UNIVERSITY OF OTTAWA
Faculty of Engineering

GNG1103 Design Project – Deliverable F

Group C2

Adrianna Chouliotis	300225514
Hunter Fleming	300213232
Jim Zhou	300197576
Minjung Gong	300080238
Toby Thai	300201869

Submitted to: Professor David Knox

Submitted on: March 6th, 2022

Wrike Snapshot Link:
https://www.wrike.com/frontend/ganttchart/index.html?snapshotId=xH9ZzWTfshXFNcrx1t6L7nlmGXFxZG9W%7CIE2DSNZVHA2DELSTGIYA

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CODE

All code was completed using Python.

Table 1. Inverse Kinematics Code

```
from numpy import *

#a: lengths of arm pieces in cm
a1=7.4
a2=23
a3=25

#insert desired positioning
px=
py=

#phi is the angle of the end effector with respect to the x-axis
phi=                        #insert phi value
phi=deg2rad(phi)


wx=px-a3*cos(phi)
wy=py-a3*sin(phi)

delta=wx**2+wy**2
c2=(delta-a1**2-a2**2)/(2*a1*a2)
s2=sqrt(1-c2**2)
theta_2=arctan2(s2,c2)

s1=((a1+a2*c2)*wy-a2*s2*wx)/delta
c1=((a1+a2*c2)*wx+a2*s2*wy)/delta
theta_1=arctan2(s1,c1)
theta_3=phi-theta_1-theta_2

#converting angles to degrees
b1=rad2deg(theta_1)
b2=rad2deg(theta_2)
b3=rad2deg(theta_3)

print('theta 1:',b1)
print('theta 2:',b2)
print('theta 3:',b3)
```

One can determine the required angles that need to be connected to the robotic arm through the use of the inverse kinematic code found in Table 1. The only input needed is the desired positioning and the angle of the end effector with respect to the x-axis. With the determined angles, one inputs them into the Arduino code that connects to the motors and allows the motion to proceed. The Arduino coding will be completed during the next deliverable.

Table 2. User Interface Code

```
from tkinter import *
from numpy import *

root=Tk()
frame=Frame(root)
frame.pack()

from numpy import *

def startik():
    #a: lengths of arm pieces in cm
    a1=7.4
    a2=23
    a3=25

    #insert desired positioning
    px=
    py=

    #phi is the angle of the end effector with respect to the x-axis
    phi=                        #insert phi value
    phi=deg2rad(phi)


    wx=px-a3*cos(phi)
    wy=py-a3*sin(phi)

    delta=wx**2+wy**2
    c2=(delta-a1**2-a2**2)/(2*a1*a2)
    s2=sqrt(1-c2**2)
    theta_2=arctan2(s2,c2)

    s1=((a1+a2*c2)*wy-a2*s2*wx)/delta
    c1=((a1+a2*c2)*wx+a2*s2*wy)/delta
    theta_1=arctan2(s1,c1)
    theta_3=phi-theta_1-theta_2

    #converting angles to degrees
    b1=rad2deg(theta_1)
    b2=rad2deg(theta_2)
    b3=rad2deg(theta_3)

    print('theta 1:',b1)
    print('theta 2:',b2)
    print('theta 3:',b3)

#Legend:
    #Verify:confirm position input. If confirm; it goes to the choose end effector menu.
        #If cancel; it goes back to choose orientation
    #Orientpost:input desired postioning by inserting coordinate values
    #Endeffect: choose end effector (scanner, painter or blaster)
    #Scannerend: confirm that the scanner was the end effector that was meant to be chosen,
        #if not cancel brings back to choice menu
    #Paintend: painting end effectorm, confirm or cancel to bring back to choice menu
    #Blasterend: choose between water, sand or return to menu
        #sblaster: sand blaster, confirm or return to menu
        #wblaster: water blaster, confirm or return to menu
    #StartIK:calculates required angles
```

```


def verify():
    frame2=Frame(root,bg="blue")
    frame.pack()
    leftframe=Frame(root,bg="green",bd=3)
    leftframe.pack(side=LEFT)
    rightframe=Frame(root,bg="red",bd=3)
    rightframe.pack(side=RIGHT)
    label=Label(frame,text="Verify Data Input")
    label.pack()
    confirm=Button(leftframe,text="CONFIRM",command=endeffect)
    confirm.pack(padx=3,pady=3)
    cancel=Button(rightframe,text="CANCEL",command=orientpost)
    cancel.pack(padx=3,pady=3)

def orientpost():
    label=Label(frame,text="Insert Desired Postion")
    label.pack()
    entry1=Entry(frame,width=20)
    entry1.insert(0,"x-coordinate")
    entry1.pack(padx=5,pady=5)
    entry2=Entry(frame,width=20)
    entry2.insert(0,"y-coordinate")
    entry2.pack(padx=5,pady=5)
    entry3=Entry(frame,width=20)
    entry3.insert(0,"z-coordinate")
    entry3.pack(padx=5,pady=5)
    xcoord=entry1.get()
    ycoord=entry2.get()
    zcoord=entry3.get()
    button1=Button(text="Enter",command=verify)
    button1.pack()


label=Label(frame,text="Insert Desired Postion")
label.pack()

entry1=Entry(frame,width=20)
entry1.insert(0,"x-coordinate")
entry1.pack(padx=5,pady=5)

entry2=Entry(frame,width=20)
entry2.insert(0,"y-coordinate")
entry2.pack(padx=5,pady=5)

entry3=Entry(frame,width=20)
entry3.insert(0,"z-coordinate")
entry3.pack(padx=5,pady=5)

xcoord=entry1.get()
ycoord=entry2.get()
zcoord=entry3.get()


button1=Button(text="Enter",command=verify)
button1.pack()

