Instituto Tecnológico de Tuxtla Gutiérrez

RESIDENCIA PROFESIONAL

Nombre del proyecto: robot didáctico

Alumno: Jesús Cabrera García 07270046

Asesor: Ing. Álvaro Hernández Sol

Tuxtla Gutiérrez, Chiapas; Junio de 2011

ÍNDICE:

1. INTRODUCCIÓN

Desde la invención del circuito integrado, el desarrollo constante de la electrónica digital ha dado lugar a dispositivos cada vez más complejos. Entre ellos los microprocesadores y los microcontroladores, los cuales son básicos en las carreras de Ingeniería Electrónica.

Los microcontroladores se utilizan en circuitos electrónicos comerciales desde hace unos años de forma masiva, debido a que permiten reducir el tamaño y el precio de los equipos. Un ejemplo de son los teléfonos móviles, las cámaras de video, la televisión digital, la transmisión por satélite y los hornos de microondas.

En los últimos años se ha facilitado enormemente el trabajo con los microcontroladores al bajar los precios, aumentar las prestaciones y simplificar los montajes, de manera que en muchas ocasiones merece la pena utilizarlos en aplicaciones donde antes se utilizaba lógica discreta.

En este proyecto se implementa un “robot didáctico” usando estos microcontroladores PIC. Además, en este reporte se intenta que el lector logre entender de manera sencilla los pasos en la implementación del carrito evasor de obstáculos y obtenga un panorama de la programación utilizada en estos dispositivos electrónicos.

Se pretende explicar de manera concreta el funcionamiento del microcontrolador PIC18F4455, aunque cabe mencionar que si se comprende a fondo un microcontrolador en particular, los demás pueden aprenderse con facilidad.

En el capítulo 7 se describe el fundamento teórico de los materiales y dispositivos que se utilizan, tales como el sensor ultrasónico SRF08, el modo de conexionado, el tipo de comunicación que este utiliza para poder realizar mediciones, además se también se describen los tipos de motores que se manejan y el tipo de señal que utilizan para poder funcionar, las estructuras y accesorios que se utilizan para el armado del robot didáctico.

En el capítulo 8 se muestra el procedimiento de las actividades realizadas, tales como el modelo de implementación que se utiliza para construir el robot didáctico, se muestran los diagramas de flujo que ayudan a realizar las funciones que el robot debe seguir y se describen cada una de las funciones de rutina a seguir.

En el capítulo 9 se muestran los esquemáticos que se utilizan para realizar la placa principal del controlador, se dan a conocer los tipos de software que se utilizan para programar, simular las rutinas y además se muestran los resultados que se obtuvieron al realizar pruebas.

2. JUSTIFICACIÓN

La construcción del robot didáctico permitirá al alumno aplicar los conocimientos adquiridos en la carrera de Ingeniería Electrónica pudiendo realizar cambios o anexos a éste, tanto de hardware como software, con la posibilidad de aportar a la actividad de difusión de la carrera, realizando presentaciones en instituciones que deseen y de esta manera dar a conocer el lo que se trabaja en el ITTG.

3. OBJETIVOS

Objetivo general

Diseñar y construir un robot didáctico, el cual es capaz de evadir obstáculos que se encuentre en su trayectoria, para ello tiene un sensor ultrasónico capaz de moverse de derecha a izquierda y viceversa, de los cuales puede realizar mediciones para así poder decidir hacia donde cambiar su trayectoria.

Objetivos específicos

- Implementar un móvil evasor de obstáculos que es controlado por un microcontrolador 18F4455, mediante diferentes tipos de dispositivos electrónicos.
- Utilizar el sensor ultrasónico SRF08 para detectar objetos.
- Crear diferentes rutinas de programación para que el móvil pueda ser didáctico.

4. CARACTERIZACIÓN DEL ÁREA EN QUE PARTICIPÓ

El siguiente proyecto se realizó en el Instituto Tecnológico de Tuxtla Gutiérrez, en el Departamento de Ingeniera Eléctrica y Electrónica, en el laboratorio de Ingeniería Electrónica ubicado en el edificio I con las herramientas necesarias para poder realizar el proyecto, el robot didáctico se estuvo construyendo en el área de investigación y desarrollo tecnológico.

5. PROBLEMAS A RESOLVER, PRIORIZANDOLOS

En el área de Ingeniería Electrónica no se cuenta con una materia de robótica en la que se puedan crear robots que puedan ser didácticos, además se carece de material didáctico, para poder armar un robot.

En los robots móviles existen problemas de que solo pueden moverse en lugares planos, además de que la mayoría utiliza ruedas normales los cuales les impide poder subir en algún tipo de tope pequeño, quizás un pequeño trozo de madera, etc. Pero el robot evasor de obstáculos que se creará cuenta con ruedas de atracción (orugas) le permite pasar por encima de esos pequeños obstáculos, ya que su diseño esta realizado para que este tipo de limitaciones no le afecte además de que sus motores cuentan con el suficiente torque. Con este proyecto quizás se puedan explorar diferentes tipos de terrenos.

6. ALCANCES Y LIMITACIONES

Dentro de los alcances que se encuentra el robot didáctico es que cuenta con brazos que están en constante movimiento con tres grados de libertad, Además de que gracias a sus ruedas de atracción (orugas), puede desplazarse a diferentes tipos de terrenos, otros de los alcances es su sensor ultrasónico SRF08 puede medir hasta 11m y tener un amplio panorama de visión.

Una de las limitaciones puede ser el sistema de amortiguamiento, ya que no cuentan con este sistema para estabilizarse en terrenos pedregosos, las orugas pueden tener un poco de dificultad en superficies como el fomix, en este caso el robot didáctico solo será evasor de obstáculos.

7. FUNDAMENTO TEÓRICO

7.1 Sensor ultrasónico SRF08

DESCRIPCION

El módulo SRF08 consiste en un medidor ultrasónico de distancias de bajo costo desarrollado por la firma **DEVANTECH Ltd**. y es una versión mejorada del módulo SRF04. Emplea un microcontrolador PIC16F872 que realiza todas las funciones de control e interface, dos capsulas ultrasónicas de 40KHz y una célula LDR capaz de proporcionar una medida de luz ambiente. Se muestra en la figura 1.

a) *b)*

Figura 1: Sensor ultrasónico SRF08. a) vista frontal, b) vista posterior.

Las principales diferencias del SRF08 frente al SRF04 son las siguientes:

- Rango máximo de distancia hasta 11 m.
- El consumo se reduce a 3 mA en modo de espera y 15mA en funcionamiento.
- Es capaz de medir diferentes ecos recibidos por la señal ultrasónica que puede rebotar contra uno o varios objetos a diferentes distancias.
- Tanto la ganancia de los amplificadores internos como el rango de mediadas es ajustable por el usuario.
- Dispone de una célula LDR que permite realizar medidas de luz ambiente.
- Ofrece una lectura directa que se puede representar en centímetros, pulgadas o micro segundos.
- Un diodo led en la parte posterior del módulo genera un código de intermitencias que expresa la dirección I2C actual del módulo así como el inicio de una nueva medida.

  La comunicación con el módulo SRF08 se realiza según el protocolo I2C, disponible en la mayor parte de los microcontroladores actuales aunque también puede ser implementado por software. La comunicación se realiza de la misma manera que con cualquier otro dispositivo I2C. La dirección del módulo es, por defecto, la 0xE0, aunque existe la posibilidad de que el usuario cambie esta dirección por cualquiera de las 16 siguientes:

  0xE0, 0xE2, 0xE4, 0xE6, 0xE8, 0xEA, 0xEC, 0xEE, 0xF0, 0xF2, 0xF4, 0xF6, 0xF8, 0xFA, 0xFC o 0xFE.

  Esto permite controlar hasta 16 módulos SRF08 con un mismo bus (2 líneas). Además de estas 16 direcciones, todos los módulos responden a la dirección 0x00 de llamada general.

7.1.1. Conexionado

El módulo emplea tan sólo 4 conexiones que se pueden realizar soldando directamente 4 cables o bien mediante un conector de 5 vías. Estas se muestran en la figura 2.

+5Vcc	Tensión positiva de alimentación
SDA	Línea de E/S de datos correspondiente al bus I2C
SCL	Línea de entrada de la señal de reloj del bus I2C
N.C.	Línea sin conexión. Se emplea en la fase de fabricación y comprobación del propio módulo. No conectar nada.
GND	Tierra de alimentación.

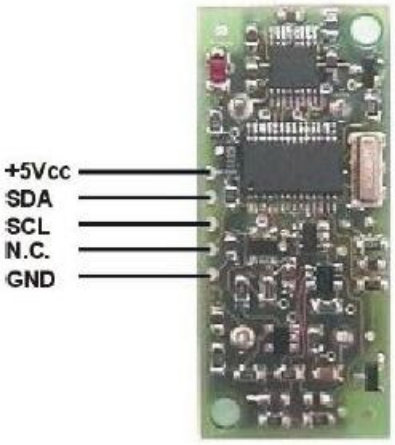


Figura 2: Diagrama de conexionado del SRF08

7.1.2. Comunicación I²C

Característica básica del protocolo I2C

- El bus de comunicación utiliza dos cables. Uno para transmitir los datos (SDA) y otro para la señal de reloj (SCL). Ambas líneas se tienen que conectar a la alimentación (Vcc) a través de resistencias Pull-UP (en el esquema del ejemplo se muestra el conexionado). Además la masa de los componentes que se interconexionan entre si debe de ser común.

