Guía para el Proyecto 6: Esquivador de Obstáculos

Club de Robótica - FI - UNLP

1. Introducción

La idea de la siguiente práctica es lograr familiarizarse con los sensores y actuadores (sensor de proximidad, motores) que conformarían un Esquivador de Obstáculos para su posterior implementación. Para ello se darán una serie de ejercicios que promuevan la utilización de los mismos mediante el Arduino UNO.

Los elementos de circuito que se deben dominar para realizar un esquivador de obstáculos son básicamente:

- Motores de corriente continua
- Servomotor
- Sensor de proximidad HC-SR04

2. Sugerencias para antes de programar

Para lograr hacer los programas en el Arduino satisfactoriamente es conveniente tener en cuenta previamente los siguientes ítems:

1. Tener las nociones básicas de programación en Arduino, así como saber manipular las funciones más utilizadas.
2. Conocer los diferentes módulos que dispone el microcontrolador del Arduino (Timmer, ADC, UART).
3. Segmentar el programa del controlador en subprogramas, para facilitar el desarrollo del mismo.
4. Implementar los programas en alguno de los prototipos desarrollados por el CDR FI-UNLP.
5. Contar con los esquemáticos necesarios que describan el pinout de cada placa o shield implementado en el proyecto.
6. Contar con las hojas de datos de todos los elementos de circuito utilizados.

3. Ejercicios para practicar con el Arduino

Ejercicio 1: Motores de Corriente Continua

Los motores utilizados para la tracción de las ruedas funcionan con tensión continua, y para invertir el giro en los mismos basta con invertir la polaridad de la tensión aplicada. Recordar que el motor izquierdo está controlado por los pines 3 y 11 del Arduino, mientras que el derecho lo está por los pines 5 y 8.

a) Realizar un programa en Arduino que haga mover al carro en línea recta hacia adelante y luego hacia atrás ciclicamente.

   Ayuda: Recordar que los pines que serán salida deberán ser configurados previamente como tales con la instrucción **pinMode(x,y)**, donde **x** es el numero de pin a configurar, e **y** debe ser **INPUT** (para configurarlo como entrada) u **OUTPUT** (para configurarlo como salida). Para poner algún valor digital en los pines de salida se utiliza la instrucción **digitalWrite(x,y)**, donde **x** es el numero de pin a setear, e **x** debe ser **HIGH** (para que sea un 1 lógico) o **LOW** (para que sea un 0 lógico).

b) Realizar un programa en Arduino que haga girar al carro en una circunferencia de un determinado radio.

   Ayuda: Para llevarlo a cabo es necesario tener en cuenta que los motores del carro deben girar a velocidades diferentes para efectuar el giro. Para ello es conveniente utilizar señales de PWM y asignar un ancho de pulso distinto en cada motor. Recordar que para generar señales de PWM se utiliza la instrucción **analogWrite(x,y)**, donde **x** es el número de pin por donde se quiere sacar la señal PWM, e **y** es un numero entre 0 y 255 que indica el ciclo de trabajo (0 para el 0 % y 255 para el 100 %)

Ejercicio 2: Servomotor

Los servomotores son motores microcontrolados que se caracterizan por poder ubicarse en cualquier posición (dentro de su rango de operación) y mantenerse estable en ella. El servomotor utilizado en el proyecto tiene un ángulo de giro de 180 grados y posee un pin de control a través del cual se lo comanda (la placa shield hecha por el CDR utiliza el pin 9 del Arduino para hacerlo).

a) Realizar un programa en Arduino que haga girar al servomotor desde un extremo del rango de operación hasta el otro, y lo vuelva a hacer ciclicamente.

   Ayuda: En la programación en Arduino existen funciones que facilitan enormemente el manejo de los servomotores. La librería **Servo** contiene todas las funciones necesarias para controlarlo. Para agregar la librería al proyecto basta con añadir una línea al inicio del programa que diga **#Include <Servo.h>**.

   A continuación se muestra un ejemplo de como utilizar dicha librería:

```
#include <Servo.h> // Incluye la librería para usar el servomotor.

Servo hola;  // Es necesario crear una estructura para
             // controlar al servomotor.
void setup()
{
  hola.attach(9);  //  Configura al pin 9 para controlar
}                  //  al servomotor.
void loop()
{                        // Mueve al servomotor a la posición pos
  hola.write(pos);       // (pos puede valer entre 0 y 179 [grados]).
  delay(15);             // Retardo necesario para esperar que se
}                        // mueva el servomotor.
```

Figura 1: Uso de la librería Servo

Ejercicio 3: Sensor de proximidad HC-SR04

El sensor HC-SR04 permite medir la distancia que hay con un determinado objeto a través de pulsos de ultrasonido. Básicamente el funcionamiento consta en enviar pulsos de ultrasonido y esperar a que retorne el eco del mismo. El tiempo que tarda en regresar el eco depende proporcionalmente de la distancia del objeto. El sensor tiene un pin para accionar el mecanismo de sensado (Trigger Input), y otro pin por donde se recibe la información del mismo (Echo Pulse Output):

