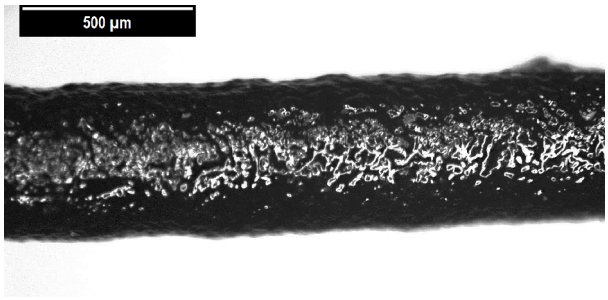
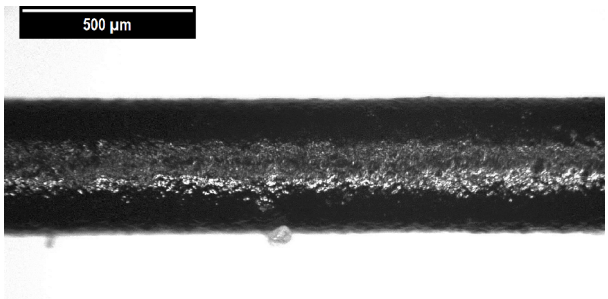
Given than the parameters will be different for each material and potentially layer thickness, it was decided that for the purposes of proving the machines capabilities, the base material would be the Stratasys polycarbonate / acrylonitrile butadiene styrene blend (PC-ABS) supplied with the machine. A layer thickness of 0.1mm was chosen as it would demonstrate a higher resolution capability than the OEM machine (0.127mm at best).

Initial testing was conducted with the door removed and the chamber cold. This allowed for easier monitoring of the build. A 3mm thick acrylic sheet was used on the build table. Adhesion was a problem, however this was solved with the application of a layer of adhesive from a glue stick.

8.1 Temperature

Before determining any other parameters, the correct extrusion temperature must be found.

The temperature was initially set to 260ºC, based on previous experience with the material. The extruder motor was turned on and a sample length of extruded filament collected. The temperature was then increased by 2ºC and the process repeated. The filaments collected were then photographed under a microscope. The surface topology along with observations made at the time of extrusion were used to determine the optimum temperature for extrusion.

Temperature	Observations	Microscopy Image
260ºC	Did not flow smoothly, kept curling up around the extruder nozzle. Rough to touch. Dull, matt finish. Inconsistent extrusion speed.	
262ºC	Flowed better but still rough to touch. More glossy in appearance.	
264ºC	Much better flow but occasional rough areas. Nice glossy appearance.	

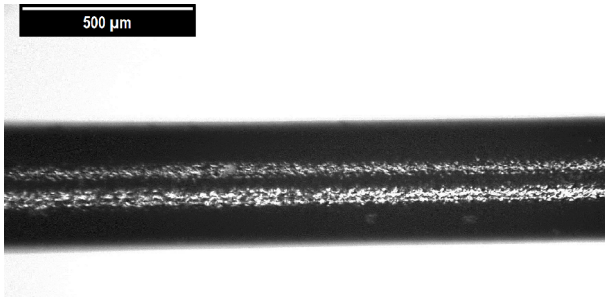
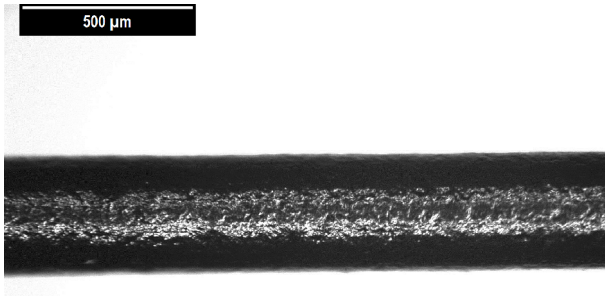
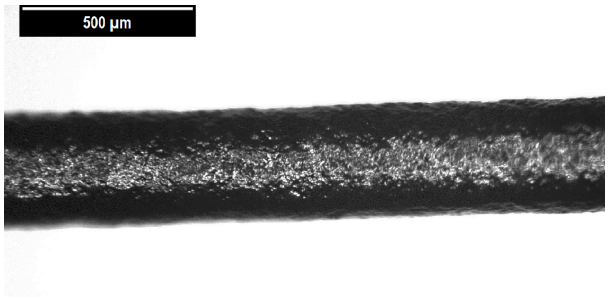
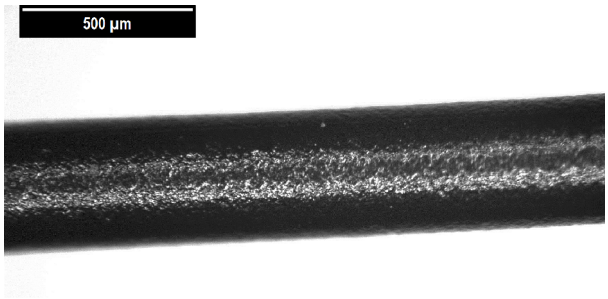
Temperature	Observations	Microscopy Image
265ºC	Consistent flow with smooth surface and glossy appearance.	
266ºC	Consistent flow with smooth surface and glossy appearance.	
268ºC	Good flow, but slightly rougher. Some stringing after extrusion was stopped and remained soft for several seconds.	
270ºC	Good flow, excessive stringing and remaine very soft for several seconds after extrusion.	

