UNIVERSIDAD POLITÉCNICA
DE LA ZONA METROPOLITANA DE GUADALAJARA

# *"Robot Cartesiano CNC"*


Fonseca Camarena Jonathan

Alvarado Contreras Cesar Omar

Manzo Torres Marcos

Robles Vázquez Eduardo

Tapia Casillas Víctor Gabriel

Universidad Politécnica de la Zona Metropolitana de Guadalajara

# Índice General:

# Introducción

## Meta:

Crear un robot cartesiano de cuatro grados de libertad, cuyo último eslabón deberá soportar una carga de 500 gramos y como aplicación útil podrá realizar cualquier actividad propia de una CNC fresadora.

## Objetivos:

- Diseñar estructura mecánica para el Robot Cartesiano.
- Desarrollar cálculos estructurales, simulación de funcionamiento del robot.
- Construir el robot cartesiano con 4 grados de libertad
- Agregar sensores a cada eje.
- Desarrollar la comunicación de Psoc 5lp con ROS para manejar el robot.
- Implementación de una cámara para el procesamiento de imágenes.

## Justificación:

El propósito de este proyecto surge a partir de la necesidad de implementar los conocimientos obtenidos de las materias presentes de estos últimos 2 cuatrimestres, así como de cuatrimestres pasados. Retomando lo mencionado con anterioridad, se desarrollará un robot cartesiano CNC, el cual consistirá de 4 grados de libertad.

## Marco Teórico:

Un robot de coordenadas cartesianas (también llamado robot cartesiano) es un robot industrial cuyos tres ejes principales de control son lineales (se mueven en línea recta en lugar de rotar) y forman ángulos rectos unos respecto de los otros. Además de otras características, esta configuración mecánica simplifica las ecuaciones en el control de los brazos robóticos. Los robots de coordenadas cartesianas con el eje horizontal limitado y apoyado en sus extremos se denominan robots pórtico y normalmente son bastante grandes.

Una aplicación muy extendida para este tipo de robots es la máquina de control numérico (CN). Las aplicaciones más sencillas son las usadas en las máquinas de fresado o dibujo, donde un taladro o pluma se traslada a lo largo de un plano x-y mientras la herramienta sube y baja sobre la superficie para crear un preciso diseño.

## Desarrollo

Tabla de materiales:

| Pieza: | Precio: |
|---|---|
| Soporte De Riel | $180 |
| Acopladores | $256 |
| Polea Dentada De Aluminio | $150 |
| Banda Correa Dentada | $150 |
| Bloque Deslizador Lineal | $369 |
| 4 motores A Pasos | $2400 |
| Tensores, Banda Y Coples | $521 |
| Aluminio Rectificado | $2000 |
| Mdf | $70 |
| Tornillos | $100 |
| Sensores Encoder | $200 |
| Total | $6396 |

Cronograma de actividades:

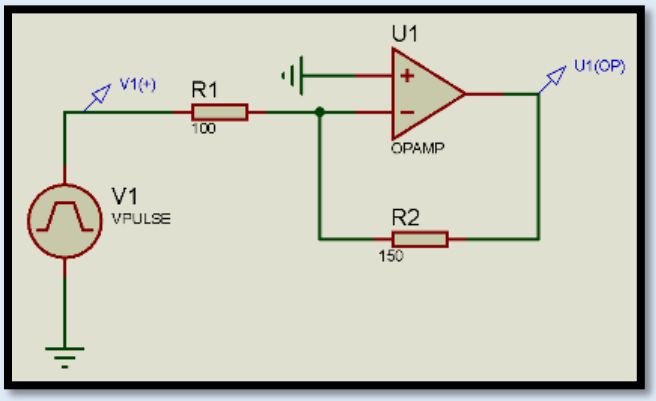
Cronograma_de_actividades.mpp

## Avances relacionados con las materias:

Ingeniería de Control:

En esta materia desarrollamos el control proporcional con ayuda de la PSoC 5Lp

Para desarrollar el controlador proporcional usaremos amplificadores operacionales y dependiendo del valor de nuestras resistencias dependerá la ganancia que obtendremos:




Donde la ganancia está dada por la fórmula:

$$\frac{V_{out}}{V_{in}} = -\frac{R_2}{R_1}$$

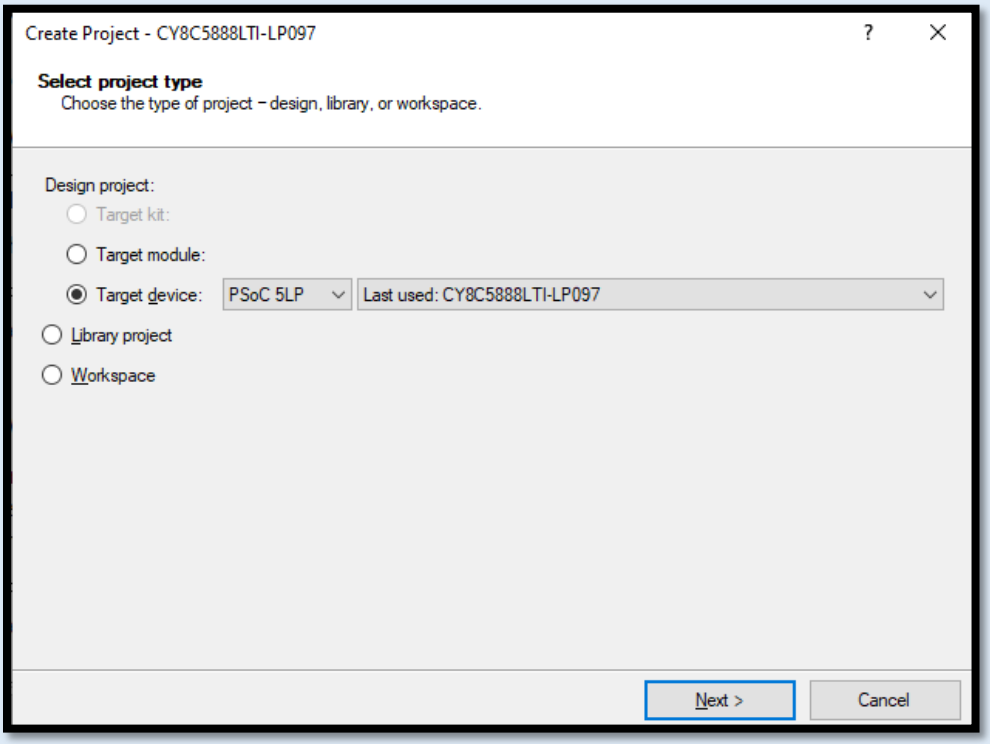
En nuestro primer caso por el valor de nuestras resistencias nuestra ganancia es de -1.5.

