Intel Galileo controlled cnc machine

Author

Brendan Scullion

G00302400

Supervisor

Michael Fahy, Galway-Mayo Institute of Technology

Course

Computer and electronic engineering

GMIT

INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO

GALWAY-MAYO INSTITUTE OF TECHNOLOGY

1. ACKNOWLEDGEMENTS

I would like to thank my close friends and family who gave given me support and shown great interest throughout the course of the year.
A special thanks is due to Mike Fahy, Tom Frawley, Sean Coffey, and all the lecturers who have given me advice as well as given me the means to carry out the project. Without the help from these people the project would not have been completed as smoothly If not at all.

2. Table of Contents

3. Summary

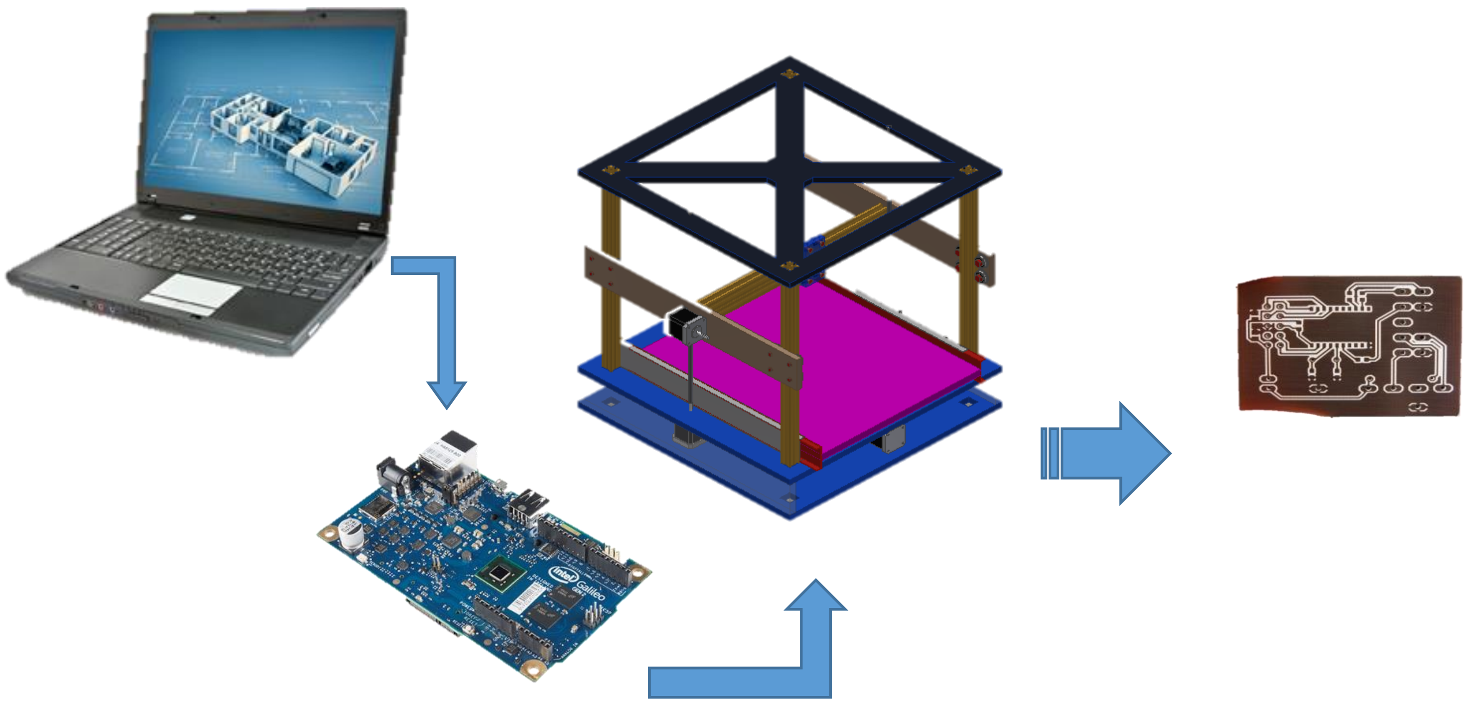
The purpose of this report is to give the reader a detailed account of the procedures, skills and techniques needed to design and build a functioning cnc plotter that is easily altered for use as a cnc mill or 3d printer. The report gives details about the mechanics, electronics and software used throughout the curse of this project.
The design of the project is based around easily obtainable hardware as I myself living in rural Ireland often have trouble finding or sourcing parts for various projects which leads to delays and often termination.
This report proves that it is possible to build a functioning cnc/3d printer using common items found in DIY stores and or recycled appliances such as office printers.
Not only is this an enjoyable albeit time consuming project, the range of skills learned throughout the throughout the build will no doubt aid any engineer in his or her future studies.

4. Introduction

Due to what has been dubbed the “maker movement” or the rise of “maker culture”, the popularity of domestic manufacturing machines such as 3d printers, desktop cnc (computer numerically controlled) machines, PCB prototyping machines etc. has risen exponentially over the last few years. This has brought about a huge new market for affordable and easy to use domestic manufacturing machines. To go with this trend is also the need for people with the knowledge and skills to design, build and operate these machines.

The aim of this project is to research and investigate the current technologies and methods used for domestic cnc machines and to design and build a functioning prototype desktop sized cnc machine controlled using the Intel Galileo gen 2 development board, which can be later used to develop a fully functioning multi use cnc machine/3d printer.

A simple diagram of how such a machine is controlled is shown below



5. Hardware Description

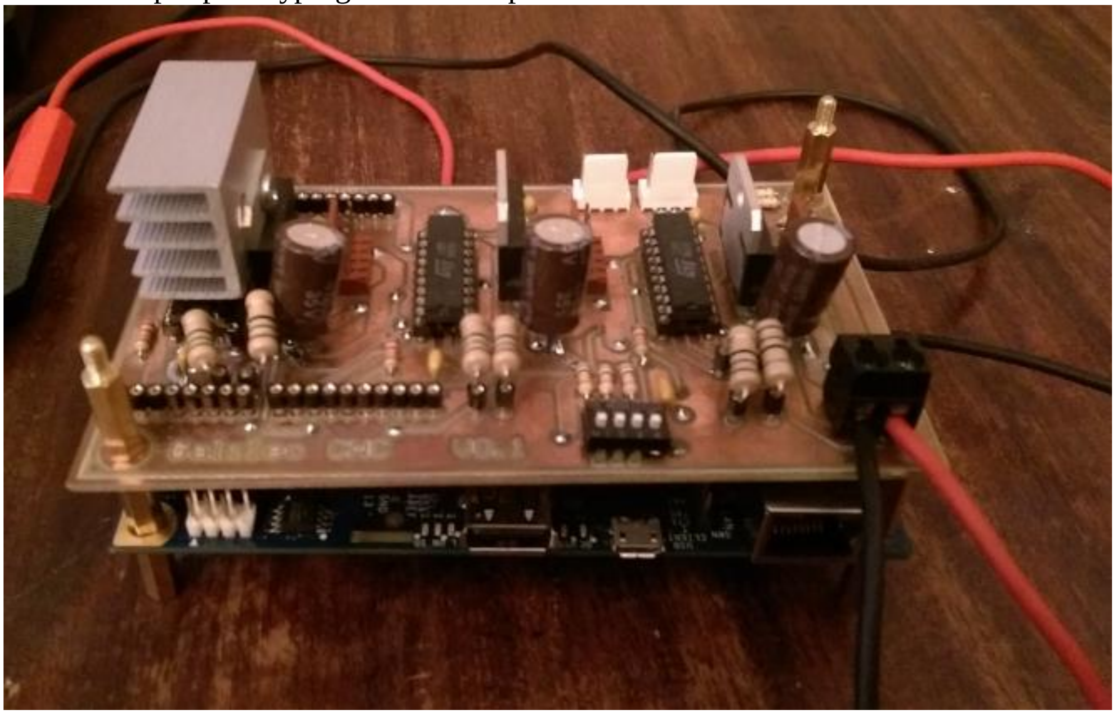
5.1 Intel Galileo gen 2 x86 development board

"The Intel® Galileo Gen 2 board is the first in a family of Arduino*-certified development and prototyping boards based on Intel® architecture and specifically designed for makers, students, educators, and DIY electronics enthusiasts. "

The Galileo is tasked with dealing with the majority of data processing and mathematical calculations, as well as the control of all the electronic components of the machine.
Raw g code is read into the Galileo through a serial Usb port, each line of code is then processed from a string and turned into usable information to which the Galileo then determines the movements of the machine.
The Galileo has a lot of functionality that is unused in this project, however with future developments this extra functionality will come to use. For example future developments could make use of the Galileo's Ethernet port giving the cnc machine network connectivity much like a networked printer found in most businesses and homes.

5.2 Stepper motor interface shield

To interface the Galileo with the stepper motors designed and built a shield to fit on top of the Galileo. I used altium to design the circuit and PCB and the used the LPKF Protomat rapid prototyping machine to plot the circuit onto a double sided PCB.



5.2.1 L297 Stepper controller: The L297 generates four phase signals for a four phase unipolar stepper motor. The advantage of using as this device is that it requires only a clock, direction and mode input. As the phases are generated internally the burden on the Galileo and the amount of programming needed is greatly reduced. In this case only the clock and direction pin are controlled using the Galileo and the mode select pin is controlled using a dip switch. In most cases the L297 is used with monolithic bridge drives such as the l298N or L293E, this build required me to use the L298N.

5.2.2 L298N dual full-bridge driver: Used in pair with the L297, the L298N is an integrated monolithic circuit. It is a high current high voltage dual full-bridge driver designed to accept standard TTL logic levels. It can handle a supply voltage of up to 46v and a total current of up to 4 A. the steppers used in this project use 12v and have a holding current of just over 3 A. the L298N is more than capable of handling these values. Figure 1 shows how the L297 and L298N are connected. It is recommended in the datasheet that for the sensing resistors labelled R6 and R7 that a 1 ohm sense resistor is used. As I was not able to obtain a 1 ohm resistor that could handle the current so a 4.7 ohm 3W wire-wound type resistor was used and worked without any flaws. Due to excessive heat being generated from the IC a heat sink was added to dissipate some of the heat.

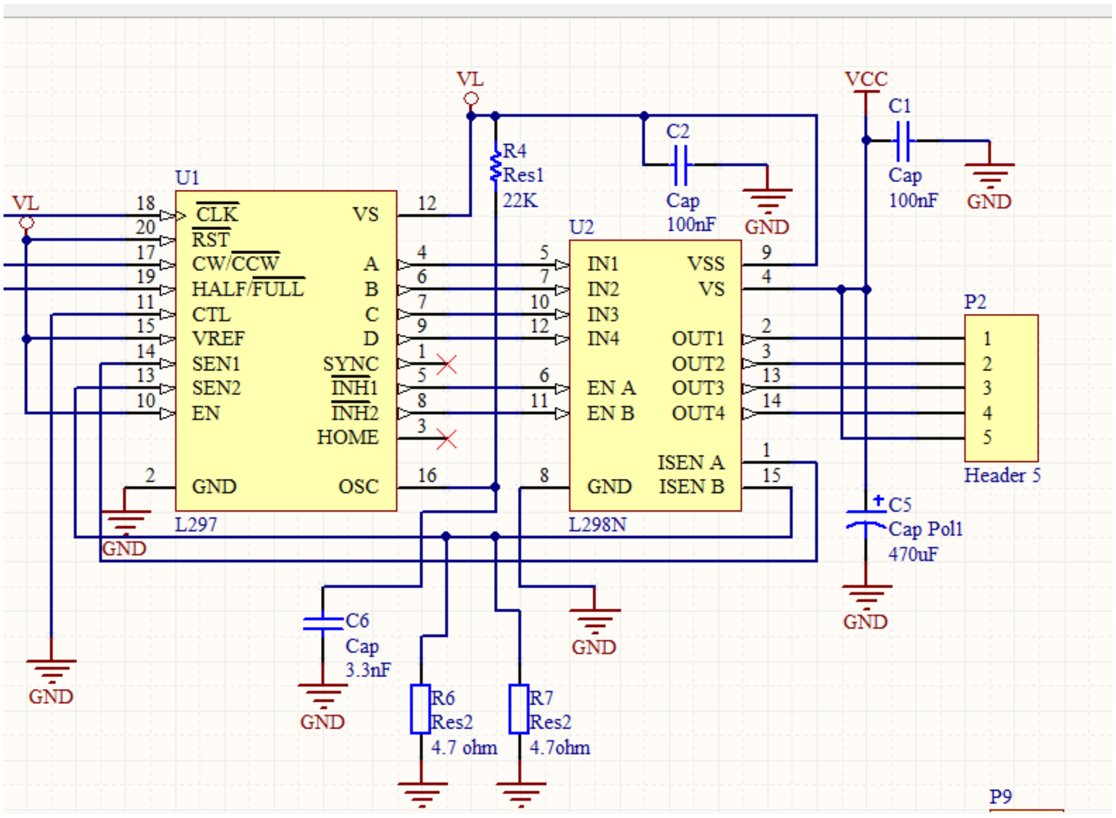


Figure 1

Capacitors marked C1 and C2 are for noise suppression and a 470uF capacitor marked C5 acts as a reservoir to stabilize the current when the motor is switched on.

The circuit above is my version of the recommended circuit given in the datasheet of both the L297 and L298N

5.3 Stepper motors

5.3.1 Astrosyn MY5602 size 14 Stepper Motor: The smallest of the three motors used in this project, the MY5602 is a hybrid unipolar stepper. As the y-axis takes the least amount of effort to move this is where this motor is used. This motor is rated at 0.75A per phase

5.3.2 Astrosyn 37-0514 size 17 stepper motor: As the x and y-axis take more effort to move a larger stepper motor is needed. At a speed of <300 rpm the size 17 motor produces more than enough torque to turn the screw thread and move the x-axis and y-axis. This motor has a holding current of 1.2 A per phase.

Both of the steeper motors above require 200 steps to make 1 full revolution or 1.8 degreesper step. This coupled with a M8*1.25mm threaded rod gives a minimum travel of 6.25µm for 1 step.

5.4 Power

Due to the complexity of this project I decided that building my own power supply from scratch would be too time consuming, I opted to use a hacked ATX power supply from a Dell GX280, this power supply can supply up to 14A at 12v which is more than enough power my project which draws a maximum current of 10A at 12v.

In order to use the power supply in m project I first had to make a few modifications. First of all in order to turn the power supply on the green pin (14) need to be brought low. To do this I connected a toggle switch between pin 14 and ground.

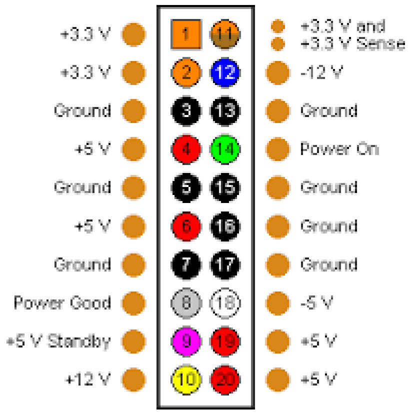


Figure 2 (image taken from www.pcguide.com)

Some ATX power supplies such as the one I used sense is there is a load on the 5v line and if no load is detected the power supply is turned off again automatically. To work around this function I soldered a 560Ω resistor across one of the 5v lines and ground. This was enough to trick the power supply into thinking it was connected to a computer and allowed me to use it in my project.

To finish of the power supply I cut all the output wires to manageable length that could fit inside the atx case. I then drilled 5 holes in the side of the case to fit the

toggle switch and 4 chassis mounted terminals for 3.3v, 5v, 12v and GND. The finished product is neat and can also double as a bench power supply for other projects.
I used information from http://www.instructables.com/id/ATX--%3E-Lab-Bench-Power-Supply-Conversion/ to build this power supply.

