ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

PROYECTO DE GRADO:

"SISTEMA DE CONTROL DE BRAZO ROBÓTICO MEDIANTE ONDAS CEREBRALES DESARROLLADO EN SOFTWARE LIBRE PARA ASISTENCIA A PERSONAS CON CAPACIDADES ESPECIALES "

Autores:

Sr. Leonardo Solís

Sr. Andrés Tapia

Tutor:

PhD. Víctor Andaluz

AGENDA:

- INTRODUCCIÓN.
- IMPLEMENTACIÓN DEL SISTEMA.
- ANÁLSIS DE RESULTADOS.
- CONCLUSIONES.
- RECOMENDACIONES.

INTRODUCCIÓN:

OBJETIVO GENERAL:

- Realizar el diseño e implementación de un sistema de control de brazo robótico mediante ondas cerebrales desarrollado en software libre para asistencia a personas con capacidades especiales.

OBJETIVOS ESPECÍFICOS:

- Estudiar el funcionamiento de la cinemática del brazo robótico de seis grados de libertad, así como también la utilización del casco Emotiv EPOC para la captación de las señales cerebrales.
- Desarrollar en Python una plataforma que permita reconocer y adquirir las diferentes señales cerebrales proporcionadas por el casco Emotiv EPOC.
- Desarrollar un algoritmo de control que tenga como entradas las señales cerebrales emitidas por el casco Emotiv EPOC para controlar al brazo robótico de seis grados de libertad.
- Realizar pruebas de funcionamiento y verificar el correcto movimiento del brazo robótico.

Elementos para el desarrollo del proyecto

- CASCO EMOTIV EPOC
- SOFTWARE PYTHON
- BRAZO ROBÓTICO DE SEIS GRADOS DE LIBERTAD

CASCO EMOTIV EPOC

- El casco Emotiv EPOC permite captar y amplificar ondas cerebrales realizadas por acciones mentales o gestos faciales, permitiendo así el control de funciones de una computadora.
- El EPOC es un aparato tipo diadema que posee dieciséis electrodos distribuidos en diferentes puntos de la cabeza como se muestra en la siguiente figura.

Casco Emotiv Epoc

SOFTWARE PYTHON

- Python es un lenguaje de programación interpretado que hace énfasis en presentar una sintaxis que beneficie a realizar un código legible, es potente y fácil de aprender, posee estructuras de alto nivel con una perspectiva orientado a objetos.

Icono de Python

BRAZO ROBÓTICO

- El brazo robótico de 6 grados de libertad proporcionado por la Universidad de las Fuerzas Armadas Extensión Latacunga, esta ensamblado con servomotores Dynamixel de las series: MX-28, MX-64 y MX-106, los mismo deben ser energizados con un voltaje de 11,1Vdc a 14,8Vdc y su consumo de corriente es de 1.7A, 5.2A y 6.3A para los modelos de la serie MX-28, MX-64 y MX-106 respectivamente.

Interfaz USB2Dynamixel

La comunicación entre la red de servomotores de marca Dynamixel y la PC, será mediante el uso de la interfaz USB2Dynamixel, la misma que debe ser conectada a un puerto USB del ordenador para poder establecer el enlace requerido.

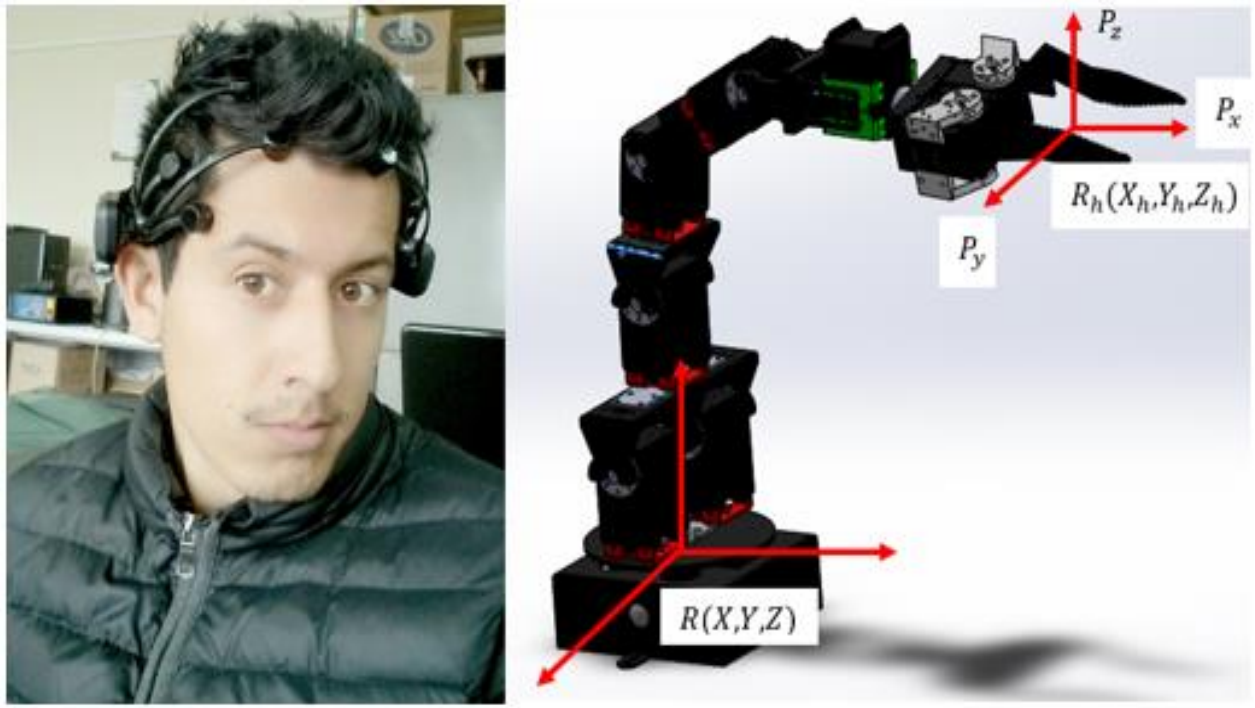
Interfaz USB2Dynamixel

IMPLEMENTACIÓN DEL SISTEMA

Modelamiento cinemático del brazo robótico

- El control del brazo robótico a realizar es mediante el uso de las señales cerebrales obtenidas por medio del casco Emotiv EPOC, las mismas que se transforman en posición hacia el extremo operativo del brazo y al ser enviadas al robot ejecuta el movimiento requerido por el usuario.

- El operador humano controla el brazo robótico mediante el envío de comandos de posición hacia el extremo operativo del robot: $h_l,\ h_m, h_n,$ uno por cada eje con respecto al marco inercial $R(X, Y, Z)$ utilizando un dispositivo háptico. Los comandos del operador humano P_x, P_y, P_z son generados con el uso del dispositivo Emotiv EPOC como se indica en la siguiente figura.



Comandos generados por el operador humano hacia el brazo robótico

Matriz de rotación

- Las posiciones P_x, P_y, P_z son traducidas en comandos de posición del extremo operativo h_l, h_m, h_n para el modo de manipulación, a través de la siguiente matriz de rotación.

$$\begin{bmatrix} h_l \\ h_m \\ h_n \end{bmatrix} = \begin{bmatrix} cos(q_1) & -sin(q_1) & 0 \\ sin(q_1) & cos(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

donde q_1 representa la primera articulación del brazo robótico que gira alrededor del eje Z.