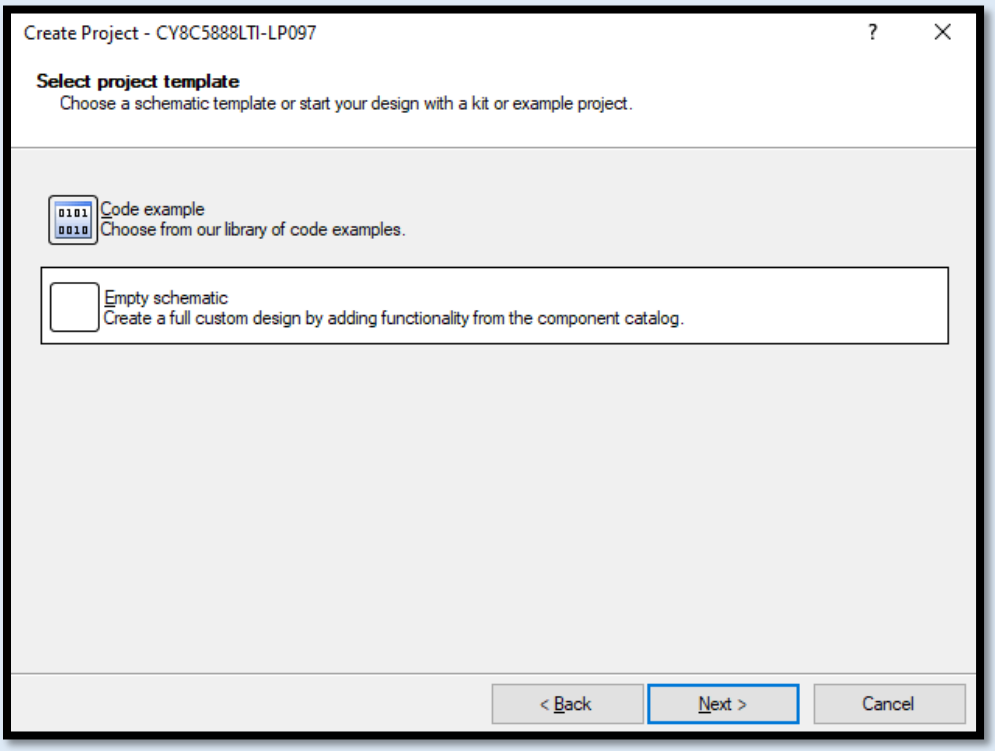
$$\frac{V_{out}}{V_{in}} = -\frac{R2}{R1} = -\frac{150}{100} = -1.5$$

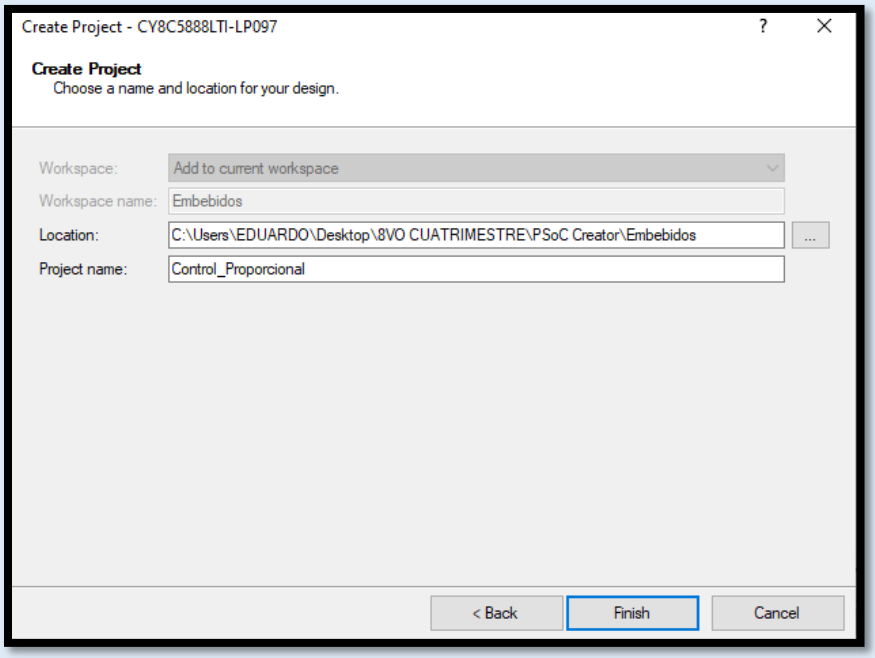
Como se puede observar en la gráfica:

Ahora procederemos a crear nuestra programación para simular el controlador en nuestra tarjeta PSoC 5 lp.

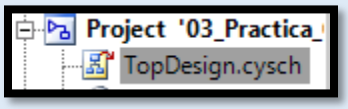
1. En el software PSoC Creator crear un nuevo proyecto con las características necesarias para programar la tarjeta CY8CKIT-059 PSoC.