5.5 Mechanical hardware

Probably the most time consuming parts of the build was building the mechanical components of the machine. To get the best results from any cnc type machine. Each component must be designed and selected to minimize unwanted movement, vibration and various other factors that could interfere with the accuracy of the finished product.

5.5.1 Frame

Initially I designed the frame to be made out of 20x20mm aluminium extrusion and bolted together. As I could not obtain this at a reasonable price I opted to go with a heavier steel design with welded joints instead of bolted. I used 25x25mm mild box steel for the main part of the frame.
Procedure for building frame:

1. Cut all the steel into the required lengths, each piece was measured closely and checked for squareness as small inaccuracies here would translate into larger problems later on.
2. Pieces marked for components such as drawer sliders bearing holders etc. and 6mm holes drilled using pillar drill and threaded using a M6 tap
3. As the heat generated from welding causes the steel to bend and warp a jig is needed to hold everything together. I used a sheet of plywood and screws to hold the steel while welding.
4. Frame was made I two sections. Two ‘H’ shaped sides were first made, then these two sides were welded together spaced apart by two equal lengths of steel box. This creates the main frame shape for the cnc.
5. Components such as the drawer slides and bearing blocks were then bolted in place using M6 bolts. The 22mm bearings are held in place using 20mm PVC electrical conduit clamps and the height adjusted by placing layers of PVC under the bearings.

5.5.2 X-axis

The x axis consists of two 350mm long roller bearing drawer sliders similar to those found in mechanics tool boxes, a ‘H’ shaped base made out of 44x30mm red deal, and a length of M8x1.25 stainless steel threaded rod.
Procedure for building x axis:

1. Cut red deal into lengths and screwed together into ‘H’ shape. An 8mm hole is drilled into the centre of the cross member and an M8 nut is then pressed and glued into the hole. This is attached to the drawer sliders using wood screws.
2. The threaded bar is screwed through the nut in the centre and cut to the appropriate length. Approx. 400mm.
3. The bearing are then adjusted to be in line with the rod and tightened in place. Nuts are placed both side of the bearings and tightened onto the rod. The threaded rod should turn freely and move the x-axis back and forward.
4. A bed of mdf is cut to size and bolted to the red deal using M6 nuts.

5. The stepper motor is attached below the bearing block on the front side using M6 bolts and a steel plate.
6. The motor is coupled with the threaded rod using two 16 tooth pulleys and a rubber timing belt. The belt I used is from a canon laser printer, a salvaged also from the same printer is used to keep tension on the belt to prevent slippages.

5.5.3 Y/Z-axis

As the Y-axis and Z-axis are one unit and are therefore made together. The unit consists of a frame made from 12mm MDF;

1. Two aluminium rails are bolted to the sides with a nut in the middle.
2. Two 8mm hardened steel rods along with 4 x 8mm linear bearings are placed parallel with each other in the centre of the unit and attached to the mdf using purpose made clamps.
3. A threaded rod with bearings at each end is attached alongside the hardened steel rods and screwed in place.
4. The stepper motor is attached to the frame using a special made bracket and coupled with the threaded rod using two 16 tooth pulleys and a rep-rap timing belt.
5. The tool holder is made out of salvaged pieces of various printers and attached to the linear bearings using M5 bolts and steel plates, a bracket was made up to attach it to a nut that travelled along the threaded bar.
6. For the Z-axis the unit was attached to two hanging threaded bars which are secured by bearings at the top of the steel frame.
7. The two threaded bars are connected using rep-rap timing belt and pulleys to a stepper motor that is attached to the top of the frame.

6. Software Description

This section is split u into two sub-sections one for the Galileo program and one for the windows application.

6.1 GalileoCNC windows application

The main purpose of this program is to read g-code or Gerber code from a file and send the code line by line to the machine via serial Usb. The application also contains various functions for setting up the machine. The code is explained in this section

6.1.1 Main loop

The code for communicating with the serial port is a mixture of code from Microsoft's website https://msdn.microsoft.com/en-us/library/windows/desktop/ee663264(v=vs.85).aspx

and https://batchloaf.wordpress.com/2013/02/13/writing-bytes-to-a-serial-port-in-c/

Connecting to com port.

```
 int baudrate = 9600;
int dev_num = 50;
char dev_name[MAX_PATH];
DCB dcbSerialParams = {0};
COMMTIMEOUTS timeouts = {0};

 printf("Searching serial ports...\n");
while(dev_num >= 0)
{
   printf("\r                    ");
   printf("\rTrying COM%d...", dev_num);
   sprintf(dev_name, "\\\\.\\COM%d", dev_num);
   hSerial = CreateFile(
               dev_name,
               GENERIC_READ|GENERIC_WRITE,
               0,
               NULL,
               OPEN_EXISTING,
               FILE_ATTRIBUTE_NORMAL,
               NULL );
   if (hSerial == INVALID_HANDLE_VALUE) dev_num--;
   else break;
}

if (dev_num < 0)
{
   printf("No serial port available\n");
   return 1;
}
```

```
    printf("OK\n");

    // Set device parameters (9600 baud, 1 start bit,
    // 1 stop bit, no parity)
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
    if (GetCommState(hSerial, &dcbSerialParams) == 0)
    {
        printf("Error getting device state\n");
        CloseHandle(hSerial);
        return 1;
    }
    //dcbSerialParams.BaudRate = CBR_38400;
    dcbSerialParams.BaudRate = baudrate;
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;
    if(SetCommState(hSerial, &dcbSerialParams) == 0)
    {
        printf("Error setting device parameters\n");
        CloseHandle(hSerial);
        return 1;
    }
    // Set COM port timeout settings
    timeouts.ReadIntervalTimeout = 50;
    timeouts.ReadTotalTimeoutConstant = 50;
    timeouts.ReadTotalTimeoutMultiplier = 10;
    timeouts.WriteTotalTimeoutConstant = 50;
    timeouts.WriteTotalTimeoutMultiplier = 10;
    if(SetCommTimeouts(hSerial, &timeouts) == 0)
    {
        printf("Error setting timeouts\n");
        CloseHandle(hSerial);
        return 1;
    }
```

```
Declare variables such as
Baudrate
Device name
Array for dcbSerialParams
Array for timeouts

While device number is greater than 0
    Serial handle = creatfile( device number, device params read/write etc. ).

    If serial handle = INVALID_HANDLE_VALUE
        Reduce device number and try again.
    Otherwise
        Exit loop

If no serial port available
    Print “no serial port available” and exit program.

Initialise dcb serial params such as baud rate, parity, stop bits etc..
```

```
Set commstate
      If 0 is returned print error, close handle and exit.

Initialise timouts
Set timouts
      If 0 is returned print error, close handle, and exit.
```

Command line style interface.

```
char *cmd = NULL;
  int keepGoing = 1;

//***************************************************************
**********************************************

  char cmdLine[200];
  while(keepGoing == 1)
  {
    printf(">>");
    gets(cmdLine);
    cmd = strtok(cmdLine, " ");

    if(cmd!=false)
    {
      if(strcmp(cmd, "help")== 0)
      {
        help();
      }
      else if(strcmp(cmd, "getg")== 0)
      {
        getgcode(dev_name);
      }
      else if(strcmp(cmd, "setup") == 0)
      {
        setup(dev_name);
      }

      else if(strcmp(cmd, "exit") == 0)
      {
        break;
      }
      else
      {
        printf("Unknown command!\n");
      }
      printf("\n");
    }
    }
```

```
Declare pointer to char array for storing command

Print '>>' read command and store in array
If command is not empty
        Compare command to functions
        if command == "help"
                Display help dialog;
        If command == "getg"
                Call getgcode function
        If command == "setup"
                Call setup function
        If command = "exit"
                exit loop
        If command not recognised
                Print error and continue
Print next line character
```

6.1.2 Major functions

The major functions are all the functions that are called directly from main.

6.1.2.1 Heading()

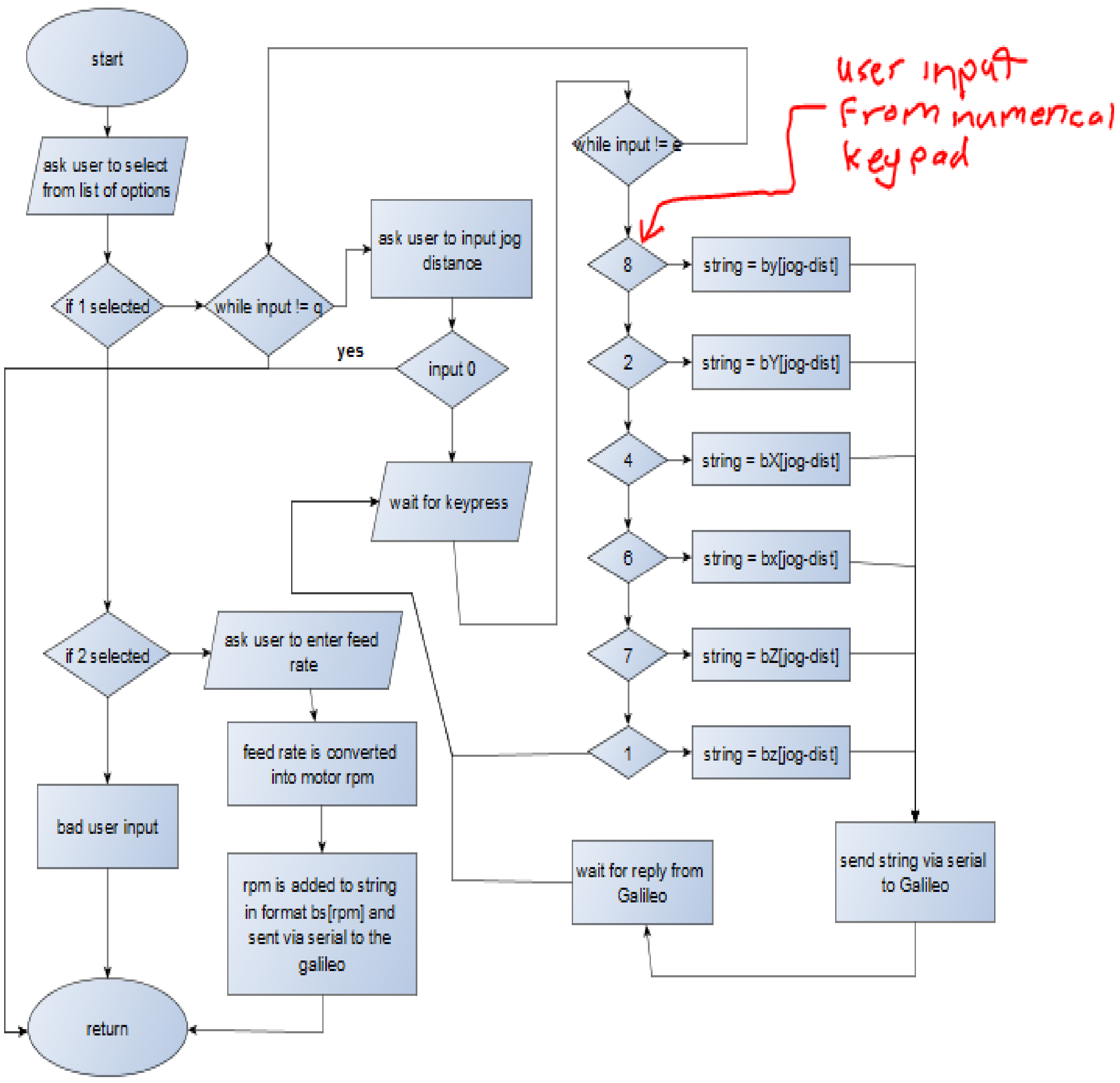
Displays application heading containing logo, welcome message and some information about the application

```
void heading() //print header logo and intro text
{
printf("|======================================================
====================|\n");
    printf("|\t            ___ __          _______   ________               |");
    printf("\n|\t   _____ ______ _/ (_) /__  _____  / ____/ | / / ____/               |\n");
    printf("|\t  / __ `/ __ `/ / / / _ \\/ __ \\/ /   /  |/ / /                   |\n");
    printf("|\t / /_/ / /_/ / / / /  __/ /_/ / /___/ /|  / /___                 |\n");
    printf("|\t \\__, /\\__,_/_/_/_/\\___/\\____/\\____/_/ |_/\\____/                  |\n");
    printf("|\t/____/                                                  |\n");
    printf("|=================================================by
brendan scullion=======|\n");

   printf("\nWelcome, this is a program to to control a 3d printer/cnc machine using an
\narduino or galileo micro controller.\n\n");
   printf("use the [help] command for a list of commands.\n");
}
```

6.1.2.2 Setup function

This function send a string containing setup information to the machine. Command strings can also be sent to manually move the tool to the home position.

Flow chart:


start
ask user to select from list of options
if 1 selected
while input != q
ask user to input jog distance
yes
input 0
wait for keypress
while input != e
user input From numerical keypad
8
string = by[jog-dist]
2
string = bY[jog-dist]
4
string = bX[jog-dist]
6
string = bx[jog-dist]
7
string = bZ[jog-dist]
1
string = bz[jog-dist]
if 2 selected
ask user to enter feed rate
feed rate is converted into motor rpm
rpm is added to string in format bs[rpm] and sent via serial to the galileo
bad user input
return
wait for reply from Galileo
send string via serial to Galileo

Code:

```
printf("please select option from the list below\n 1\t adjust tool position.\n 2\t adjust
feed rate(mm/s).\n");
   switch(_getch())
   {
      case '1':
         while(ch!='q')
         {
            printf("use the up and down numerical keys to move tool head up and
down(press q to exit)\n");
            printf("enter jog distance ( 1000 = 1cm)(0 to exit):"), scanf("%i", &dist);
            if(dist == 0){
               return;
            }
            ch = '0';
            while(ch!='e'&&ch!= 'q')
            {
               ch = _getch();
               if(ch == '7'){ // Z axis up
                  printf("Z up\n");
                  sprintf(instruct, "bZ%d*\n",dist); // creats string if format [function
identifier][axis]<distance to move>
               }
               if(ch == '1'){ // Z-axis down
                  printf("Z down\n");
                  sprintf(instruct, "bz%d*\n",dist);
               }
               if(ch == '2'){ // y-axis back
                  printf(" y++ \n");
                  sprintf(instruct, "bY%d*\n",dist);
               }
               if(ch == '8'){ // Y-axis forward
                  printf("y--\n");
                  sprintf(instruct, "by%d*\n",dist);
               }
               if(ch == '6'){ // X-axis right
                  printf("X -- \n");
                  sprintf(instruct, "bX%d*\n",dist);
               }
               if(ch == '4'){ // X-axis left
```

```
                printf("X ++\n");
                sprintf(instruct, "bx%d*\n",dist);
              }
              sendSerial(instruct, devName); // send instruction to galileo
              read_serial(serialIn, devName); // waits for reply from machine and adds
it t char array (serialIn)
              printf("%s", serialIn); // print reply
            }
          }
          break;
        case '2':
          printf("enter feed rate( < 6.25mm/s recomended):"),scanf("%f", &feed);
          if(feed > 6.25){
            MessageBeep(0);
            if(MessageBoxA(0,"warning: feed rate greater than 6.25 can result in loss of
position during process!\ndo u still want do use this speed?", "warning",
MB_YESNO) == 7)
            {
              printf("enter feed rate( < 6.25mm/s recomended):"),scanf("%f", &feed);
            }
          }
          rpm = feed*48; // feed rate is converted to motor rpm
          sprintf(instruct, "bf%d*", rpm);
          sendSerial(instruct, devName);
          Sleep(100);
          read_serial(serialIn, devName);
          break;
        default:
          printf("bad user input, please choose from options above");
          break;
    }
}
```

6.1.2.3 Help function

This function reads a txt file called help.f, the help file is stored in the same directory as the application. The advantage of using a help function such as this is that the help file can be altered and updated after the application has been compiled.

