INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

DIVISIÓN:
INGENIERÍA MECATRÓNICA

MATERIA:
MICROCONTROLADORES

DOCENTE:
JOSE ANGEL NIEVES VAZQUEZ

UNIDAD 5:
PROGRAMACIÓN DEL MÓDULO CCP DEL MICROCONTROLADOR.

INVESTIGACION DE LA UNIDAD 5

ALUMNOS:
ISRAEL ANTONIO LÓPEZ ESCRIBANO
JESUS ALEJANDRO ROSAS ROSAS
ARGELIO SANTIAGO REYES

GRUPO:
711-B

FECHA DE ENTREGA:
San Andrés Tuxtla Ver. 01 de Diciembre de 2025

Introducción:

Para este documento realizamos el análisis del funcionamiento de la programación de los módulos CCP de los microcontroladores este sistema requiere una alta precisión en la temporización y el control de eventos a nivel de hardware. Los microcontroladores, como componentes centrales de estos sistemas, incorporan periféricos internos diseñados específicamente para estas tareas. el Módulo de Captura, Comparación y PWM (CCP) destaca como una herramienta fundamental para interactuar de manera eficiente con el entorno físico.

La programación correcta y eficiente del Módulo CCP es un factor determinante para el rendimiento y la fiabilidad de las aplicaciones en áreas como la robótica, el control industrial, y los sistemas de adquisición de datos. Un uso inadecuado puede resultar en causante de fallos temporales, consumo excesivo de recursos o fallos en el sistema. En esta investigación documental se comenzará con una revisión de los conceptos de temporización y microcontroladores, para luego detallar el proceso de programación del módulo CCP y la presentación de los resultados obtenidos de la implementación práctica.

Además, también se anexan imágenes representativas demostrando como se realizan este tipo de conexiones mediante los diagramas de bloques, de igual forma algunos diagramas cuenta con el ejemplo de cómo se realiza su programación, tablas que muestran las frecuencias a las que se pueden manejar entre otras cosas sin más por el momento pasamos a mostrar el contenido en el documento.

5.1 Descripción del módulo CCP.

El módulo CCP (Captura/Comparación/PWM) es un periférico que le permite medir y controlar diferentes eventos. El modo de captura proporciona el acceso al estado actual de un registro que cambia su valor constantemente. En este caso, es el registro del temporizador Timer1. El modo de comparación compara constantemente valores de dos registros. Uno de ellos es el registro del temporizador Timer1. Este circuito también le permite al usuario activar un evento externo después de que haya expirado una cantidad de tiempo predeterminada. PWM (Pulse Width Modulation - modulación por ancho de pulsos) puede generar señales de frecuencia y de ciclo de trabajo variados por uno o más pines de salida. El microcontrolador PIC16F887 dispone de dos módulos CCP - CCP1 y CCP2. Ambos son idénticos en modo normal de funcionamiento, mientras que las características del PWM mejorado están disponibles sólo en el modo CCP1.

El PIC16F877 posee dos módulos CCP, denominados CCP1 y CCP2. Ambos módulos son prácticamente idénticos con la excepción de la operación del disparo de evento especial. Cada uno de estos dos módulos poseen un registro de 16 bits, el cual puede operar como:

- Registro de captura de 16 bits.
- Registro de comparación de 16 bits.
- Registro de Ciclo de Trabajo de módulo PWM.

Cada modo de operación requiere como recurso uno de los Timers del PIC. En la tabla, se muestran los Timers usados por cada modo:

Modo de operación del CCP	Recurso Utilizado
Captura	Timer 1
Comparación	Timer 1
PWM	Timer 2

El Timer1: El timer1 es un segundo módulo disponible para temporizar en muchos microcontroladores PIC de clase media.

El Timer1 consta fundamentalmente de un contador ascendente de 16 bits precedido por un pre-divisor programable con factor de división de 1, 2, 4 u 8. El Timer1 puede operar como temporizador (al contar ciclos de máquina) o como contador (si cuenta pulsos externos).

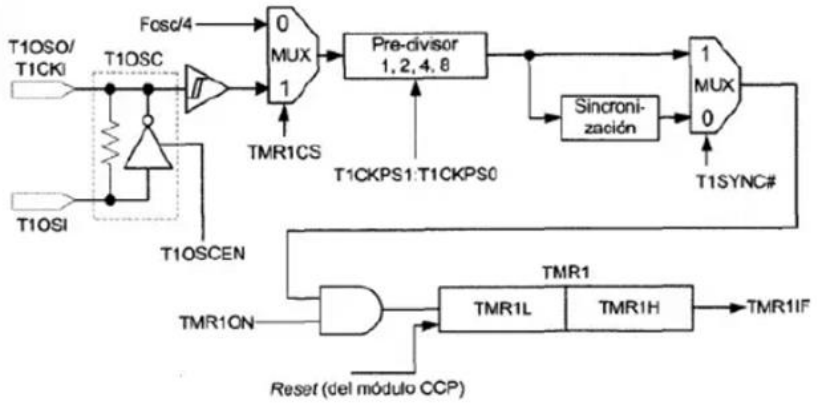


Figura 1. Diagrama de bloques del Timer1.