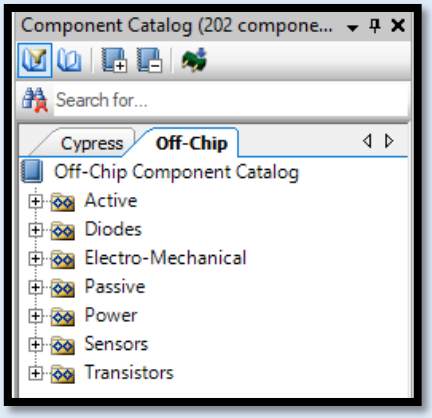



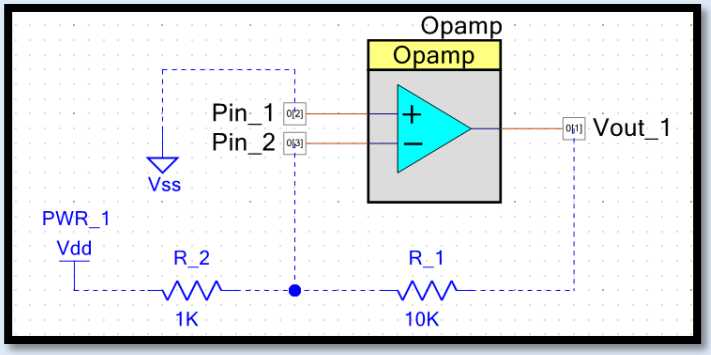



2. Insertar los componentes necesarios para su posterior uso. Una vez hecho esto se puedo construir el proyecto sin problema alguno.
    - Primero ingresamos a TopDesign a través del menú que se encuentra en la izquierda.

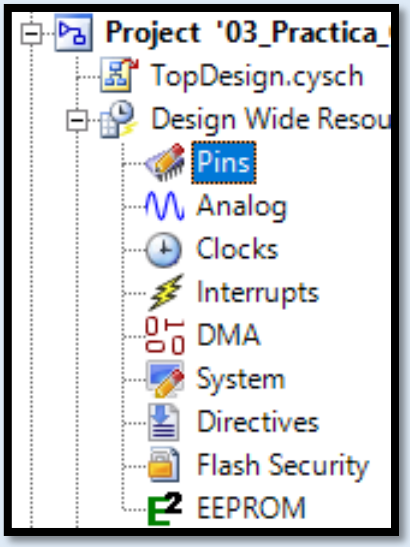


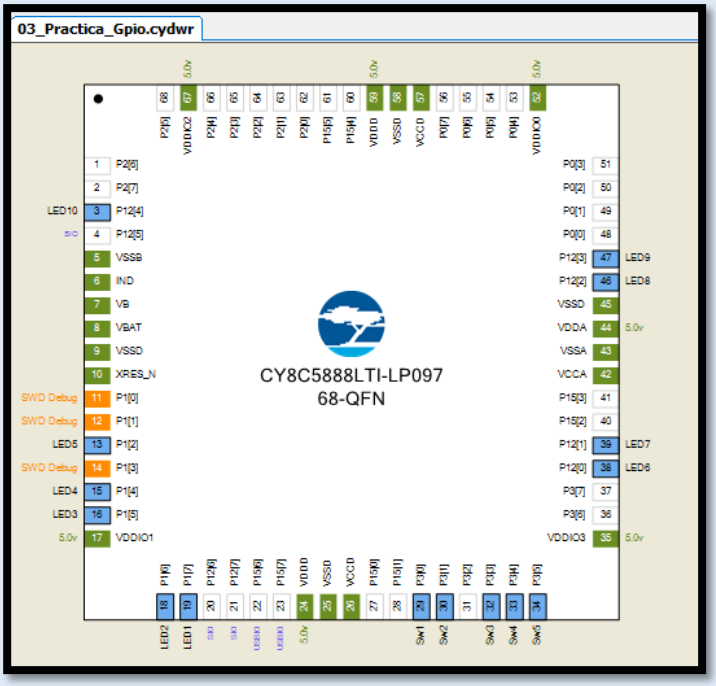

    - En el menú que se encuentra a la derecha podremos escoger los componentes que necesitemos.



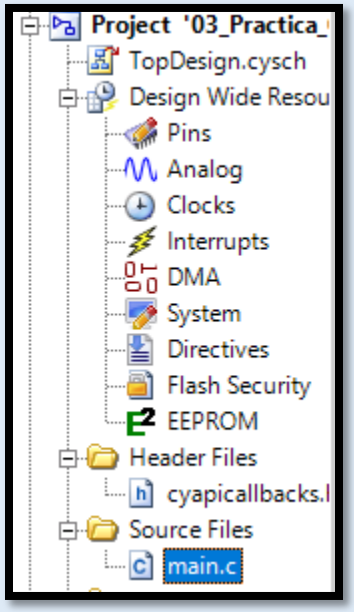
3. Definir los pines que usaremos para cada componente. Este lo haremos ingresando al menú dando clic en Pins.