7.1.3. Gestión del I²C todo esto en C y con el compilador CCS

CCS dispone de librerías específicas para gestionar la comunicación I2C de determinados componentes, entre ellos algunas memorias EEPROM. Basta con incluir la librería correspondiente por medio de la directiva **#include** y utilizar sin más las funciones de la librería. Pero si el componente que queremos utilizar no dispone del driver pertinente en CCS, se tiene otra opción, que es desarrollar un propio driver, para ello se dispone como ayuda de una directiva homologa a **#use rs232 (options)**, se trata de **#use i2c (options)**.

Al incluir esta directiva el compilador permite utilizar las siguientes funciones para controlar la comunicación serie I2C entre varios dispositivos:

- i2c_poll() . Detecta un byte en el buffer
- i2c_read(). Modo de lectura
- i2c_slaveaddr() . Dirección de esclavo
- i2c_start() . Condición de inicio
- i2c_stop(). Condición de alto
- i2c_write() . Modo de escritura

7.2. Servomotor VEX

Los servomotores VEX son un tipo de motor que pueden ser activados por señal PWM y pueden girar hacia adelante o hacia atrás. Alguna de sus características son:

- Pueden realizar 100 grados de rotación
- Hacer un posicionamiento exacto

Figura 6: Servo VEX

Especificaciones técnicas

- Rotación: 100 grados
- Par sin movimiento: 6.5 in-lbs
- Voltaje: 4,4 a 9,1 volt (la vida del motor se reducirá de operación fuera de este rango)
- PWM de entrada: 1 ms - 2 ms dará marcha atrás total, 1.5ms es neutral

- Cable Negro: Tierra
- Cable Naranja: energía
- Cable blanco: señal PWM
- Consumo de corriente: 20 mA y 1,5 A por servo

7.3. Modulación por ancho de pulso, PWM

La técnica de PWM o modulación por ancho de pulso consiste en modificar el ciclo de trabajo de una señal periódica para controlar la cantidad de energía que se envía a un motor y así regular la velocidad de giro de éste.

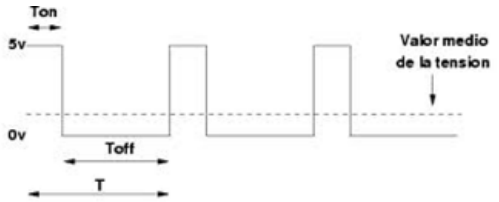


Figura 7: Señal PWM

Al modificar la tensión eléctrica se modificaba también el par motor; pero usando PWM, no se modifica la tensión eléctrica aplicada sobre el motor de modo que el par se mantiene constante.

Para controlar el giro del motor VEX se necesita un canal de PWM, sin embargo, si se quiere que además gire en ambos sentidos se necesita el uso combinado de PWM , ya que para que el robot camine hacia adelante se necesita que las señales de entrada sean diferentes, al igual para que se mueva hacia atrás, izquierda y derecha.

7.4. Estructura y accesorios

Las estructuras y accesorios son marca VEX de buena calidad para realizar prototipos de robot, se adecuan a diferentes diseños que se requieran. La estructura y los subsistemas de movimiento están muy bien integrados en el chasis del robot.

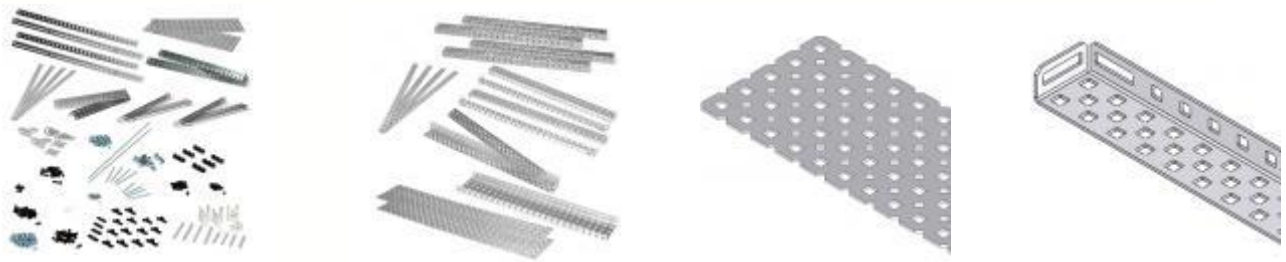


Figura 8: Estructuras

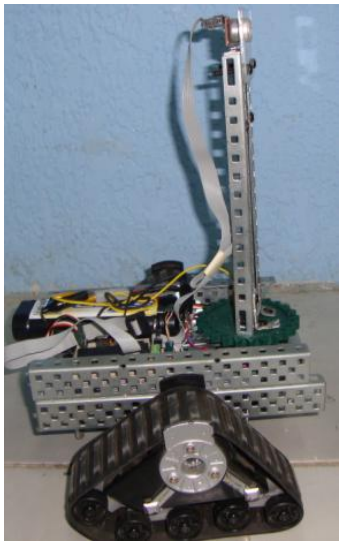


Figura 8.1 estructura general del robot didáctico

8. PROCEDIMIENTO Y DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS

8.1. Modelo de implementación

En la figura 8, se muestra el modelo del robot didáctico, como se observa se encuentra conformado por dos ruedas de atracción (orugas), motores VEX que funcionan con señal PWM para poder mover las orugas, un sensor ultrasónico SRF08 para realizar mediciones, un servo motor VEX que se encarga de girar hacia la izquierda y derecha al torso donde se encuentra sensor, una batería de 7.2v a 2000 mah y una placa donde se encuentra el microcontrolador y dispositivos electrónicos para realizar el control del móvil.

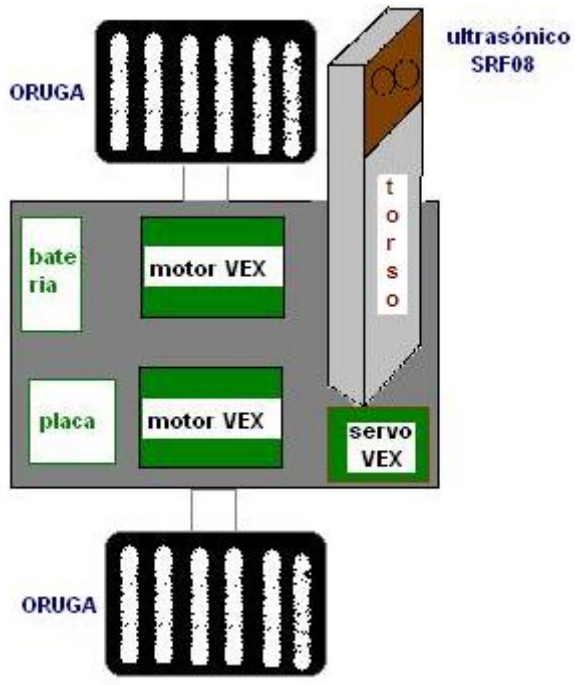


Figura 9: Modelo del robot didáctico.

El móvil comienza analizando el espacio libre que posee adelante. Si este espacio es suficiente decide avanzar. Si por el contrario detecta obstáculo, el servo del

sensor gira de derecha a izquierda para analizar las opciones y decidir cuál es el camino más adecuado.

8.2. Diagrama de flujo del programa principal.

En un inicio el programa comienza reconociendo el tipo de los puertos que se utilizaran como entrada y salida del microcontrolador, así como la los tipos de comunicaciones que va a utilizar. Que en este caso sería el protocolo I^2C. A continuación presentamos el diagrama de flujo principal.

Una vez tomada la decisión se asegura que el espacio elegido es el correcto y comienza a avanzar hasta el siguiente obstáculo.

Para realizar las mediciones y saber si el robot dispone de espacio libre se emplea el sensor ultrasónico SRF08. Cuando éste quiere detectar la distancia al objeto, el microcontrolador manda un pulso no inferior a 10 microsegundos al sensor y éste responde con un tren de impulsos ultrasónicos que "rebotarán" en el obstáculo, y por otro lado pasa la señal de retorno de eco a estado alto hasta recibir el tren

ultrasónico reflejado. Cuanto menor sea la distancia menor en tiempo será este pulso.

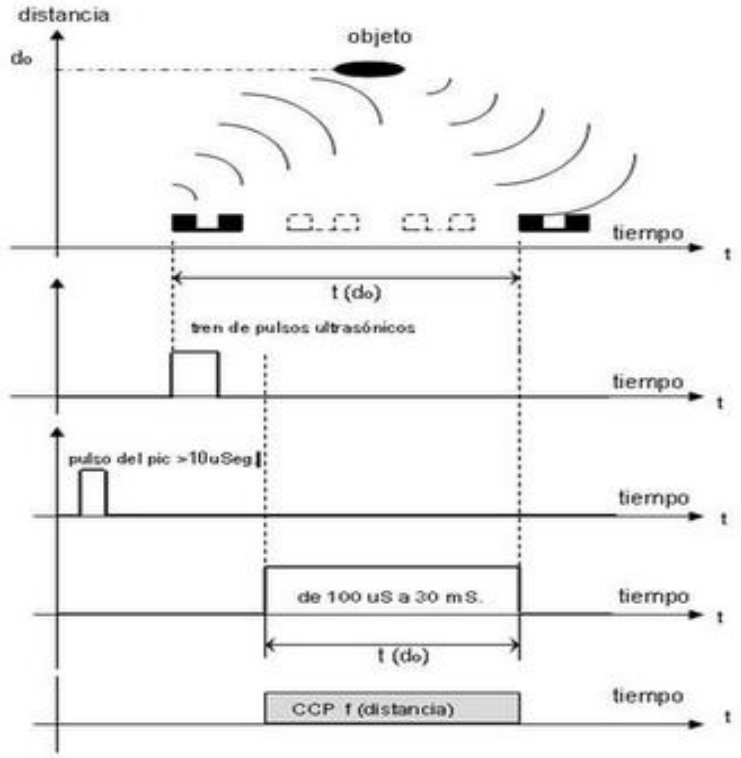


Figura 10: pulsos ultrasónicos del SRF08.