```

```python

def endeffect():
    frame=Frame(root)
    frame.pack()
    label=Label(frame,text="Choose End Effector Type:")
    label.pack()

    def scannerend():
        frame=Frame(root,bg="blue")
        frame.pack()
        label=Label(frame,text="Verify End Effector Choice: Scanner?")
        label.pack()
        leftframe=Frame(root,bg="green",bd=3)
        leftframe.pack(side=LEFT)
        rightframe=Frame(root,bg="red",bd=3)
        rightframe.pack(side=RIGHT)
        confirm=Button(leftframe,text="CONFIRM",command=startik)
        confirm.pack()
        cancel=Button(rightframe,text="CANCEL",command=endeffect2)
        cancel.pack()

    def blasterend():
        frame=Frame(root)
        frame.pack()
        label=Label(frame,text="Choose Blaster Type:")
        label.pack()
        RBttn4=Radiobutton(frame,text="Water",command=wblaster)
        RBttn4.pack(padx=5,pady=5)
        RBttn5=Radiobutton(frame,text="Sand",command=sblaster)
        RBttn5.pack(padx=5,pady=5)
        RBttn6=Radiobutton(frame,text="Return to End Effector Menu",command=endeffect2)
        RBttn6.pack(padx=5,pady=5)

    def sblaster():
        frame=Frame(root,bg="blue")
        frame.pack()
        frame=Frame(root,bg="purple")
        frame.pack()
        leftframe=Frame(root,bg="green",bd=3)
        leftframe.pack(side=LEFT)
        rightframe=Frame(root,bg="red",bd=3)
        rightframe.pack(side=RIGHT)
        label=Label(frame,text="Confirm Blaster Choice: Sand?")
        label.pack()
        confirm=Button(leftframe,text="CONFIRM",command=startik)
        confirm.pack()
        cancel=Button(rightframe,text="CANCEL",command=blasterend)
        cancel.pack()

    def wblaster():
        frame=Frame(root,bg="purple")
        frame.pack()
        leftframe=Frame(root,bg="green",bd=3)
        leftframe.pack(side=LEFT)
        rightframe=Frame(root,bg="red",bd=3)
        rightframe.pack(side=RIGHT)
        label=Label(frame,text="Confirm Blaster Choice: Water?")
        label.pack()
```

```
        confirm=Button(leftframe,text="CONFIRM",command=startik)
        confirm.pack()
        cancel=Button(rightframe,text="CANCEL",command=blasterend)
        cancel.pack()

    def paintend():
        frame=Frame(root,bg="pink")
        frame.pack()
        leftframe=Frame(root,bg="green",bd=3)
        leftframe.pack(side=LEFT)
        rightframe=Frame(root,bg="red",bd=3)
        rightframe.pack(side=RIGHT)
        label=Label(frame,text="Verify End Effector Choice: Painter?")
        label.pack()
        confirm=Button(leftframe,text="CONFIRM",command=startik)
        confirm.pack()
        cancel=Button(rightframe,text="CANCEL",command=endeffect2)
        cancel.pack()

    def endeffect2():
        label=Label(frame,text="Choose End Effector Type:")
        label.pack()
        RBttn1=Radiobutton(frame,text="Scanner",command=scannerend)
        RBttn1.pack(padx=5,pady=5)
        RBttn2=Radiobutton(frame,text="Blaster",command=blasterend)
        RBttn2.pack(padx=5,pady=5)
        RBttn3=Radiobutton(frame,text="Painter",command=paintend)
        RBttn3.pack(padx=5,pady=5)

    RBttn1=Radiobutton(frame,text="Scanner",command=scannerend)
    RBttn1.pack(padx=5,pady=5)
    RBttn2=Radiobutton(frame,text="Blaster",command=blasterend)
    RBttn2.pack(padx=5,pady=5)
    RBttn3=Radiobutton(frame,text="Painter",command=paintend)
    RBttn3.pack(padx=5,pady=5)

root.mainloop()
```

