Universidad Nacional Autónoma de México

Facultad de Ingeniería

Semestre 2023-2

Laboratorio de Microcomputadoras

Proyecto 5:

Uso de interrupciones y timers.

Profesor:

M.I. Ruben Anaya García

Alumnos:	Número de cuenta:
Barreiro Valdez Alejandro	317520888
Gil Márquez Arath Emiliano	317083875
Herrera Carrillo Cristhian	317094662
Zepeda Baeza Jessica	317520747

Grupo Teoría: 1

Fecha de realización: 5 de junio de 2023

Fecha de entrega: 25 de junio de 2023

Introducción.

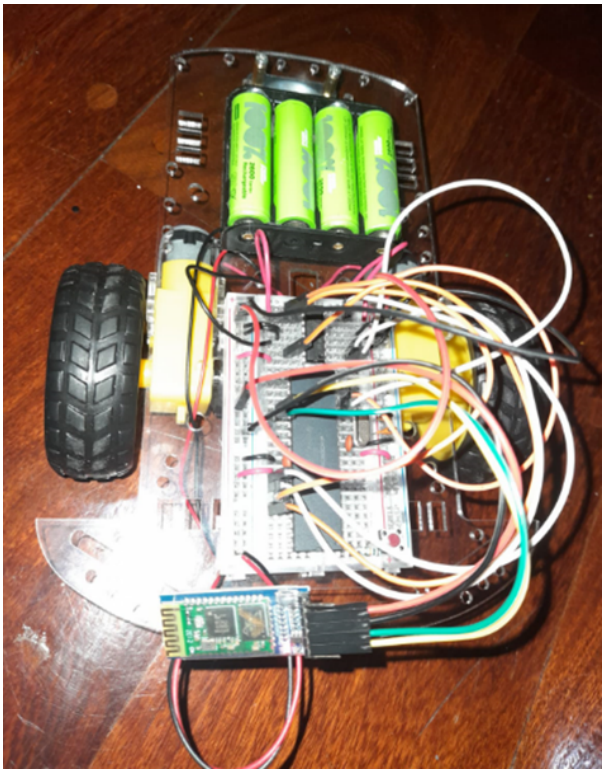
Las interrupciones son señales que se generan en la microcomputadora para indicar la ocurrencia de un evento particular. Estos eventos pueden ser externos, como una pulsación de un botón o la llegada de datos a través de un puerto de comunicación, o internos, como un desbordamiento de un contador o una solicitud de servicio de un periférico. Cuando se produce una interrupción, la microcomputadora suspende temporalmente la ejecución del programa principal y salta a un programa de manejo de interrupciones específico, también conocido como rutina de interrupción. Allí se procesa el evento y se toman las acciones necesarias. Una vez finalizada la rutina de interrupción, el programa principal se reanuda desde el punto en el que se interrumpió.

Los timers (temporizadores) son dispositivos en la microcomputadora que generan señales periódicas o cuentan pulsos de reloj. Estos temporizadores se utilizan para medir intervalos de tiempo, generar señales de sincronización, controlar el funcionamiento de periféricos y realizar tareas programadas. Los timers se pueden configurar para generar una interrupción cuando se alcanza un valor de conteo específico, lo que permite al programa principal tomar acciones en momentos determinados. Además, los timers también se utilizan en conjunto con otros componentes, como los convertidores analógico-digitales (ADC) o los periféricos de comunicación, para sincronizar operaciones y asegurar un flujo adecuado de datos.

El uso de interrupciones y timers en las microcomputadoras brinda numerosos beneficios. En primer lugar, permiten un manejo eficiente de eventos en tiempo real, ya que las interrupciones pueden ser atendidas de manera inmediata y el programa principal no se ve afectado por largas esperas. Además, los timers ofrecen la capacidad de controlar el tiempo de ejecución de tareas y sincronizar diferentes partes del sistema, mejorando la precisión y la coordinación.

