Author & Email:

Ayling, Robin ([r.ayling267@canterbury.ac.uk](mailto:r.ayling267@canterbury.ac.uk))

SCRABEL Work Placement Report

**COURSE:** MCOSS2PLA

**YEAR:** 2

**TUTOR:** Reza Mousoli

**DUE DATE:** 24/05/2017

**DATE HANDED IN:** 24/05/2017

**NAME:** Robin Ayling

Table of Contents

[Placement Overview 3](#_Toc483328016)

[Placement Company 3](#_Toc483328017)

[Team Structure 3](#_Toc483328018)

[Description of Projects 3](#_Toc483328019)

[Raspberry Pi Setup 3](#_Toc483328020)

[Motor 4](#_Toc483328021)

[Printer Simulator (stepV9.py) 4](#_Toc483328022)

[Variables 4](#_Toc483328023)

[Motor() 4](#_Toc483328024)

[DrawLine() 4](#_Toc483328025)

[DrawLineByCoords() 5](#_Toc483328026)

[GoTo() 5](#_Toc483328027)

[Circle() 5](#_Toc483328028)

[DrawPolygon() 5](#_Toc483328029)

[PrintPlot() 5](#_Toc483328030)

[RunFile() and DoIt() 5](#_Toc483328031)

[Robot Arm (ArmV4.py) 5](#_Toc483328032)

[Design 5](#_Toc483328033)

[Variables 6](#_Toc483328034)

[Line() 6](#_Toc483328035)

[CoordsToDegrees() 6](#_Toc483328036)

[DegreesToSteps() 7](#_Toc483328037)

[StepsToData() 7](#_Toc483328038)

[Circle() 7](#_Toc483328039)

[InterpretFile() 7](#_Toc483328040)

[DoIt() 7](#_Toc483328041)

[WriteFile() 7](#_Toc483328042)

[ExecuteFile() 7](#_Toc483328043)

[Pen() 7](#_Toc483328044)

[Motor() 8](#_Toc483328045)

[DinoV1.py 8](#_Toc483328046)

[Testing and Debugging 8](#_Toc483328047)

[Project Completion 8](#_Toc483328048)

[Learning Outcomes 8](#_Toc483328049)

[Python 9](#_Toc483328050)

[3D Design 9](#_Toc483328051)

[Simulation 9](#_Toc483328052)

[Independent work 9](#_Toc483328053)

[Appendix 1 – Provider Assessment 9](#_Toc483328054)

[Appendix 2 – Email Communication 10](#_Toc483328055)

[Appendix 3 – Python Files 10](#_Toc483328056)

[StepV9.py 10](#_Toc483328057)

[ArmV4.py 20](#_Toc483328058)

[DinoV1.py 31](#_Toc483328059)

[Appendix 4 – Printer Simulation Test Images 35](#_Toc483328060)

[Appendix 5 – Debugging Via Print To Screen 38](#_Toc483328061)

[Appendix 6 – Robot Arm Test Data 39](#_Toc483328062)

[Data from CoordsToDegrees() 39](#_Toc483328063)

[Data from DegreesToSteps() 39](#_Toc483328064)

[Appendix 7 – 3D models 41](#_Toc483328065)

[DrawArmV1.stl 41](#_Toc483328066)

[DrawArmV2.stl 41](#_Toc483328067)

[DrawArmV3.stl 42](#_Toc483328068)

[Appendix 8 – Progress Log 42](#_Toc483328069)

[15/2/17 SCRABEL 42](#_Toc483328070)

[20/2/17 Canterbury Campus, Makerspace 42](#_Toc483328071)

[22/2/17 SCRABEL 42](#_Toc483328072)

[24/2/17 Canterbury Campus, Makerspace 42](#_Toc483328073)

[1/3/17 SCRABEL 43](#_Toc483328074)

[4/3/17 Google Hangouts 43](#_Toc483328075)

[8/3/17 SCRABEL 43](#_Toc483328076)

[15/3/17 SCRABEL 43](#_Toc483328077)

[19/3/17 Google Hangouts 43](#_Toc483328078)

[22/3/17 SCRABEL 44](#_Toc483328079)

[29/3/17 SCRABEL 44](#_Toc483328080)

[5/4/17 SCRABEL 44](#_Toc483328081)

[12/4/17 SCRABEL 44](#_Toc483328082)

[19/4/17 SCRABEL 45](#_Toc483328083)

[26/4/17 SCRABEL 45](#_Toc483328084)

[3/5/17 SCRABEL 45](#_Toc483328085)

# Placement Overview

Our placement was at SCRABEL on Christ Church University’s Medway campus led by Saif Ahmed. Here we would complete projects in the field of robotics, electronics and programming. The placement was set to last 8 weeks, although we attended two extra weeks prior and one extra week after. Where much of our work was independent and done from home we would still come in every Wednesday to touch base, discuss projects with Saif and the other members of the team, and test our projects.

# Placement Company

SCRABEL (Stem Cell Research and Adavanced Bio-Engineering Laboratory) is part of the Institute of Medical Sciences. This lab is dedicated to advanced regenerative medicine with aspects of engineering, computing and 3D design. It is led by Saif Ahmed, a consultant orthopaedic surgeon. Saif believes in working from scratch and as such our projects were built, designed and programmed from scratch.

# Team Structure

The team consisted of myself as the back-end programmer, Sabina as electronics, Daniel as front-end programming, and Saif as project leader. Saif directed me to familiarise myself with the motors we were using and how to control them. This led to me designing a program that would simulate the movements of a 3D printer. Later Saif prompted me to come up with my own project idea, to which I decided to make a robotic arm that could draw images based on simple instructions.

Sabina was taught to solder and put in charge of electronics. She was given the project of designing and creating a port expander so that we could control multiple motors on one port. When asked to come up with her own project she decided to make a toy dinosaur that could move around using wheels controlled by the motors. I put together a quick script to test these motors called DinoV1.py.

Daniel was tasked with designing an interpreter for my printer simulator program that would take a text file of simple instructions and execute parts of my code based on those instructions. Later he designed a web interface were one could edit the instruction file and run it to see the resulting image. Daniel also helped me test and debug my programs.

# Description of Projects

## Raspberry Pi Setup

In our first week, we setup our Raspberry Pi’s as they would serve as servers to host the programs. The first step was to download the latest version of raspbian and copy it to the micro sd card of the Pi. Next we connected a connected the pi to a monitor, keyboard and mouse and installed the OS.

Once that was done, then next step was to connect it to the Wi-Fi. I couldn’t connect it to the university Wi-Fi so I set up my phone as a wireless hotspot and connected my Pi to that. From there I used sudo apt-get update to ensure I had the latest edition of the OS.

From here we installed Apache2, php, python and i2c. After this the set up was done and I could start working on my programs. We decided on python as a programming language because it was easy to read and standard for robotics. At the beginning of the placement I didn’t know any python but by the end I was as fluent in it as I am with any programming language.

## Motor

The projects we were working on were based on a particular stepper motor () and as such I was instructed to familiarise myself with how to control it with scripts. I learned that to turn the stepper motor a sequence of data had to be sent to the motor. This sequence was 0x03, 0x06, 0x0c, 0x09. If this sequence is reversed, then the motors move anti-clockwise. Using a port expander I could send multiple instructions to multiple motors, for example 0x33 followed by 0x66 would advance both motors in the clockwise direction. The steps must be consecutive to move the motor.

## Printer Simulator (stepV9.py)

This first project was intended to help me gain an understanding of how to control the motors we were using and to form a base for future projects. The aim was to create a program that would take a destination coordinate and simulated movements of two motors which would move in the x axis and y axis respectively. This would be represented in a 3-dimensional array, with 2 of the dimensions representing the pixels in an image and the third dimension representing the colour of the pixels. Every time the current position changes the array is updated to change the colour of the pixel. This array is then converted to an image and saved in the local directory. The program is designed so that the lines incrementing or decrementing the current coordinates can be replaced with lines that move a motor backwards or forwards, which could be used to control an extruder on a 3D printer.

This project was initially developed so that I could test my code without the need of hardware.

### Variables

Plot: this is a 3-dimensional array which represents an image that is 1024x1024 with 3 spaces for rgb colour values. By default each entry is set to 0.

Func: this is the function that is passed in from the command line and is stored as a string.

xCoord/yCoord: these represent the current position in plot and default at 512,512.

threadCount: this is a variable which keeps track of the number of active threads.

maxSpeed: this is the maximum delay between steps and is usually the default.

### Motor()

This method checks which axis it is incrementing/decrementing (xory) and initiates a for loop of range length. It then adds the direction (-1 or 1) to the relative coordinate, with a check to ensure it doesn’t go out of the bounds of the array. If draw is equal to 1 then the current coordinate is populated with [254,0,0] which is the RGB value for red. The method then waits for the during of the wait variable which determines the time it takes to finish the method. The final coordinate of the line is made blue so that we can see where the line ended in the final picture. Finally, threadCount is increased by 1 so that we know the thread is finished.

### DrawLine()

This method takes an angle and a length and calculates the required distance in x and y to get to the same destination (xlength and ylength). It then calculates the delay, ensuring that the smallest delay is the constant maxSpeed. The direction is calculated by checking the angle of the line. threadCount is set to 0 and two threads are created, each bound to an instance of the Motor() method. A while loop ensures that the program waits until both threads have finished before continuing. The coordinates are updated and the method ends.

### DrawLineByCoords()

This method works much the same as the previous one, only that it calculates xlength and ylength based on the destination coordinates.

### GoTo()

This method was designed to move the cursor to a designated destination without drawing a line to it. It is identical to DrawLineByCoords() except that the final parameter sent to the Motor() method is a 0. This method could be removed entirely and replaced with an optional parameter in the DrawLine() method to set the Draw variable.

### Circle()

This method was designed to approximate a circle as close as one can with pixels/motor steps. This method works by using loops to draw 8 arcs that together make up a full circle. It performs various checks to get the current position in the loop and adjust the direction based on that. The start and stop variables allow the user to draw partial circles, if left blank these will default to 0 and 360 to draw a full circle. The variable a is used to keep track of the current position in the main loop. However, the total number of loops does not match up to the 360 degrees of a circle, so to compensate for this the start and stop values are converted with the formula degrees\*radius\*pi/200. The program then loops through and increments the x or y position of the cursor as required. Should the loop be between the stop and start value then the program will populate the plot array or move a step on the motor.

### DrawPolygon()

This method was originally designed to approximate a circle until I designed to above method, however I decided to keep it and rename it DrawPolyogon(). This is one of the simpler and more elegant methods in my program. It works by dividing 360 by the amount of faces (theta) and then draws a line based on this calculation and the radius. It then increments the angle by theta and draws another line until the loop has reached the stop value or until it hits 360 degrees.

### PrintPlot()

This method uses the PIL module to convert the plot array into a png image. It also has some code to add text to the image, however this text does not simulate a printer and is only used for the image. The image is saved as htmlimage.png if the mode variable is set to html. Otherwise the image is saved as image.png.

### RunFile() and DoIt()

These methods were developed by Daniel as part of his php file. RunFile() works by opening the specified file and looping through the lines in the file. Each line is read and split into two parts; a command and a list of parameters. These command and parameters are then sent to the DoIt function where it simply checks each command and calls the respective methods with the relevant parameters. Once RunFile() has looped through every line in the instruction file then it closes the file and exits the method.

