UNIVERSIDAD AUTÓNOMA METROPOLITANA

AZCAPOTZALCO

DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

LICENCIATURA EN INGENIERÍA ELÉCTRONICA

PROYECTO TECNOLOGICO

Sistema de Medición y control de fuerza de tensado para Fibra Óptica de Vidrio.

TRIMESTRE 18-O

ALUMNO

Fabian Josafat Cárdenas Aldana

2112044123

fcardenasaldana@gmail.com

ASESOR

M. en C. José Luis Zamorano Flores

jlzf@azc.uam.mx

Declaratoria

Yo, José Luis Zamorano Flores, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Firma

_____

Yo, Fabian Josafat Cardenas Aldana, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Firma

_____

Tabla de contenido

Resumen

A continuación, se presenta el reporte final del proyecto de integración con nombre de “Sistema de Medición y control de fuerza de tensado para Fibra Óptica de Vidrio” donde se realizaron actividades para el funcionamiento óptimo de un aparato que realiza el tensado de la Fibra Óptica y se desarrolló un protocolo para su perfecta ejecución:

I. EL uso de una galga extensiométrica (no lineal) para medir la tensión de la fibra y uso de motores para crear dicha tensión.
II. El uso de un microcontrolador que pudiera satisfacer las necesidades del proyecto en este caso un PSOC5LP.
III. El uso del lenguaje C como un lenguaje de programación muy usado y que además permita tener contacto con la arquitectura de microcontrolador en cuestión.
IV. El uso de diversos módulos de hardware para hacer más simplificado el proyecto.
V. El uso del protocolo de comunicación RS485 para que este dispositivo pueda reducir considerablemente el factor de ruido cuando se requiere transferir datos al ordenador.

Este proyecto se desarrollo por que en la actualidad los sistemas con fibras ópticas han ganado terreno en el de las comunicaciones y las necesidades a tener mas herramientas para trabajar con fibras ópticas por eso es que se decidió construir este sistema. Ese se enfoca a cambiar las características de las fibras ópticas mediante el tensado de la misma. Además, este sistema se le agregaron mas herramientas para hacer más fácil la recolección de datos arrojados como resultado de la tensión que se ejerce en la fibra.

Este proyecto tiene la posibilidad de agregarle mas herramientas conforme el usuario las valla requiriendo ya que el microcontrolador que se uso tiene recursos de sobra para llevar a cabo las tareas adicionales que se le soliciten en un futuro.

Otro aspecto importante a mencionar sobre nuestro proyecto es su comunicación con otro tipo de sensores se ha elegido un bus485 que proporciona una robustez contra el ruido y además muchos de los sensores actuales utilizan este bus para comunicarse haciendo más fácil la comunicación y más sencilla la adaptación con nuestro proyecto.

Introducción

La tecnología de fibra óptica nos ha permitido desarrollar sistemas de comunicaciones con un gran ancho de banda y sistemas distribuidos de sensores. Para varios de estos sistemas es necesario el empleo de rejillas de Bragg. Las rejillas de Bragg es un dispositivo hecho con un tramo de fibra óptica, al cual, mediante algún método se le graba una secuencia de zonas de diferente índice de refracción este método tiene como medio base un fenómeno de fotosensibilidad y después de grabarlas se le hacer incidir un patrón de luz láser que comúnmente es ultravioleta sobre su núcleo de tal forma que cuando la luz incide dentro de ella lo hace perpendicularmente sobre las películas que forman la rejilla. La figura 1 muestra el esquema de una rejilla de Bragg.

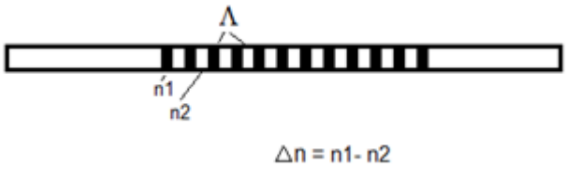

Figura 1:Rejilla de Bragg

Entre las zonas con diferente índice de refracción existe una diferencia $\Delta n = n1 - n2$ y un paso Λ de estos dos parámetros depende la propagación de la luz a través de la rejilla, si la luz que ingresa consiste de varias longitudes de onda λ algunas de ellas serán reflejadas y otras la atravesarán, es decir actúa como un filtro óptico. Repitiendo, el rango de longitudes de onda que pasan a través de la rejilla es dependiente de los parámetros Λ y Δn, entonces si estos son modificados, la rejilla será selectiva en otro rango diferente de longitudes de onda. En una rejilla, el valor de Δn ya no lo podemos cambiar, pero si el de Λ. Existen varias maneras de modificar Λ entre las más conocidas, están la de someter la fibra a cambios de temperatura o tensionando la fibra. Por lo que podemos lograr un filtro sintonizable estirando la rejilla. Este proyecto se enfocará en la segunda forma de modificar estos parámetros, así que se propone el diseño y construcción de un dispositivo que estire, controlando y midiendo la fuerza aplicada a una rejilla de Bragg hecha sobre fibra óptica.

Antecedentes

Un sistema como el propuesto en este proyecto hasta ahora no lo hemos encontrado en el mercado, en la UAM Azcapotzalco en 2017 se ha hecho un sistema parecido que tiene como título “Elongation-based fiber optic tunable filter” [4], que solo tensa la fibra, haciéndolo sin ningún tipo de retroalimentación y mucho menos mide la fuerza aplicada. Otra limitación del sistema mencionado es que no cuenta con un medio de comunicación que permita enviar los datos obtenidos hacia otro sistema que en algún momento ayude a automatizar las tareas a realizar.

Justificación

Un sistema como el propuesto en este proyecto es necesario para la caracterización de dispositivos hechos de fibra óptica, en particular se usará para obtener la respuesta espectral de rejillas de Bragg sujetas a fuerzas de tensado. Dado que no existen sistemas similares en el mercado, es necesario el desarrollo de un sistema como el propuesto.

Objetivos Generales

Diseñar e implementar un instrumento que estire y controle y mida la fuerza de tensión aplicada a una fibra óptica de vidrio, con el fin de obtener las características de propagación en rejillas de Bragg sometidas a fuerzas de tensión.

Objetivos Particulares

- Diseñar y construir un aparato que estire una fibra óptica de tal manera que no afecte la geometría de la fibra más que la debida al estiramiento.
- Diseñar y construir un instrumento de medición y control de la fuerza de estiramiento del aparato anteriormente mencionado.
- Diseñar e implementar una interfaz que permita la transferencia de datos de los datos de medición de fuerza a través de un bus 485.

Marco Teórico

Fibras ópticas

La baja perdida de las fibras ópticas se debe a las propiedades fortuitas del material. El índice de refracción de una fibra óptica se muestra en la figura 1. La región del núcleo tiene un índice de refracción más alto que el material de revestimiento circundante, que por lo regular está hecho de sílice. Esto hace que la luz queda atrapada en el núcleo por la reflexión interna total en los límites del revestimiento del núcleo y puede viajar decenas de kilómetros con poca atenuación en la región de longitud de onda de 1550 nm [6].

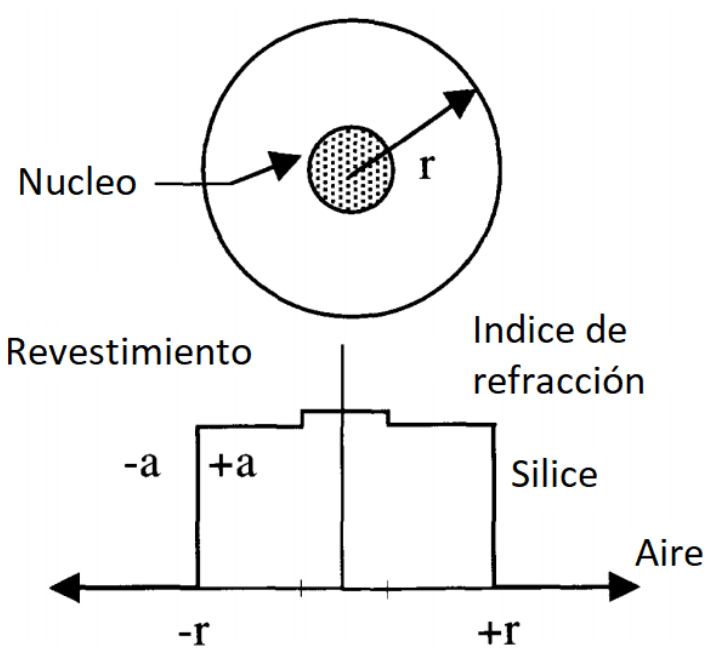

Figura 2: Sección transversal de una fibra óptica con el perfil de índice de refracción correspondiente. Por lo general, la diferencia del índice de refracción de núcleo a revestimiento para la fibra de telecomunicaciones monomodo a una longitud de onda de $1.5\ \mu m$ es $\sim 4.5\ X\ 10^{-3}$ con un radio de núcleo de $4\ \mu m$.

Desarrollo del Proyecto

Nuestro proyecto se hizo de un una base que soporta un motor a pasos que hizo girar una polea de al menos 7 cm de diámetro, suficientes para no alterar la propagación óptica. El motor fue accionado por un microcontrolador que a su vez procesa la señal de retroalimentación (medición de la fuerza de estiramiento) con el fin de mantener un valor de fuerza de estiramiento previamente programado. En esta misma figura se observa una celda de carga (se ha considerado que responde de forma no lineal), la cual generalmente se instala en forma de voladizo, pero en este caso se instaló verticalmente de tal manera que al estirar la fibra ésta se flexiono. Al flexionarse, se deforma una galga extensiométrica incluida en la celda e instalada en un puente de Wheatstone cuya respuesta es un pequeño voltaje que

fue acondicionado por el amplificador de instrumentación y este fue ingresado a un convertidor analógico digital y enviado a un microcontrolador, para que el valor correspondiente a la fuerza de estiramiento se digitalice, escale, y compare con el valor requerido. En la siguiente figura se muestra nuestro esquema de medición.

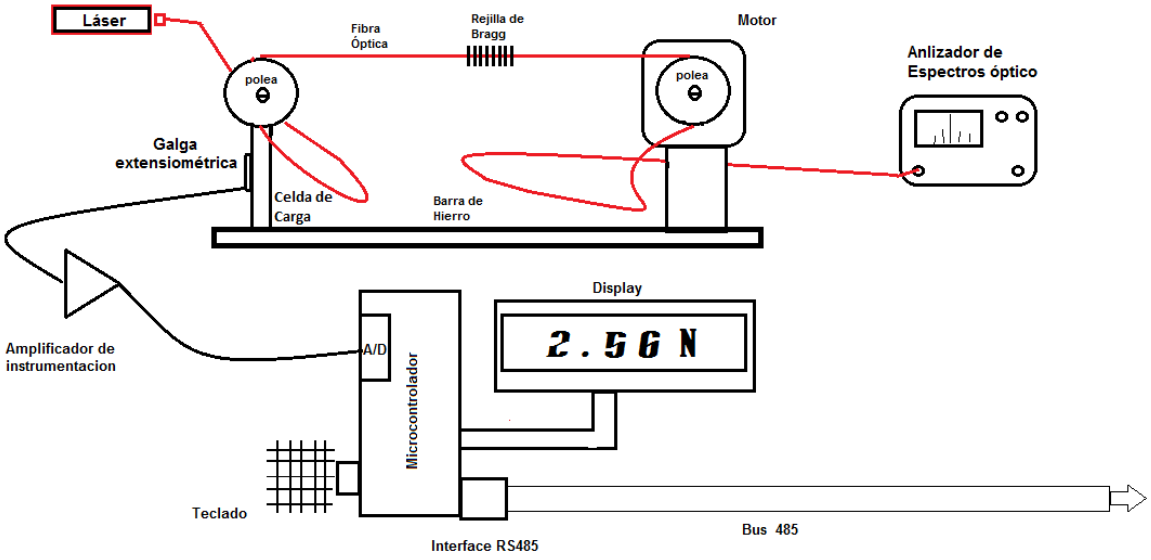

Figura 3: Esquema del Sistema de Medición y Control de fuerza de tensado para Fibra Óptica de Vidrio

Ahora se mostrará el diagrama a bloques del sistema de control y medición de fuerza de tensado para fibra óptica, en este se muestran con mayor detalle cada uno de los elementos electrónicos que forman parte del sistema.

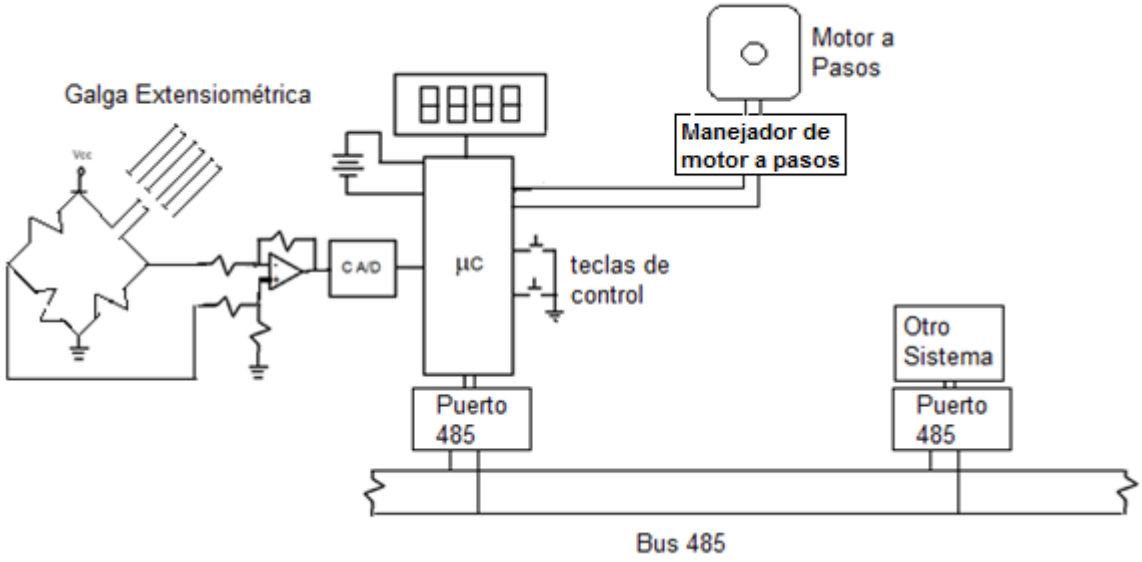

Figura 4: Diagrama a bloques del Sistema de Medición y control de fuerza de tensado para Fibra Óptica de vidrio.

La metodología utilizada para la ejecución del presente proyecto sigue una ordenada secuencia entre las que destacan:

- La elección de un microcontrolador que pueda realizar las tareas que requiere la aplicación.
- La elección de los módulos que usara el microcontrolador para la aplicación
- El desarrollo de un programa en el microcontrolador que permita realizar todas las tareas requeridas.
- Calibración y colocación de una galga extensiométrica junto con un motor paso a paso.

La elección de un microcontrolador que pueda realizar las tareas que requiere la aplicación.

En la propuesta de esta aplicación se propuso hacerlo con una MSP430G2553, pero debido a su limitación en memoria se tuvo que cambiar por otro microcontrolador más sofisticado y el que se eligió fue PSoC® 5LP.

PSoC® 5LP

La tarjeta de desarrollo PSOC 5 Prototyping Kit contiene un dispositivo del fabricante de semiconductores Cypress el cual es llamado PSOC sistema programable en un chip el cual combina periféricos analógicos y lógica digital programable basada en CPLD con un microcontrolador ARM Córtex M3 algunas especificaciones de este dispositivo se muestran a continuación.

- 32-bit ARM Cortex-M3 CPU.
- 32 Entradas de interrupción.
- 24 canales de acceso directo a memoria.
- Procesador para filtro digital de 24 bits de punto fijo.
- 24 universal digital blocks (UDB).
- 4 convertidores digital a analógico de 8 bits.
- 4 amplificadores operacionales.
- 4 bloques analógicos programables.
- Etc.

Como se puede observar el dispositivo se adapta perfectamente para nuestro propósito y además se requiere un alto conocimiento del entorno de desarrollo.

La tarjeta de Cypress semiconductor ofrece la flexibilidad de contar con bloques programables digitales y analógicos además de contar con un hardware especializado para DSP por si lo requerimos. Entonces por lo anteriormente dicho

se optó por seleccionar la tarjeta PSOC 5 Prototyping Kit como la plataforma de desarrollo para nuestro proyecto.

La elección de los módulos que usara el microcontrolador para la aplicación

Para esta aplicación se usaron varios integrados para llevar a cabo su correcto funcionamiento se hace un listado de los integrados y algunos componentes usados en este proyecto

- Display LCD 20x4.
- PCF8574.
- MM74C922.
- MAX485.
- A4988.
- Celda de Carga.
- HX711.
- Motor paso a paso.

Display LCD 20x4
Los LCD Alfanuméricos más utilizados en el mercado son el LCD1602 y LCD2004 con tamaños de 16x2 y 20x4. Entre estos tamaños hay diferentes modelos los cuales varían en color y sobre todo en presencia un Backlight o no.

Figura 5: Display 20x4

PCF8574 I2C
Para empezar a describir este integrado primero se tiene que describir que es el protocolo de comunicación I2C.

I2C es un puerto y protocolo de comunicación serial, define la trama de datos y conexiones físicas para transferir bits entre 2 dispositivos digitales. El puerto incluye dos cables de comunicación, SDA y SCL. Además, el protocolo permite conectar hasta 127 dispositivos esclavos con esas dos líneas, con hasta velocidades de 100, 400 y 1000 kbits/s. También es conocido como IIC ó TWI – Two Wire Interface.

Este protocolo es ampliamente utilizado para comunicar diversos sensores digitales ya que a diferencia del puerto Serial, su arquitectura permite tener una confirmación de los datos recibidos, dentro de la misma trama, entre otras ventajas.

