Universidad Politécnica de la Zona metropolitana de Guadalajara

Ing. Mecatrónica Dinámica De robots

- **Flores Macias Cesar Fabian**
- **Gutiérrez Chávez Amaury Efraín**
- **Canales Ochoa Fabian**
- **Martínez Hernández Samuel Caleb**

Brazo SCADA

Contenido

Introducción

Damos el nombre de Scada (Supervisory Control And Data Acquisition o Control con Supervisión y Adquisición de Datos) a cualquier software que permita el acceso a datos remotos de un proceso y permita, utilizando las herramientas de comunicación necesarias en cada caso, el control del mismo.

Atendiendo a la definición vemos que no se trata de un sistema de control, sino de una utilidad software de monitorización o supervisión, que realiza la tarea de interfase entre los niveles de control (PLC) y los de gestión a un nivel superior.

- SCADA proviene de las siglas de Supervisor y Control And Data Acquisition (Adquisición de datos y supervisión de control).
- Es una aplicación software de control de producción, que se comunica con los dispositivos de campo y controla el proceso de forma automática desde la pantalla del ordenador.
- Proporciona información del proceso a diversos usuarios: operadores, supervisores de control de calidad, supervisión, mantenimiento, etc.
- Los sistemas de interfaz entre usuario y planta basados en paneles de control repletos de indicadores luminosos, instrumentos de medida y pulsadores, están siendo sustituidos por sistemas digitales que implementan el panel sobre la pantalla de un ordenador.
- El control directo lo realizarlos controladores autónomos digitales y/o autómatas programables y están conectados a un ordenador que realiza las funciones de diálogo con el operador, tratamiento de la información y control de la producción, utilizando el SCADA.