Modelamiento cinemático directo

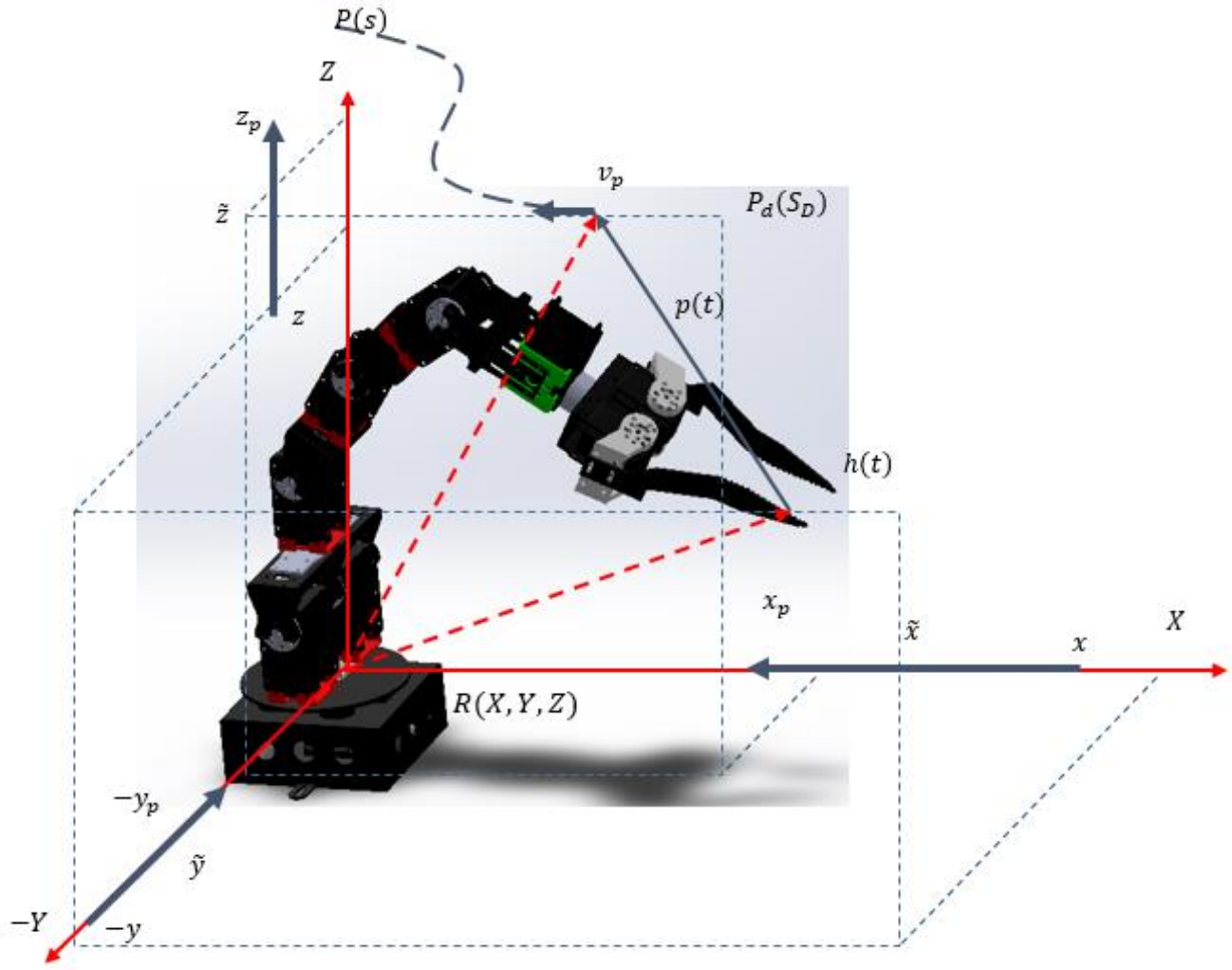
- El modelo cinemático instantáneo de un brazo robótico está definido por la derivada de la ubicación del extremo operativo como una función de las derivadas de la configuración del brazo robótico.

$$\dot{\mathbf{h}}(\mathrm{t}) = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}(\mathrm{t})$$

donde $\boldsymbol{J}(\boldsymbol{q})$ es la matriz Jacobiana que define un operador lineal entre el vector de las velocidades de las articulaciones del brazo robótico $\dot{\boldsymbol{q}}(t)$ y el vector de velocidad del extremo operativo $\dot{\boldsymbol{h}}(t)$.

Control para el brazo robótico

- El diseño del controlador cinemático del brazo robótico se basa en el modelo cinemático del brazo. En la siguiente figura se muestra la proyección ortogonal de un punto deseado.



Ley de control

- Se propone la siguiente ley de control

$$\dot{\boldsymbol{q}}_{\boldsymbol{c}} = \boldsymbol{J}^{\#}\left(\boldsymbol{v}_{\boldsymbol{d}} + \boldsymbol{L}_{\boldsymbol{K}}\, \boldsymbol{tanh}\left(\boldsymbol{L}_{\boldsymbol{K}}^{-\boldsymbol{1}}\boldsymbol{K}\tilde{\boldsymbol{h}}\right)\right) + (\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J})\boldsymbol{L}_{\boldsymbol{B}}\boldsymbol{tanh}\left(\boldsymbol{L}_{\boldsymbol{B}}^{-\boldsymbol{1}}\boldsymbol{B}\Lambda\right)$$

donde $\boldsymbol{J}^{\#} = \boldsymbol{W}^{-1}\boldsymbol{J}^{\boldsymbol{T}}\left(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^{\boldsymbol{T}}\right)^{-1}$, siendo $\boldsymbol{W}$ una matriz positiva definida que pesa las acciones de control del sistema, $\boldsymbol{v_d}$ es el vector velocidad deseado del extremo operativo $\boldsymbol{h}$, $\tilde{\boldsymbol{h}}$ es el vector de errores de control, definido como $\tilde{\boldsymbol{h}} = \boldsymbol{h_d} - \boldsymbol{h}$, $\boldsymbol{B}$ y $\boldsymbol{L_B}$ son matrices diagonales positivas definidas que pesan el vector Λ. Con el fin de incluir una saturación analítica de velocidades en el brazo robótico se propone el uso de la función de $\boldsymbol{tanh}(.)$, lo que limita el error en $\tilde{\boldsymbol{h}}$ y la magnitud del vector Λ. El segundo término representa la proyección sobre el espacio nulo de $\boldsymbol{J}$, donde Λ es un vector arbitrario que contiene las velocidades asociadas al brazo robótico. Por lo tanto, cualquier valor dado de Λ tendrá efectos solamente en la estructura interna del brazo, y no afectará el control final del extremo operativo en absoluto.

Desarrollo de algoritmo de control en Python

- El código está estructurado mediante funciones las mismas que son accionadas mediante botones y checkbox ubicados estratégicamente en la interfaz gráfica desarrollada para interactuar con el usuario.
- La interfaz permite realizar una trayectoria pre-establecida (trayectoria de una circunferencia o una silla de montar), además la interfaz es capaz de establecer un enlace de comunicación de Python y el casco Emotiv EPOC o Python y un Joystick; los dispositivos antes mencionados son los encargados de establecer una trayectoria mediante un control de posición para el extremo operativo del brazo robótico.
- Cabe mencionar que las unidades utilizadas para la codificación del algoritmo de control del brazo robótico se encuentran en: $[m]$, $[rad]$ y $[rad/s]$;

Declaración de Funciones

- **Comunicación ().-** Permite establecer la comunicación de la interfaz USB2Dynamixel con el software Python

```
#COMUNICAR SERVOMOTORES CON PC
def Comunicacion (evt):

    global usb
    print("INICIANDO COMUNICACION CON USB2Dynamixel...")
    usb = libc.dxl_initialize()
    if usb == 0:
        print("ERROR CON USB2Dynamixel!")
        exit()
    print("USB2Dynamixel ABIERTO CORRECTAMENTE!\n")
    arti()
    time.sleep(1)
    libc.dxl_write_byte(1,26,1),libc.dxl_write_byte(1,27,3),libc.dxl_write_byte(1,28,32)
    libc.dxl_write_byte(2,26,2),libc.dxl_write_byte(2,27,2),libc.dxl_write_byte(2,28,32)
    libc.dxl_write_byte(3,26,2),libc.dxl_write_byte(3,27,2),libc.dxl_write_byte(3,28,32)
    libc.dxl_write_byte(4,26,5),libc.dxl_write_byte(4,27,2),libc.dxl_write_byte(4,28,32)
    libc.dxl_write_byte(5,26,1),libc.dxl_write_byte(5,27,2),libc.dxl_write_byte(5,28,32)
    libc.dxl_write_byte(6,26,2),libc.dxl_write_byte(6,27,5),libc.dxl_write_byte(6,28,40)
    libc.dxl_write_byte(7,26,2),libc.dxl_write_byte(7,27,3),libc.dxl_write_byte(7,28,35)
    libc.dxl_write_byte(8,26,2),libc.dxl_write_byte(8,27,5),libc.dxl_write_byte(8,28,32)
    libc.dxl_write_byte(9,26,2),libc.dxl_write_byte(9,27,5),libc.dxl_write_byte(9,28,32)
```

