**CINEMÀTICA C.A.R.E.L.E.S:**

Aquest codi esta orientat a moure un braç amb un punt fixe en un entorn controlat i amb un moviments X precisos, invariables.

El codi de cinemàtica serà molt senzill, tindrà una sèrie de funcions que faran que el robot es mogui a unes coordenades prefixades del nostre entorn controlat. Des de un main es cridarà a cadascuna d’aquesta depenent del codi que ens arribi per wifi enviat des del nostre server que es on estarà la part gran del programa.

Encara no tenim 100% tancat els graus de cada angle per cada moviment, ja que encara no em tancat al mil·límetre totes les posicions de l’entorn controlat. Però ja ho tenim iniciat, mostro foto a continuació d’esbós.



COdi molt senzill en c per arduino que transformaré a python, ja que el nostre programa el farem amb Rasberry-pi i faltarà també la part de connexió via wifi.

**CODI:**

#include <SoftwareSerial.h>

#include <Servo.h>

Servo servo01;

Servo servo02;

Servo servo03;

Servo servo04;

Servo servo05;

Servo servo06;

Software Bluetooth serie ( 3 , 4 ) ; // Arduino (RX, TX) - HC-05 Bluetooth (TX, RX)

int servo1Pos, servo2Pos, servo3Pos, servo4Pos, servo5Pos, servo6Pos; // posición actual

int servo1PPos, servo2PPos, servo3PPos, servo4PPos, servo5PPos, servo6PPos; // posición anterior

int servo01SP [ 50 ] , servo02SP [ 50 ] , servo03SP [ 50 ] , servo04SP [ 50 ] , servo05SP [ 50 ] , servo06SP [ 50 ] ; // para almacenar posiciones / pasos

int speedDelay = 20 ;

int index = 0 ;

String dataIn = "" ;

configuración nula ( ) {

servo01. adjuntar ( 5 ) ;

servo02. adjuntar ( 6 ) ;

servo03. adjuntar ( 7 ) ;

servo04. adjuntar ( 8 ) ;

servo05. adjuntar ( 9 ) ;

servo06. adjuntar ( 10 ) ;

Bluetooth. comenzar ( 38400 ) ; // Velocidad de transmisión predeterminada del módulo Bluetooth

Bluetooth. setTimeout ( 1 ) ;

retraso ( 20 ) ;

// Posición inicial del brazo del robot

servo1PPos = 90 ;

servo01. escribir ( servo1PPos ) ;

servo2PPos = 150 ;

servo02. escribir ( servo2PPos ) ;

servo3PPos = 35 ;

servo03. escribir ( servo3PPos ) ;

servo4PPos = 140 ;

servo04. escribir ( servo4PPos ) ;

servo5PPos = 85 ;

servo05. escribir ( servo5PPos ) ;

servo6PPos = 80 ;

servo06. escribir ( servo6PPos ) ;

}

bucle vacío ( ) {

// Verificar datos entrantes

if ( Bluetooth. disponible ( ) > 0 ) {

dataIn = Bluetooth. readString ( ) ; // Leer los datos como una cadena

// Si el control deslizante "Cintura" ha cambiado de valor, mueva el Servo 1 a la posición

if ( dataIn. comienza con ( "s1" ) ) {

Cadena dataInS = dataIn. substring ( 2 , dataIn. length ( ) ) ; // Extrae solo el número. Por ejemplo, de "s1120" a "120"

servo1Pos = dataInS. toInt ( ) ; // Convierte la cadena en entero

// Utilizamos bucles para poder controlar la velocidad del servo

// Si la posición anterior es más grande que la posición actual

if ( servo1PPos> servo1Pos ) {

for ( int j = servo1PPos; j> = servo1Pos; j-- ) { // Ejecutar servo hacia abajo

servo01. escribir ( j ) ;

retraso ( 20 ) ; // define la velocidad a la que gira el servo

}

}

// Si la posición anterior es menor, la posición actual

if ( servo1PPos <servo1Pos ) {

for ( int j = servo1PPos; j <= servo1Pos; j ++ ) { // Ejecutar servo arriba

servo01. escribir ( j ) ;

retraso ( 20 ) ;

}

}

servo1PPos = servo1Pos; // establece la posición actual como posición anterior

}

// Move Servo 2

if ( dataIn. comienza con ( "s2" ) ) {

Cadena dataInS = dataIn. substring ( 2 , dataIn. length ( ) ) ;

servo2Pos = dataInS. toInt ( ) ;

if ( servo2PPos> servo2Pos ) {

for ( int j = servo2PPos; j> = servo2Pos; j-- ) {

servo02. escribir ( j ) ;

retraso ( 50 ) ;