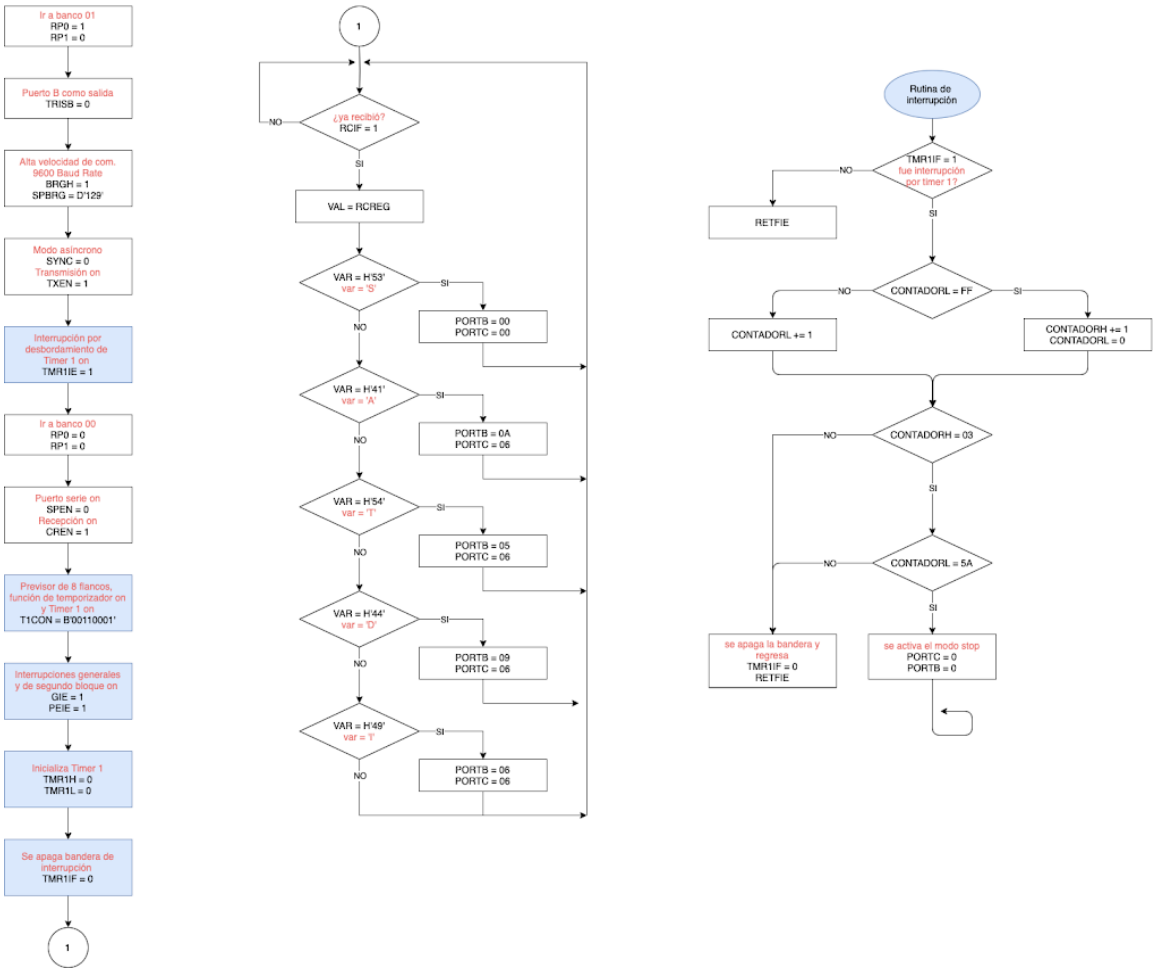
Desarrollo

Para la realización del proyecto 5 se implementó el microcontrolador PIC16F877A en su sistema mínimo ya que será el cerebro del robot y se encargará de procesar las señales y enviar las instrucciones a los motores. Estos motores serán responsables de la tracción del robot, deben ser compatibles con el microcontrolador y capaces de mover el robot con ayuda de unas ruedas adecuadas al tamaño y peso, que serán controladas a lo especificado en el código cargado en el microcontrolador, donde dependiendo de los comandos enviados debe realizar los movimientos de avanzar hacia adelante, desplazamiento hacia atrás, girar a la derecha, girar a la izquierda y parar, estas acciones se podrán realizar durante el transcurso de un minuto y medio (1:30), transcurrido este tiempo se ejecutara la interrupción que dejara al robot inmovil. Para la transmisión de señales es requisito utilizar un módulo Bluetooth para permitir la comunicación inalámbrica entre el robot y la aplicación móvil, el cual va colocado a los pines TX (transmisión) y RX (recepción) del microcontrolador, que fue configurado como receptor de los comandos. Es necesario utilizar baterías adecuadas para alimentar el sistema de manera eficiente (microcontrolador, motores y los demás componentes).



Algoritmos

Para este algoritmo se utilizó la misma carta ASM que el proyecto anterior con la adición de instrucciones en la configuración inicial para el uso de interrupciones junto con la rutina de interrupción. A continuación se muestran las cartas ASM:



Como se mencionó en el proyecto pasado, la primera columna indica las configuraciones iniciales, básicamente se define el puerto B como salida y se habilita la comunicación asíncrona con los puertos TX y RX definiendo una tasa de alta velocidad con 9600 de Baud Rate. Aquellos cuadrados en azul muestran las nuevas configuraciones que corresponden a las interrupciones. Primero se habilita la interrupción particular: desbordamiento de Timer 1. Después se configura Timer 1 para que actúe como temporizador, tenga un predivisor de 8 flancos y se prenda. También se habilitan las interrupciones generales y las del segundo bloque. Por último se inicia el contador del Timer en 0 limpiando su registro alto y bajo y se limpia la bandera de interrupción para comenzar con el programa.

El programa después espera la recepción de un dato, verificando la bandera de recepción completa PCIF. Una vez que la bandera es 1, guarda el dato recibido en una variable VAR.

Después se compara el valor de la variable con el ASCII de ‘S’,’A’,’T’,’I’,’D’. En caso de no ser ninguno de estos caracteres, vuelve a esperar a recibir un nuevo dato. En caso de ser alguna de las letras antes mencionadas, se modifica el Puerto B y C para controlar los motores y regresa a recibir un nuevo dato. La letra ‘S’ no mueve los motores, la ‘A’ hace que el coche vaya hacia delante, la ‘T’ hacia atrás, la ‘I’ hace que gire hacia la izquierda y la ‘D’ hacia la derecha.

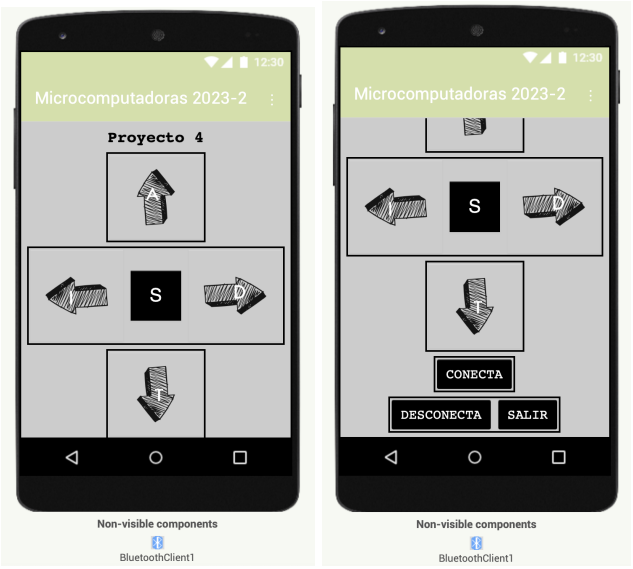
Además se incluyó la rutina de interrupción que detiene todo el programa después de minuto y medio. Debido a que la interrupción se genera cada 104.8 ms se calculó el número de veces que tiene que suceder la interrupción para llegar al minuto y medio:

$X = \frac{90\,[s]}{104.8\,[ms]} = 858.77$ que en hexadecimal es 35A

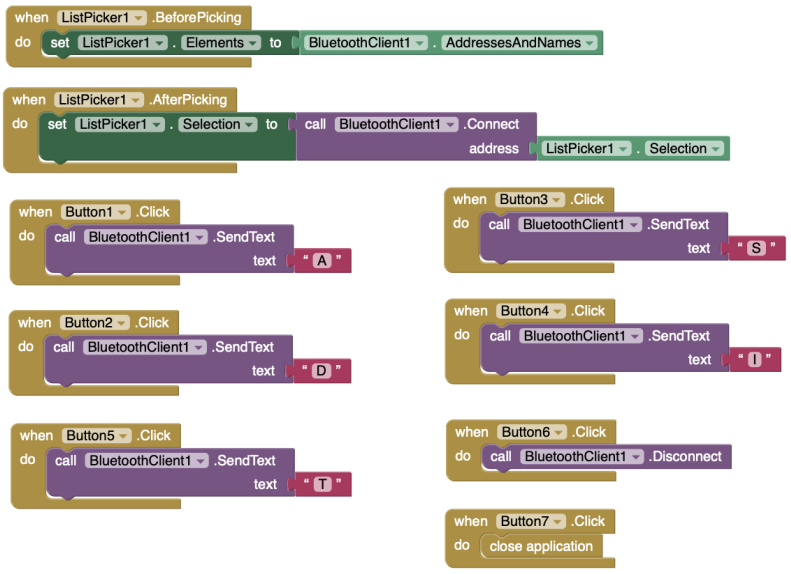
Entonces se utilizaron dos contadores, uno alto y uno bajo.

Cada que se genera la interrupción se verifica que haya sido a causa del Timer 1. Si no es así se regresa al flujo del programa. Si es por el Timer 1, se incrementa el contador; para ello primero se verifica si el contador bajo es igual a FF ya que si es así entonces se incrementa en 1 el contador alto y se pone en ceros el bajo. De no ser igual a FF tan solo se incrementa en 1 el contador bajo. Después se verifica si el contador alto es igual a 3 y en caso de serlo se verifica que el contador bajo sea igual a 5A. De ser así se sabe que se llegó al minuto y medio, se paran los motores y se queda en un loop de no hacer nada ni recibir más datos. En caso de que los contadores bajo y alto sean diferentes a 5A y 3 respectivamente, se apaga la bandera de la interrupción y se regresa el flujo al programa.

Debido a que el robot se controló mediante Bluetooth, se desarrolló una aplicación utilizando el entorno de desarrollo MIT App Inventor. La interfaz está compuesta por 5 botones que controlan la dirección en la que se moverá el robot, cada botón con la letra que se manda y en forma de flecha. También se tienen 3 botones más para conectar el dispositivo Bluetooth, para desconectarlo y para salir de la aplicación. El diseño de la interfaz es el siguiente:



Además se programó la aplicación mediante la unión de bloques de código. En ellos se define como mostrar los dispositivos Bluetooth y cómo enlazar el que se escoja. También se programa qué letra mandar al presionar cierto botón, cómo desconectar el dispositivo Bluetooth y cómo cerrar la aplicación. Los bloques se muestran a continuación.