El Timer2: El Timer2 es un tercer módulo disponible para temporizar en muchos microcontroladores PIC de clase media. Está formado por un contador ascendente de 8 bits, un pre-divisor, un post-divisor y un registro para almacenar el módulo de conteo. El Timer2 sólo trabaja como temporizador y es un contador de ciclos de máquina.

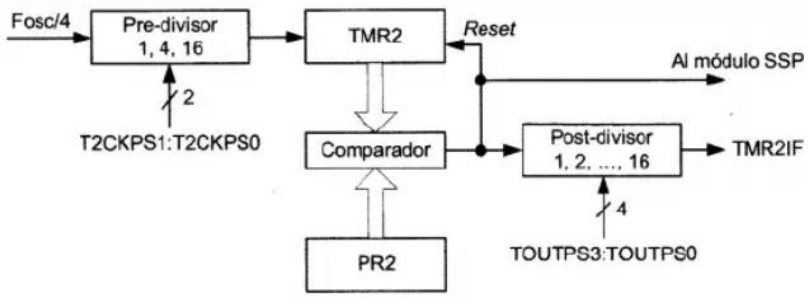


Figura 2. Diagrama de bloques del Timer2.

Un módulo CCP está formado básicamente por una pareja de registros de 8 bits denominados CCPRXH y CCPRXL. la letra "x" se debe sustituir por 1 ò 2, indicando el módulo CCP al que se esté haciendo referencia. En estos registros se puede almacenar, respectivamente, la parte alta y baja de un número de 16 bits. Cada módulo utiliza también el registro CCPXCON para el control y el bit CCPxIF del registro PIR como indicador de que se ha producido un evento, si la interrupción del módulo está habilitada, (con el bit CCPxIE del registro PIE), cuando CCPXIF pasa a 1 se genera una solicitud de interrupción.

El módulo CCP1:

El registro principal de este módulo (CCPR1) se compone de dos registros de 8 bits, denominados CCPR1H (parte más significativa) y CCP1L (Parte menos significativa).

La operación del módulo se controla mediante el registro CCP1CON y el disparo de evento especial, el cual es generado al alcanzarse la igualdad en un registro de comparación reseteará el Timer1.

El Módulo CCP2:

El registro principal de este módulo (CCPR2) se compone de dos registros de 8 bits, denominados CCPR2H (parte más significativa) y CCPR2L (parte menos significativa).

La operación del módulo se controla mediante el registro CCP2CON y el disparo de evento especial, el cual es generado al alcanzarse la igualdad en un registro de comparación reseteará el Timer1 e iniciará una conversión analógico/digital si el módulo convertidor A/D está habilitado.

Nota: Los terminales CCPx son CCP1 o CCP2 son entradas en el modo de captura y salidas en los modos comparador y PWM. Estos terminales, uno por cada módulo CCP existente en el PIC, comparten funciones con los terminales del puerto C.

En los modos de captura y comparador, El Timer1 es utilizado por los módulos CCP como base de tiempo, en estos modos el Timerl debe ser programado como temporizador o como contador en modo sincronizado. En el modo PWM, el Timer2, que opera siempre como temporizador, determina la frecuencia de la señal PWM.

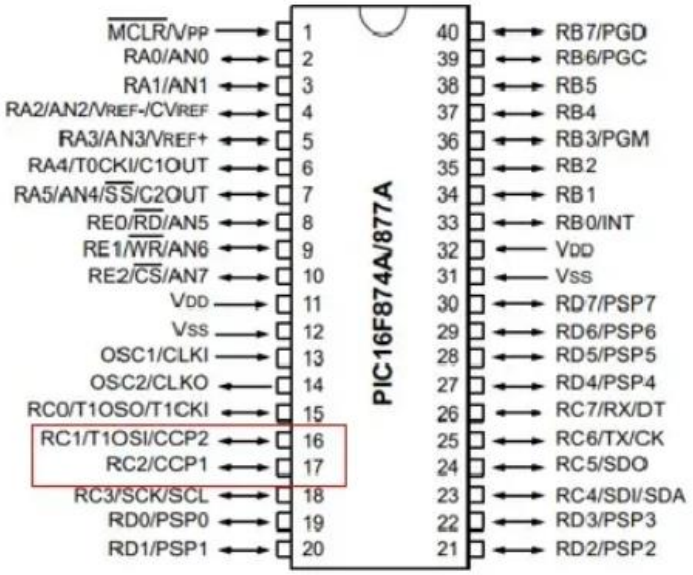


Figura 3. Distribución de pines pic16F877A

Dado que los módulos CCP comparten Funciones con los temporizadores Timer1 y Timer2 hay que tener en cuenta el uso compartido de estos temporizadores la siguiente tabla muestra las posibles interacciones que hay que considerar para programar los módulos CCP.