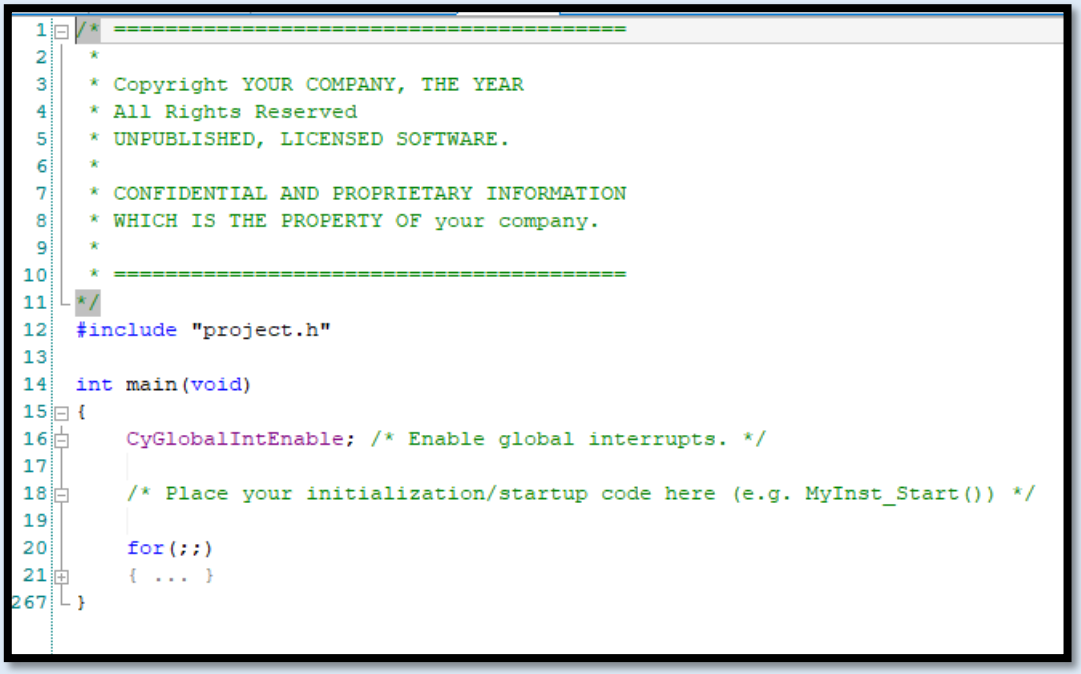





4. Procedemos a realizar la programación ingresando al Main.c



- Cuando ingresamos al main se nos muestra la siguiente ventana donde escribiremos nuestro programa dentro del For.



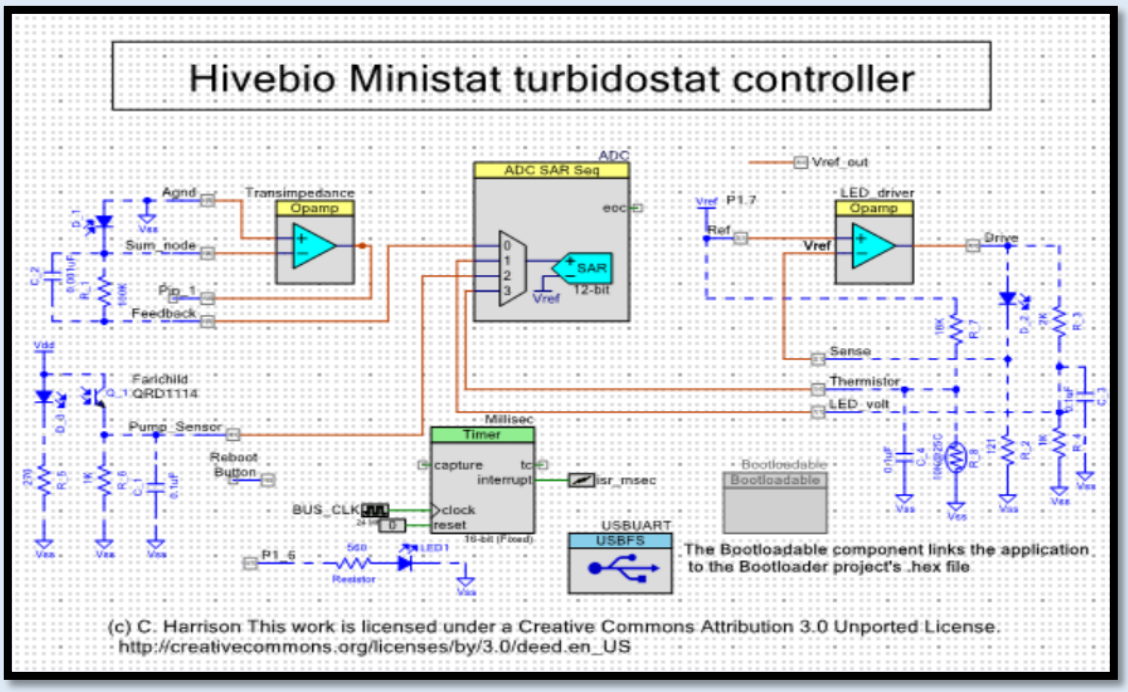
```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include "project.h"

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for(;;)
    { ... }
}
```

- A continuación, se muestra el código empleado. En nuestro caso solo debemos de activar el Opamp.

```
#include "project.h"

int main(void)
{
    Opamp_Start();
    CyGlobalIntEnable;

    for(;;)
    {

    }
}
```

Dinámica de Robots:

Actualmente nos encontramos en el proceso de lograr la comunicación de ROS con la tarjeta Psoc 5lp lo cual esta teniendo un avance lento puesto que tenemos algunos obstáculos con las librerías.




Sistemas de Visión Artificial:

Realización una investigación acerca de los métodos de segmentación de bordes y obtención de coordenadas de una imagen basada en sus pixeles. Prueba de códigos para la obtención de bordes de una imagen.

```
Archivo  Edición  Formato  Ver  Ayuda
import numpy as np
import cv2
import picamera
import io
from matplotlib import pyplot as plt
stream = io.BytesIO()
cam = cv2.VideoCapture(0)
with picamera.PiCamera() as camera:
    camera.resolution = (520, 440)
    camera.capture(stream, format='jpeg')#face_cascade = cv2.CascadeClassifier('/usr/share/opencv/haarcascades/haarcascade_frontalface_alt.xml')
buff = np.fromstring(stream.getvalue(), dtype=np.uint8)
image = cv2.imdecode(buff, 1)
gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 2)

sure_bg = cv2.dilate(opening,kernel,iterations=3)

dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)

sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_fg)


ret, markers = cv2.connectedComponents(sure_fg)

markers = markers+1

markers[unknown==255] = 0
markers = cv2.watershed(image,markers)
image[markers == -1] = [255,0,0]
cv2.imshow('imagen capturada',image)
cv2.waitKey(0)
cv2.imwrite('resultado.jpg',image)
cv2.destroyAllWindows()
```