Para interpretar esta señal de retorno el sensor ultrasónico emplea comunicación I^2C, donde la señal es convertida a centímetros y guardada en variables, para realizar comparaciones y decidir qué espacio es el más adecuado.

Para que el robot se mueva de derecha a izquierda se emplea un servomotor VEX, al inicio el servo se encuentra hacia el frente, cuando el sensor SFR08 detecta un objeto el servo gira por un instante de 100 milisegundos hacia la derecha para que el sensor pueda realizar una medición, después el servo gira hacia la izquierda por un instante de 100 milisegundos y poder realizar otra medición, una vez hecha la mediciones el sensor gira nuevamente hacia su posición central y éste queda en esa posición hasta que se encuentre el siguiente obstáculo.

En la figura 11 se muestra los ciclos de trabajo de las señales PWM para mover el torso donde se encuentra el sensor SRF08.

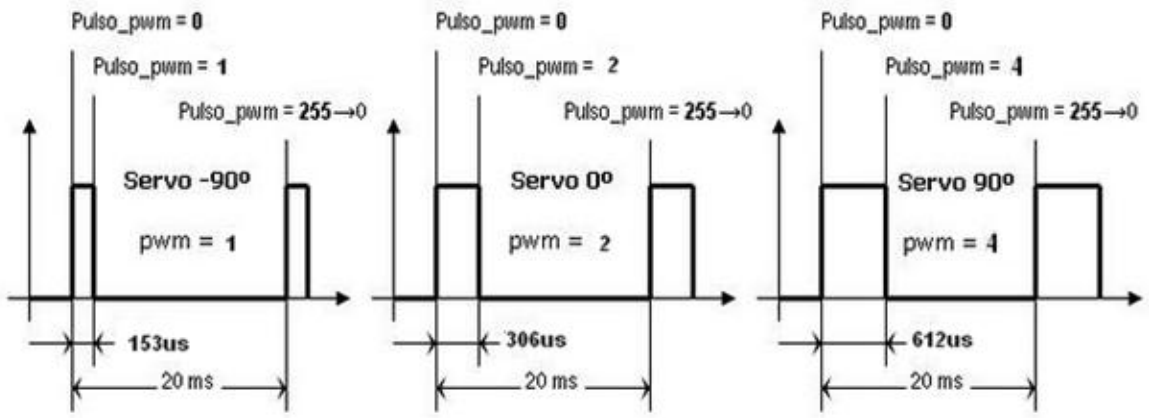
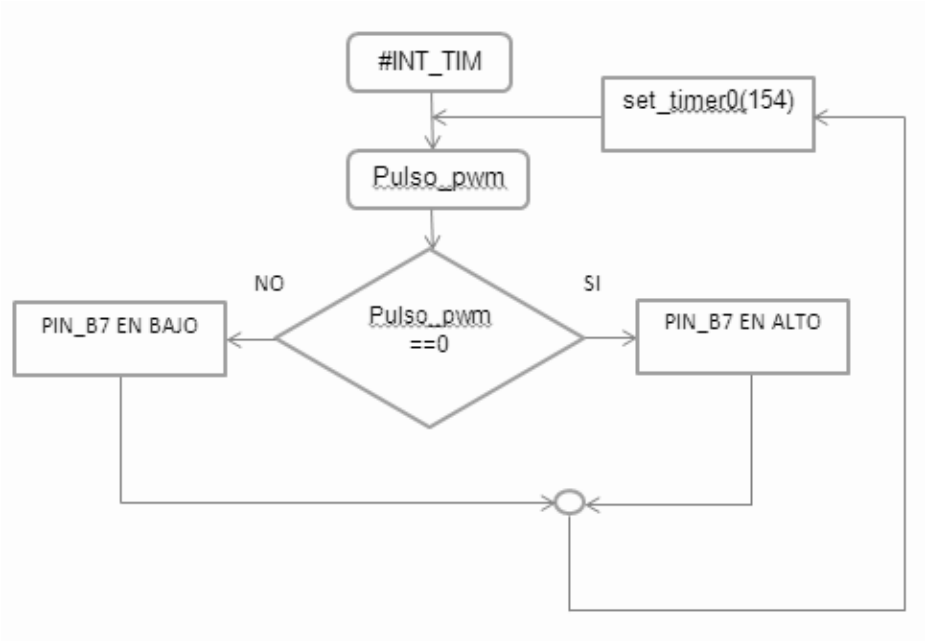


figura 11. Ciclos de trabajo.

La orientación del servo se establece con el valor que se carga como condiciones iniciales en la variable PWM. Con cada llamada a la subrutina de generación de la señal por mediación de la interrupción por desbordamiento del timer0, se incrementa la variable pulso_pwm que comparándose con el valor en PWM, decide en qué momento debe cortar el pulso.

8.3. Diagrama de flujo para que el servo VEX pueda mover al sensor ultrasónico

La generación de un PWM por software está hecha por medio de una interrupción se genera un por software con un periodo de 20 milisegundos para que el servomotor pueda mover al torso donde está el SRF08.



Para realizar la detección de objeto se utiliza un sensor ultrasónico SRF08, para ello se crea una rutina donde se tiene acceso hacia la dirección de lectura y escritura para poder indicarle que realice una medición, por lo cual estas rutinas se llaman: Medir1, Medir2 y Medir3.

Estas funciones permiten realizar mediciones y detectar a que distancia se encuentran los objetos, para ello se programa que a una distancia de 30 cm el robot se detenga y el servomotor donde se encuentra el sensor ultrasónico SRF08 pueda girar y realizar dos mediciones y comparar que medición es la más conveniente y así poder decidir hacia donde debe ir el robot.

La programación para realizar las mediciones se encuentran en el anexo B.

8.4. RUTINAS

Una vez programada las tres funciones para que el sensor ultrasónico SRF08 realice mediciones, se crean rutinas para que robot avance hacia adelante, atrás, derecha e izquierda.

De acuerdo a las especificaciones de los motores, estos requieren de una señal de 244 Hz para poder funcionar, así que se crean funciones para que se puedan mover variando sus ciclos de trabajos. CCS contiene funciones para poder variar el ciclo de trabajo esta función se llama set_pwm1_duty(x), donde x es un dato de 8 o 16 bits que determina el ciclo de trabajo, este valor junto con el valor del preescalador de TMR2 que en este caso es de 16, determinan el valor del ciclo de trabajo.
En la tabla 1 se muestra los ciclos de trabajo para que el robot didáctico se pueda mover

Tabla1: ciclos de trabajo

ADELANTE	
MOTOR1: set_pwm1_duty(95)	MOTOR2: set_pwm2_duty(595)
ATRÁS	
MOTOR1: set_pwm1_duty(595)	MOTOR2: set_pwm2_duty(95)
ALTO	
MOTOR1: set_pwm1_duty(0)	MOTOR2: set_pwm2_duty(0)
IZQUIERDA	
MOTOR1: set_pwm1_duty(595)	
DERECHA	
	MOTOR2: set_pwm2_duty(95)

set_pwm1_duty(95) es un equivalente a 57% del ciclo de trabajo
set_pwm2_duty(595) es un equivalente a 36% del ciclo de trabajo

ADELANTE

Esta función permitirá que el robot avance hacia adelante mientras el sensor no detecte ningún objeto frente a él para eso se requiere una señal PWM de 244 Hz aproximadamente 4.10 milisegundos. De los cuales para el motor1, 2.35 milisegundos deben de estar en alto y 1.75 milisegundos en bajo es por eso se utiliza la primera señal PMW del PIC18F4455 la cual se encuentra en la terminal RC1.

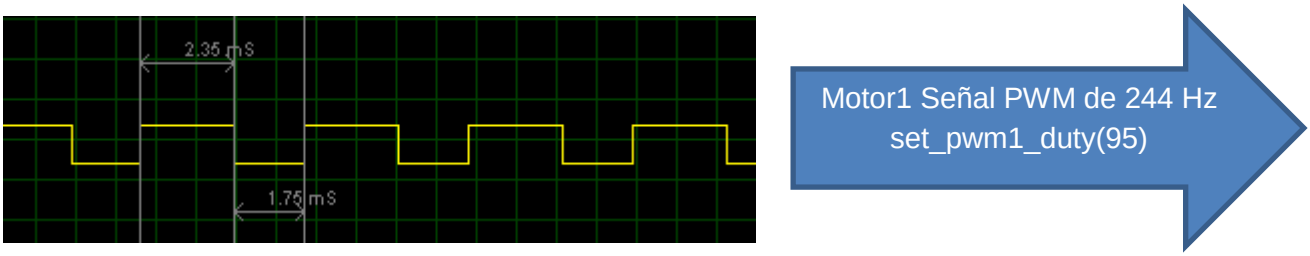


Figura 12: Ciclo de trabajo del motor1 adelante.