Modo CCPx	Modo CCPy	Interacción
Captura	Captura	La misma base de tiempos de TMB1
Captura	Comparación	El comparador debe configurarse para el modo de disparo especial que pone a cero el TMR1
Comparación	Comparación	El comparador(es) debe configurarse para el modo de disparo especial que pone a cero el TMR1
PWM	PWM	El PWM tendrá la misma frecuencia y proporción de actuación (interrupción de TBR2)
PWM	Captura	Ninguna
PWM	Comparación	Ninguna

Cada módulo CCP puede operar en cualquier de los siguientes modos:

Modo de captura. El módulo CCP captura el valor del TImerl cuando ocurre un evento externo en el terminal CCPx.

Modo comparador. El registro del módulo CCP almacena un número de 16 bits que se compara con el valor del Timer1 y, según el resultado de la comparación, se genera un evento, que puede incluir un cambio en el terminal CCPx.

Modo modulador de pulsos en anchura (PWM). El módulo CCP y el timer2 forman un modulador de ancho de pulsos con salida por el terminal CCPX.

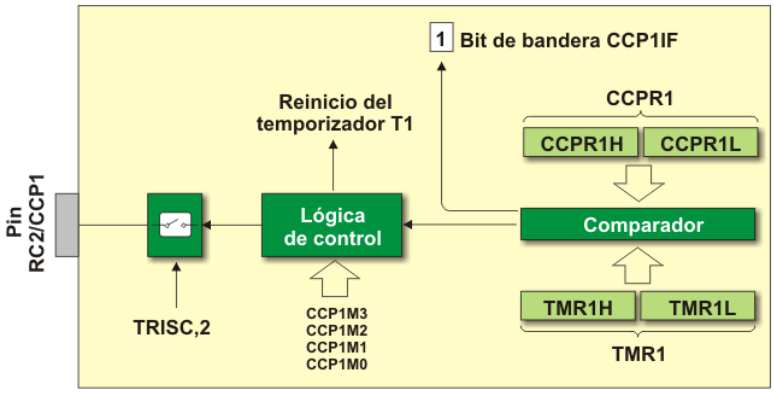
5.2 Configuración y programación como Comparador.

En el modo de comparación, el registro de 16 bits CCPR1 (CCPR1H:CCPR1L) se compara constantemente con el valor del registro de 16 bits TMR1. De manera que cuando sus valores coinciden, además de activarse la bandera para solicitar la interrupción CCP1IF (PIR1<2>), puede ocurrir en el pin RC2/CCP1 (previa configuración) alguna de las siguientes acciones:

- RC2/CCP1 se pone en alto
- RC2/CCP1 se pone en bajo
- RC2/CCP1 no cambia

La acción que ocurre en este pin se configura mediante los bits de control CCP1M3:CCP1MO (CCP1CON<3:0>). En la Figura 5 se muestra un diagrama de bloques donde se ilustra la manera en que funciona el módulo CCP en modo comparador.

El valor almacenado en el registro CCP1 se compara constantemente al valor almacenado en el registro del temporizador Timer1. Al igualarse los valores, el estado lógico en el pin de salida puede ser cambiado, lo que depende del estado de bits en el registro de control (CCP1M3 - CCP1M0). El bit de bandera CCP1IF se pone a uno simultáneamente.



Consideraciones que se deben hacer para configurar adecuadamente el modo de comparación:

- El pin RC2/CCP1 deberá configurarse como salida limpiando el bit TRISC<2>.
- Al limpiar el registro CCP1CON, el pestillo de salida del pin RC2/CCP1 se fuerza a su valor "default" a cero.
- El temporizador 1 debe estar funcionando en modo temporizador (o en modo contador sincronizado)
- Si se está manejando por poleo el indicador de solicitud de interrupción, se deberá limpiar por software antes de un posible evento que la active; de lo contrario, no se notará la activación.