Con la ayuda de OpenCv, procesaremos imágenes tomadas con la computadora, para después exportarlas a LaserGRBL con el cual trabajaremos para realizar un grabado laser.

```
7
8 import numpy as np
9 import cv2 as cv
10 from matplotlib import pyplot as plt
11
12 img = cv.imread('4.jpg',0)
13 edges = cv.Canny(img,100,200)
14
15 plt.subplot(121),plt.imshow(img,cmap = 'gray')
16 plt.title('Original Image'), plt.xticks([]), plt.yticks([])
17 plt.subplot(122),plt.imshow(edges,cmap = 'gray')
18 plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
19 cv.imshow("Image", edges)
20 cv.waitKey(0)
21 cv.destroyAllWindows()
22
23 plt.show()
```

Emplearemos el siguiente código llamado “bordes de cany” para recalcar solo los contornos de la imagen que serán los futuros trazos del laser para el grabado

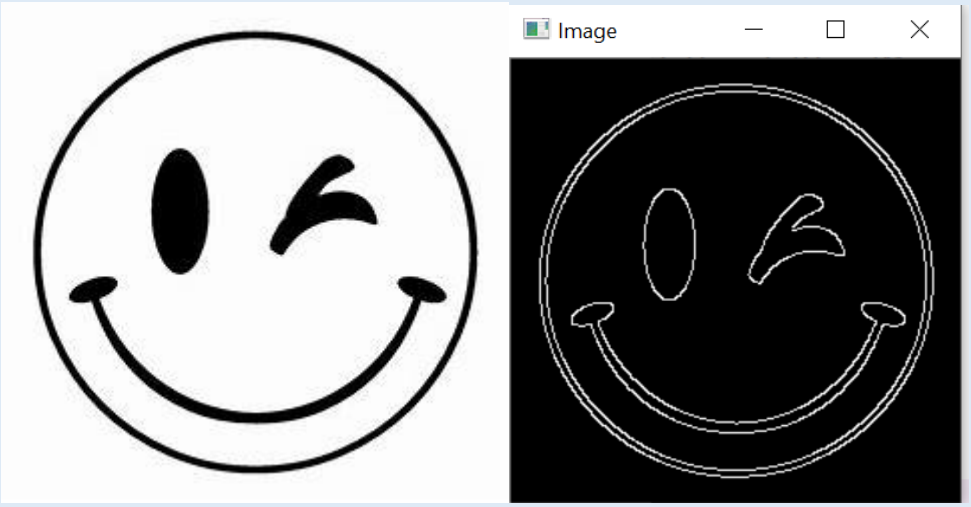
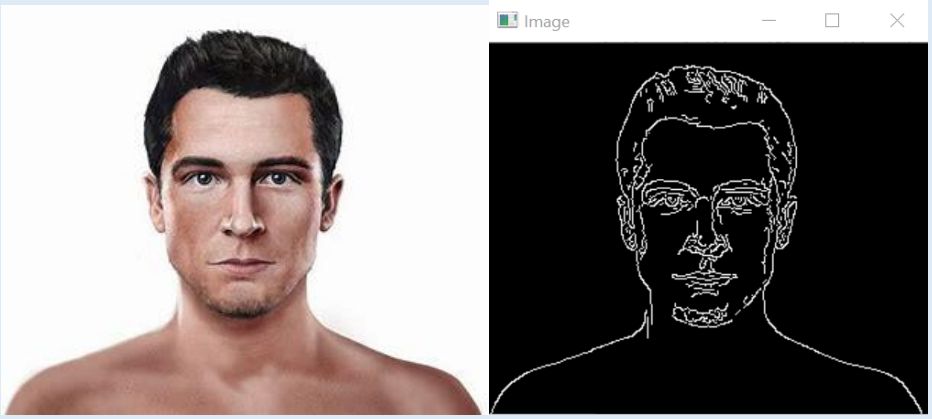
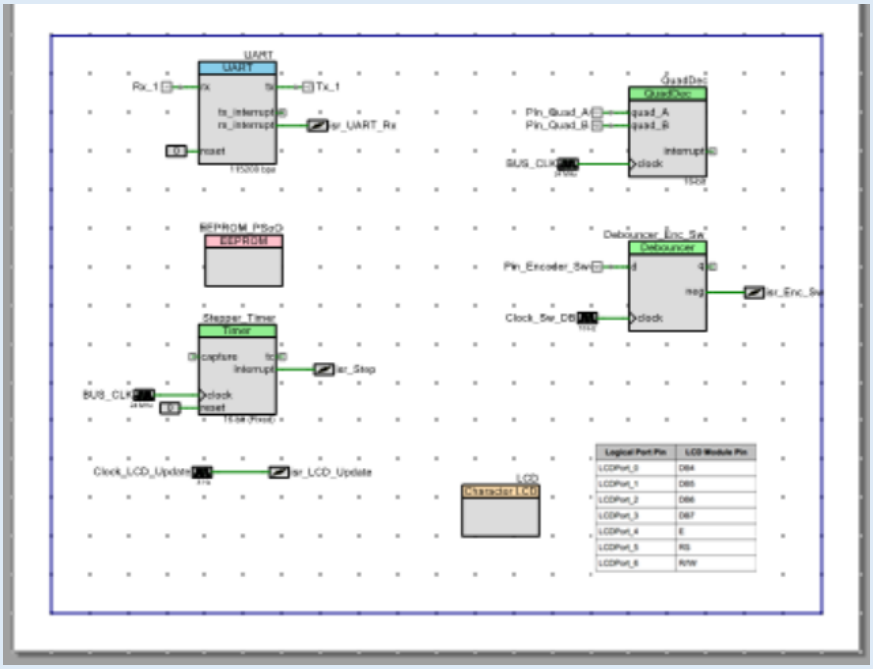