- **Rueda ().-** Por medio de las direcciones de memoria 6 y 8 de los servomotores Dynamixel se puede configurar para que funcionen en modo rueda (como motores) asignando a estas direcciones el valor de cero, a continuación se configura la dirección 32 (velocidad de movimiento) para que cada uno de los servomotores se quede estático. Los valores entre 0-1023 hacen girar al servomotor en sentido antihorario y en el rango de 1024-2047 para el giro en sentido horario.

```
#ESTABLECER A LOS SERVOMOTORES COMO RUEDA
def rueda():

    libc.dxl_write_word(1,6,0)
    libc.dxl_write_word(1,8,0)
    libc.dxl_write_word(2,6,0)
    libc.dxl_write_word(2,8,0)
    libc.dxl_write_word(3,6,0)
    libc.dxl_write_word(3,8,0)
    libc.dxl_write_word(4,6,0)
    libc.dxl_write_word(4,8,0)
    libc.dxl_write_word(5,6,0)
    libc.dxl_write_word(5,8,0)
    libc.dxl_write_word(6,6,0)
    libc.dxl_write_word(6,8,0)
    libc.dxl_write_word(1,32,0)
    libc.dxl_write_word(2,32,0)
    libc.dxl_write_word(3,32,0)
    libc.dxl_write_word(4,32,0)
    libc.dxl_write_word(5,32,0)
    libc.dxl_write_word(6,32,0)
```

- **Arti ().-** Esta función es la encargada de configurar a los servomotores para que trabajen en modo articulación, para lo cual es necesario escribir el dato 0 en la dirección 6 y el valor de 4095 en la dirección 8 de cada servomotor Dynamixel.

```
#ESTABLECER A LOS SERVOMOTORES COMO ARTICULACIONES
def arti():

    libc.dxl_write_word(1,32,20)
    libc.dxl_write_word(2,32,20)
    libc.dxl_write_word(3,32,20)
    libc.dxl_write_word(4,32,20)
    libc.dxl_write_word(5,32,20)
    libc.dxl_write_word(6,32,20)
    libc.dxl_write_word(7,32,20)
    libc.dxl_write_word(8,32,20)
    libc.dxl_write_word(9,32,20)
    libc.dxl_write_word(1,6,0)
    libc.dxl_write_word(1,8,4095)
    libc.dxl_write_word(2,6,0)
    libc.dxl_write_word(2,8,4095)
    libc.dxl_write_word(3,6,0)
    libc.dxl_write_word(3,8,4095)
    libc.dxl_write_word(4,6,0)
    libc.dxl_write_word(4,8,4095)
    libc.dxl_write_word(5,6,0)
    libc.dxl_write_word(5,8,4095)
    libc.dxl_write_word(6,6,0)
    libc.dxl_write_word(6,8,4095)
    libc.dxl_write_word(7,6,0)
    libc.dxl_write_word(7,8,4095)
    libc.dxl_write_word(8,6,0)
    libc.dxl_write_word(8,8,4095)
    libc.dxl_write_word(9,6,0)
    libc.dxl_write_word(9,8,4095)
```

- **Velocidad ().-** Asigna mediante la dirección 32 de cada uno de los servomotores un valor en el rango de 0-2047, para establecer la velocidad y sentido de giro.

```
#ESTABLECER A UNA VELOCIDAD DETERMINADA PARA CADA SERVOMOTOR
def velocidad(articulacion,radian):

    if(articulacion=="Pinza"):
        libc.dxl_write_word(9,32,radian)
        libc.dxl_write_word(8,32,radian)
    elif(articulacion=="Munieca"):
        libc.dxl_write_word(7,32,radian)
    elif(articulacion=="Art_4"):
        libc.dxl_write_word(6,32,radian)
    elif(articulacion=="Art_3"):
        libc.dxl_write_word(5,32,radian)
    elif(articulacion=="Art_2"):
        libc.dxl_write_word(4,32,radian)
    elif(articulacion=="Art_1"):
        libc.dxl_write_word(3,32,radian)
        libc.dxl_write_word(2,32,radian)
    elif(articulacion=="Base"):
        libc.dxl_write_word(1,32,radian)
```

- **Leer_servo ().-** Devuelve la posición en la que se encuentra cada uno de los servomotores, ingresando el nombre de la articulación de la que se requiere su posición, se debe tener en cuenta que la pinza está configurada con una apertura de cierre entre 0 y 100%.

```
#LECTURA EN GRADOS DE CADA SERVOMOTOR
def leer_servo(articulacion):
    if articulacion=="Base":
        b=round((libc.dxl_read_word(1,36)*0.087890625),2)
    elif articulacion=="Art_1":
        b=round((libc.dxl_read_word(2,36)*0.087890625),2)
    elif articulacion=="Art_2":
        b=round((libc.dxl_read_word(4,36)*0.087890625),2)
    elif articulacion=="Art_3":
        b=round((libc.dxl_read_word(5,36)*0.087890625),2)
    elif articulacion=="Art_4":
        b=round((libc.dxl_read_word(6,36)*0.087890625),2)
    elif articulacion=="Munieca":
        b=round((libc.dxl_read_word(7,36)*0.087890625),2)
    elif articulacion=="Pinza":
        pinza_servo1=round((0.183486238*libc.dxl_read_word(8,36)-121.1009174))
        pinza_servo2=round((-0.181818181*libc.dxl_read_word(9,36)+254.5454544))
        b=round(((pinza_servo2+pinza_servo1)/2),2)
        if(b>100):
            b=100
        elif(b<0):
            b=0
    return(b)
```

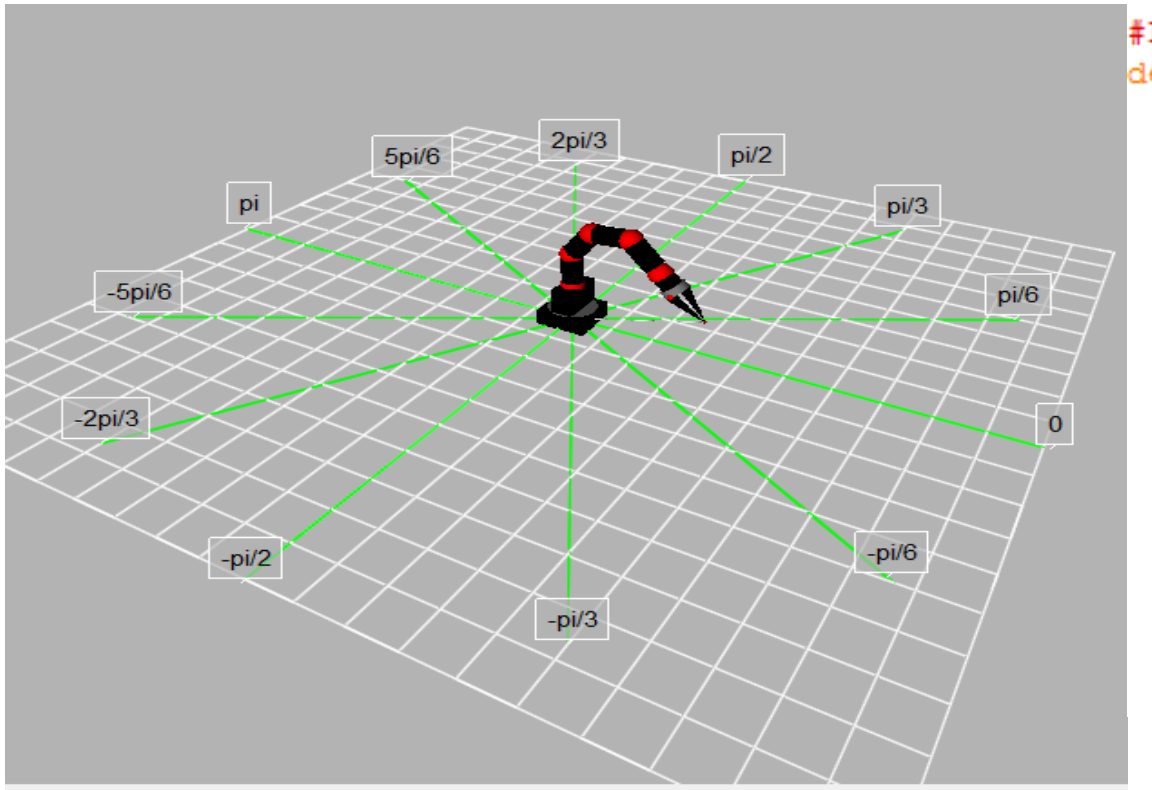
- **Multiplicar ().-** Sirve para multiplicar matrices de n filas por m columnas.

```
#MULTIPLICACIÓN DE MATRICES NXM
def multiplicar (x,y,r,t):
    aux=ones((x,y),float)
    for i in range(0,x):
        for j in range(0,y):
            aux[i][j]=sum(r[i]*t[:,j])
    return aux
```