Pseudo code:

```
Create char array ch to store 1 line of the file.

Open file and associate it with a file handle.
If file can’t be opened or does not exist
        Print error and exit.
While line != null
        Print line to screen.
        Move to next line.
```

```
Close file.
Exit function.
```

```
int help()
{
   char ch[MAX_STRING_LENGHT]; // char array to store one line of the help file.

   file = fopen("helpfile.f","r"); // open help file
   if(file == NULL) // if the file cannot be opened or does not exist
   {
      perror("Error while opening the helpfile.\nmake sure help file exists!\n");
      return 1;
   }
   while(fgets(ch,100,file)!=NULL)// print out file line by line until end of file
   {
      printf("%s",ch);
   }
   fclose(file); // clode file
   return 0; // exit function
```

6.1.2.4 Getgcode

This function allows the user to enter the name of a file containing g-code, the file is opened and the contents are read by another function which determines the minimum x and y coordinates. These coordinates can be used to set offsets so that the image is closer to the origin. The code is then read line by line and sent as a string via serial port to the Galileo and waits for a reply. When a reply is received the next line of code is then sent and this process continues until the end of the file, a message pops up letting the user know the process has finished and the function is returned.

Pseudo code:

```
Declare
Char arrays for:
-Line of g-code        “gline”
-Serial in       “serialIn”
-G-code file name      “file_name”
-Popup box message   “message”
Pointer to char array for parsing
Integer
-line_total = 1
-current line = 0

Ask user to enter name of file they wish to use and store it in filename array

Calculate image position and get offsets using getoffset() function
```

```
Open file (read only)
If file open returns null
    Print error and return

While
    While line being read is not = null
        Read line from file and store to char array gline
        Update current line integer variable
        Print gline to screen

        Send gline to Galileo via serial
        If send line function returns 1
            Return

        Wait for response from Galileo
        Store response to array serialIn
        If read serial function returns 1
            Return.
        Print serialIn to screen.
        Display progress bar
        If line is a G54 aperture change code
            Parse string for aperture number G54D[aperture number]*
            Display pop up box prompting user to change aperture.

    When file is finished display message box notifying user that the job is
    complete
    Break while loop
Close file
Return function
```

Code:

```
void GetGcode(char *devName)//read g code and set up for machine
{
   char gline[MAX_STRING_LENGHT];
   char serialIn[MAX_STRING_LENGHT];
   char file_name[20];
   char *pch;
   char message[20];
   unsigned int line_total = 1, current_line = 0;

  printf("Enter the name of file you wish to use\n");
  gets(file_name);
  //calculate image position and then calculate best offset for maximum space on
board
  getOffset(file_name, // file to be calculated
        devName,   //device to send date to
         &line_total); // number of lines of code
  file = fopen(file_name,"r"); // read mode

  if( file == NULL )
  {
    perror("Error while opening the file.\n");
    return;
  }
  printf("The contents of %s file are :\n", file_name);

  while(1){
      while(fgets(gline,100,file)!=NULL){
         current_line++;
         printf("%s\n",gline);//print one line of the gcode
         Sleep(10);
         if(sendSerial(gline, devName) == 1){ // sent gcode to cnc machine
            return;
         }
         if(read_serial(serialIn, devName) == 1) // wait for response from machine and
print
         {
            return;
         }
         printf("%s\n", serialIn);
         progressBar(current_line, line_total, 50);
         if(gline[0] == 'G'&& gline[1] == '5'&&gline[2] == '4'){
            pch = strtok(gline,"G54D * ");
            sprintf(message, "insert aperture no: %s", pch);
            messageup("aperture change", message);
         }
      }
      messageup("finished!","process has finished."); // when it reaches the end of the
file
      break;
  }
  fclose(file); // close file
}
```

6.1.3 Minor functions

These functions are not called functions which are called within the major functions

6.1.3.1 Messageup

This function sound a beep and then displays a message box containing information
Two of windows API's are used in this function, messageBeep (), and messageBox ()

```
void messageup(char *title, char *body)
```

```
{
   MessageBeep(0); // produce a beep sound
   MessageBox(0,
                  body,    // messgeto display n body of box
                  title, // title string
                  MB_OK // box contains only 1 o button
                   );
}
```

6.1.3.2 sendSerial(instruction, device name)

This function is passed a string and device name, the string is then sent to the device determined by device name via serial.

Pseudo code:

```
Declare
Array text_to_send
Array buffer

Empty buffer
Copy passed string into buffer

If buffer is empty
        Return
Declare integers n and m and initialise to 0
While n is less than buffer length
        Text_to_send[m] = buffer[n]
        Increment m and n

Set last character in text_to_send to '\0'

Declare variables to store bytes_written and total_bytes_written =0

While total_bytes written is less that size of text to send
        Write character to serial port
        If write function returns false
                Print error, close handle, and return.
        Update total_bytes _written

Return function
```

Code:

```
int sendSerial(char *instr, char *devName)
{
   /***********************send
string*********************************************/
   unsigned char text_to_send[MAX_PATH];
   char buffer[MAX_STRING_LENGHT];
   strcpy(buffer, ""); // empty buffer
   strcpy(buffer, instr);
   // Check that some text to send was provided
   if (strlen(buffer) == 0)
   {
      return 1;
   }
   int n = 0, m = 0;
   while(n < strlen(buffer))
   {
      text_to_send[m] = buffer[n];
      m++; n++;
   }
   text_to_send[m] = '\0'; // Null character to terminate string

    // Send specified text
   DWORD bytes_written, total_bytes_written = 0;
  //printf("Sending text... ");
   while(total_bytes_written < m)
   {
      if(!WriteFile(hSerial, text_to_send + total_bytes_written,
         m - total_bytes_written, &bytes_written, NULL)){
         fprintf(stderr, "Error writing text to %s\n", devName);
         CloseHandle(hSerial);
         return 1;
      }
      total_bytes_written += bytes_written;
   }
   //fprintf(stderr, "\t\t%ld bytes written to %s\n", total_bytes_written, devName);
   return 0;
}
```

6.1.3.3 readSerial(message, device name)

This function waits for a serial message from a specified device and then stores it to a char array. When the device has finished sending its message the function returns
If the function returns 0 the function was successful if 1 is returned an error occurred.

```
Declare array buffer
Declare variables bytes received = 40, written =0
Integer index = 0

Empty buffer.

While buffer != null
      Read byte from device
      If read unsuccessful
            Print error and return 1
      If bytes_recieved
            Write incoming bytes to buffer
            For index = 0 until buffer [index] == '*'
```

```
                Message[index] = buffer[index]
            Message[index]  = ‘\0’
            Return 0
        If keypress detected
            If key pressed is ‘q’
                Return 1
```

```
int read_serial( char *message, char *devName )
{
   char buffer[MAX_STRING_LENGHT];
   DWORD bytes_recieved = 40, written = 0;
   int index = 0;
   strcpy(buffer,""); //empty buffer

   while(buffer!=NULL){ // wait untill serail message recieved
      if(ReadFile(hSerial, buffer,sizeof(buffer), // read serial
            &bytes_recieved, NULL ) == FALSE){
               perror("ERROR READING SERIAL");
            return 1;
            }
      if(bytes_recieved){             // if something to read
         WriteFile(screen, buffer, bytes_recieved, &written, NULL);
         for(index = 0; buffer[index] == '*'; index++){
            message[index] = buffer[index];
         }
         message[index] = '\0';
         return 0;
      }
      if(kbhit()){
         if(getch() == 'q')
         {
            perror("\nUSER ABORT!!\n");
            return 1;
         }
      }
   }
}
```

6.1.3.4 progress bar(current position, total, width)

This function is somewhat buggy and only displays on occasion. It is passed current line position and total amount of lines. It then works out what percentage is completed and displays this in a progress bar. The width of the progress bar can be set using the third argument. This function is an adaptation of the function found at https://www.ross.click/2011/02/creating-a-progress-bar-in-c-or-any-other-console-app/

<u>Pseudo code:</u>

```
If current position = total
        Return

Float ratio = current position/total position
Int c = ratio*width

Print start of progress bar "[percent complete] % [
For current position = 0 until current position = c
        Print "="
For current position = c until current position =width
        Print " "

Print end "]"

Return
```

<u>Code:</u>

```
void progressBar(unsigned int i, unsigned int n, unsigned int width)
{
   //when 100% is reached fuction is exited
  if ((i == n) && (i % (n/100+1) != 0) )
      return;

   float ratio  =  i/(float)n;
   int   c      =  ratio * width;
    // Show the percentage complete.
   printf("%3d%% [", (int)(ratio*100) );

   // Show the load bar.
   for ( i=0; i<c; i++){ // progress so far
     printf("=");
   }

   for ( i=c; i<width; i++){ // remainder filled with spaces
     printf(" ");
   }

   printf("]\n");
}
```

6.1.3.5 getoffset(filename, device name, total lines)

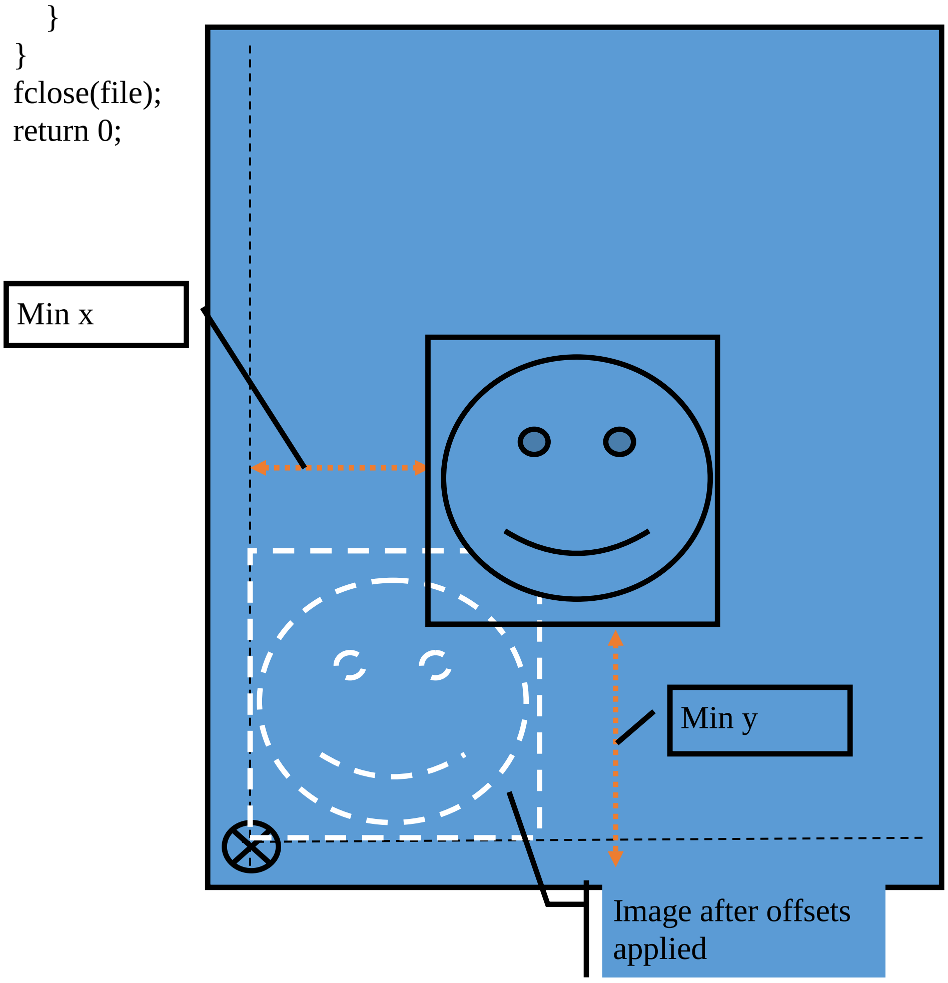
This function reads through each line of a Gerber file and finds the minimum x and y coordinates. It uses these coordinates to create offsets so that the image is closer to the origin allowing a larger file to be plotted in the space available. This function also counts the amount of lines in the file.

```
Declare variables
Start =false
Yy
X, Y
Xmin, Ymin
Count


Open file
If file doesn’t open
    Print error and return
Print “calculating offsets”
While line read isn’t = null
    Update counter

    If string starts with ‘X’ or’Y’
        If string contains ‘Y’
            Yy = true
        If string starts with ‘X’
        Pch  = whatever is after ‘X’ and before ‘Y’ or ‘D’
        X = (long)pch
        If yy = true
            Pch = whatever is between ‘Y’ and ‘D’
            Y = (long) pch
        Else if string starts with ‘Y’
            Pch =  whatever is between ‘Y’ and ‘D’
            Y = (long) pch
    If start is false
        Xmin = x, ymin  = y
        Start = true
    If x < xmin
        Xmin = x
    If y < ymin
        Ymin = y
    Yy = false

TotalLines = count

Print minimum x and y coordinates
Offsets = minimum position – 1000
Ask user if he or she would like to use these offsets
If user chooses yes
    Create string in format box[x offset]y[y offset]*
    Send string to device via serial
    Wait for reply from device
Close file
Return 0
```

<u>Code:</u>

```
int getOffset(char *filename,char *devName, unsigned int *totalLines)// find
minimum x and y coordinates and set offset
{
    char gline[MAX_STRING_LENGHT];
    char serialIn[MAX_STRING_LENGHT];
    char *pch;
    char select;
    char str[MAX_STRING_LENGHT];
    bool yy = FALSE;
    bool start = FALSE;
    unsigned long x = 0, y = 0, xMin = 0 , yMin  = 0;
    unsigned int count = 0;

    file = fopen(filename,"r"); // read mode
    if(file == NULL ){
      perror("Error while opening the file.\n");
      return 1;
    }
    printf("\ncalculating offsets....\n");
    while(fgets(gline,100,file)!=NULL)
    {
        count++;
        if (gline[0]=='X'||gline[0]=='Y'){
            if(strpbrk(gline,"Y")!= NULL){
                yy = true;
            }
           if(gline[0]=='X'){
                pch = strtok(gline,"X Y");
                x = atol(pch);

                if(yy == true){
                    pch = strtok(NULL,"Y- D");
                    y = atol(pch);
                }
            }
            else if(gline[0]=='Y'){
                pch = strtok (gline, "Y- D *");
                y = atol(pch);
            }
            if(!start){
            xMin = x, yMin = y;
            }
            start = TRUE;
            if(x<xMin){
                xMin = x;
            }
            if(y<yMin){
                yMin = y;
```

```
          }
          yy = FALSE;
        }
    }
    *totalLines = count;
    printf("Min x position = %lu , Min y position = %lu\n", xMin, yMin);
    xMin = xMin - 1000;
    yMin = yMin - 1000;
    printf("Most appropriate offsets are x -%lu y -%lu\nwould you like to use these 
offsets?(y/n):",xMin, yMin);
    scanf("%c", &select);
    if(select == 'y')
    {
        sprintf(str, "box%luy%lu*", xMin, yMin);
        if(sendSerial(str, devName) == 1){
            return 1;
        }
        if(read_serial(serialIn, devName) == 1){ // wait for response from machine and 
print
            return 1;
        }
    }
    fclose(file);
    return 0;
}
```