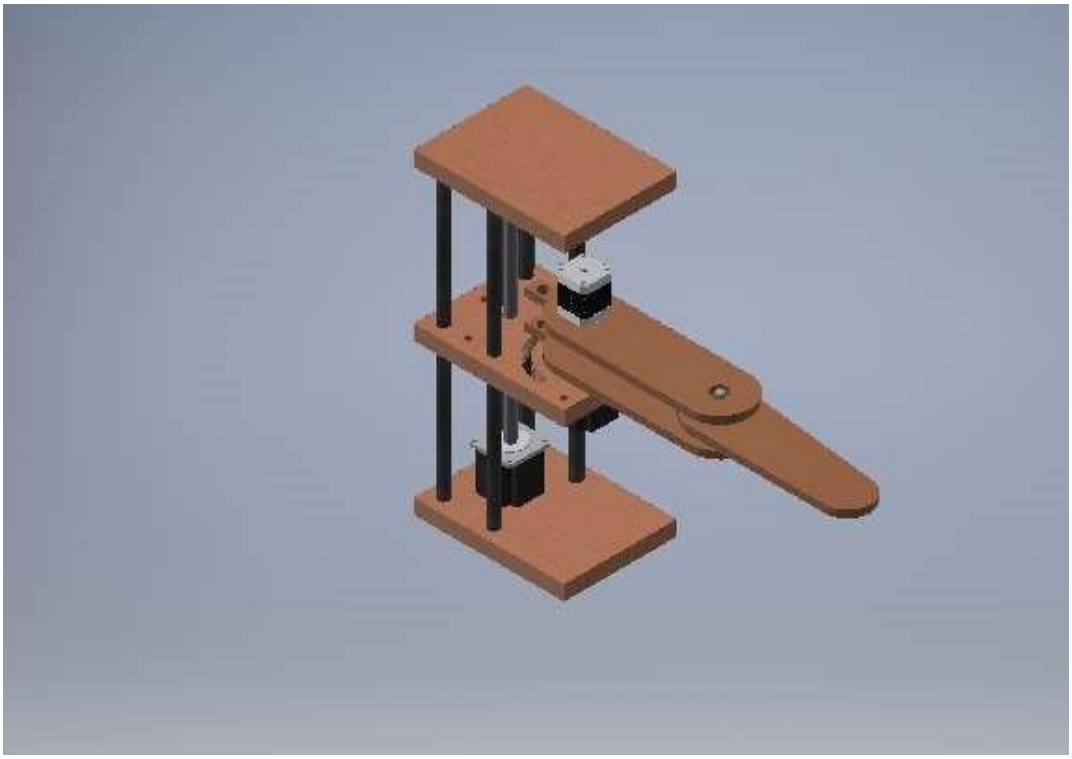
Diseño y objetivo

El brazo SCADA (remachadora) tuvo ciertos cambios con respecto al diseño anterior, este nuevo diseño tiene mejoras notables, entre ellas se destacan las siguientes.

- Mas espacio en la plataforma móvil
- Una mejor distribución de los motores de carga
- Una mayor estabilidad al momento de su construcción
- Mejor acomodo del cableado
- Posibilidades de actualización futura

No todo en el diseño es perfecto, siguieron ciertos detalles de fricción entre las barras estabilizadoras y las bases (no podían introducirse debido al diferente diámetro entre uno y otro), Unas de las cosas que se veden mejorar son las siguientes.

- Desvió del equilibro del motor principal (sube y baja la base) • Mejorar presentación visual (todo está correcto, únicamente pintarlo)
- Creación de polea móvil para el grado de libertad.



Materiales y controladores

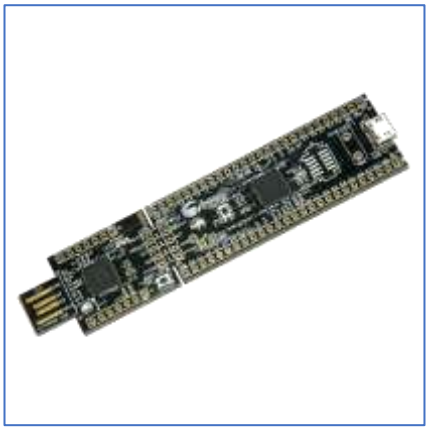
Una de las mayores problemáticas al momento de diseñar y crear un robot es el hecho de tener en cuenta las principales problemáticas que se pueden presentar, en este proceso se debe tener el claro es peso estimado del producto terminado y funcionando, así como los costos estimados de las piezas y herramientas que se utilizaran para darle vida a este tipo de proyectos, de una tarjeta raspberry pi hasta un motor a pasos bipolar, con controlador específico. Los sensores utilizados para este proyecto son:

- Sensores de posición
- Cámara de raspberry pi v2

Los cerebros lógicos a utilizar serán correspondientes a los tipos de sensores

- Raspberry pi 3b+
- Psoc 5Lp

- 2 motores nena 23 Bipolares
- 1 motor nema 17



Utilizando los códigos y usos de la visión artificial se decidió agregar un código con la librería OpneCV para que este pueda realizar la acción de hacer saber al robot donde es una superficie plana o un espacio vació, dependiendo de hacia a donde este apuntando el robot.

Para ello se utilizará el código siguiente, el cual tiene como función poder detectar la profundidad dependiendo del nivel de luz en el ambiente.

Inclusión de visión artificial

Codigo:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 31 23:12:05 2020

@author: sarah
"""

import cv2
import argparse
import sys
import math
import numpy as np

##########################################################################

keep_processing = True
selection_in_progress = False # Apoya a la seleccion de región interactiva
fullscreen = False #ejecutar en modo pantalla completa

# analizar argumentos de línea de comandos para ID de cámara o archivo de video.

parser = argparse.ArgumentParser(description='Perform ' + sys.argv[0] + ' example operation on incoming camera/video image')
parser.add_argument("-c", "--camera_to_use", type=int, help="specify camera to use", default=0)
parser.add_argument("-r", "--rescale", type=float, help="rescale image by this factor", default=1.0)
parser.add_argument('video_file', metavar='video_file', type=str, nargs='?', help='specify optional video file')
args = parser.parse_args()

