

# Taller de Interrupciones

Organización del Computador 1

Primer Cuatrimestre 2017

## 1. Introducción

Para este taller se propone utilizar un **ArduinoMega** como controlador de un robot móvil que cuenta con distintos sensores. La funcionalidad deseada para el robot es que sea capaz de seguir una línea blanca sobre un fondo negro, utilizando dichos sensores.

El taller será resuelto (y evaluado) en un simulador *online* de circuitos, en el cual se programará un **ArduinoUno** (similar al **ArduinoMega** en términos de la funcionalidad a utilizar). Opcionalmente, la solución obtenida podrá ser observada ejecutando en el robot real.

El simulador se accede desde la siguiente dirección: [www.123d.circuits.io](http://www.123d.circuits.io).

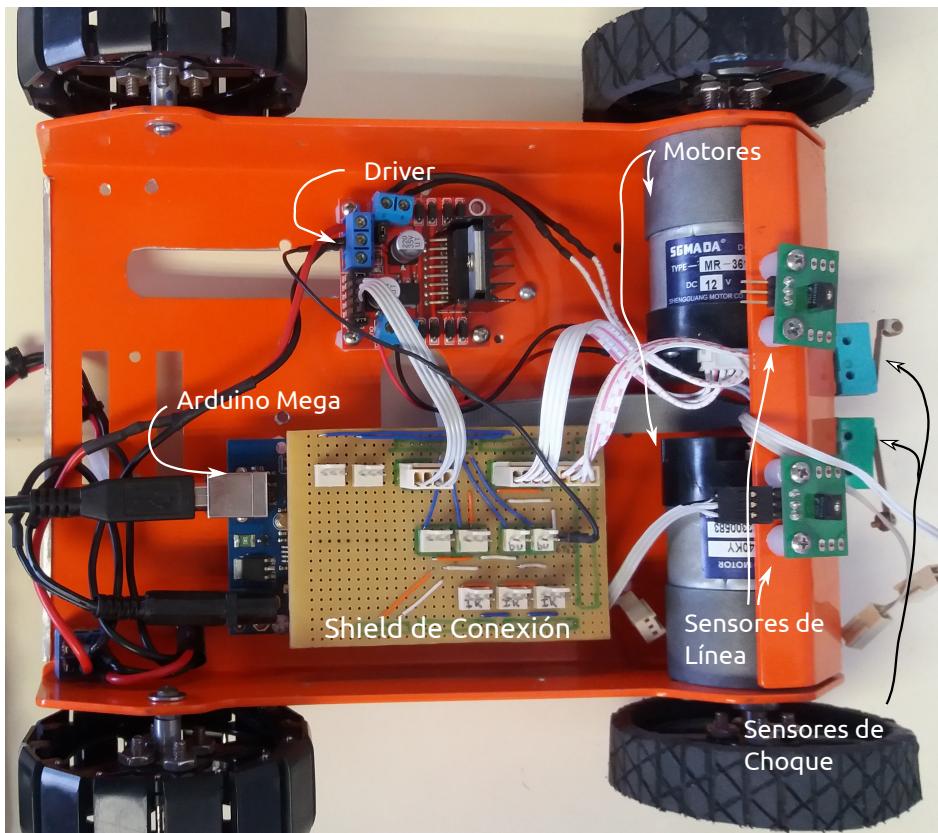


Figura 1: El robot

## El robot

El robot que se va a utilizar posee un **Arduino**, diversos sensores y dos motores controlados por un *driver* (ver Fig. 1):

- **El Arduino:** es una placa de desarrollo que tiene como unidad de procesamiento un microcontrolador ATmega328. Esta placa cuenta con terminales (o *pines*) a las que podemos conectar sensores, motores, LEDs, etc. El **Arduino** cuenta también con un LED incorporado en la placa que está asociado al *pin 13*, este LED puede ser utilizado, por ejemplo, para *debuguear* el código.
- **Los motores:** sirven para mover el robot en línea recta (los dos motores con igual velocidad) o doblando en algún sentido (diferentes velocidades en cada motor).

- **El driver:** maneja la corriente que necesitan los motores. En nuestra simulación los motores seleccionados cuentan con *drivers* internos, por lo que no es necesario incluirlos en el circuito.

De los sensores con los que cuenta, utilizaremos los siguientes:

- **Sensores de línea:** sensores de salida binaria, permiten medir si el piso es negro o blanco. Funcionan a partir de emitir una luz infrarroja y observar el retorno sobre la superficie medida y decidir si es negro o blanco. **Nota:** cuando detectan un cambio de color estos sensores generan una interrupción.
- **Sensores de choque:** sensores que pueden detectar cuándo el robot colisiona contra un obstáculo. Son esencialmente pulsadores. **Nota:** este sensor se deberá leer por *polling*.