Para el motor2 se necesita la misma señal PWM de 244 Hz aproximadamente 4.10 milisegundos, es la misma frecuencia que el motor1 pero con diferente ciclo de trabajo ya que si le aplica el mismo ciclo de trabajo el robot solo estaría girando. El tiempo de 1.50 milisegundos debe de estar en alto y 2.60 milisegundos en bajo, para ello se utilizó la segunda señal PMW del PIC18F4455 que se encuentra en la terminal RC2.

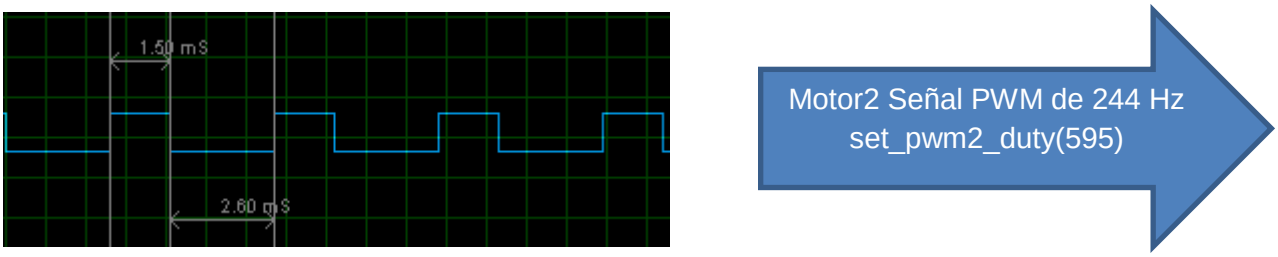
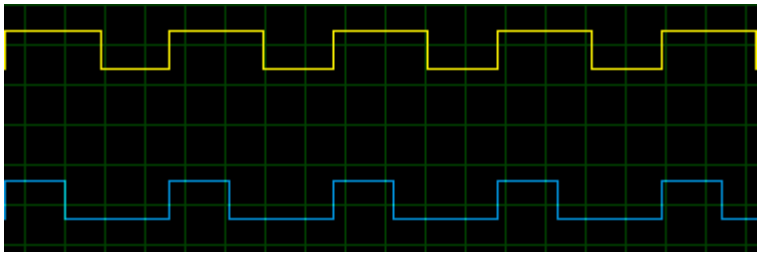


Figura 13: Ciclo de trabajo del motor2 adelante.

De esta manera se pueden ver las señales juntas para que motor1 y motor2 avancen hacia adelante con distintos ciclos de trabajos



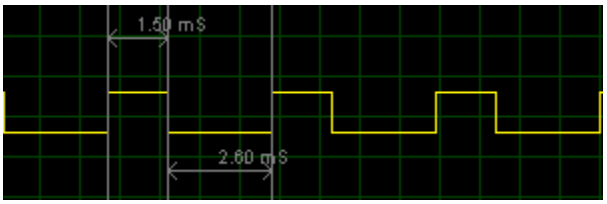
Misma señal PMW con diferentes ciclos de trabajo

Figura 14: Señales PWM adelante

ATRÁS

En ocasiones cuando el sensor SRF08 detecte un objeto muy cercano al robot, este tendrá que retroceder para eso se necesita cambiar los ciclos de trabajo de las señales PWM, ya que se sigue trabajando con la misma frecuencia.

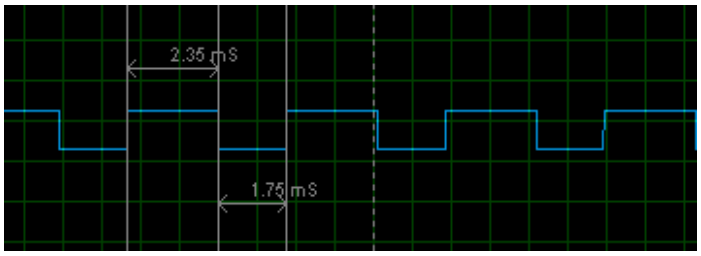
Para el motor1 se necesita 1.50 milisegundos en alto y 2.60 milisegundos en bajo



Motor1 Señal PWM de 244 Hz
set_pwm1_duty(595)

Figura 15: Ciclo de trabajo del motor1 atrás.

Para el motor2 se necesita 2.35 milisegundos en alto y 1.75 milisegundos en bajo.



Motor2 Señal PWM de 244 Hz
set_pwm2_duty(95)

Figura 16: Ciclo de trabajo del motor2 atrás.

Las dos señales para que motor1 y motor2 camine hacia atrás se ven de la siguiente forma:

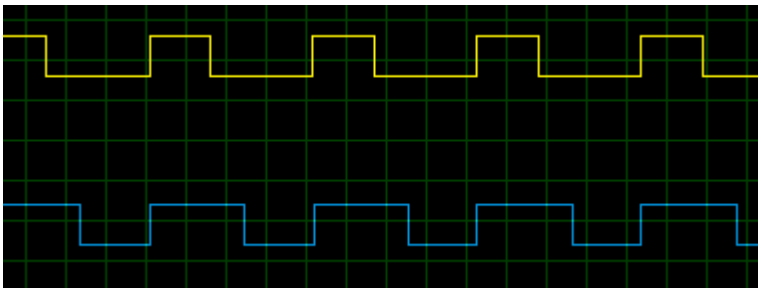
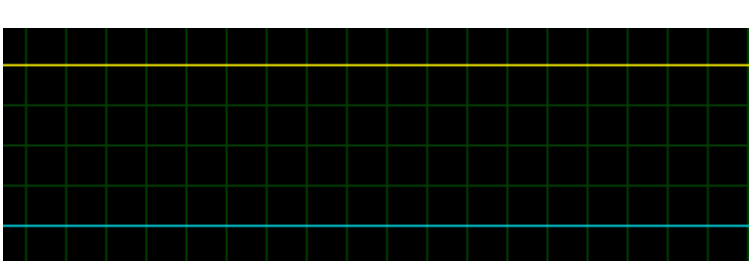


Figura 17: Señales PWM atrás.

ALTO

Esta rutina sirve para que los motores 1 y 2 se detengan y el robot no pueda avanzar, en este caso se les manda un ciclo de trabajo cero para que no realice nada los motores.



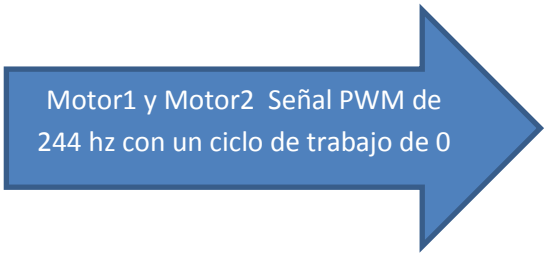


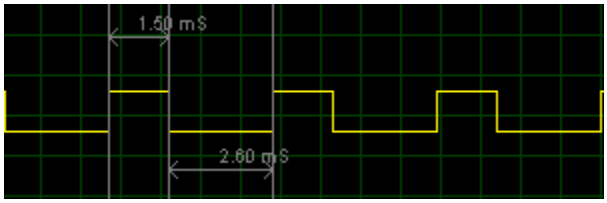
Figura 18: Señales PWM bajo.

IZQUIERDA

Esta función permitirá que el robot pueda girar hacia la izquierda cuando lo requiera únicamente el motor1, solo se activara la pmw1 ya que no es necesario activar el

motor2, para eso se trasmitirá la misma señal se ha estado trabajando pero con un ciclo de trabajo de 595.

Para el motor1 se necesita 1.50 milisegundos en alto y 2.60 milisegundos en bajo

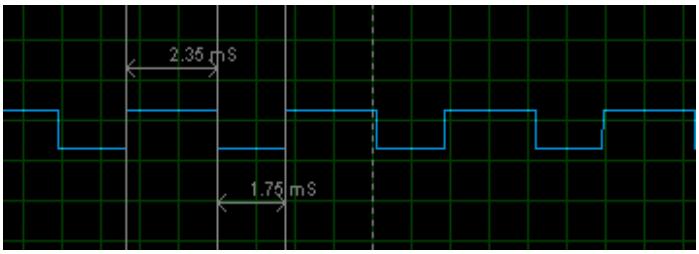


Motor1 Señal PWM de 244 Hz
set_pwm1_duty(595)

Figura 19: Señal PWM izquierda.

DERECHA

Con esta función el robot podrá girar hacia la derecha cuando este lo requiera, para ello solo se requiere activar el motor2 con un ciclo de trabajo de 95.



Motor2 Señal PWM de 244 Hz
set_pwm2_duty(95)

Figura 20: Señal PWM Derecha.

Una vez programada las rutinas necesarias se realizara el programa principal donde se estarán ejecutando todas estas rutinas junto con otros parámetros.

Programa principal

Al inicio el microcontrolador manda a llamar la función Medir1 y realiza una medición y guarda el dato en la variable distancia1 y la compara para saber si no se encuentra algún tipo de obstáculo frente al robot menor a 30 cm de ser así avanza hacia adelante en el caso contrario si se encontrara un objeto menor a 30 cm del robot, el microcontrolador ejecuta la función alto para que el robot se detenga y el servomotor donde se encuentra el sensor SRF08 gire hacia la derecha y después mande a llamar la función Medir2 y toma una nueva lectura para guardarla en una variable llamada distancia2, después el servomotor gira hacia la izquierda para que el microcontrolador mande a llamar la función Medir3 y nuevamente toma una lectura y la guarda en otra variable llamada distancia3.