- **Vel_servo ().-** Determina el sentido de giro y la velocidad de cada uno de los servomotores; con valores entre 0 a 1023 los servomotores giran en sentido horario, y de no ser este el caso se suma 1024 para que giren en sentido antihorario.

```
#DATO DE VELOCIDAD PARA CADA SERVO MOTOR
def vel_servo (radian):
    velocidad = int(round(radian*83.49513168359434))
    if velocidad < 0:
        velocidad = abs(velocidad) + 1024
    return velocidad
```

- **Bits ().-** Su objetivo es determinar el valor que tienen los servomotores, este valor es un dato en el rango de 0 y 4096 que permite determinar su equivalente de grados a radianes positivos en un rango de 0 a π, y de ser el caso que sobrepase el valor de π se trabaja con el mismo valor de signo negativo.




```
#LECTURA EN RADIANES DE CADA SERVOMOTOR
def bits (articulacion):
    if articulacion=="Base":
        b = libc.dxl_read_word(1,36)
    elif articulacion=="Art_1":
        b = libc.dxl_read_word(2,36)
    elif articulacion=="Art_2":
        b = libc.dxl_read_word(4,36)
    elif articulacion=="Art_3":
        b = libc.dxl_read_word(5,36)
    elif articulacion=="Art_4":
        b = libc.dxl_read_word(6,36)
    pos_base = (round(b * 0.087890625))/180
    if pos_base > 1:
       pos_base =  (pos_base - 2)
    pos_base = pos_base * pi
    return pos_base
```

- **Trayectoria ().-** Cumple una determinada trayectoria, dicha trayectoria puede ser una circunferencia o una silla de montar.

```
def Trayectoria(n,radio,altura):
    #ASIGNACIÓN DE CONDICIONES INICIALES

        q1[0] = bits("Base")    # articulacion de la base del manipulador
        q2[0] = bits("Art_1")   # articulacion del primer brazo
        q3[0] = bits("Art_2")   # articulacion del segundo brazo
        q4[0] = bits("Art_3")   # articulacion del tercer brazo
        q5[0] = bits("Art_4")   # articulacion del cuarto brazo

        l1= 0.115
        l2= 0.095
        l3= 0.09
        l4= 0.09
        l5= 0.26
    #TRAYECTORIAS
        if n==0:
        #CIRCUNFERENCIA
            for k in range(0,t_muestreo):
                hxd[k] = radio*cos(0.30*t[k])
                hyd[k] = radio*sin(0.30*t[k])
                hzd[k] = altura

                hxd_p[k] = -radio*sin(0.30*t[k])*0.30
                hyd_p[k] = radio*cos(0.30*t[k])*0.30
                hzd_p[k] = 0
        else:
        #SILLA DE MONTAR
            for k in range(0,t_muestreo):

                hxd[k] = radio*cos(0.30*t[k])
                hyd[k] = radio*sin(0.30*t[k])
                hzd[k] = 0.1*cos(0.60*t[k])+altura

                hxd_p[k] = -radio*sin(0.30*t[k])*0.30
                hyd_p[k] = radio*cos(0.30*t[k])*0.30
                hzd_p[k] = -0.1*sin(0.60*t[k])*0.60
```

- **Torque_todos ().-** Escribe en la dirección 24 de cada uno de los servomotores el valor de 1 para habilitar el torque y el valor de cero en la misma dirección para deshabilitar el torque.

```
def torque_todos(evt):

    if torque.GetValue():
        libc.dxl_write_word(1,24,1)
        libc.dxl_write_word(2,24,1)
        libc.dxl_write_word(3,24,1)
        libc.dxl_write_word(4,24,1)

        libc.dxl_write_word(5,24,1)
        libc.dxl_write_word(6,24,1)
        libc.dxl_write_word(7,24,1)
        libc.dxl_write_word(8,24,1)
        libc.dxl_write_word(9,24,1)
    else:
        libc.dxl_write_word(1,24,0)
        libc.dxl_write_word(2,24,0)
        libc.dxl_write_word(3,24,0)
        libc.dxl_write_word(4,24,0)
        libc.dxl_write_word(5,24,0)
        libc.dxl_write_word(6,24,0)
        libc.dxl_write_word(7,24,0)
        libc.dxl_write_word(8,24,0)
        libc.dxl_write_word(9,24,0)
```

- **Emoengine ().-** Habilita la comunicación con casco el Emotiv EPOC, mientras que deshabilita la opción para usarlo el joystick.

```
def emoengine(evt):
    joystick.SetValue(0)
```

- **Joys ()-** Habilita la comunicación con el joystick, mientras que deshabilita la opción para usarlo con el casco Emotiv EPOC.

```
def joys(evt):
    casco.SetValue(0)
```

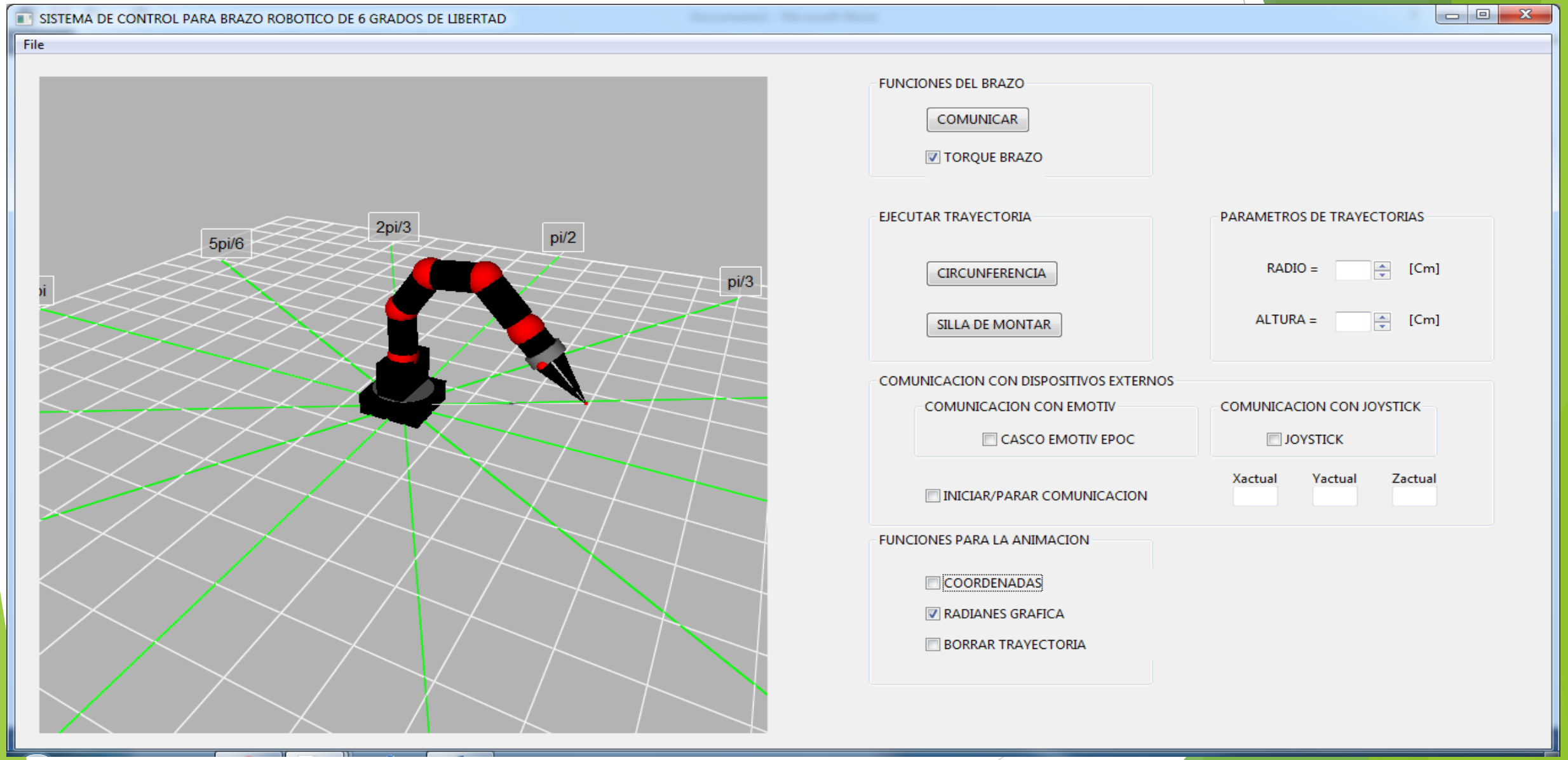
- **Circunferencia ().-** Obtiene los valores de radio y de altura para realizar la trayectoria de una circunferencia.

```
def Circunferencia(evt):
    r = (radio.GetValue())/100
    a = (altura.GetValue())/100
    Trayectoria(0,r,a)
```