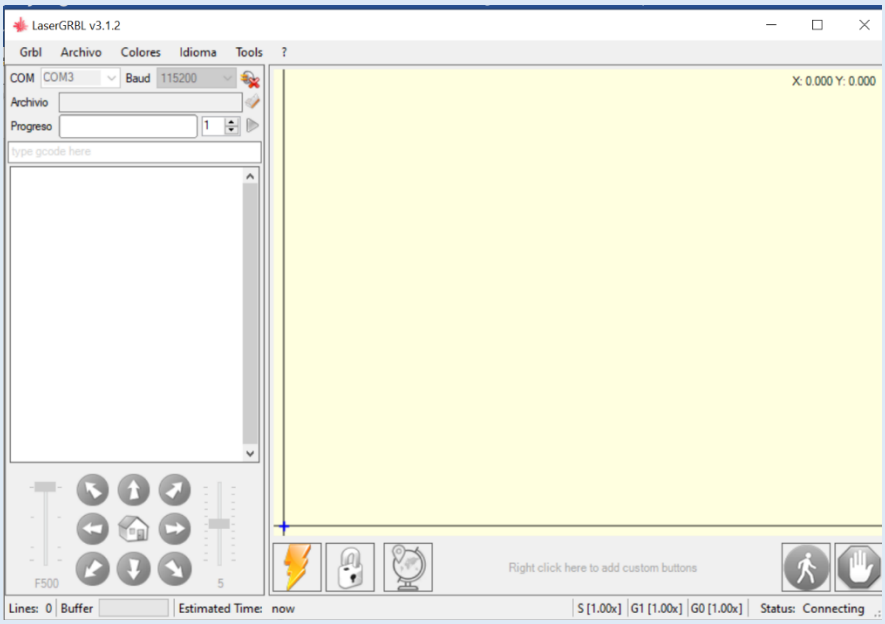

Imagen procesada



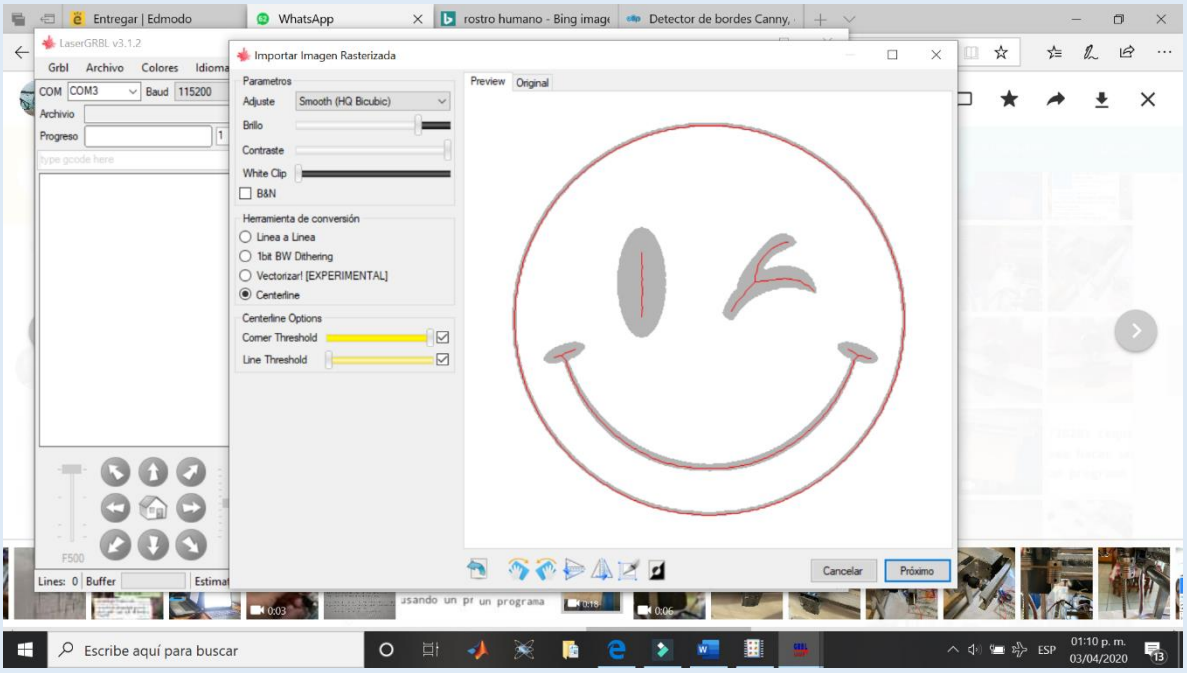
Utilizando la PSOC 5lp para realizar el control del robot, la conectamos a laserGRBL por puerto serial y realizamos el control de los motores de manera automática a la misma vez que usamos el láser que también tiene un control automático.

