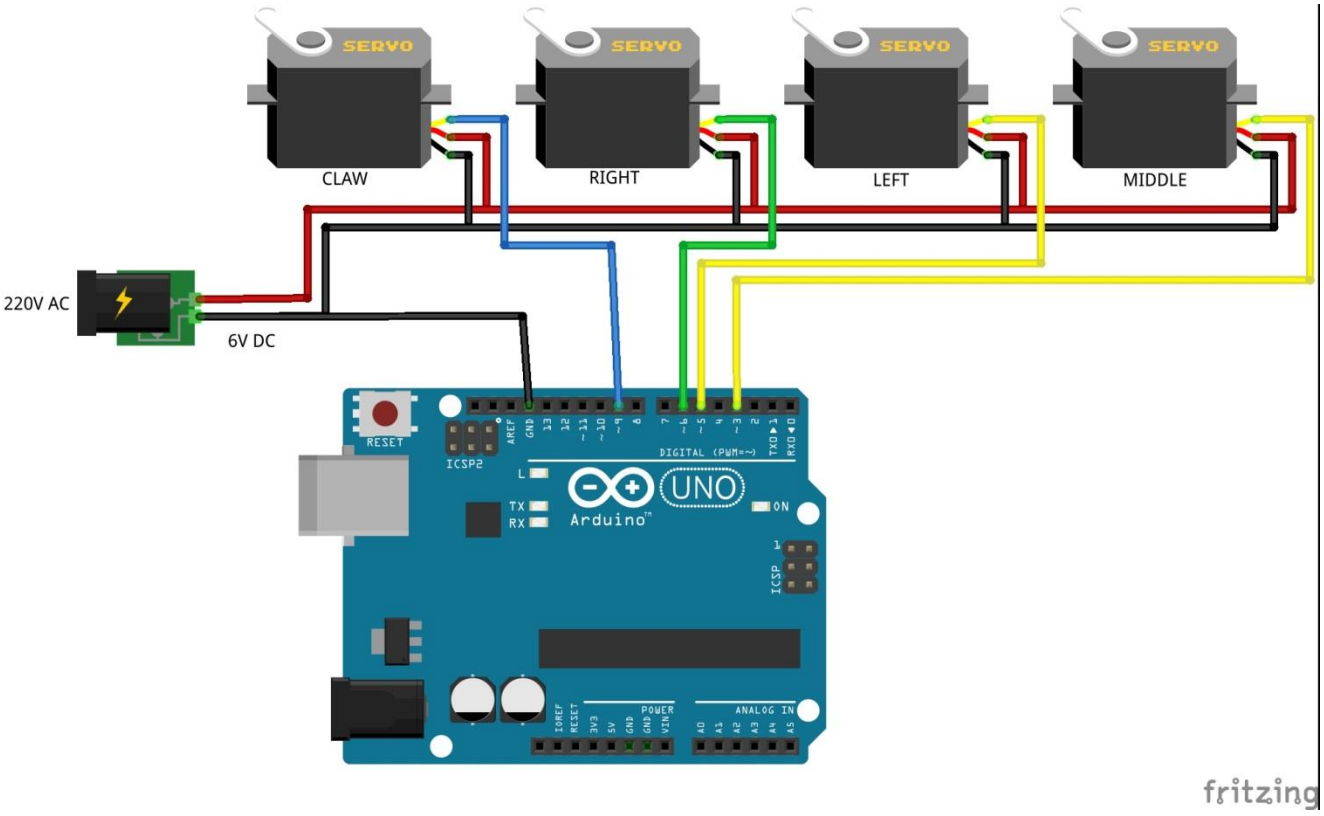
Pilotage robot MeArm avec interface Python
Via liaison série USB

1. Câblage :

J'ai utilisé un petit robot MeArm de chez Mime Industries (https://mime.co.uk), disponible à la vente sur la plupart des sites de robotique amateur ou dont le .DXF pour découpe laser est dispo gratuitement sur http://www.thingiverse.com/thing:993759/#files et un microcontrôleur Arduino Genuino/Uno. Une alimentation externe est utilisée pour les servomoteurs du robot en 6V. Attention, prévoir une puissance suffisante sur cette seconde alimentation : le traditionnel boitier de piles est un peu juste, il vaut mieux lui préférer un bloc prise 200VAC / 6VCC.