Una vez realizadas las lectura las compara si distancia2 es mayor que distancia1 y distancia3 el microcontrolador ejecuta la función Derecha para se active el motor1 y el robot gire hacia la derecha y luego avance hacia adelante, en caso contrario manda a llamar a la función izquierda para que active el motor2 y gire hacia la izquierda y luego avance hacia adelante.

Si en un inicio hubiera un obstáculo menor a 15cm el robot lo primero que hace es retroceder y luego detenerse y realizar las mediciones que acabamos de describir.

El programa principal se llama Evasor y fue creado con el programa CCS C compiler V4.1. Se encuentra en el anexo B.

9. RESULTADOS, PLANOS, GRÁFICAS, PROTOTIPOS Y PROGRAMAS

Para obtener resultados se hicieron diferentes pruebas, tales como ponerles obstáculos frente al robot para ver si los puede evadir y para ver si las mediciones del SRF08 son eficientes, al estar realizando estas pruebas se obtienen buenos resultados, ya que cuando el robot detecta un objeto frente a él se detiene para que el sensor se mueva y decide hacia donde le convenía dirigirse evadiendo así el obstáculo.

Para realizar diferentes pruebas el robot didáctico se presentó en las instalaciones de la UNACH en el evento expotecnologías, el 13 de mayo de 2011.

Figura 21: evento UNACH

Figura 22: Patio cívico UNACH

Figura 23: Alumnos apreciando al robot didáctico

También se presentó en las instalaciones del CONALEP PLANTEL TUXTLA GUTIÉRREZ, EN EL EVENTO: ROBOTS RECOLECTORES DE PET EN APOYO A LOS ALUMNOS DE LA CARRERA DE EQUIPOS DE COMPUTOS. El evento fue el 27 de mayo de 2011.

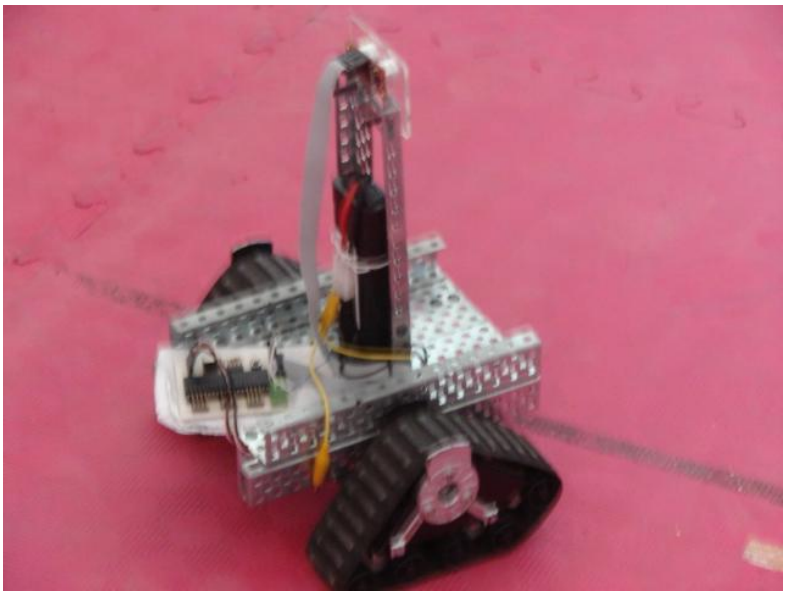


Figura 24: robot didáctico en el CONALEP

Figura 25: robot didáctico con alumnos del CONALEP

Figura 26: explicación del funcionamiento del robot a alumno del CONALEP

Figura 27: Alumnos observando al robot didáctico

También se realizaron presentaciones en el 2º foro de tecnologías de información, en el EVENTO centro de convenciones Tuxtla Gutiérrez, Chiapas. El evento fue del 16 al 18 de junio de 2011.

Figura 28: evento en el POLYFORUM

Figura 29: asesor y alumnos en el POLYFORUM

Se presentó en también en el evento EXPOVOCACIONES en el centro de convenciones Tuxtla Gutiérrez.

PROTOTIPO

Para realizar la placa principal se requiere una lista de material que a continuación se mencionan en la tabla1:

Tabla 2: material para la placa principal

MATERIAL	CANTIDAD
Pic18f4455	1
Cristal oscilador de 4Mhz	1
Capacitor cerámico de 33pF	2
Resistencia eléctrica de 4.7KΩ	1
Push button	1
Culca de dos tornillos	1
Regulador de voltaje Lm7805	1
Postes hembra	7
Postes macho	7
Conectores DB9	10
Cable plano	1m.
Placa de baquelita	5.5X7.5 cm

El esquemático de la placa principal, esta se realizó en el programa de Eagle 5.9.0

9.1. Diagrama esquemático para realizar la placa donde se encuentra montada la circuitería del robot.

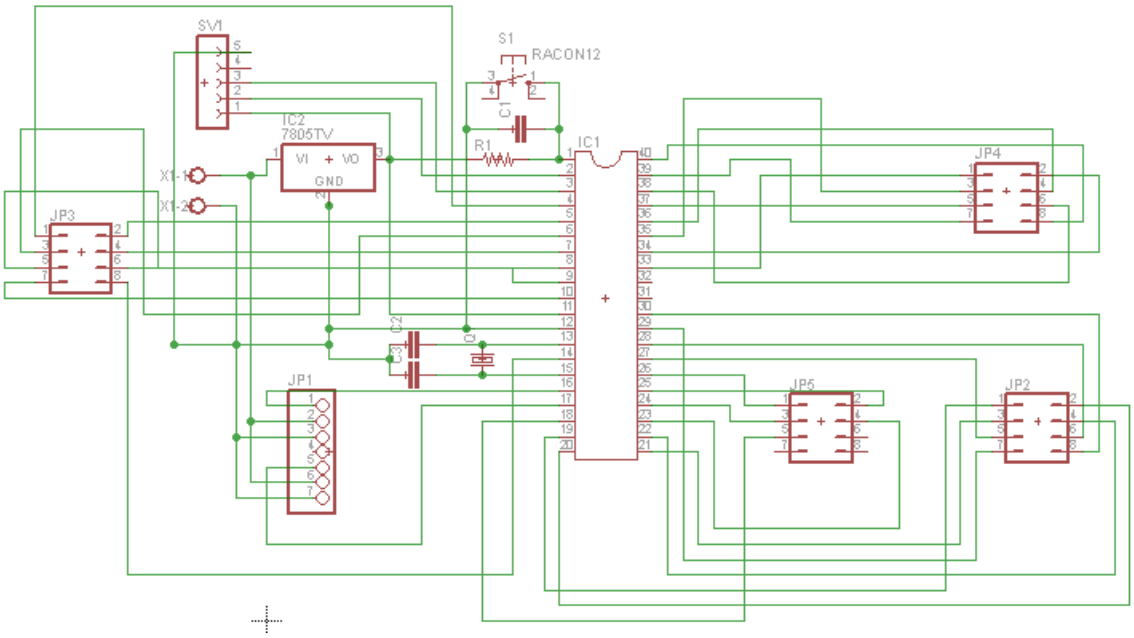


Figura 30: esquemático de la placa principal

Obtenido el diagrama se procede a realizar el rutado que está en una placa de baquelita esto se realizara en el programa de Eagle 5.9.0

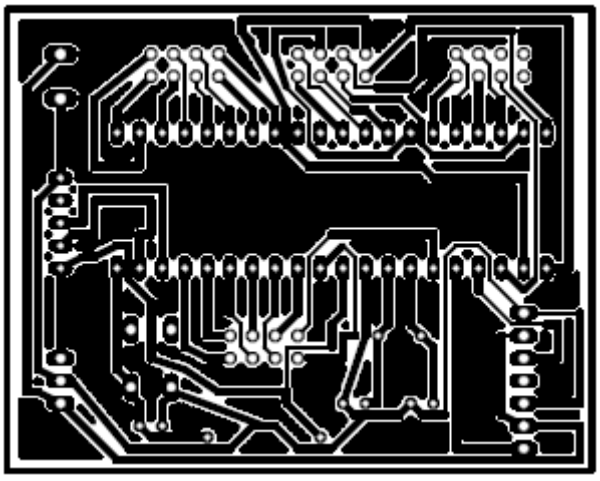


Figura 31: rutado de la placa principal

PLACA PRINCIPAL

En la figura 32 se puede apreciar la placa principal donde se encuentran montados todos los dispositivos electrónicos para que el robot didáctico pueda realizar sus funciones.

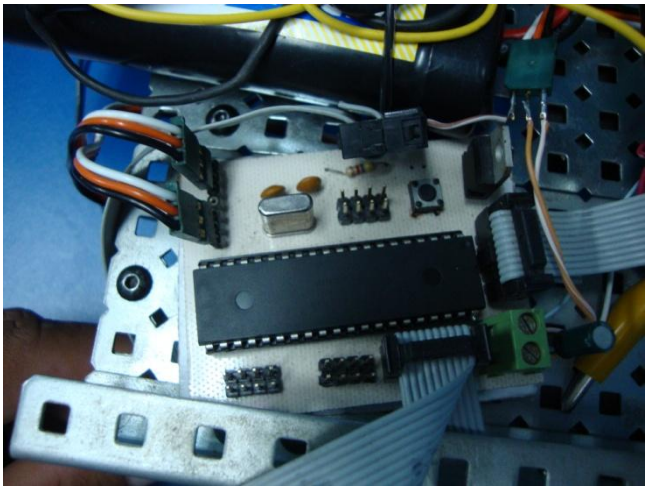


Figura 32: placa y componentes

9.2. Programas que se utilizaron para diseñar y programar

- CCS C compiler V4.1 : sirve para programar los microcontroladores.
- Proteus 7 profesional: sirve para realizar simulaciones de microcontroladores.
- Eagle 5.9.0: sirve para realizar esquemáticos y rutados de circuitos.