La conexión de tantos dispositivos al mismo bus, es una de las principales ventajas. Además, si comparamos a I2C con otro protocolo serial, como Serial TTL, este incluye más bits en su trama de comunicación que permite enviar mensajes más completos y detallados.

El PCF8574 / 74A proporciona una expansión de E / S remotas de propósito general a través del bus I2C bidireccional de dos cables (reloj de serie (SCL), datos de serie (SDA)).

Los dispositivos constan de ocho puertos cuasi bidireccionales, interfaz de bus I2C de 100 kHz, tres entradas de dirección de hardware y una salida de interrupción que opera entre 2.5 V y 6 V. El puerto cuasi bidireccional se puede asignar de forma independiente como una entrada para monitorear el estado de interrupción o los teclados, o como una salida para activar dispositivos indicadores como los LED. El maestro del sistema puede leer desde el puerto de entrada o escribir en el puerto de salida a través de un solo registro.

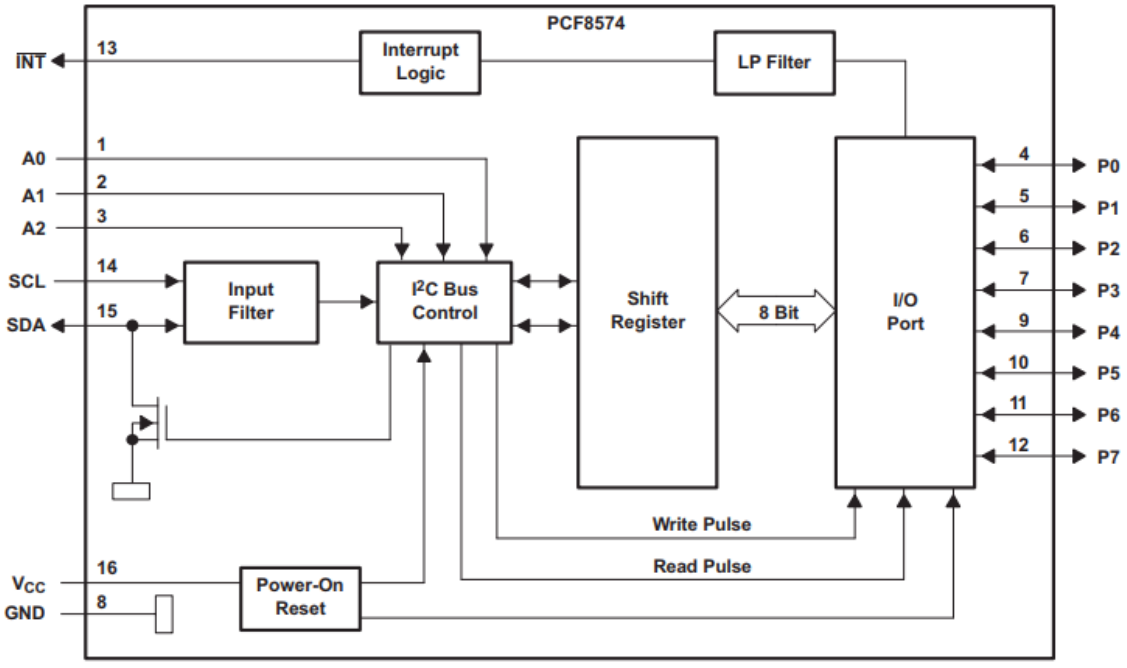

Figura 6: Integrado PCF8574

Decodificador de teclado

El codificador de teclado MM74C922 proporcionan toda la lógica necesaria para codificar completamente una matriz de interruptores. Este tiene una ventaja ya que tiene un pin que cambia de estado cuando es presionado un botón entonces es posible que este pin se usó como interrupción para el microcontrolador cuando se ha presionado un interruptor y así evitar que el micro este checando siempre si alguna tecla se presionó.

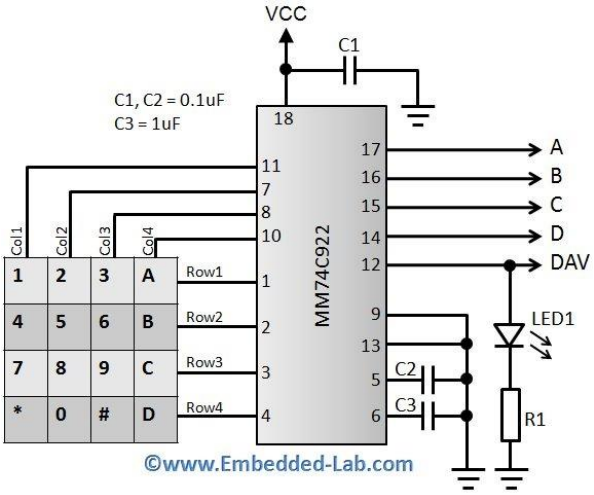

Figura 7: Configuración típica del MM74C922

Transceptor Rs485

Este Integrado como su nombre lo dice va a pasar a una comunicación Serial a Rs485 esto para mejorar la comunicación en ambientes ruidosos. Esta comunicación es ideal para transmitir a altas velocidades sobre largas distancias. Este usa un par trenzada como medio físico de transmisión. Algunas otras características sobresalientes son:

- Intercambio de datos bidireccional a través de un par de hilos trenzados.
- Soporte para varios transceptores conectados a la misma línea, es decir, la capacidad de crear una red.

Entonces se propone usar esta este tipo de comunicación para este proyecto el módulo que se usara es el de la figura siguiente.

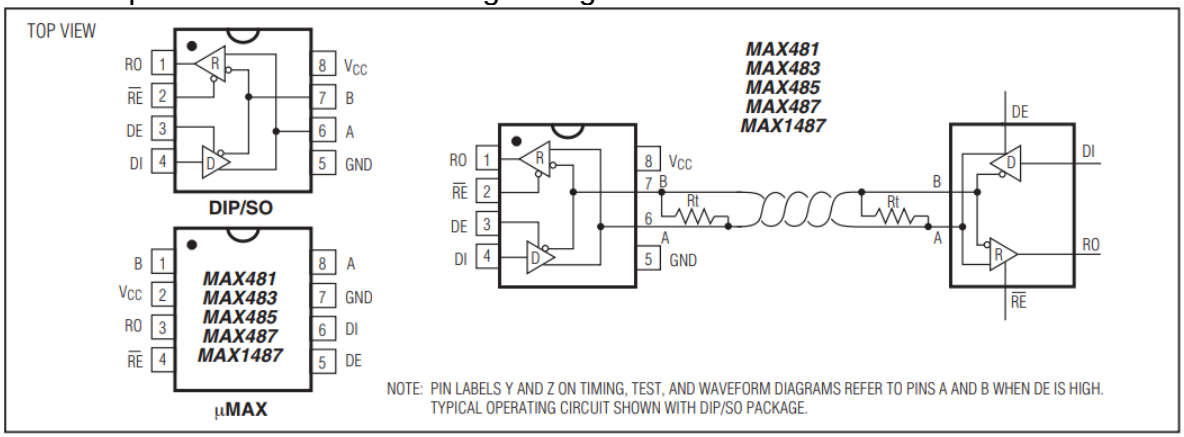

Figura 8: MAX485.

Manejador de motor a pasos A4988

Esto integrado es muy versátil por que simplifica el manejo de los motores paso a paso. Dicho integrado nos permite manejar altos voltajes que requieren estos motores y además proporciona una protección para evitar daños en los circuitos este A4988 ha tomado mucha popularidad en estos tiempos ya que se usa para en impresoras 3D caseras este integrado nos permite trabajar con voltajes de hasta 35 volts.

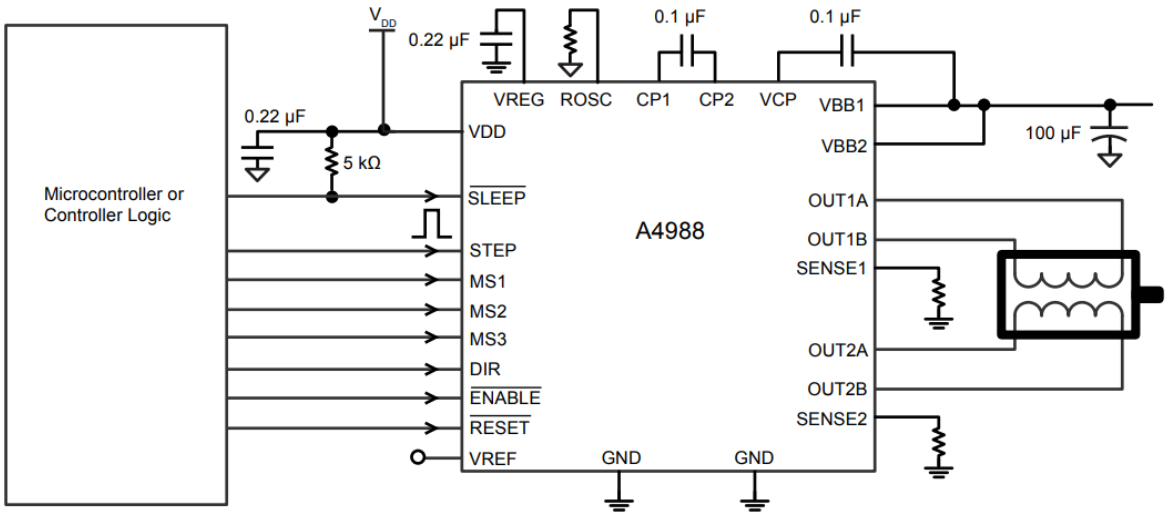

Figura 9: Integrado A4988

Galga extensométrica

La celda de carga es un tipo de transductor que permite convertir la fuerza en señal eléctrica y esto se logra a través de una o más galgas internas que posee, configuradas en un puente Wheatstone.

Las Galgas extensométricas es un tipo de sensor que mide la deformación, carga, etcétera y se base en un efecto llamado piezorresistivo, que es una cualidad que tiene ciertos materiales de cambiar su resistencia cuando se les somete a esfuerzos y se deforma en dirección de sus ejes mecánicos.

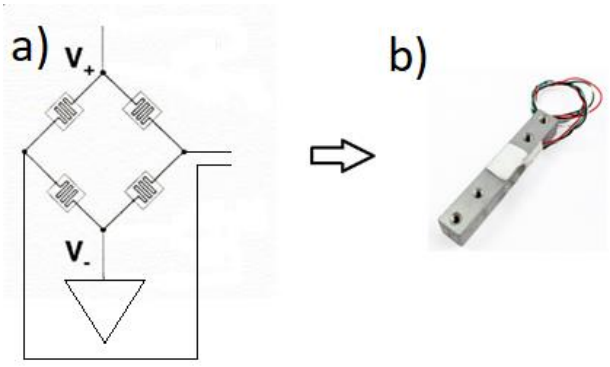

Figura 10: a) Galga extensométrica. b) Galga extensométrica recubierta de uso comercial

HX711

Este Integrado es un intermedio entre las celdas de cargar y el microcontrolador, permitiendo poder leer el peso de manera sencilla. Internamente se encarga de la lectura del puente Wheatstone formado por la celda de carga, pasándola por un mux y después convirtiendo la lectura analógica a digital con su conversor A/D interno de 24 bits. Se comunica con el microcontrolador mediante 2 pines (Clock y Data) de forma serial.

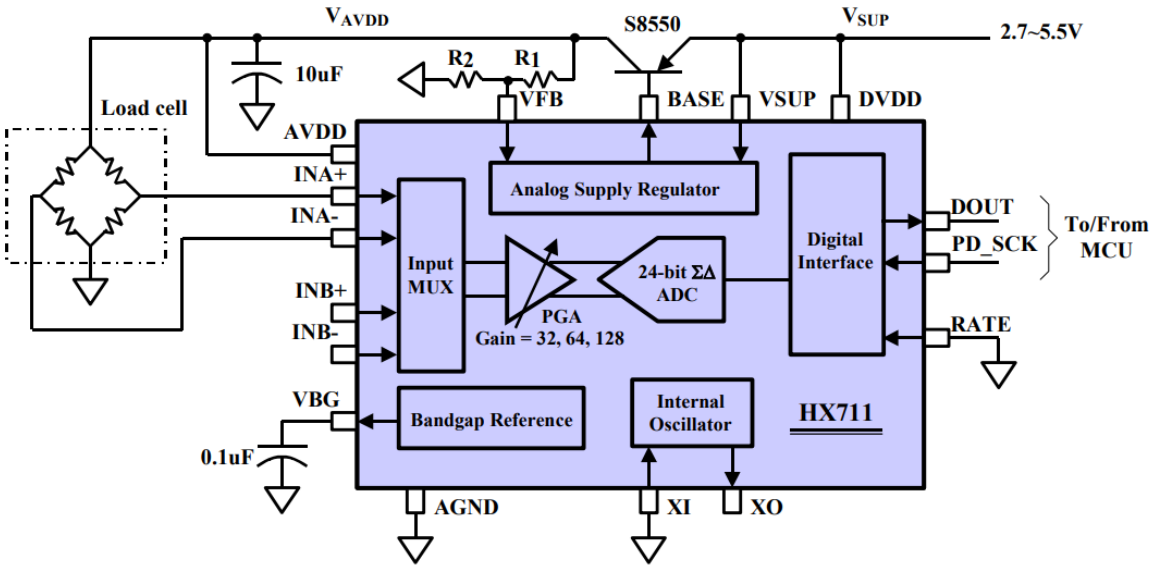

Figura 11: Integrado HX711

Motor Paso a Paso

El motor usado en este proyecto es motor bipolar este tipo de motores por lo general tiene 4 cables de salida, necesitan ciertas manipulaciones para poder ser controlados, debido a que requieren del cambio de dirección de flujo de corriente a través de sus bobinas en la secuencia apropiada para realizar un movimiento, es necesario un puente H por cada bobina del motor, es decir que para controlar un motor paso a paso de 4 cables (dos bobinas), se necesitan usar dos puentes H.

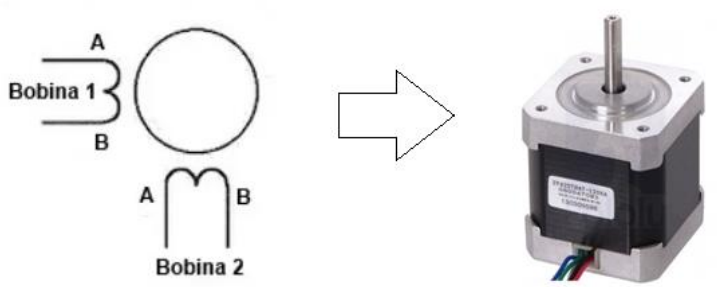

Figura 12: a) diagrama eléctrico del motor a pasos b) motor a pasos

El desarrollo de un programa en el microcontrolador que permita realizar todas las tareas requeridas.

Implementación de un LCD por medio del BUS IC2 al PSOC

En esta parte se hizo la implementación del LCD el diagrama de conexión solo es hacer que el módulo con el PCF8574 coincida perfectamente con el LCD como se muestra en el diagrama de conexión de la siguiente imagen.

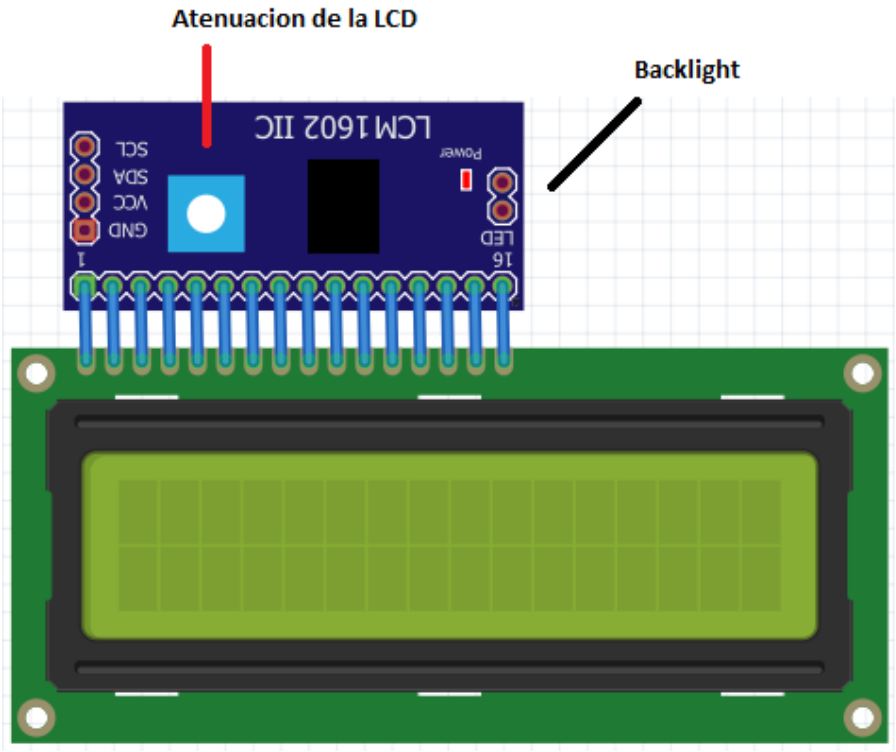

Figura 13: Diagrama de conexión de la LCD

Por obvias razones se conecta el GND a GND de nuestra placa y el VCC al VCC de nuestra placa y el SDA y SCL se conecta al pin 12.5(SDA) y 12.4(SCL) de nuestra tarjeta de desarrollo.

Ahora en el entorno de desarrollo de nuestra tarjeta la cual tiene un entorno grafico bastante amigable empezamos agregando el módulo LCD I2C para esto nos vamos a “Project” y después “Dependencies” se nos abrirá una ventana como la que se muestra a continuación.