##########################################################################

# seleccione una región usando el mouse
```

```
def on_mouse(event, x, y, flags, params):

    global boxes
    global selection_in_progress

    current_mouse_position[0] = x
    current_mouse_position[1] = y

    if event == cv2.EVENT_LBUTTONDOWN:
        boxes = []
        # imprimir "Iniciar posición del mouse:" + str(x)","+str(y)
        sbox = [x, y]
        selection_in_progress = True
        boxes.append(sbox)

    elif event == cv2.EVENT_LBUTTONUP:
        # imprimir "Finalizar posición del mouse:" +str(x)+","+str(y)
        ebox = [x, y]
        selection_in_progress = False
        boxes.append(ebox)
##########################################################################

# centro de retorno de un conjunto de puntos que representan un rectángulo

def center(points):
    x = np.float32((points[0][0] + points[1][0] + points[2][0] + points[3][0]) / 4.0)
    y = np.float32((points[0][1] + points[1][1] + points[2][1] + points[3][1]) / 4.0)
    return np.array([np.float32(x), np.float32(y)], np.float32)

##########################################################################
```

Este no es el código en su forma completa, es una muestra de como esta desarrollado el código que se utilizara y el resultado al momento de ejecutar este código es el siguiente.

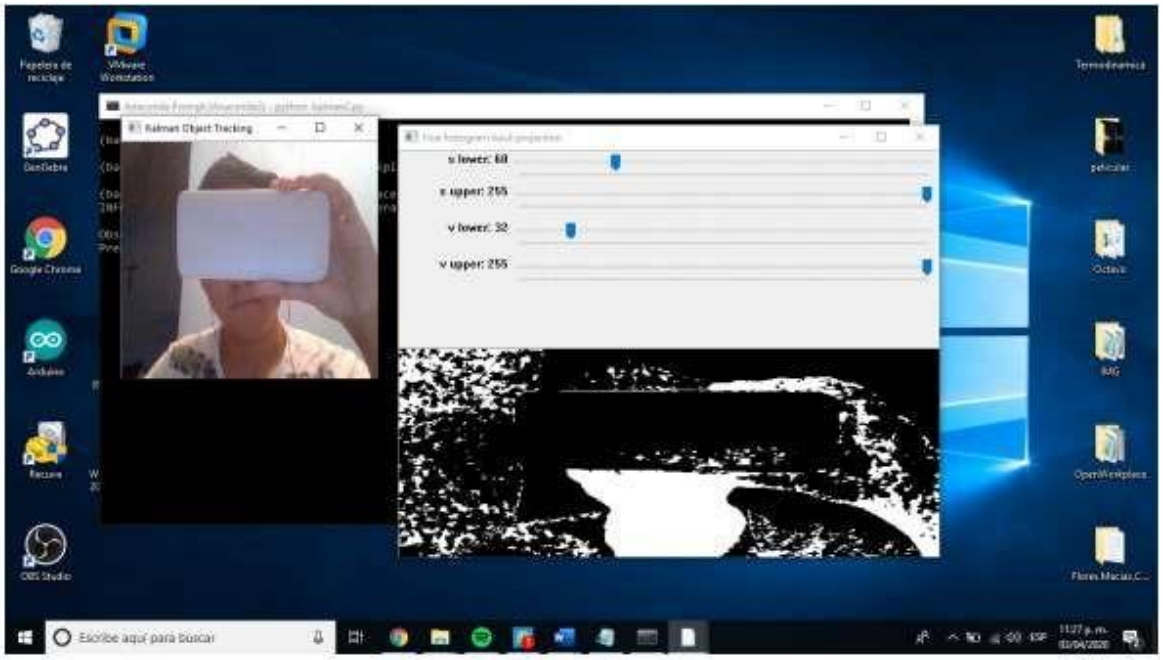


Imagen #1

Imagen #2

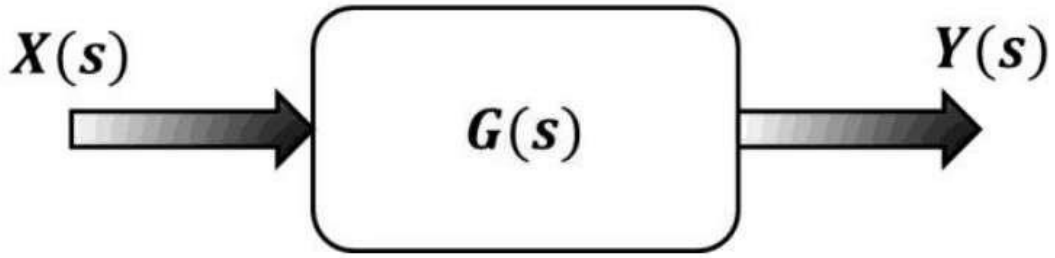
Al correr este código nos muestra una serie de 4 parámetros modificables, esto nos será de gran utilidad para poder configurar la cámara al nivel de luz en el que este trabajando el brazo. Por lo cual podrá trabajar en entornos de bajo nivel de iluminación, siempre y cuando se configure a un nivel estable estos parámetros del código.

Función de transferencia de los motores