}

}

if ( servo2PPos <servo2Pos ) {

for ( int j = servo2PPos; j <= servo2Pos; j ++ ) {

servo02. escribir ( j ) ;

retraso ( 50 ) ;

}

}

servo2PPos = servo2Pos;

}

// Move Servo 3

if ( dataIn. comienza con ( "s3" ) ) {

Cadena dataInS = dataIn. substring ( 2 , dataIn. length ( ) ) ;

servo3Pos = dataInS. toInt ( ) ;

if ( servo3PPos> servo3Pos ) {

for ( int j = servo3PPos; j> = servo3Pos; j-- ) {

servo03. escribir ( j ) ;

retraso ( 30 ) ;

}

}

if ( servo3PPos <servo3Pos ) {

for ( int j = servo3PPos; j <= servo3Pos; j ++ ) {

servo03. escribir ( j ) ;

retraso ( 30 ) ;

}

}

servo3PPos = servo3Pos;

}

// Move Servo 4

if ( dataIn. comienza con ( "s4" ) ) {

Cadena dataInS = dataIn. substring ( 2 , dataIn. length ( ) ) ;

servo4Pos = dataInS. toInt ( ) ;

if ( servo4PPos> servo4Pos ) {

for ( int j = servo4PPos; j> = servo4Pos; j-- ) {

servo04. escribir ( j ) ;

retraso ( 30 ) ;

}

}

if ( servo4PPos <servo4Pos ) {

for ( int j = servo4PPos; j <= servo4Pos; j ++ ) {

servo04. escribir ( j ) ;

retraso ( 30 ) ;

}

}

servo4PPos = servo4Pos;

}

// Mover Servo 5

if ( dataIn. comienza con ( "s5" ) ) {

Cadena dataInS = dataIn. substring ( 2 , dataIn. length ( ) ) ;

servo5Pos = dataInS. toInt ( ) ;

if ( servo5PPos> servo5Pos ) {

for ( int j = servo5PPos; j> = servo5Pos; j-- ) {

servo05. escribir ( j ) ;

retraso ( 30 ) ;

}

}

if ( servo5PPos <servo5Pos ) {

for ( int j = servo5PPos; j <= servo5Pos; j ++ ) {

servo05. escribir ( j ) ;

retraso ( 30 ) ;

}

}

servo5PPos = servo5Pos;

}

// Mover Servo 6

if ( dataIn. comienza con ( "s6" ) ) {

Cadena dataInS = dataIn. substring ( 2 , dataIn. length ( ) ) ;

servo6Pos = dataInS. toInt ( ) ;

if ( servo6PPos> servo6Pos ) {

for ( int j = servo6PPos; j> = servo6Pos; j-- ) {

servo06. escribir ( j ) ;

retraso ( 30 ) ;

}

}

if ( servo6PPos <servo6Pos ) {

for ( int j = servo6PPos; j <= servo6Pos; j ++ ) {

servo06. escribir ( j ) ;

retraso ( 30 ) ;

}

}

servo6Pos = servo6Pos;

}

// Si se presiona el botón "GUARDAR"

if ( dataIn. comienza con ( "GUARDAR" ) ) {

servo01SP [ índice ] = servo1PPos; // guardar posición en la matriz

servo02SP [ índice ] = servo2PPos;

servo03SP [ índice ] = servo3PPos;

servo04SP [ índice ] = servo4PPos;

servo05SP [ índice ] = servo5PPos;

servo06SP [ índice ] = servo6PPos;

index ++; // Aumenta el índice de la matriz

}

// Si se presiona el botón "EJECUTAR"

if ( dataIn. comienza con ( " EJECUTAR " ) ) {

runservo ( ) ; // Modo automático: ejecuta los pasos guardados

}

// Si se presiona el botón "RESET"

if ( dataIn == "RESET" ) {

memset ( servo01SP, 0 , sizeof ( servo01SP ) ) ; // Borrar los datos de la matriz a 0

memset ( servo02SP, 0 , sizeof ( servo02SP ) ) ;

memset ( servo03SP, 0 , sizeof ( servo03SP ) ) ;

memset ( servo04SP, 0 , sizeof ( servo04SP ) ) ;

memset ( servo05SP, 0 , sizeof ( servo05SP ) ) ;

memset ( servo06SP, 0 , sizeof ( servo06SP ) ) ;