## 1.1. El simulador

Para simular el robot se utilizarán componentes similares a los del mismo en un el simulador web [www.123d.circuits.io](http://www.123d.circuits.io). Pero antes es necesario hacer una introducción a los componentes que se utilizarán.

**El protoboard** es una plataforma de pruebas que permite interconectar componentes electrónicos sin la necesidad de utilizar soldadura. Las conexiones se hacen por medio de láminas metálicas internas que ejercen presión sobre los terminales de los componentes. Algunos agujeros se encuentran conectados horizontalmente y otros verticalmente (ver Fig. 2).

**Los cables** son los elementos por los cuales circula la corriente eléctrica. Utilizaremos la siguiente convención de colores:

- rojo para los cables de 5V
- negro para los cables de 0V (masa)
- amarillo para las conexiones de entradas al Arduino
- naranja para las conexiones con los motores
- verde para conexiones internas del protoboard

**Las resistencias** sirven para limitar la corriente que circula por un circuito y así evitar que se queme un componente.

**Los botones o switches** no son más que simples interruptores. Existen dos tipos, dependiendo de si mantienen el valor una vez que se sueltan (funcionamiento similar al de una llave de luz) o no (funcionamiento similar al de un timbre).

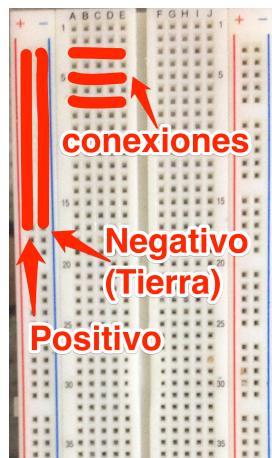


Figura 2: Protoboard

## 1.2. Antes de empezar

Para poder utilizar el simulador se debe ingresar a la página web [www.123d.circuits.io](http://www.123d.circuits.io) y crear una cuenta.

Para la resolución de este taller no será necesario hacer un nuevo circuito sino utilizaremos uno existente (el link se encuentra en el enunciado a continuación) que viene con un código de prueba. Este circuito debe ser duplicado con el botón de **Duplicate Project** para obtener una copia propia que puede ser modificada. Para acceder (y modificar) el código del Arduino, se debe hacer click en el botón **Code Editor**. En esta misma ventana es donde se puede acceder al **Serial Monitor** donde se puede observar la salida de *debug*.

Para comenzar la simulación es necesario tocar el botón **Start Simulation**. El código solo puede ser editado si la simulación está detenida.

### 1.3. Funciones importantes

El lenguaje Arduino está basado en C/C++. Referencia en [www.arduino.cc/en/Reference/HomePage](http://www.arduino.cc/en/Reference/HomePage). Las funciones relevantes para este taller son:

`unsigned long millis()` devuelve el número de milisegundos desde que el arduino comenzó a correr el programa.

`void attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)` : permite setear una interrupción, `pin` es la entrada del arduino a la cual está conectada la interrupción. `ISR` es el nombre de la función que va a atender la interrupción, `mode` es el tipo de interrupción, en este taller siempre se utilizará `CHANGE`.

`void analogWrite(pin, value)` es el mecanismo que cuenta el Arduino para tener una salida analógica, capaz de ser leída por el *driver* de los motores para su control por velocidad. Con dicha función, en *pin* se emite una señal de una intensidad *value* en formato PWM (*Pulse Width Modulation*, en español: Modulación por ancho de pulsos). La señal analógica que se emite puede variar entre el valor mínimo 0 (que corresponde a 0V) y el máximo 255 (que corresponde a 5V). En el caso del presente taller, el 0 corresponde al motor detenido y el 255 a su funcionamiento a máxima velocidad.

`void pinMode(pin, mode)` configura *pin* como entrada o salida. Los parámetros de *mode* son: INPUT, OUTPUT o INPUT\_PULLUP. **Importante:** para leer un sensor conectado a *pin* se tiene que usar INPUT\_PULLUP.

`int digitalRead(pin)` lee el valor de una entrada digital `pin`, devolviendo un 1 ó un 0 (también se cuenta con las constantes `HIGH` y `LOW` respectivamente).