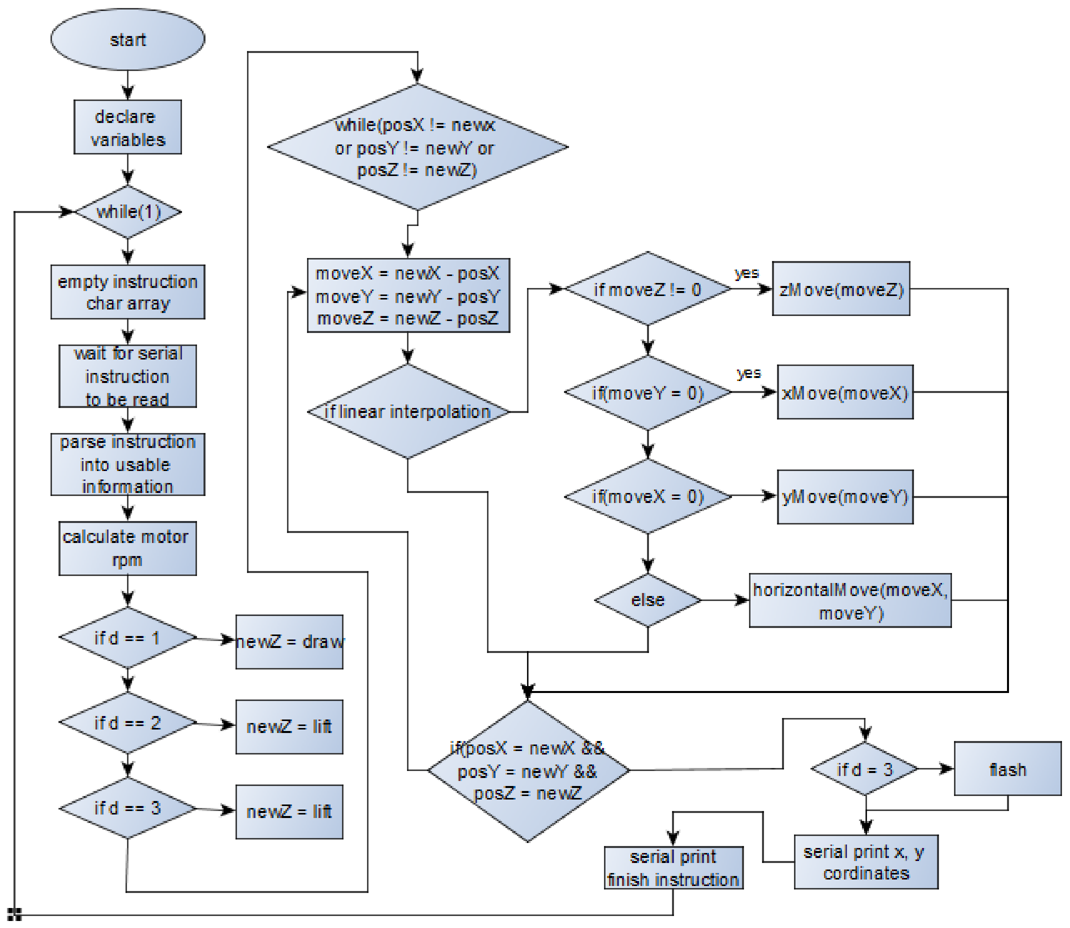
6.2 Code for upload to Galileo development board

6.2.1 Setup()

The setup function in this program is straight forward,
Open serial port and set baud rate
Set pin modes,
And print a message to the serial port

```
void setup()
{
        Serial.begin(9600); // start serial with baudrate 19200
      // initialise pins
      pinMode(13, OUTPUT);
        pinMode(12, OUTPUT);
        pinMode(11, OUTPUT);
        pinMode(10, OUTPUT);
        pinMode(9, OUTPUT);
        pinMode(8, OUTPUT);
        pinMode(7, INPUT);
        pinMode(6, INPUT);

      Serial.println ("\nGalileo cnc V1.0 by Brendan scullion");
}
```

6.2.2 Void loop()


start
declare variables
while(1)
empty instruction char array
wait for serial instruction to be read
parse instruction into usable information
calculate motor rpm
if d == 1
newZ = draw
if d == 2
newZ = lift
if d == 3
newZ = lift
while(posX != newx or posY != newY or posZ != newZ)
moveX = newX - posX
moveY = newY - posY
moveZ = newZ - posZ
if linear interpolation
if moveZ != 0
yes
zMove(moveZ)
if(moveY = 0)
yes
xMove(moveX)
if(moveX = 0)
yMove(moveY)
else
horizontalMove(moveX, moveY)
if(posX = newX && posY = newY && posZ = newZ
if d = 3
flash
serial print x, y cordinates
serial print finish instruction

In short the void loop function waits for an instruction via serial, that instruction is then processed and translated into movements of the machine or setup routines. When the instruction is finished it is emptied and the function waits for the next.

__Code:__

```
void loop()
{

   long moveX = 0;
   long moveY = 0;
   long moveZ = 0;
   unsigned long stepDelay = 1;
   unsigned int steppRpm = 240;

   //keeping track of position
   unsigned long posX = 0;  // current position (in steps)
   unsigned long newX = 0;  // destination position (in steps)
   unsigned long posY = 0;
   unsigned long newY = 0;
   unsigned long xOffset = 0;
   unsigned long yOffset = 0;
   long posZ = 0;
   int newZ = 0;
   float unitconverter = 1;
   unsigned int interpolation = 1; // 1 for linear, 2 for clockwise circular, 3 for
counterclockwise circular
   unsigned int d = 1, dps = 4;    // 4 for FSLAX24Y24 etc..
   unsigned int stepsPerUnit = 160;  //160 per mm
   const int instructionsize = 50;
   unsigned int index = 0;
   int draw = 0, lift = 300;
   struct apeture D[30];
   char *instruction;
   instruction = (char*)malloc(instructionsize*sizeof(char)); // Allocate some space
for the string
   newZ = lift; // always start in the lifted position

   while(1)
   {
      strcpy(instruction, ""); // clear instruction array
      readserial(instruction, instructionsize, index); // read in one line of gerber file
      //parse string into useable data and instructions and convert chars to integer value
      parseString(
               D,              // structure for storing apeture data
               instruction,      // g code or gerber code
               &dps,            // decimal points in coordinates
               &stepsPerUnit,    // amount of steps it takes the motor to move one unit of
measurement mm/inch
               &interpolation,   // are u drawing a circle or a straight line
               &newX, &newY,     // destination x,y coordinate
               &d,               // what do do with the z axis
               stepDelay,          // delay between steps to controle speed of movement
               &xOffset, &yOffset, // offsets for x, y
               &steppRpm           // rpm of stepper motor
               );
      // process offsets
      newX -= xOffset;
      newY -= yOffset;
      calcDelay(steppRpm, &stepDelay);

      if(d == 1)
        newZ = draw; // set newZ to draw height
      if (d == 2)
        newZ = lift; // set newZ to lift height
      if(d == 3)
        newZ = lift;//for D03 flashing apeture must be in the lifted position prior to
moving position

     while(posX != newX || posY != newY || posZ != newZ)
        {
```

```
        //calculate distance to move
        moveX = newX  - posX ;
        moveY = newY - posY ;
        moveZ = newZ - posZ;

        //decide which function to use dependng on slope of line and direction

        // for linear interpolation
       if(interpolation == 1)
         {
            if(moveZ>0||moveZ<0)
            {
               zMove(moveZ,stepDelay, &posZ);
            }
            if(moveY == 0)
            {
               xMove(moveX,stepDelay,&posX, stepsPerUnit);
            }
            else if(moveX == 0)
            {
               yMove(moveY,stepDelay,&posY, stepsPerUnit);
            }
            else
            {
               horizontalMove(moveX, moveY,  // x and y distance machine has to
move
                           stepDelay,    // delay for speed control
                           newY, newX,    // destination x, y coordinates
                           &posY, &posX,  // current X,Y coordinates
                           stepsPerUnit  //steps per unit of measurement
                           );
            }
          }
        if(posX==newX&&posY==newY&&posZ==newZ)
        {
          if(d == 3)
          {
           flash(draw, lift, &posZ, stepDelay);
          }
          Serial.print((float)posX/stepsPerUnit),Serial.print(" y:"),
Serial.println((float)posY/stepsPerUnit);
        }
       }

     Serial.print("Instruction finished X:");

   }
   free(instruction);
```

}

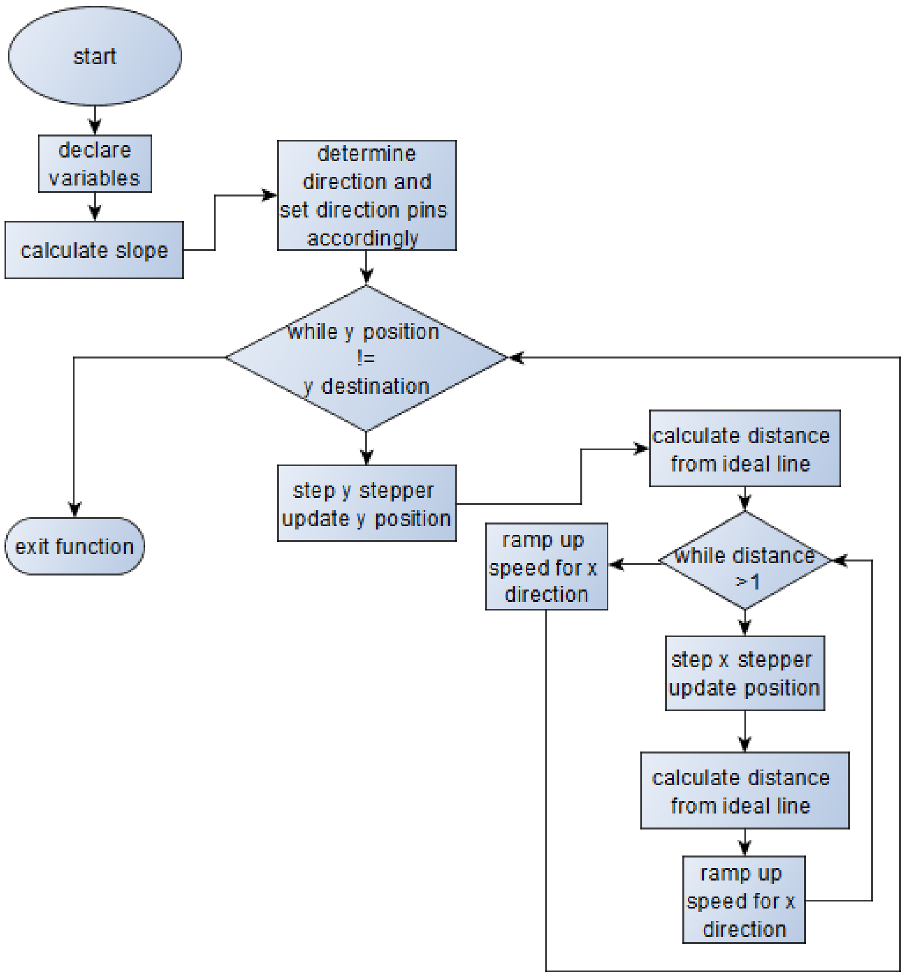
6.2.3 Functions

6.2.3.1 HorizontalMove()

This is the most complicated of the movement functions. It is called when when machine is required to move in a direction other than the x or y plane. In order to move at different angles to the x or y axis both stepper motors must work together e.g to move 45º both motors must turn at the same speed for the same amount of time. To do this I have come up with a function that moves one step at a time, when the position of the aperture is greater than a set amount away from the ideal line the other motor starts stepping until the aperture is back on the line, this process is repeated until the required x and y positions are reached.
Formula for calculating distance from line:

$$distanceToLine=\sqrt{\frac{((0-xDestination\times slope\times yDestination)+slope\times xPosition-yPosition)^2}{slope^2+1}}$$

Flow chart:


start
declare variables
calculate slope
determine direction and set direction pins accordingly
while y position != y destination
exit function
step y stepper update y position
calculate distance from ideal line
ramp up speed for x direction
while distance >1
step x stepper update position
calculate distance from ideal line
ramp up speed for x direction

Code:

```
void horizontalMove(long x ,long y , unsigned long dly, unsigned long ny, unsigned
long nx, unsigned long *yp, unsigned long *xp, unsigned int spu) // if slope of line is
greater than 1 or less than - 1 this function is used
{
   int ii,aa,bb;
   float accel = 3;
   int count  = 0;
   unsigned long posx,posy;
   float lineDist = 0;
   float m = (float)y/(float)x;

   if(y<=0)
   {
      digitalWrite(Y_DIRECTIONPIN,LOW);// counter clockwise
      bb = -1;
   }
   if(y>=0)
   {
      digitalWrite(Y_DIRECTIONPIN,HIGH);// clockwise
```

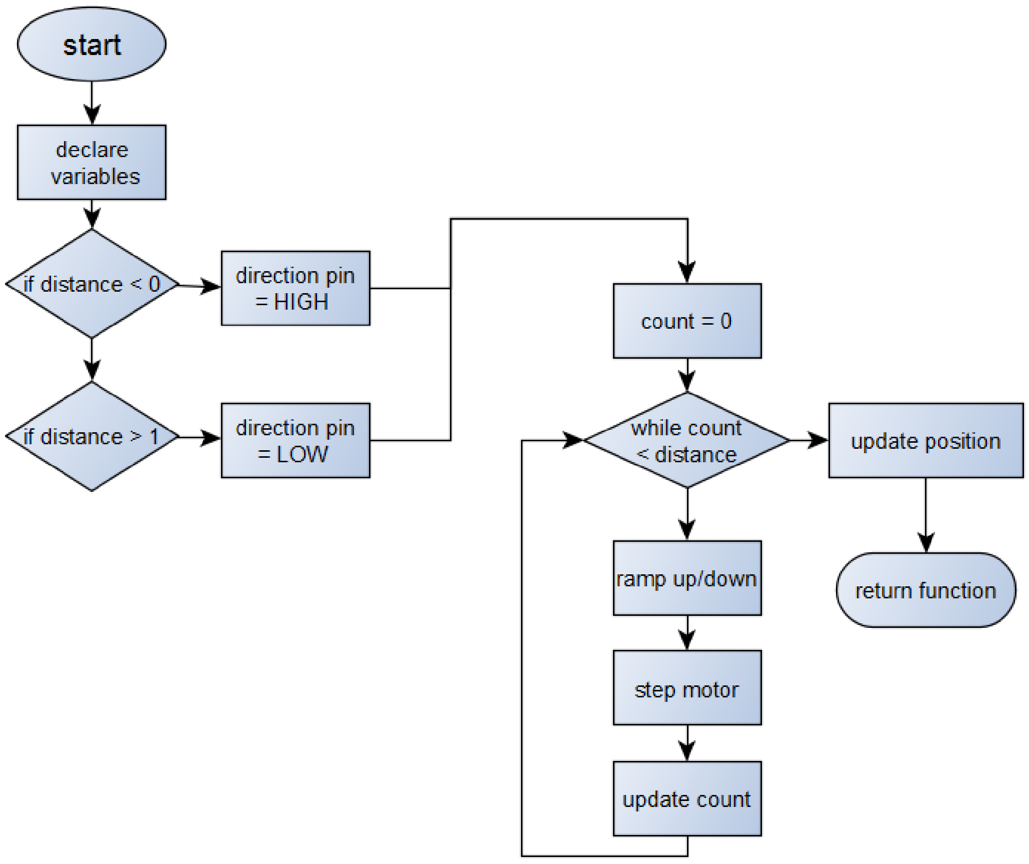
```
        bb = 1;
    }

    if(x<=0)
    {
        digitalWrite(X_DIRECTIONPIN,LOW);// counter clockwise
        aa = -1;
    }
    if(x>=0)
    {
        digitalWrite(X_DIRECTIONPIN,HIGH);// clockwise
        aa = 1;
    }
    while(*yp != ny)
    {
        digitalWrite(Y_step,HIGH);
        delayMicroseconds((dly/1.2)*accel);
        digitalWrite(Y_step, LOW);
        *yp += bb;
        lineDist = sqrt((pow(((0-nx*m+ny)+m*(*xp)-(*yp)),2 ))/((m*m)+1));//calculate
current distance from ideal line
        while(lineDist > 1)
        {
            digitalWrite(X_step,HIGH);
            delayMicroseconds((dly/1.2)*accel);
            digitalWrite(X_step, LOW);
            *xp += aa;
            lineDist = sqrt((pow(((0-nx*m+ny)+m* (*xp) - (*yp)),2
))/((m*m)+1));//calculate current distance from ideal line
            delayMicroseconds(10);
             if(count<120 && accel > 1){
              accel -= 0.0175;
              count ++;
              }
        }
        if(count<120 && accel > 1){
          accel -= 0.0175;
          count ++;
        }

    }
}
```

6.2.3.2 *X/Y/Z move()*

As this function is the same for the 3 axis's I will only describe it once
The function is passed the distance to move and then steps the motor. The function ramps the speed up at the beginning of the movement and ramps it down again at the end. This creates a smoother transition from one direction to another as well as

providing more torque at the beginning of the movement to which helps break the inertia.