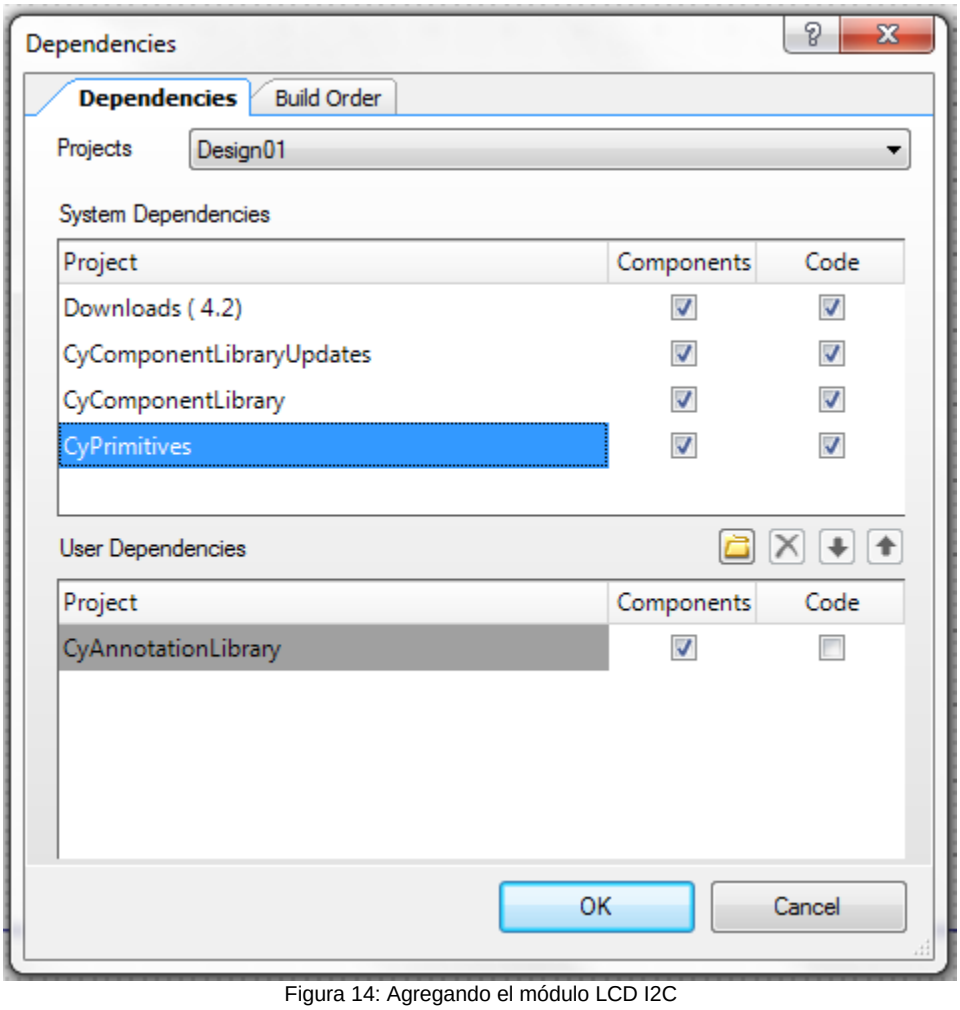

Figura 14: Agregando el módulo LCD I2C

Seleccionamos en el Icono de la carpeta amarilla y buscamos el archivo de nuestro bloque de hardware este será un tipo de archivo con extensión “.cyprj” y nuestro entorno de desarrollo podrá identificarlos fácilmente una vez dado clic en “OK” se cerrará la ventana y volveremos a nuestra ventana de Dependencies aquí solo damos clic en “OK” y ya tendremos agregado el bloque de hardware.

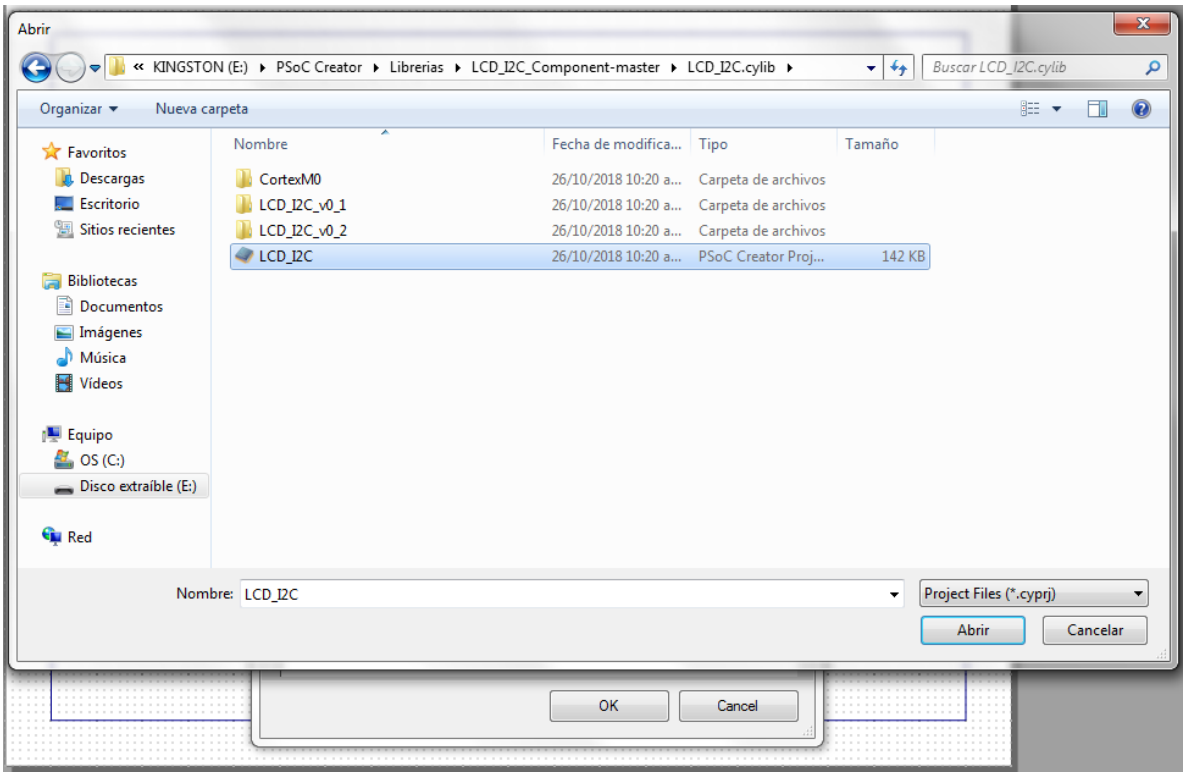

Figura 15: Agregando el módulo LCD I2C

Una vez agregado el bloque de hardware solo resta agregar dicho bloque a nuestro diseño junto con un bloque de I2C del “Component Catalog” situado a la derecha de nuestro IDE donde nuestro bloque recién agregado se deberá buscar en la pestaña de Community después Display y arrastramos el LCD_I2C a nuestro diseño lo mismo se hará con el bloque de I2C situado en la pestaña de Cypress después Communications en la carpeta de I2C seleccionaremos el bloque de “I2C Master (Fixed Function) [v3.50]”.

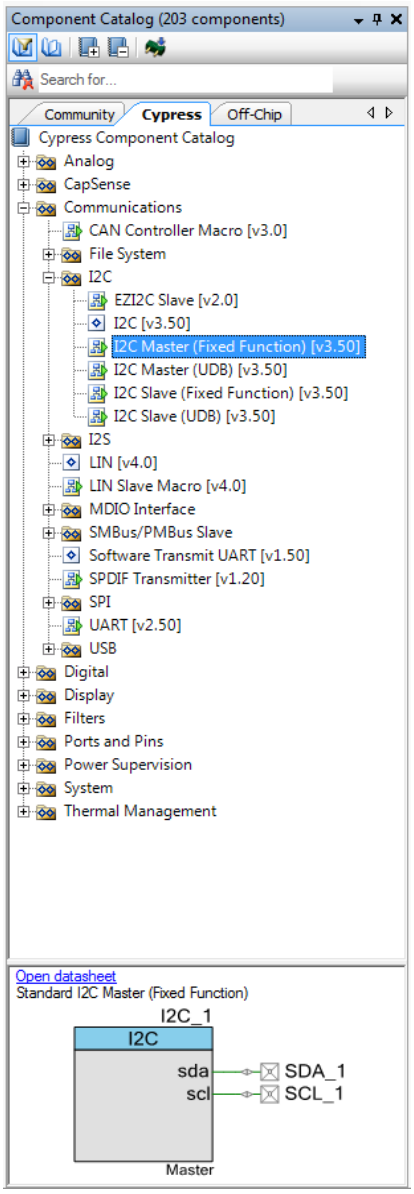

Figura 16: Component Catalog

Una vez agregados nuestros bloques procederemos a configurar el bloque LCD_I2C donde se tendrán que aplicar algunos cambios.

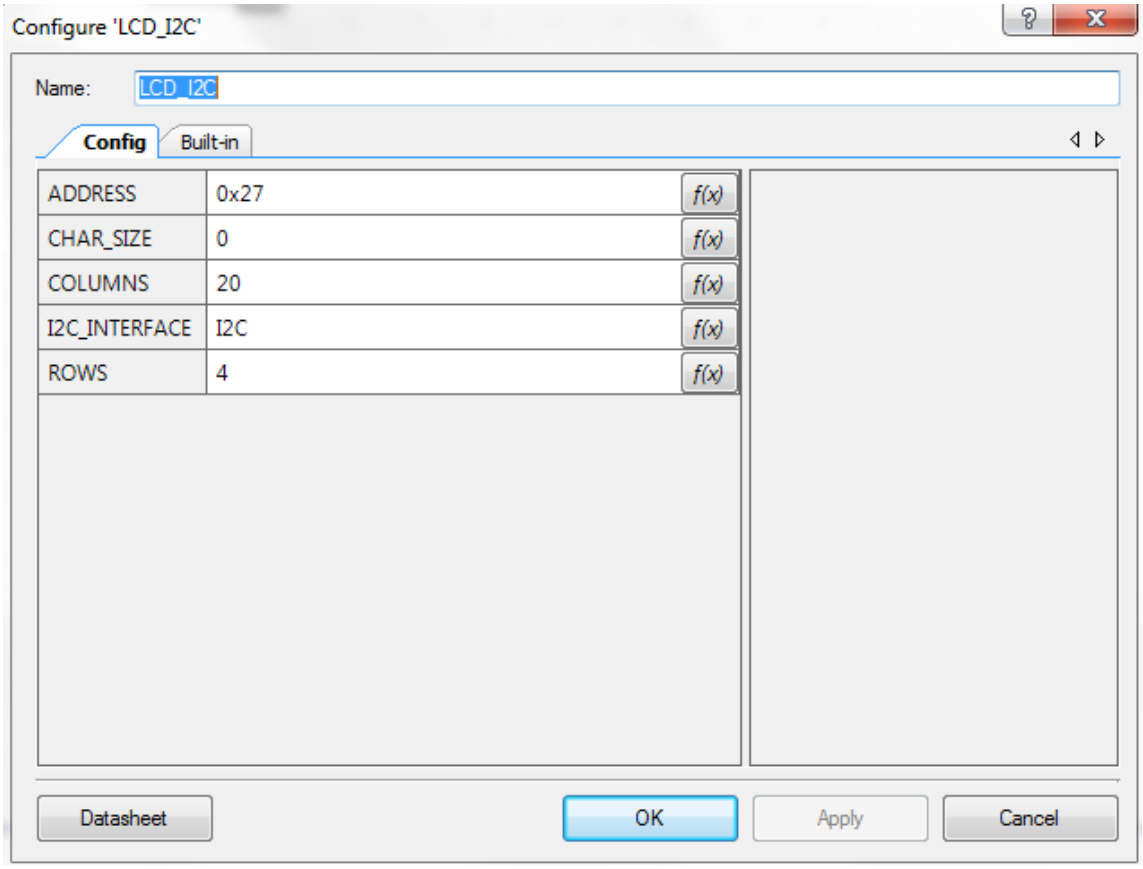

Figura 17: Configuración del LCD_I2C

Para esta configuración se empieza cambiaron el nombre del LCD de “LCD_I2C_1” a “LCD_I2C” después en donde dice address se pone la dirección de esclavo de nuestro módulo I2C que se puede encontrar en su Datasheet.

Address maps

The PCF8574 and PCF8574A are functionally the same, but have a different fixed portion (A6 to A3) of the slave address. This allows eight of the PCF8574 and eight of the PCF8574A to be on the same I²C-bus without address conflict.

Table 4. PCF8574 address map

Pin connectivity			Address of PCF8574								Address byte value		7-bit hexadecimal address without R/W
A2	A1	A0	A6	A5	A4	A3	A2	A1	A0	R/W	Write	Read	
V_{SS}	V_{SS}	V_{SS}	0	1	0	0	0	0	0	-	40h	41h	20h
V_{SS}	V_{SS}	V_{DD}	0	1	0	0	0	0	1	-	42h	43h	21h
V_{SS}	V_{DD}	V_{SS}	0	1	0	0	0	1	0	-	44h	45h	22h
V_{SS}	V_{DD}	V_{DD}	0	1	0	0	0	1	1	-	46h	47h	23h
V_{DD}	V_{SS}	V_{SS}	0	1	0	0	1	0	0	-	48h	49h	24h
V_{DD}	V_{SS}	V_{DD}	0	1	0	0	1	0	1	-	4Ah	4Bh	25h
V_{DD}	V_{DD}	V_{SS}	0	1	0	0	1	1	0	-	4Ch	4Dh	26h
V_{DD}	V_{DD}	V_{DD}	0	1	0	0	1	1	1	-	4Eh	4Fh	27h

Figura 18: Dirección de esclavo del I2C PCF8574

Se nos muestra diferentes direcciones porque hay unos pines en el módulo I2C llamados A0, A1 y A2 que no están conectado a nada por defecto los dejamos así y tiene valor 1 entonces tomamos el valor que le corresponde en este caso 0x27 en Char_size no se le mueve nada donde dice Columns se le ponen las columnas de nuestra LCD en esta caso usamos un LCD 20x4, donde dice I2C_Interface le ponemos el nombre del módulo I2C Master (Fixed Function) [v3.50] que agregamos es importante que tengan en el mismo porque no funcionaría si se le pone aquí un nombre y renombramos el módulo de otra manera y en Rows se pone el número de Filas de nuestro LCD. En el Módulo I2C Master (Fixed Function) [v3.50] solo se renombra y no se le mueve nada más.

Para nuestro Proyecto solo es necesario conocer 3 APIS de la LCD que nos ayudaran en todo el proyecto las cuales son.

```
LCD_I2C_start(); /*Esta se usa para inicializar la LCD
LCD_I2C_setCursor(columna,fila); /*Sitúa la posición del cursor de la LCD
LCD_I2C_print("Texto"); /* Recibe una Cadena y la imprime en el LCD
```

Implementación de un Teclado Matricial

Para la configuración del teclado del teclado se usó el MM74C922 donde en la figura 4 se puede apreciar una configuración típica de ese modulo para la comunicación el PSOC se utilizó la siguiente configuración.

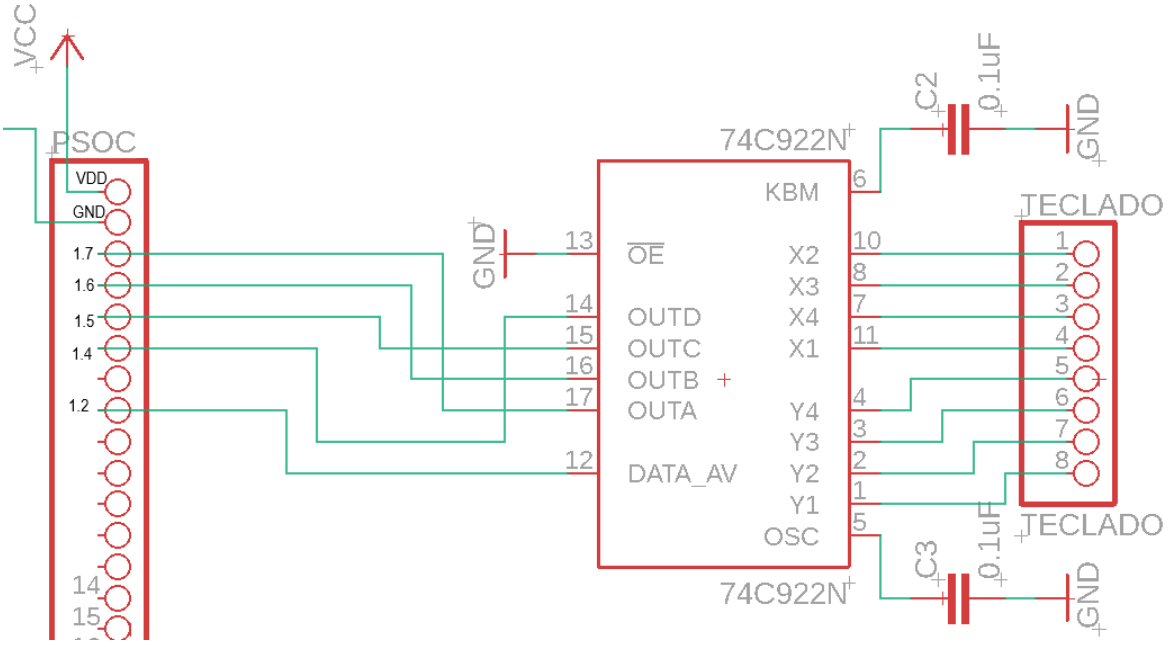

Figura 19: Configuración Teclado con el 74C922 y el PSOC

La Función de este integrado es codificar un teclado matricial entonces de acuerdo al botón que se presione el 74C922 te lo entrega en una salida digital que puede ser fácilmente leída por el PSOC además también nos ofrece una interrupción esto con el fin de leer los valores solo cuando se apriete una tecla.

Ahora en el en nuestro entorno de desarrollo se agregaron los pines necesarios para este integrado en nuestro entorno vamos al catálogo de componentes en la pestaña de Cypress en la carpeta de “Ports and Pins” y se agregaron 5 entradas digitales en nuestro Ide aparece como “Digital Input Pin”.