- **Silla ().-** Obtiene los valores de radio y de altura para realizar la trayectoria de una silla de montar.

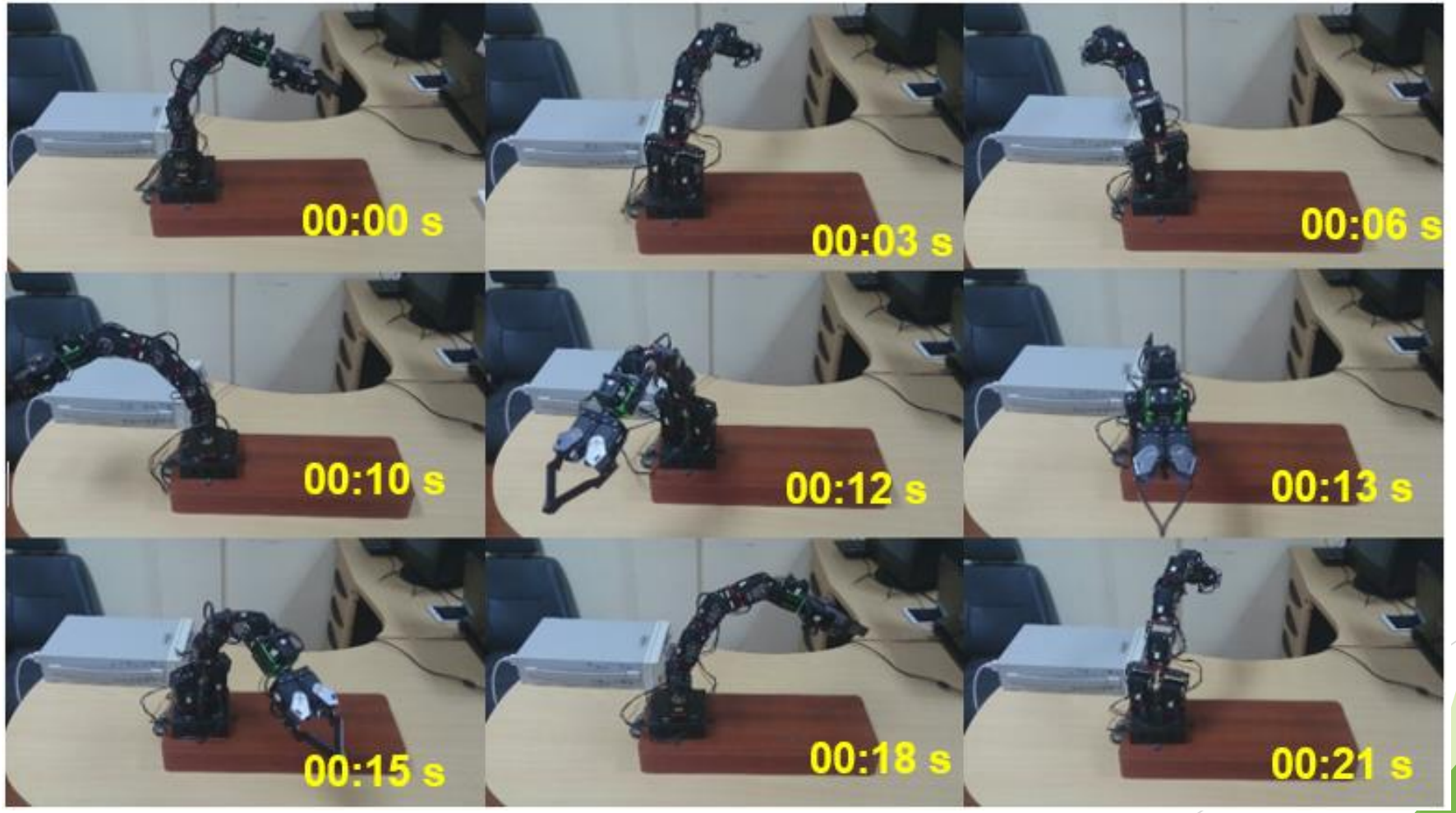
```
def Silla(evt):
    r = (radio.GetValue())/100
    a = (altura.GetValue())/100
    Trayectoria(1,r,a)
```