Table 11: Extrusion Temperature for Stratasys PC-ABS Filament

From the observations above, it was evident that at 260ºC the extruder was too cool and incomplete melting was occuring. Even at 262ºC the filament was melting more but still incomplete as was evident from the lumpy texture. 264ºC still contained evidence of incomplete melting with occasional lumps and rough

areas. 266ºC was observed to be consistently melting. An extra sample was taken at 265ºC, given that 264ºC was almost right and was seen to be an optimal temperature. Above 266ºC, although the filament extruded well, it remained too hot for too long after leaving the nozzle, leading to deformation and string bridges back to the nozzle when pulled away.

Since the extruder was capable of maintaining temperature to ±1ºC and 264ºC was seen to still have evidence of incomplete melting, the optimum temperature for the extruder, for the PC-ABS material, was selected at 266ºC, allowing for a range of 265-267ºC at the nozzle.

8.2 Flow & Feed Rate

The default Skeinforge settings for the Thing-o-Matic profile were used to get a baseline for parameter modification. The 20mm_Calibration_Box.stl model in the RepG examples directory was used and sliced using Skeinforge35. The gcode was then run on the machine.

Initial testing was unsuccessful, as there appeared to be insufficient material layed down, the extruder head ended up with a melted mass clogged around the nozzle.

It was decided that it would be quicker to conduct a series of trial and error tests, using the X & Y co-ordinates generated by Skeinforge and manually adjusting the layer height, flow rate and feed rate paramenters. The gcode was modified in the text editor interface of RepG rather than modifying parameters and re-slicing each time, given the length of time taken to perfom the gcode generation.

8.2.1 Manual Base Layer Parameters

It was determined that a consistent raft (layed down before the part gets built) could be achieved with a Z height of 0.4mm, with a flow rate of 35 (PWM setting) and a feed rate of 100mm/minute for the base layer. Giving an extrusion height of approximately 0.55mm, 2mm wide.

8.2.2 Manual Interface Layer Parameters

With a consistent base layer, the interface layer was added. With a Z height of 0.6mm, and maintaining the flow rate of 35, a feed rate of 1200mm/minute appeared to give sufficient material to bridge the gap between the base layer filaments. However, the extrusion height was not consistent due to thinning of the extrusion over the base layer filaments and thickening over the gaps. Rather than increasing the flow rate or slowing the feed rate to put down more material, a second interface layer at a Z height of 0.7mm was added with the same parameters. This resulted in a nice smooth consistent extrusion height.