Attention, au moment des branchements, il vaut mieux tout brancher sauf l'alim des servomoteurs, téléverser le code et ensuite seulement alimenter les servomoteurs.

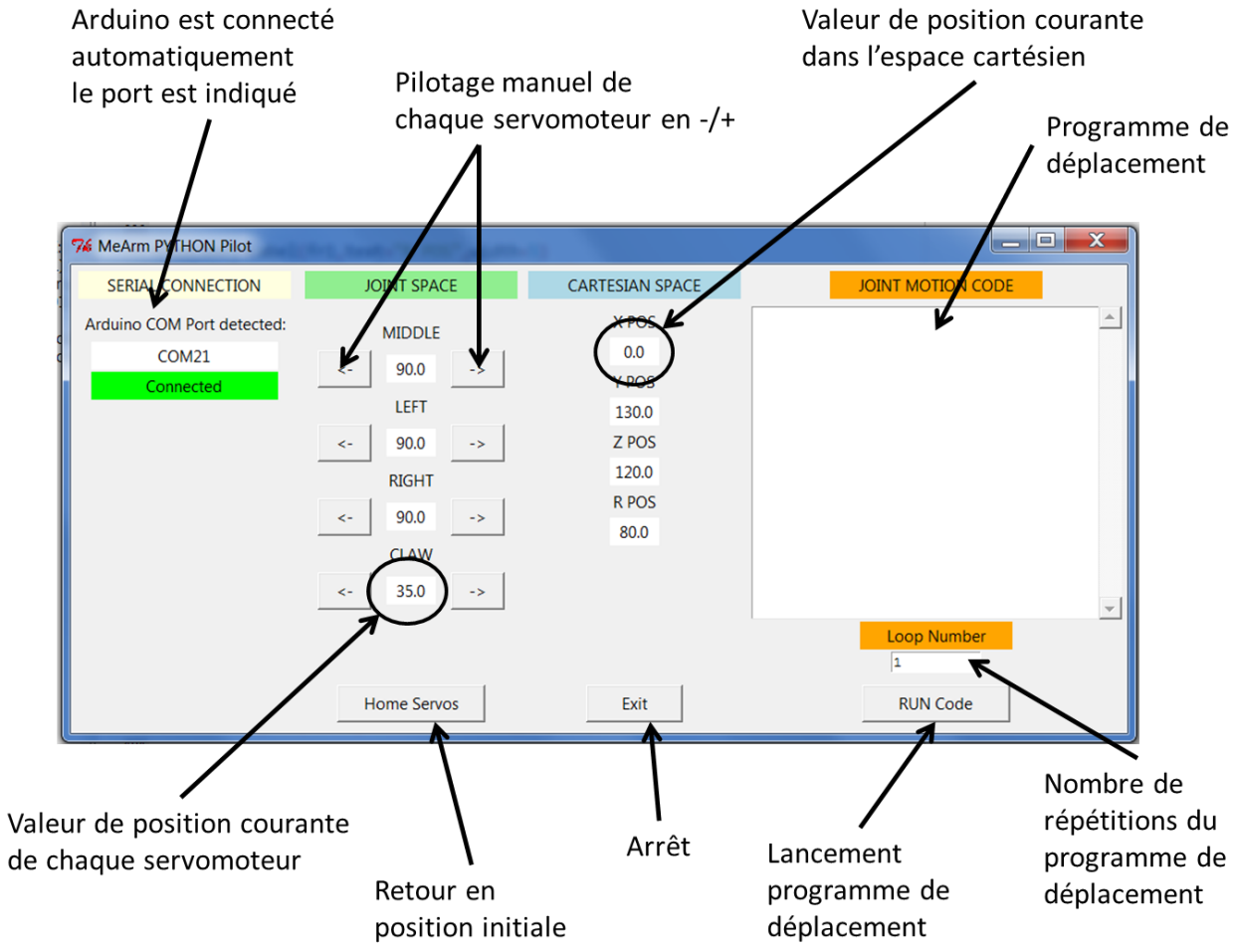


Pin système	Signal	Description	Branchement Pin Arduino
GND	Masse	Commune avec l'alimentation 6VCC	GND
Middle	Position	Position servo en degrés	3
Left	Position	Position servo en degrés	5
Right	Position	Position servo en degrés	6
Claw	Position	Position servo en degrés	9