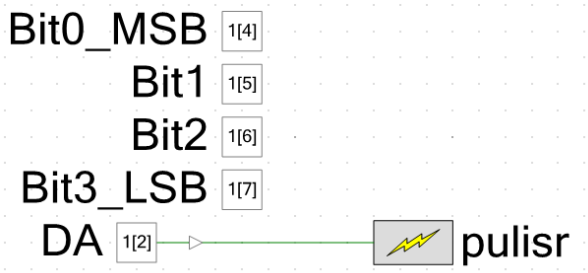

Figura 20: Agregando entradas digitales

Después se le cambiaron los nombres le desmarcamos la casilla de “HW connection” a 4 de ellas y se puso como Pull down y su estado inicial en bajo como en la siguiente imagen.

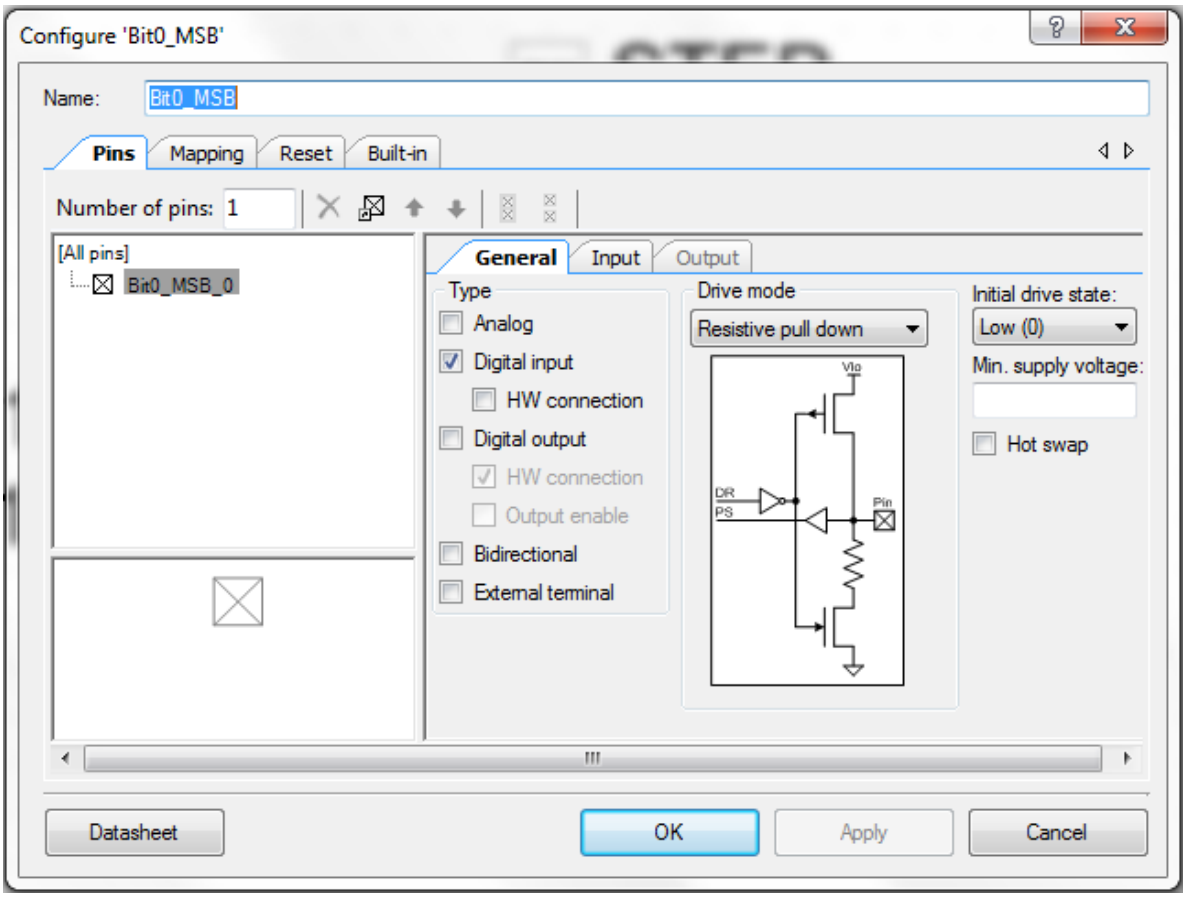

Figura 21: Configurando los pines

Se agregó un bloque de interrupción como el de la figura 18 ese bloque se encuentra en la pestaña de Cypress en la carpeta System y se llama “Interrupt” y se configuro como en la siguiente imagen.

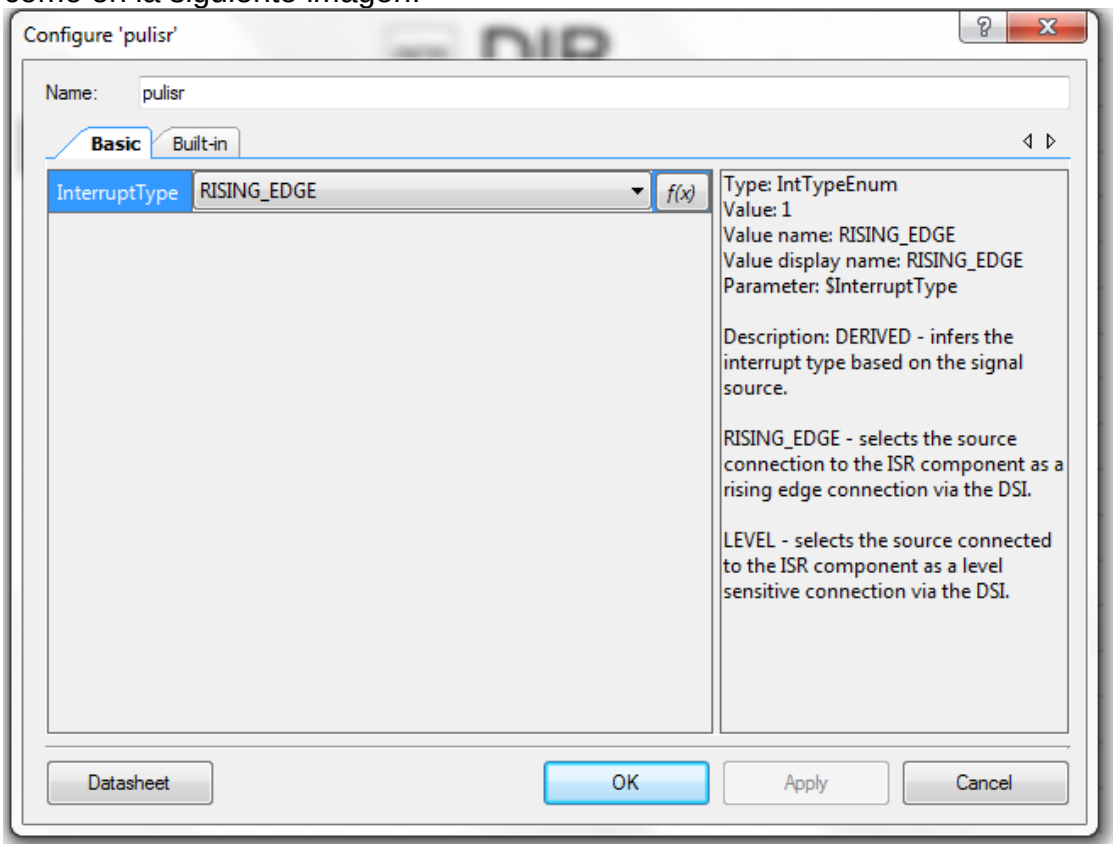

Figura 22: Configurando Interrupción

En la programación se hicieron dos rutinas una para obtener los valores del teclado y otra para leer dicho valor se presenta a continuación el código para obtener el valor del teclado.

```
void LeerTecla(){
    bit0_MSB=Bit0_MSB_Read();
    bit1=Bit1_Read();
    bit2=Bit2_Read();
    bit3_LSB=Bit3_LSB_Read();
    if((bit0_MSB==0)&&(bit1==0)&&(bit2==0)&&(bit3_LSB==0))//1
        i=Keypad_1_Key[0][0];
    if((bit0_MSB==0)&&(bit1==0)&&(bit2==0)&&(bit3_LSB==1))//2
        i=Keypad_1_Key[1][0];
    if((bit0_MSB==0)&&(bit1==0)&&(bit2==1)&&(bit3_LSB==0))//3
        i=Keypad_1_Key[2][0];
    if((bit0_MSB==0)&&(bit1==0)&&(bit2==1)&&(bit3_LSB==1))//A
        i=Keypad_1_Key[3][0];
    if((bit0_MSB==0)&&(bit1==1)&&(bit2==0)&&(bit3_LSB==0))//4
        i=Keypad_1_Key[0][1];
    if((bit0_MSB==0)&&(bit1==1)&&(bit2==0)&&(bit3_LSB==1))//5
        i=Keypad_1_Key[1][1];
    if((bit0_MSB==0)&&(bit1==1)&&(bit2==1)&&(bit3_LSB==0))//6
        i=Keypad_1_Key[2][1];
```

```
    if((bit0_MSB==0)&&(bit1==1)&&(bit2==1)&&(bit3_LSB==1))//B
        i=Keypad_1_Key[3][1];
    if((bit0_MSB==1)&&(bit1==0)&&(bit2==0)&&(bit3_LSB==0))//7
        i=Keypad_1_Key[0][2];
    if((bit0_MSB==1)&&(bit1==0)&&(bit2==0)&&(bit3_LSB==1))//8
        i=Keypad_1_Key[1][2];
    if((bit0_MSB==1)&&(bit1==0)&&(bit2==1)&&(bit3_LSB==0))//9
        i=Keypad_1_Key[2][2];
    if((bit0_MSB==1)&&(bit1==0)&&(bit2==1)&&(bit3_LSB==1))//C
        i=Keypad_1_Key[3][2];
    if((bit0_MSB==1)&&(bit1==1)&&(bit2==0)&&(bit3_LSB==0))//*
        i=Keypad_1_Key[0][3];
    if((bit0_MSB==1)&&(bit1==1)&&(bit2==0)&&(bit3_LSB==1))//0
        i=Keypad_1_Key[1][3];
    if((bit0_MSB==1)&&(bit1==1)&&(bit2==1)&&(bit3_LSB==0))//#
        i=Keypad_1_Key[2][3];
    if((bit0_MSB==1)&&(bit1==1)&&(bit2==1)&&(bit3_LSB==1))//D
        i=Keypad_1_Key[3][3];
}
```

Básicamente lo que se hace es preguntar por el estado de cada uno de los pines que elegimos como entrada asignarle un valor en nuestro caso quedo de esta manera si usted intenta implementar dicho código debe fijarse como coloca el teclado porque usted puede obtener lecturas diferentes a las mostradas anteriormente.

En la siguiente Rutina se lee dicho valor arrojado por la rutina anterior:

```
CY_ISR(LeerTeclaInt){
    LeerTecla();
    if((i!='A')&&(i!='B')&&(i!='C')&&(i!='D')&&(i!='*')&&(i!='#'))
    {
        if(cuenta<=3)
        {
            numero[cuenta]=i;
            cuenta++;
        }
    }
    if(i=='C')
    {
        numero[0]='\0';
        numero[1]='\0';
        numero[2]='\0';
        numero[3]='\0';
        cuenta=0;
    }
    if(i=='*')
    {
        DIR_Write(0);
        STEP_Write(1);
        CyDelay(10);
        STEP_Write(0);
        CyDelay(100);
    }
    if(i=='#')
    {
        DIR_Write(1);
```

```
        STEP_Write(1);
        CyDelay(10);
        STEP_Write(0);
        CyDelay(100);
    }
```

Aquí se pide el valor del teclado y se decide qué hacer con él, en el primer if se manda a un arreglo llamado número en el segundo if se limpia dicho arreglo y en el tercer y cuarto if se hace girar el motor paso a paso.

Implementación de controlador del Motor Paso a Paso.

Para la implementación del motor se usó la configuración del siguiente diagrama:

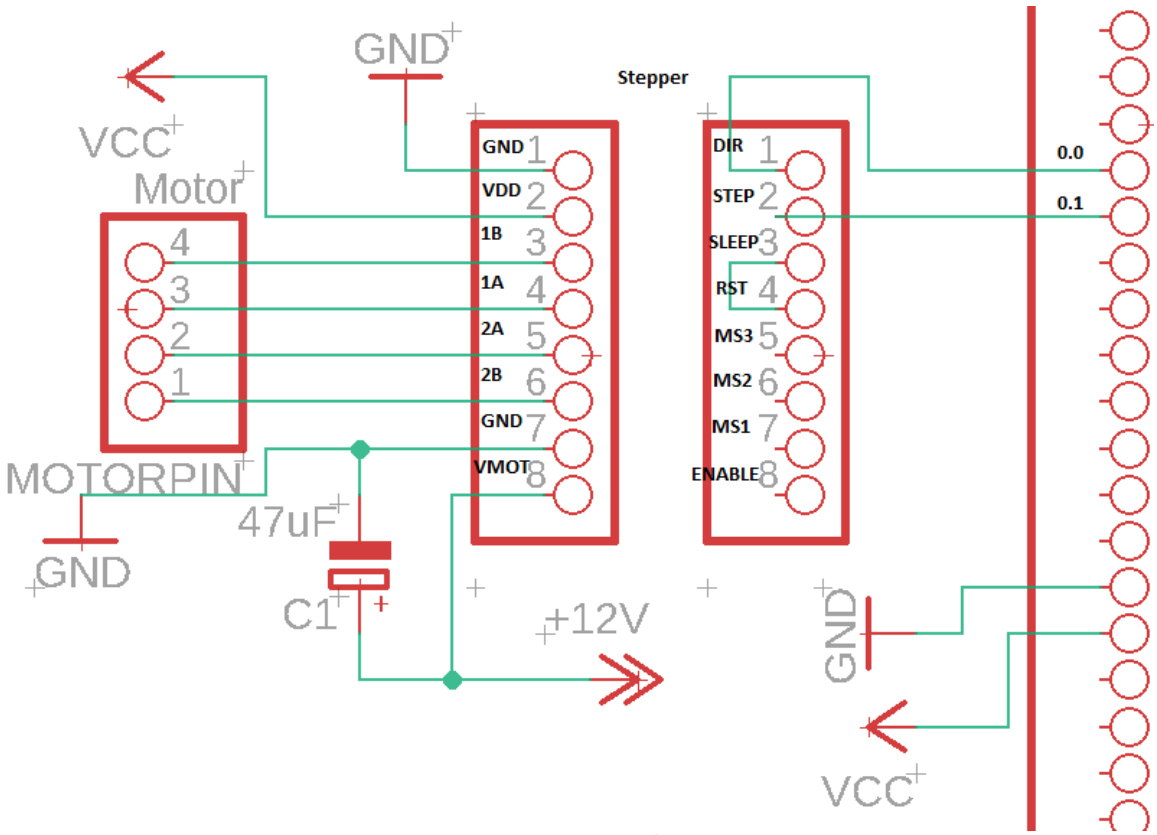

Figura 23: Diagrama del Conexión de Stepper

En el VMOT se le introdujeron 12V y en el VDD se le introdujo 5V. Ahora en el ambiente de programación se introdujeron dos Salidas Digitales una con el nombre de STEP y otra con el nombre de DIR se configuraron como en la siguiente imagen, estas salidas digitales en encuentran donde estaban las entradas digitales agregadas anteriormente.

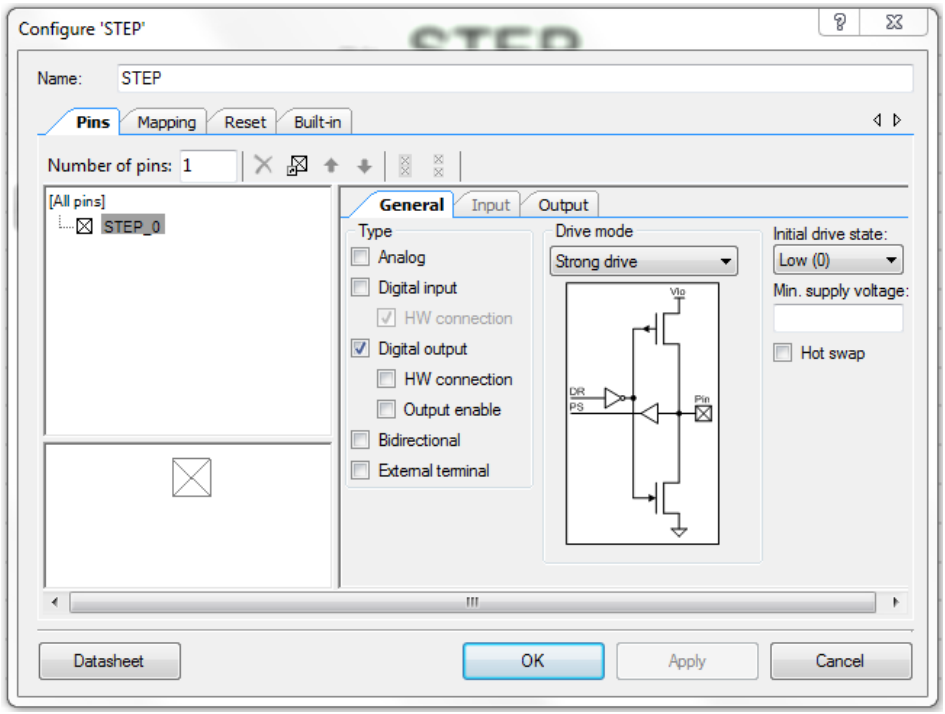

Figura 24: Diagrama del Conexión de Stepper

La programación de este driver fue fácil solo se requiere dar un pulso si quiere dar un paso y el otro pin gira a la izquierda si es 0 y a la derecha si es uno.

```
    /*Girar a la derecha*/
      STEP_Write(1);
      CyDelay(10);
      STEP_Write(0);
      CyDelay(100);
  /*Girar a la izquierda*/
      DIR_Write(1);
      STEP_Write(1);
      CyDelay(10);
      STEP_Write(0);
```

Este es un ejemplo de cómo se usa este driver una operación hace que el motor gire a la derecha y la otra hacer que gire a la izquierda.

