24-1-2020

Prototipo de control Robot

Dinámica y control de robots



Brian Oswaldo Ramos Chavez

Diego Armando Becerra Iñiguez

Lisbeth Martínez Velázquez

Nadia Saraí Murguia Chavez

Mario Alcalá Villa Gómez

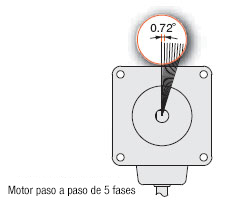
ing.mecatrónica 8-at/m prof: carlos garabito

**Marco Teórico**:

**Control de proporción de un motor nema 17**

Los motores paso a paso se usan para obtener un posicionamiento preciso por medio de un control digital. El motor opera al sincronizarse con precisión con la salida de señal de pulsos del controlador al excitador. Los motores paso a paso, con su capacidad para producir un par elevado a baja velocidad mientras se minimiza la vibración, son ideales para aplicaciones que requieren un posicionamiento rápido en una distancia corta.

Los motores paso a paso permiten un posicionamiento preciso con facilidad. Se usan en diversos tipos de equipos para obtener un ángulo de rotación y un control de velocidad precisos utilizando señales de pulsos. Los motores paso a paso generan un par de alto valor con un cuerpo compacto, y son ideales ya que aceleran y responden con rapidez. Los motores paso a paso mantienen también su posición al detenerse, debido a su diseño mecánico. Las soluciones de motor paso a paso constan de un excitador (recibe señales de pulsos y las convierte en movimiento del motor) y un motor paso a paso. Oriental Motor ofrece muchas soluciones para una amplia variedad de aplicaciones.

Un motor paso a paso gira con un ángulo en paso fijo, igual que el segundero de un reloj. A este ángulo se le llama “ángulo de paso básico”. Oriental Motor ofrece motores paso a paso con un ángulo de paso básico de 0.36°, 0.72°, 0.9° y 1.8°.

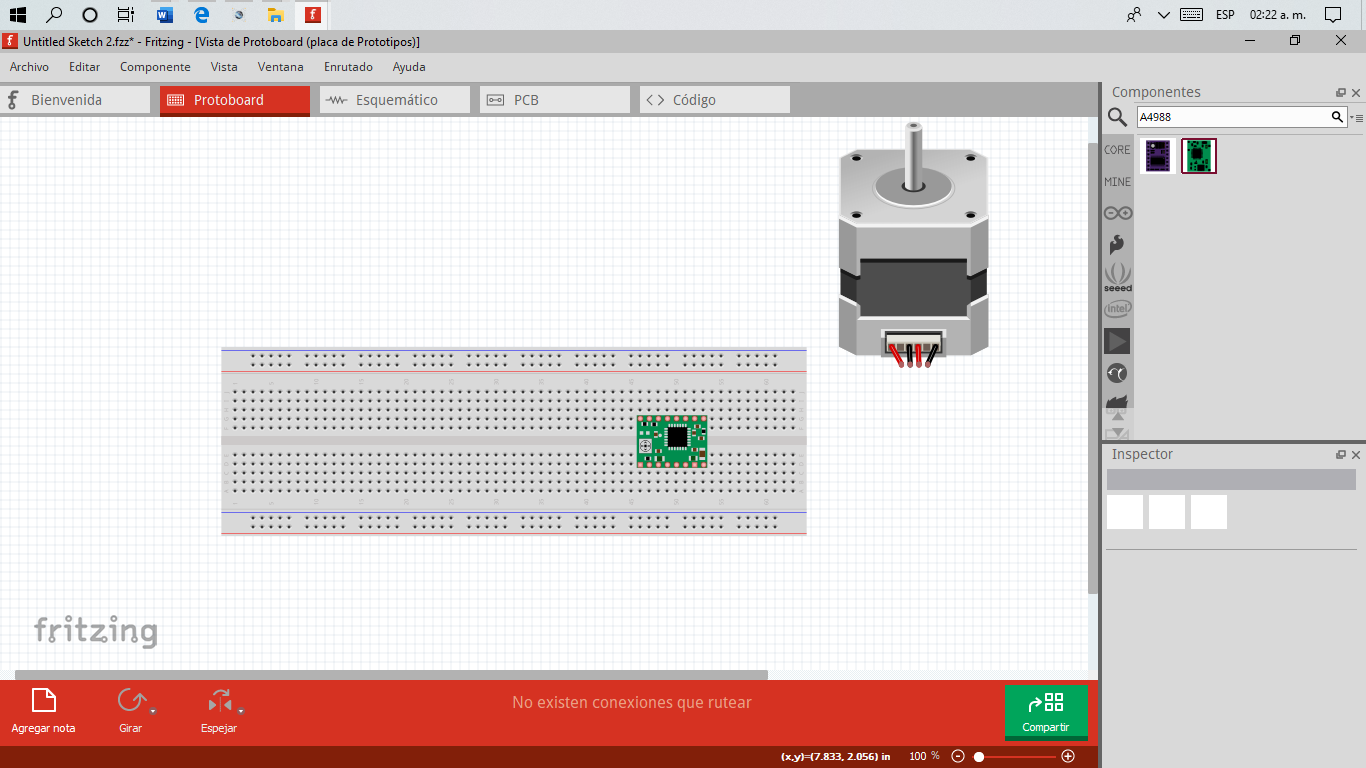
En esta sección se puede seleccionar uno de los tres tipos de movimiento (modo de excitación) para los que está preparado el sistema.