Interfaz Gráfica

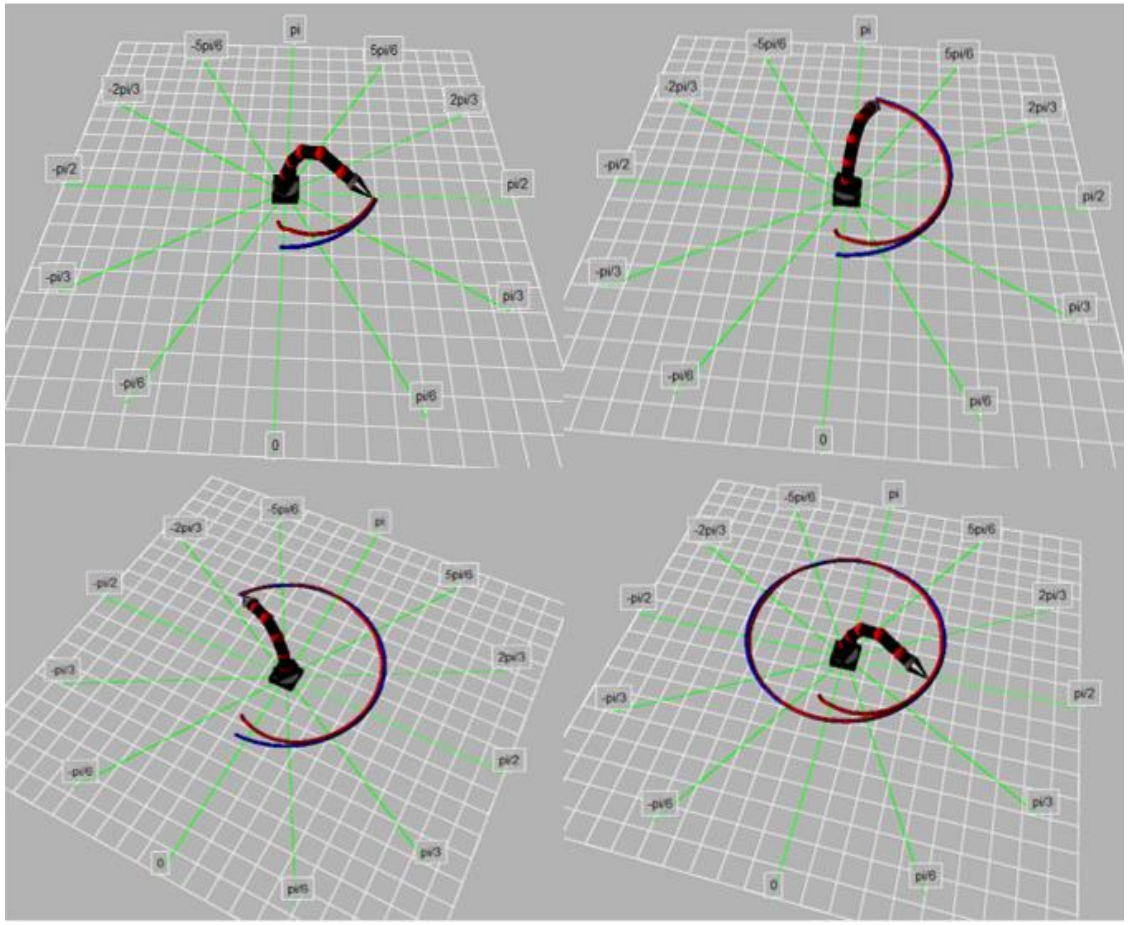


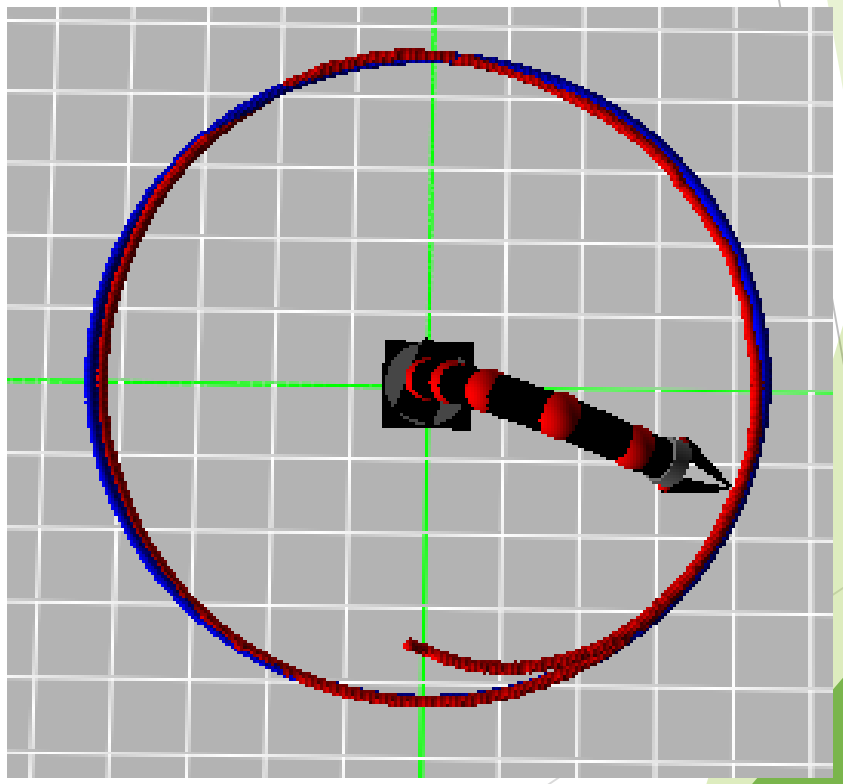
ANÁLISIS DE RESULTADOS

Trayectoria de una circunferencia con radio de 40 cm y una altura igual a 20 cm



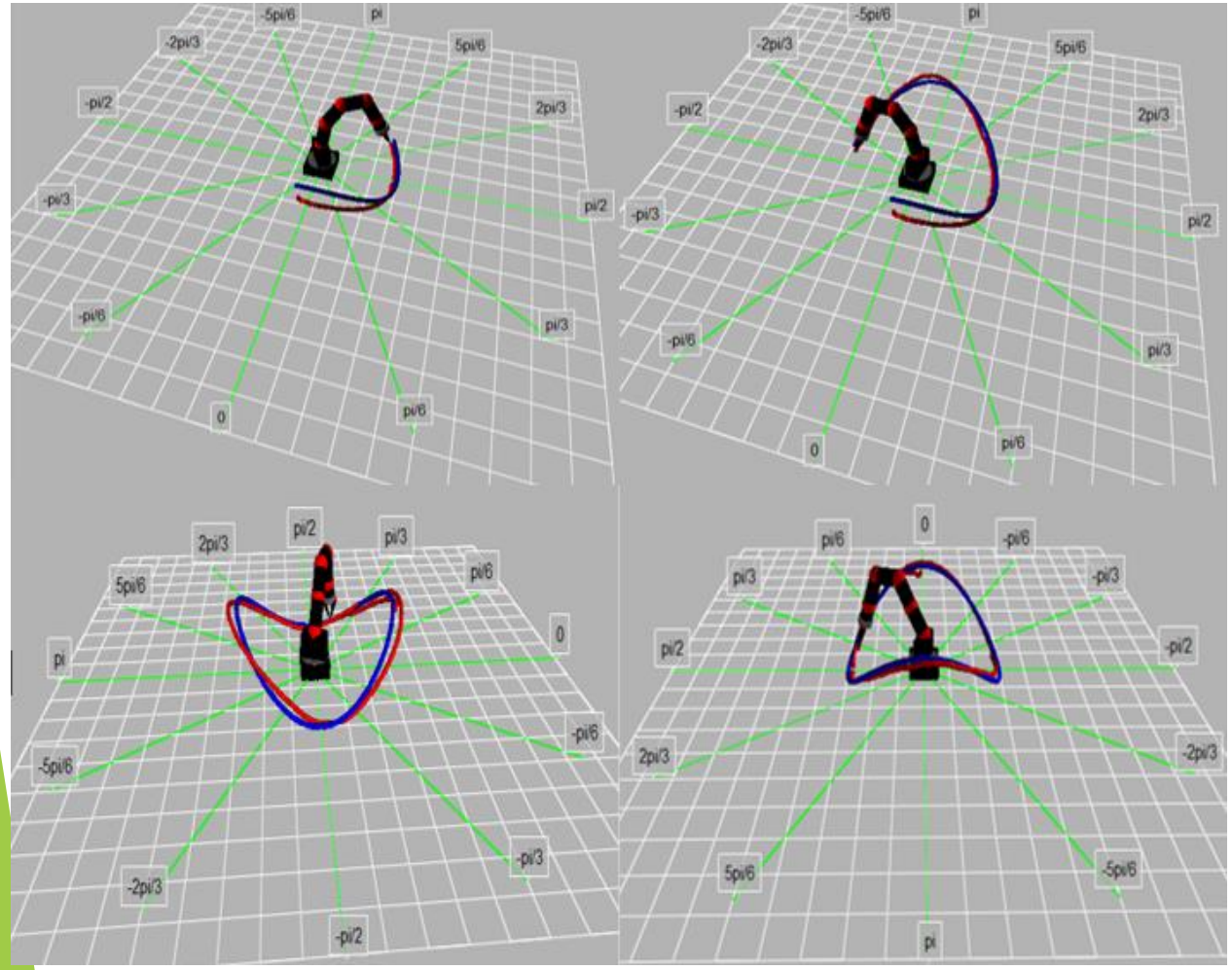
Trayectoria de una circunferencia con radio de 40 cm y una altura igual a 20 cm

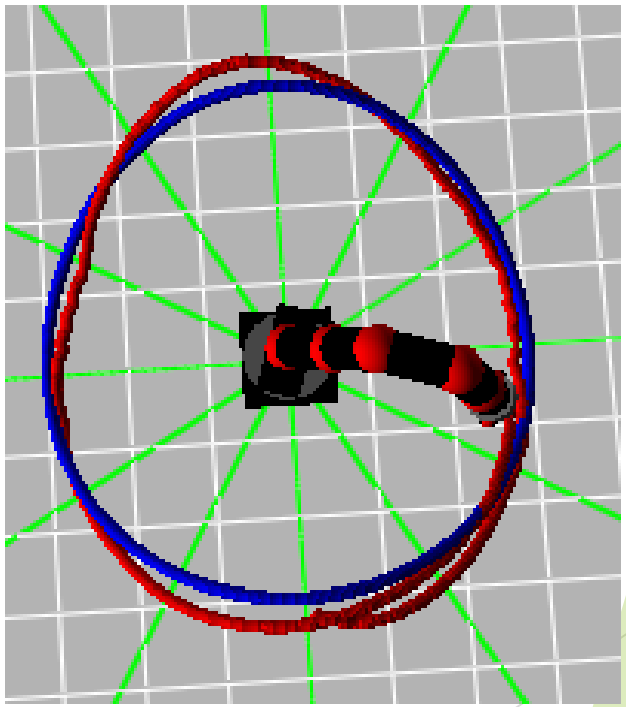




Trayectoria de una silla de montar con valores de radio igual a 30 cm y con una altura igual a 10 cm