Implementación de los dos Canales analógicos

Para la implementación de los canales analógicos se hizo uso de dos potenciómetros esto solo con fines de prueba nada mas de igual manera los

potenciómetros tienen 3 patas una se conecta a GND la otra a VCC y la otra al pin del PSOC5LP las entradas del PSOC que usaremos serán el 3.4 y 3.5 de nuestro PSOC.

En nuestro entorno de desarrollo se agregó el Bloque de hardware “Delta Sigma ADC” el mismo que se encuentra en la pestaña de Cypress en la carpeta Analog se le cambia el nombre y se configuro de la siguiente manera.

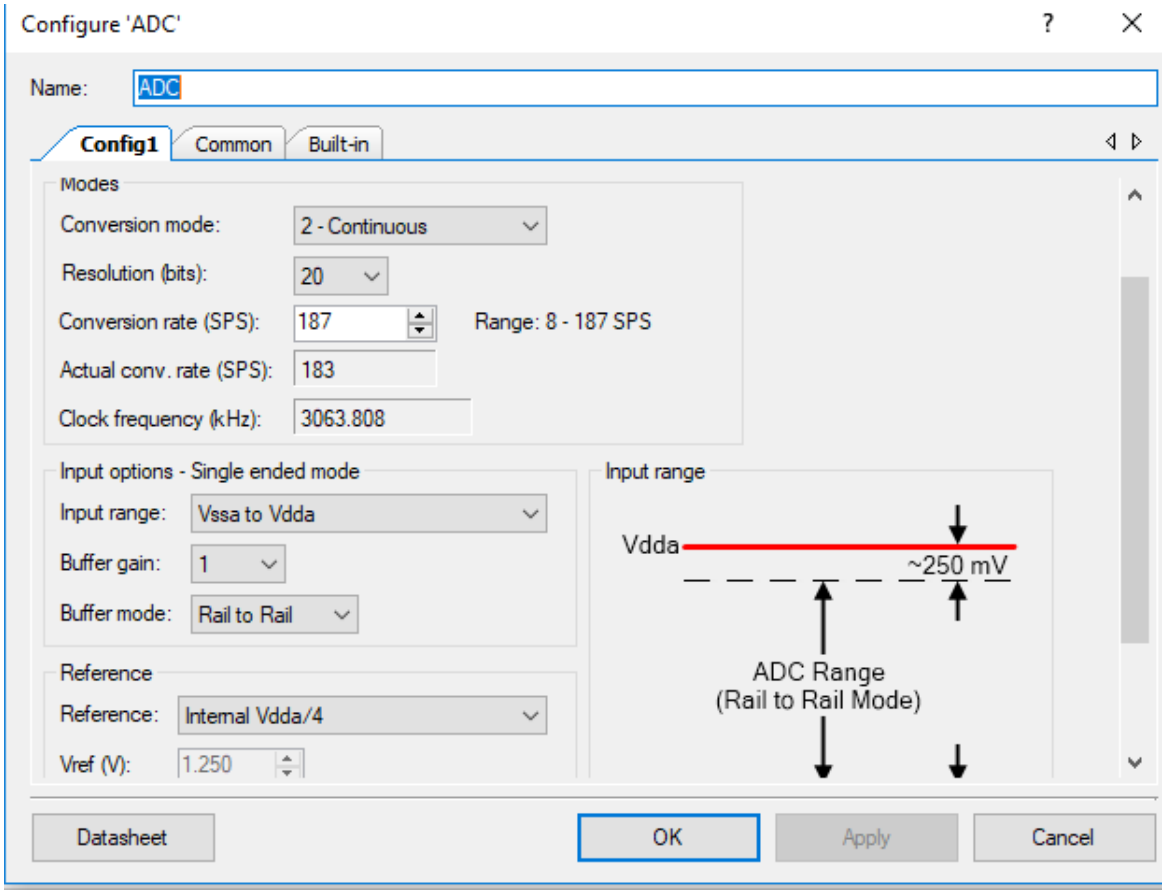

Figura 25-1: Configuración del ADC

Se configuro con una resolución de 20bits en rango de entrada se le puso “Vssa to Vdda” y en referencia se le puso “Internal Vdda/4” ahora en la pestaña de Common se hicieron los siguientes cambios se puso en “Clock Source Intenal” y Input mode como Single ended como se puede apreciar en la figura 21-2.

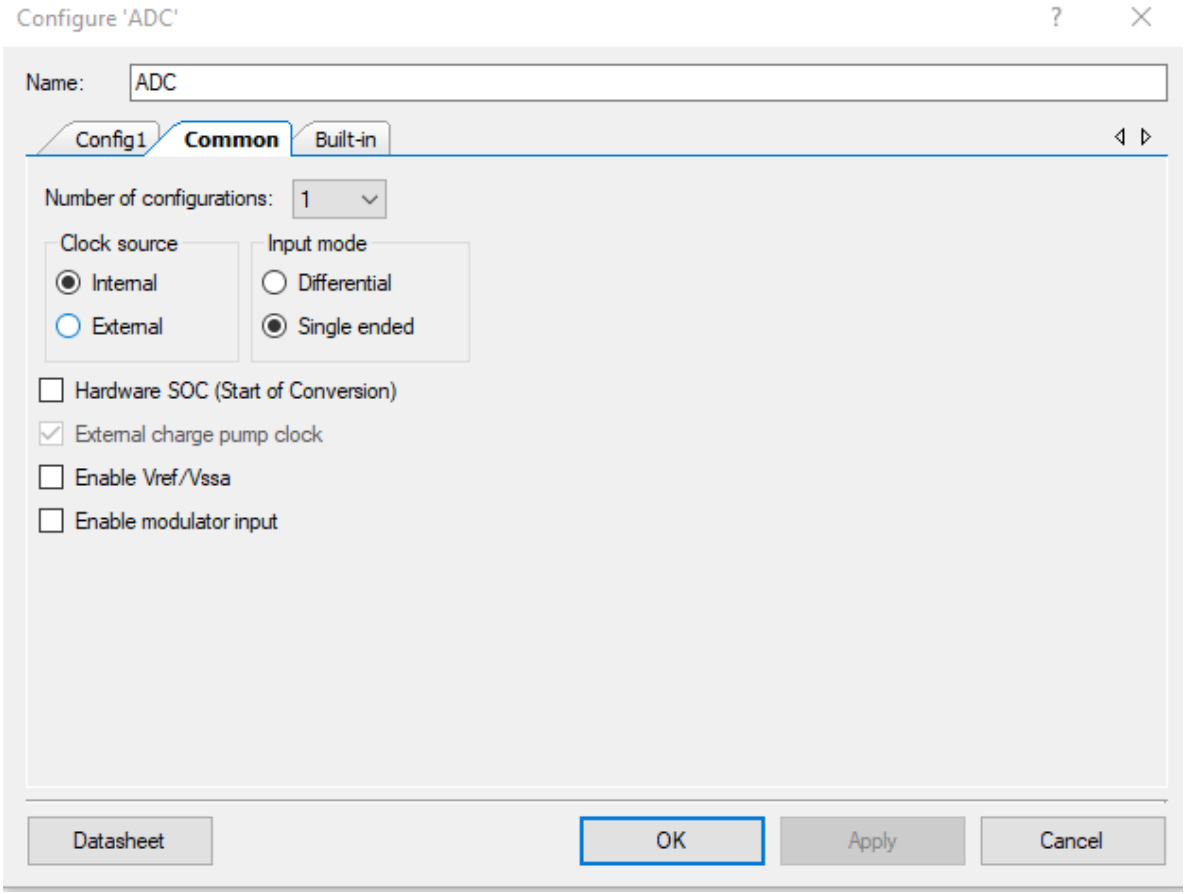

Figura 25-2: Configuración del ADC

Después se agregó un Mux en nuestro entorno de desarrollo que se encuentra en la pestaña de Cypress en la dirección Analog/Analog MUX y se agregó el bloque de Analog Mux por defecto cuando se abren las configuraciones del mux está en 4 canales se la cambio a 2 canales.

Después se agregaron Amplificadores Operacionales estos se encuentran en Analog/Amplifiers y tiene el nombre de Opamp ya agregados se configuraron como seguidores como en la siguiente imagen.

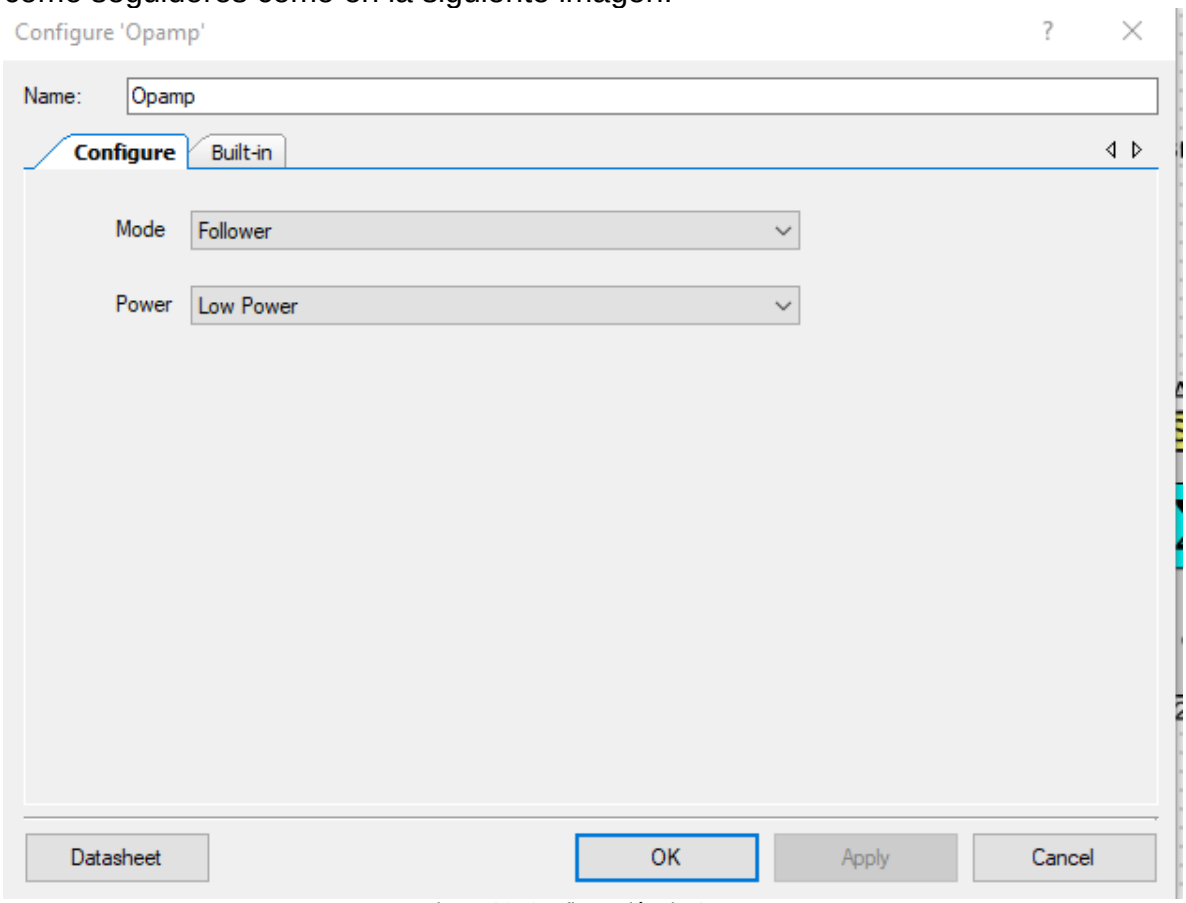

Figura 26: Configuración de Opamp

Y por último se agregaron dos entradas analógicas estas se encuentran donde encontramos las salidas y entradas digitales en “Ports and Pins” esta entrada analógica por defecto viene configurada con estado inicial bajo y en modo de alta impedancia analógica así lo vamos a dejar solo le cambiamos el nombre a InAnalog al final todo lo conectamos de la siguiente manera.

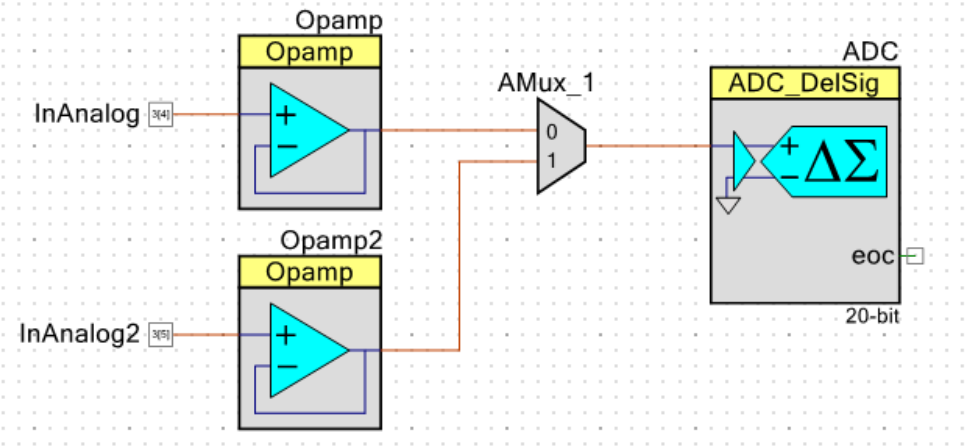

Figura 27: Diseño de los dos Canales Analógicos

Para la programación de nuestro ADC se necesitó consultar el Datasheet del componente que se encuentra dentro del mismo IDE al seleccionar el componente en la siguiente tabla se pueden observar algunas APIS obtenidas del Datasheet que se muestran a continuación [5].

Functions

Function	Description
ADC_Start()	Sets the initVar variable, calls the ADC_Init() function, and then calls the ADC_Enable() function.
ADC_Stop()	Stops ADC conversions and powers down
ADC_SetBufferGain()	Selects input buffer gain (1,2,4,8)
ADC_StartConvert()	Starts conversion
ADC_StopConvert()	Stops conversions
ADC_IRQ_Enable()	Enables interrupts at the end of conversion
ADC_IRQ_Disable()	Disables interrupts
ADC_IsEndConversion()	Returns a nonzero value if conversion is complete
ADC_GetResult8()	Returns an 8-bit conversion result
ADC_GetResult16()	Returns a 16-bit conversion result
ADC_GetResult32()	Returns a 32-bit conversion result
ADC_Read8()	Starts ADC conversions, waits for the conversion to be complete, stops ADC conversion and returns the signed 8-bit value of result.
ADC_Read16()	Starts ADC conversions, waits for the conversion to be complete, stops ADC conversion and returns the signed 16-bit value of result.
ADC_Read32()	Starts ADC conversions, waits for the conversion to be complete, stops ADC conversion and returns the signed 32-bit value of result.
ADC_SetOffset()	Sets the offset used by the ADC_CountsTo_mVolts(), ADC_CountsTo_uVolts(), and ADC_CountsTo_Volts() functions.
ADC_SelectConfiguration()	Sets one of up to four ADC configurations
ADC_SetGain()	Sets the gain used by the ADC_CountsTo_mVolts(), ADC_CountsTo_uVolts(), and ADC_CountsTo_Volts() functions.
ADC_CountsTo_mVolts()	Converts ADC counts to millivolts
ADC_CountsTo_uVolts()	Converts ADC counts to microvolts
ADC_CountsTo_Volts()	Converts ADC counts to floating point volts

Figura 28: Tabla de APIS para el ADC

Implementación del UART

Para esta parte lo primero que se hizo fue configurar el hardware del PSOC para que este implemente esta configuración en hardware dentro de los bloques de lógica programable que contiene.

Para esto en nuestro entorno de desarrollo vamos nuestro Catálogo de componentes y ubicamos la pestaña de comunications y agregamos el bloque UART a nuestro diagrama de hardware como ya lo hemos hecho anteriormente.

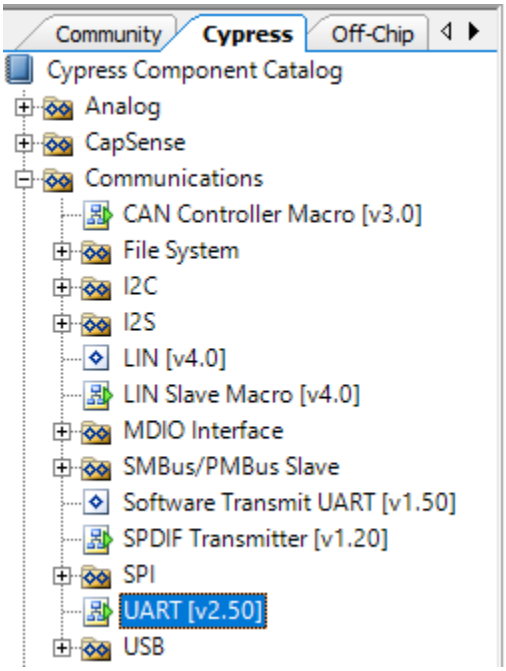

Figura 29: Bloque UART

Una vez que tenemos el bloque agregado hay que configurarlo para ello se abrio la configuración del UART.