índice = 0 ; // Índice a 0

}

}

}

// Función personalizada en modo automático: ejecuta los pasos guardados

vacío runservo ( ) {

while ( dataIn! = "RESET" ) { // Ejecute los pasos una y otra vez hasta que presione el botón "RESET"

for ( int i = 0 ; i <= index - 2 ; i ++ ) { // Ejecute todos los pasos (index)

if ( Bluetooth. disponible ( ) > 0 ) { // Verificar datos entrantes

dataIn = Bluetooth. readString ( ) ;

if ( dataIn == "PAUSE" ) { // Si se presiona el botón "PAUSE"

while ( dataIn! = "RUN" ) { // Espere hasta que se presione nuevamente "RUN"

if ( Bluetooth. disponible ( ) > 0 ) {

dataIn = Bluetooth. readString ( ) ;

if ( dataIn == "RESET" ) {

romper ;

}

}

}

}

// Si se cambia el control deslizante de velocidad

if ( dataIn. comienza con ( "ss" ) ) {

Cadena dataInS = dataIn. substring ( 2 , dataIn. length ( ) ) ;

speedDelay = dataInS. toInt ( ) ; // Cambiar la velocidad del servo (tiempo de retraso)

}

}

// Servo 1

if ( servo01SP [ i ] == servo01SP [ i + 1 ] ) {

}

if ( servo01SP [ i ] > servo01SP [ i + 1 ] ) {

para ( int j = servo01SP [ i ] ; j> = servo01SP [ i + 1 ] ; j-- ) {

servo01. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

if ( servo01SP [ i ] <servo01SP [ i + 1 ] ) {

para ( int j = servo01SP [ i ] ; j <= servo01SP [ i + 1 ] ; j ++ ) {

servo01. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

// Servo 2

if ( servo02SP [ i ] == servo02SP [ i + 1 ] ) {

}

if ( servo02SP [ i ] > servo02SP [ i + 1 ] ) {

para ( int j = servo02SP [ i ] ; j> = servo02SP [ i + 1 ] ; j-- ) {

servo02. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

if ( servo02SP [ i ] <servo02SP [ i + 1 ] ) {

para ( int j = servo02SP [ i ] ; j <= servo02SP [ i + 1 ] ; j ++ ) {

servo02. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

// Servo 3

if ( servo03SP [ i ] == servo03SP [ i + 1 ] ) {

}

if ( servo03SP [ i ] > servo03SP [ i + 1 ] ) {

para ( int j = servo03SP [ i ] ; j> = servo03SP [ i + 1 ] ; j-- ) {

servo03. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

if ( servo03SP [ i ] <servo03SP [ i + 1 ] ) {

para ( int j = servo03SP [ i ] ; j <= servo03SP [ i + 1 ] ; j ++ ) {

servo03. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

// Servo 4

if ( servo04SP [ i ] == servo04SP [ i + 1 ] ) {

}

if ( servo04SP [ i ] > servo04SP [ i + 1 ] ) {

para ( int j = servo04SP [ i ] ; j> = servo04SP [ i + 1 ] ; j-- ) {

servo04. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

if ( servo04SP [ i ] <servo04SP [ i + 1 ] ) {

para ( int j = servo04SP [ i ] ; j <= servo04SP [ i + 1 ] ; j ++ ) {

servo04. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

// Servo 5

if ( servo05SP [ i ] == servo05SP [ i + 1 ] ) {

}

if ( servo05SP [ i ] > servo05SP [ i + 1 ] ) {

para ( int j = servo05SP [ i ] ; j> = servo05SP [ i + 1 ] ; j-- ) {

servo05. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

if ( servo05SP [ i ] <servo05SP [ i + 1 ] ) {

para ( int j = servo05SP [ i ] ; j <= servo05SP [ i + 1 ] ; j ++ ) {

servo05. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

// Servo 6

if ( servo06SP [ i ] == servo06SP [ i + 1 ] ) {

}

if ( servo06SP [ i ] > servo06SP [ i + 1 ] ) {

para ( int j = servo06SP [ i ] ; j> = servo06SP [ i + 1 ] ; j-- ) {

servo06. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

if ( servo06SP [ i ] <servo06SP [ i + 1 ] ) {

para ( int j = servo06SP [ i ] ; j <= servo06SP [ i + 1 ] ; j ++ ) {

servo06. escribir ( j ) ;

retraso ( speedDelay ) ;

}

}

}

}

}