Flow chart:



Code:

```
void xMove(long dist, unsigned long dly, unsigned long *xp, unsigned int spu)//move
x axis
{

   int ii;
   int absDist;
   float accel  = 3;
   absDist = sqrt(dist*dist);

   if(dist>=1)
   {
      digitalWrite(X_DIRECTIONPIN,HIGH);// counter clockwise
   }
   if(dist<=0)
   {
      digitalWrite(X_DIRECTIONPIN,LOW);// clockwise
   }

   for(ii=0;ii<absDist;ii++)
   {
     digitalWrite(X_step, HIGH);
     delayMicroseconds(dly*accel);
```

```
        digitalWrite(X_step,LOW);
        delayMicroseconds(10);
        if(ii<120 && accel > 1){
          accel-=0.025;
        }
        if(ii > absDist-120 && ii > 120){
          accel+=0.025;
        }
    }
    *xp=*xp+dist;
}
```

6.2.3.3 Jog X/Y/Z()

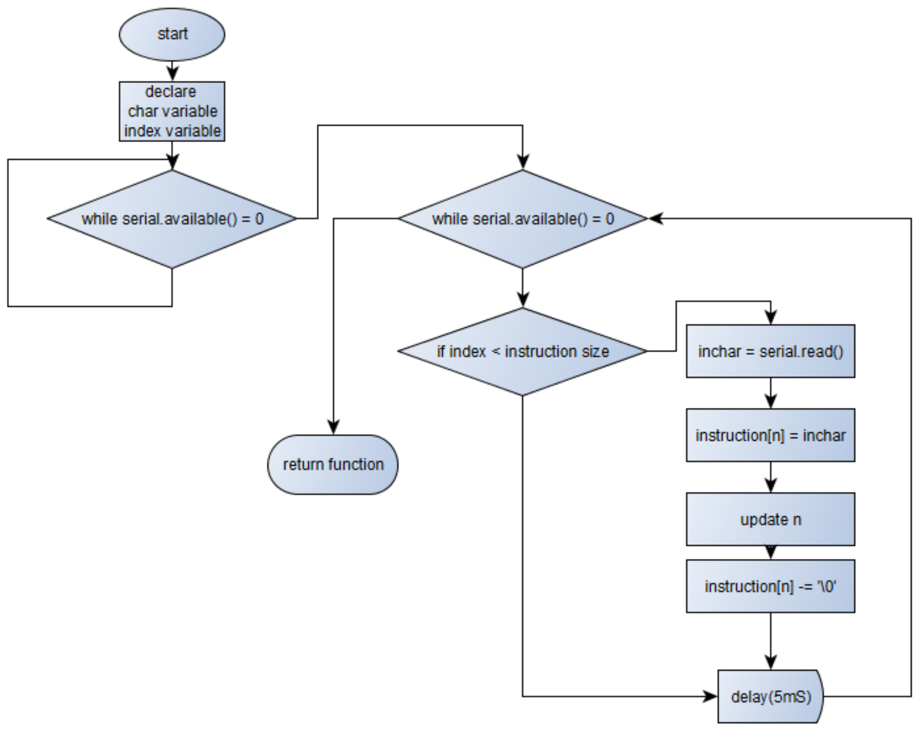
The only difference between the jog function and the move function is that jog does not update the position of the aperture. This is used for setting the origin and drawing height.

Code:

```
void jogX(int dist, unsigned long dly)// manual x move
{
  int ii;
  int absDist;
  absDist = sqrt(dist*dist);
  if(dist>=1)
  {
      digitalWrite(X_DIRECTIONPIN,HIGH);// counter clockwise
  }
  if(dist<=0)
  {
      digitalWrite(X_DIRECTIONPIN,LOW);// clockwise
  }

  for(ii=0;ii<absDist;ii++)
  {
     digitalWrite(X_step, HIGH);
     delayMicroseconds(dly);
     digitalWrite(X_step,LOW);
     delayMicroseconds(10);
  }
}
```

6.2.3.4 readSerial()

This function waits until a serial message is received. The message is then read byte by byte into a char array. When the message is finished the function returns


start
declare char variable index variable
while serial.available() = 0
while serial.available() = 0
if index < instruction size
inchar = serial.read()
instruction[n] = inchar
update n
instruction[n] -= '\0'
return function
delay(5mS)

Code:

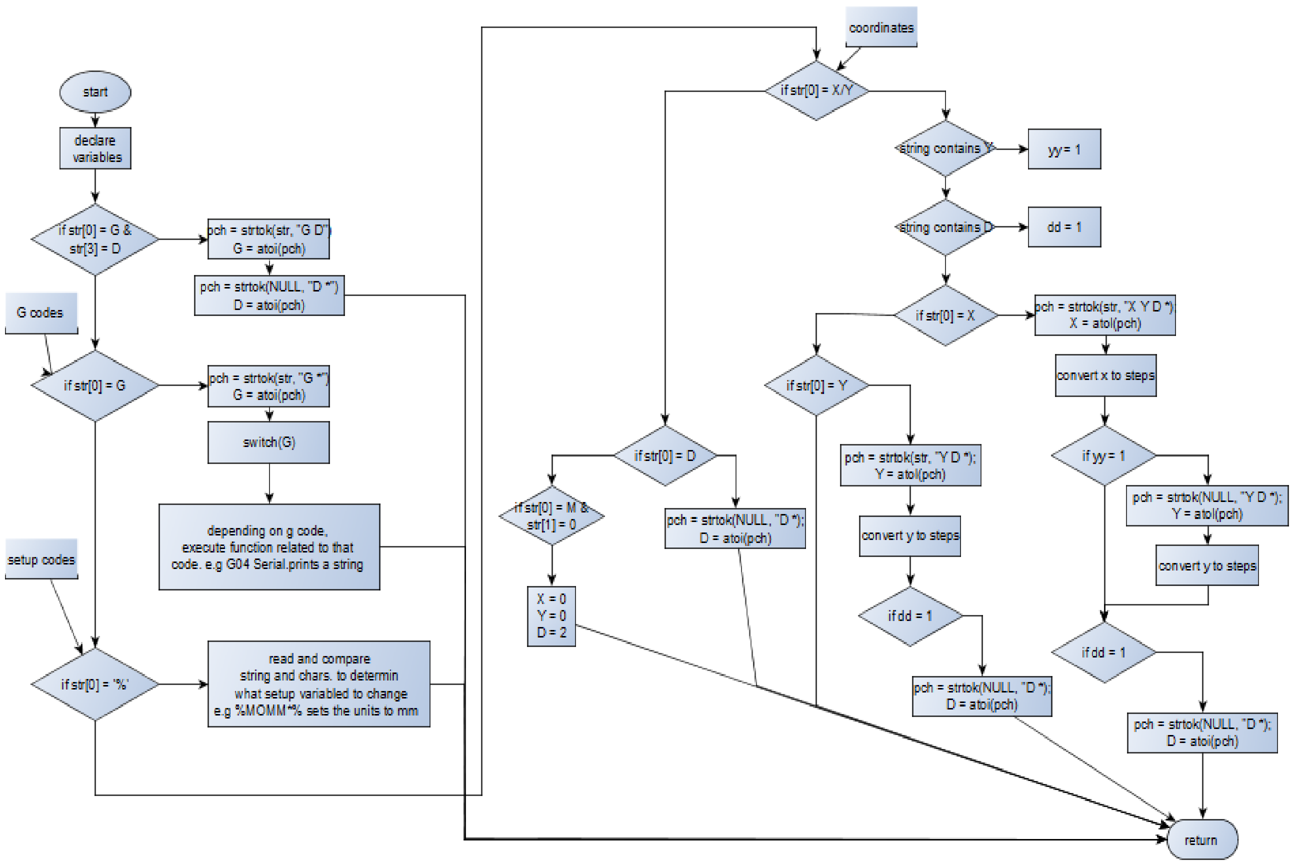
```
void readserial(char *inC,int a, int n)
{
   char inChar; // Where to store the character read
   while(Serial.available() == 0);
   while(Serial.available() > 0) // Don't read unless    // there you know there is data
   {
     if(n < a) // One less than the size of the array
     {
        inChar = Serial.read(); // Read a character
        inC[n] = inChar; // Store it
        n++; // Increment where to write next
        inC[n] = '\0'; // Null terminate the string
     }
     delay(5);
   }
}
```

6.2.3.5 parseString()

this function takes an instruction string and parses it to extract usable information such as oordinated, setup instruction, aperture definitions, units etc.
Formula for converting coordinates to steps:

$$steps = \frac{coordinates}{10^{decimal\ points}} \times steps\ per\ unit$$

Flow chart:



Code:

```
void parseString(struct apeture *D, char *str, unsigned int *dps, unsigned int
*spu,unsigned int *intpl, unsigned long *x, unsigned long *y, unsigned int *z,
unsigned long dly, unsigned long *xo, unsigned long *yo, unsigned int *rpm)
{
   char * pch; //temperary variable for holding string
   char tmp[25]; // temp varible for float conversion
   float tempf;  //
   float xf, yf;
   unsigned int G, num = 0;
   int dist;

   //FOR G54 apeture change CODES
   if(str[0] == 'G'&&str[3]=='D'){
      pch = strtok(str,"G D *");
      G = atoi(pch);

      pch = strtok(NULL,"D * ");
      *z = atoi(pch);
      Serial.print("D = "),Serial.println(*z);
   }
   //for G codes
   else if(str[0]=='G'){
      pch = strtok(str,"G *");
```

```
G = atoi(pch);
switch(G)
{
   case 1:
      Serial.println("linear interpolation");
      *intpl = 1;
      break;
   case 2:
      Serial.println("circular interpolation clockwise");
      *intpl = 2;
      break;
   case 3:
      Serial.println("circular interpolation counterclockwise");
      *intpl = 3;
      break;
   case 4:
      pch = strtok(NULL," ");
      Serial.print("\n");
      while(pch != NULL){
        Serial.println(pch);
        pch = strtok(NULL," *");

      }
      break;
   case 36:
      Serial.println("Area fill ON");
      break;
   case 37:
      Serial.println("Area fill OFF");
      break;
   case 70:
      Serial.println("units inches (deprecated command)");
      break;
   case 71:
      Serial.println("units MM (deprecated command)");
      break;
   case 74:
      Serial.println("Single Quadrant mode enabled");
      break;
   case 75:
      Serial.println("Multi-Quadrant mode enabled");
   case 90:
      Serial.println("absolute mode (deprecated command)");
      break;
   case 91:
      printf("incremental mode (deprecated command)");
      break;

   default:
```

```
              Serial.println("non standard gerber definition");
              break;
        }
  }

  //FOR SETUP INSTRUCTIONS
  else if(str[0]=='%'){
     //for format specification (FS)
     if(str[1]=='F'&&str[2]=='S'){
        if(str[3] == 'L'){
           Serial.println("leading zeroes omitted\n");
        }
        else if(str[3] == 'T'){
           Serial.println("trailing zeroes omitted\n");
        }
        if(str[4]=='A'){
           Serial.println("absolute notation\n");
        }
        else if(str[4]=='I'){
           Serial.println("incremental notation\n");
        }
        pch = strtok(str,"%FSLAX  Y");
        *dps = pch[1]-48;
        Serial.print("format
"),Serial.print(pch[0]),Serial.print(":"),Serial.println(pch[1]);
     }
     //unit select
     if(str[1]=='M'&&str[2]=='O'){
       if(strcmp("%MOIN*%",str)==0){
          delay(100);
           Serial.println("inches\n");
           *spu = 6350;
        }
        else if(strcmp("%MOMM*%",str)==0){
          delay(100);
           Serial.println("millimetres\n");
           *spu = 250;
        }
         else{
          Serial.println("STRING ERROR");
         }
     }
     //apeture define
     if(str[1]=='A'&&str[2]=='D'){
        char shape = str[6];
        pch = strtok(str,"%AD  C R , *%");
        num = atoi(pch)-10;
        D[num].location = atoi(pch);
        if(shape=='C'){
           strcpy(D[num].type,"circlular");
        }
        else if(shape=='R'){
           strcpy(D[num].type, "rectangular");
        }
        D[num].location = atoi(pch);
        // get diameter of apeture
        pch = strtok(NULL,", *%");
        printf("\n%s\n",pch);
        strcpy(tmp, pch);
        sscanf(tmp,"%f", &tempf);
        D[num].diam = tempf;
       // Serial.print("apeture D"),Serial.print(num),Serial.print("
"),Serial.print(D[num].type),Serial.print(" size
"),Serial.print(D[num].diam),Serial.println("MM");
     }
 }
 else if (str[0]=='X'|| str[0] == 'Y'){
   int yy = 0;
   int dd = 0;
   if (strpbrk(str, "Y")!= NULL){
     yy = 1;
   }
   if(strpbrk(str,"D")!= NULL){
     dd = 1;
   }
   if(str[0] == 'X'){
       pch = strtok (str,"X Y- D * ");
       *x = atol(pch);
       xf = *x;
       xf/=1*pow(10,*dps);
       *x = (xf*(*spu));

       if(yy == 1){
          pch = strtok (NULL, "Y- D");
          *y = atol(pch);
          yf = *y;
          yf/=1*pow(10,*dps);
          *y = (yf*(*spu));
       }
   }
   else if(str[0] == 'Y'){
       pch = strtok (str, "Y- D *");
       *y = atol(pch);
       yf = *y;
       yf/=1*pow(10,*dps);
       *y = (yf*(*spu));
   }
   if(dd == 1){
     pch = strtok (NULL, "D *");
     *z = atoi(pch);
   }
 }
 else if(str[0] == 'D')
 {
   pch = strtok (str, "D *");
   *z = atoi(pch);
 }
 else if(str[0] == 'M' && str[1] == '0')
 {
   *x = 0, *y = 0, *z = 2;
 }
 //CODES FOR MANUAL MOVEMENT
 else if(str[0] == 'b')
   {
     if(str[1] == 'o'){
             pch = strtok(str,"box y"); // get x offset
             *xo = atol(pch);
             xf = *xo;
             xf/=1*pow(10,*dps);
             *xo = (xf*(*spu));

             pch = strtok(NULL,"y *"); // get y offset
             *yo = atol(pch);
             yf = *yo;
             yf/=1*pow(10,*dps);
             *yo = (yf*(*spu));
             Serial.print("offsets set. x-"), Serial.print(xf), Serial.print("y: -
"),Serial.println(yf);
     }

     else if(str[1] == 'f'){ // set stepper rpm for feed rate. feed rate = rpm/60 * 1.25 (
1.25 is the pitch of the threaded bar)
       pch = strtok(str, "bf *");
       *rpm = atoi(pch);
       Serial.print("feedrate set to "),Serial.print(*rpm/48),Serial.println("mm//S");
     }

     else if(str[1] == 'Z'){
        pch = strtok(str,"bZ *");
        dist = atoi(pch);
        jogZ(dist, dly);
     }
     else if(str[1] == 'z'){
        pch = strtok(str,"bz *");
        dist = atoi(pch);
        jogZ(0-dist, dly);
     }
```

```
        else if(str[1] == 'Y'){
          pch = strtok(str,"bY *");
          dist = atoi(pch);
          jogY(dist, dly);
        }
        else if(str[1] == 'y'){
          pch = strtok(str,"by *");
          dist = atoi(pch);
          jogY(0-dist, dly);
        }
        else if(str[1] == 'X'){
          pch = strtok(str,"bX *");
          dist = atoi(pch);
          jogX(dist, dly);
        }
        else if(str[1] == 'x'){
          pch = strtok(str,"bx *");
          dist = atoi(pch);
          jogX(0-dist, dly);
        }

    }
}
```