2. Interface utilisateur

Forcément, il faut une petite explication sur le fonctionnement de l'interface utilisateur, un genre de notice quoi… Même si tout cela reste simple !

La partie « SERIAL CONNECTION » est purement informative.
« JOINT SPACE » permet à la fois de piloter chaque articulation (en anglais « joint »), ou servomoteur, en réduisant (<-) ou augmentant (->) la valeur de l'angle. Le nombre donné entre les touches est la valeur courante de l'angle de l'articulation considérée.
« Home Servos » permet le retour physique du robot à sa position initiale.
« CARTESIAN SPACE » indique la position courante du centre de la pince du robot dans l'espace cartésien XYZ, l'origine étant le centre de l'axe « Middle » et la face inférieure du socle du robot.
« JOINT MOTION CONTROL » mérite un peu plus d'explications : cette fenêtre permet de saisir une petite séquence de déplacements dans l'espace articulaire (positions angulaires des servomoteurs). Chaque ligne est une position, la syntaxe est « GO:middle,left,right,claw » soit par exemple : GO:110,80,65,40.
Pour exécuter de petit programme, il suffit de renseigner le nombre de boucles (répétitions) que l'on souhaite effectuer dans la fenêtre « Loop Number » et de cliquer sur « RUN Code ».



3. Code :

Deux séries de codes Arduino : un pour la mise à zéro des servomoteurs indispensable avant le montage même du robot MeArm, l'autre pour le pilotage du robot monté ensuite. Le code Python 2.7 inclut la partie interface graphique, la détection du port de communication de l'Arduino est automatique, il faut bien penser à connecter dans l'ordre :

- 1. L'Arduino au PC
- 2. L'alimentation +6V des servomoteurs au 220V
- 3. Lancer ensuite le programme de pilotage Python

Code de mise à zéro initiale des servomoteurs

```
#include <Servo.h>
Servo middle, left, right, claw ;

void setup() {
 Serial.begin(9600) ;
 middle.attach(3);
 left.attach(5);
 right.attach(6);
 claw.attach(9) ;
}

void loop() {
middle.write(90);
left.write(90);
right.write(90);
claw.write(35);
delay(300);
}
```

Code de pilotage complet

```
/*****************************************************************
  MEARM ROBOT PILOT BY PYTHON 2.7 INTERFACE
*****************************************************************
  Written by Nicolas Blanchard - (CC BY-NC-SA 4.0)
*****************************************************************/

#include <Servo.h>  /*ajout Servo Library */

Servo middle, left, right, claw ;  /*creation des classes servo*/
char incomingByte = 0;
int middlePos = 90; /*positions initiales des servomoteurs*/
int leftPos = 90;
int rightPos = 90;
int clawPos = 35;
int servoStep = 2; /*increment de deplacement des servo en degres*/

void setup() {
  Serial.begin(9600) ;

  middle.attach(3);  /*assignation des broches d'Arduino connectees aux
servomoteurs*/
  left.attach(5);
  right.attach(6);
  claw.attach(9) ;
```

```
  middle.write(middlePos);  /*initialisation des positions des
servomoteurs*/
  left.write(leftPos);
  right.write(rightPos);
  claw.write(clawPos);
  delay(200);
}

void loop() {

  if (Serial.available() > 0) {
    // read the incoming byte:
    incomingByte = Serial.read();
    //
    // Servo middle pilot with left (l) and right (r)
    if (incomingByte == 'l') {
      middlePos = middlePos - servoStep;
      middle.write(middlePos);
    }
    if (incomingByte == 'r') {
      middlePos = middlePos + servoStep;
      middle.write(middlePos);
    }
    //
    // Servo right pilot with forward (f) and backward (b)
    if (incomingByte == 'f') {
      rightPos = rightPos - servoStep;
      right.write(rightPos);
    }
    if (incomingByte == 'b') {
      rightPos = rightPos + servoStep;
      right.write(rightPos);
    }
    //
    // Servo left pilot with up (u) and down (d)
    if (incomingByte == 'u') {
      leftPos = leftPos - servoStep;
      left.write(leftPos);
    }
    if (incomingByte == 'd') {
      leftPos = leftPos + servoStep;
      left.write(leftPos);
    }
    //
    // Servo claw pilot with open (o) and close (c)
    if (incomingByte == 'o') {
      clawPos = clawPos - servoStep;
      claw.write(clawPos);
    }
    if (incomingByte == 'c') {
      clawPos = clawPos + servoStep;
      claw.write(clawPos);
    }
    //
    // Homing all servos positions
    if (incomingByte == 'h') {
      leftPos = 90;
      left.write(leftPos);
      rightPos = 90;
      right.write(rightPos);
      middlePos = 90;
```

```
      middle.write(middlePos);
      clawPos = 35;
      claw.write(clawPos);
    }
    //
    // say what you received in ASCII
    //Serial.print("I received: ");
    Serial.println(incomingByte);
  }
}
```