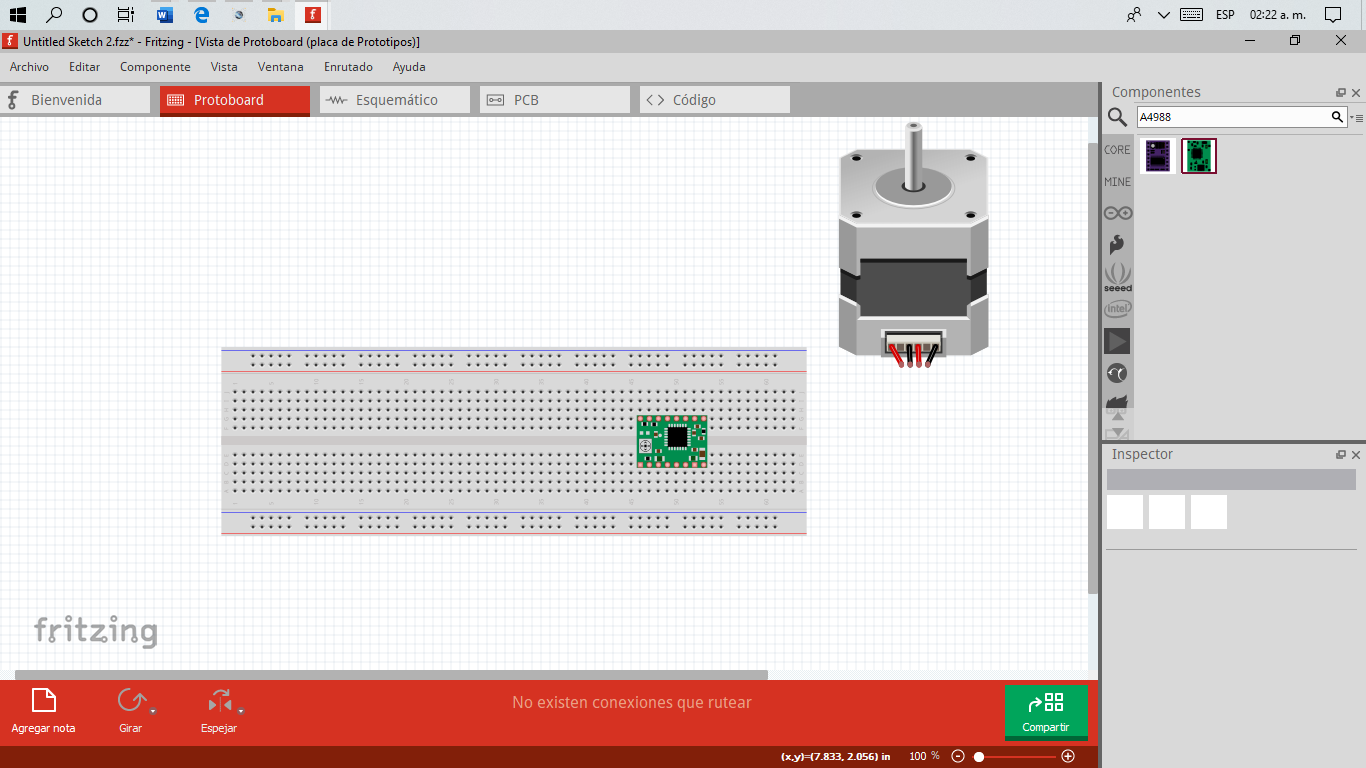
• Paso entero (dos fases): En este modo de excitación después de cada conmutación siempre resultan estar excitadas las dos fases

• Paso entero (una fase): En este modo de excitación, en cada secuencia de comunicación solamente una fase está excitada

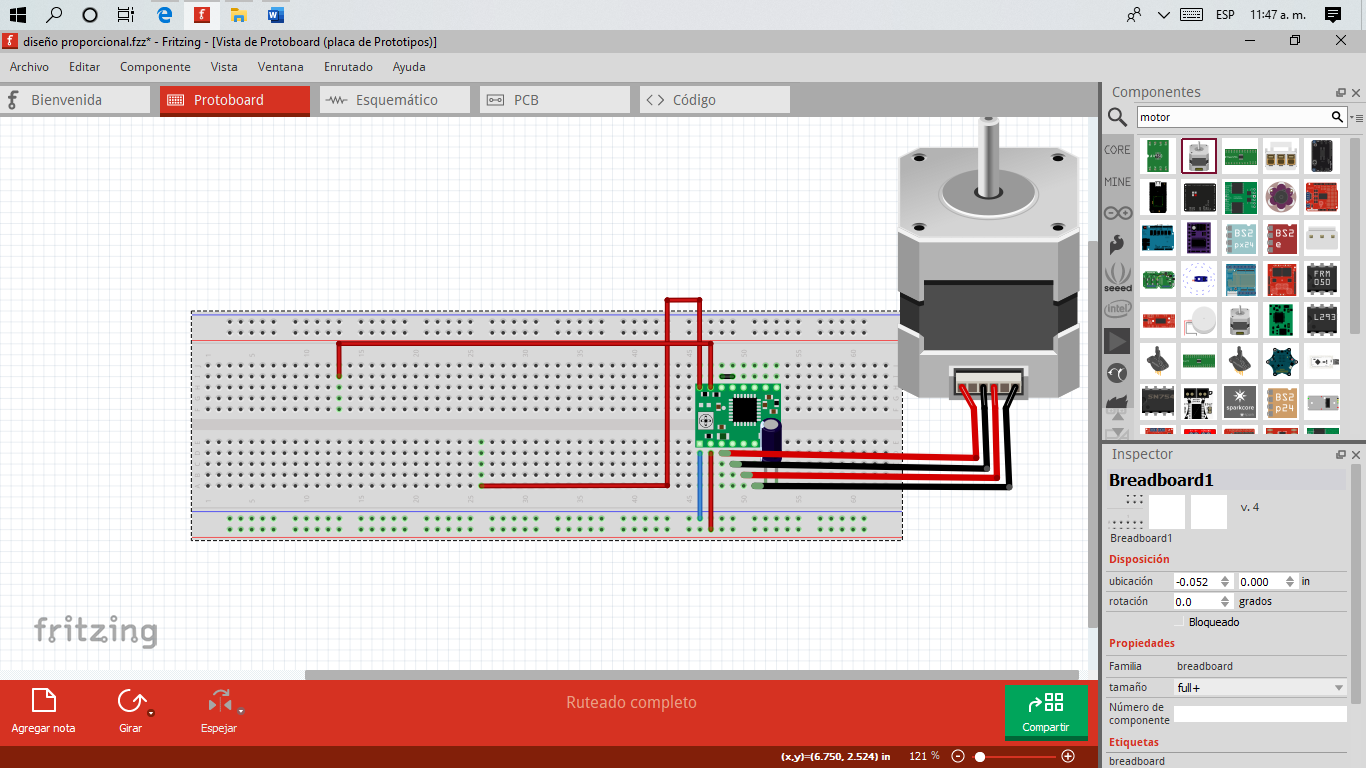
• Medio paso: En este modo de excitación el motor se desplaza en cada conmutación la mitad del ángulo de paso