La función de transferencia consiste en la relación matemática en el dominio de Laplace, que asocia la variable de salida de un sistema con la variable de entrada al mismo (Ogata, 2010).

$$G(s) = \frac{Y(s)}{X(s)}; \qquad (1)$$

$$\frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \ldots + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n}; \qquad (2)$$



Parámetro	*Relación matemática*
Frecuencia natural del sistema	ω_0
Factor de Amortiguamiento	$\zeta = \frac{1}{\sqrt{1 + \frac{\pi^2}{ln^2(M_p)}}};$
Frecuencia natural amortiguada	$\omega_d = \omega_0\sqrt{1-\zeta^2};$
Tiempo de levantamiento	$t_r = \frac{1}{\omega_d} arctan\left(\frac{-\omega_d}{\zeta\omega_0}\right);$
Tiempo pico	$t_p = \frac{\pi}{\omega_d};$
Sobre paso máximo	$M_p = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}};$
Tiempo de establecimiento (criterio del 2% y del 5%)	$t_s(2\%) = \frac{4}{\zeta\omega_0};$ $t_s(5\%) = \frac{3}{\zeta\omega_0};$

De acuerdo al principio de funcionamiento y a los detalles constructivos de una máquina eléctrica de CC, el Torque generado por la misma es directamente