5.3 Configuración y programación como Captura.

En el modo de captura, el registro CCPR1 (CCPR1H:CCPR1L) captura el valor de 16 bits del registro TMR1 cuando ocurre un evento en el pin RC2/CCP1. El evento en cuestión puede especificarse previamente como alguno de los siguientes:

- Cada transición de bajada
- Cada transición de subida 45
- Cada 4 transiciones de subida
- Cada 16 transiciones de subida

Además de capturar el valor de TMR1, se activa la bandera de solicitud de interrupción CCP1IF, la cual deberá ser limpiada por software para poder detectarla si se está consultando por poleo.

El tipo de acción que se desea detectar en este pin se configura mediante los bits de control CCP1M3:CCP1M0 (CCP1CON<3:0>

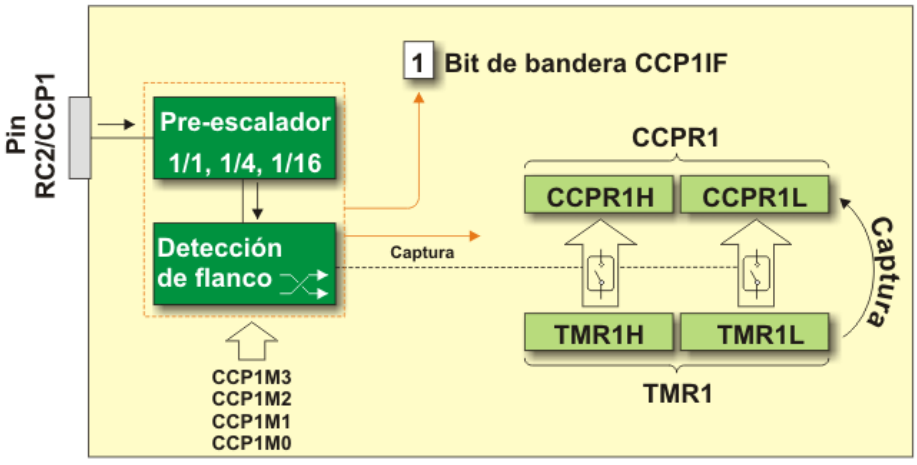
Nota: Si ocurre otro evento de captura antes de haber leído el registro CCPR1, el valor capturado anterior se perderá, ya que con la nueva captura este registro está reescrito.

el registro del temporizador Timer1 (que consiste en los TMR1H y TMR1L) se copia al registro CCP1 (que consiste en los CCPR1H y CCPR1L) en las siguientes situaciones:

- Cada flanco ascendente (1 -> 0) en el pin RC2/CCP;
- Cada flanco descendente (0 -> 1) en el pin RC2/CCP1;
- Cada cuarto flanco ascendente (0 -> 1) en el pin RC2/CCP1; y
- Cada decimosexto flanco descendente (0 -> 1) en el pin RC2/CCP1.

Una combinación de cuatro bits (CCP1M3 - CCP1M0) del registro de control determina cuál de estos eventos causará transmisión de dato de 16 bits. Además, se deben cumplir los siguientes requisitos:

- El pin RC2/CCP1 debe estar configurado como entrada; y
- El Timer1 debe funcionar como temporizador o contador síncrono.



El bit de bandera CCP1IF se pone a uno después de acabar la captura. Si se pone a 1 el bit CCP1IE del registro PIE1, se producirá una interrupción. En caso de que el módulo CCP1 esté en modo de captura, puede producirse una interrupción no deseada. Para evitarlo, antes de que ocurra un cambio en el registro de control se deben poner a 0 tanto el bit que habilita la interrupción CCP1IE, como el bit de bandera CCP1IF. Las interrupciones no deseadas pueden producirse al cambiar el valor del pre-escalador. Para evitarlo, el módulo CCP1 debe

estar apagado temporalmente antes de cambiar el valor del pre-escalador. Se recomienda la siguiente secuencia de programa, escrita en ensamblador:

```
BANKESEL CCP1CON

CLRF     CCP1CON  ;REGISTRO DE CONTROL BORRADO
                  ;MÓDULO CCP1 ESTÁ APAGADO
MOVLW    XX       ;NUEVO MODO DEL PRE-ESCALADOR ESTÁ SELECCIONADO
MOVWF    CCP1CON  ;EN EL REGISTRO DE CONTROL SE INTRODUCE UN NUEVO VALOR
                  ;MÓDULO CCP1 SE ENCIENDE SIMULTÁNEAMENTE
```