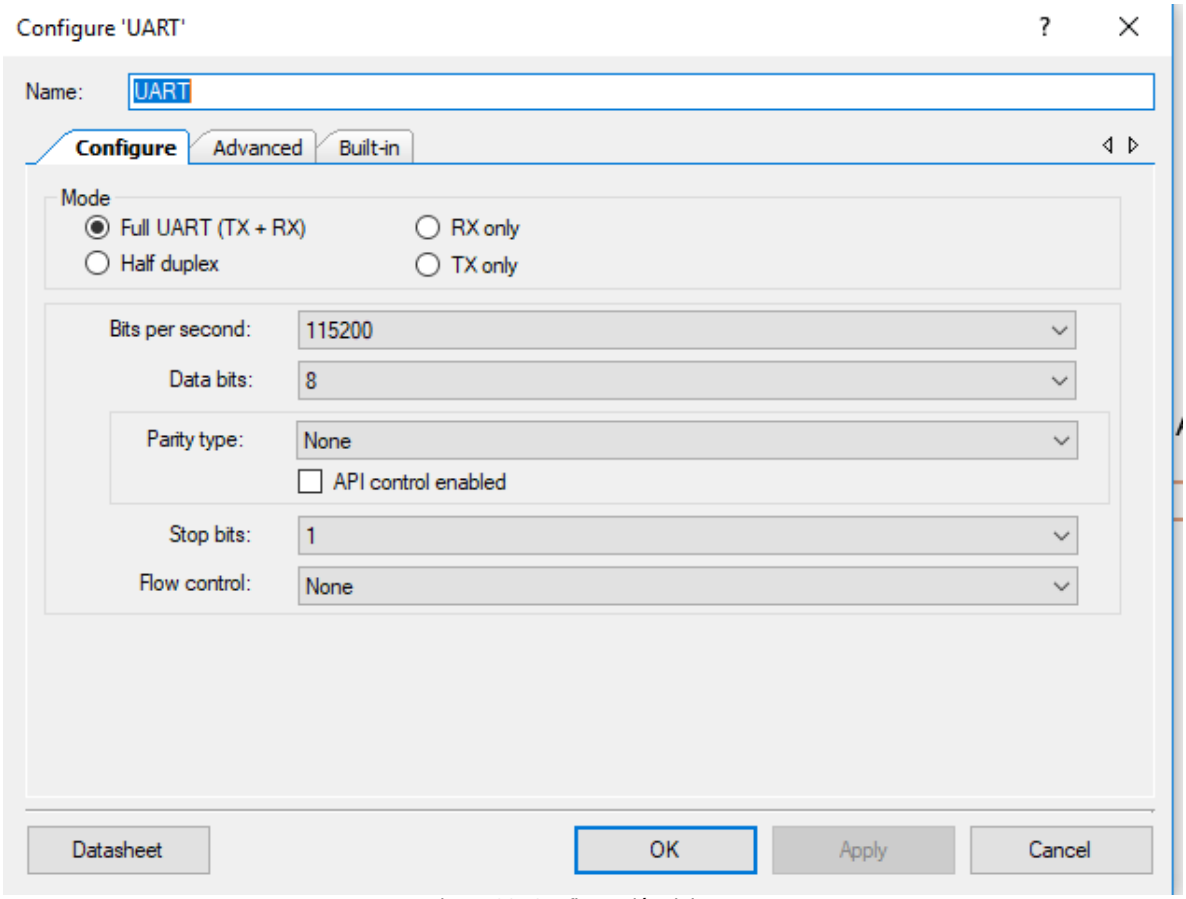

Figura 30: Configuración del UART

Dentro de esta pestaña hay diversos parámetros que se pueden configurar como el tipo de comunicación que se desea realizar puede ser solo de transmisión o solo de recepción, halfduplex o full duplex dependiendo del tipo de comunicación que se desee realizar, la implementación ocupara más o menos bloques UDB. Otro parámetro es la velocidad de transmisión que está dada en baudios, un parámetro común para esta es 115200 baudios, después nos encontramos con el número de bits a enviar por transmisión que normalmente son 8, los siguientes parámetros son el bit de paridad para checar errores, el bit de parada y el control de flujo.

Una vez que se agregó y configuro dicho bloque UART como en la figura anterior procedió a agregar nuestro bloque de PWM esto para configurar la interrupción y envió de datos por UART. De modo que se tomó un bloque PWM de nuestro catálogo de componentes esta vez lo tomamos de Digital/Funtions/PWM y lo agregamos a nuestro espacio de trabajo.

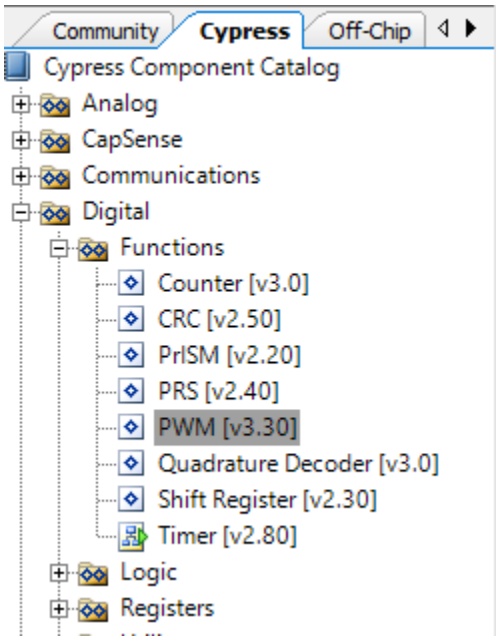

Figura 31: Agregando el PWM.

También se agregó un Reloj este se encuentra en System y se llama “Clock” y también un cero lógico es lo podemos encontrar en Digital/Functions tiene el nombre de “Logic Low ‘0’” también se agregó una interrupción como las que agregamos anteriormente y una salida digital también agregadas anteriormente cuando se hizo la parte del teclado.

Después de configurar todos estos componentes procedemos con el PWM este se configuro con una Resolución de 8bits, y se configuro el modo de una salida con un periodo de 255 con en la siguiente imagen.

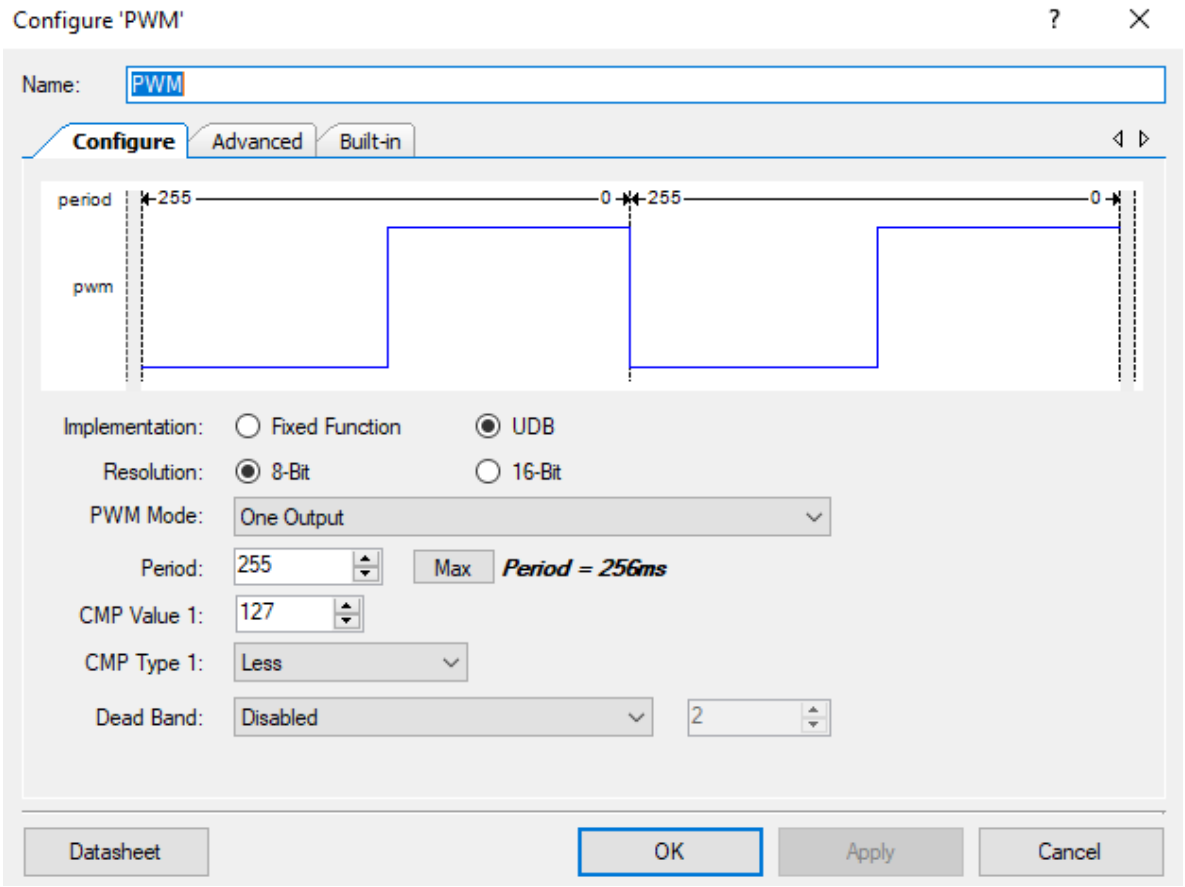

Figura 32: Agregando el PWM

El reloj se configuro con una frecuencia de 1Khz, el cero lógico lo dejo, así como esta, la interrupción se configura de la misma manera que se configura la interrupción del teclado y la salida digital se configura de la siguiente solo se le agrega un pequeño cambio.

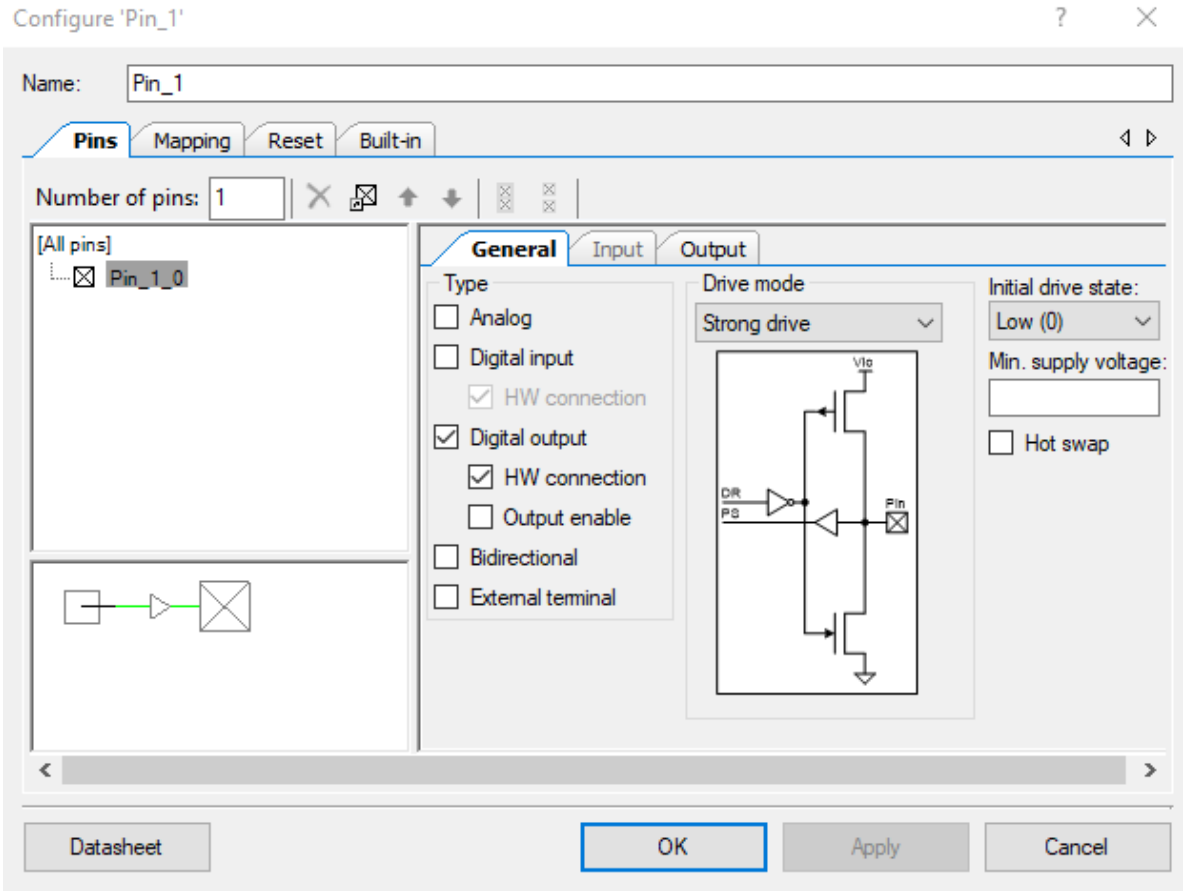

Figura 33: Configurando la salida digital

Todos los componentes mencionados anteriormente se conectaron de la siguiente manera.

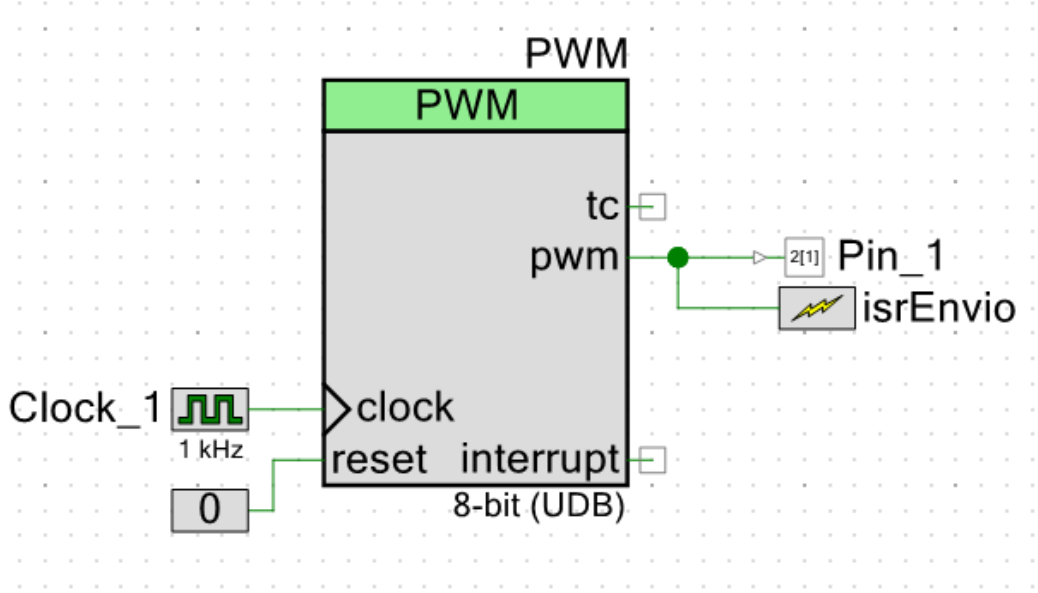

Figura 34: Diseño del PWM

Uniendo todos los bloques de Hardware de nuestro PSOC

Después se juntaron todos los bloques de hardware en un mismo esquema de trabajo para que cuando compilemos nuestro IDE genere todas las aplis que utilizaremos nos quedara un ambiente de trabajo como este.

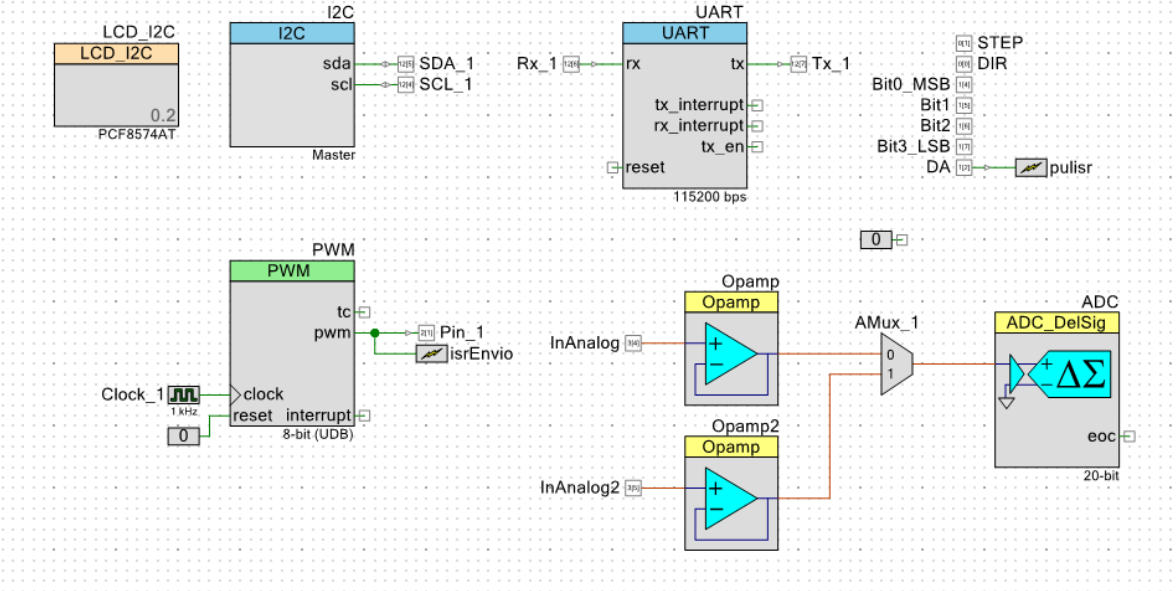

Figura 35: Esquema de trabajo completo

Después de asignar las configuraciones correspondientes para cada bloque de hardware solo queda asignar los pines que usaremos en nuestro PSOC y finalmente compilar.