Se debe mencionar que los pasos de aceleración y deceleración no tienen por qué ser los mismos, el motor paso a paso puede configurarse para tener elevada aceleración y lenta deceleración o viceversa. Algo parecido ocurre con las corrientes por la fase A o B, se pueden introducir diferentes corrientes en cada fase, pero se debe tener en cuenta que funcionando en modo paso entero con dos fases, si las corrientes son diferentes, el funcionamiento no será correcto. Hay que asegurarse que en cualquier caso la corriente que circula por cada fase sea, suficiente para superar el par que el motor deba generar para mover la carga. Estas variables también tienen la peculiaridad de que los valores que se pueden introducir están restringidas, con lo cual no es posible introducir valores incoherentes o fuera de los márgenes que se han determinado:

* Agregamos un motor nema con su driver A4988 para su control del mismo y empezamos agregar los componentes.
* Colocamos la tarjeta psoc para hacer la simulación y dicho esta la conexión para que tenga un funcionamiento.

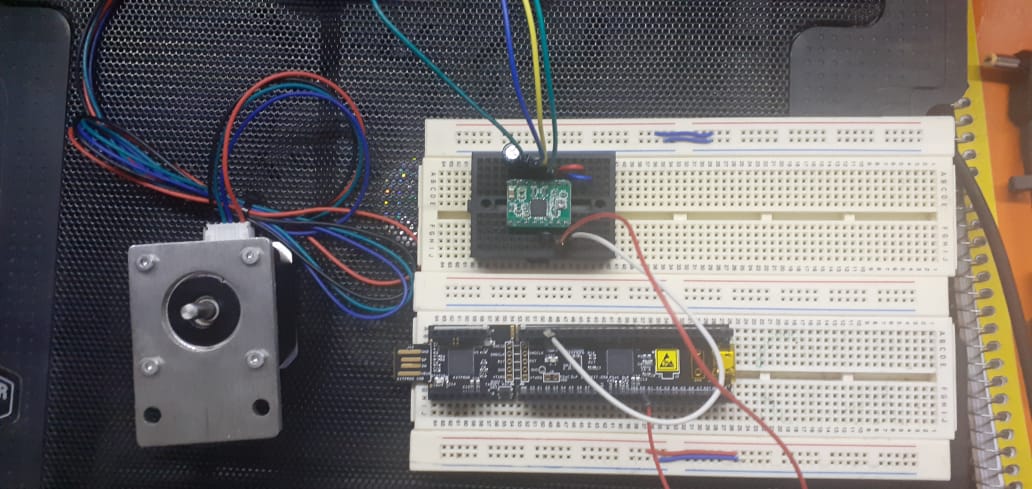




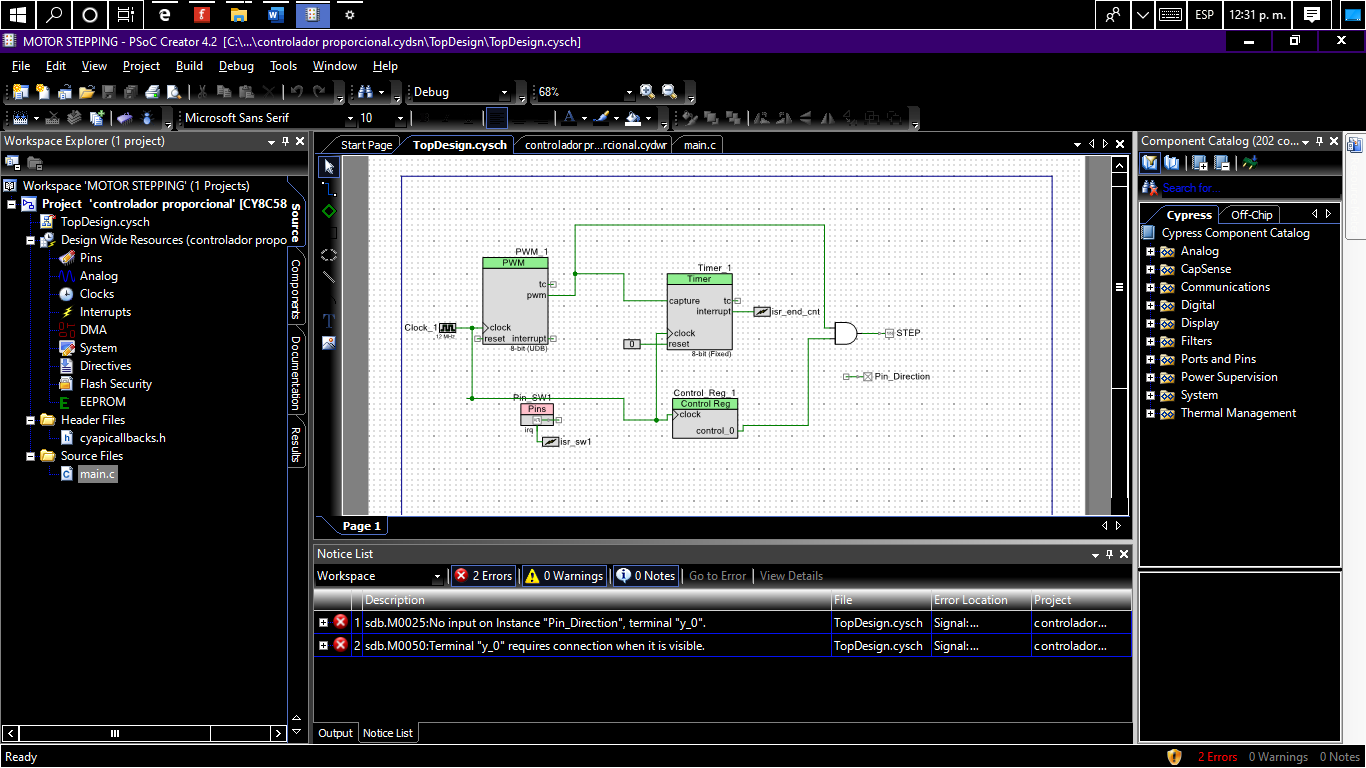
* Se realizan las conexiones correspondientes del circuito:



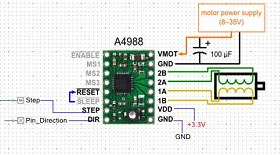
* Conectamos físicamente el circuito y realizamos la simulación y mas adelante se mostrara la programacion respectivamente.