En mikroC:

```
...
ASM {
  BANKESEL CCP1CON
  CLRF CCP1CON      // REGISTRO DE CONTROL BORRADO
                    // MÓDULO CCP1 ESTÁ APAGADO
  MOVLW XX          // NUEVO MODO DEL PRE-ESCALADOR ESTÁ SELECCIONADO
  MOVWF CCP1CON     // EN EL REGISTRO DE CONTROL SE INTRODUCE NUEVO VALOR
}                   // MÓDULO CCP1 SE ENCIENDE SIMULTÁNEAMENTE
...
```

Consideraciones que se deben hacer para configurar adecuadamente el modo de captura:

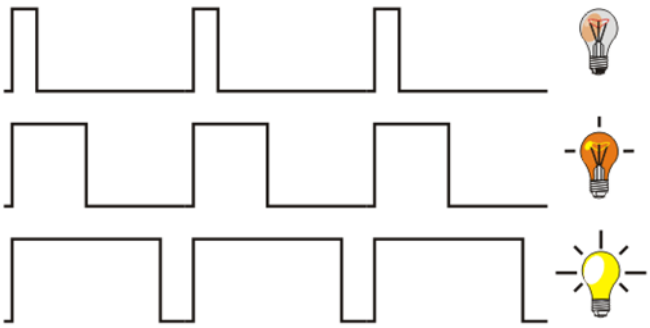
En el modo de captura, el pin RC2/CCP1 deberá configurarse como entrada, poniendo en alto el bit TRISC<2>. Si por alguna razón el pin RC2/CCP1 está configurado como salida, se deberá tener en cuenta que una escritura en el puerto C puede causar una condición de captura.

El temporizador 1 debe estar funcionando en modo temporizador (o en modo contador sincronizado); de lo contrario, el modo de captura puede no funcionar.

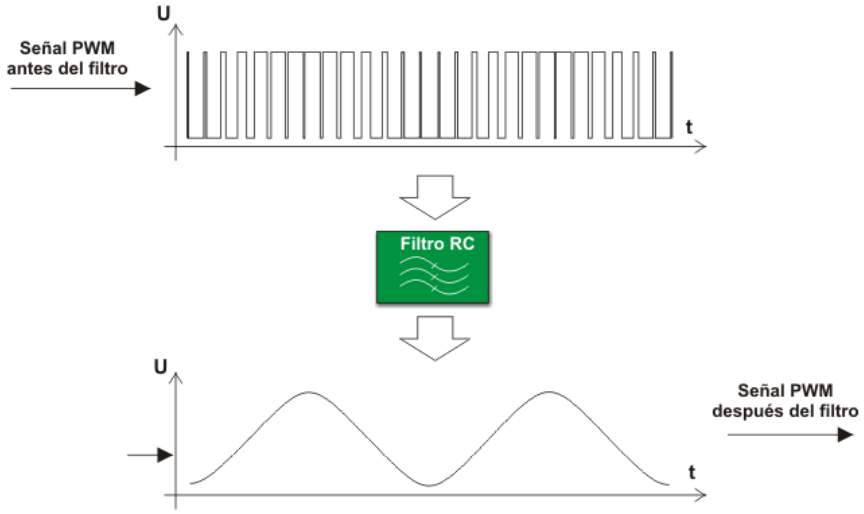
Cuando se realiza un cambio de un modo de captura a otro modo de captura, se puede generar una solicitud de interrupción falsa. Esto debe evitarse limpiando la máscara de interrupción correspondiente (CCP1IE (PIE1<2>)) cuando se realiza un cambio de estos para evitar una interrupción falsa.

5.4 Configuración y programación como PWM

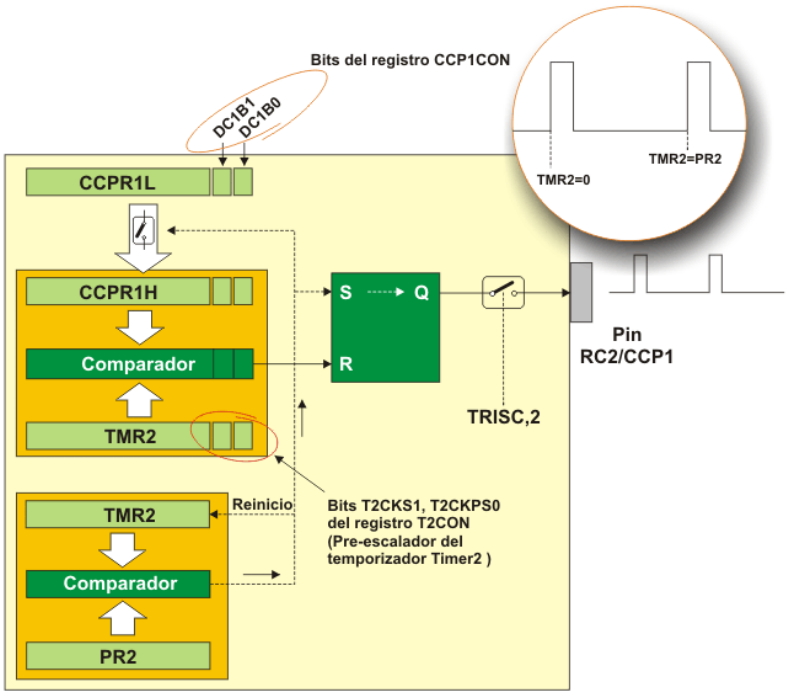
Las señales de frecuencia y de ciclo de trabajo variados tienen una amplia gama de aplicaciones en automatización. Un ejemplo típico es un circuito de control de potencia. Refiérase a la siguiente figura. Si unos cero lógicos (0) indica un interruptor abierto y un uno lógico (1) indica un interruptor cerrado, la potencia eléctrica que se transmite a los consumidores será directamente proporcional a la duración del pulso. Esta relación se le denomina Ciclo de Trabajo.