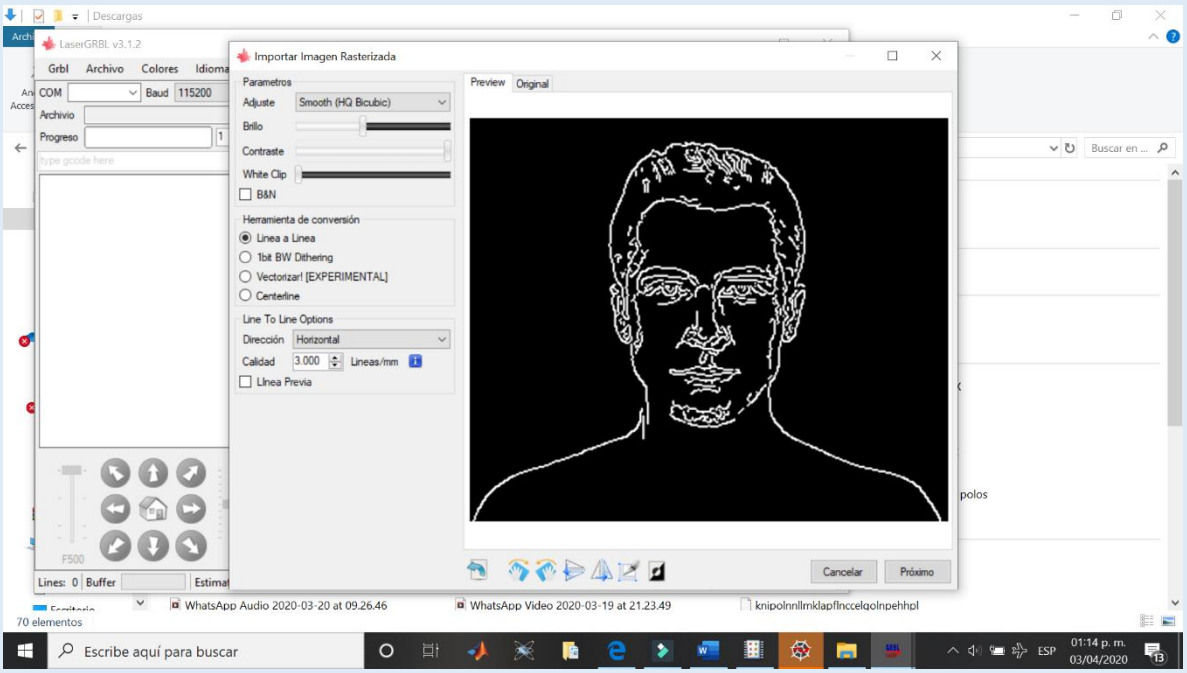



Observamos la conexión con el programa donde realizaremos el grabado laser y la tarjeta de control



Listo, podemos comenzar a trabajar en el grabado con el robot, usando opencv.



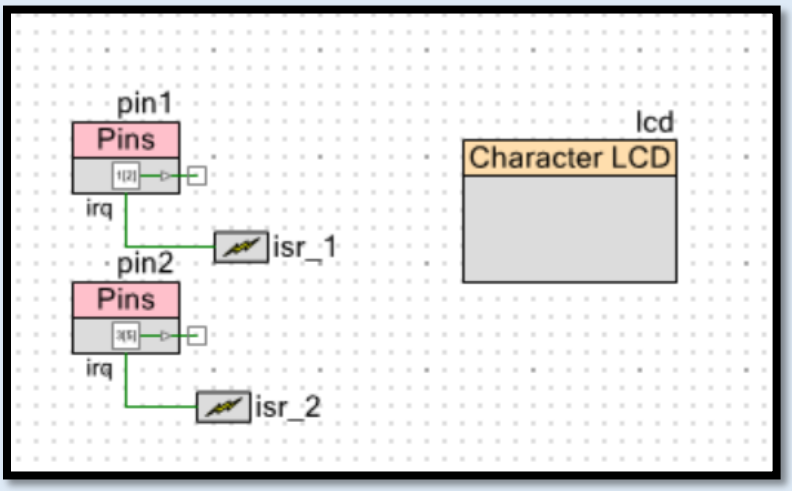



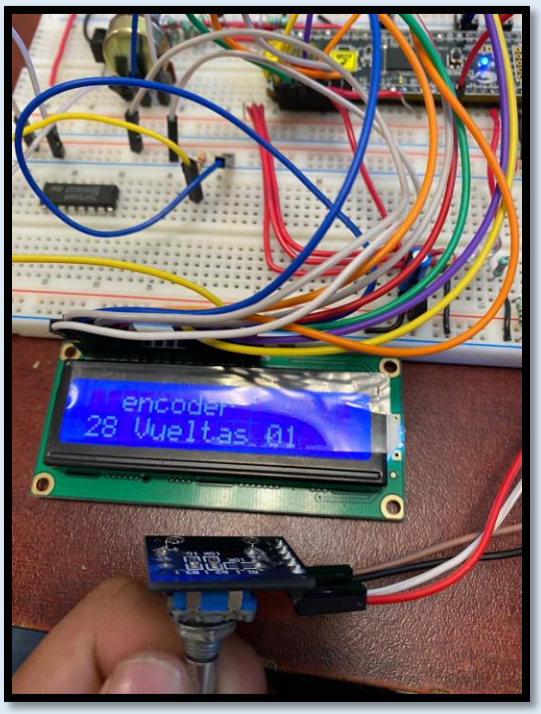

El siguiente link, contiene un video del funcionamiento del robot, solo restando el empleamiento de un láser que ya ha sido configurado y se representa con un led en color azul:

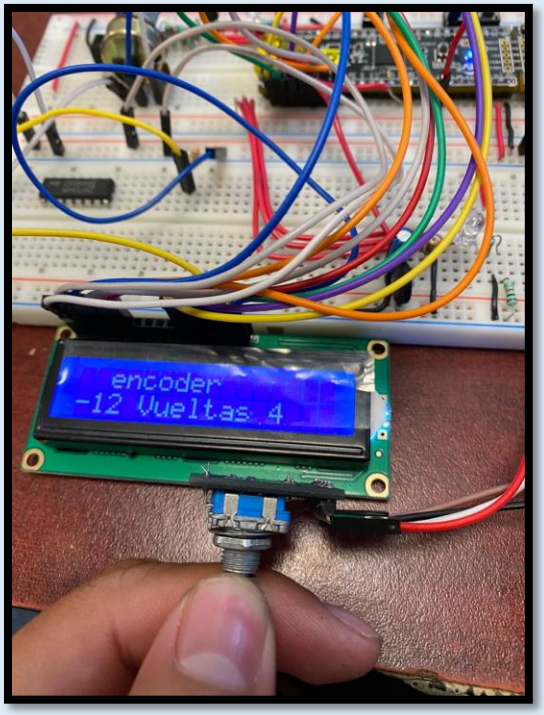
- https://youtu.be/Ir4ygJhyW9Y

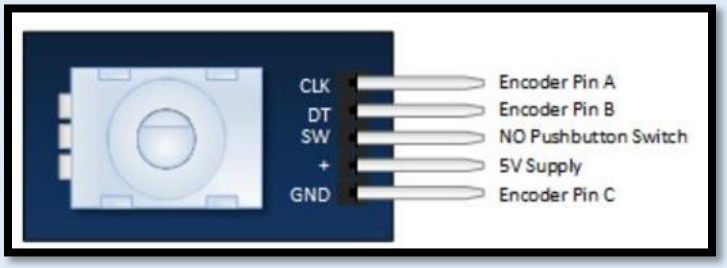
Programación de sistemas embebidos:

En esta materia hemos desarrollado un código con el cual programamos la tarjeta Psoc 5lp para poder utilizar los sensores Encoders:

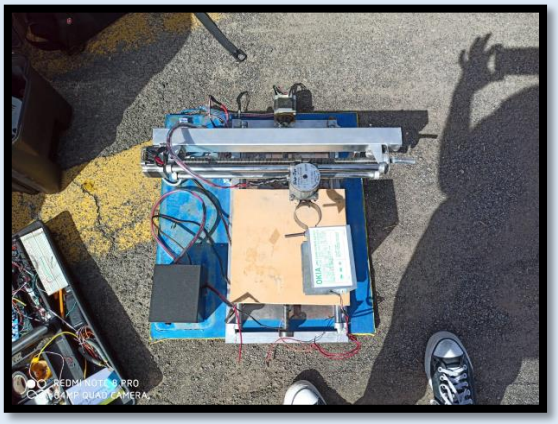


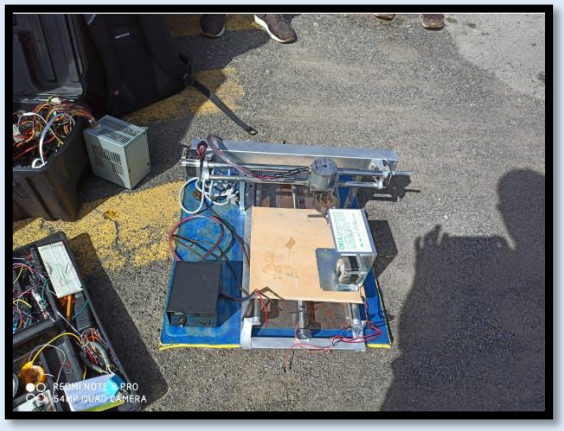



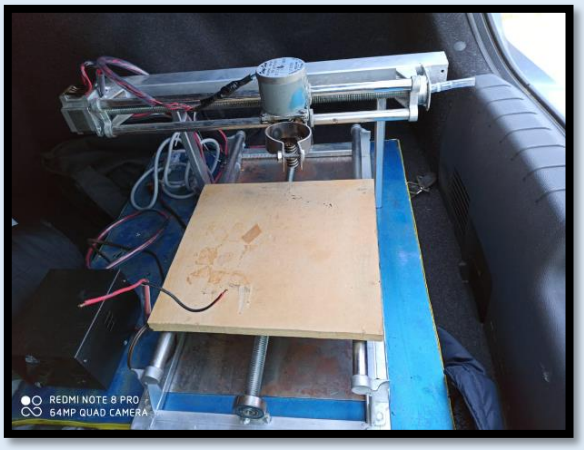



Resultados Fotográficos:

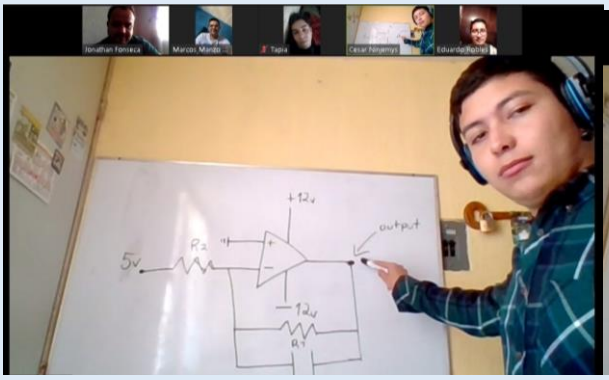


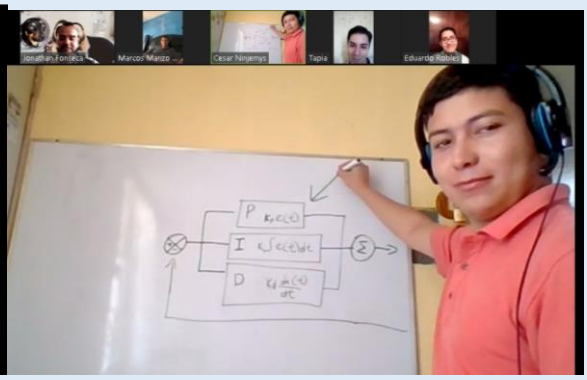








Conferencias en línea:



# Bibliografía:


- López, J. L. (s.f.). Robot Cartesiano. Recuperado 20 febrero, 2020, de https://www.uaeh.edu.mx/docencia/Tesis/icbi/licenciatura/documentos/Robot%20cartesiano%20seguimiento%20de%20trayectorias.pdf

- Gutiérrez, A. (2020, 24 enero). Modelado de un Motor. Recuperado 20 febrero, 2020, de http://www.robolabo.etsit.upm.es/asignaturas/seco/apuntes/modelado.pdf

- Jonathan Ruiz de Garibay Pascual. Robótica: Estado del arte. Universidad de Deuston. Numero. Fecha, page 54, 2006.

- E.F. Morales and L.E. Sucar. Los robots del futuro y su importancia para México. Komputer Sapiens,1(2):7–12, 2009.

- Pardo, C. (s.f.). Controlador PID - Control Automático - Picuino. Recuperado 7 abril, 2020, de https://www.picuino.com/es/arduprog/control-pid.html

- Villajulca, J. C. (2019, 22 junio). El control proporcional: definiciones prácticas y precisas – Instrumentación, Control y Automatización Industrial. Recuperado 7 abril, 2020, de https://instrumentacionycontrol.net/el-control-proporcional-definiciones-practicas-y-precisas/