8.2.3 Manual First Layer Parameters

From the default Skeinforge gcode, it was observed that the first layer of the part was layed down at a slightly slower feed rate than the rest of the part. This was put down to the need to ensure the first layer made good contact with the interface layer to prevent lifting the part off the raft. The co-ordinates of the fill for the first layer were modified based upon the thickness of the filament extruded to ensure complete coverage without too much excess. The first layer was layed down at a Z height of 0.8mm, with flow rate remaining at 35 and feed rate for the inner perimeter and the fill at 1200mm/minute and the outer perimeter at 1920mm/minute (the default 32mm/s used by Skeinforge).

8.2.4 Manual Remaining Layer Parameters

The remaining layers were printed using the feed rate calculated by Skeinforge with just the Z height modified to match the first layer at 0.8mm + 0.1mm per layer.

8.2.5 Result of Manual Coded Test Box

The 20mm Calibration Box was successfully printed at 0.1mm resolution using the above parameters. Additionally, the printed part was analysed under a microscope and as can be seen in Figure 69, the layers are consistently 0.1mm.

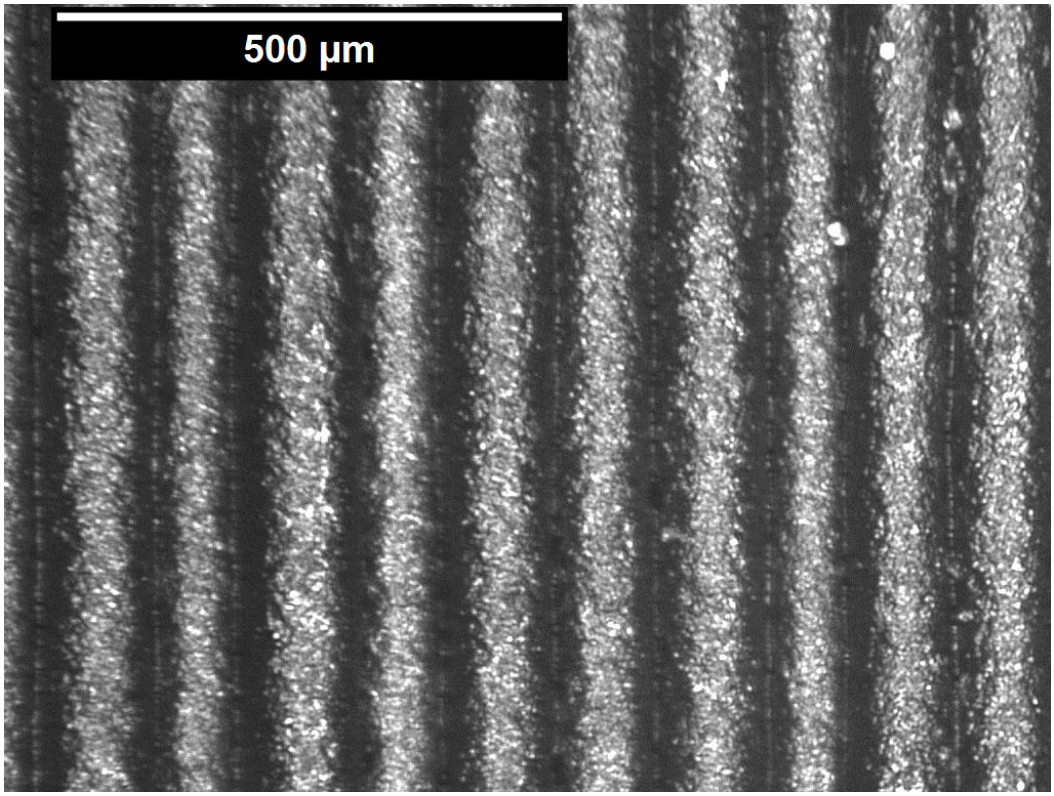


Figure 69: Microscopic image of Manually Coded Test Box

8.3 Skeinforge Raft Modifications

From past experience, it has generally been seen that if the raft, is layed down well, then generally the part itself gets built well. If the raft does not go down well and stick to the build platform, the part moves around too much and either fails due to the extruder head hitting the part or has significant errors due to relative movement between the part and the extruder head.