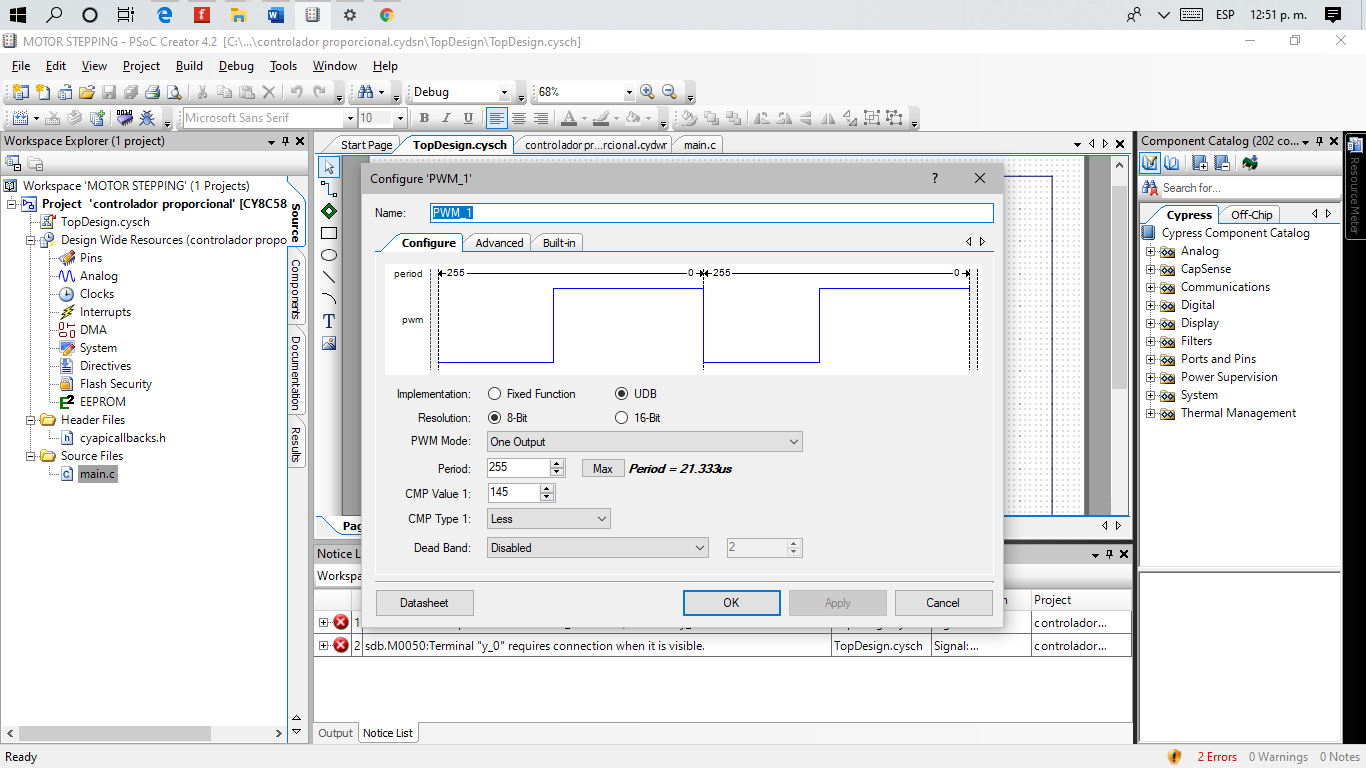


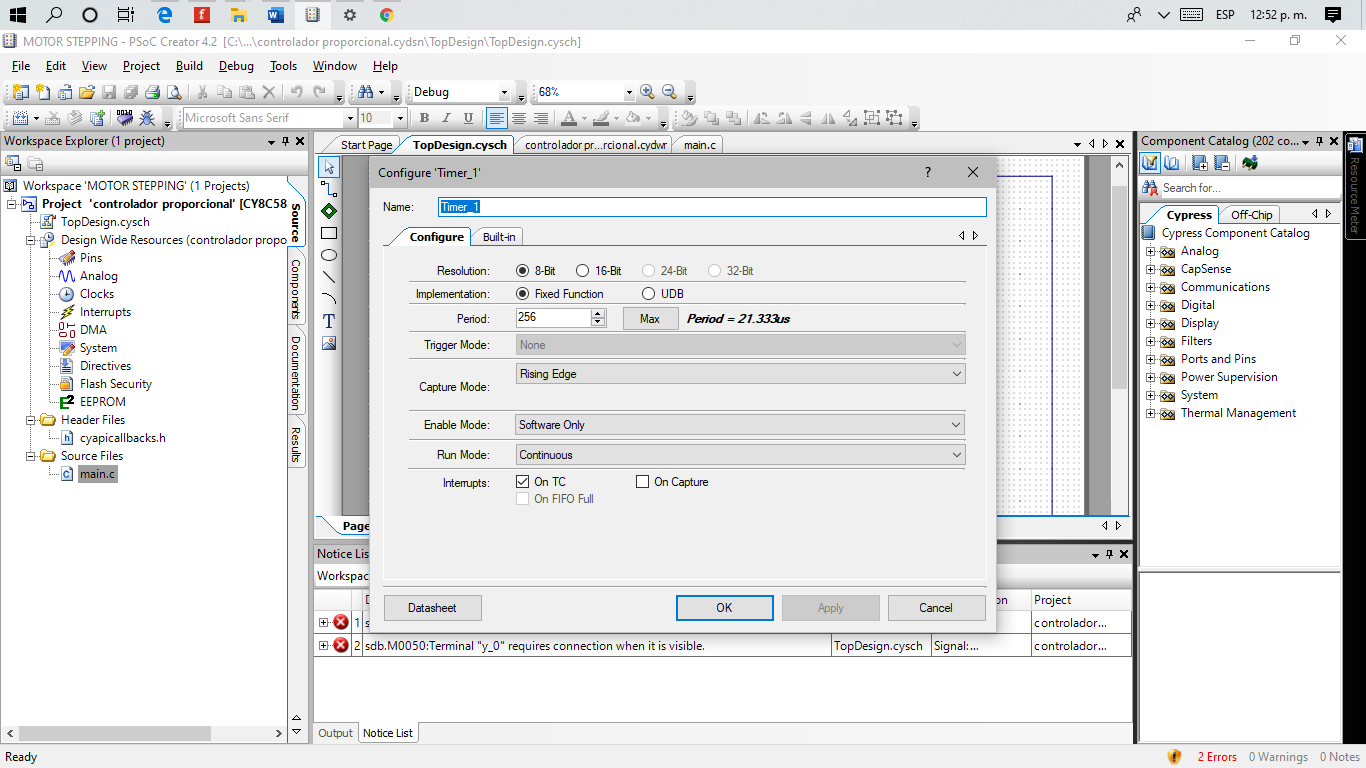
* Realizamos las conexiones obteniendo un pwm un timer y un control regulable para el sistema entre lazado, conectando en si interrupciones con la finalidad de inicializar y finalizar el ciclo de la prueba del control proporción del sistema del motor paso a paso.