proporcional a la corriente que circula por la armadura de la misma y al flujo magnético generado por el inductor, como muestra la siguiente expresión:

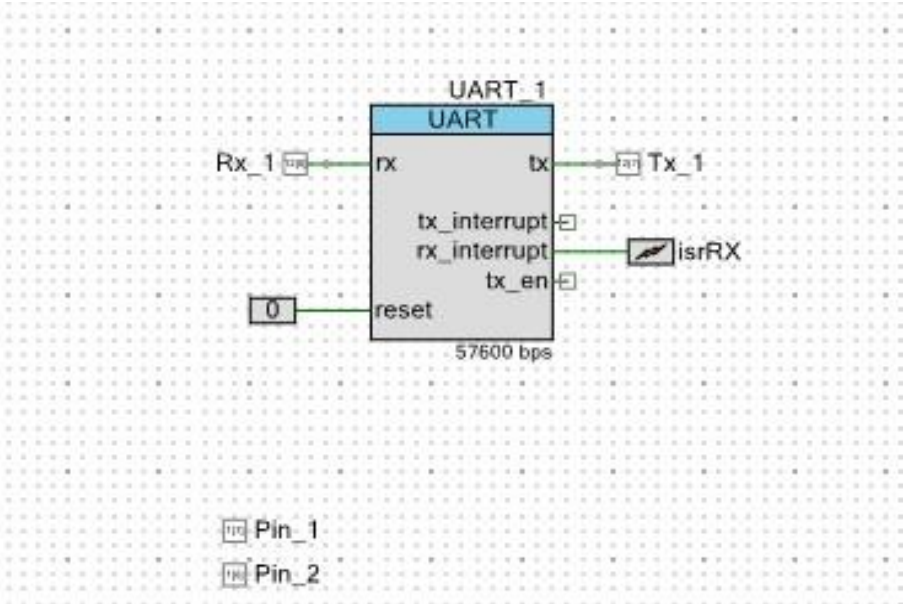
$$T(t) = \left(\frac{p}{a}\right).\frac{Z_a}{2\pi}.\Phi(t).i_a(t) = K_t.i_a(t)$$

donde:
p : número de pares de Polos de la máquina
a : número de pares de circuitos paralelos de la armadura
Z_a : número total de conductores en la armadura
La constante de proporcionalidad K_t, se denomina "Constante de Par del Motor", y depende de las características constructivas de la máquina y del flujo magnético generado por el campo del inductor.

Control de motores

En el caso del control del robot ya que no pudimos realizar el movimiento con “ros” sobre los motores optamos por utilizar la práctica UART el objetivo es hacer una comunicación entre la comunicación de la Psco con nuestros motores a pasos y la terminal PUTTY para lograr hacer que los motores giren 360 y -360 también que indique el proceso del motor. Según el movimiento del motor.