## Robot Arm (ArmV4.py)

### Design

The robot arm was initially design with OpenSCAD 3D design software. It was designed to move horizontally in the x and y axis with a pen held at the end. It has two joints, one at the elbow and one at the shoulder, which are controlled by motors. The motor which controls the elbow is mounted directly onto the upper arm so that it is always a fixed distance from the elbow as the arm moves. It is also mounted near the other motor to keep the centre of gravity close to the shoulder so that the arm is not weighed down. The elbow motor is connected to the elbow via a belt which turns a gear attached to the forearm. The arm was originally intended to include a joint that raised the arm up and lifted the pen from the canvas, however this was dropped due to time constraints. As such, the arm is only capable of drawing pictures with a continuous line.

### Variables

Bus: this is an instance of bus from the module smbus and is used to write data to the motors.

Func: as in the previous program this is the function passed in from the command line.

Device: this is the device number from the i2c chip. It can be changed in the case that another device is active.

Port1/port2: these store the active ports of the port expander and can be changed as desired.

maxSpeed: this is the same as the previous program, the minimum speed between motor steps.

Segment: this determines the length of the segments that the line in the method Line() is broken into and thus the accuracy of the drawn line(the smaller the segment the more accurate).

stepsToDegrees: this is a constant that holds the ratio of motor steps to degrees.

Step1/step2: these are constants that hold the list of steps for each motor.

cStep1/cStep2: these hold the current position in the previously mentioned list of steps.

Draw: this determines whether the pen is on the canvas or not. This was never implemented however.

Elbow: this is the current interior angle of the elbow or the arm.

Shoulder: this is the current angle of the shoulder measured from south.

Forearm: this is length of the forearm, accurate to one tenth of a millimetre.

Upperarm: this is length of the upperarm, accurate to one tenth of a millimetre.

Coord: this stores the current coordinates of the pen in the form [x,y]

Excess1/excess2: this keeps track of the error in the calculation of the method DegreesToSteps() so that it can correct itself next time.

### Line()

This method takes a destination coordinate and constructs a line (c) between it and the current position and calculates the horizontal and vertical distances to that point. Using the ratio between the horizontal and vertical distances it splits the line c into sections with a length equal to the variable segment. It then sends each coordinate for each section to the method CoordsToDegrees() and appends the results to the list degrees[]. It updates the coordinates and sends degrees[] to DegreesToSteps() and stores the results in the variable data. the method then returns data.

### CoordsToDegrees()

This method is based on the principle that the destination coordinate can be described by the angles between the joints and the lengths of the sections of the arm. The method constructs a right-angled triangle abc where the hypotenuse c is the distance between the destination and the origin(in this case the shoulder joint). Using the lengths of the upperarm and the forearm and the hypotenuse the method calculates the interior angle of the elbow. The angle of the shoulder is similarly calculated except that it has exceptions depending on which quadrant the current position resides in. these quadrants are adjusted with a semi-circle of radius equal to the forearm. From these angles, it then calculates the difference between them and the current angles of the elbow and shoulder. These numbers are then returned in the form [angle1, angle2].

### DegreesToSteps()

This method takes a list of pairs of angles in the form [[angle1, angle2], [angle1, angle2], …] and iterates through each pair of angles. It then calculates the number of steps needed to turn the motor for the required degrees. It then distributes the steps evenly, for example if it works out to 4 steps in one motor and 2 for the other then it will convert it to [1,0], [1,1], [1,0], [1,1] with negatives if the motor needed to move in the other direction. These pairs of steps are passed to the method StepToData() which returns the byte data which will turn each motor in the desired direction. There is an except for when the steps equal [0,0], in which case the loop skips the iteration and continues until there is a non-zero entry. The byte data is appended to the list data[] and the list is returned.

### StepsToData()

This method receives a pair of numbers and checks the sign of each number. Depending on the sign of the number the method will check the current step (cStep1/cStep2) and picks the next step from the list of steps. This method was necessary so that the lists of steps can be treated as cyclical arrays. Finally the two values are added together (with 0 as a default if no step was necessary), converted to hex for human readability and returned.

### Circle()

This method is very similar to the DrawPolygon() method in the previous program except that it calculation each destination coordinate from the length and angle using trigonometry.

### InterpretFile()

This method is based on Daniel’s method in the previous program. It opens a file and creates a while loop iterating through each line of the file. For each line, it splits the line into a list, with the first entry becoming the command and the remaining entries becoming a list of parameters. The command and the parameters are then passed into the DoIt() method and the results are appended to the list data[]. The program then checks if there is another line to read, if not then the list data[] is returned.

### DoIt()

This method is also based on Daniel’s method from the previous program. This is a simple series of if statements that execute different function depending of what command was passed in from the previous method.

### WriteFile()

This loops through the list of data and writes each entry to a newline in the specified text file.

### ExecuteFile()

The method reads each line of the designated file and appends each value to a list. It then loops through this list and if the entry is a ‘P’ then it calls the Pen() function. Otherwise, it writes the byte data to the specified device and port.

### Pen()

This method toggles whether the pen is on or off the paper, however this was never implemented.

### Motor()

This is a test method which was used to test the motors. It takes an angle in degrees and converts it into the number of steps required. Then it loops through the steps. Depending on the number given it will move a different motor. 1 moves the first motor, 2 moves the second, and 3 moves both. Currently the test method only moves the motors clockwise where a more complete test would move them anti-clockwise as well and alternate directions.

## DinoV1.py

This is a test file I put together to help Sabina test her dinosaur. It contains four methods, two to turn the dinosaur and two to make it move back and forward. The turning methods have steps that alternate direction to make the wheels move in opposite directions which makes the dinosaur turn a specified number of degrees. The back and forward method simply move the motors forward or backward in unison for a specified number of rotations, which should move the dinosaur forward.

## Testing and Debugging

I used three methods to test my code: printing to the screen, outputting an image, and outputting a text file. These allowed me to effectively debug my programs with the aid of Daniel.

I used the Print() method in python to print certain messages depending where in the code the program is. For example, I would print the message “DrawLine” when the DrawLine method was called. I would also print the current iteration of any loop along with the values of relevant variables. This was especially useful when the program would crash as it allowed me to pinpoint exactly where the program was when it crashed.

I saved the plot array in the printer simulator as an image. This allowed me to see the path traced by the simulated cursor and check if it was the correct shape. This was particularly useful when I was trying to make a circle, as it took many attempts to get the correct shape, resulting in many obscure and unexpected shapes (see appendix).

My last method of testing was saving lists to text files which was used in my robotic arm project. This was very useful as it allowed me to save each list to a text file and check it visually between each conversion to ensure that the data was correct. It also allowed me to tell if the steps were distributed properly in the DegreesToSteps() method.

## Project Completion

The result of the first project is that I have a program that can print an image with made of lines, circles, or regular polygons. Further, due to the file interpretation by Daniel, it can reference another file which defines a more complicated shape. For example, we can create a file that defines a smiley face and use this as a new shape. I could have improved this program by implementing a method that takes a list of coordinates and draws an irregular polygon between the points.

The results for the arm project are mixed. On the positive side, the arm moves with each joint capable of moving a specified number of degrees. On the downside, it doesn’t quite move in as straight a line as I would want. I suspect this is due to a mix up of the motors in the code and could be fixed with more time. I would have liked to implement a way for the arm to lift off the canvas, which also would have required more time.

# Learning Outcomes

I learned a variety of skills during this placement, ranging from technical skills such as learning to program in python and soft skills like time management.

## Python

The main skill I developed was programming in Python. In the beginning of my placement I did not know anything about python. By the end of the placement I had learned how to research and import modules into python and the concepts involved in multithreading to get multiple functions running concurrently. At this point I would say I am as fluent in Python as I am in any programming language.

## 3D Design

I did my 3D design work in OpenSCAD which I had previous experience in. However, what I didn’t have experience in was the rapid prototyping and re-designing that is present in the workplace. With the aid of 3D printers, it is possible to design a model, print and test it, and then re-design it in a quick succession. During this process Saif and I would discuss faults and benefits of the design which is how we came to settle of the design for the arm.

## Simulation

I learned to simulate movements in 2-dimensional space as data in an array. This was necessary as the hardware was not yet ready and I had no other way to test my program. This proved very useful for debugging and resulted in its own project.

## Independent work

Most of my work for this placement was done at home, meaning I had to manage my time to ensure I could get all the work done. I also had to maintain communication with Saif both in keeping him updated with my work via email and being available for google hangouts meetings. I also used this time to teach myself python as I had no prior teaching of the language.

# Appendix 1 – Provider Assessment

|  |  |  |
| --- | --- | --- |
| Industrial Placement Provider Assessment Schedule | | |
|  | Yes | No |
| Was the student’s timekeeping and attendance reliable? | Yes |  |
| Did the student adopt a methodical approach to the work he was asked to do? | Yes |  |
| Was the student capable of working independently of his supervisor? | Yes |  |
| Did the student fit into the team that he was working with? | Yes |  |
| Did the student use his initiative in developing the scope of the work? | Yes |  |
| Did the student document his work? | Yes |  |
| Did the student adequately test his work? | Yes |  |
| Were the deliverables of the placement met? | Yes |  |
| Were the deliverables of the placement exceeded? | Yes |  |
| Would you be willing to provide an industrial placement next year? | Yes |  |

|  |
| --- |
| Industrial Placement Providers Comments |
| Please provide any comments that you wish to make in this box |
| Robyn has been an excellent worker. Considering he has had no previous programming experience in Python or embeddded devices or hardware interfacing he managed to develop and extend the scope of the work in the short interval he was with me.   1. He learnt Python coding and applied it 2. He created a software simulation of the robotic interface that can be used to test the code while hardware was in development 3. He learnt Unipolar stepper motor control over a port expander using I2C and driver boards 4. He devised algorithms for synchronous motor control using multithreaded and single threaded coding 5. He designed 3D objects in OpenSCAD and these were used as the basis for the final draw Arm |

Signed…Saif Ahmed Position……Research Projects Lead SCRABEL..

# Appendix 2 – Email Communication

# Appendix 3 – Python Files

## StepV9.py

import time

import sys

import math

import threading

import numpy as np

from PIL import Image

from PIL import ImageFont

from PIL import ImageDraw

print('script started')

plot = np.zeros( (1024,1024,3), dtype=np.uint8 )

func = sys.argv[1]

xCoord = 512

yCoord = 512

threadCount = 0

maxSpeed = 0.01

def PrintPlot(mode=''):

global plot

img = Image.fromarray(plot, 'RGB')

if mode == 'html':

img.save('htmlimage.png')

else:

img.save('image.png')

img = Image.open("image.png")

draw = ImageDraw.Draw(img)

# font = ImageFont.truetype(<font-file>, <font-size>)

font = ImageFont.truetype("PressStart2P-Regular.ttf", 16)