Investigating the Skeinforge Python code used, it was seen that the parameters for the raft generated was dependent upon the layer thickness. There seemed to be no logical reason for this. The raft should remain a constant, with layer height only affecting what is above the raft. It was decided to modify the Python code (raft.py) to make the raft a consistent thickness allowing for repeatable raft generation regardless of layer thickness.

The raft consisted of two layer types, the base layer and the interface layer(s). The code used to change the thickness of the base and interface layers was commented out and the thickness parameters passed directly to the addLayerFromSegments() function.

```
        def addBaseLayer(self):
                "baseLayerThickness = self.layerThickness *
self.baseLayerThicknessOverLayerThickness"
                "zCenter = self.extrusionTop + 0.5 * baseLayerThickness"
                "z = zCenter + baseLayerThickness *
self.repository.baseNozzleLiftOverBaseLayerThickness.value"
                if len( self.baseSegments ) < 1:
                        print('This should never happen, the base layer has a size of zero.')
                        return
                "feedRateMultiplier = self.repository.baseFeedRateMultiplier.value"
                "self.addLayerFromSegments( feedRateMultiplier,
self.repository.baseFlowRateMultiplier.value, baseLayerThickness,
self.baseLayerThicknessOverLayerThickness, self.baseSegments, z )"
```

```
                self.addLayerFromSegments( 0.052, 1, 0.50 , 1, self.baseSegments, 0.4 )

        def addInterfaceLayer(self):
                "interfaceLayerThickness = self.layerThickness *
self.interfaceLayerThicknessOverLayerThickness"
                "zCenter = self.extrusionTop + 0.5 * interfaceLayerThickness"
                zCenter = self.extrusionTop + 0.1
                "z = zCenter + interfaceLayerThickness *
self.repository.interfaceNozzleLiftOverInterfaceLayerThickness.value"
                self.interfaceIntersectionsTableKeys.sort()
                if len( self.interfaceSegments ) < 1:
                        print('This should never happen, the interface layer has a size of zero.')
                        return
                "feedRateMultiplier = self.repository.interfaceFeedRateMultiplier.value"
                "flowRateMultiplier = self.repository.interfaceFlowRateMultiplier.value"
                "self.addLayerFromSegments( feedRateMultiplier, flowRateMultiplier,
interfaceLayerThickness, self.interfaceLayerThicknessOverLayerThickness,
self.interfaceSegments, z )"
                self.addLayerFromSegments( 0.625, 1, 0.1, 1, self.interfaceSegments,
zCenter )
```

8.3.1 Raft Modification Discussion

Having successfully modified the raft settings to produce a consistent raft, it was discovered that by fixing the parameters for the raft, the ability to use alternate materials would have been lost as the process parameters for different material would be different. It was therefore decided to return the raft generation code to the original, so that material changes would be passed through to the raft.

8.4 Automatic gcode Generation

Having modified the raft generation code in Skeinforge, the 20mm Calibration Box was sliced again and the resulting gcode run.

8.4.1 Results of Automatically Coded Box

The result of the automatically coded test box was indistinguishable from the manually coded box. After inspection, the top few layers were cut away from the box using a scalpel, to demonstrate the hexagonal fill pattern used.