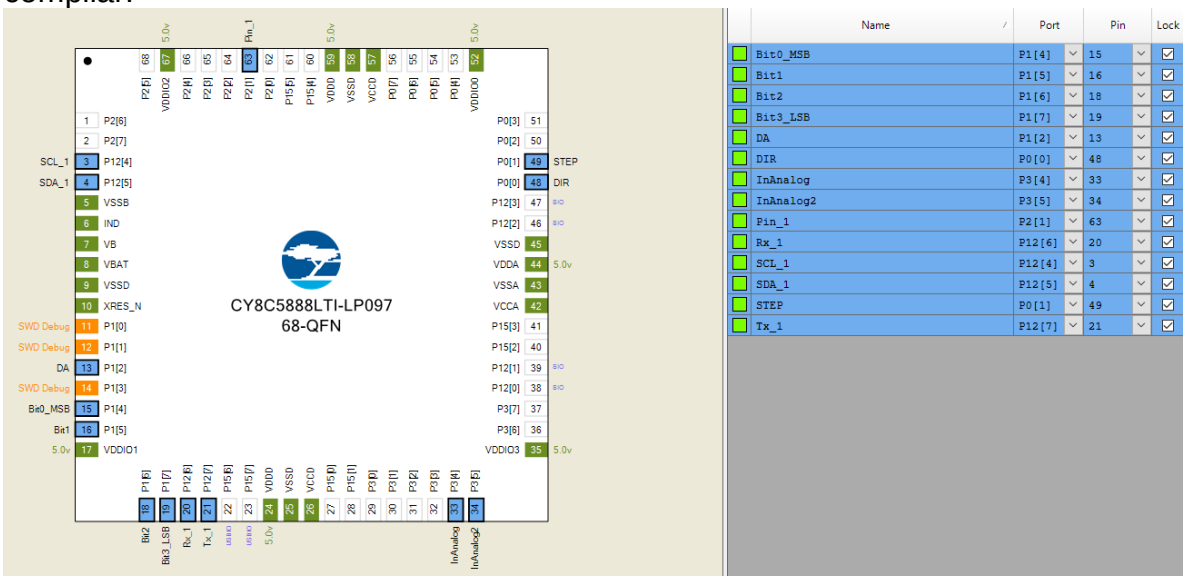

Figura 36: Esquema de trabajo completo

Donde ya unificado todo el código quedo de la siguiente manera ya después de muchas pruebas se logró obtener el siguiente código para que nuestro PSOC haga las tareas requeridas.

```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
    Proyecto Terminal
    Sistema de Medición y control de fuerza de tensado para Fibra Óptica
de Vidrio
    Fabian Josafat Cardenas Aldana
*/

#include "project.h"
#include <stdlib.h>
#include <stdio.h>
/**************I2C*********/
int fila;
int columna;
/**********UART*************/
#define STRBUF_LEN 16
int recibir=0;
int str_index=0;
char str4[STRBUF_LEN+1]={0};
/***Variables para ADC**/
int32 entero[2],voltaje[2];
```

```
float32 volt[2];
float aux[2];
char str[16];
char str2[16];
char str3[16];
char str5[STRBUF_LEN+1];
/**Variables para el Teclado*/
unsigned int rango;
unsigned int tx=0;
unsigned int limite=4;

int bit0_MSB,bit1,bit2,bit3_LSB,cuenta=0,numeroint=0;
char i='\0';
char numero[5];

char8 Keypad_1_Key[4][4]={
    {'1','4','7','*'},
    {'2','5','8','0'},
    {'3','6','9','#'},
    {'A','B','C' ,'D'}
};
/**Valores del Teclado**/
void LeerTecla(){
    bit0_MSB=Bit0_MSB_Read();
    bit1=Bit1_Read();
    bit2=Bit2_Read();
    bit3_LSB=Bit3_LSB_Read();
    if((bit0_MSB==0)&&(bit1==0)&&(bit2==0)&&(bit3_LSB==0))//1
        i=Keypad_1_Key[0][0];
    if((bit0_MSB==0)&&(bit1==0)&&(bit2==0)&&(bit3_LSB==1))//2
        i=Keypad_1_Key[1][0];
    if((bit0_MSB==0)&&(bit1==0)&&(bit2==1)&&(bit3_LSB==0))//3
        i=Keypad_1_Key[2][0];
    if((bit0_MSB==0)&&(bit1==0)&&(bit2==1)&&(bit3_LSB==1))//A
        i=Keypad_1_Key[3][0];
    if((bit0_MSB==0)&&(bit1==1)&&(bit2==0)&&(bit3_LSB==0))//4
        i=Keypad_1_Key[0][1];
    if((bit0_MSB==0)&&(bit1==1)&&(bit2==0)&&(bit3_LSB==1))//5
        i=Keypad_1_Key[1][1];
    if((bit0_MSB==0)&&(bit1==1)&&(bit2==1)&&(bit3_LSB==0))//6
        i=Keypad_1_Key[2][1];
    if((bit0_MSB==0)&&(bit1==1)&&(bit2==1)&&(bit3_LSB==1))//B
        i=Keypad_1_Key[3][1];
    if((bit0_MSB==1)&&(bit1==0)&&(bit2==0)&&(bit3_LSB==0))//7
        i=Keypad_1_Key[0][2];
    if((bit0_MSB==1)&&(bit1==0)&&(bit2==0)&&(bit3_LSB==1))//8
        i=Keypad_1_Key[1][2];
    if((bit0_MSB==1)&&(bit1==0)&&(bit2==1)&&(bit3_LSB==0))//9
        i=Keypad_1_Key[2][2];
    if((bit0_MSB==1)&&(bit1==0)&&(bit2==1)&&(bit3_LSB==1))//C
        i=Keypad_1_Key[3][2];
    if((bit0_MSB==1)&&(bit1==1)&&(bit2==0)&&(bit3_LSB==0))//*
        i=Keypad_1_Key[0][3];
    if((bit0_MSB==1)&&(bit1==1)&&(bit2==0)&&(bit3_LSB==1))//0
        i=Keypad_1_Key[1][3];
    if((bit0_MSB==1)&&(bit1==1)&&(bit2==1)&&(bit3_LSB==0))//#
        i=Keypad_1_Key[2][3];
```

```
    if((bit0_MSB==1)&&(bit1==1)&&(bit2==1)&&(bit3_LSB==1))//D
        i=Keypad_1_Key[3][3];
}
/**Rutina para Leer el teclado**/
CY_ISR(LeerTeclaInt){
    LeerTecla();
    if((i!='A')&&(i!='B')&&(i!='C')&&(i!='D')&&(i!='*')&&(i!='#'))
    {
        if(cuenta<=3)
        {
            numero[cuenta]=i;
            cuenta++;
        }
    }
    if(i=='C')
    {
        numero[0]='\0';
        numero[1]='\0';
        numero[2]='\0';
        numero[3]='\0';
        cuenta=0;
    }
    if(i=='*')
    {
        DIR_Write(0);
        STEP_Write(1);
        CyDelay(10);
        STEP_Write(0);
        CyDelay(100);
    }
    if(i=='#')
    {
        DIR_Write(1);
        STEP_Write(1);
        CyDelay(10);
        STEP_Write(0);
        CyDelay(100);
    }
}
/**Rutina para enviar datos*/
CY_ISR(Enviar)
{
    sprintf(str3,"%d %d ",voltaje[0],voltaje[1]);
    UART_PutString(str3);
    UART_PutString(str4);
    UART_PutString(" \n");
}
/*************************************************************************
***********/
int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    /*********Inicializar Variables*****/
    I2C_Start();
    LCD_I2C_start();
    Opamp_Start();
    Opamp2_Start();
```

```
    pulisr_StartEx(LeerTeclaInt);
    isrEnvio_StartEx(Enviar);
    UART_Start();
    ADC_Start();
    AMux_1_Start();
    /************Inicio de Programa****/
    columna=0;
    fila=0;
    LCD_I2C_setCursor(columna,fila);
    LCD_I2C_print("Ingresa los valores ");
    fila=1;
    LCD_I2C_setCursor(columna,fila);
    LCD_I2C_print("en MiliNewton(mN)");
    CyDelay(600);
    LCD_I2C_clear();
    /*********************************/
    for(;;)
    {
        /****************Recogiendo el valor del arduino*******/
        while(UART_GetRxBufferSize()>=1)
        {
            recibir=UART_GetChar();
            if (str_index < STRBUF_LEN)
            {
                str4[str_index++] = recibir;
            }
        }
        str_index = 0 ;
        //UART_PutString(str4);
        /***************************Imprimir el Valor de la
Galga*******************************/
        columna=0;
        fila=2;
        LCD_I2C_setCursor(columna,fila);
        LCD_I2C_print("mN=");
        LCD_I2C_print(str4);
        LCD_I2C_print("   ");
        /***************************Recoger los valores del
ADC*******************************/
        /***************************Canal
1*************************************************/
        AMux_1_FastSelect(0);
        columna=0;
        fila=3;
        LCD_I2C_setCursor(columna,fila);
        LCD_I2C_print("mV=");

        ADC_StartConvert();
        ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
        entero[0]=ADC_GetResult32();
        ADC_StopConvert();
        voltaje[0]=ADC_CountsTo_mVolts(entero[0]);
        aux[0]=voltaje[0];

        sprintf(str,"%.0f",aux[0]);
        LCD_I2C_print(str);
        LCD_I2C_print("   ");
```

```
        /*************************Canal
2************************************************/
        AMux_1_FastSelect(1);
        columna=10;
        fila=3;
        LCD_I2C_setCursor(columna,fila);
        LCD_I2C_print("mV=");

        ADC_StartConvert();
        ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
        entero[1]=ADC_GetResult32();
        ADC_StopConvert();
        voltaje[1]=ADC_CountsTo_mVolts(entero[1]);
        aux[1]=voltaje[1];

        sprintf(str2,"%.0f",aux[1]);
        LCD_I2C_print(str2);
        LCD_I2C_print(" ");

/*************************************************************************
**************/

/*********************************Teclado*****************************
**************/

/*************************************************************************
**************/
        if((i!='A')&&(i!='C'))//SI no se presiona A ni C
        {
            columna=0;
            fila=0;
            LCD_I2C_setCursor(columna,fila);
            LCD_I2C_print("Ingrese Valor");
            columna=0;
            fila=1;
            LCD_I2C_setCursor(columna,fila);
            LCD_I2C_print("mN=");
            if(cuenta<5)
            LCD_I2C_print(numero);
        }
        if(i=='A')//Si se presiona A
        {
            DIR_Write(0);
            STEP_Write(1);
            CyDelay(10);
            STEP_Write(0);
            CyDelay(100);
        }
        if(i=='B')
        {
            DIR_Write(1);
            STEP_Write(1);
            CyDelay(10);
            STEP_Write(0);
            CyDelay(100);
        }
        if(i=='C')//Si se presiona B
```

```
        {
            i='\0';
            LCD_I2C_clear();
        }
        if(i=='D')//Si se presiona C
        {
            if(tx==0)
            {
                tx=1;
                LCD_I2C_setCursor(0,0);
                LCD_I2C_print(" Transmitiendo  ");
                i='\0';
                /**********************/
                PWM_Start();/*PWM*/
                CyDelay(1000);
                /**********************/
                i='\0';
                LCD_I2C_clear();
            }
            else
            {
                PWM_Stop();
                tx=0;
                i='\0';
                LCD_I2C_clear();
            }
        }
    }
}

/* [] END OF FILE */
```

Para entender nuestro programa nos podemos apoyar con este diagrama de flujo

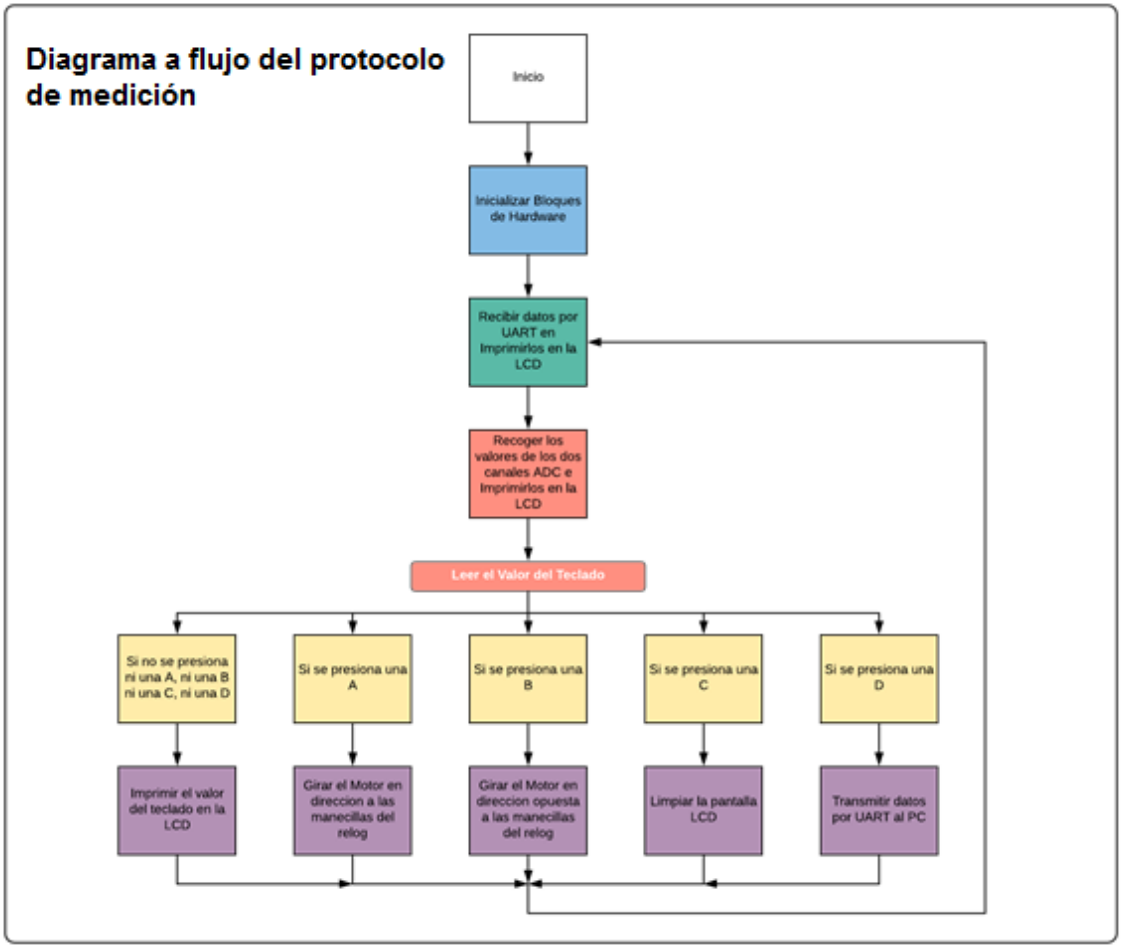

Figura 37: Diagrama de flujo

Implementación, Calibración de la galga y Comunicación con el PSOC

Después nos enfocamos en la parte de la galga para esta parte se use en otro microcontrolador que es el ATMEGA328 la función de este microcontrolador es recibir los datos entregados por el integrado HX711 y mandarlos a puerto UART de nuestro PSOC, para esto se conectó la galga, se descargó la librería, y se calibro la galga extensométrica y se desarrolló un programa para envió por UART del ATMEGA328 al PSOC para esto fuimos al IDE de arduino

El siguiente diagrama muestra la conexión de la galga y la conexión al PSOC

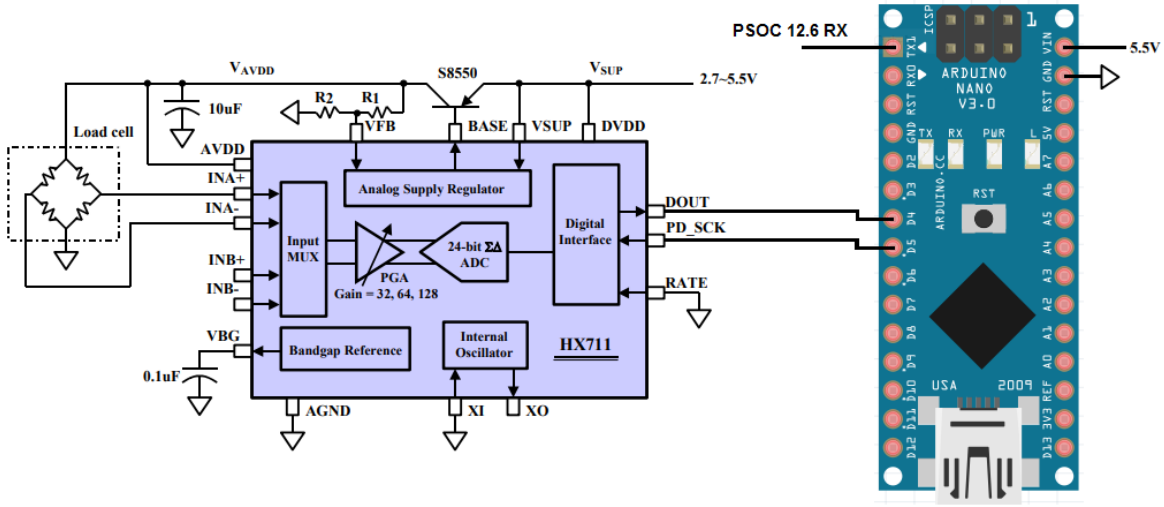

Figura 38: Diagrama de conexión del HX711

Hay que aclarar que el VCC solo se va únicamente cuando se termine de calibrar la galga y cuando se cargue el programa que mandara los datos al PSOC. Una vez conectado el diagrama procedió a descargar la librería en nuestro IDE de Arduino para esto nos vamos a “Programa” después “Incluir Librería “y por último “Gestionar Librerías” se nos abrió una ventana como esta y Buscamos HX711 y se seleccionó y se instaló la librería HX711_ADC de Olav Kallhovd.