# draw.text((x, y),"Sample Text",(r,g,b))

draw.text((500, 500),"Sample Text",(0,255,0),font=font)

img.save('sample-out.png')

print("printed")

def Motor(wait, length, direction, xory, draw):

global threadCount

global xCoord

global yCoord

global plot

print('thread started')

length = int(length)

if xory == 'x':

for i in range(0,length):

xCoord += direction

if xCoord > 1023:

xCoord -= 1023

if draw == 1:

plot[xCoord,yCoord] = [254,0,0]

time.sleep(wait)

elif xory == 'y':

for i in range(0,length):

yCoord += direction

if yCoord > 1023:

yCoord -= 1023

if draw == 1:

plot[xCoord,yCoord] = [254,0,0]

time.sleep(wait)

if draw == 1:

plot[xCoord,yCoord] = [0,0,254]

threadCount += 1

def DrawLine(angle, length):

global threadCount

global xCoord

global yCoord

xwait = maxSpeed

ywait = maxSpeed

angle = angle % 360

xlength = round(length \* abs(math.cos(math.radians(angle))))

ylength = round(length \* abs(math.sin(math.radians(angle))))

xdir = 1

ydir = 1

xy = 1

yx = 1

if xlength > 0 and ylength > 0:

xy = xlength / ylength

yx = ylength / xlength

if xy < 1:

xwait = xwait / xy

elif yx < 1:

ywait = ywait / yx

if (angle >= 0 and angle < 90) or (angle >= 270 and angle < 360):

xdir = -1

if(angle >= 180 and angle < 360):

ydir = -1

threadCount = 0

print('starting threads')

if \_\_name\_\_ == '\_\_main\_\_':

x = threading.Thread(target=Motor, args=(xwait,xlength,xdir,'x',1))

y = threading.Thread(target=Motor, args=(ywait,ylength,ydir,'y',1))

x.start()

y.start()

while (threadCount < 2):

time.sleep(maxSpeed)

print(xCoord)

print(yCoord)

return

def DrawLineByCoords(x,y):

global threadCount

global xCoord

global yCoord

xwait = maxSpeed

ywait = maxSpeed

xlength = abs(xCoord - x)

ylength = abs(yCoord - y)

xdir = 1

ydir = 1

xy = 1

yx = 1

if xlength == 0:

xy = 1

elif ylength == 0:

yx = 1

else:

xy = xlength/ylength

yx =ylength/xlength

if xy < 1 and xy != 0:

xwait = xwait / xy

elif yx < 1 and yx != 0:

ywait = ywait / yx

if x < xCoord:

xdir = -1

if y < yCoord:

ydir = -1

threadCount = 0

if \_\_name\_\_ == '\_\_main\_\_':

x = threading.Thread(target=Motor, args=(xwait,xlength,xdir,'x',1))

y = threading.Thread(target=Motor, args=(ywait,ylength,ydir,'y',1))

x.start()

y.start()

while (threadCount < 2):

time.sleep(maxSpeed)

print(xCoord)

print(yCoord)

return

def GoTo(x,y):

global threadCount

global xCoord

global yCoord

'''

xwait = maxSpeed

ywait = maxSpeed

xlength = abs(xCoord - x)

ylength = abs(yCoord - y)

xdir = 1

ydir = 1

xy = 1

yx = 1

if xlength == 0:

xy = 1

elif ylength == 0:

yx = 1

else:

xy = xlength/ylength

yx =ylength/xlength

if xy < 1 and xy != 0:

xwait = xwait / xy

elif yx < 1 and yx != 0:

ywait = ywait / yx

if x < xCoord:

xdir = -1

if y < yCoord:

ydir = -1

threadCount = 0

if \_\_name\_\_ == '\_\_main\_\_':

x = threading.Thread(target=Motor, args=(xwait,xlength,xdir,'x',0))

y = threading.Thread(target=Motor, args=(ywait,ylength,ydir,'y',0))

x.start()

y.start()

while (threadCount < 2):

time.sleep(maxSpeed)

print(xCoord)

print(yCoord)

'''

xCoord=x

yCoord=y

return

def Circle(r, start = 0, stop = 360):

global xCoord

global yCoord

stop = stop\*r/200\*math.pi

start = start\*r/200\*math.pi

x = 0

y = r

a = 0

r2 = int((r\*\*2/2)\*\*0.5)

count = 'y'

xDirection = 1

yDirection = -1

for j in range(0,8):

if j == 2 or j == 6:

xDirection \*= -1

if j == 4:

yDirection \*= -1

if j == 0 or j == 4:

x += xDirection

for i in range(0,r2):

x += xDirection

a += 1

if (abs(r\*\*2 - y\*\*2)\*\*0.5) < abs(x):

y += yDirection

if a >= start and a <= stop:

plot[(x+yCoord)%1024,(y+xCoord)%1024] = [254,0,0]

if a > stop:

break

if j == 1 or j == 5:

for i in range(0,r2):

y += yDirection

a += 1

if r-(abs(r\*\*2 - x\*\*2)\*\*0.5) < r-abs(y):

x += xDirection

if a >= start and a <= stop:

plot[(x+yCoord)%1024,(y+xCoord)%1024] = [254,0,0]

if a > stop:

break

if j == 2 or j == 6:

for i in range(0,r2):

y += yDirection

a += 1

if (abs(r\*\*2 - y\*\*2)\*\*0.5) < abs(x):

x += xDirection

if a >= start and a <= stop:

plot[(x+yCoord)%1024,(y+xCoord)%1024] = [254,0,0]

if a > stop:

break

if j == 3 or j == 7:

for i in range(0,r2):

x += xDirection

a += 1

if r-(abs(r\*\*2 - x\*\*2)\*\*0.5) < r-abs(y):

y += yDirection

if a >= start and a <= stop:

plot[(x+yCoord)%1024,(y+xCoord)%1024] = [254,0,0]

if a > stop:

break

if a > stop:

break

def DrawPolygon(r,faces,start = 0,stop = 360):

theta = int(360 / faces)

length = math.cos(math.radians(90 - theta / 2)) \* r \* 2

for i in range(start, stop,theta):

DrawLine(i,length)

print('drew circle')

def RunFile(fileName):

file = open(fileName)

line = file.readline()

while line:

instruction = line.split()

command = instruction[0]

parameters = instruction[1:]

doIt(command,parameters)

line = file.readline()

file.close()

def doIt(command,parameters):

if command == 'GT':

GoTo(int(parameters[0]), int(parameters[1]))

elif command == 'DC':

if len(parameters) == 1:

Circle(int(parameters[0]))

elif len(parameters) == 3:

Circle(int(parameters[0]),int(parameters[1]),int(parameters[2]))

elif command == 'DL':

DrawLine(float(parameters[0]),int(parameters[1]))

elif command == 'RF':

RunFile(parameters[0])

if func == "RF":

RunFile(sys.argv[2])

PrintPlot(sys.argv[3])

## ArmV4.py

import time

import smbus

import sys

import math

import threading

import numpy as np

from PIL import Image

print('script started')

global threadCount

global shoulder

global elbow

global forearm

global upperarm

global maxSpeed

global stepsToDegrees

global step1

global step2

global draw

global segment

global motor1Count

global motor2Count

bus = smbus.SMBus(1)

func = sys.argv[1]

device = 0x27

port1 = 0x14

port2 = 0x15

maxSpeed = 0.01 #delay between motor steps

segment = 10 #determines accuracy of line

stepsToDegrees = 5.8 #ratio of steps to degrees

step1 = [0x03,0x06,0x0c,0x09] #values to move motor1

step2 = [0x30,0x60,0xc0,0x90] #values to move motor2

cStep1 = 0x03 #current position in steps for motor1

cStep2 = 0x30 #current position in steps for motor2

draw = 0 #0 = pen off paper, 1 = pen on paper

elbow = 90 #position of motor1

shoulder = 90 #position of motor2

forearm = 1430 #length of forearm

upperarm = 1300 #length of upperarm

coord = [-upperarm,forearm] #current position of pen

excess1 = 0 #error in stepsToAngle for motor1

excess2 = 0 #error in stepsToAngle for motor2

#bus.write\_byte\_data(device, 0x00, 0x00)

#bus.write\_byte\_data(device, 0x01, 0x00)

if func == "IF":

data = InterpretFile(sys.argv[2])

WriteFile(data,sys.argv[3])

elif func == "M":

Motor(sys.argv[2],sys.argv[3])

elif func == "EF":

ExecuteFile(sys.argv[2])

def InterpretFile(fileName): #reads file, splits line into command and params, passes command/params to DoIt and appends results to data[]

data = []

file = open(fileName)

line = file.readline()

while line:

instruction = line.split()

command = instruction[0]

parameters = instruction[1:]

data.append(DoIt(command,parameters))

line = file.readline()

file.close()

return data

def DoIt(command,parameters):

data = []

if command == 'C':

if len(parameters) == 2:

data = Circle(int(parameters[0]),int(parameters[1]))

elif len(parameters) == 4:

data = Circle(int(parameters[0]),int(parameters[1]),int(parameters[2]),int(parameters[3]))

elif command == 'L':

data = Line(float(parameters[0]),int(parameters[1]))

elif command == 'P':

data = Pen(parameters[0])

elif command == 'IF':

data = InterpretFile(parameters[0])

return data

def WriteFile(data,fileName): #writes list to file

output = open(fileName,"w")

for i in range(len(data)):

for j in range(len(data[i])):

output.write(str(data[i][j]) + "\n")

print("loop")

output.close()

def ExecuteFile(fileName):

data = []

i=0

file = open(fileName)

line = file.readline()

while line:

i+=1

data.append(line[:-1]) #removes new line character, appends to data[]

line = file.readline()

file.close()

for i in range(len(data)): #iterates through data[], calls functions based on item in list

if data[i] == "P":

Pen()

else:

bus.write\_byte\_data(device,port1,int(data[i]))

time.sleep(maxSpeed)

print("yes")

def Line(x,y):

global coord

global segment

degrees = []

#constructs a right-angled triangle abc with hypotenuse between current coord and destination

a = abs(coord[0] - float(x)) #horizontal

b = abs(coord[1] - float(y)) #verticle

c = (a\*\*2 + b\*\*2)\*\*0.5 #hypotenuse

#hyp is split into segments of length segment, new coordinates are calculated at each segment

ratio = segment/c

xInc = a\*ratio #calculates increment for x direction per segment

yInc = b\*ratio #calculates increment for y direction per segment

d = coord[0]

e = coord[1]

i = 0

while(i < c - segment): #iterates between segments, increments coords

if coord[0] < x:

d += xInc

elif coord[0] > x:

d -= xInc

if coord[1] < y:

e += yInc

elif coord[1] > y:

e -= yInc

degrees.append(CoordsToDegrees(d,e)) #inputs coords, returns degrees required to reach coord [angle1,angle2], appends to degrees[]

i += segment

degrees.append(CoordsToDegrees(x,y)) #for remainder segment

coord[0] = x

coord[1] = y

data = DegreesToSteps(degrees) #inputs list of degrees, returns list of steps

return data

def CoordsToDegrees(x,y): #takes coords, returns angles to move

global maxSpeed

global forearm

global upperarm

global elbow

global shoulder

#constructs right-angled triangle xyz

x = float(x)

y = float(y)

z = (x\*\*2 + y\*\*2)\*\*0.5

#constructs non-right-angled triangle abc where c=z

a = forearm

b = upperarm

c = z

C = math.degrees(math.acos((a\*\*2 + b\*\*2 - c\*\*2)/(2\*a\*b))) #angle oppsite side c

A = math.degrees(math.acos((b\*\*2 + c\*\*2 - a\*\*2)/(2\*b\*c))) #angle oppsite side a

#calculates angle Y in triangle xyz

if x == 0:

Y = 90

elif x > 0 and y == 0:

Y = 180

elif x <= 0 and y == 0:

Y = 0

else:

Y = math.degrees(math.acos((x\*\*2 + z\*\*2 - y\*\*2)/(2\*x\*z)))

Y = 180 - Y

#calculates angle D, D = angle from negative Y axis to line c

if ((x-upperarm)\*\*2+(y)\*\*2)\*\*0.5 < forearm and x < 0:

D = 90 + Y - A

elif x <= 0 or ((x)\*\*2+(y-upperarm)\*\*2)\*\*0.5 < forearm:

D = 90 + Y - A

elif x > 0:

D = 270 - Y - A

else:

print("Error: coordinates out of bounds " + str(x) + " " + str(y))

return

#calculates difference in current postions and required postions

angle1 = elbow - C

angle2 = D - shoulder

elbow = C

shoulder = D

return [angle1, angle2]

def DegreesToSteps(params): #takes degrees, returns steps

data = []

global excess1

global excess2

for j in range(len(params)): #iterates through list of degrees

steps1Dir = 1 #direction of motor1, 1=clockwise, -1=anticlockwise, 0=dont move

steps2Dir = 1

steps1 = float(params[j][0]) \* stepsToDegrees #converts degrees to steps

steps2 = float(params[j][1]) \* stepsToDegrees

steps1 += excess1 #adds previous error to steps

steps2 += excess2

#updates excess, floors steps, exceptions to correct for modulus on negative numbers

if steps1 > 0:

excess1 = steps1%1

steps1 = math.floor(steps1)

else:

excess1 = steps1%1\*-1

steps1Dir\*=-1

steps1 = math.floor(steps1)+1

if steps2 > 0:

excess2 = steps2%1

steps2 = math.floor(steps2)

else:

excess2 = steps2%1-1

steps2Dir\*=-1

steps2 = math.floor(steps2)+1

if max(steps1,steps2) == 0: #if steps both = 0 then continue to next iteration

continue

else:

ratio = abs(min(steps1,steps2)/max(steps1,steps2)) #calculates ratio between steps

count = ratio + 0.5

#iterates between largest of steps, distributes steps according to ratio

if steps1 > steps2:

for i in range(int(abs(steps1))):

if count > 1:

data.append(StepsToData([steps1Dir,steps2Dir])) #converts directions into next step, appends to list

count = count%1

count += ratio

elif count >= 0 and count < 0.999:

data.append(StepsToData([steps1Dir,0]))

count += ratio

else:

data.append(StepsToData([steps1Dir,steps2Dir]))

count = ratio

elif steps1 < steps2:

for i in range(int(abs(steps2))):

if count > 1:

data.append(StepsToData([steps1Dir,steps2Dir]))

count = count%1

count += ratio

elif count >= 0 and count < 0.999:

data.append(StepsToData([0,steps2Dir]))

count += ratio

else:

data.append(StepsToData([steps1Dir,steps2Dir]))

count = ratio

else:

for i in range(int(steps2)):

data.append(StepsToData([steps1Dir,steps2Dir]))

#data.append([steps1,steps2])

return data

def StepsToData(step): #receives directions e.g. [1,1], returns next steps [0x33]

global step1

global step2

global cStep1

global cStep2

result1 = 0x00 #default result

result2 = 0x00

#checks current step and direction, returns next step

if step1.index(cStep1) == 3 and step[0] == 1:

result1 += step1[0]

cStep1 = step1[0]

elif step1.index(cStep1) == 0 and step[0] == -1:

result1 += step1[3]

cStep1 = step1[3]

elif step[0] != 0:

result1 += step1[step1.index(cStep1)+int(step[0])]

cStep1 = step1[step1.index(cStep1)+int(step[0])]

if step2.index(cStep2) == 3 and step[1] == 1:

result2 += step2[0]

cStep2 = step2[0]

elif step2.index(cStep2) == 0 and step[1] == -1:

result2 += step2[3]

cStep2 = step2[3]

elif step[1] != 0:

result2 += step2[step2.index(cStep2)+int(step[1])]

cStep2 = step2[step2.index(cStep2)+int(step[1])]

return hex(result1 + result2) #hex conversion for readability

def Circle(r,faces,start = 0,stop = 360):

data = []

theta = int(360 / faces)

length = math.cos(math.radians(90 - theta / 2)) \* r \* 2

for i in range(start, stop,theta):

x = coord[0] + math.cos(i)\*length

y = coord[1] + math.sin(i)\*length

data.append(Line(x,y))

return data

def Pen(): #stub method - to toggle pen on paper

global draw

if draw == 1:

draw = 0

time.sleep(0.5)

else:

draw = 1

time.sleep(0.5)

return "D"

def Motor(motor, angle): #test method to move motors

if motor == 1:

for i in range(angle\*stepsToDegrees):

bus.write\_byte\_data(device, port, 0x03)

i += 1

time.sleep(maxSpeed)

bus.write\_byte\_data(device, port, 0x06)

i += 1

time.sleep(maxSpeed)

bus.write\_byte\_data(device, port, 0x0c)

i += 1

time.sleep(maxSpeed)

bus.write\_byte\_data(device, port, 0x09)

time.sleep(maxSpeed)

if motor == 2:

for i in range(angle\*stepsToDegrees):

bus.write\_byte\_data(device, port, 0x30)

i += 1

time.sleep(maxSpeed)

bus.write\_byte\_data(device, port, 0x60)

i += 1

time.sleep(maxSpeed)

bus.write\_byte\_data(device, port, 0xc0)

i += 1

time.sleep(maxSpeed)

bus.write\_byte\_data(device, port, 0x90)

time.sleep(maxSpeed)

if motor == 0:

for i in range(angle\*stepsToDegrees):

bus.write\_byte\_data(device, port, 0x33)

i += 1

time.sleep(maxSpeed)

bus.write\_byte\_data(device, port, 0x66)

i += 1

time.sleep(maxSpeed)

bus.write\_byte\_data(device, port, 0xcc)

i += 1

time.sleep(maxSpeed)

bus.write\_byte\_data(device, port, 0x99)

time.sleep(maxSpeed)

#bus.write\_byte\_data(device,port1,0x00)

print("script ended")

## DinoV1.py

import time

import smbus

import sys

import math

import threading

import numpy as np

from PIL import Image

print('script started')

global maxSpeed

global stepsToAngle

bus = smbus.SMBus(1)

func = sys.argv[1]

device = 0x27

port1 = 0x14

port2 = 0x15

maxSpeed = 0.01

stepsToAngle = 5.8

bus.write\_byte\_data(device, 0x00, 0x00)

bus.write\_byte\_data(device, 0x01, 0x00)

def RunFile(fileName):

file = open(fileName)

line = file.readline()

while line:

instruction = line.split()

command = instruction[0]

parameters = instruction[1:]

doIt(command,parameters)

line = file.readline()

file.close()

def doIt(command,parameters):

if command == 'GT':

GoTo(int(parameters[0]), int(parameters[1]),int(parameters[2]))

elif command == 'DC':

if len(parameters) == 2:

DrawCircle(int(parameters[0]),int(parameters[1]))

elif len(parameters) == 4:

DrawCircle(int(parameters[0]),int(parameters[1]),int(parameters[2]),int(parameters[3]))

elif command == 'DL':

DrawLine(float(parameters[0]),int(parameters[1]))

elif command == 'RF':

RunFile(parameters[0])

elif command == "D":

coord = [-arm1,arm2]

Draw()

def Right(angle):

for i in range(angle\*stepsToAngle):

bus.write\_byte\_data(device,port1,0x39)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0x6c)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0xc6)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0x93)

time.sleep(wait)

def Left(angle):

for i in range(angle\*stepsToAngle):

bus.write\_byte\_data(device,port1,0x93)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0xc6)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0x6c)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0x39)

time.sleep(wait)

def Forward(distance):

for i in range(distance):

bus.write\_byte\_data(device,port1,0x33)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0x66)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0xcc)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0x99)

time.sleep(wait)

def Backward(distance):

for i in range(distance):

bus.write\_byte\_data(device,port1,0x99)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0xcc)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0x66)

time.sleep(wait)

bus.write\_byte\_data(device,port1,0x33)

time.sleep(wait)

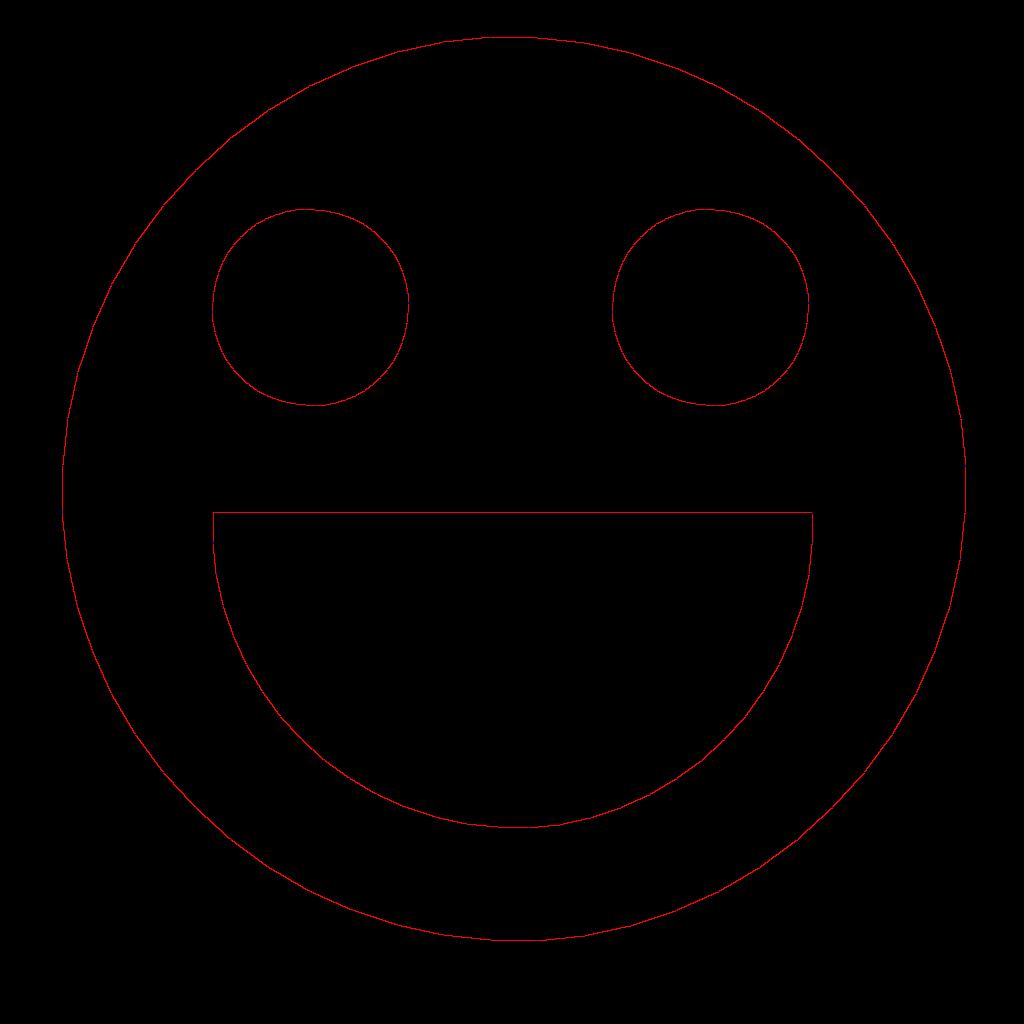
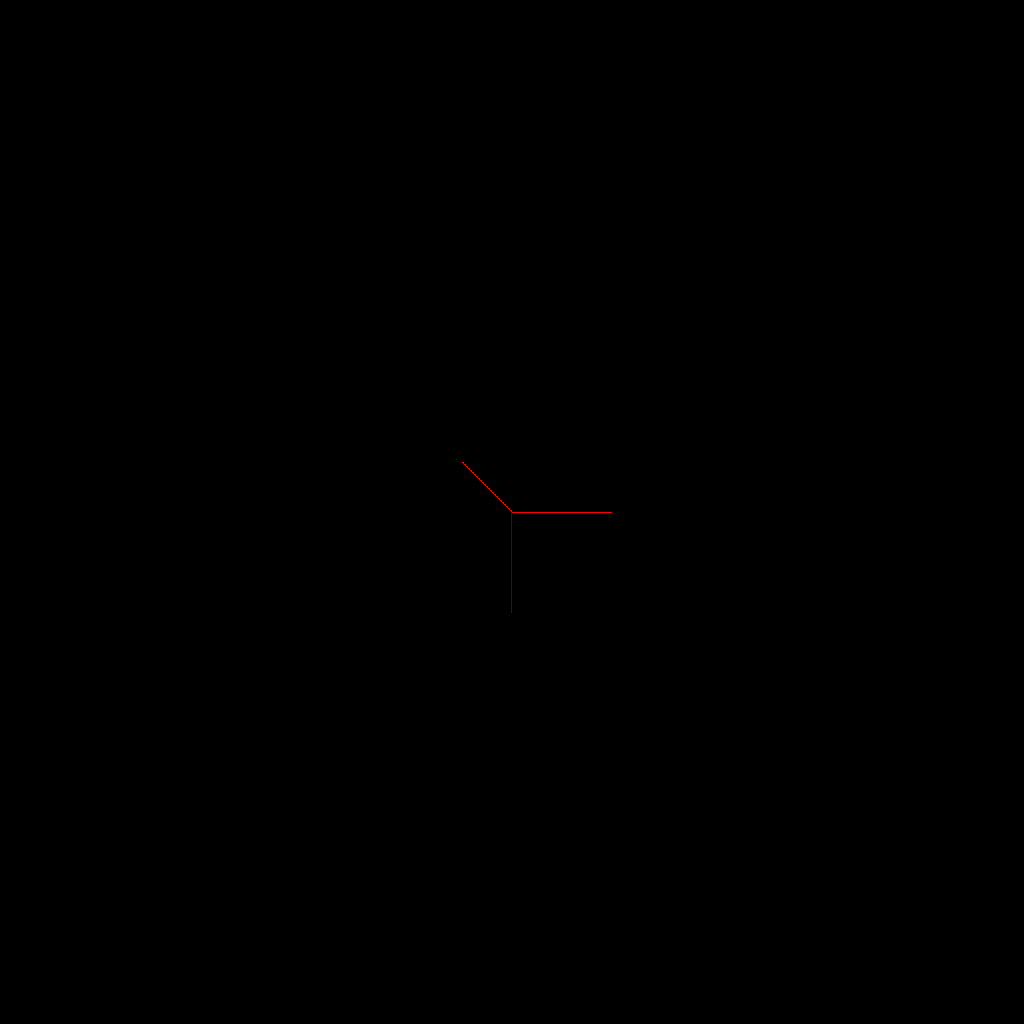
if func == "RF":