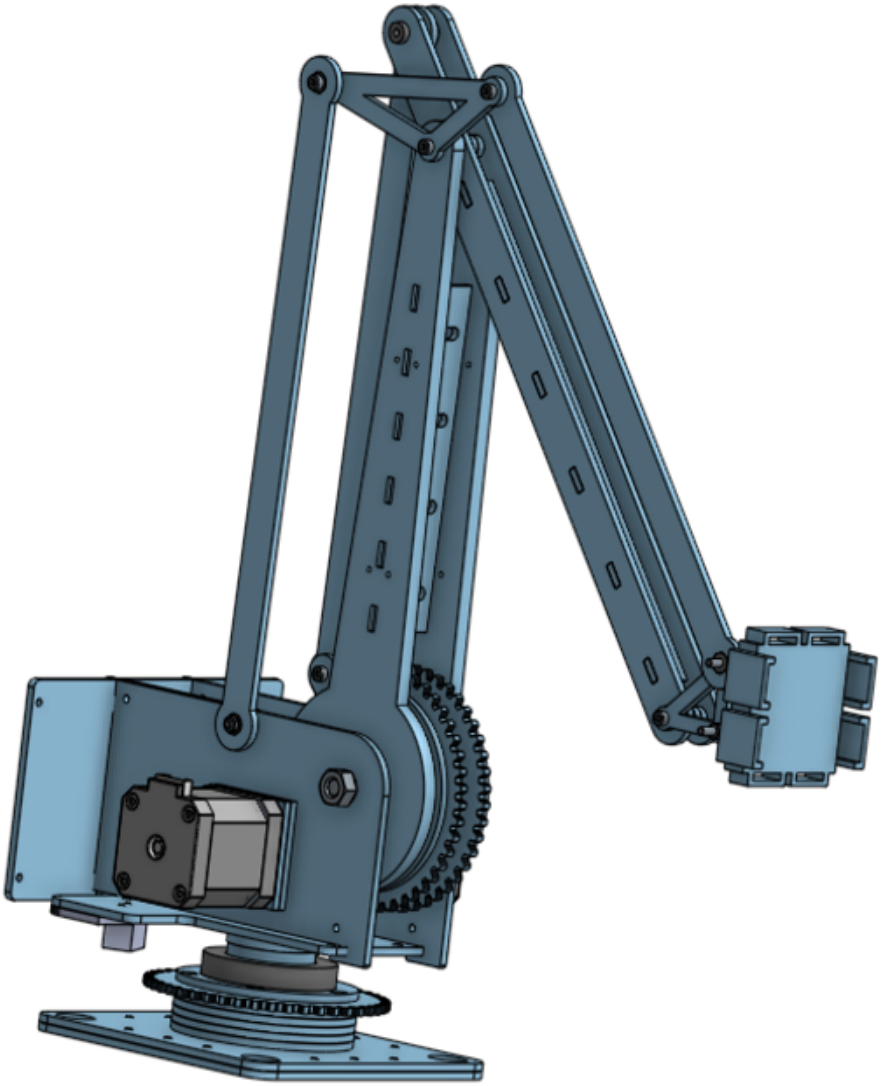

Figure 1. Arm and End Effector Connection

TEST RUN DATA

Assembly

To test the assembly of the mechanical components and build a physical arm with an end effector. Through the CAD model, it was confirmed that the arm can move flexibly within a range of 1 meter. The end effector will also be able to hold the blaster or painter using a Velcro band.