10. CONCLUSIONES Y RECOMENDACIONES

El objetivo del proyecto se logró diseñando y construyendo el robot didáctico en un 90%, debido a los brazos que no se pudieron anexarles porque los servomotores están en proceso de adquisición, pero utilizando todos sus componentes necesarios para poder funcionar es un avance importante. El sensor ultrasónico SRF08 es utilizado como medio de detección de objetos y se puede concluir que cuando el robot detecte un objeto a 30 cm es suficiente para poder detenerse frente al obstáculo y decidir hacía donde dirigirse.

Alguna de las recomendaciones para que este móvil sea explorador de terrenos sería agregarle una cámara y algún tipo de dispositivo para que esta se encuentre enviado videos o fotografías a hacia una computadora para tener una vista panorámica del tipo de terreno en que se encuentre y también para que esta pueda controlar el servomotor que estará moviendo al sensor junto con la cámara y decidir hacia dónde dirigirse, controlando también la señal PWM para que pueda manipular la velocidad con que el robot quiera que se mueva.

11. REFERENCIAS BIBLIOGRÁFICAS Y VIRTUALES

http://148.206.53.231/UAMI13915.PDF

www.forosdeelectronica.com

http://www.todopic.com.ar/foros/index.php?action=search2

http://www.iesjuandelacierva.es/~fremiro/Hojas%20de%20Caracteristicas/Ultrasosidos/SRF08.pdf

http://www.vexrobotics.com/products/accessories/motion/276-2162.html

http://www.todopic.com.ar/foros/index.php?topic=21559.0

12. ANEXOS

A) Registros internos del SRF08

A continuación se describe el significado de los 36 registros internos que dispone cada módulo SRF08 y que se resumen en la tabla 2:

Tabla 3: registros internos

Nº registro	Modo lectura	Modo escritura
0 – 0x00	Se lee la revisión actual del firmware interno	Registro de comandos
1 – 0x01	Valor del sensor de luz ambiente	Registro de ganancia
2 – 0x02	Byte alto del 1er. eco recibido	Registro de rango
3 – 0x03	Byte bajo del 1er. eco recibido	No disponible
....	....	....
34 - 0x22	Byte alto del eco Nº 17 recibido	No disponible
35 - 0x23	Byte bajo del eco Nº 17 recibido	No disponible

Los únicos registros que pueden ser escritos corresponden a los de las posiciones 0, 1 y 2. La posición 0 es el registro de comandos. Cada vez que se escribe un comando sobre el mismo, el módulo SRF08 inicia una nueva medida ultrasónica. Si se lee esta posición no se lee el último comando ejecutado, sino la versión del firmware interno del propio módulo.

La posición 1 es, en el modo de escritura, el registro de ganancia, donde se puede variar el factor de amplificación interno del SRF08. Cuando se lee esta posición, se lee el valor procedente del sensor de luz ambiente (LDR) disponible en el módulo SRF08. Este valor se actualiza cada vez que finaliza la ejecución de un comando.

La posición 2, en el modo de escritura, se corresponde con el registro de rango. En el mismo se puede establecer el rango de medidas que se desea realizar. En el modo de lectura este registro, junto con el de la posición 3.

Posición 3, contienen la parte alta y baja del valor del primer eco recibido. La unidad de medida con la que se representa este eco depende del comando empleado y puede expresar pulgadas, centímetros o tiempo empleado por el haz ultrasónico en hacer el recorrido de ida y vuelta (en mS). Si se lee el valor 0 indica que no se ha detectado ningún objeto. Las 32 posiciones, de la 4 hasta la 35, son únicamente de lectura. Expresan 16 valores de 16 bits (1º byte alto y 2º byte bajo) que corresponden con la medida de otros 16 ecos recibidos desde objetos más distantes.

La figura 33 muestra el diagrama de tiempos que permite el acceso de cualquiera de los registros internos del módulo SRF08. Toda comunicación comienza con el bit de inicio.
A continuación se envía la dirección del módulo deseado (o la 0x00 de llamada general) en modo escritura. Seguidamente se transmite la dirección o número de registro interno al que se quiere acceder. En este punto se vuelve a generar un bit de inicio y a mandar la dirección del módulo pero en modo de lectura. Acto seguido se empieza a recibir el contenido de los registros internos a partir del indicado.

Cuando se reciba el último byte deseado, se debe transmitir el bit NACK (“1”) y el bit de stop.

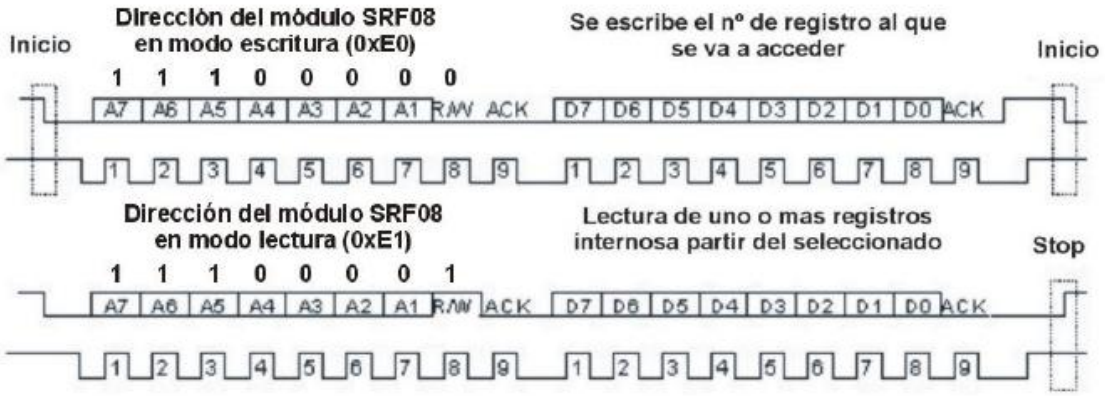


Figura 33. Diagrama de tiempos.

Comandos

El registro de comandos admite un total de 9 códigos diferentes con los cuales se realiza una determinada operación. Estos se resumen en la tabla 4:

Tabla 4: Comandos

COMANDO		DESCRIPCION
Dec.	Hex.	
80	0x50	Modo de medición, el resultado se ofrece en pulgadas
81	0x51	Modo de medición, el resultado se ofrece en centímetros
82	0x52	Modo de medición, el resultado se ofrece en micro segundos
83	0x53	Modo ANN, el resultado se ofrece en pulgadas
84	0x54	Modo ANN, el resultado se ofrece en centímetros
85	0x55	Modo ANN, el resultado se ofrece en micro segundos
160	0xA0	Cambio de la dirección I2C, 1ª secuencia
170	0xAA	Cambio de la dirección I2C, 2ª secuencia
165	0xA5	Cambio de la dirección I2C, 3ª secuencia

Modo de medición

Para realizar una nueva medida se escribe uno de los tres códigos sobre el registro de comandos, se espera el tiempo necesario para completar el comando y, finalmente, se lee el resultado desde cualquiera de las posiciones que representan el buffer con los diferentes ecos recibidos (posiciones 2-35). Estas posiciones (buffer) se ponen a 0 cada vez que se inicia una nueva medida. El primer eco recibido se almacena en las posiciones 2 y 3, el segundo en las 4 y 5, etc. Se recogen un total de 17 posibles ecos. Si las posiciones correspondientes a un determinado eco valen 0, los siguientes ecos también lo valdrán. El análisis de los 17 ecos posibles permite hacer una composición del objeto detectado frente al módulo SRF08: objeto irregular, objetos superpuestos, etc.
El tiempo recomendado y por defecto entre una medida y la siguiente es de 65 mS. Este tiempo se puede reducir escribiendo un valor distinto en el registro de rango (posición 2) antes de realizar una nueva medida. Cuando finaliza la medida en curso también se actualiza el valor captado por la célula LDR y que mide la luz ambiente. Dicho valor se puede leer desde la posición 1.

Cambio de dirección I^2C

La dirección I2C por defecto del módulo SRF08 es la 0XE0. Con objeto de que en un mismo bus puedan conectarse varios módulos, es imprescindible que cada uno tenga una dirección I2C diferente.

Para proceder a cambiar la dirección I2C de un módulo es necesario que sólo haya uno conectado al bus en cada momento y escribir sobre el mismo una secuencia de 4 comandos en el orden apropiado. Por ejemplo, para cambiar la dirección actual de un módulo SRF08 (0xE0 por defecto) por la dirección 0xF2, se escribe la siguiente secuencia sobre el registro de comandos: 0xA0, 0xAA, 0xA5, 0xF2. El módulo queda configurado con la nueva dirección I2C y se recomienda etiquetarlo. En cualquier caso, si el usuario olvidara la nueva dirección actual, el módulo la indica mediante una serie de intermitencias del led que hay en su parte posterior, siempre que se

conecta la alimentación del mismo. Este código de intermitencias se muestra a continuación en la tabla 5.