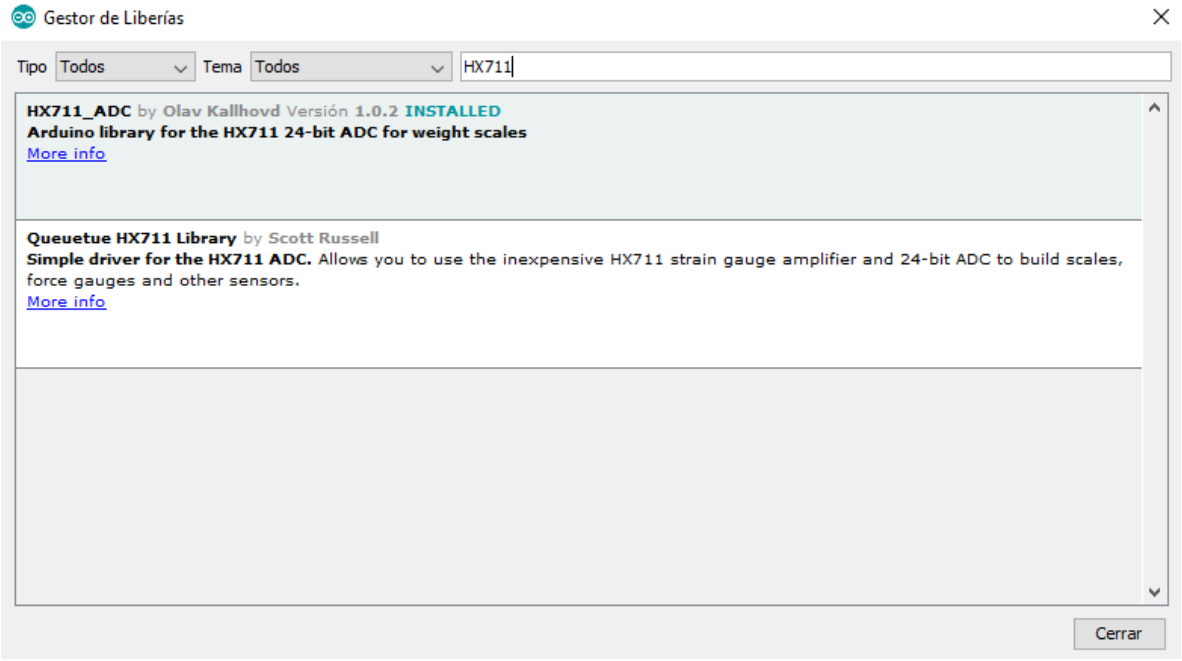

Figura 39: Agregando Liberia HX711

Ahora primero se tuvo que calibrar la galga para eso fuimos a nuestro entorno y luego vamos a “Archivo” después “Ejemplos” y después HX711_ADC y seleccionamos el sketch que dice “Calibrate” si todo está bien conectado podremos

compilar y subir el archivo al microcontrolador abrimos el monitor serial y nos aparecerá algo como esto.

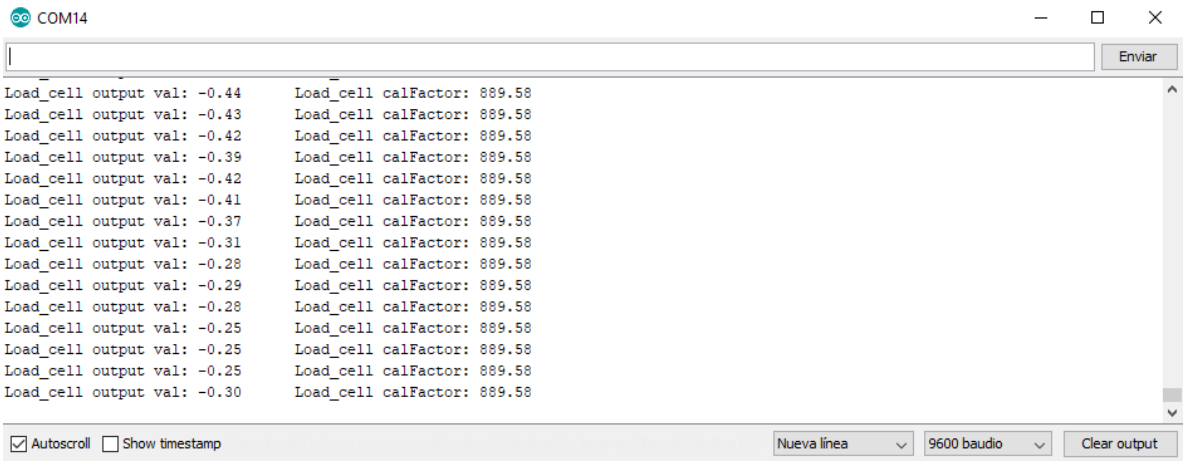

Figura 40: Calibrando la galga

Para la calibración de la galga se le puso al principio sin ningún peso y se le mando la letra “t” para poner en 0 el valor de salida hecho esto se le puso un objeto de peso conocido y por el monito serial se le enviará los siguientes valores:

- Enviar “l” desde el monitor seria disminuye el factor en 1.0
- Enviar “L” desde el monitor seria disminuye el factor en 10.0
- Enviar “h” desde el monitor seria incremente el factor en 1.0
- Enviar “H” desde el monitor seria incrementa el factor en 10.0

Se jugó con estos valores hasta que el número que da “Load_cell output val:” sea igual al peso del objeto una vez logrado esto se guardó el valor del cellFactor en un bloc de notas y ahora mostraremos un código modificado de otro ejemplo de la galga

```
//-------------------------------------------------------------------------------------
// HX711_ADC.h
// Arduino master library for HX711 24-Bit Analog-to-Digital Converter for Weigh Scales
// Olav Kallhovd sept2017
// Tested with      : HX711 asian module on channel A and YZC-133 3kg load cell
// Tested with MCU  : Arduino Nano
//-------------------------------------------------------------------------------------
// This is an example sketch on how to use this library
// Settling time (number of samples) and data filtering can be adjusted in the HX711_ADC.h file

#include <HX711_ADC.h>

//HX711 constructor (dout pin, sck pin)
HX711_ADC LoadCell(4, 5);
```

```
long t;
char buffer[10]="  ";
int gramos;
float absoluto;
void setup() {
  Serial.begin(115200);
  //Serial.println("Wait...");
  LoadCell.begin();
  long stabilisingtime = 2000; // tare preciscion can be improved by adding a few seconds of stabilising time
  LoadCell.start(stabilisingtime);
  LoadCell.setCalFactor(1688.00); // user set calibration factor (float)
  //Serial.println("Startup + tare is complete");
}
void loop() {
  //update() should be called at least as often as HX711 sample rate; >10Hz@10SPS, >80Hz@80SPS
  //longer delay in scetch will reduce effective sample rate (be carefull with delay() in loop)
  LoadCell.update();
  //get smoothed value from data set + current calibration factor
  if (millis() > t + 250) {
    int i = LoadCell.getData();

    //Serial.print("Load_cell output val: ");
    gramos=fabs(i);
    Serial.print(gramos);
    Serial.print("\n");
    //Serial.println(buffer);
    t = millis();
    delay(500);
  }
  //receive from serial terminal
  if (Serial.available() > 0) {
    float i;
    char inByte = Serial.read();
    if (inByte == 't') LoadCell.tareNoDelay();
  }
  //check if last tare operation is complete
  if (LoadCell.getTareStatus() == true) {
    //Serial.println("Tare complete");
  }
}
```

El valor que guardamos lo ponemos en lugar del valor que puesto en la línea de código en color rojo de esta forma ya tenemos listo nuestro programa. Importante mencionar si queremos mirar por el monitor serial lo que envía el programa cambiar la velocidad de baudios a 115200.

Análisis y discusión de los resultados

En las Referencias se ha dejado un software para la pc de muy fácil uso para recibir los datos hecho Visual Studio dicho software ya está configurado para recibir los datos envía nuestro PSOC solo es cuestión de instalar los drivers (puesto en las referencias y la instalación es muy fácil solo es seguir los pasos) y para abrir la aplicación descomprimir el archivo y ubicar la siguiente dirección \puerto\puerto\bin\Debug ejecutar el archivo puerto.exe después seleccionar el puerto COM en el que está conectado el USB serial una vez seleccionado dar clic en conectar y listo ya estaremos recibiendo datos cuando se deje de usar solo dar clic en botón de desconectar y abajo tenemos un cuadro de texto para enviar datos y un botón más para enviar datos al PSOC desde la computadora.

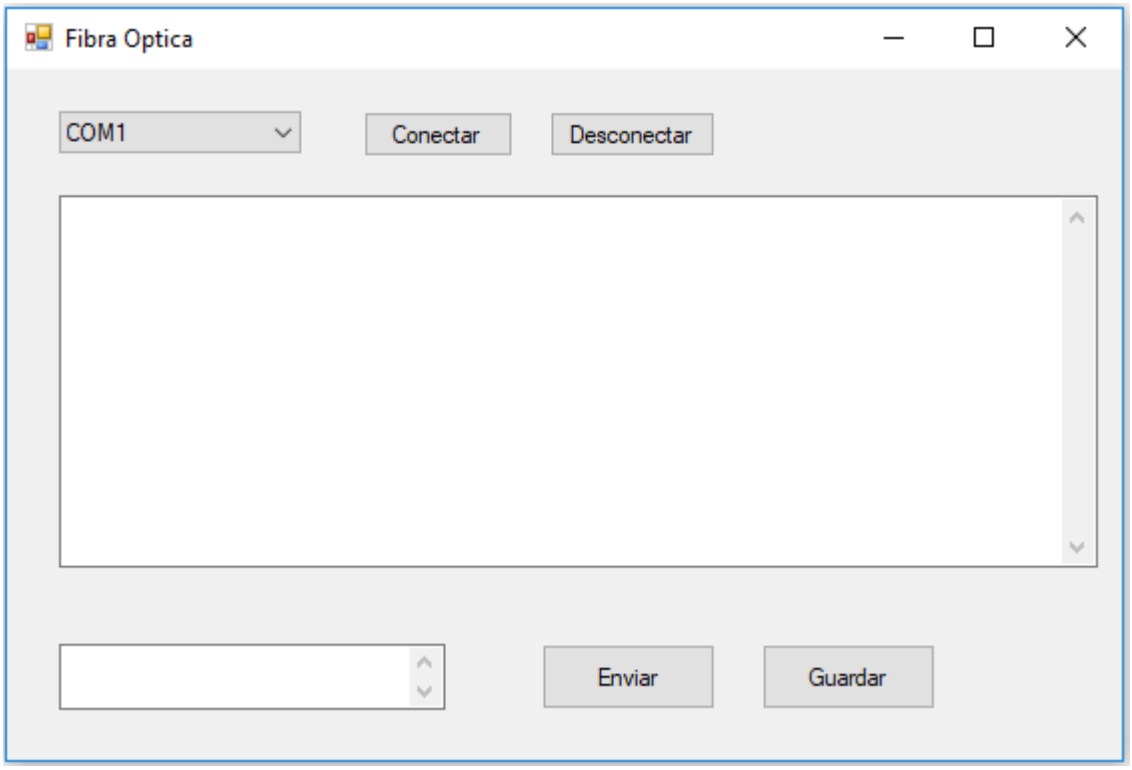

Figura 41: Software para recibir datos en la computadora.

Hasta aquí tenemos desarrollado nuestro dispositivo final de este proyecto es capaz de estirar la fibra óptica. Un video con el funcionamiento de este dispositivo se muestra en el siguiente vinculo asociado a la imagen.

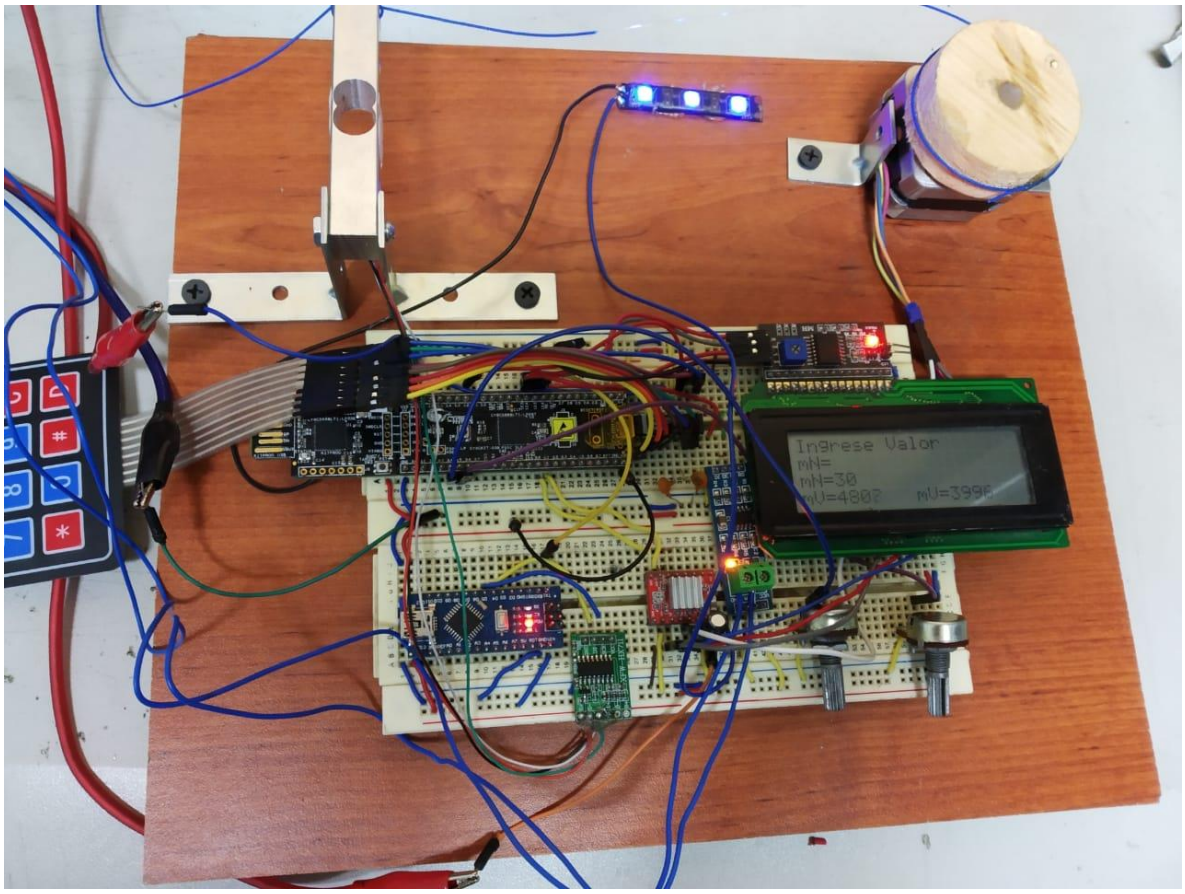

Figura 42: Sistema Completo

Como se puede observar en el video del funcionamiento de la etapa final de nuestro proyecto, se consiguió realizar el diseño e implementación de un sistema de tensión de fibra óptica el cual tiene la capacidad de medir y mandar los datos por bus485 y se le adicionaron dos entradas analógicas por si el usuario requiere adicionar otro tipo de sensor y pueda ser conectado fácilmente a nuestro diseño.

La cantidad de hardware digital utilizado para la implementación de este proyecto en nuestro PSOC fue baja dado que se utilizó menos del 25% eso quiere decir que nuestro PSOC aún puede realizar otras tareas más y este programa puede ser modificado para que sea de mayor utilidad:

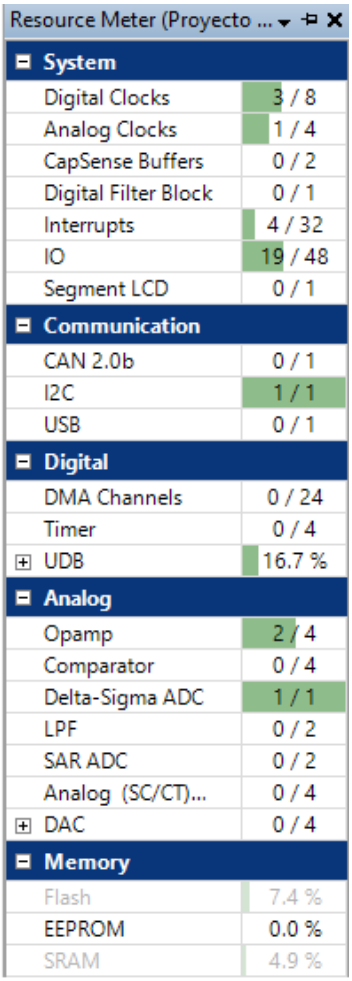

Figura 43: Reporte de los recursos Utilizados dentro del PSOC 5

Se hicieron muchas modificaciones y experimentos con varias plataformas de hardware para obtener nuestro resultado final sin embargo todo esto permitió tener el dispositivo que se tiene actualmente.

Conclusiones

En conclusión, el dispositivo actualmente desarrollado cumple con las características planteadas en un inicio, aunque se presentaron muchos problemas de software y hardware en este proyecto se logró obtener un resultado satisfactorio y además se logró obtener experiencia en el diseño y desarrollo de sistema embebido, así como la interconexión de 2 plataformas de desarrollo.

Referencias Bibliográficas:

[1] H. Jardón and R. Linares y Miranda, Sistemas de comunicaciones por fibras ópticas. México: Alfaomega, 1995.

[2] B. Rubio, Introducción a la ingeniería de la fibra óptica. Madrid: Ra-Ma, 1994.

[3] M. Ilyas and H. Mouftah, The handbook of optical communication networks. Boca Raton: CRC Press, 2003.

[4] G. G. Pérez, J. A. Mejia, E. A. Andrade, J. R. Pérez, "Elongation-based fiber optic tunable filter", Proc. SPIE 10404, Infrared Sensors, Devices, and Applications VII, 1040412 (7 September 2017). Disponible en: https://doi.org/10.1117/12.2274650.

[5] Cypress Semiconductors, CY8C58LP Family Datasheet, Available: http://www.cypress.com/file/45906/download.

[6] Raman Kashyap, “Fiber Bragg Gratings”, Academic Press, 1th ed, U.S.A, 1st March 1999.

Referencias a herramientas y Software utilizado:

PSOC Creator disponible en: http://www.cypress.com/products/psoc-creator-integrated-design-environment-ide

Arduino disponible en: https://www.arduino.cc/en/Main/Software

Proyecto PSOC disponible en: https://github.com/josafath597/PSOC