Code de Python 2.7 complet

```
#--------------------------------------------------------------------------
-----
# Name:        MeArm Nuka robot Pilot via serial COM Port
# Author:      Nicolas Blanchard
# Created:      11/05/2017
# Copyright:    (CC BY-NC-SA 4.0)
#--------------------------------------------------------------------------
-----

from serial import *
from Tkinter import *
from math import *
from ScrolledText import *
import serial.tools.list_ports
import time
import threading

# Global variables initialization
comPort='COM1'
# All servo positions given in absolute degrees
midPos=90.0
leftPos=90.0
righPos=90.0
clawPos=35.0
servoStep=2
# All cartesian positions given in millimeters
Rpos=80.0
Xpos=0.0
Ypos=80+50.0
Zpos=80+40.0
Xstep=10.0
# Code computation vriables
midCurrent=90.0
leftCurrent=90.0
righCurrent=90.0
clawCurrent=35.0
midNext=90.0
leftNext=90.0
righNext=90.0
clawNext=35.0
midStep=0
leftStep=0
righStep=0
clawStep=0
maxStep=0
# Sign=0 corresponds to 0(no move), sign=1 corresponds to -, sign=2
correponds to +
midSign=0
```

```
leftSign=0
rightSign=0
clawSign=0
currentLine=""
coord=[0,0,0]
loopNb=1

#Communication Data Settings:
# 'l' = mid "-" function mid1
# 'r' = mid "+" function mid2
# 'f' = left "-" function left1
# 'b' = left "+" function left2
# 'u' = right "-" function right1
# 'd' = right "+" function right2
# 'o' = claw "-" function claw1
# 'c' = claw "+" function claw2
# 'h' = home all servos function homeServos

# Getting COM PORT for connection
def connectPort(event):
    global comPort
    comPort=entryCOM.get()
    print comPort

def computePosition():
    global midPos,righPos,leftPos,Xpos,Ypos,Zpos,Rpos
    Rpos=80*sin((leftPos-90)/180*pi)+80*cos((righPos-90)/180*pi)+50
    Xpos=Rpos*sin((midPos-90)/180*pi)
    Ypos=Rpos*cos((midPos-90)/180*pi)
    Zpos=80*cos((leftPos-90)/180*pi)-80*sin((righPos-90)/180*pi)+40
##    print str(round(Xpos,1))+str(round(Ypos,1))+str(round(Zpos,1))
    refreshValues()

def refreshValues():
    global midPos,leftPos,righPos,clawPos,Xpos,Ypos,Zpos,Rpos
    labelMiddleValue.configure(text=str(midPos))
    labelLeftValue.configure(text=str(leftPos))
    labelRightValue.configure(text=str(righPos))
    labelClawValue.configure(text=str(clawPos))
    XPosValue.configure(text=str(round(Xpos,1)))
    YPosValue.configure(text=str(round(Ypos,1)))
    ZPosValue.configure(text=str(round(Zpos,1)))
    RPosValue.configure(text=str(round(Rpos,1)))

def mid1():
    global midPos,servoStep,comPort
    midPos=midPos-servoStep
    computePosition()
    refreshValues()
    if port_serie.isOpen():
        port_serie.write('l')

def mid2():
    global midPos,servoStep,comPort
    midPos=midPos+servoStep
    computePosition()
    refreshValues()
    if port_serie.isOpen():
        port_serie.write('r')

def left1():
```

```
    global leftPos,servoStep,comPort
    leftPos=leftPos-servoStep
    computePosition()
    refreshValues()
    if port_serie.isOpen():
        port_serie.write('f')

def left2():
    global leftPos,servoStep,comPort
    leftPos=leftPos+servoStep
    computePosition()
    refreshValues()
    if port_serie.isOpen():
        port_serie.write('b')

def right1():
    global righPos,servoStep,comPort
    righPos=righPos-servoStep
    computePosition()
    refreshValues()
    if port_serie.isOpen():
        port_serie.write('u')

def righ2():