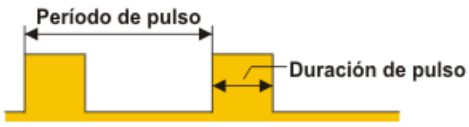
El otro ejemplo, común en la práctica, es el uso de señales PWM en un circuito para generar señales de forma de onda arbitraria como una onda sinusoidal. Vea la siguiente figura:



Los dispositivos que funcionan según este principio se utilizan con frecuencia en la práctica como variadores de frecuencia ajustable que controlan motores eléctricos (velocidad, aceleración, desaceleración etc.)



La Figura anterior muestra el diagrama de bloques del módulo CCP1 puesto en el modo PWM. Para generar un pulso de forma arbitraria en el pin de salida, es necesario ajustar el período de pulsos (frecuencia) y la duración de pulsos



PERÍODO DE PWM

El período de pulso de salida (T) se determina por el registro PR2 del temporizador Timer2. El período de PWM se puede calcular por la siguiente ecuación: Período PWM = (PR2 +1) * 4Tosc * Valor de pre-escala del Timer2 Si el período de PWM (T) es conocido, es fácil determinar la frecuencia de señal F, porque estos dos valores están relacionados por la ecuación F=1/T.

CICLO DE TRABAJO DE PWM

El ciclo de trabajo de PWM se especifica al utilizar en total 10 bits: los ocho bits más significativos del registro CCPR1L y los dos bits menos significativos adicionales del registro CCP1CON (DC1B1 y DC1B0). El resultado es un número de 10 bits dado por la siguiente fórmula: Ancho de pulsos = (CCPR1L,DC1B1,DC1B0) * Tosc * Valor de pree-scala del Timer2 La siguiente tabla muestra cómo generar las señales PWM de diferentes frecuencias cuando el microcontrolador utiliza un cristal de cuarzo de 20 MHz (Tosc=50nS).

FRECUENCIA [KHZ]	1.22	4.88	19.53	78.12	156.3	208.3
Pre-escalador del TMR2	16	4	1	1	1	1
Registro PR2	FFh	FFh	FFh	3Fh	1Fh	17h

- El pin de salida se va a poner a 1 constantemente, si por error el ancho de pulso generado es más largo que el período de PWM.
- En esta aplicación, no se puede utilizar el post-escalador del temporizador Timer2 para generar períodos de PWM largos.

RESOLUCIÓN DE PWM

Una señal PWM no es nada más que una secuencia de pulsos que varían su ciclo de trabajo. Para una frecuencia específica (número de pulsos por segundo), hay un número limitado de combinaciones de ciclos de trabajo. Este número representa una resolución medida en bits. Por ejemplo, si una resolución es de 10 bits estarán disponibles 1024 ciclos de trabajo discretos; si una resolución es de 8 bits estarán disponibles 256 ciclos de trabajo disretos etc. En este microcontrolador la resolución es determinada por el registro PR2. El máximo valor se obtiene al usar el número FFh. Frecuencias y resoluciones de PWM (Fosc = 20MHz):

FRECUENCIA DE PWM	1.22KHZ	4.88KHZ	19.53KHZ	78.12KHZ	156.3KHZ	208.3KHZ
Pre-escala del temporizador	16	4	1	1	1	1
Valor del PR2	FFh	FFh	FFh	3Fh	1Fh	17h
Resolución máxima	10	10	10	8	7	6

Frecuencias y resoluciones de PWM (Fosc = 8MHz):

FRECUENCIA DEL PWM	1.22KHZ	4.90KHZ	19.61KHZ	76.92KHZ	153.85KHZ	200.0KHZ
Pre-escala del temporizador	16	4	1	1	1	1
Valor del PR2	65h	65h	65h	19h	0Ch	09h
Resolución máxima	8	8	8	6	5	5

5.5 Desarrollo de aplicaciones.

Los módulos CCP (Capture/Compare/PWM) son partes del microcontrolador que sirven para medir tiempos, generar señales y controlar dispositivos de manera automática, sin que el programa tenga que hacerlo todo directamente. Son muy útiles en proyectos de electrónica y control, como robots, sistemas automáticos o mediciones con sensores.