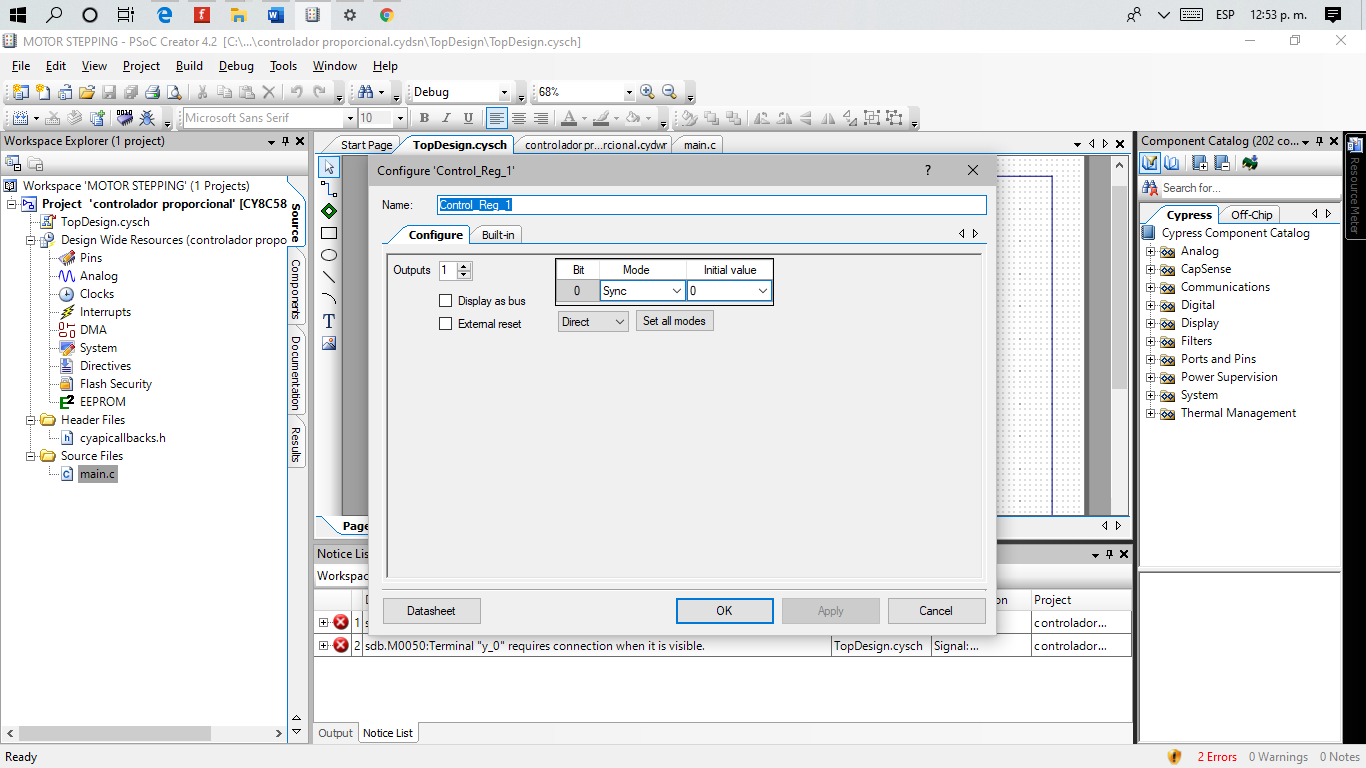


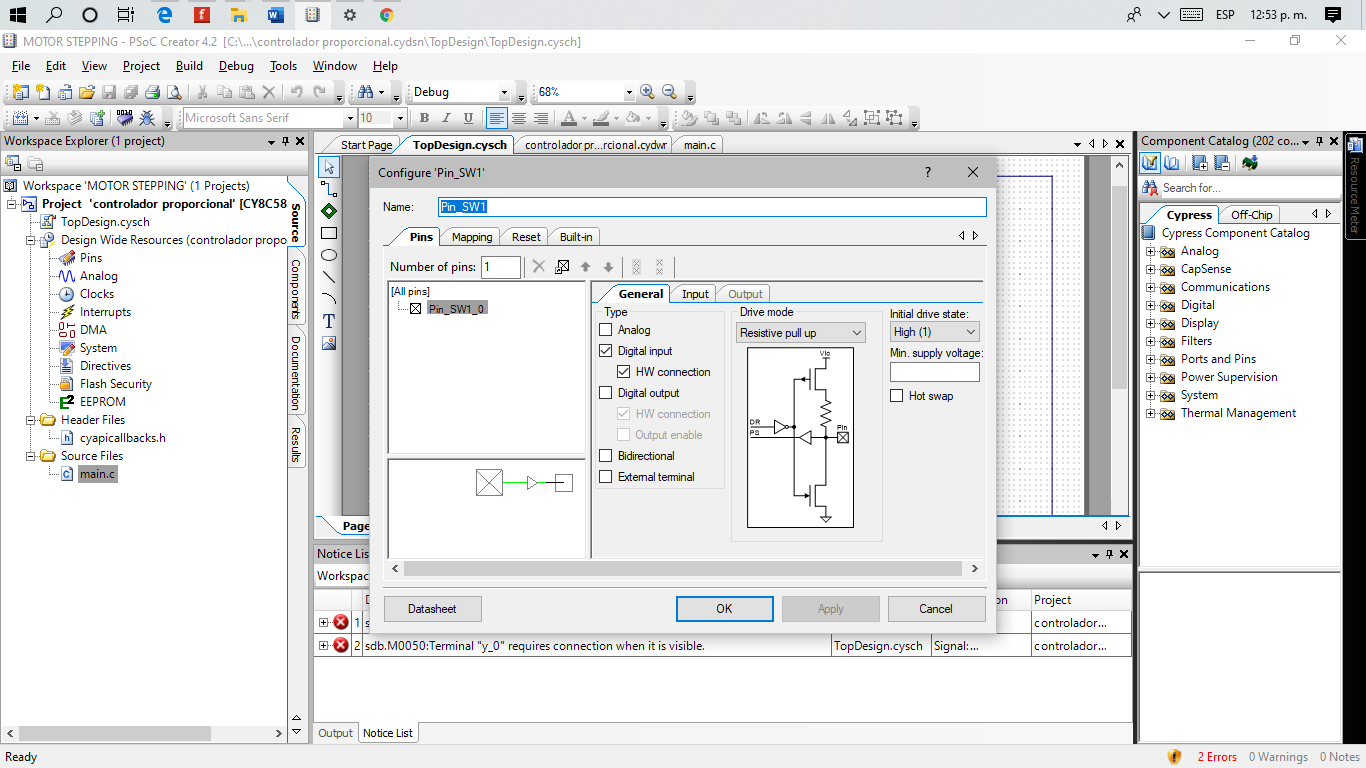
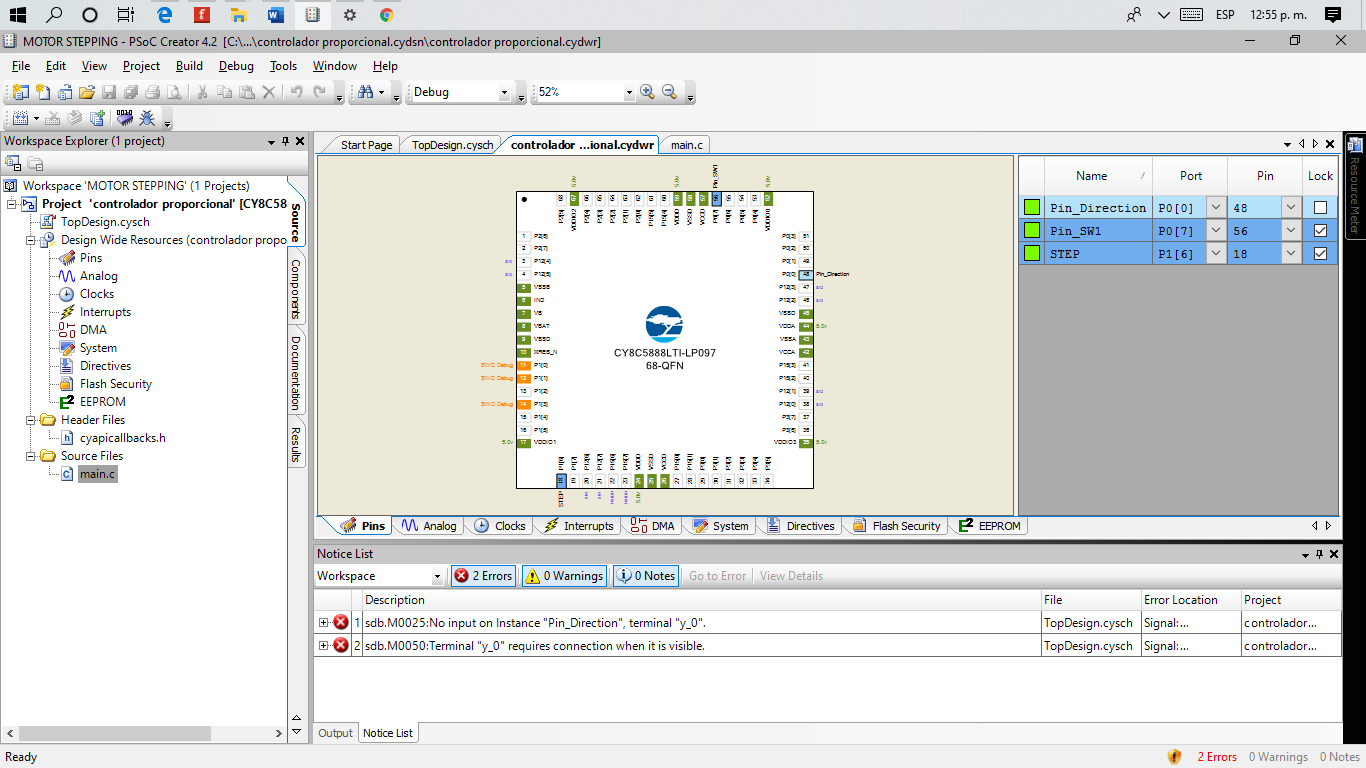
* **Físicamente se conecta el driver que hara que tenga conectado la psoc con el driver que se mueva que es fundamental para el funcionamiento del sistema de control de proporcionalidad.**
* **La frecuencia de Clock\_1 se divide por PWM\_1 de acuerdo con la velocidad de rotación establecida. En el terminal "line\_n" tendremos una onda cuadrada con la frecuencia deseada con un ciclo de belleza del 50%.**
* **TIMER\_1 se encarga de contar el número de impulsos emitidos por pwm y generar una interrupción (ISR\_end\_cnt) que alcanzó el número establecido.**
* **Control\_Reg\_1 se encarga de habilitar o no, a través de la puerta AND, los pulsos generados por el PWM en el pin "Paso", por lo tanto, hacia el controlador A4988.**
* **Clock\_1 está conectado a todos los componentes del circuito para su sincronización.**
* **La tecla Pin\_SW1 a través de otra interrupción (ISR\_sw1) arranca el motor.**





* Aquí se colocó una resolución de 8-bit y un periodo de 255 como máximo y con el valor que se comparan porque serán modificados por la programacion.
* En el timer se modifico con el periodo de 256 que teníamos en el pwm con 8-bit e igual manera hace la comparación para que el sistema tenga un funcionamiento.