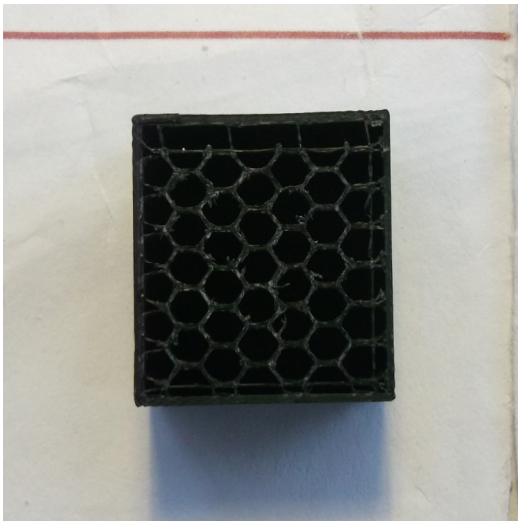


Figure 70: Cut-away of Test Print, Showing Hexagonal Fill Pattern

8.5 Chamber Temperature

It was noticed during testing, that without the fans and blowers running, which were controlled by the chamber, the temperature at the top of the extruder melt chamber was getting hot, caused the filament to get too hot and caused it to "ball-up" around the end of the melt chamber tube and the drive pinions. Further testing was conducted with the door on and the chamber up at 110°C. This switched on the fans and blowers cooling the top of the extruders and prevented the "balling-up" of filament between the melt chamber tube and the drive pinions.

Although the air temperature in the chamber was upto temperature, it was determined that the thermal inertia for the rest of the chamber was significantly

greater. It took approximately 4 hours for the entire chamber to reach a stable temperature. For this reason, it was decided to leave the chamber at temperature. During testing, the chamber was left at 110ºC for over a week without any problems.

9 Conclusion

A quick search on Ebay.com shows an increasing number of legacy commercial grade FDM 3D Printers becoming available for purchase at relatively low cost. Although these machines are usually mechanically sound, the software is generally no longer supported and spares are increasingly hard to find. Those that can be found tend to be very expensive. It is therefore becoming more viable to modify these legacy machines to use more readily supportable open-source hardware and software solutions.

The FDM Vantage X machine donated to Massey University by Wellington Drive Technologies Ltd. was successfully modified to operate using new open-source hardware and software. The use of proprietary 'chipped' canisters was removed to enable further research work to be carried out at the University, using the modified machine to investigate alternative materials and processing methods. The machine was modified to be independent of software, although ReplicatorG was the interface of choice. Gcode can be written in any text editor and converted to S3G to operate the machine. Within the limitations of the extruder geometry, gcode can be executed to create non-linear surfaces. The machine has been demonstrated printing at a resolution of 0.1mm, which shows an approximately 20% improvement in resolution from the machine prior to modification.

The modifications involved changing the drive motors from AC servo motors to DC stepping motors. The 3-phase power supply was re-designed to use the New Zealand standard 3-phase (230V line to neutral) rather than United States standard 3-phase (240V phase-phase), eliminating the need for a heavy duty external transformer. The power supply was split into two sections, the high power 3-phase elements on the Power Supply PCB, which re-used a large number of components from the original Power Supply Board, and the low power digital control elements on the Power Control PCB, based upon the Arduino Mega 2560 (Atmel ATMega2560). The electronics were changed, with

the design based in part upon the MakerBot Industries – Thing-o-matic, comprising a Motherboard, based on the Arduino Mega 2560 (Atmel ATMega2560) and two individual Extruder Controllers, based on the Arduino Uno (Atmel ATMega328p). Firmware for the new electronics was based upon the MakerBot Gen3 firmware, with modifications to match the machine requirements. Software for driving the machine was a modified version of ReplicatorG Version 0037.

Based upon the experience gained through the modification of the machine, there are a number of areas for improvement and further work: The control of the blower for cooling the extruder head should be changed to be under the control of the extruder rather than the chamber. This would require a re-design to the extruder PCB and backplane to add an output to drive a relay and separate the wiring for the fans and blowers; The descision was made early in the project to retain the DC motors in the extruder head, this limited the choice of firmware that could be used. By changing the extruder motors to stepping motors, better control of the extruder could be achieved and newer more feature-rich firmware could be developed; The mass of the extruder head is significantly larger than most open-source 3D printers. This has been evident during testing with high vibration and noise due to resonance. Ways to reduce the overall mass of the extruder head, such as moving the Extruder Controllers onto the stationary part of the machine, or the use of Bowden Cables, should be investigated; The user interface available on the Front Panel Controller should also be improved to enable more functional testing to be carried out without the use of a computer connected, for example loading and un-loading filament, changing pre-heat settings and test extrusions would be useful additions; The code used to generate the raft could also be re-visited to enable the production of a consistent thickness raft, regardless of layer thickness or material selection. This should be achievable by passing the flow-rate and feed-rate parameters for the particular material to calculate the same layer thickness.