En el modo Capture, el módulo CCP se usa para medir cuánto tiempo pasa entre dos eventos o para saber la frecuencia de una señal. Por ejemplo, se puede usar para medir la velocidad de un motor, el tiempo que tarda en regresar una señal ultrasónica (como en un sensor de distancia), o el tiempo de respuesta de un sensor. Básicamente, el microcontrolador "captura" el valor de un contador cuando ocurre un cambio en una entrada, y con eso calcula el tiempo o la frecuencia.

En el modo Compare, el CCP sirve para hacer que algo ocurra en un momento exacto. El microcontrolador compara el valor de un temporizador con un número que uno le pone, y cuando son iguales, puede encender o apagar una salida, o generar una interrupción. Este modo se usa mucho para controlar eventos que deben ocurrir a cierto tiempo, como encender un motor o una luz después de unos segundos, generar señales con una frecuencia fija, o sincronizar diferentes partes de un sistema.

Por último, el modo PWM (Modulación por Ancho de Pulso) se usa para controlar la potencia que se entrega a un dispositivo. Consiste en mandar una señal cuadrada que se enciende y apaga muy rápido, pero variando el tiempo que está encendida. Mientras más tiempo esté "encendida" la señal, más potencia recibe el dispositivo. Este modo se usa mucho para controlar la velocidad de motores DC, ajustar el brillo de LEDs, mover servomotores, regular temperatura o incluso controlar fuentes de energía.

Conclusión:

La investigación se centró en el estudio y la implementación de la programación del Módulo de Captura, Comparación y PWM (CCP) del microcontrolador la programación adecuada del módulo CCP es crítica para el desarrollo de sistemas embebidos que requieren un control de tiempo preciso y eficiente de las señales de hardware.

la Programación del Módulo CCP es una habilidad esencial en el desarrollo de firmware, proporcionando la base para construir sistemas de control digital sofisticados, precisos y eficientes.

Además este tipo de módulos son aptos para la aplicación de sistemas trifásicos que pueden ser optimizados por medio de códigos implementando su manejo del CCP mediante el uso de librerías de hardware especifico proporcionados por los fabricantes para un buen rendimiento y obtener las velocidades requeridas al momento de que se ejecute la programación, por este motivo Se recomienda investigar y aplicar la programación del Módulo CCP en sus otros modos de operación (Captura y Comparación) simultáneamente, para desarrollar sistemas de control de lazo cerrado más robustos.

Referencias:

[1]F. Pallás R., Microcontroladores: Fundamentos y aplicaciones con PIC, Marcombo, España, 2007.

[2]“modulos-ccp - MIKROE”. MIKROE. Accedido el 24 de octubre de 2025. [En línea]. Disponible: https://www.mikroe.com/ebooks/microcontroladores-pic-programacion-en-c-con-ejemplos/modulos-ccp?srsltid=AfmBOopDNsu8qVs4y44guy4pw7u0OLeFfl0rdkqfU9Qf8gAlpkt3Pa4A

[3]Microchip Technology Inc., PIC16F877A Datasheet, Microchip Technology, [En línea]. Disponible en: https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf

[4]Sergio A. Castaño Giraldo. “PWM con PIC – (Modulación por Ancho de Pulso)”. Controlautomaticoeducacion. Accedido el 23 de octubre de 2025. [En línea]. Disponible: https://controlautomaticoeducacion.com/sistemas-embebidos/microcontroladores-pic/pwm-modulacion-por-ancho-de-pulso/

INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

DIVISIÓN:

INGENIERÍA MECATRÓNICA

MATERIA:

MICROCONTROLADORES

DOCENTE:

JOSE ANGEL NIEVES VAZQUEZ

UNIDAD 5

PROYECTO DE LA UNIDAD 5

ALUMNOS:

ISRAEL ANTONIO LÓPEZ ESCRIBANO
JESUS ALEJANDRO ROSAS ROSAS
ARGELIO SANTIAGO REYES

GRUPO:

711-B

FECHA DE ENTREGA:

San Andrés Tuxtla Ver. 01 de diciembre del 2025

Introducción:

En este proyecto se desarrolló una maqueta en la cual se colocó un sistema de control de acceso utilizando un teclado matricial 4x4 y un servomotor, programado con una tarjeta Arduino Uno. El objetivo principal fue crear un mecanismo de apertura que solo se active al ingresar una contraseña correcta.