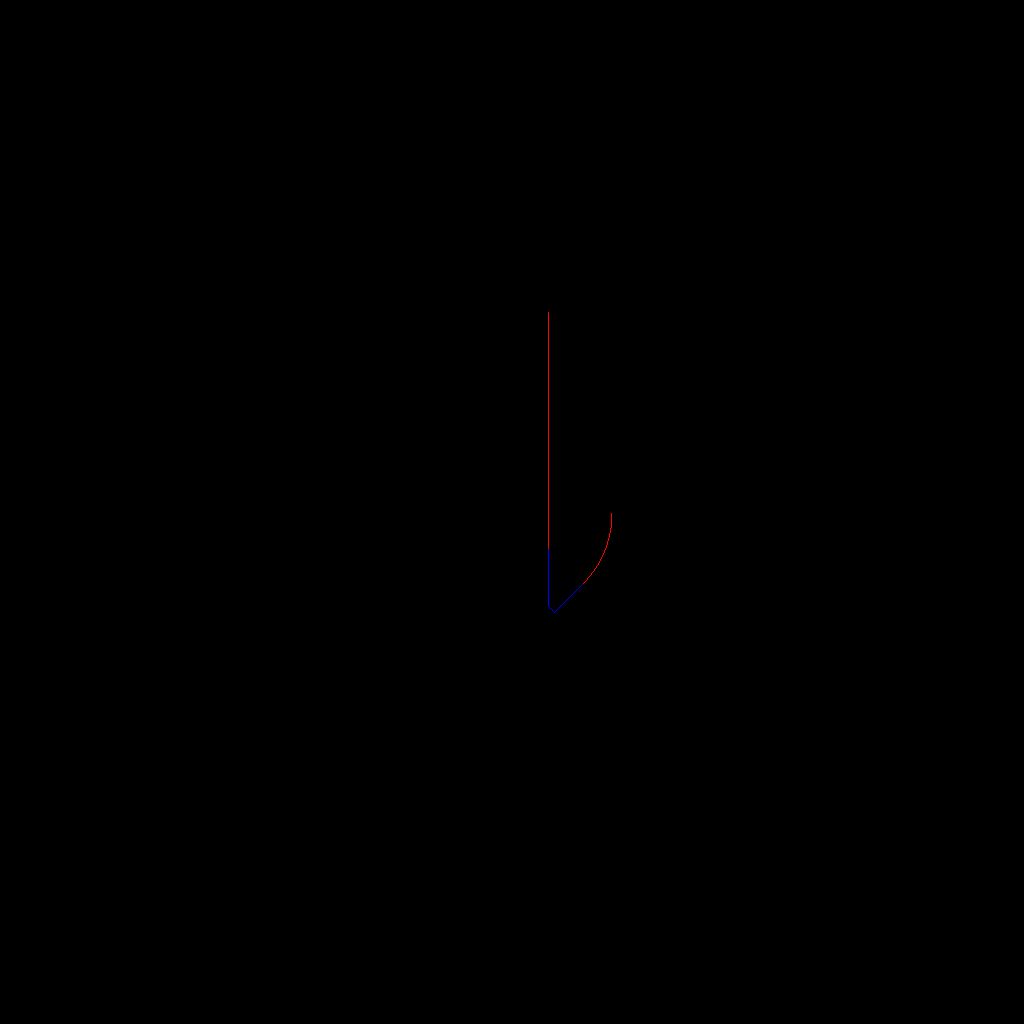
RunFile(sys.argv[2])

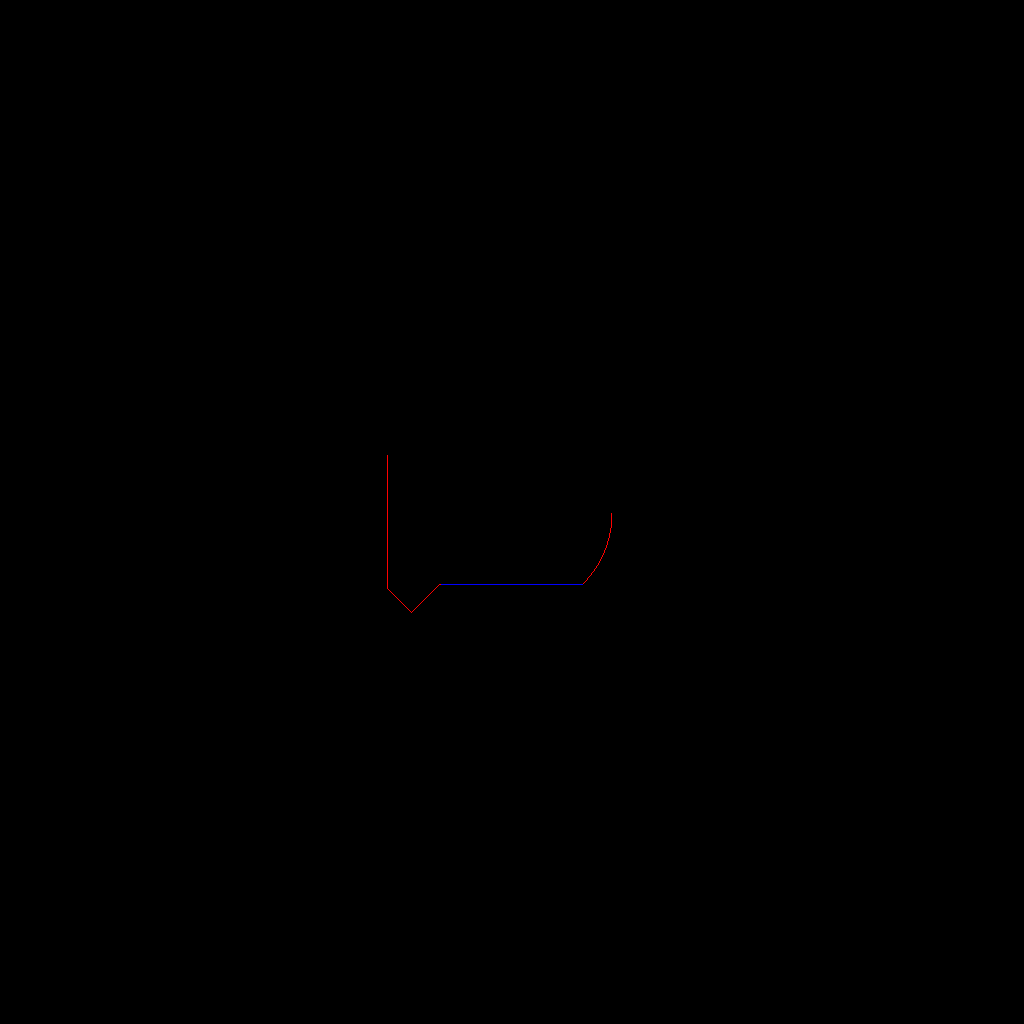
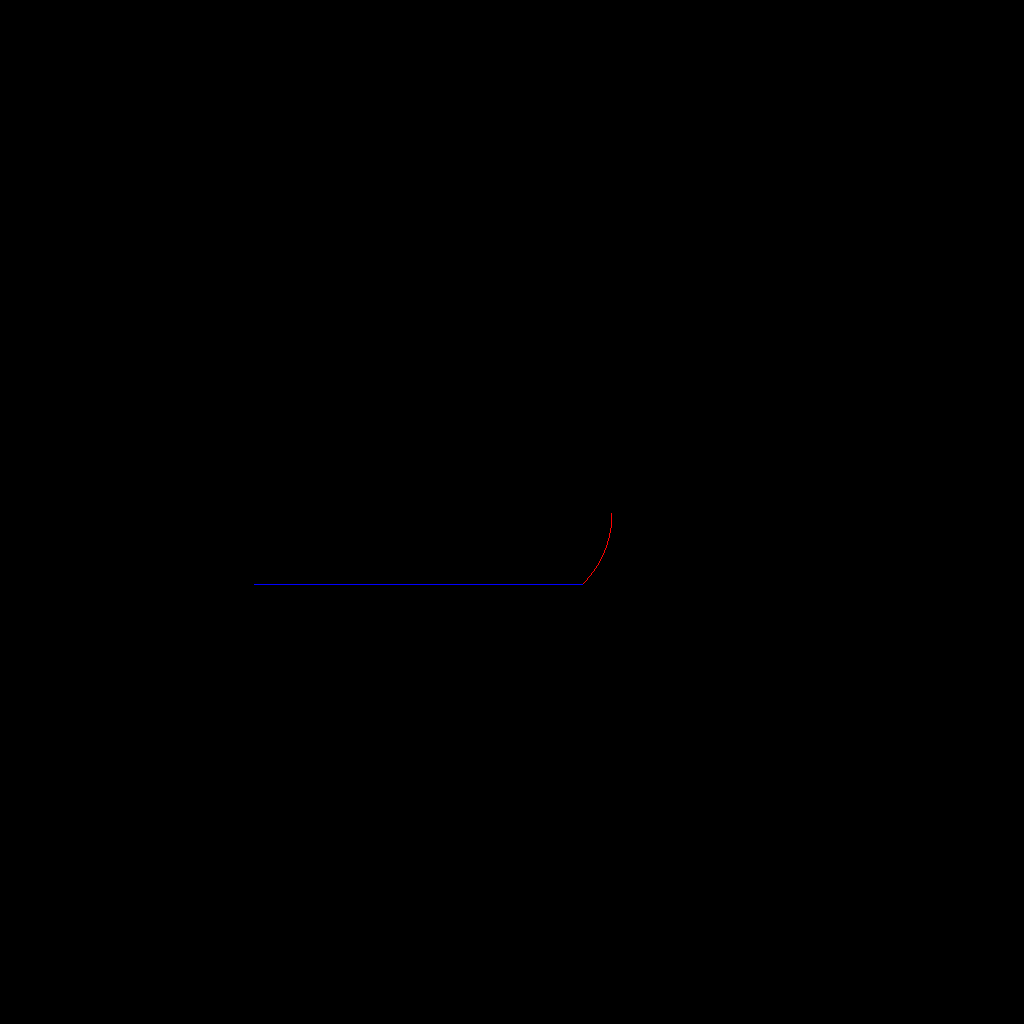
bus.write\_byte\_data(device,port1,0x00)