Trayectoria de una silla de montar con valores de radio igual a 30 cm y con una altura igual a 10 cm





CONTROL DE POSICIÓN MEDIANTE JOYSTICK

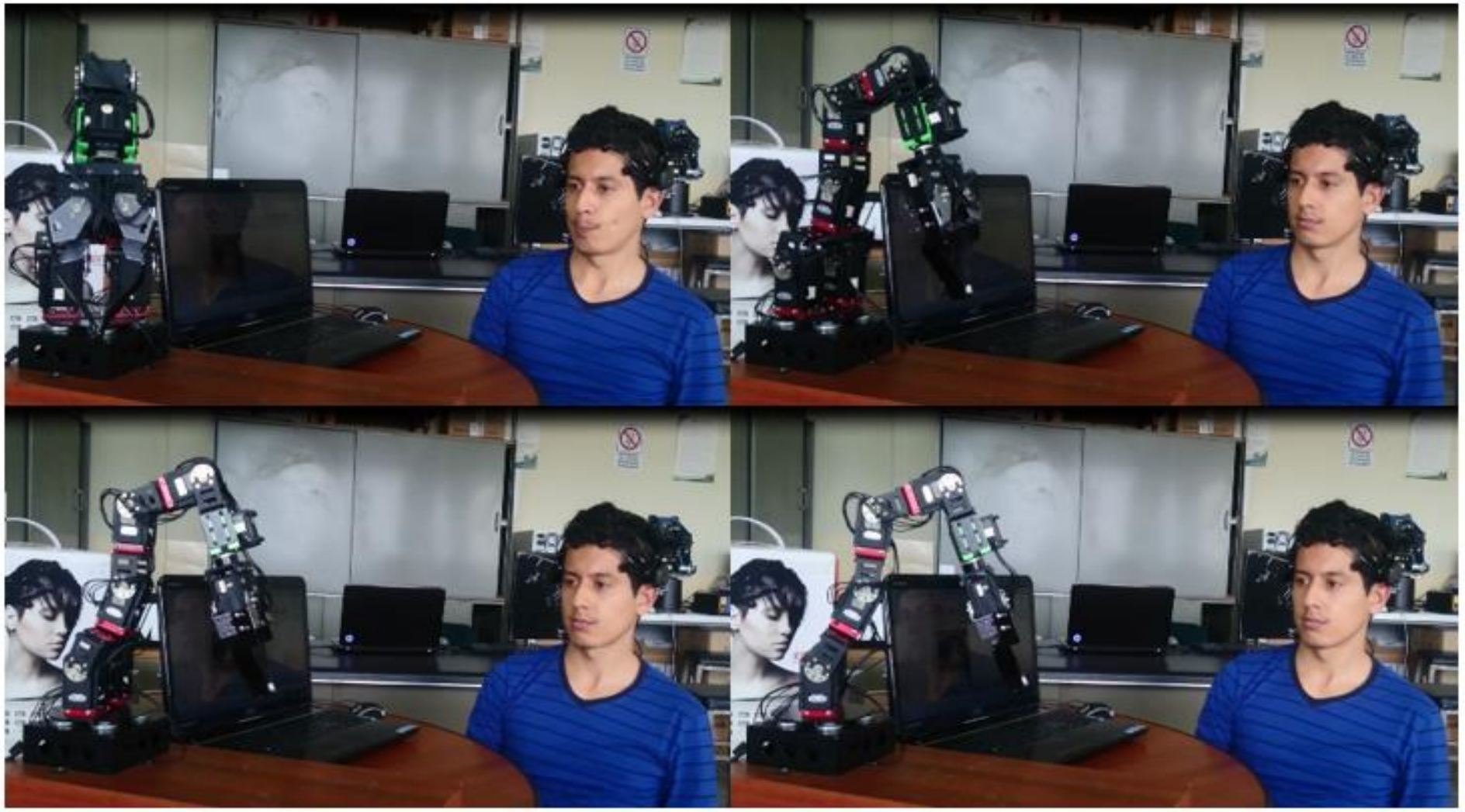
Control de brazo robótico utilizando el casco Emotiv EPOC

- Las acciones obtenidas por el casco Emotiv EPOC son las siguientes: guiñar el ojo derecho/izquierdo el cual sirve para realizar el movimiento del brazo robótico en el eje X, mueca derecha/izquierda que permite desplazar al brazo robótico en el eje Y, mirar hacia la derecha/izquierda para mover el brazo robótico en eje Z positivo.

Movimiento que realiza el brazo robótico hacia adelante a través de un guiño del ojo derecho

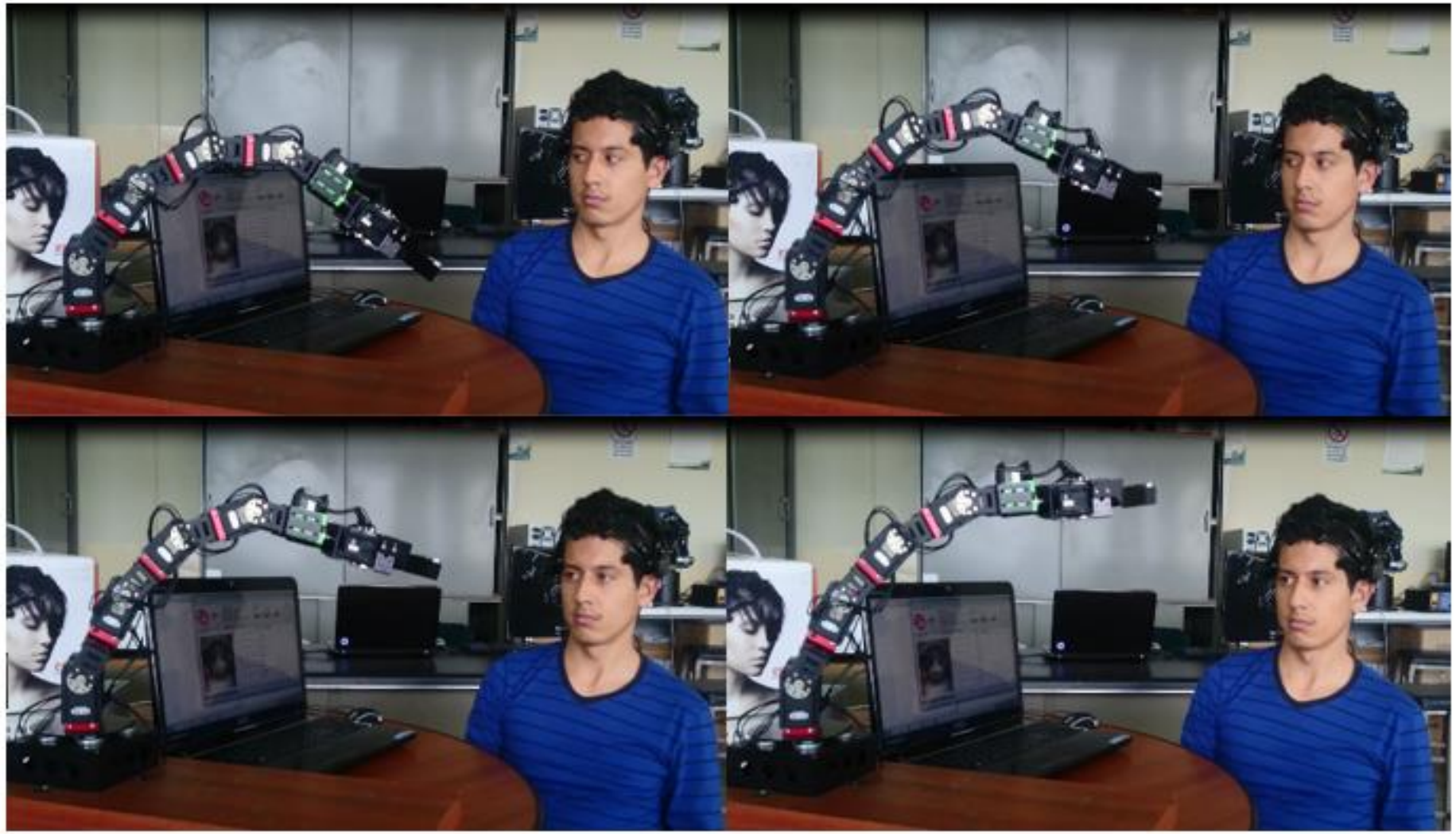
Movimiento que realiza el brazo robótico hacia atrás a través de un guiño del ojo izquierdo