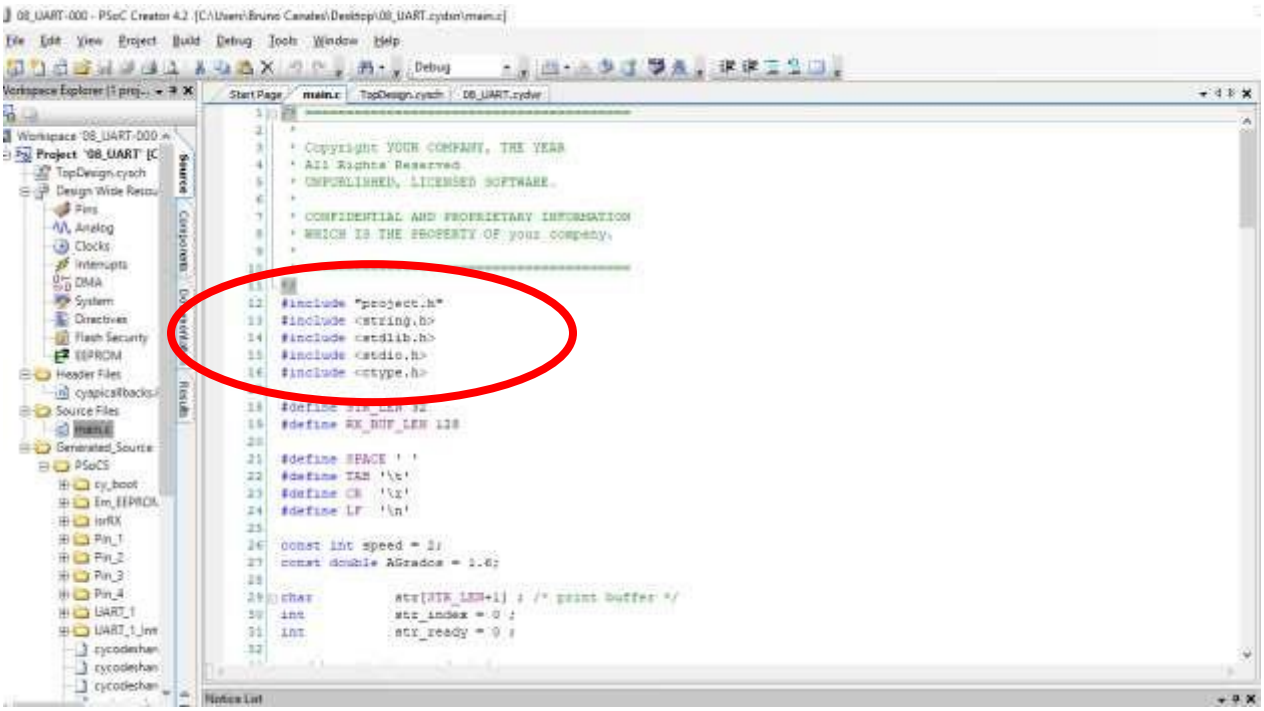
- La primera parte fue es realizar el diagrama para el puerto UART:



En este caso solo se necesitaron dos pines de salida ya que el motor va de la mano con un driver que controla el motor así que

solo se necesita un pulso y dirección por eso los dos pines. Esta es la dirección de los pines hacia mi psoc:

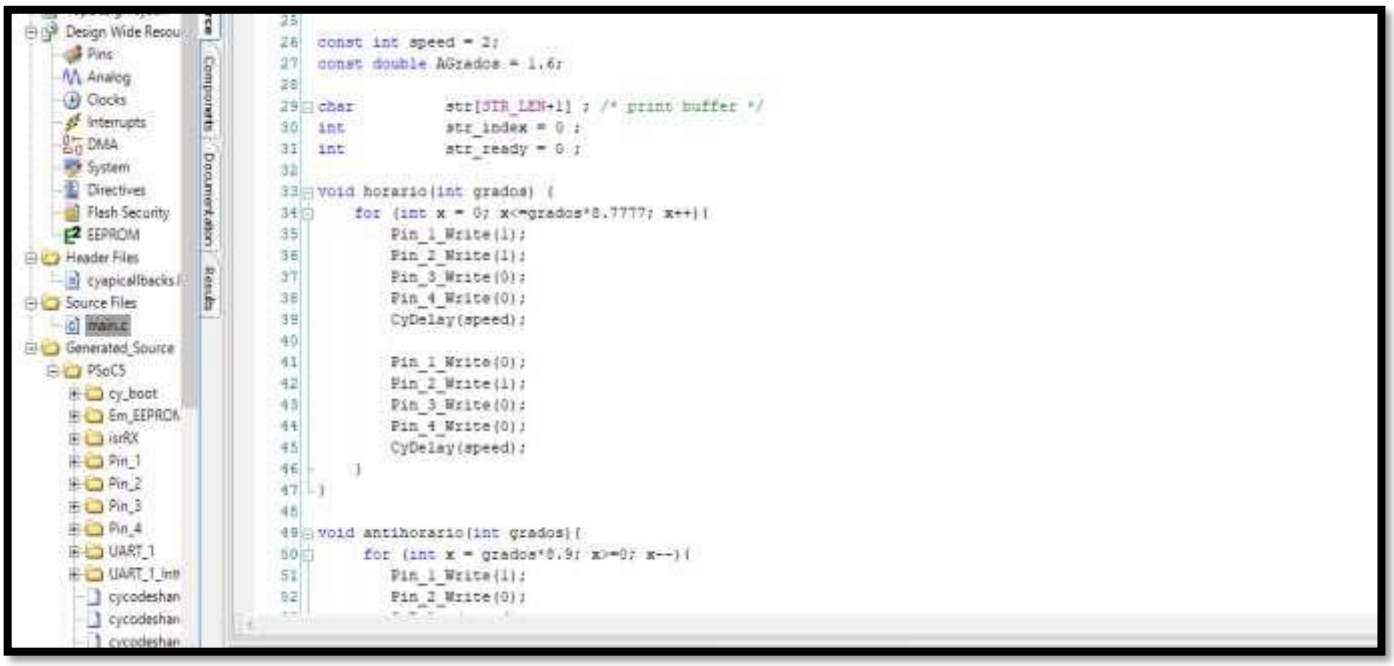
Después compilamos, ya que todo esté bien pasamos a la programación.



Esas son las librerías necesarias para poder hacer la comunicación de la psoc y putty.

Esta parte del programa es para controlar la velocidad y precisión del motor y darles control a los pines de salida:

Las siguientes imágenes forman parte del código que utilizamos para manipular los motores:



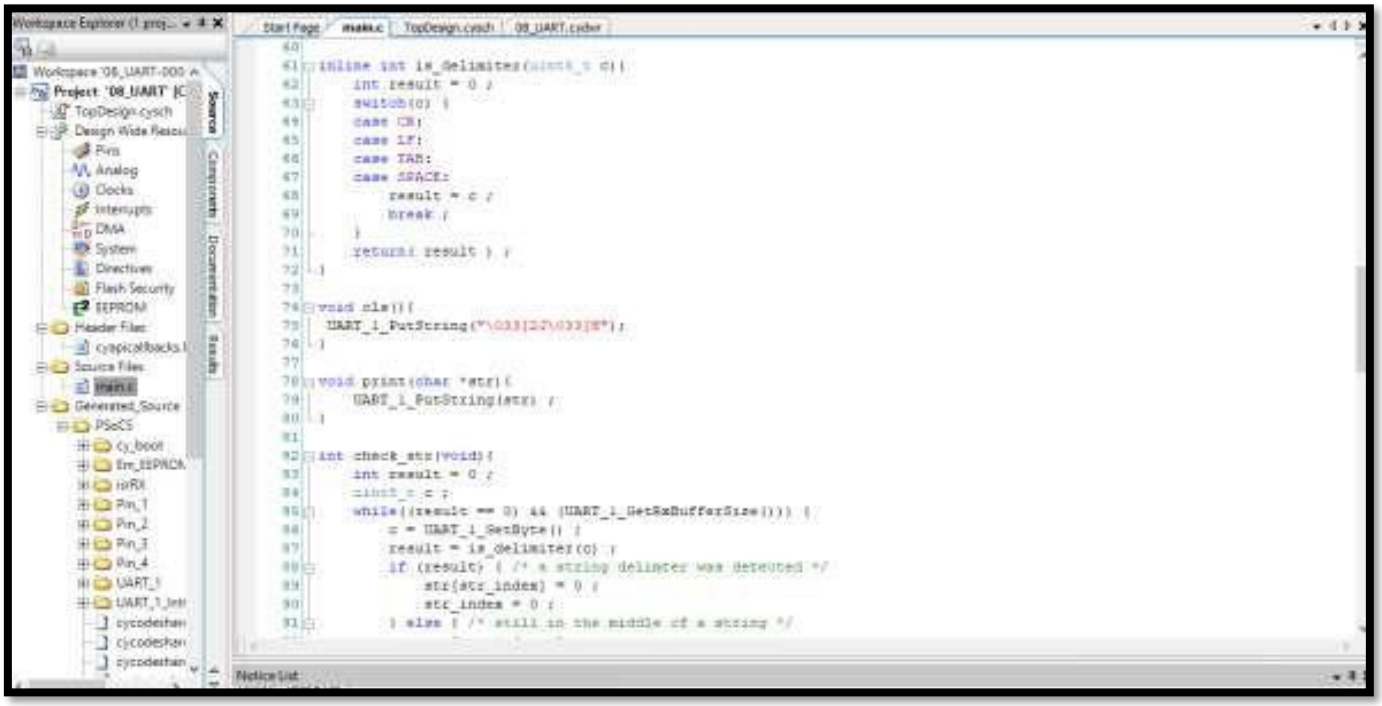
```
const int speed = 2;
const double AGrados = 1.6;