References

Allegro Microsystems LLC. (2012). “A3950 DMOS Full-Bridge Motor Driver”. Retrieved 13 September 2012 from http://www.allegromicro.com/~/media/Files/Datasheets/A3950-Datasheet.ashx

ASTM. (2012). F2792-12a, “Standard Terminology for Additive Manufacturing Technologies”. Retrieved 3 September 2014 from http://www.astm.org/Standards/F2792.htm

ASTM. (2013). ISO/ASTM52915-13, “Standard Specification for Additive Manufacturing File Format (AMF) Version 1.1”. Retrieved 15 September 2014, from http://www.astm.org/Standards/ISOASTM52915.htm

Beauregard, Brett. (2012, December 2). On *GitHub.* br3ttb / Arduino-PID-Library. Retrieved 16 October 2014 from https://github.com/br3ttb/Arduino-PID-Library/

Bowyer, Adrian et al. (2014, September 11). “G-code”. Retrieved 14 September 2014 from http://reprap.org/wiki/G-code

Chiou, Kevin. (1996, July 30). “Megatec Protocol Information”. Retrieved 16 April 2013 from http://www.networkupstools.org/protocols/megatec.html

Crocker, Stephen D. (2009, April 6). “How the Internet Got Its Rules”. Retrieved 8 September 2014 from http://www.nytimes.com/2009/04/07/opinion/07crocker.html?_r=2&em&

Crump, Scott. (1992). U.S. Patent 5,121,329, June 9, 1992, "Apparatus and Method for Creating Three-Dimensional Objects" (A system and a method for building three-dimensional objects in a layer-by-layer manner via fused deposition modeling). Retrieved 12 November 2012 from http://www.google.com/patents/US5121329

Crump, Scott. (1999). U.S. Patent 5,866,058, February 2, 1999, "Method for Rapid Prototyping of Solid Models", Sam Batchelder; Scott Crump. (A method for building three-dimensional physical objects with reduced levels of curl and distortion). Retrieved 31 August 2014 from http://www.google.com/patents/US5866058

Engineering Industries Association. (1979, February). “EIA Standard - EIA-274-D, Interchangeable Variable, Block Data Format for Positioning, Contouring, and Contouring/Positioning Numerically Controlled Machines.” Retrieved 14 September 2014, from http://www.standardsebook.com/--techamerica-c-65/techamerica-eia274d1979r1988-pdfelectronic-format-p-98443.html

Faulhaber. (2012). “DC-Micromotors Series 2342 … CR”. Retrieved 18 August 2012, from https://fmcc.faulhaber.com/resources/img/EN_2342_CR_DFF.pdf

Faulhaber. (2011). “Planetary Gearheads Series 23/1”. Retrieved 18 August 2012, from https://fmcc.faulhaber.com/resources/img/EN_23-1_FMM.pdf

Faulhaber. (2012). “Encoders Series IE2 - 1024”. Retrieved 18 August 2012, from https://fmcc.faulhaber.com/resources/img/EN_IE2-1024_DFF.pdf

Flink, James J. (1977). “The Car Culture”. MIT Press ISBN 0-262-56015-1

Future Technology Devices International Ltd. (2012, March). “FT232R USB UART IC Datasheet Version 2.10”. Retrieved 18 August 2012, from http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

Gilloz, Emmanuel. (2012, April 14). “RepRap Family Tree”. Retreived 31 August 2014 from http://reprap.org/mediawiki/images/e/ec/RFT_timeline2006-2012.png

G. W. Lisk Company. (2011, July 15). “Solenoid Assembly – S-2458-48”. Recieved on 10 September 2014 by email from Simon Hopkins (shopkins@gwlisk.com)

Kushner, David. (2011, October 26). “The Making of Arduino”. Retrieved 8 September 2014 from http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino

LaMonica, Martin. (2013, April 23). “MIT Technology Review – 10 Breakthrough Technologies 2013: Additive Manufacturing”. Retrieved 26 April 2013 from http://www.3ders.org/articles/20130426-ge-to-mass-produce-critical-jet-engine-part-use-3d-printing.html

MakerBot Industries. (2010, December 9). “MakerBot Motherboard v2.4”. Retrieved 18 March 2013 from http://www.thingiverse.com/thing:4968

MakerBot Industries. (2010, December 9). “MakerBot Extruder Contoroller v3.6”. Retrieved 18 March 2013 from http://www.thingiverse.com/thing:4969

MakerBot Industries. (2011, March 29). “MakerBot Gen4 Interface Kit v.2”. Retrieved 8 April 2014 from http://downloads.makerbot.com/support/pdf/Thing-O-Matic/Docs/MakerBot Gen4 Interface Kit.pdf

Markillie, Paul. (2012, April 21). “A third industrial revolution”. *The Economist*. Retrieved 21 September 2014 from http://www.economist.com/node/21552901

Maxey, Kyle. (2014, January 15). “Integrating 3D Prining Into Your Product Development Lifecycle”. *Engineering.com. Retrieved 26 April 2015 from http://www.engineering.com/3DPrinting/3DPrintingArticles/ArticleID/6959/Integrating-3D-Printing-Into-Your-Product-Development-Lifecycle.aspx*

Maxim Integrated. (2012, February). “MAX31855 Cold-Junction Compensated Thermocouple-to-Digital Converter”. Retrieved 14 September 2012 from http://datasheets.maximintegrated.com/en/ds/MAX31855.pdf

Metts, Matt. (n.d.). “S3G Protocol (formerly RepRap Generation 3 Protocol Specification)”. Retrieved 27 August 2012 from https://docs.google.com/a/makerbot.com/document/d/1oq-oEogcRxJ91ex4_cJLs8bXPmWoTKJRNPz9Amh0Hb4/edit?pli=1

Midas Components Ltd. (2011, February). “Specification MC44005A6W-BNMLW”. Retrieved 8 April 2014 from http://www.farnell.com/datasheets/1485439.pdf

Netscape Public License. (1998). “Netscape Public License, Version 1.0” Retrieved 31 August 2014, from https://www.mozilla.org/MPL/NPL/1.0/

Newman, Daniel. “[makerbot-users] RepG Help required.” Message to the author. 7 April 2014. Email received from dan.newman@mtbaldy.us

Open Source Initiative. (n.d.). “Licenses by Name”. Retrieved 8 September 2014 from http://opensource.org/licenses/alphabetical

ReplicatorG. (n.d.). “ReplicatorG is a simple, open source 3D printing program”. Retrieved 9 April 2014 from http://replicat.org/

STL (file format). (2009, October 30). “STL 2.0 May Replace Old, Limited File Format” *Rapidtoday.com*. Retrieved 29 July 2013, from http://www.rapidtoday.com/stl-file-format.html

Storr, Wayne. (2014, September). In *Electronics Tutorials*. “Linear Solenoid Actuator Theory and Tutorial” Retrieved 10 September 2014 from http://www.electronics-tutorials.ws/io/io_6.html

Stratasys. (2006, February). “FDM Vantage: Multiple levels of performance from one system”. Retrieved 18 August 2012 from http://www.protech.se/elements/common/files/docs/vantage_06.pdf

Stratasys. (n.d.). *Stratasys Board of Directors*. Retrieved 31 August 2014, from http://www.stratasys.com/corporate/about-us#stratasys-board-members

Szczys, Mike. (2012, September 19). “This hack can refill your Stratasys 3D printer”. Retrieved 19 September 2012 from http://hackaday.com/2012/09/19/this-hack-can-refill-your-stratasys-3d-printer

Tiemann, Michael. (2006, September 19). “History of the OSI” *Open Source Initiative.* Retrieved 8 September 2014 from http://opensource.org/history