6.2.3.6 calcDelay()

Formula for calculating delay for a given RPM

$$delay(\mu S)=\left(\frac{\frac{60}{stepsPerRevolution}}{RPM}\times(1\times10^{6})\right)-10$$

```
void calcDelay(unsigned int x, unsigned long *dly)
{
   float y=60;
   y = y/200;
   *dly = (y/x*1000000)-10;
}
```

7. CONCLUSION

7.1 Problems Encountered

Throughout the course of this project I have encountered numerous problems but luckily none were serious enough to halt the project
Here is a list of some of the problems I encountered and how I corrected them:

7.1.1 Building the machine

My original plan was to build the machine out of 20x20mm aluminium profile, as I could not source any locally for a reasonable price I decided to build the frame out of 25x25mm mild steel box section. This was a much stronger option but as the steel had to be welded problems arose. The head from the welding caused the frame to warp leaving a noticeable difference in width between the top and bottom of the frame.
Solution
My immediate solution was to cut the top cross member and attach lugs to which I could screw a bolt through allowing me to adjust the width with precision. However when I cut the cross member the steel sprung back into alignment. This gave two advantages:

1. The springiness allowed the bearings to run up and down the steel with an almost constant pressure.
2. The cross member which is now only attached on one side acts as a tensioner for the belt which moves the z-axis

7.1.2 Designing the PCB

Due to an error I made while designing the PCB in altium I places all 3 of the 470µF electrolytic capacitors the wrong way around this resulted in all 3 of them blowing. At first I thought this was due to a bit of solder crossing a joint and as the capacitors were inserted as shown in the PCB design in altium it took some time to find the problem.
I found the problem when I cross referenced all the footprints from the PCB design with the components in the schematic design. The footprint in the PCB design had the pins 1 and 2 in reverse order. The fix was as simple as de-soldering the blown capacitors and replacing them with new ones in the right orientation.

7.2 Goals Achieved

7.2.1 To design and build the mechanical components for a 3 axis cnc machine

7.2.2 To design, build and test an Intel Galileo compatible 3 stepper motor driver shield

7.2.3 To write a windows application to communicate via serial interface with the Intel Galileo and similar development boards.

7.2.4 To program the Intel Galileo to understand Gerber (rs-274) and Gerberx (rs-274- x) codes and to control a 3-axis cnc machine

7.3 Goals Not Achieved

7.3.1 To control the cnc machine via network connection

7.3.2 Install ir limit switches to prevent cnc from working outside of its limits

7.3.3 To install a 3d printer extruder and use the cnc machine as a 3d printer

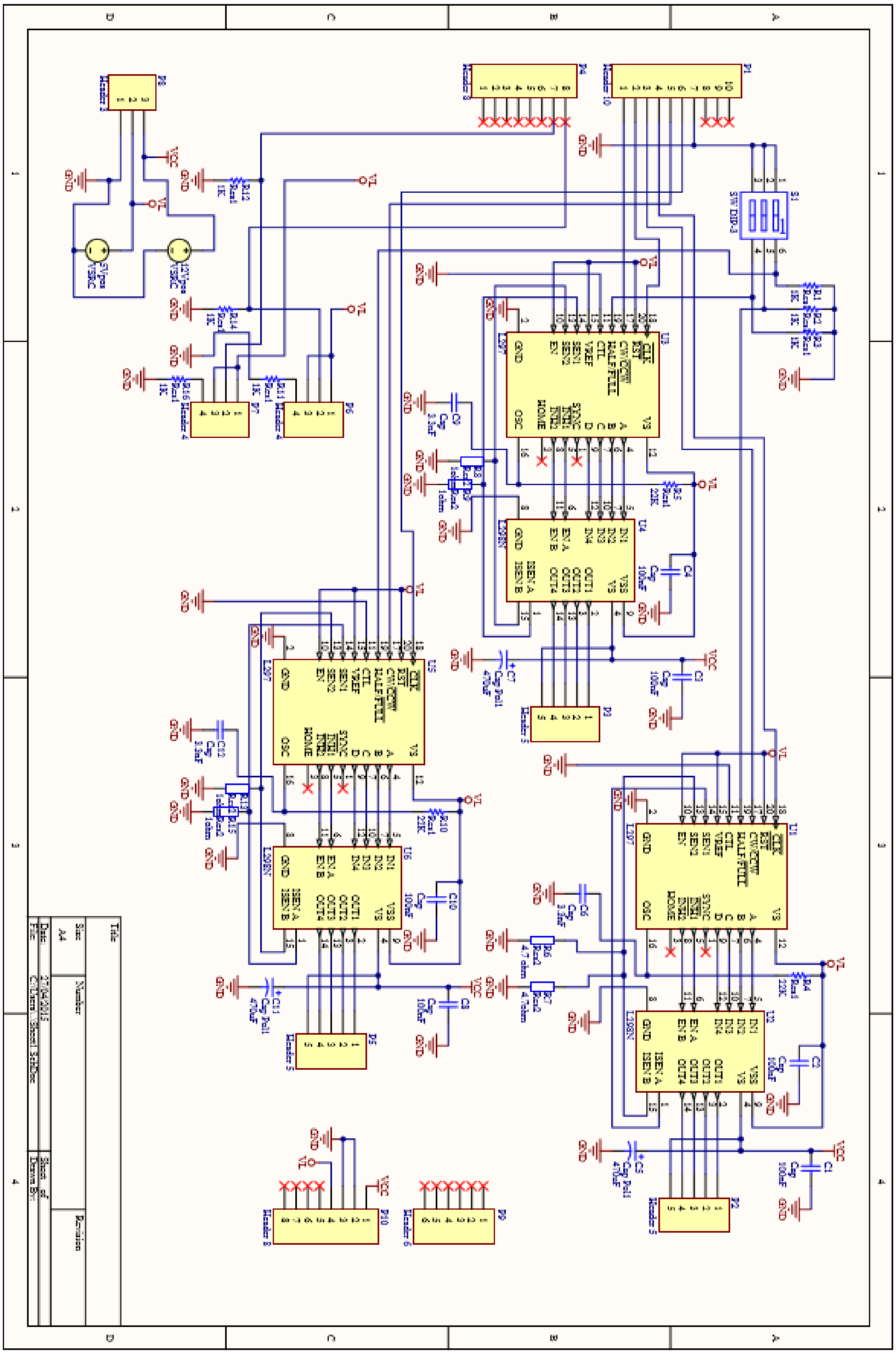
7.4 Final Conclusions

As the popularity of 3d printers and cnc machines rises in both the home and industrial setting, the skills I have learned throughout this project will no doubt benefit me greatly. I have proved that a cnc machine or 3d printer can be built with reasonable accuracy on a low budget and using tools and equipment that can be purchased or borrows by almost anybody. I would like the chance to continue this project in the future to add more functionality and improve on what I have done.

APPENDIX A: - BILL OF MATERIALS

Item #	Quantity	Designator	Description	Footprint	Comment
1	9	C1, C2, C3, C4, C6, C8, C9, C10, C12	Capacitor	CAPR5.08-7.6x2.5	Cap
2	3	C5, C7, C11	Polarized Capacitor (Radial)	Cap1	Cap Pol1
3	1	P1	Header, 10-Pin	HDR1X10	Header 10
4	3	P2, P3, P5	Header, 5-Pin	HDR1X5	Header 5
5	2	P4, P10	Header, 8-Pin	HDR1X8	Header 8
6	2	P6, P7	Header, 4-Pin	HDR1X4	Header 4
7	1	P8	Header, 3-Pin	HDR1X3	Header 3
8	1	P9	Header, 6-Pin	HDR1X6	Header 6
9	10	R1, R2, R3, R4, R5, R10, R11, R12, R14, R16	Resistor	AXIAL-0.3	Res1
10	6	R6, R7, R8, R9, R13, R15	Resistor	AXIAL-0.4	Res2
11	1	S1	DIP Switch, 3 Position, SPST	DIP-6	SW DIP-3
12	3	U1, U3, U5	Stepper Motor Controller	DIP20	L297
13	3	U2, U4, U6	Dual Full Bridge Driver	Multiwatt15V	L298N
15	1	n/a	dell atx power supply	n/a	
16	2	n/a	astrosyn 37-0514 stepper motor	n/a	size 17
17	1	n/a	astrosyn my-5602 stepper motor	n/a	size 14
18	1	n/a	intel galileo gen2 development board	n/a	
19	3	n/a	push to make switch		6mm panel mount
20	1	n/a	usb jack	n/a	type A
21	1	n/a	usb jack	n/a	type B
22	3	n/a	female din connection	n/a	female
23	3	n/a	male din connection	n/a	male
24	14	n/a	m6x50 bolt	n/a	
25	14	n/a	m6 nut	n/a	
26	1	n/a	8mm timing belt	n/a	rep-rap
27	3	n/a	16 tooth 5mm bore pulley	n/a	rep-rap
28	4	n/a	16 tooth 8mm bore pulley	n/a	rep-rap
29	2	n/a	350mm drawer slide	n/a	B&Q

APPENDIX B: - SCHEMATIC



APPENDIX D: - C CODE

8. GalileoCNC

8.1 Main.c

```
/**********************************************************************
***********************************
 _                _ _ _             ___  ___   ___ _____  ___ ____  _  _    ___   ___
| |__   ___  ___ _  _| | (_) ___  _ __     / _ \/ _ \ / _ \___ / / _ \___ \| || |  / _ \ / _ \
| '_ \ / __|/ __| | | | | | |/ _ \| '_ \   / /_\/ | | | | | ||_ \| | | |__) | || |_| | | | | | |
| |_) |\__ \ (__| |_| | | | | (_) | | | |  / /_\\| |_| | |_| |__) | |_| / __/|__   _| |_| | |_| |
|_.__(_)___/\___|\__,_|_|_|_|\___/|_| |_|  \____/ \___/ \___/____/ \___/_____|  |_|  \___/
\___/ \

programm: galileo cnc
   main
***************************************************************************
***********************************/
#include <string.h>
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <ctype.h>
#include <conio.h>
#include "functions.h"



int main()
{
   heading();
   screen = GetStdHandle(STD_OUTPUT_HANDLE);
   system("COLOR 1F");

   // Declare variables and structures
   int baudrate = 9600;
   int dev_num = 50; // comm port to start search at
   char dev_name[MAX_PATH];
   DCB dcbSerialParams = {0};  // aray for storing serial parameters
   COMMTIMEOUTS timeouts = {0}; // array for storing timout information

    printf("Searching serial ports...\n");
   while(dev_num >= 0)
   {
        printf("\r                       ");
        printf("\rTrying COM%d...", dev_num);
        sprintf(dev_name, "\\\\.\\COM%d", dev_num);
        hSerial = CreateFile(
                    dev_name,
                    GENERIC_READ|GENERIC_WRITE,
                    0,
                    NULL,
                    OPEN_EXISTING,
                    FILE_ATTRIBUTE_NORMAL,
                    NULL );
        if (hSerial == INVALID_HANDLE_VALUE) dev_num--;
        else break;
    }

    if (dev_num < 0)
    {
        printf("No serial port available\n");
        return 1;
    }

    printf("OK\n");

    // Set device parameters (9600 baud, 1 start bit,
    // 1 stop bit, no parity)
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
    if (GetCommState(hSerial, &dcbSerialParams) == 0)
    {
        printf("Error getting device state\n");
        CloseHandle(hSerial);
        return 1;
    }
    //dcbSerialParams.BaudRate = CBR_38400;
    dcbSerialParams.BaudRate = baudrate;
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;
    if(SetCommState(hSerial, &dcbSerialParams) == 0)
    {
        printf("Error setting device parameters\n");
        CloseHandle(hSerial);
        return 1;
    }
    // Set COM port timeout settings
    timeouts.ReadIntervalTimeout = 50;
    timeouts.ReadTotalTimeoutConstant = 50;
    timeouts.ReadTotalTimeoutMultiplier = 10;
    timeouts.WriteTotalTimeoutConstant = 50;
    timeouts.WriteTotalTimeoutMultiplier = 10;
```

```
   if(SetCommTimeouts(hSerial, &timeouts) == 0)
   {
      printf("Error setting timeouts\n");
      CloseHandle(hSerial);
      return 1;
   }

   char *cmd = NULL;

//*****************************************************************
*********************************************

   char cmdLine[200];
   while(1)
   {
      printf(">>");
      gets(cmdLine);
      cmd = strtok(cmdLine, " ");

      if(cmd!=false)
      {
         if(strcmp(cmd, "help")== 0)
         {
            help();
         }
         else if(strcmp(cmd, "getg")== 0)
         {
            GetGcode(dev_name);
         }
         else if(strcmp(cmd, "setup") == 0)
         {
            setup(dev_name);
         }

         else if(strcmp(cmd, "exit") == 0)
         {
            break;
         }
         else
         {
            printf("Unknown command!\n");
         }
         printf("\n");
      }
   }
   // Close serial port
   printf("Closing serial port...");
   if (CloseHandle(hSerial) == 0)
   {
        printf("Error\n");
        return 1;
    }
    printf("OK\n");
    return 0;
}
```

8.2 Functions.h

```
#ifndef FUNCTIONS_H_INCLUDED
#define FUNCTIONS_H_INCLUDED
#define MAX_STRING_LENGHT 80

#include <string.h>
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <ctype.h>
#include <conio.h>


void heading();
void setup(char*);
void exit();
int help();
void messageup(char*, char*);
void GetGcode(char*);
int getOffset(char*, char*, unsigned int*);
void delay(unsigned int);
int sendSerial(char *,char *);
int read_serial(char*, char*);
void progressBar(unsigned int, unsigned int, unsigned int);
```