Tabla 5: Direcciones

DIRECCION		Intermitencias largas	Intermitencias cortas
Dec.	Hex.		
224	0xE0	1	0
226	0xE2	1	1
228	0xE4	1	2
230	0xE6	1	3
232	0xE8	1	4
234	0xEA	1	5
236	0xEC	1	6
238	0xEE	1	7
240	0xF0	1	8
242	0xF2	1	9
244	0xF4	1	10
246	0xF6	1	11
248	0xF8	1	12
250	0xFA	1	13
252	0xFC	1	14
254	0xFE	1	15

El fin de una medida

El tiempo mínimo que se debe emplear entre una medida y la siguiente es de 65mS. De todas formas, cuando el módulo SRF08 se encuentra ocupado, no responde a ninguna transferencia ni comando enviado vía I2C. No es necesario por tanto esperar los 65mS. Se puede tratar de realizar una transferencia que, si el módulo no responde o nos devuelve 0xFF, es porque se encuentra ocupado realizando la medida anterior.

Ajuste del rango

El máximo rango de medida del SRF08 queda determinado por un timer interno. Por defecto este timer es de 65mS que equivale a una distancia de unos 11m (65000/58=1120cm), teniendo en cuenta que la velocidad del sonido es de 58 mS/cm de ida y vuelta. Es posible reducir el tiempo de espera del eco y con ello el rango de medida, escribiendo el valor apropiado sobre el registro de rango de la posición 2. Este rango se ajusta en pasos de 43mm (unos 58mS) hasta los 11m. El rango total de medida se obtiene de multiplicar por 43 el valor actual del registro de rango y sumarle 43 ((reg.Rango x 43)+43). Si el registro vale 0x00, el rango de medida es de 43mm. Si vale 1, el rango aumenta hasta los 11m.

El rango total de medida se obtiene de multiplicar por 43 el valor actual del registro de rango y sumarle 43 ((reg.Rango x 43)+43). Si el registro vale 0x00, el rango de medida es de 43mm. Si vale 1, el rango aumenta hasta 86mm. Los valores mas utilizados son de 24 (0x18) cuyo rango es de 1m y de 140 (0x8C) que corresponde a 6m. Con el valor 255 (0xFF) se establece el rango original de 11m (65mS). Pueden existir razones para reducir el rango: permite medir distancias cortas, es una forma mas rápida de obtener el resultado de una medida y permite hacer mas medidas por unidad de tiempo (menos tiempo entre una medida y la siguiente). Si se reduce el rango es posible que también haya que reducir la ganancia de los amplificadores internos.

Aplicaciones

El módulo SRF08 es capaz de proporcionar información acerca de la distancia que hay entre el propio módulo y un objeto. También es capaz de medir la luz ambiente. Las aplicaciones son numerosas, citamos unas cuantas a modos de ejemplo:

- Aplicaciones de control conde se deba actuar en función de la distancia o tamaño de objetos diversos.
- Alarmas activadas cuando el intruso se aproxima a una determinada distancia
- Medición de luz ambiente

- ***Microbótica en donde es necesario que se actúe en función de la distancia que separa al robot de cualquier otro objeto.***

Condiciones de START y STOP:

Antes de que se establezca un intercambio de datos entre el circuito Master y los Esclavos, el Master debe informar el comienzo de la comunicación (condición de Start): la línea SDA cae a cero mientras SCL permanece en nivel alto. A partir de este momento comienza la transferencia de datos. Una vez finalizada la comunicación se debe informar de esta situación (condición de Stop). La línea SDA pasa a nivel alto mientras SCL permanece en estado alto.

B) Programas en CCS

Mover al servo del sensor ultrasónico

```
#INT_TIMER0                          // interrupción  del timer0 para el servo
void Generacion_pwm()
  {
  Pulso_pwm++;                       //Incremento cada rebose del timer0
    if (Pulso_pwm==0)
     {
     OUTPUT_HIGH(PIN_B7);
     }
  if (Pulso_pwm==PWM)
     {
     OUTPUT_LOW(PIN_B7);
     }
  set_timer0(154);                    //Se fija un periodo de onda pwm de 20msg.
  }
```

A continuación se muestran las rutinas de mediciones:

Mediciones

```
Medir1(int8 &distancia1){       //Rutina Medir1 se declara una variable de tipo entera de
                                  8bits con un apuntador llamado distacia1 donde quedara
                                  guardada la distancia a la que se encuentra el objeto

  char distancia_L, distancia_H;    //Variables donde quedan guardados los pulso para el
                                    calculo de la distancia.
  i2c_start();                      // Bit de Start.
  i2c_write(0xE2);                  // Dirección del SRF08 en escritura.
  i2c_write(0x00);                  // El registro de comando está en la ubicación 0.
  i2c_write(0x51);                  // 0x51 es el comando de cálculo de distancia en cm.
  i2c_stop();                       // Bit de Stop.
  delay_ms(70);                     // Espera más de 65 ms hasta la lectura.
  i2c_start();                      // Bit de Start.
  i2c_write(0xE2);                  // Dirección del SRF08 en escritura.
  i2c_write(0x02);                  // Apunta a la ubicación 0x01 que es a partir donde se
  i2c_stop();                       // encuentran los registros para leer luz y distancia.
  i2c_start();
  i2c_write(0xE3);                  // Dirección del SRF08 en lectura.
  distancia_H = i2c_read(1);        // Lee el byte alto de la distancia.
  distancia_L = i2c_read(0);        // Lee el byte bajo de la distancia.
  i2c_stop();
  distancia1= (distancia_H + distancia_L);          // Valor de la distancia.

  }

Medir2(int8 &distancia2){       //Rutina Medir2 se declara una variable de tipo entera de
                                  8bits con un apuntador llamado distacia2 donde quedara
                                  guardada la distancia a la que se encuentra el objeto

  char distancia_L, distancia_H;    //Variables donde quedan guardados los pulso para el
                                    calculo de la distancia.
  i2c_start();                      // Bit de Start.
  i2c_write(0xE2);                  // Dirección del SRF08 en escritura.
  i2c_write(0x00);                  // El registro de comando está en la ubicación 0.
  i2c_write(0x51);                  // 0x51 es el comando de cálculo de distancia en cm.
  i2c_stop();                       // Bit de Stop.
  delay_ms(70);                     // Espera más de 65 ms hasta la lectura.
  i2c_start();                      // Bit de Start.
```

```
    i2c_write(0xE2);                // Dirección del SRF08 en escritura.
    i2c_write(0x02);                // Apunta a la ubicación 0x01 que es a partir donde se
    i2c_stop();                     // encuentran los registros para leer luz y distancia.
    i2c_start();
    i2c_write(0xE3);                // Dirección del SRF08 en lectura.
    distancia_H = i2c_read(1);      // Lee el byte alto de la distancia.
    distancia_L = i2c_read(0);      // Lee el byte bajo de la distancia.
    i2c_stop();
    distancia2= (distancia_H + distancia_L);// Valor de la distancia.


    }

Medir3(int8 &distancia3){       //Rutina Medir1 se declara una variable de tipo entera de 8bits con un apuntador llamado distacia3 donde quedara guardada la distancia a la que se encuentra el objeto


    char distancia_L, distancia_H;      //Variables donde quedan guardados los pulso para el calculo de la distancia.
    i2c_start();                        // Bit de Start.
    i2c_write(0xE2);                    // Dirección del SRF08 en escritura.
    i2c_write(0x00);                    // El registro de comando está en la ubicación 0.
    i2c_write(0x51);                    // 0x51 es el comando de cálculo de distancia en cm.
    i2c_stop();                         // Bit de Stop.
    delay_ms(70);                       // Espera más de 65 ms hasta la lectura.
    i2c_start();                        // Bit de Start.
    i2c_write(0xE2);                    // Dirección del SRF08 en escritura.
    i2c_write(0x02);                    // Apunta a la ubicación 0x01 que es a partir donde se
    i2c_stop();                         // encuentran los registros para leer luz y distancia.
    i2c_start();
    i2c_write(0xE3);                    // Dirección del SRF08 en lectura.
    distancia_H = i2c_read(1);          // Lee el byte alto de la distancia.
    distancia_L = i2c_read(0);          // Lee el byte bajo de la distancia.
    i2c_stop();
    distancia3= (distancia_H + distancia_L);// Valor de la distancia.
   }
```

Adelante

```
void Adelante(){                                    //Rutina adelante no recibe nada y no devuelve
nada
   setup_ccp1(CCP_OFF);setup_ccp2(CCP_OFF);  //Necesitamos apagar las señales pwm para
                                                no tener problemas al momentos de
                                                mandarlas a llamar.
   setup_ccp1(CCP_PWM);setup_ccp2(CCP_PWM); // Inicializamos las señales pwm.
   set_pwm1_duty(95);          //Primer motor con un ciclo de trabajo 95
   delay_ms(50);               //Retardo de 50 milisegundos.
   set_pwm2_duty(595);         //Segundo motor con un ciclo de trabajo de 595
   delay_ms(50);               //Retardo de 50 milisegundos.
}
```

Atras

```
void Atras(){                          //Rutina atrás no recibe nada y no devuelve  nada
   setup_ccp1(CCP_OFF);setup_ccp2(CCP_OFF);  //Necesitamos apagar las señales pwm para
                                                no tener problemas al momentos de
                                                mandarlas a llamar.
   setup_ccp1(CCP_PWM);setup_ccp2(CCP_PWM); // Inicializamos las señales pwm.
   set_pwm1_duty(595);         //Primer motor con un ciclo de trabajo 95
   delay_ms(50);               //Retardo de 50 milisegundos.
   set_pwm2_duty(95);//Segundo motor con un ciclo de trabajo de 595
   delay_ms(50);               //Retardo de 50 milisegundos.
}
```