Programa comentado

```
processor 16f877
include<p16f877.inc>

VAR EQU 0X20                    ;Valor de control
CONTADORL EQU 0X21
CONTADORH EQU 0X22


        ORG 0
        GOTO inicio

        ORG 4                   ;Vector interrupciones
        GOTO INTERRUPCIONES

        ORG 5

inicio
        BSF STATUS,RP0
        BCF STATUS,RP1          ;Se cambia al banco 01
        CLRF TRISB              ;Puerto B como salida
        BSF TXSTA,BRGH          ;Cambia bandera BRGH
        MOVLW D'129'
        MOVWF SPBRG             ;Baud rate = 9600
        BCF TXSTA,SYNC          ;Comunicación síncrona
        BSF TXSTA,TXEN          ;Activa transmisión

        BSF PIE1,TMR1IE         ;Se activan interrupcion por desbordamiento de Timer1

        BCF STATUS,RP0          ;Se cambia al banco 00

        BSF RCSTA,SPEN          ;Habilita puerto serie
        BSF RCSTA,CREN          ;Recepción sencilla

        MOVLW B'00110001'
        MOVWF T1CON             ;Predivisor de 8 flancos, función de temporizador y Timer1 ON
        BSF INTCON, PEIE        ;Habilita interrupciones de segundo bloque
        BSF INTCON, GIE         ;Habilita interrupciones generales
        CLRF TMR1H              ;Inicializa parte alta TMR1
        CLRF TMR1L              ;Inicializa parte baja TMR1
        CLRF CONTADORL          ;ContadorL en cero
        CLRF CONTADORH          ;ContadorH en cero
        BCF INTCON, TMR1IF      ;Limpia bandera desbordamiento

RECIBE:
        BTFSS PIR1,RCIF         ;Bandera de recepción completa
        GOTO RECIBE             ;Esperar recepción

        MOVF RCREG,W
        MOVWF VAR               ;Control = Recepción

        MOVLW 0x53
        SUBWF VAR,W
        BTFSC STATUS,Z          ;Si control = 0x53 "S"
        GOTO STOP               ;Entonces parar
        MOVLW 0x41
```

```
        SUBWF VAR,W
        BTFSC STATUS,Z              ;Si control = 0x41 "A"
        GOTO ADELANTE               ;Entonces avanzar
        MOVLW 0x54
        SUBWF VAR,W
        BTFSC STATUS,Z              ;Si control = 0x54 "T"
        GOTO TRAS                   ;Entonces retroceder
        MOVLW 0x44
        SUBWF VAR,W
        BTFSC STATUS,Z              ;Si control = 0x44 "D"
        GOTO DER                    ;Mover a la derecha
        MOVLW 0x49
        SUBWF VAR,W
        BTFSC STATUS,Z              ;Si control = 0x49 "I"
        GOTO IZQ                    ;Mover a la izquierda
        GOTO RECIBE                 ;Recibir otro dato

STOP                                ;Parar ambos motores
        CLRF PORTC                  ;C = 0
        CLRF PORTB                  ;B = 0
        GOTO RECIBE                 ;Recibir otro dato

ADELANTE                            ;Horario ambos motores
        MOVLW 0X06
        MOVWF PORTC                 ;C = 0x06
        MOVLW 0X0A
        MOVWF PORTB                 ;B =  0x0A
        GOTO RECIBE                 ;Recibir otro dato

TRAS                                ;Antihorario ambos motores
        MOVLW 0X06
        MOVWF PORTC                 ;C = 0x06
        MOVLW 0X05
        MOVWF PORTB                 ;B = 0x05
        GOTO RECIBE                 ;Recibir otro dato

DER                                 ;Dirección a la derecha
        MOVLW 0X06
        MOVWF PORTC                 ;C = 0x06
        MOVLW 0X09
        MOVWF PORTB                 ;C = 0x09
        GOTO RECIBE                 ;Recibir otro dato

IZQ                                 ;Dirección a la izquierda
        MOVLW 0X06
        MOVWF PORTC
        MOVWF PORTB                 ;B = C = 0x06
        GOTO RECIBE                 ;Recibir otro dato

INTERRUPCIONES:
        BTFSS PIR1, TMR1IF          ;Checar bandera de desbordamiento TMR1
        GOTO SAL_NO_FUE_TMR1        ;Si fue otra interrupción salir

        MOVLW 0XFF
        SUBWF CONTADORL, W          ;ContadorL - FF
        BTFSS STATUS, Z             ;Si el resultado es 0
        GOTO INC_CL                 ;Checar el contador
        INCF CONTADORH              ;Incrementa contador H
```

```
            CLRF CONTADORL          ;Limpia contador L
            GOTO COMPARACION        ;Sigue la comparación

INC_CL:
            INCF CONTADORL          ;Incrementa ContadorL
COMPARACION:
            MOVLW 0X03
            SUBWF CONTADORH, W      ;ContadorH - 0x03
            BTFSS STATUS, Z         ;Si el resultado es cero
            GOTO SAL_INT            ;Sale de la interrupción
            MOVLW 0X5A
            SUBWF CONTADORL, W      ;ContadorL - 0x5A
            BTFSS STATUS, Z         ;Si el resultado es cero
            GOTO SAL_INT            ;Sale de la interrupción
            CLRF PORTC              ;Limpia salida C
            CLRF PORTB              ;Limpia salida B
            GOTO $                  ;Loop infinito

SAL_INT:
            BCF PIR1, TMR1IF        ;Limpia bandera de interrupción
SAL_NO_FUE_TMR1:
            RETFIE                  ;Regresa de la interrupción

END
```