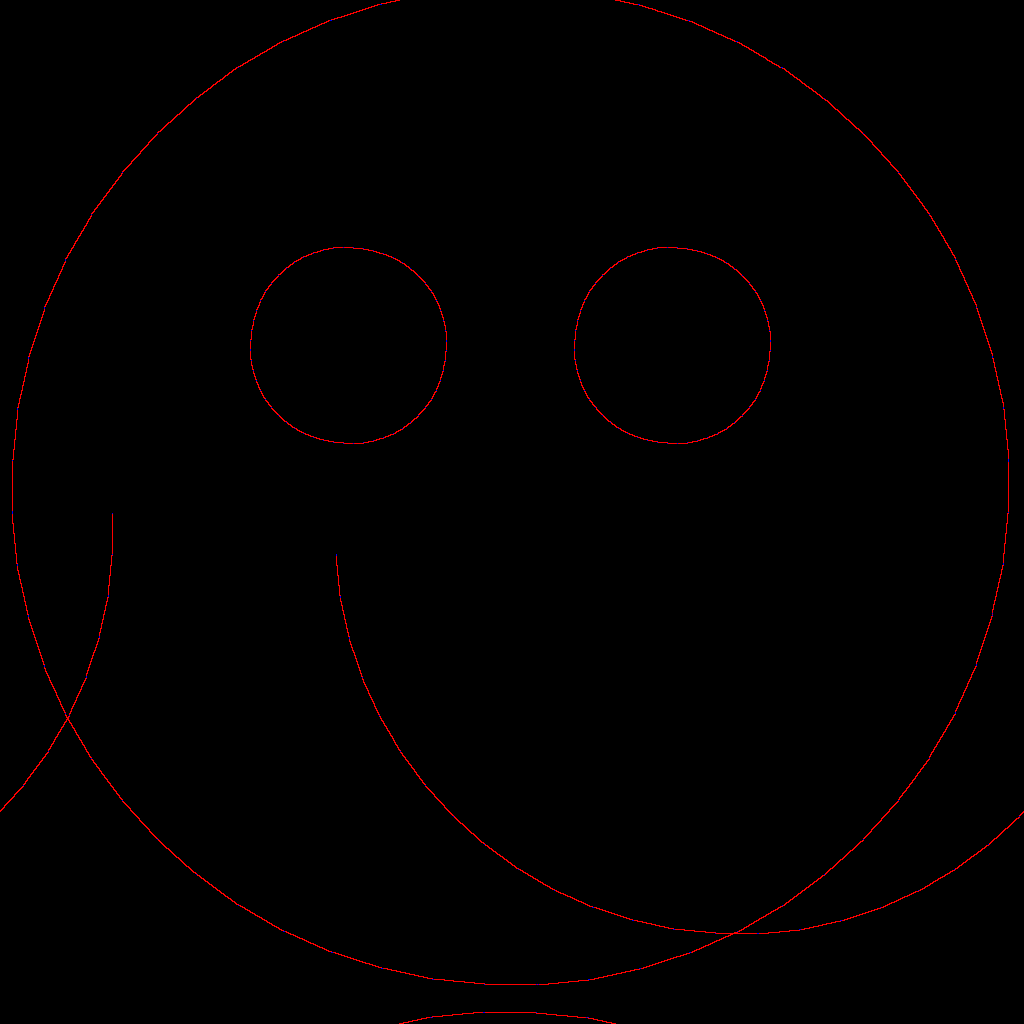
print("script ended")

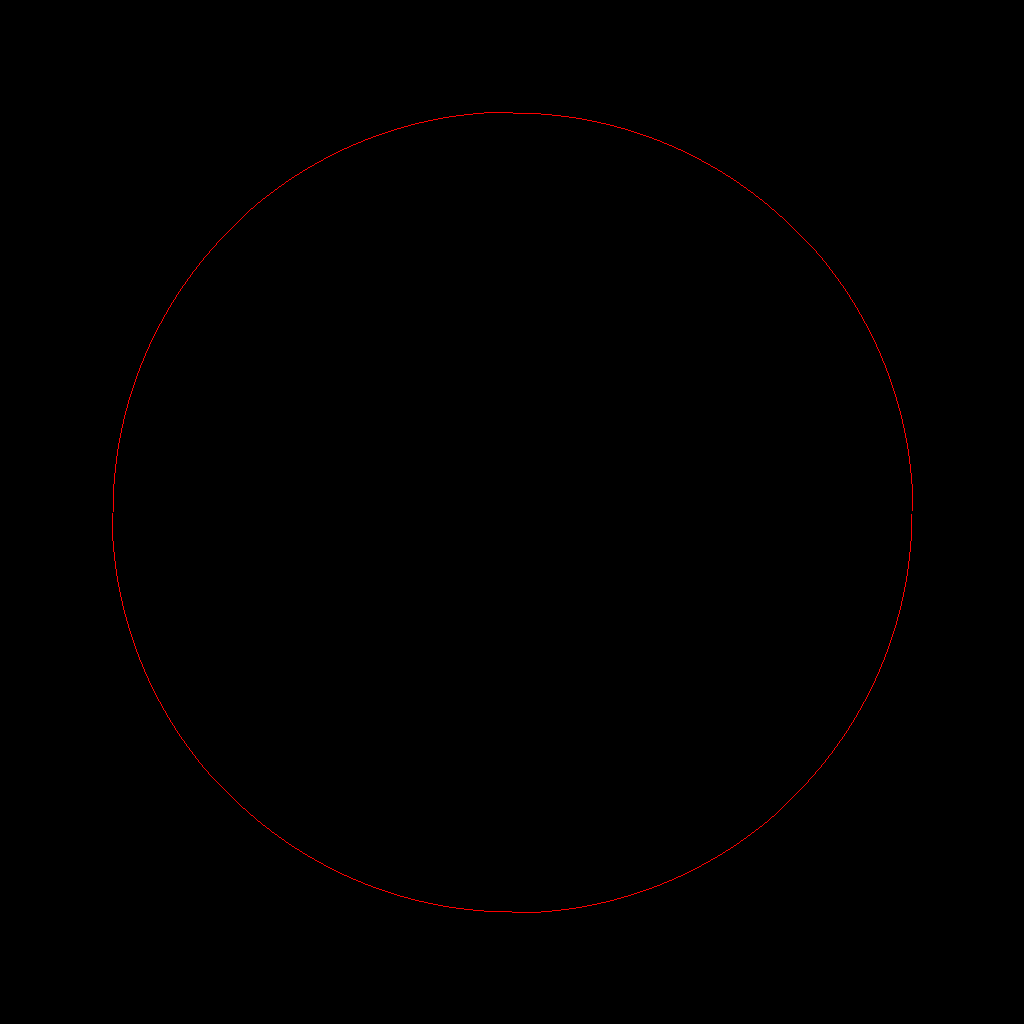
# Appendix 4 – Printer Simulation Test Images