* En el control regulable lo colocamos en sync y con un valor de 0 para su valor inicial con una salida de clock controlable dentro del sistema.
* Colocamos un switch que viene desde la psoc internamente en pull-up ya que este hara el accionamiento del motor la iniciación del mismo.
* Como se muestra en el top-desing se le asignaron los pines respectivamente para que el sistema sea funcional.
* **Con la herramienta para la gestión de los pines, Pin\_SW1 se asigna al puerto P0.7 que corresponde al conmutador en el kit. Al pin "Paso" he asignado el puerto P1.6 que corresponde al LED del kit para que pueda actuar como monitor. Pin\_Direction fue asignada automáticamente por la herramienta al puerto P0.0 y no la he modificado**.

**Programacion:**

El cuerpo principal asocia las rutinas de interrupción e inicializa los componentes. Dentro del bucle infinito for, el motor gira dos veces en ambas direcciones de rotación. Tenga en cuenta que la interrupción para el botón SW1 se deshabilita durante la rotación y se vuelve a habilitar al final de las dos vueltas. Esto es necesario porque la rotación del motor es independiente de la CPU. la interrupción generada por el contador Timer\_1 informará al firmware que el motor se ha detenido, con el indicador "IsRunning".

int main ()

{

CyGlobalIntEnable; / \* Habilitar interrupciones globales. \* /

ISR\_end\_cnt\_StartEx (My\_end\_cnt\_Handler);

ISR\_sw1\_StartEx (My\_sw1\_Handler);

Clock\_1\_Start ();

PWM\_1\_Start ();

Timer\_1\_Start ();

para (;;)

{

si (botón == 1)

{

if (vueltas == 2)

{

vueltas = 0;

ISR\_sw1\_Enable ();

// volver a habilitar la tecla pulsar interrupción

botón = 0;

}

si no (isRunning == 0)

{

CyDelay (500);

ISR\_sw1\_Disable ();

// deshabilita la tecla pulsar interrupción

runStepper (199, dirección, velocidad);

dirección = (dirección == 0)? 1: 0;

convierte + = 1;

velocidad + = 5;

}

}

}

* Como se mencionó, la interrupción de Timer\_1 (My\_end\_cnt\_Handler) baja el indicador IsRunning y deshabilita el pin "Paso", a través del registro de Control.

CY\_ISR\_PROTO (My\_end\_cnt\_Handler);

CY\_ISR (My\_end\_cnt\_Handler)

{

// interrumpir la rutina de gestión al final del tren de pulsos

Control\_Reg\_1\_Write (0); // deshabilita el pin 'step' a través del puerto y

isRunning = 0;

Timer\_1\_ClearInterrupt (Timer\_1\_INTR\_MASK\_TC);

};

La interrupción generada por la tecla SW1 (My\_sw1\_Handler) no hace más que levantar el indicador "botón", para que el programa pueda manejar el evento. De hecho, es una buena regla tener pocas funciones complejas realizadas dentro de las rutinas de interrupción.

CY\_ISR\_PROTO (My\_sw1\_Handler);

CY\_ISR (My\_sw1\_Handler)

{

botón = 1;

Pin\_SW1\_ClearInterrupt ();

};

La función "RunStepper" se encarga de establecer los parámetros correctos para el PWM y el temporizador. Establece la dirección de rotación. Active el pin "Paso" por medio del registro de control y levante el indicador "isRunning".

void runStepper(uint16 numberOfSteps, uint8 dir, uint16 motorSpeed)

{

isRunning = 1;

uint16 tempvar = PWM\_Period(motorSpeed);

Pin\_Direction\_Write(dir);

Control\_Reg\_1\_Write(1);

// porta and: para abilitar del pin 'step'

PWM\_1\_WritePeriod(tempvar);

PWM\_1\_WriteCompare(tempvar / 2);

// cycle al 50%

PWM\_1\_WriteCounter(0);

// necessario resettare a 0 il contatore

Timer\_1\_WritePeriod(numberOfSteps);

// impuesto el timer

Timer\_1\_WriteCounter(0);

// necessario resetear

}

El cálculo del período con el que establecer el PWM se delega a la función PWM\_Period. Con el reloj configurado en 6khz, el período más alto para la onda cuadrada generada por el PWM será igual a 1 / 6k x 65535 = 10.9 segundos, mientras que el valor máximo será de 3khz. Sería posible aumentar el rango también trabajando con el preescalador PWM y variando la frecuencia del reloj. No he implementado esta posibilidad para simplificar el código.

uint16 period;

period = (MAX\_FREQ / ((rpm \* MOTOR\_STEPS) / 60) );

if (period > 65535) period = 65535; //maximo valor de division del PWM

else if (period < 2) period = 2; //minimo valor de divion del PWM

return period;

};