    global righPos,servoStep,comPort
    righPos=righPos+servoStep
    computePosition()
    refreshValues()
    if port_serie.isOpen():
        port_serie.write('d')

def claw1():
    global clawPos,servoStep,comPort
    clawPos=clawPos-servoStep
    computePosition()
    refreshValues()
    if port_serie.isOpen():
        port_serie.write('o')

def claw2():
    global clawPos,servoStep,comPort
    clawPos=clawPos+servoStep
    computePosition()
    refreshValues()
    if port_serie.isOpen():
        port_serie.write('c')

def homeServos():
    global midPos,leftPos,righPos,clawPos,Xpos,Ypos,Zpos
    midPos=90
    leftPos=90
    righPos=90
    clawPos=35
    computePosition()
    refreshValues()
    if port_serie.isOpen():
        port_serie.write('h')

def runCode():
    global
midPos,leftPos,righPos,clawPos,comPort,midCurrent,midNext,leftCurrent,leftN
```

```
ext,righCurrent,righNext,clawCurrent,clawNext,currentLine,coord,loopNb,midP
os,leftPos,righPos,clawPos,Xpos,Ypos,Zpos
    if port_serie.isOpen():
        loopNb=int(textLoop.get(1.0,END))
        while loopNb>0:
            port_serie.write('h')
            midCurrent=90.0
            leftCurrent=90.0
            righCurrent=90.0
            clawCurrent=35.0

            textBuffer=textCode.get(1.0,END)
            a=textBuffer.split('\n')
            j=0
            while a[j]!="":
                currentLine=a[j]
                # Writing the moves lines
                if currentLine[0:3]=="GO:":
                    currentLine=currentLine.strip("GO:")
                    coord=currentLine.split(",")
                    midNext=float(coord[0])
                    leftNext=float(coord[1])
                    righNext=float(coord[2])
                    clawNext=float(coord[3])
                    midStep=round((midNext-midCurrent)/2,0)
                    print midStep
                    if midStep==0:
                        midSign=0
                    if midStep>0:
                        midSign=2
                    if midStep<0:
                        midSign=1
                        midStep=-midStep

                    leftStep=round((leftNext-leftCurrent)/2,0)
                    print leftStep
                    if leftStep==0:
                        leftSign=0
                    if leftStep>0:
                        leftSign=2
                    if leftStep<0:
                        leftSign=1
                        leftStep=-leftStep

                    righStep=round((righNext-righCurrent)/2,0)
                    if righStep==0:
                        righSign=0
                    if righStep>0:
                        righSign=2
                    if righStep<0:
                        righSign=1
                        righStep=-righStep

                    clawStep=round((clawNext-clawCurrent)/2,0)
                    if clawStep==0:
                        clawSign=0
                    if clawStep>0:
                        clawSign=2
                    if clawStep<0:
                        clawSign=1
                        clawStep=-clawStep
```

```
                    maxStep=max(midStep,leftStep,righStep,clawStep)
                    print maxStep
                    i=0
                    while i<=maxStep:
                        if midStep>0:
                            if midSign==1:
                                port_serie.write('l')
                                midPos=midPos-servoStep
                            if midSign==2:
                                port_serie.write('r')
                                midPos=midPos+servoStep
                            midStep=midStep-1
                        if leftStep>0:
                            if leftSign==1:
                                port_serie.write('f')
                                leftPos=leftPos-servoStep
                            if leftSign==2:
                                port_serie.write('b')
                                leftPos=leftPos+servoStep
                            leftStep=leftStep-1
                        if righStep>0:
                            if righSign==1:
                                port_serie.write('u')
                                righPos=righPos-servoStep
                            if righSign==2:
                                port_serie.write('d')
                                righPos=righPos+servoStep
                            righStep=righStep-1
                        if clawStep>0:
                            if clawSign==1:
                                port_serie.write('o')
                                clawPos=clawPos-servoStep
                            if clawSign==2:
                                port_serie.write('c')
                                clawPos=clawPos+servoStep
                            clawStep=clawStep-1
                        computePosition()
                        #time.sleep(1)
                        event= threading.Event()
                        event.wait(timeout=0.1)
                        i=i+1
                    midCurrent=midNext
                    leftCurrent=leftNext
                    righCurrent=righNext
                    clawCurrent=clawNext
                j=j+1
            loopNb=loopNb-1

# Interface graphique
win1=Tk()
win1.title("MeArm PYTHON Pilot")

# Com Port Connection
labelJoint=Label(win1,text="SERIAL CONNECTION",bg='lightyellow',width=24)
labelJoint.grid(row=0,column=0,padx=5,pady=6)

fr3=Frame(win1)
fr3.grid(row=1,column=0,padx=5,pady=1,sticky='n')

labelCom=Label(fr3,text="Arduino COM Port detected:")
```

```
labelCom.grid(row=0,column=0,padx=5,pady=3)
labelPort=Label(fr3,text="NONE",width=21,bg='white')
labelPort.grid(row=1,column=0,padx=5,pady=1)
labelConnected=Label(fr3,text="Disconnected",bg='Red',width=21)
labelConnected.grid(row=3,column=0,padx=5,pady=1)

# Getting the connected ComPorts and selecting the one connected to Arduino
ports = list(serial.tools.list_ports.comports())
for p in ports:
    if p[1].find('Arduino')!=-1:
        comPort=p[0]
        print comPort
        labelPort.configure(text=comPort)
        labelConnected.configure(text="Connected",bg='Green')
        port_serie=Serial(port=comPort, baudrate=9600, timeout=1,
writeTimeout=1)
 #       connectPort()

# Servo pilot and display
labelJoint=Label(win1,text="JOINT SPACE",bg='lightgreen',width=24)
labelJoint.grid(row=0,column=2,padx=5,pady=6,sticky='n')

fr2=Frame(win1)
fr2.grid(row=1,column=2,padx=5,pady=1)

    # Servo Middle
labelMiddle=Label(fr2,text="MIDDLE",width=6)
labelMiddle.grid(row=0,column=3,padx=5,pady=3)
labelMiddleValue=Label(fr2,text=str(midPos),width=5,bg='white')
labelMiddleValue.grid(row=1,column=3,padx=1,pady=1)