CALCULATIONS

Approximate Volume Calculation: 0.974 m^2 (Onshape)

Maximum Reach Calculations:

$$\theta_2 = \cos^{-1}\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}\right) = \cos^{-1}\left(\frac{1^2 + 1^2 - (0.3m)^2 - (0.3m)^2}{2(0.3m)^2}\right) = N/A$$

⇒ No need to calculate θ1, as a 1m reach is already impossible.

Estimation with 45-degree value for each arm

$$A1 = \begin{matrix} 0.707 & -0.707 & 0.707 & 0.212 \\ 0.707 & 0.707 & -0.707 & 0.212 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$A2 = \begin{matrix} 0.707 & -0.707 & 0.707 & 0.212 \\ 0.707 & 0.707 & -0.707 & 0.212 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

UPDATED BILL OF MATERIALS

Table 3. Updated Bill of Materials					
Name	**Description**	**Dimensions**	**Quantity**	**Unit Cost**	**Total Cost**
Software					
Python	End effector needs open-source Python to work. Python can be download for free from the Python site and is useful and easy for coding.	-	1	$0	$0
Hardware					
Laptop	To code, prototype, and test with Python, we need a laptop. But it is free because everyone on the team has their own laptop already.	-	1	$0	$0
Arduino Uno	The Arduino uno is an open-source microcontroller board. It is required to operate the robot arm using Python.	-	1	$0.00	$0.00
ABS Filament	ABS filament is 3d printer plastic. It is easy to process, has high impact resistance and good heat resistance. It is often used as a substitute for metal in industrial products such as automobile parts and electronic device parts because of its very high strength.	-	1	$0	$0
Screw	Screws are used to connect the arm to the end effector.	M5-0.8×45mm	25	$0.00	$0.00
Wire Set	Wires are used to connect the Arduino to the end effector, supply and operate power.	-	120pcs	$0.00	$0.00
Velcro Industrial Roll	Velcro band is used in this project to strap the items together by attaching them to the End effector. This is suitable for strapping items as it has a good adhesion if only a certain amount of contact area is secured.	Length: 5m Width: 3cm	1	$20.99	$20.99
Servo Motor	Servo motor is a motor that can move exactly as much as inputted by the control and measurement circuit when movement is specified. This is where the movement of the end effector helps.	13.69×9.8×3.61 cm 50g	4	$0.00	$0.00
			TOTAL	$20.99	
			TAX (13%)	$2.73	
			TOTAL COST	$23.72	

UPDATED DESIGN SPECIFICATIONS

Table 4. Updated Target Design Specifications

	Design Specifications	Relation	Value	Units	Verification	Prototype 1: Specs Met?
	Functional Requirements					
1	3D Printed	-	Yes	>	Design	N/A
2	Coded off Python	-	Yes	-	Design	Yes
3	Open Source	-	Yes	-	Uploading	Yes
4	Ground Mounted	>	Yes	-	Design	Yes
	Constraints					
1	Cost per Arm	≤	50	$ CAD	Cost Calculations	Yes
2	Temperature to Withstand	Range	-30 - 60	∘C	Testing/Analysis	N/A
3	Degrees of Freedom	=	3	-	Design	Yes
4	Weight	≤	9	Kg	Weighing	Yes
5	Size (L x H x W)	≤	1	m^2	Calculation	Yes
6	Payload	≈	1	Kg	Testing	N/A
7	Pressure to Withstand	≤	8	Bar	Testing/Analysis	
8	Range of Motion	≥	±0.5	m	Testing/Calculations	
9	Time to Paint 4 ft^2	≲	4	Hours	Testing	N/A
	Non-Functional Requirements					
1	Ease of Use	≤	6	Hours	Training Testing	Yes
2	Pinch Points	≥	4	-	Design	Yes
3	UV/Corrosion Resistance	-	Yes	-	Testing	N/A
4	Life Span of the Product	≥	1	Year	Use	N/A

REFERENCES

Open-Source Code Used

https://github.com/aakieu/3-dof-planar/blob/master/InverseKinematics.py

Inverse Kinematics

https://os.mbed.com/users/ms523/notebook/3d-inverse-kinematics/
http://motion.cs.illinois.edu/software/klampt/latest/pyklampt_docs/Manual-IK.html

Code Dictionary

https://realpython.com/python-gui-tkinter/
https://coderslegacy.com/python/python-gui/python-tkinter-radio-button/