8.3 Functions_major.c

```
#include "functions.h"
#include <string.h>
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <ctype.h>
#include <conio.h>

void heading()//print header logo and intro text
```

```
{

printf("|=======================================================
=====================|\n");
     printf("|\t                ___ __            _______    ________                   |");
     printf("\n|\t   ____ _____ _/ (_) /__  ____  / ____/ | / / ____/                  |\n");
     printf("|\t  / __ `/ __ `/ / / / _ \\/ __ \\/ /   /  |/ / /                      |\n");
     printf("|\t / /_/ / /_/ / / / /  __/ /_/ / /___/ /|  / /___                    |\n");
     printf("|\t \\__, /\\__,_/_/_/_/\\___/\\____/\\____/_/ |_/\\____/                  |\n");
     printf("|\t/____/                                                    |\n");
     printf("|====================================================by
brendan scullion=======|\n");

    printf("\nWelcome, this is a program to to control a 3d printer/cnc machine using an
\narduino or galileo micro controller.\n\n");
    printf("use the [help] command for a list of commands.\n");

}
/**=======================================================
===================================
===== setup function is used to setup the machine prior to use
===== set up feed rate, position, etc...
=======================================================
===================================*/
void setup(char *devName)
{

    char ch = '0';
    char instruct[10];
    char serialIn[MAX_STRING_LENGHT]; // char  store string
    int dist, rpm;
    float feed; // float for storing feed rate

    printf("please select option from the list below\n 1\t adjust tool position.\n 2\t adjust
feed rate(mm/s).\n");
    switch(_getch())
    {
        case '1':
            while(ch!='q')
            {
                printf("use the up and down numerical keys to move tool head up and
down(press q to exit)\n");
                printf("enter jog distance ( 1000 = 1cm)(0 to exit):"), scanf("%i", &dist);
                if(dist == 0){
                    return;
                }
                ch = '0';
                while(ch!='e'&&ch!= 'q')
                {
```

```
                ch = _getch();
                if(ch == '7'){ // Z axis up
                    printf("Z up\n");
                    sprintf(instruct, "bZ%d*\n",dist); // creats string if format [function
identifier][axis]<distance to move>
                }
                if(ch == '1'){ // Z-axis down
                    printf("Z down\n");
                    sprintf(instruct, "bz%d*\n",dist);
                }
                if(ch == '2'){ // y-axis back
                    printf(" y++ \n");
                    sprintf(instruct, "bY%d*\n",dist);
                }
                if(ch == '8'){ // Y-axis forward
                    printf("y--\n");
                    sprintf(instruct, "by%d*\n",dist);
                }
                if(ch == '6'){ // X-axis right
                    printf("X -- \n");
                    sprintf(instruct, "bX%d*\n",dist);
                }
                if(ch == '4'){ // X-axis left
                    printf("X ++\n");
                    sprintf(instruct, "bx%d*\n",dist);
                }
                sendSerial(instruct, devName); // send instruction to galileo
                read_serial(serialIn, devName); // waits for reply from machine and adds
it t char array (serialIn)
                printf("%s", serialIn); // print reply
            }
        }
        break;
    case '2':
        printf("enter feed rate( < 6.25mm/s recomended):"),scanf("%f", &feed);
        if(feed > 6.25)
        {
            MessageBeep(0);
            if(MessageBoxA(0,"warning: feed rate greater than 6.25 can result in loss of
position during process!\ndo u still want do use this speed?", "warning",
MB_YESNO) == 7)
            {
                printf("enter feed rate( < 6.25mm/s recomended):"),scanf("%f", &feed);
            }
        }
        rpm = feed*48; // feed rate is converted to motor rpm
        sprintf(instruct, "bf%d*", rpm);
        sendSerial(instruct, devName);
        Sleep(100);
```

```
            read_serial(serialIn, devName);
            break;
        default:
            printf("bad user input, please choose from options above");
            break;
    }
}
/**=========================================================
==========================================================
==================== help function opens a help file and displays it on the
screen ==========================================
==========================================================
==========================================================*/
int help()
{
    char ch[MAX_STRING_LENGHT]; // char array to store one line of the help file.

    file = fopen("helpfile.f","r"); // open help file
    if(file == NULL) // if the file cannot be opened or does not exist
    {
        perror("Error while opening the helpfile.\nmake sure help file exists!\n");
        return 1;
    }
    while(fgets(ch,100,file)!=NULL)// print out file line by line until end of file
    {
        printf("%s",ch);
    }
    fclose(file); // clode file
    return 0; // exit function
}
/**=========================================================
========================================
============== gets g code calculates offsets and sends it to the galileo
============================
==========================================================
========================================*/
void GetGcode(char *devName)//read g code and set up for machine
{
    char gline[MAX_STRING_LENGHT];
    char serialIn[MAX_STRING_LENGHT];
    char file_name[20];
    char *pch;
    char message[20];
    unsigned int line_total = 1, current_line = 0;

   printf("Enter the name of file you wish to use\n");
   gets(file_name);
   //calculate image position and then calculate best offset for maximum space on
board
```

```
  getOffset(file_name, // file to be calculated
          devName,   //device to send date to
           &line_total); // number of lines of code

  file = fopen(file_name,"r"); // read mode

  if( file == NULL )
  {
    perror("Error while opening the file.\n");
    return;
  }
  printf("The contents of %s file are :\n", file_name);

  while(1){
      while(fgets(gline,100,file)!=NULL){
          current_line++;
          printf("%s\n",gline);//print one line of the gcode
          Sleep(10);
          if(sendSerial(gline, devName) == 1){ // sent gcode to cnc machine
              return;
          }
          if(read_serial(serialIn, devName) == 1) // wait for response from machine and
print
          {
              return;
          }
          printf("%s\n", serialIn);
          progressBar(current_line, line_total, 50);
          if(gline[0] == 'G'&& gline[1] == '5'&&gline[2] == '4'){
              pch = strtok(gline,"G54D * ");
              sprintf(message, "insert aperture no: %s", pch);
              messageup("aperture change", message);
          }
      }
      messageup("finished!","process has finished."); // when it reaches the end of the
file
      break;
  }
  fclose(file); // close file
}
```

8.4 Functions_minor.c

```
*********************************************************************
********************************/
#include "functions.h"
#include <string.h>
#include <windows.h>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <ctype.h>
#include <conio.h>
/**=============================================================
================
=========================== pop up message
===================================
=============================================================
=================*/
void messageup(char *title, char *body)
{
   MessageBeep(0);
   MessageBox(0,
            body,    // messgeto display n body of box
            title, // title string
            MB_OK // box contains only 1 o button
             );
}

int sendSerial(char *instr, char *devName)
{
   /**********************send
string*********************************************/
   unsigned char text_to_send[MAX_PATH];
   char buffer[MAX_STRING_LENGHT];
   strcpy(buffer, ""); // empty buffer
   strcpy(buffer, instr);

   // Check that some text to send was provided
   if (strlen(buffer) == 0)
   {
      return 1;
   }
   int n = 0, m = 0;
   while(n < strlen(buffer))
   {
      text_to_send[m] = buffer[n];
      m++; n++;
   }
   text_to_send[m] = '\0'; // Null character to terminate string

    // Send specified text
   DWORD bytes_written, total_bytes_written = 0;
  //printf("Sending text... ");
   while(total_bytes_written < m)
   {
      if(!WriteFile(hSerial, text_to_send + total_bytes_written,
            m - total_bytes_written, &bytes_written, NULL)){
            fprintf(stderr, "Error writing text to %s\n", devName);
            CloseHandle(hSerial);
            return 1;
        }
        total_bytes_written += bytes_written;
    }
    //fprintf(stderr, "\t\t%ld bytes written to %s\n", total_bytes_written, devName);
    return 0;
}

int read_serial( char *message, char *devName )
{
    char buffer[MAX_STRING_LENGHT];
    DWORD bytes_recieved = 40, written = 0;
    int index = 0;
    strcpy(buffer,""); //empty buffer

    while(buffer!=NULL){ // wait untill serail message recieved
        if(ReadFile(hSerial, buffer,sizeof(buffer), // read serial
                &bytes_recieved, NULL ) == FALSE){
                    perror("ERROR READING SERIAL");
                return 1;
                }
        if(bytes_recieved){                // if something to read
            WriteFile(screen, buffer, bytes_recieved, &written, NULL);
            for(index = 0; buffer[index] == '*'; index++){
                message[index] = buffer[index];
            }
            message[index] = '\0';
            return 0;
        }
        if(kbhit()){
            if(getch() == 'q')
            {
                perror("\nUSER ABORT!!\n");
                return 1;
            }
        }
    }
}

void progressBar(unsigned int i, unsigned int n, unsigned int width)
{
    //when 100% is reached fuction is exited
   if ((i == n) && (i % (n/100+1) != 0) )
        return;

    float ratio  =  i/(float)n;
    int   c      =  ratio * width;
     // Show the percentage complete.
    printf("%3d%% [", (int)(ratio*100) );

    // Show the load bar.
    for ( i=0; i<c; i++){ // progress so far
       printf("=");
    }

    for ( i=c; i<width; i++){ // remainder filled with spaces
       printf(" ");
    }

    printf("]\n");
}

int getOffset(char *filename,char *devName, unsigned int *totalLines)// find
minimum x and y coordinates and set offset
{
    char gline[MAX_STRING_LENGHT];
    char serialIn[MAX_STRING_LENGHT];
    char *pch;
    char select;
    char str[MAX_STRING_LENGHT];
    bool yy = FALSE;
    bool start = FALSE;
    unsigned long x = 0, y = 0, xMin = 0 , yMin  = 0;
    unsigned int count = 0;

    file = fopen(filename,"r"); // read mode
    if(file == NULL ){
      perror("Error while opening the file.\n");
      return 1;
    }
    printf("\ncalculating offsets....\n");
    while(fgets(gline,100,file)!=NULL)
    {
        count++;
        if (gline[0]=='X'||gline[0]=='Y'){
            if(strpbrk(gline,"Y")!= NULL){
                yy = true;
            }
           if(gline[0]=='X'){
                pch = strtok(gline,"X Y");
                x = atol(pch);

                if(yy == true){
                    pch = strtok(NULL,"Y- D");
                    y = atol(pch);
                }
            }
            else if(gline[0]=='Y'){
                pch = strtok (gline, "Y- D *");
                y = atol(pch);
            }
            if(!start){
            xMin = x, yMin = y;
            }
            start = TRUE;
            if(x<xMin){
                xMin = x;
            }
            if(y<yMin){
                yMin = y;
            }
            yy = FALSE;
        }
    }
    *totalLines = count;
    printf("Min x position = %lu , Min y position = %lu\n", xMin, yMin);
    xMin = xMin - 1000;
    yMin = yMin - 1000;
    printf("Most appropriate offsets are x -%lu y -%lu\nwould you like to use these
offsets?(y/n):",xMin, yMin);
    scanf("%c", &select);
    if(select == 'y')
    {
        sprintf(str, "box%luy%lu*", xMin, yMin);
        if(sendSerial(str, devName) == 1){
            return 1;
        }
        if(read_serial(serialIn, devName) == 1){ // wait for response from machine and
print
            return 1;
        }
    }
    fclose(file);
    return 0;
}
```

9. Galileo cnc controller

9.1 GalileoCncController.ino

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

```
 _                    _ _ _                ___  ___   ___ _____  ___ ____  _  _    ___   ___
| |__   ___  ___ _   _| | (_) ___  _ __     / _ \/ _ \ / _ \___ / / _ \___ \| || |  / _ \ / _ \
| '_ \ / __|/ __| | | | | | |/ _ \| '_ \   / /_\/ | | | | | ||_ \| | | |__) | || |_| | | | | | |
| |_) |\__ \ (__| |_| | | | | (_) | | | |  / /_\\| |_| | |_| |__) | |_| / __/|__   _| |_| | |_| |
|_.__(_)___/\___|\__,_|_|_|_|\___/|_| |_|  \____/ \___/ \___/____/ \___/_____|  |_|  \___/
\___/ \
* programm: Intel Galileo CNC cntroller
* date: 30/01/2015
* Galileo cnc controller
* description: program to cont x,y and z movements of a cnc machine or similer via
serial usb
* works in conjunction with galileo cnc windows application
*
* by brendan scullion G00302400
******************************************************************************
****************************************************/
#include <Arduino.h>
#include <string.h>

/*********************************declare
variables******************************************/
//pins to controle stepper direction
const int X_DIRECTIONPIN = 10;
const int Y_DIRECTIONPIN = 8;
const int Z_DIRECTIONPIN = 12;
//pins to which a square wave is output to increment stepper motor
const int X_step = 11;
const int Y_step = 9;
const int Z_step = 13;

const int X_HOME = 7;
const int Y_HOME = 6;

/*****************************************************************************
********/

void xMove(long, unsigned long, unsigned long*, unsigned int);
void yMove(long, unsigned long, unsigned long*, unsigned int);
void zMove(long, unsigned long, long*);
void jogX(int, unsigned long);
void jogY(int, unsigned long);
void jogZ(int, unsigned long);
void horizontalMove(long, long, unsigned long, unsigned long, unsigned long,
unsigned long*, unsigned long*, unsigned int);
void calcDelay(unsigned int, unsigned long*);
void readserial(char*, int, int);
void parseString(struct apeture *D, char*, unsigned int*, unsigned int*,unsigned int*,
unsigned long*, unsigned long*, unsigned int*, unsigned long, unsigned long *,
unsigned long *, unsigned int* );
```

```
void flash(int, int, long *, unsigned long);

//structure for storing apeture details and locstions
struct apeture {
   int location;
   char type[20];
   float diam;
};

void setup()
{
         Serial.begin(9600); // start serial with baudrate 19200
      // initialise pins
      pinMode(13, OUTPUT);
         pinMode(12, OUTPUT);
         pinMode(11, OUTPUT);
         pinMode(10, OUTPUT);
         pinMode(9, OUTPUT);
         pinMode(8, OUTPUT);
         pinMode(7, INPUT);
         pinMode(6, INPUT);

      Serial.println("\ngalileo cnc V1.0 by brendan scullion");
}

void loop()
{

   long moveX = 0;
   long moveY = 0;
   long moveZ = 0;
   unsigned long stepDelay = 1;
   unsigned int steppRpm = 240;

   //keeping track of position
   unsigned long posX = 0;  // current position (in steps)
   unsigned long newX = 0;  // destination position (in steps)
   unsigned long posY = 0;
   unsigned long newY = 0;
   unsigned long xOffset = 0;
   unsigned long yOffset = 0;
   long posZ = 0;
   int newZ = 0;
   float unitconverter = 1;
   unsigned int interpolation = 1; // 1 for linear, 2 for clockwise circular, 3 for
counterclockwise circular
   unsigned int d = 1, dps = 4;    // 4 for FSLAX24Y24 etc..
   unsigned int stepsPerUnit = 160;  //160 per mm or 6,350 per inch
   const int instructionsize = 50;
   unsigned int index = 0;
   int draw = 0, lift = 300;
   struct apeture D[30];
   char *instruction;
   instruction = (char*)malloc(instructionsize*sizeof(char)); // Allocate some space
for the string
   newZ = lift;

   while(1)
   {
      strcpy(instruction, ""); // clear instruction array
      readserial(instruction, instructionsize, index); // read in one line of gerber file
      //parse string into useable data and instructions and convert chars to integer value
      parseString(
               D,              // structure for storing apeture data
               instruction,      // g code or gerber code
               &dps,            // decimal points in coordinates
               &stepsPerUnit,    // amount of steps it takes the motor to move one unit of
measurement mm/inch
               &interpolation,   // are u drawing a circle or a straight line
               &newX, &newY,     // destination x,y coordinate
               &d,               // what do do with the z axis
               stepDelay,          // delay between steps to controle speed of movement
               &xOffset, &yOffset, // offsets for x, y
               &steppRpm           // rpm of stepper motor
               );

      //newX -= xOffset;
      //newY -= yOffset;
      calcDelay(steppRpm, &stepDelay);

      if(d == 1)
        newZ = draw;
      if (d == 2)
        newZ = lift;
      if(d == 3)
        newZ = lift;

  while(posX != newX || posY != newY || posZ != newZ)
     {

      //calculate distance to move
      moveX = newX  - posX ;
      moveY = newY - posY ;
      moveZ = newZ - posZ;

      //decide which function to use dependng on slope of line and direction

      // for linear interpolation
      if(interpolation == 1)
        {
            if(moveZ>0||moveZ<0)
            {
               zMove(moveZ,stepDelay, &posZ);
            }
            if(moveY == 0)
            {
               xMove(moveX,stepDelay,&posX, stepsPerUnit);
            }
            else if(moveX == 0)
            {
               yMove(moveY,stepDelay,&posY, stepsPerUnit);
            }
            else
            {
               horizontalMove(moveX, moveY,  // x and y distance machine has to move
                           stepDelay,    // delay for speed control
                           newY, newX,    // destination x, y coordinates
                           &posY, &posX,  // current X,Y coordinates
                           stepsPerUnit  //steps per unit of measurement
                           );
            }
         }
       if(posX==newX&&posY==newY&&posZ==newZ)
       {
         if(d == 3)
         {
          flash(draw, lift, &posZ, stepDelay);
         }
         Serial.print((float)posX/stepsPerUnit),Serial.print(" y:"),
Serial.println((float)posY/stepsPerUnit);
       }
     }

     Serial.print("Instruction finished X:");

   }
   free(instruction);
}
```