Materiales

- Arduino Uno

- Teclado matricial 4x4

- Servomotor

- Cables jumper

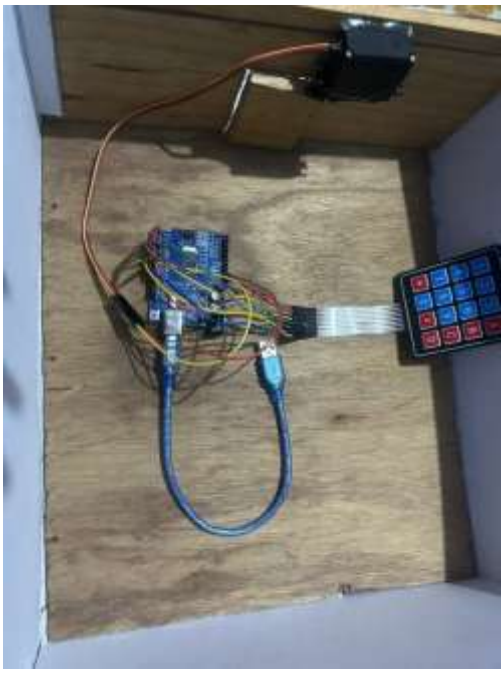
- Maqueta de puerta

Desarrollo

El sistema permite que el usuario ingrese una contraseña de cuatro dígitos mediante un teclado matricial. Si la clave coincide con la contraseña programada, el servomotor ejecuta una secuencia de apertura y cierre de puerta. En caso contrario, el sistema notifica error por el monitor serial.

Se configuró el teclado matricial con la librería Keypad.h, se programó el manejo del servomotor mediante señales PWM y se desarrolló la lógica de ingreso y validación de contraseña. Si la clave es correcta, se activa una secuencia de movimiento del servo

Esto lo colocamos en la maqueta de donde la puerta tiene movimiento:



Código

Para esto realizamos el siguiente código el cual al momento de ingresar la combinación correcta (1,2,3,4) la puerta se abre y luego se cierra:

```
#include <Keypad.h>
#include <Servo.h>

Servo myservo;

// ---- CONFIGURAR TECLADO 4x4 ----
const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'D','C','B','A'},
  {'#','9','6','3'},
  {'0','8','5','2'},
  {'*','7','4','1'}
};

byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

```
// ---- VARIABLES ----
String inputCode = "";
String correctCode = "1234";

void setup() {
  Serial.begin(9600);
  myservo.attach(10);

  // Posición neutral (servo detenido)
  myservo.writeMicroseconds(1500);
}

void loop() {
  char key = keypad.getKey();

  if (key) {
    Serial.println(key);

    if (key == '#') {
      inputCode = "";
      Serial.println("Codigo borrado");
    } else {
      inputCode += key;
    }

    if (inputCode.length() == 4) {
      if (inputCode == correctCode) {
        Serial.println("Codigo correcto");
        moverServoSecuencia();
      } else {
        Serial.println("Codigo incorrecto");
      }
      inputCode = "";
    }
  }
}

// ---- FUNCION PARA SERVO CONTINUO POR TIEMPO ----
void moverServoSecuencia() {

  // 1️⃣ Girar primero al sentido inverso por 0.5 s
  myservo.writeMicroseconds(1000);  // Giro hacia el otro lado
  delay(500);

  // 2️⃣ Regresar al sentido contrario por 0.5 s
```

```
  myservo.writeMicroseconds(2000);  // Regresa
  delay(1000);

  // 3️⃣ Detener el servo
  myservo.writeMicroseconds(1500);
}
```

Resultados

Al colocar la contraseña al teclado (1234) la puerta abre y después de un momento cierra como lo muestra las siguientes imágenes.

En el siguiente link se encuentra el video donde se muestra que el proyecto funciona de manera muy eficaz y correcta.

https://drive.google.com/file/d/1s78Ijom6kR12SoUqMn2L25k5pQmY3_gP/view?usp=drivesdk

Conclusión:

El proyecto permitió aplicar conceptos fundamentales de microcontroladores utilizando una tarjeta Arduino uno, logrando un sistema de control de acceso de una puerta completamente funcional y demostrando la interacción efectiva entre hardware y software.

Podemos observar como un microcontrolador controla el servo y logra mandar las señales necesarias para poder abrir y cerrar la puerta en el momento que la contraseña sea correcta.