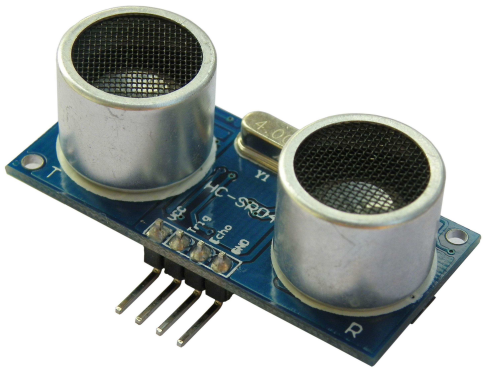

Figura 2: Sensor HC-SR04

El diagrama de tiempos de la *Figura 3* indica que es necesario enviar un pulso de $10\mu S$ por el pin Trigger Output para comenzar el sensado, luego el módulo HC-SR04 enviará 8 pulsos de una señal de ultrasonido de 40kHz y esperará el eco. Una vez recibido el eco, el sensor generará un pulso de duración Δt en el pin Echo Pulse Output. La relación entre la distancia y el ancho del pulso generado está dada por la relación:

$$distancia = \Delta t * 340\frac{m}{s}/2$$

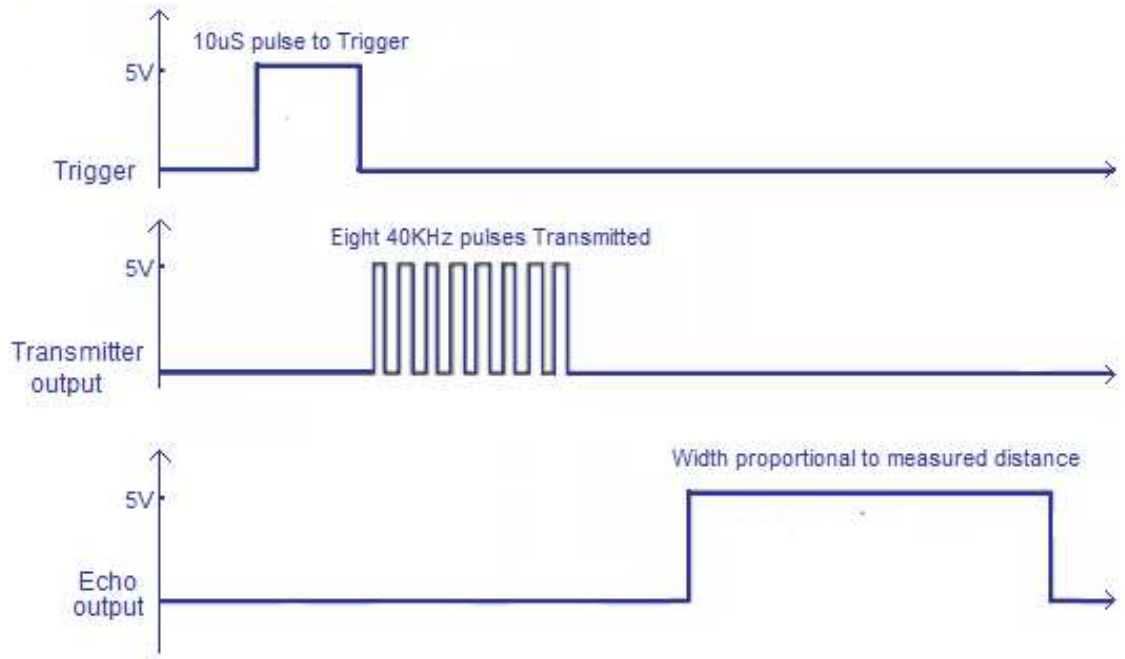

Figura 3: Diagrama temporal de señales para el sensor HC-SR04

a) Realizar un programa en Arduino para medir la distancia que hay entre el sensor y un objeto, y enviarla a la PC a través del puerto serie para poder visualizarla en pantalla.

Ayuda: Para utilizar el sensor fácilmente con Arduino se debe incluir la librería **Ultrasonic** añadiendo al inicio del programa **#Include <Ultrasonic.h>**. En la *figura 4* se muestra un ejemplo de como utilizar dicha librería:

```
#include <Ultrasonic.h>     // Para incluir la librería Ultrasonic.

Ultrasonic ultrasonic(a,b); // a es el número de PIN asignado al Trigger.
                            // b es el número de PIN asignado al Echo.
void setup() {
}

void loop()
{
  ultrasonic.Timing();    // Devuelve el tiempo de rebote en ms.
  ultrasonic.Ranging(CM); // Devuelve la distancia del objeto en cm.
  delay(100);
}
```

Figura 4: Uso de la librería Ultrasonic

También, para utilizar el módulo UART para la comunicación por el puerto serie se debe utilizar las funciones **Serial**. En la figura 5 se muestran las instrucciones más comunes:

```
void setup()
{
  Serial.begin(9600); // Establece la velocidad de tranferencia de datos en baudios/s.
}                     // Normalmente se elige 9600 baudios/s.
void loop()
{
  Serial.print();    // Imprime los datos al puerto serie como texto ASCII.
  Serial.println();  // Imprime los datos al puerto serie como texto ASCII,
                     // seguido de un caracter de avance de línea.
}
```

Figura 5: Uso de las funciones Serial