9.2 Functions.ino

```
/*********************************************functions*************
*************************************************************/
```

```
void horizontalMove(long x ,long y , unsigned long dly, unsigned long ny, unsigned
long nx, unsigned long *yp, unsigned long *xp, unsigned int spu) // if slope of line is
greater than 1 or less than - 1 this function is used
{
   int ii,aa,bb;
   float accel = 3;
   int count  = 0;
   unsigned long posx,posy;
   float lineDist = 0;
   float m = (float)y/(float)x;

   if(y<=0)
   {
      digitalWrite(Y_DIRECTIONPIN,LOW);// counter clockwise
      bb = -1;
   }
   if(y>=0)
   {
      digitalWrite(Y_DIRECTIONPIN,HIGH);// clockwise
      bb = 1;
   }

   if(x<=0)
   {
      digitalWrite(X_DIRECTIONPIN,LOW);// counter clockwise
      aa = -1;
   }
   if(x>=0)
   {
      digitalWrite(X_DIRECTIONPIN,HIGH);// clockwise
      aa = 1;
   }
   while(*yp < ny||*yp > ny)
   {
      digitalWrite(Y_step,HIGH);
      delayMicroseconds((dly/1.2)*accel);
      digitalWrite(Y_step, LOW);
      *yp += bb;
      lineDist = sqrt((pow(((0-nx*m+ny)+m*(*xp)-(*yp)),2 ))/((m*m)+1));//calculate
current distance from ideal line
      while(lineDist > 1)
      {
         digitalWrite(X_step,HIGH);
         delayMicroseconds((dly/1.2)*accel);
         digitalWrite(X_step, LOW);
         *xp += aa;
         lineDist = sqrt((pow(((0-nx*m+ny)+m* (*xp) - (*yp)),2
))/((m*m)+1));//calculate current distance from ideal line
         delayMicroseconds(10);
```

```
           if(count<120 && accel > 1){
            accel -= 0.0175;
            count ++;
            }
       }
       if(count<120 && accel > 1){
         accel -= 0.0175;
         count ++;
       }

   }
}
void flash(int down, int up, long *pz, unsigned long dly)
{
   long posz = *pz;
   zMove(down - posz, dly, &posz);
   delay(200);
   zMove(up - posz, dly, &posz);
   *pz = posz;
}
/******************************************************************
*************************************************
============================================ move x axis
==================================================================
=
*******************************************************************
*************************************************/
void xMove(long dist, unsigned long dly, unsigned long *xp, unsigned int spu)//move
x axis
{

   int ii;
   int absDist;
   float accel  = 3;
   absDist = sqrt(dist*dist);

   if(dist>=1)
   {
      digitalWrite(X_DIRECTIONPIN,HIGH);// counter clockwise
   }
   if(dist<=0)
   {
      digitalWrite(X_DIRECTIONPIN,LOW);// clockwise
   }

   for(ii=0;ii<absDist;ii++)
   {
     digitalWrite(X_step, HIGH);
     delayMicroseconds(dly*accel);
```

```
      digitalWrite(X_step,LOW);
      delayMicroseconds(10);
      if(ii<120 && accel > 1){
        accel-=0.025;
      }
      if(ii > absDist-120 && ii > 120){
        accel+=0.025;
      }
   }
   *xp=*xp+dist;
}

/*******************************************************************
*************************************************
                                    move y axis
********************************************************************
*************************************************/
void yMove(long dist, unsigned long dly, unsigned long *yp, unsigned int spu)//move
y axis
{

   int ii;
   int absDist;
   float accel = 3;
   absDist = sqrt(dist*dist);//make an absolute value for distance (no negative values)

   if(dist>=1)
   {
      digitalWrite(Y_DIRECTIONPIN,HIGH);// counter clockwise
   }
   if(dist<=0)
   {
      digitalWrite(Y_DIRECTIONPIN,LOW);// clockwise
   }

   for(ii=0;ii<absDist;ii++)
   {
      //create square wave
      digitalWrite(Y_step, HIGH);
      delayMicroseconds(dly*accel);
      digitalWrite(Y_step,LOW);
      delayMicroseconds(10);
      if(ii<120 && accel > 1){
        accel-=0.025;
      }
      if(ii > absDist-120 && ii > 120){
        accel+=0.025;
      }
```

```
    }
    *yp = *yp+dist;
}
/*******************************************************************
**************************************************
                                move z axis
*******************************************************************
*************************************************/
void zMove(long dist, unsigned long dly, long *zp)//move y axis
{

    int ii;
    int absDist;
    absDist = sqrt(dist*dist);//make an absolute value for distance (no negative values)
    float accel = 3;


    if(dist>=1)
    {
        digitalWrite(Z_DIRECTIONPIN,HIGH);// counter clockwise
    }
    if(dist<=0)
    {
        digitalWrite(Z_DIRECTIONPIN,LOW);// clockwise
    }

    for(ii=0;ii<absDist;ii++)
    {
        //create square wave
       digitalWrite(Z_step, HIGH);
       delayMicroseconds(dly*accel);
       digitalWrite(Z_step,LOW);
       delayMicroseconds(10);
       if(ii<120 && accel > 1){
         accel-=0.025;
       }
       if(ii > absDist-120 && ii > 120){
         accel+=0.025;
       }

    }
    *zp = *zp+dist;
}

/*=================================================================
=====================================================
========================================= functions for manual
movement  =============================================//
```

```
===========================================================
=======================================================*/
void jogX(int dist, unsigned long dly)// manual x move
{
  int ii;
  int absDist;
  absDist = sqrt(dist*dist);
  if(dist>=1)
  {
      digitalWrite(X_DIRECTIONPIN,HIGH);// counter clockwise
  }
  if(dist<=0)
  {
      digitalWrite(X_DIRECTIONPIN,LOW);// clockwise
  }

  for(ii=0;ii<absDist;ii++)
  {
     digitalWrite(X_step, HIGH);
     delayMicroseconds(dly);
     digitalWrite(X_step,LOW);
     delayMicroseconds(10);
  }
}
void jogY(int dist, unsigned long dly)
{
  int ii;
  int absDist;
  absDist = sqrt(dist*dist);//make an absolute value for distance (no negative values)

  if(dist>=1)
  {
      digitalWrite(Y_DIRECTIONPIN,HIGH);// counter clockwise
  }
  if(dist<=0)
  {
      digitalWrite(Y_DIRECTIONPIN,LOW);// clockwise
  }

  for(ii=0;ii<absDist;ii++)
  {
      //create square wave
     digitalWrite(Y_step, HIGH);
     delayMicroseconds(dly);
     digitalWrite(Y_step,LOW);
     delayMicroseconds(10);

  }
}
```

```
void jogZ(int dist, unsigned long dly)
{
   int ii;
   int absDist;
   absDist = sqrt(dist*dist);//make an absolute value for distance (no negative values)


   if(dist>=1)
   {
      digitalWrite(Z_DIRECTIONPIN,HIGH);// counter clockwise
   }
   if(dist<=0)
   {
      digitalWrite(Z_DIRECTIONPIN,LOW);// clockwise
   }


   for(ii=0;ii<absDist;ii++)
   {
      //create square wave
     digitalWrite(Z_step, HIGH);
     delayMicroseconds(dly/0.8);
     digitalWrite(Z_step,LOW);
     delayMicroseconds(10);

   }
}
/*********************************************************************
****************************************************
                           read instruction string from serial
**********************************************************************
****************************************************/
void readserial(char *inC,int a, int n)
{
   char inChar; // Where to store the character read
   while(Serial.available() == 0);
   while(Serial.available() > 0) // Don't read unless    // there you know there is data
   {
    if(n < a) // One less than the size of the array
    {
       inChar = Serial.read(); // Read a character
       inC[n] = inChar; // Store it
       n++; // Increment where to write next
       inC[n] = '\0'; // Null terminate the string
    }
    delay(5);
   }
}
```

```
/*************************************************************************
*******************************************************
                           read instructions from recieved string
*************************************************************************
*******************************************************/
void parseString(struct apeture *D, char *str, unsigned int *dps, unsigned int
*spu,unsigned int *intpl, unsigned long *x, unsigned long *y, unsigned int *z,
unsigned long dly, unsigned long *xo, unsigned long *yo, unsigned int *rpm)
{
   char * pch; //temperary variable for holding string
   char tmp[25]; // temp varible for float conversion
   float tempf;  //
   float xf, yf;
   unsigned int G, num = 0;
   int dist;

   //FOR G54 apeture change CODES
   if(str[0] == 'G'&&str[3]=='D'){
      pch = strtok(str,"G D *");
      G = atoi(pch);

      pch = strtok(NULL,"D * ");
      *z = atoi(pch);
      Serial.print("D = "),Serial.println(*z);
   }
   //for G codes
   else if(str[0]=='G'){
      pch = strtok(str,"G *");
      G = atoi(pch);
      switch(G)
      {
         case 1:
            Serial.println("linear interpolation");
            *intpl = 1;
            break;
         case 2:
            Serial.println("circular interpolation clockwise");
            *intpl = 2;
            break;
         case 3:
            Serial.println("circular interpolation counterclockwise");
            *intpl = 3;
            break;
         case 4:
            pch = strtok(NULL," ");
            Serial.print("\n");
            while(pch != NULL){
              Serial.println(pch);
              pch = strtok(NULL," *");
            }
            break;
         case 36:
            Serial.println("Area fill ON");
            break;
         case 37:
            Serial.println("Area fill OFF");
            break;
         case 70:
            Serial.println("units inches (deprecated command)");
            break;
         case 71:
            Serial.println("units MM (deprecated command)");
            break;
         case 74:
            Serial.println("Single Quadrant mode enabled");
            break;
         case 75:
            Serial.println("Multi-Quadrant mode enabled");
         case 90:
            Serial.println("absolute mode (deprecated command)");
            break;
         case 91:
            printf("incremental mode (deprecated command)");
            break;

         default:
            Serial.println("non standard gerber definition");
            break;
      }
   }

   //FOR SETUP INSTRUCTIONS
   else if(str[0]=='%'){
      //for format specification (FS)
      if(str[1]=='F'&&str[2]=='S'){
         if(str[3] == 'L'){
            Serial.println("leading zeroes omitted\n");
         }
         else if(str[3] == 'T'){
            Serial.println("trailing zeroes omitted\n");
         }
         if(str[4]=='A'){
            Serial.println("absolute notation\n");
         }
         else if(str[4]=='I'){
            Serial.println("incremental notation\n");
         }
```

```
          pch = strtok(str,"%FSLAX  Y");
          *dps = pch[1]-48;
          Serial.print("format
"),Serial.print(pch[0]),Serial.print(":"),Serial.println(pch[1]);
      }
      //unit select
      if(str[1]=='M'&&str[2]=='O'){
        if(strcmp("%MOIN*%",str)==0){
            delay(100);
             Serial.println("inches\n");
             *spu = 6350;
          }
          else if(strcmp("%MOMM*%",str)==0){
            delay(100);
             Serial.println("millimetres\n");
             *spu = 250;
          }
           else{
             Serial.println("STRING ERROR");
           }
      }
      //apeture select
      if(str[1]=='A'&&str[2]=='D'){
          char shape = str[6];
          pch = strtok(str,"%AD  C R , *%");
          num = atoi(pch)-10;
          D[num].location = atoi(pch);
          if(shape=='C'){
              strcpy(D[num].type,"circlular");
          }
          else if(shape=='R'){
              strcpy(D[num].type, "rectangular");
          }
          D[num].location = atoi(pch);
          // get diameter of apeture
          pch = strtok(NULL,", *%");
          printf("\n%s\n",pch);
          strcpy(tmp, pch);
          sscanf(tmp,"%f", &tempf);
          D[num].diam = tempf;
        // Serial.print("apeture D"),Serial.print(num),Serial.print("
"),Serial.print(D[num].type),Serial.print(" size
"),Serial.print(D[num].diam),Serial.println("MM");
      }
  }
  else if (str[0]=='X'|| str[0] == 'Y'){
    int yy = 0;
    int dd = 0;
    if (strpbrk(str, "Y")!= NULL){
```

```
    yy = 1;
  }
  if(strpbrk(str,"D")!= NULL){
    dd = 1;
  }
  if(str[0] == 'X'){
      pch = strtok (str,"X Y- D * ");
      *x = atol(pch);
      xf = *x;
      xf/=1*pow(10,*dps);
      *x = (xf*(*spu));

      if(yy == 1){
          pch = strtok (NULL, "Y- D");
          *y = atol(pch);
          yf = *y;
          yf/=1*pow(10,*dps);
          *y = (yf*(*spu));
      }
  }
  else if(str[0] == 'Y'){
      pch = strtok (str, "Y- D *");
      *y = atol(pch);
      yf = *y;
      yf/=1*pow(10,*dps);
      *y = (yf*(*spu));
  }
  if(dd == 1){
    pch = strtok (NULL, "D *");
    *z = atoi(pch);
  }
}
else if(str[0] == 'D')
{
  pch = strtok (str, "D *");
  *z = atoi(pch);
}
else if(str[0] == 'M' && str[1] == '0')
{
  *x = 0, *y = 0, *z = 2;
}
//CODES FOR MANUAL MOVEMENT
else if(str[0] == 'b')
  {
     if(str[1] == 'o'){
              pch = strtok(str,"box y"); // get x offset
              *xo = atol(pch);
              xf = *xo;
              xf/=1*pow(10,*dps);
```

```
            *xo = (xf*(*spu));

            pch = strtok(NULL,"y *"); // get y offset
            *yo = atol(pch);
            yf = *yo;
            yf/=1*pow(10,*dps);
            *yo = (yf*(*spu));
            Serial.print("offsets set. x-"), Serial.print(xf), Serial.print("y: -
"),Serial.println(yf);
     }

     else if(str[1] == 'f'){ // set stepper rpm for feed rate. feed rate = rpm/60 * 1.25 (
1.25 is the pitch of the threaded bar)
       pch = strtok(str, "bf *");
       *rpm = atoi(pch);
       Serial.print("feedrate set to "),Serial.print(*rpm/48),Serial.println("mm//S");
     }

     else if(str[1] == 'Z'){
       pch = strtok(str,"bZ *");
       dist = atoi(pch);
       jogZ(dist, dly);
     }
     else if(str[1] == 'z'){
       pch = strtok(str,"bz *");
       dist = atoi(pch);
       jogZ(0-dist, dly);
     }
     else if(str[1] == 'Y'){
       pch = strtok(str,"bY *");
       dist = atoi(pch);
       jogY(dist, dly);
     }
     else if(str[1] == 'y'){
       pch = strtok(str,"by *");
       dist = atoi(pch);
       jogY(0-dist, dly);
     }
     else if(str[1] == 'X'){
       pch = strtok(str,"bX *");
       dist = atoi(pch);
       jogX(dist, dly);
     }
     else if(str[1] == 'x'){
       pch = strtok(str,"bx *");
       dist = atoi(pch);
       jogX(0-dist, dly);
     }
```

```
   }
}
/*******************************************************************
*****************************************************
                        calculate delay for rpm
*******************************************************************
*****************************************************/
void calcDelay(unsigned int x, unsigned long *dly)
{
   float y=60;
   y = y/200;
   *dly = (y/x*1000000)-10;
}
```

APPENDIX E: - DATASHEETS

Astrosyn my-5602 size 14 stepper motor
http://www.farnell.com/datasheets/30484.pdf
Astrosyn 37-0514 size 17 stepper motor
http://www.rapidonline.com/pdf/37-0514.pdf
L298N
http://www.st.com/web/en/resource/technical/document/datasheet/CD00000240.pdf
L297
http://www.farnell.com/datasheets/1696835.pdf
intel galileo gen2
http://www.intel.com/content/www/us/en/embedded/products/galileo/galileo-g2-datasheet.html