char          str[STR_LEN+1] ; /* print buffer */
int           str_index = 0 ;
int           str_ready = 0 ;

void horario(int grados) {
    for (int x = 0; x<=grados*8.7777; x++){
        Pin_1_Write(1);
        Pin_2_Write(1);
        Pin_3_Write(0);
        Pin_4_Write(0);
        CyDelay(speed);

        Pin_1_Write(0);
        Pin_2_Write(1);
        Pin_3_Write(0);
        Pin_4_Write(0);
        CyDelay(speed);
    }
}

void antihorario(int grados){
    for (int x = grados*0.9; x>=0; x--){
        Pin_1_Write(1);
        Pin_2_Write(0);
```



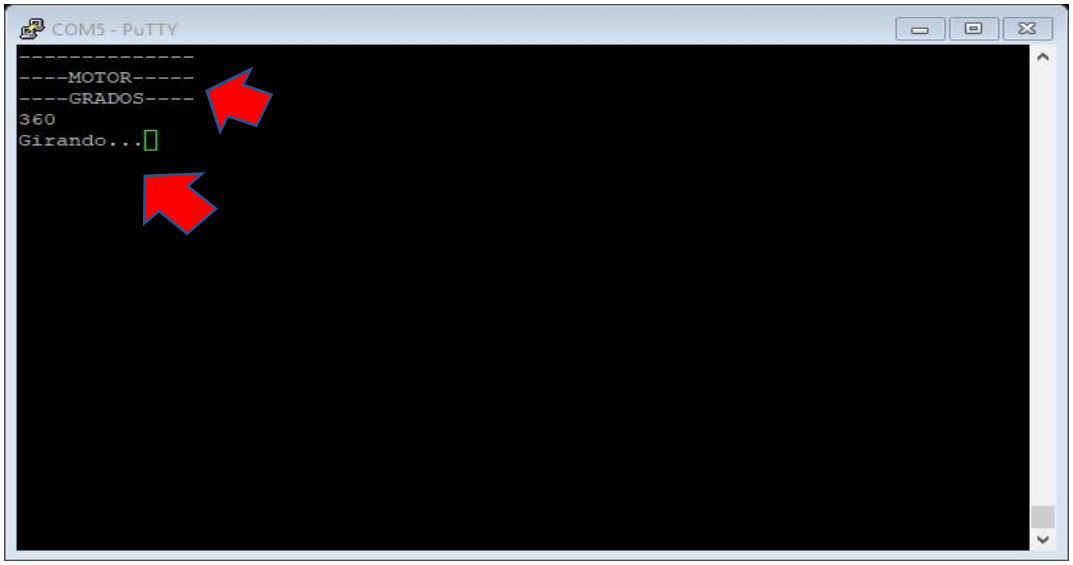
```
inline int is_delimiter(uint8_t c){
    int result = 0 ;
    switch(c) {
    case CR:
    case LF:
    case TAB:
    case SPACE:
        result = c ;
        break ;
    }
    return( result ) ;
}

void cls(){
  UART_1_PutString("\033[2J\033[H");
}

void print(char *str){
    UART_1_PutString(str) ;
}

int check_str(void){
    int result = 0 ;
    uint8_t c ;
    while((result == 0) && (UART_1_GetRxBufferSize())) {
        c = UART_1_GetByte() ;
        result = is_delimiter(c) ;
        if (result) { /* a string delimiter was detected */
            str[str_index] = 0 ;
            str_index = 0 ;
        } else { /* still in the middle of a string */
```

Como se muestra en la siguiente imagen, esta es la terminal putty y nos indica que es un motor grados y en este caso le pusimos 360 grados y nos indica que está realizando el proceso (Girando)



Y estas son fotos del circuito ya armado con motor a pasos:

Costos

- Monotes nema #23 $298
- Motor nema #17 $177
- Psoc 5Lp $210
- RaspBerry pi 3b+ $1500
- Cámara Raspberry v2 $989
- Sensores de Posición $20