}

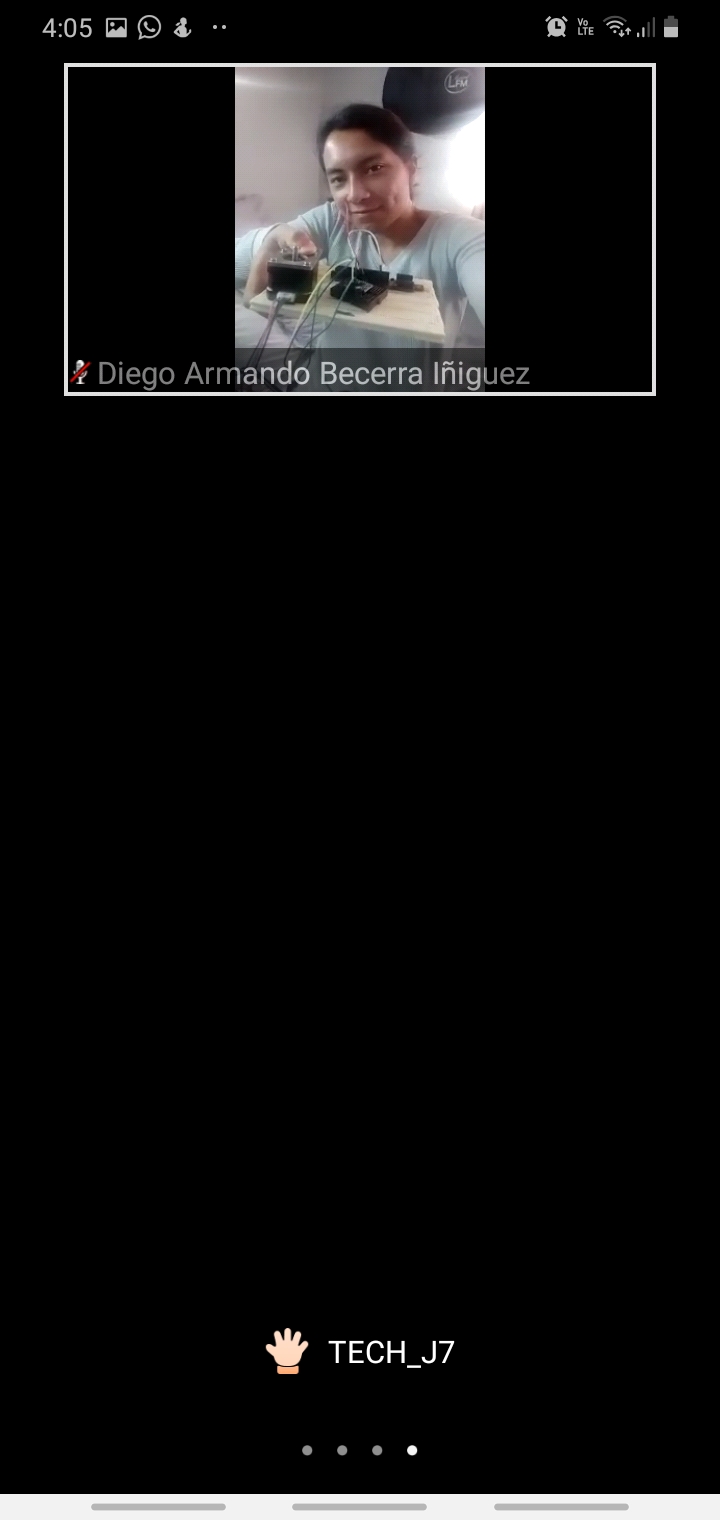
}

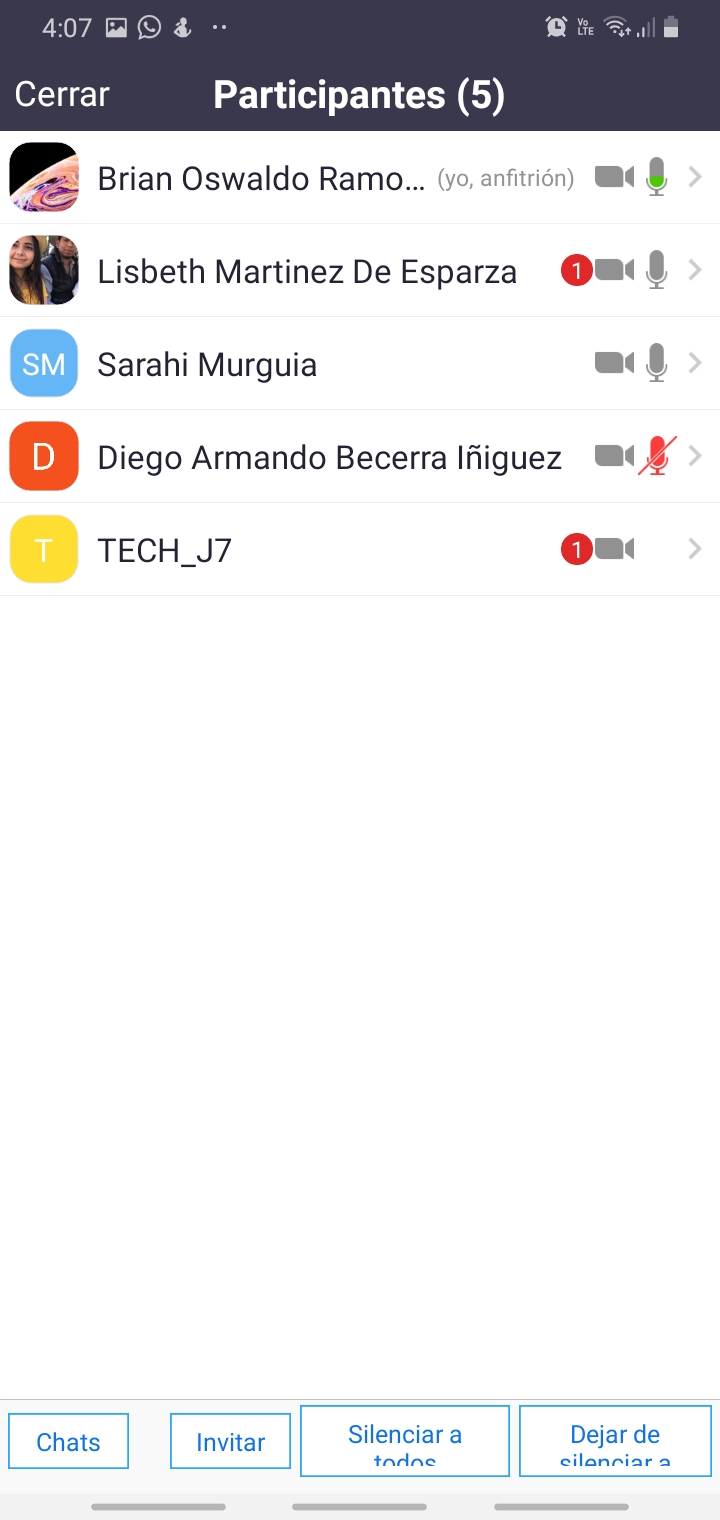
}

}

**“Reunión de trabajo en equipo”**







**Conclusión:**

En esta práctica hemos observado otro funcionamiento de la psoc. Utilizamos lo que son los pre empaquetados utilizando los bloques (UDB).

Existe un video en cypress donde podemos comenzar a trabajar con la psoc ya que existen varias complicaciones, una de ellas fue el código al presentar un error de punto y coma, por eso ya no quería compilar el programa.

Hay que estar revisando detalle tras detalle en las practicas, ya que, si no realizamos ese paso, puede que no podamos avanzar.

En fin, una práctica difícil, pero se llegó a terminar completa.

***Lisbeth Martínez Velázquez***