Conclusiones y/o comentarios.

Barreiro Valdez Alejandro:

Al usar interrupciones y timers en microcomputadoras es fundamental para lograr un control preciso, una respuesta rápida y una sincronización eficiente en diversas aplicaciones. Las interrupciones permiten manejar eventos en tiempo real sin interrumpir el flujo del programa principal, mientras que los timers brindan la capacidad de controlar el tiempo de ejecución de tareas y sincronizar operaciones. En el microcontrolador PIC16F877 se pudieron utilizar las interrupciones del desbordamiento de TIMER1 para parar todas las funciones de un programa transcurrido un minuto y medio. Para ello se tuvo que entender el funcionamiento del TIMER1 y cómo calcular el tiempo a partir del periodo de oscilación interna, el predivisor y el tiempo inicial del TIMER1. Además, se tuvo que programar la estructura de la interrupción donde se comprobó la bandera, se hizo la operación y se regresó de la interrupción. En definitiva, el uso adecuado de interrupciones y timers potencia el rendimiento y la capacidad de respuesta de las microcomputadoras.

Gil Márquez Arath Emiliano:

El uso de interrupciones en un microcontrolador puede proporcionar beneficios significativos en términos de eficiencia y capacidad de respuesta del sistema. La interrupción implementada permitió al microcontrolador detener temporalmente la ejecución del programa principal para atender una solicitud o evento externo de alta prioridad, en este caso detiene toda actividad del robot. Al utilizar la interrupción, es importante tener en cuenta las consideraciones de programación necesarias para garantizar un funcionamiento óptimo y evitar problemas de sincronización.

Herrera Carrillo Cristhian:

El uso de interrupciones y timers en microcomputadoras es fundamental para el manejo eficiente de eventos en tiempo real, el control preciso de temporizaciones y la sincronización de tareas. Estas características permiten mejorar el rendimiento, la coordinación y la respuesta en una amplia gama de aplicaciones, desde sistemas de control hasta procesamiento de señales y comunicaciones. Su implementación

adecuada garantiza un funcionamiento óptimo de las microcomputadoras y facilita el desarrollo de sistemas embebidos confiables y eficientes.

Zepeda Baeza Jessica:

Este proyecto sirvió como complemento al proyecto anterior ya que una vez construido el modelo del coche tan sólo fue necesario agregar las interrupciones. En este caso decidimos utilizar la función de temporizador del Timer 1 ya que es el Timer que puede activar la interrupción en intervalos más largos de tiempo. Con este proyecto nos dimos cuenta de los requerimientos para generar interrupciones como habilitar las interrupciones generales y particulares y definir el vector y la rutina de interrupciones. También nos dimos cuenta de lo necesario que es tener condicionales en la rutina de interrupción para poder saber con exactitud qué interrupción se activó y en base a ella realizar una cierta acción. Pudimos visualizar las diferentes aplicaciones que puede tener una interrupción como: realizar algo inmediatamente que se pulse un botón, activar un mecanismo cada cierto tiempo, hacer alguna acción cuando se reciba un dato, etc.