Movimiento que realiza el brazo robótico hacia la derecha a través de una mueca derecha

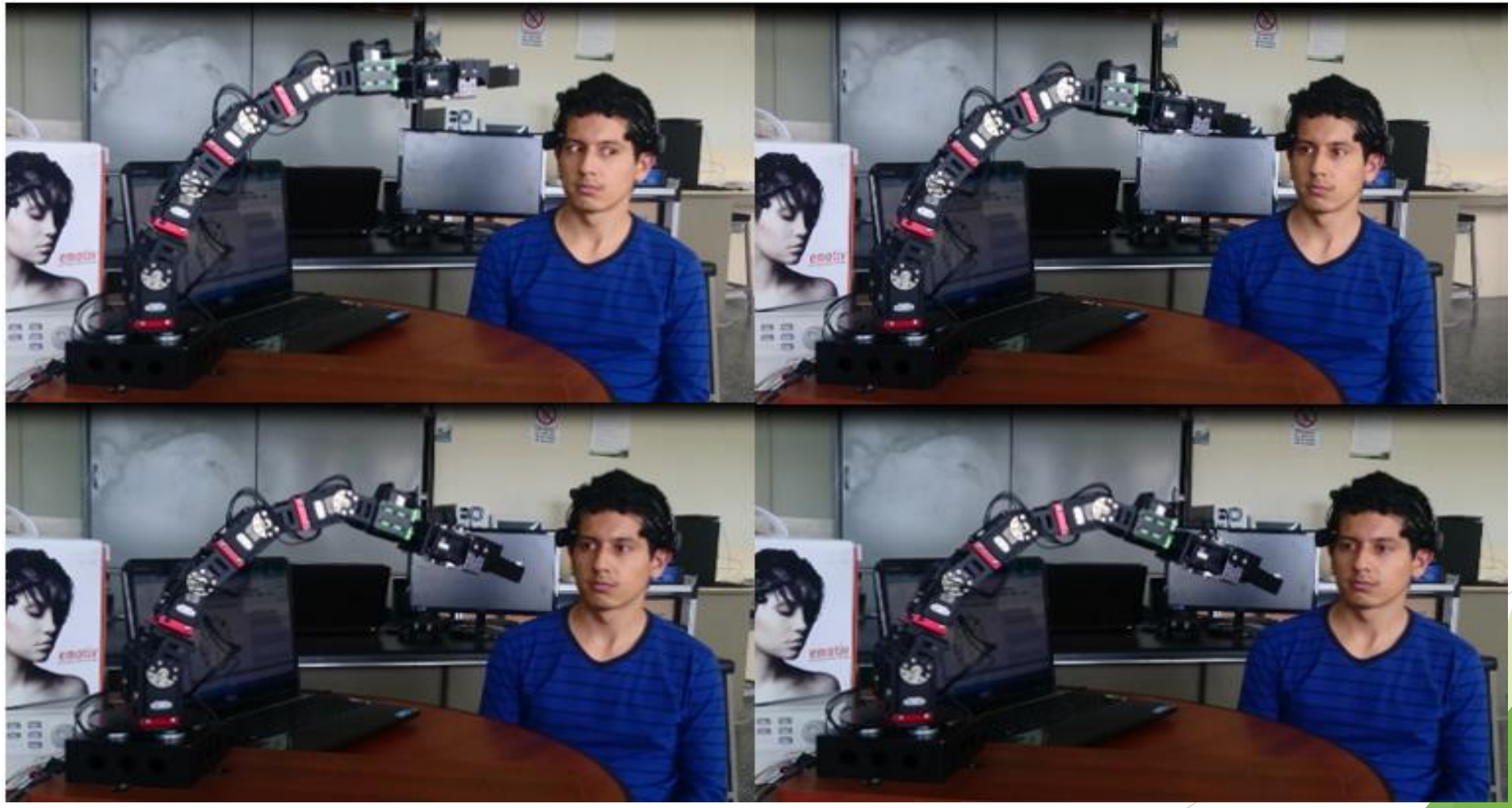


Movimiento que realiza el brazo robótico hacia la izquierda a través de una mueca izquierda

Movimiento que realiza el brazo robótico hacia arriba a través de mirar a la derecha



Movimiento que realiza el brazo robótico hacia abajo a través de mirar a la izquierda



CONCLUSIONES

Conclusiones

- Se estudió el funcionamiento del brazo robótico tomando en cuenta sus cinco primeras articulaciones obviando a la muñeca y pinza del robot, este proceso se lo realizó por medio de cinemática directa en donde se obtuvo su correspondiente relación de velocidades que ubican al extremo operativo del brazo robótico.

- Mediante el estudio y la experimentación se obtuvo conocimientos para la manipulación de las señales obtenidas por el casco Emotiv EPOC que permiten realizar un control de posición tridimensional para el extremo operativo del brazo robótico.

Conclusiones

- En Python se desarrolló una plataforma amigable e intuitiva para que el usuario pueda desenvolverse de una manera fácil y sencilla a través de la misma, pudiendo escoger diferentes modos de control para el brazo robótico, entre ellas la realización de diferentes trayectorias, el control de posición por medio de un joystick o a través de las señales cerebrales emitidas por el casco Emotiv EPOC.

- El algoritmo de control implementado para el brazo robótico que tiene como entradas las señales faciales emitidas por el usuario permite tener movimientos suaves y estables en cada uno de sus eslabones, con lo cual nos aseguramos que el brazo no realice movimientos bruscos que afecten a su estructura mecánica así como también a la red de nueve servomotores Dynamixel.

Conclusiones

- Para un óptimo control del brazo robótico se utilizó las acciones faciales que mejor resultado dieron al momento de su reconocimiento. Al aumentar el número de acciones faciales se pueden obtener falsos positivos y con ello acciones erróneas al momento de realizar el control del robot.

- Se realizó pruebas de funcionamiento con cada uno de los modos de control disponibles en este proyecto, comprobando en cada uno de ellos el correcto funcionamiento de los movimientos que ejecutó el brazo robótico utilizado, teniendo en cada prueba un error tolerable.

RECOMENDACIONES

Recomendaciones

- Se recomienda establecer una área de trabajo en los planos X, Y, Z para que el brazo robótico no sea afectado por ningún sobre esfuerzo mecánico en su estructura, dando como resultado un error en los servomotores afectados. La estructura mecánica del brazo robótico al estar sujeta a una base de madera no permite trabajar con valores negativos para el eje Z.

- Python puede establecer un enlace de comunicación solo con servomotores Dynamixel que tenga una velocidad de transmisión de 1000000 bps, por tal motivo es importante configurar mediante el software ROBOPLUS la velocidad de transmisión de cada uno de los servomotores.

Recomendaciones

- Es importante tomar en cuenta que los electrodos del casco Emotiv EPOC deben estar correctamente humedecidos con la solución líquida que viene en el kit de trabajo del casco, con ello se obtiene mejores resultados al momento de captar y reconocer las señales cerebrales, adicionalmente antes de realizar el control del brazo robótico la persona debe estar concentrada en los movimientos que ejecutará el robot por lo que se recomienda que el usuario tenga la mente descansada y libre de distracciones.

- Se recomienda utilizar una computadora con un sistema operativo Windows 7 de 32 bits, ya que Python es muy restrictivo a la hora de comunicarse con sistemas operativos actuales por falta de compatibilidad con drivers.

GRACIAS POR SU ATENCIÓN