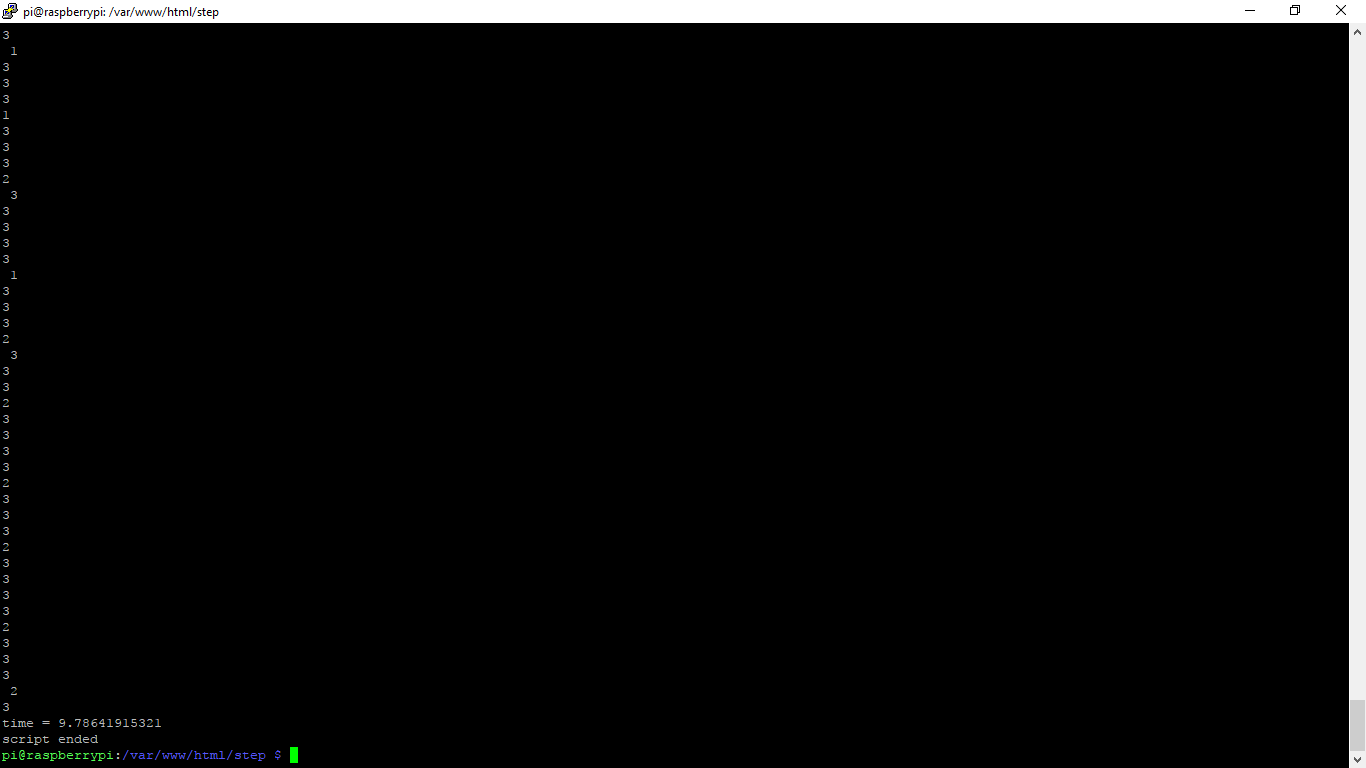








# Appendix 5 – Debugging Via Print To Screen



# Appendix 6 – Robot Arm Test Data

## Data from CoordsToDegrees()

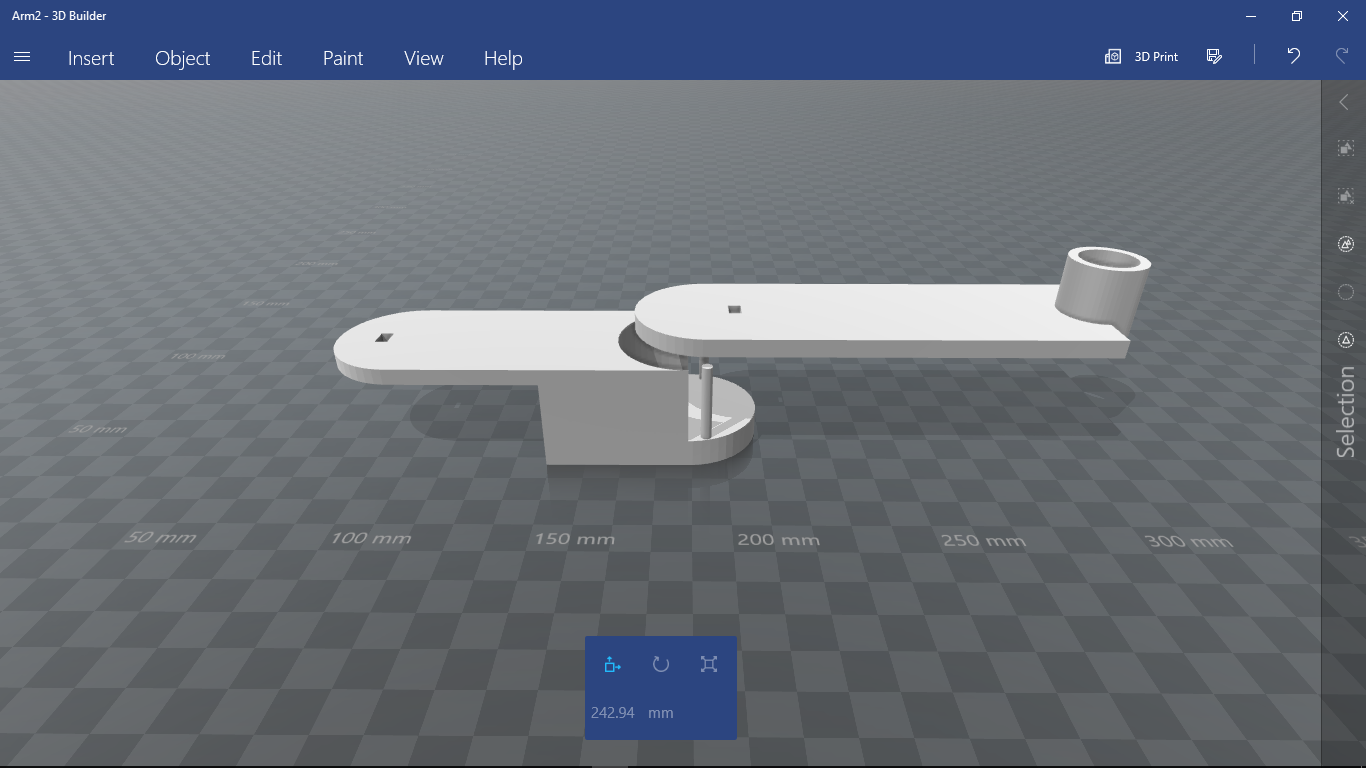
1.64940087509 -0.679353769332, 1.56988046637 -0.605918995382, 1.49414199505 -0.592988400539, 1.52943676858 -0.575557604108, 1.51020860727 -0.560263620342, 1.43963614482 -0.550869882192, 1.47504490776 -0.531145452966, 1.4582065601 -0.516256269197, 1.3919946849 -0.510044469289, 1.42724446241 -0.487578323513, 1.46472183811 -0.531466634209, 1.34957017327 -0.46966583734, 1.38448394465 -0.444000798833, 1.32423098667 -0.442541703097, 1.35880789613 -0.414464255531, 1.34628675403 -0.398798015574, 1.28904017974 -0.399950623893, 1.32295663339 -0.367707449529, 1.31149404384 -0.351012584333, 1.25693024981 -0.354633370668, 1.29002175447 -0.317514972304, 1.32985566903 -0.36188978953, 1.2271643805 -0.305711213013, 1.2592957403 -0.262868073251, 1.2492817292 -0.242620779321, 1.19933782037 -0.250766996801, 1.23027110762 -0.201109890007, 1.18167929503 -0.210596177806, 1.21170286824 -0.155724723639, 1.20222533603 -0.130407318773, 1.15555092227 -0.141711627821, 1.23451823781 -0.145114952019, 1.17459358608 -0.04839281322, 1.12967346101 -0.0611749151196, 1.15612934576 0.0142558906704, 1.14639614475 0.0497911006254, 1.10309358543 0.0360816235245, 1.12709345236 0.125464385409, 1.08476541786 0.111565204407, 1.1068209804 0.212013869431, 1.09577557886 0.261685703532, 1.05476402426 0.248430686516, 1.12591009775 0.296813515986, 1.06093435695 0.431145917914, 1.02095128135 0.420161165364, 1.03470862589 0.566894902852, 1.01979746858 0.645647078608, 0.980463798977 0.639248913218, 0.987973420521 0.818880523155, 0.969524006876 0.919591982842, 0.930228050411 0.920911687923, 0.929335202169 1.14204681211, 0.889569147367 1.1505053455, 0.938334683288 1.34197905511, 0.853548739012 1.55894741743, 0.811989953018 1.58115995386, 0.790032025903 1.89166759794, 0.752069081812 2.07986090191, 0.707193366182 2.11528811835, 0.666744511935 2.48172099705, 0.618808137767 2.52102406282, 0.569441339615 2.73062135062, 0.506456144612 3.13312396326, 0.497457864285 3.36635846602, 0.38572007176 3.37584907461, 0.297757371621 3.76196344248, 0.241996004899 3.7277925585, 0.139558304482 4.0593562419, 0.0548614750543 4.15190484635, 0.00365701638912 4.01742343853, -0.112470548115 4.21432909869, -0.196021341087 4.17739905605, -0.0404629771774 0.451018705597

## Data from DegreesToSteps()

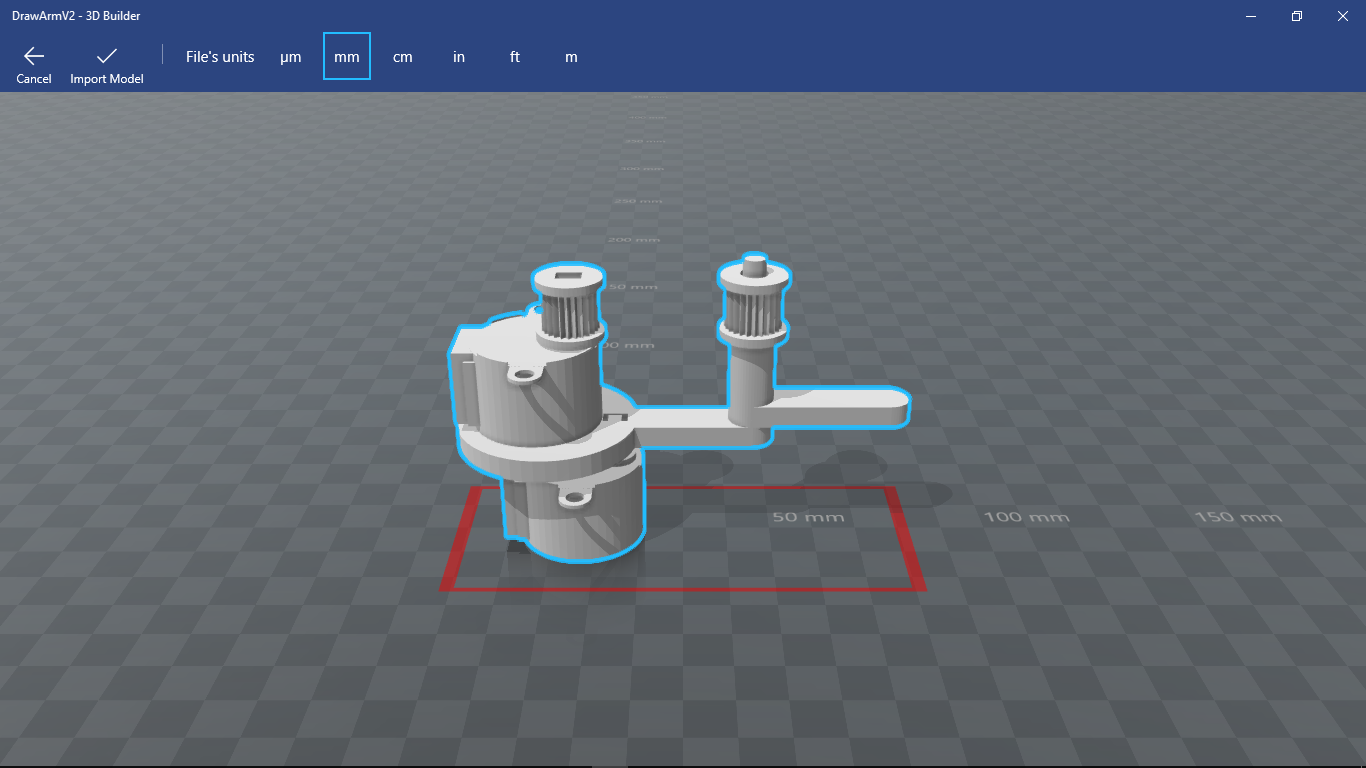
0x60 0xc9 0x90 0x3c 0x60 0xc6 0x90 0x33 0x60 0xc0 0x99 0x30 0x6c 0xc0 0x96 0x30 0x63 0xc0 0x99 0x30 0x6c 0xc0 0x96 0x30 0x63 0xc0 0x99 0x30 0x6c 0xc0 0x90 0x36 0x60 0xc3 0x90 0x39 0x60 0xcc 0x90 0x36 0x60 0xc3 0x90 0x30 0x69 0xc0 0x9c 0x30 0x66 0xc0 0x93 0x30 0x69 0xc0 0x9c 0x30 0x66 0xc0 0x93 0x30 0x69 0xc0 0x90 0x3c 0x60 0xc0 0x96 0x30 0x63 0xc0 0x99 0x30 0x6c 0xc0 0x96 0x30 0x63 0xc0 0x99 0x30 0x60 0xcc 0x90 0x36 0x60 0xc3 0x90 0x39 0x60 0xcc 0x90 0x36 0x60 0xc3 0x90 0x30 0x69 0xc0 0x9c 0x30 0x66 0xc0 0x93 0x30 0x60 0xc9 0x90 0x3c 0x60 0xc6 0x90 0x33 0x60 0xc0 0x99 0x30 0x60 0xcc 0x90 0x36 0x60 0xc3 0x90 0x39 0x60 0xc0 0x9c 0x30 0x66 0xc0 0x93 0x30 0x69 0xc0 0x9c 0x30 0x60 0xc6 0x90 0x33 0x60 0xc9 0x90 0x3c 0x60 0xc0 0x96 0x30 0x60 0xc3 0x90 0x30 0x69 0xc0 0x9c 0x36 0x60 0xc3 0x90 0x30 0x69 0xc0 0x90 0x3c 0x60 0xc0 0x96 0x30 0x63 0xc0 0x99 0x30 0x6c 0xc0 0x96 0x30 0x60 0xc3 0x90 0x39 0x60 0xc0 0x9c 0x30 0x60 0xc6 0x90 0x30 0x63 0xc0 0x99 0x30 0x6c 0xc0 0x96 0x30 0x63 0xc0 0x99 0x30 0x60 0xcc 0x90 0x30 0x66 0xc0 0x93 0x30 0x60 0xc9 0x90 0x30 0x6c 0xc0 0x96 0x30 0x60 0xc3 0x90 0x30 0x69 0xc0 0x9c 0x30 0x60 0xc6 0x90 0x30 0x63 0xc0 0x99 0x30 0x60 0xcc 0x90 0x30 0x66 0xc0 0x93 0x30 0x60 0xc9 0x90 0x3c 0x60 0xc0 0x90 0x30 0x60 0xc6 0x90 0x33 0x60 0xc0 0x99 0x30 0x6c 0xc0 0x90 0x36 0x60 0xc3 0x90 0x30 0x69 0xc0 0x90 0x30 0x60 0xcc 0x90 0x30 0x66 0xc0 0x93 0x30 0x60 0xc0 0x90 0x39 0x60 0xc0 0x9c 0x30 0x66 0xc0 0x90 0x30 0x60 0xc3 0x90 0x30 0x69 0xc0 0x90 0x30 0x60 0xcc 0x90 0x36 0x60 0xc0 0x90 0x30 0x63 0xc0 0x90 0x30 0x60 0xc9 0x90 0x3c 0x60 0xc0 0x90 0x30 0x66 0xc0 0x90 0x30 0x60 0xc3 0x90 0x30 0x60 0xc0 0x90 0x30 0x69 0xc0 0x90 0x30 0x60 0xc0 0x90 0x30 0x6c 0xc0 0x90 0x30 0x60 0xc0 0x90 0x36 0x60 0xc0 0x90 0x30 0x60 0xc0 0x90 0x30 0x60 0xc0 0x90 0x30 0x60 0xc0 0x90 0x30 0x60 0xc0 0x90 0x30 0x60 0xc0 0x90 0x30 0x60 0xc0 0x90 0x30 0x60 0xc0 0x90 0x30 0x60 0xc0 0x90 0x30 0x6c 0xc0 0x90 0x30 0x69 0xc0 0x90 0x30 0x60 0xc6 0x90 0x36 0x60 0xcc 0x90 0x39 0x60 0xc6 0x90 0x36 0x60 0xcc 0x90 0x30 0x69 0xc0 0x96 0x30 0x66 0xcc 0x99 0x30 0x66 0xc6 0x9c 0x39 0x66 0xc6 0x9c 0x39 0x66 0xc6 0x9c 0x39 0x66 0x6 0xcc 0x99 0x36 0x6 0x6c 0x9 0xc6 0x6 0x9c 0x39 0x6 0x66 0xcc 0x9 0x96 0x6 0x3c 0x9 0x66 0x6 0xcc 0x9 0x6 0x96 0xc 0x39 0x6 0x6 0xc 0x69 0x6 0x6 0xcc 0x9 0x96 0x6 0xc 0x39 0x6 0x66 0xc 0x9 0x6 0xc6 0xc 0x9 0x6 0x6 0x9c 0x9 0x6 0x6 0xc 0x39 0x6 0x6 0x6c 0x9 0x6 0x6 0xcc 0x9 0x6 0x6 0xc 0x9 0x96 0x6 0xc 0x9 0x6 0x36 0xc 0x9 0x6 0x66 0xc 0x9 0x6 0x6 0xcc 0x9 0x6 0x6 0xc 0x9 0x6 0x96 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x36 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x66 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0xc9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x96 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x36 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x69 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x9 0x6 0x6 0xc 0x39 0x6 0x6 0xc 0x9 0x6 0x6