Alto

```
void Alto(){                           //Rutina alto no recibe nada y no devuelve  nada
   setup_ccp1(CCP_OFF);setup_ccp2(CCP_OFF); //Necesitamos apagar las señales pwm
                                                para no tener problemas al momentos
                                                de mandarlas a llamar.
   setup_ccp1(CCP_PWM);setup_ccp2(CCP_PWM);       //Inicializamos las señales pwm.
   set_pwm1_duty(0);                              //Primer motor con un ciclo de trabajo 0
   set_pwm2_duty(0);                               //Segundo motor con un ciclo de trabajo 0
}
```

Izquierda

```
void Izquierda(){                                   //Rutina atrás no recibe nada y no devuelve nada
   setup_ccp1(CCP_OFF);setup_ccp2(CCP_OFF);         //Necesitamos apagar las señales pwm para no
                                                       tener problemas al momentos de
                                                       mandarlas a llamar.
   setup_ccp1(CCP_PWM);                             // Inicializamos la señal pwm1.
  set_pwm1_duty(595);                               //Primer motor con un ciclo de trabajo 595

   delay_ms(50);                                    //Retardo de 50 milisegundos.
}
```

Derecha

```
void Derecha(){                                     //Rutina atrás no recibe nada y no devuelve  nada
   setup_ccp1(CCP_OFF);setup_ccp2(CCP_OFF);         //Necesitamos apagar las señales pwm
                                                       para no tener problemas al momentos
                                                       de
                                                       mandarlas a llamar.
   setup_ccp2(CCP_PWM);             // Inicializamos la señales pwm2.
  set_pwm1_duty(95);                //Primer motor con un ciclo de trabajo 595
   delay_ms(50);                                    //Retardo de 50 milisegundos.
}
```

Programa principal

```
#include<18f4455.h>
#fuses NOWDT,XT
#use delay(clock=4Mhz)
#use i2c(Master,sda=PIN_A0,scl=PIN_A1)
#include "Adelante.c"
#include "Izquierda.c"
#include "Derecha.c"
#include "Alto.c"
#include "Atras.c"
#byte trisc=0x87
#byte portc=0x07
int8 PWM=0;                                         //Guardará los valores de la señal PWM
int8 Pulso_pwm=0;                                   //Ancho del pulso de control del servo

#INT_TIMER0
void Generacion_pwm()
```

```
{
Pulso_pwm++;                          //Incremento cada rebose del timer0

if (Pulso_pwm==0)
  {
  OUTPUT_HIGH(PIN_C7);
  }
if (Pulso_pwm==PWM)
  {
  OUTPUT_LOW(PIN_C7);
  }
set_timer0(154);                      //Se fija un periodo de onda pwm de 20msg.
}
   Medir1(int8 &distancia1){      //Rutina Medir1 se declara una variable de tipo entera de
                                    8bits con un apuntador llamado distacia1 donde quedara
                                    guardada la distancia a la que se encuentra el objeto

  char distancia_L, distancia_H;  //Variables donde quedan guardados los pulso para el
                                    calculo de la distancia.
  i2c_start();                    // Bit de Start.
  i2c_write(0xE2);                // Dirección del SRF08 en escritura.
  i2c_write(0x00);                // El registro de comando está en la ubicación 0.
  i2c_write(0x51);                // 0x51 es el comando de cálculo de distancia en cm.
  i2c_stop();                     // Bit de Stop.
  delay_ms(70);                   // Espera más de 65 ms hasta la lectura.
  i2c_start();                    // Bit de Start.
  i2c_write(0xE2);                // Dirección del SRF08 en escritura.
  i2c_write(0x02);                // Apunta a la ubicación 0x01 que es a partir donde se
  i2c_stop();                     // encuentran los registros para leer luz y distancia.
  i2c_start();
  i2c_write(0xE3);                // Dirección del SRF08 en lectura.
  distancia_H = i2c_read(1);      // Lee el byte alto de la distancia.
  distancia_L = i2c_read(0);      // Lee el byte bajo de la distancia.
  i2c_stop();
  distancia1= (distancia_H + distancia_L);          // Valor de la distancia.

  }

Medir2(int8 &distancia2){      //Rutina Medir2 se declara una variable de tipo entera de
                                 8bits con un apuntador llamado distacia2 donde quedara
                                 guardada la distancia a la que se encuentra el objeto
```

```
    char distancia_L, distancia_H;      //Variables donde quedan guardados los pulso para el
                                        calculo de la distancia.
    i2c_start();                        // Bit de Start.
    i2c_write(0xE2);                    // Dirección del SRF08 en escritura.
    i2c_write(0x00);                    // El registro de comando está en la ubicación 0.
    i2c_write(0x51);                    // 0x51 es el comando de cálculo de distancia en cm.
    i2c_stop();                         // Bit de Stop.
    delay_ms(70);                       // Espera más de 65 ms hasta la lectura.
    i2c_start();                        // Bit de Start.
    i2c_write(0xE2);                    // Dirección del SRF08 en escritura.
    i2c_write(0x02);                    // Apunta a la ubicación 0x01 que es a partir donde se
    i2c_stop();                         // encuentran los registros para leer luz y distancia.
    i2c_start();
    i2c_write(0xE3);                    // Dirección del SRF08 en lectura.
    distancia_H = i2c_read(1);          // Lee el byte alto de la distancia.
    distancia_L = i2c_read(0);          // Lee el byte bajo de la distancia.
    i2c_stop();
    distancia2= (distancia_H + distancia_L);// Valor de la distancia.


    }

Medir3(int8 &distancia3){       //Rutina Medir1 se declara una variable de tipo entera de
                                8bits con un apuntador llamado distacia3 donde quedara
guardada la                     distancia a la que se encuentra el objeto


    char distancia_L, distancia_H;      //Variables donde quedan guardados los pulso para el
                                        calculo de la distancia.
    i2c_start();                        // Bit de Start.
    i2c_write(0xE2);                    // Dirección del SRF08 en escritura.
    i2c_write(0x00);                    // El registro de comando está en la ubicación 0.
    i2c_write(0x51);                    // 0x51 es el comando de cálculo de distancia en cm.
    i2c_stop();                         // Bit de Stop.
    delay_ms(70);                       // Espera más de 65 ms hasta la lectura.
    i2c_start();                        // Bit de Start.
    i2c_write(0xE2);                    // Dirección del SRF08 en escritura.
    i2c_write(0x02);                    // Apunta a la ubicación 0x01 que es a partir donde se
    i2c_stop();                         // encuentran los registros para leer luz y distancia.
    i2c_start();
    i2c_write(0xE3);                    // Dirección del SRF08 en lectura.
```

```
    distancia_H = i2c_read(1);                  // Lee el byte alto de la distancia.
    distancia_L = i2c_read(0);                  // Lee el byte bajo de la distancia.
    i2c_stop();
    distancia3= (distancia_H + distancia_L);// Valor de la distancia.
   }

void main(int8 x, int8 y, int8 z){
      setup_timer_2(RTCC_INTERNAL|T2_DIV_BY_16,255,1);
      setup_timer_0(RTCC_DIV_1 | RTCC_INTERNAL | RTCC_8_BIT);    //Interrupción generación PWM
  enable_interrupts (GLOBAL);
   while(true){
  enable_interrupts(INT_TIMER0);
  PWM=2; delay_ms(50);                                  //El servo se encuentra en el centro
  disable_interrupts(INT_TIMER0);
    Medir1(x);
    if(x>=32){
      Adelante();
    }
    else
    if(x<=15){
      Atras(); delay_ms(300);
      setup_ccp1(CCP_OFF);setup_ccp2(CCP_OFF);
      Alto(); delay_ms(100);
      setup_ccp1(CCP_OFF);setup_ccp2(CCP_OFF);
      goto a;
    }
    else
    if(x<=30){
      Alto(); delay_ms(100);
      setup_ccp1(CCP_OFF);setup_ccp2(CCP_OFF);
      a:
      enable_interrupts(INT_TIMER0);
      PWM=1; delay_ms(50);
      disable_interrupts(INT_TIMER0);
      Medir2(y);
      enable_interrupts(INT_TIMER0);
      PWM=2; delay_ms(50);
      disable_interrupts(INT_TIMER0);
```

```
        enable_interrupts(INT_TIMER0);
        PWM=4; delay_ms(50);
        disable_interrupts(INT_TIMER0);
        Medir3(z);

        if((y>x) && (y>z)) {
          enable_interrupts(INT_TIMER0);
          PWM=2; delay_ms(50);
          disable_interrupts(INT_TIMER0);
          Izquierda(); delay_ms(450);
          setup_ccp2(CCP_OFF);
          Alto(); delay_ms(100);

          setup_ccp1(CCP_OFF);setup_ccp2(CCP_OFF);

          Adelante();
         }
        else
        if((z>x) && (z>y)){
          enable_interrupts(INT_TIMER0);
          PWM=2; delay_ms(50);
          disable_interrupts(INT_TIMER0);
          Derecha();delay_ms(450);
          setup_ccp1(CCP_OFF);
          Alto(); delay_ms(100);
          setup_ccp1(CCP_OFF);setup_ccp2(CCP_OFF);
          Adelante();
        }
      }
    }
}
```