buttonMid1=Button(fr2,text='<-',width=5,command=mid1)
buttonMid1.grid(row=1,column=2,padx=5,pady=1)
buttonMid2=Button(fr2,text='->',width=5,command=mid2)
buttonMid2.grid(row=1,column=4,padx=5,pady=1)

# Servo Left
labelLeft=Label(fr2,text="LEFT",width=6)
labelLeft.grid(row=2,column=3,padx=5,pady=3)
labelLeftValue=Label(fr2,text=str(leftPos),width=5,bg='white')
labelLeftValue.grid(row=3,column=3,padx=1,pady=1)
buttonLeft1=Button(fr2,text='<-',width=5,command=left1)
buttonLeft1.grid(row=3,column=2,padx=5,pady=1)
buttonLeft2=Button(fr2,text='->',width=5,command=left2)
buttonLeft2.grid(row=3,column=4,padx=5,pady=1)

# Servo Right
labelRight=Label(fr2,text="RIGHT",width=6)
labelRight.grid(row=4,column=3,padx=5,pady=3)
labelRightValue=Label(fr2,text=str(righPos),width=5,bg='white')
labelRightValue.grid(row=5,column=3,padx=1,pady=1)
buttonRight1=Button(fr2,text='<-',width=5,command=right1)
buttonRight1.grid(row=5,column=2,padx=5,pady=1)
buttonRight2=Button(fr2,text='->',width=5,command=righ2)
buttonRight2.grid(row=5,column=4,padx=5,pady=1)

# Servo Claw
labelClaw=Label(fr2,text="CLAW",width=6)
labelClaw.grid(row=6,column=3,padx=5,pady=3)
labelClawValue=Label(fr2,text=str(clawPos),width=5,bg='white')
labelClawValue.grid(row=7,column=3,padx=1,pady=1)
buttonClaw1=Button(fr2,text='<-',width=5,command=claw1)
```

```
buttonClaw1.grid(row=7,column=2,padx=5,pady=1)
buttonClaw2=Button(fr2,text='->',width=5,command=claw2)
buttonClaw2.grid(row=7,column=4,padx=5,pady=1)

#Servos home
buttonHome=Button(win1,text='Home Servos',width=16,command=homeServos)
buttonHome.grid(row=7,column=2,padx=5,pady=10)

# Exit Button
buttonExit=Button(win1,text="Exit",width=10,command=win1.destroy)
buttonExit.grid(row=7,column=6,padx=5,pady=10)

# Cartesian position computing
labelCartesian=Label(win1,text="CARTESIAN SPACE",bg='lightblue',width=24)
labelCartesian.grid(row=0,column=6,padx=5,pady=6)

fr1=Frame(win1)
fr1.grid(row=1,column=6,padx=5,pady=1,sticky='n')

XPosLabel=Label(fr1,text="X POS",width=5)
XPosLabel.grid(row=0,column=6,padx=5,pady=1)
XPosValue=Label(fr1,text=str(Xpos),width=5,bg='white')
XPosValue.grid(row=1,column=6,padx=5,pady=1)

YPosLabel=Label(fr1,text="Y POS",width=5)
YPosLabel.grid(row=2,column=6,padx=5,pady=1)
YPosValue=Label(fr1,text=str(Ypos),width=5,bg='white')
YPosValue.grid(row=3,column=6,padx=5,pady=1)

ZPosLabel=Label(fr1,text="Z POS",width=5)
ZPosLabel.grid(row=4,column=6,padx=5,pady=1)
ZPosValue=Label(fr1,text=str(Zpos),width=5,bg='white')
ZPosValue.grid(row=5,column=6,padx=5,pady=1)

RPosLabel=Label(fr1,text="R POS",width=5)
RPosLabel.grid(row=6,column=6,padx=5,pady=1)
RPosValue=Label(fr1,text=str(Rpos),width=5,bg='white')
RPosValue.grid(row=7,column=6,padx=5,pady=1)

labelCartesian=Label(win1,text="JOINT MOTION CODE",bg='orange',width=24)
labelCartesian.grid(row=0,column=7,padx=5,pady=6)

fr3=Frame(win1)
fr3.grid(row=1,column=7,padx=5,pady=1,sticky='n')
textCode=ScrolledText(fr3,width=40,height=18,bg='white')
textCode.grid(row=1,column=0,rowspan=18)

# Loop selection
labelLoop=Label(win1,text="Loop Number",bg='orange',width=17)
labelLoop.grid(row=5,column=7,padx=5,pady=1)
textLoop=Text(win1,width=10,height=1)
textLoop.grid(row=6,column=7,padx=5,pady=1)
textLoop.insert(END,"1")

# Run Button
buttonExit=Button(win1,text="RUN Code",width=16,command=runCode)
buttonExit.grid(row=7,column=7,padx=5,pady=10)


win1.mainloop()
```