# Appendix 7 – 3D models

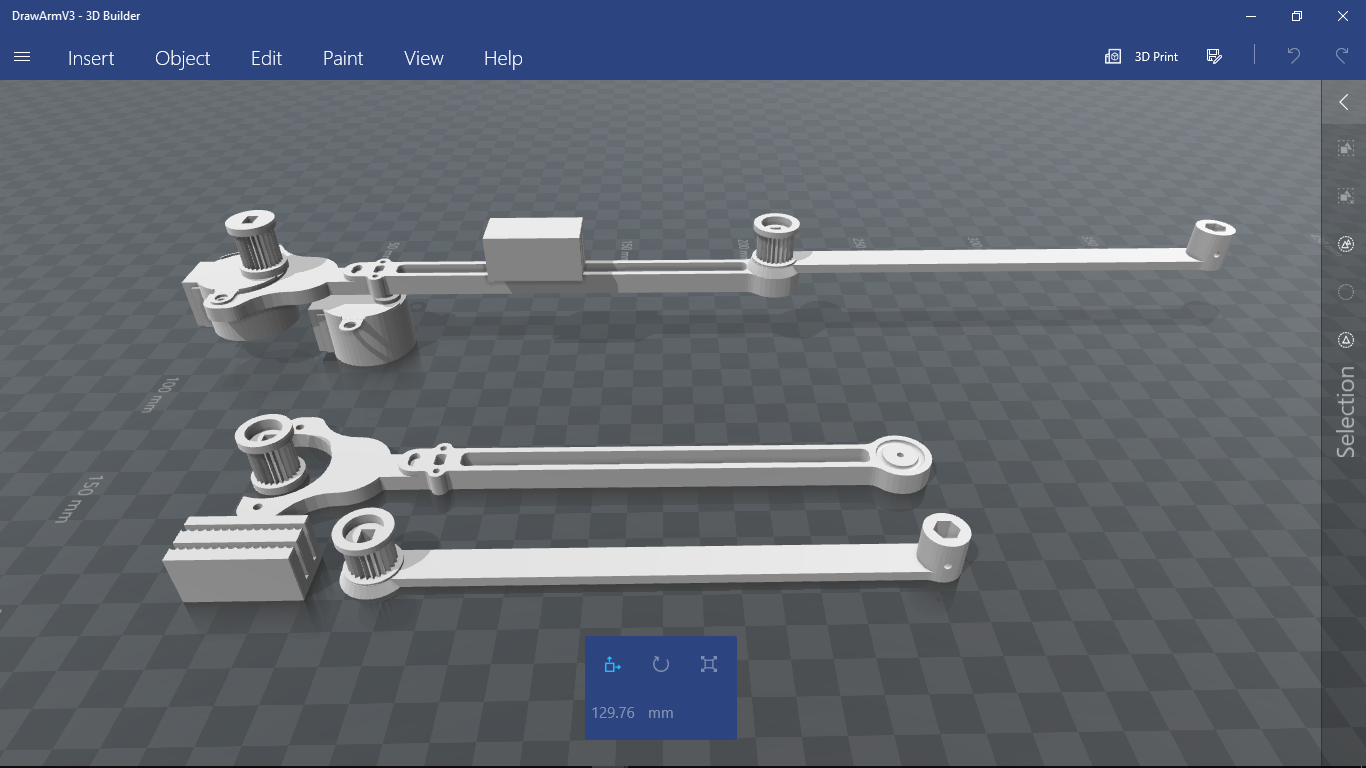
## DrawArmV1.stl



## DrawArmV2.stl



## DrawArmV3.stl



# Appendix 8 – Progress Log

## 15/2/17 SCRABEL

Present – Me, Dan H, Dan M, Sabina, Saif, Reza

Summary:

* Introduction to SCRABEL and presentation on current projects available.
* Discussion on each member’s individual strengths and areas of interest.
* Tutorial on raspberry pi setup.

## 20/2/17 Canterbury Campus, Makerspace

Present – Me, Dan H, Dan M, Sabina, Reza

Summary:

* Familiarised ourselves with the 3D printer that Reza bought, got as far as we could before we realised we needed more tools.

## 22/2/17 SCRABEL

Present – Me, Dan H, Dan M, Sabina, Saif

Summary:

* Setup Raspberry Pi, installed i2c, apache, python, etc.

Progress:

* Raspberry Pi fully set up

## 24/2/17 Canterbury Campus, Makerspace

Present – Me, Dan H, Dan M

* Continued building the 3D printer, built the frame, connected motor and belts.

## 1/3/17 SCRABEL

Present – Me, Dan M, Sabina, Saif

Summary:

* Started programming, decided on to code in python because it is easy to understand and standard for robotics.
* Learned how to control motors by sending byte data through i2c ports.

Progress:

* Made a test file in bash that ran the motors on a simple loop.

## 4/3/17 Google Hangouts

Present – Me, Dan H, Dan M, Sabina, Saif

Summary:

* Saif explained how his extruder for his stem cell printer would work and the associated design obstacles.
* Saif asked me to design a robot that required multiple motors to move, we decided on a pan and tilt system that could be used to aim a camera.

## 8/3/17 SCRABEL

Present – Me, Dan M, Sabina, Saif

Summary:

* We moved the 3D printer to SCRABEL and continued building it there where we had more tools and Saif’s expertise.
* Decided to program motors to be able to control the extruder of a 3D printer. This could be used as a base for further projects.
* Took motors and circuitry home to practice

Progress:

* Mounted extruder to printer frame.

## 15/3/17 SCRABEL

Present – Me, Dan M, Sabina, Saif

Summary:

* Developed method DrawLine which worked out the timing of each step in order to get the cursor to move in the desired direction.
* Developed method DrawCircle which uses the DrawLine method to draw an n-sided polygon which approximates a circle.

Progress:

* Program simulates movements of a motor in x and y axis, moves one step in a direction and waits a certain amount of time to execute next step. Currently have no way to test this.

## 19/3/17 Google Hangouts

Present – Me, Dan M, Sabina, Saif

Summary:

* Saif raised the idea of an interpreter to interpret a text document containing basic instructions into commands to draw a shape. This task was given to Dan Mattson.

## 22/3/17 SCRABEL

Present – Me, Dan M, Sabina, Saif

Summary:

* Changed how DrawLines works. It now uses multithreading to run multiple instances of the Motor method at different speeds do adjust the angle on the line.
* Experimented with NumPy and PIL modules so that I could use draw to an image and simulate motor movements.

Progress:

* Can draw a smiley face to an image using instructions from text document (see appendix).

## 29/3/17 SCRABEL

Present – Me, Dan M, Sabina, Saif

Summary:

* Saif suggested we take the project in a more specific direction and asked me to think of a project. I decided on a robotic arm that can draw with a pencil.

## 5/4/17 SCRABEL

Present – Me, Dan M, Sabina, Saif

Summary:

* Designed first prototype for robot arm using OpenSCAD 3D design software.
* Changed DrawCircle Method to draw a true circle using x2+y2=r2 equation rather than a polygon.

Progress:

* DrawCircle method now draws a circle, the accuracy of which is dependent on the accuracy of the motor controlling it.

## 12/4/17 SCRABEL

Present – Me, Dan M, Sabina, Saif, Reza, Muna

Summary:

* Muna and Reza visited to see how Daniel and I were getting on.
* Designed version 2 of robot arm
* Created method Line which takes a destination coordinate and creates a list of coordinates between current position and destination which are 5 units apart.
* Created method CoordsToDegrees which takes a destination and calculates the required angle the motors need to turn to bring the pen to the destination.

Progress:

* New arm design now uses a belt to control the elbow of the arm, keeping the motors at the same end.
* Program can calculate new position using an angular coordinate system and calculate the required rotations to move the arms to a new postion.

## 19/4/17 SCRABEL

Present – Me, Dan M, Sabina, Saif

Summary:

* Designed final version of robot arm.
* Created method DegreesToSteps which calculates the number of steps required to turn x degrees and spreads the steps out to adjust the angle of the line.
* Created StepsToData method which takes the direction of each motor and returns the byte data to turn each motor in the required direction.

Progress:

* Arm now has elbow motor attached to the arm so that it moves with it.
* Program now outputs a list of byte data to turn both motors to move the pen to the required destination and draw a line from a to b.

## 26/4/17 SCRABEL

Present – Me, Dan M, Sabina, Saif, Vijay

Summary:

* Vijay visited to see how Sabina was doing with her placement.
* Printed and assembled final design.
* Added methods to calculate steps and write to file and another method to execute the file.
* Tested and debugged

Progress:

* Byte data is now saved to a text file to be executed later.
* Arm does not currently work as intended, this due to a mix up in the code that swapped the motors around.

## 3/5/17 SCRABEL

Present – Me, Dan M, Sabina, Saif

Summary:

* Further testing and debugging.
* Wrapping up projects, ensuring code was commented etc.

Progress:

* Arm still does not draw straight lines but each joint is capable turning a specified number of degrees which shows that the physical design is sound.