`Serial.begin(speed)` configura e inicializa la comunicación serial a la velocidad dada por `speed`. En el taller utilizaremos un valor de 9600

`Serial.println("Esto es una salida de debug")` imprime texto en el puerto serial. Se puede utilizar tambien `Serial.print` que no genera un salto de linea. Esto es útil para imprimir valores de variables como:  
`Serial.print("Sensor: "); Serial.println(valor);`

## 2. Consigna

Ej. 1) a) Duplicar el circuito de referencia (fig. 3) desde la siguiente dirección: <https://circuits.io/circuits/2856000>. El circuito tiene cargado un código de ejemplo que se puede observar en el Apéndice de este enunciado. Si se presiona un pulsador o se activa un interruptor, se activará el LED correspondiente.

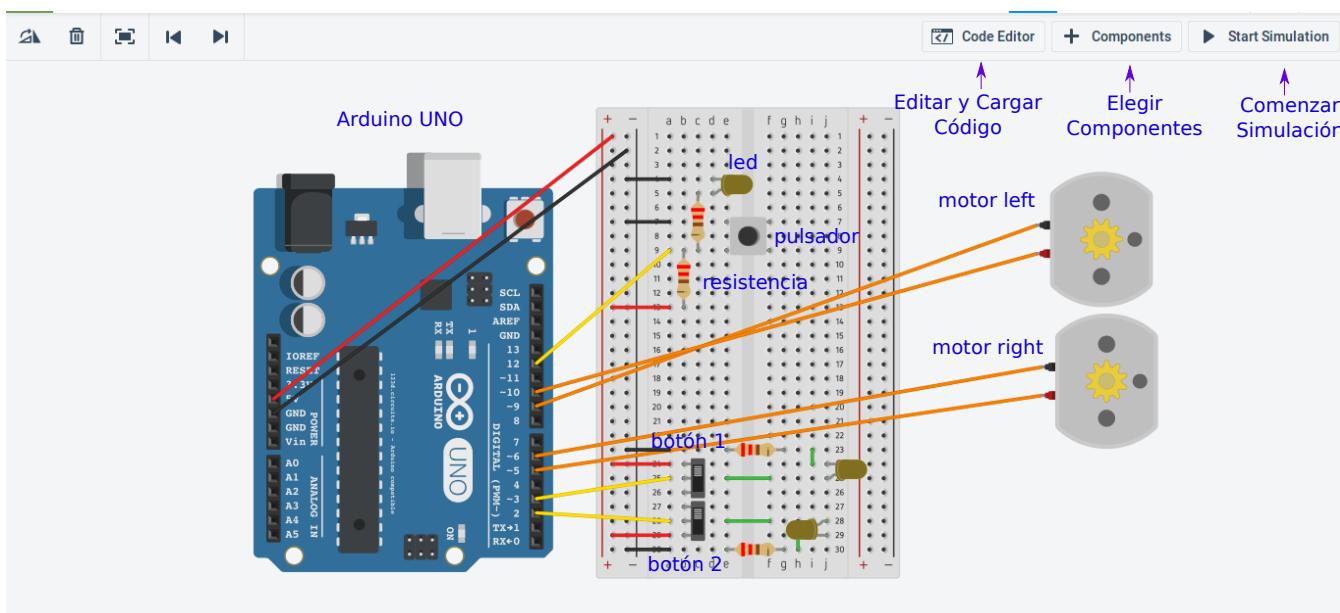


Figura 3: Circuito completo

#### Observaciones:

- el pulsador solo mantiene el valor mientras lo tenemos presionado, cuando está presionado coloca un 0 en el pin 12 y cuando no está pulsado coloca 1.
  - los botones 1 y 2 mantienen su valor una vez que los soltamos
  - el simulador no corre en tiempo real, el tiempo de simulación transcurrido se puede observar en la parte superior izquierda del simulador.

**Nota:** Los sensores de línea serán simulados mediante una llave cada uno, mientras que de los sensores de choque, trabajaremos con solo uno de ellos, el cual será simulado mediante un pulsador.

Ej. 2) Se deberán programar 2 comportamientos distintos de un robot siguiendo una línea. Para ello, utilizar el circuito anterior. Tener en cuenta que el código de ejemplo anterior solo es ilustrativo y **no** corresponde a ninguno de los comportamientos pedidos.

Los pines del Arduino deberán ser configurados como entrada o salida y teniendo en cuenta las conexiones del circuito:

pin	Conexión
2	Sensor de línea izquierdo
3	Sensor de línea derecho
5	Potencia motor derecho
6	Masa motor derecho
9	Masa motor izquierdo
10	Potencia motor izquierdo
12	Sensor de choque

Los *pines* que van a masa de un motor deben *setearse* inicialmente en cero y no modificarse.

**Importante:** El control de los motores se debe realizar desde la rutina principal y no desde una rutina de atención de una interrupción.

**Comportamiento 1** Utilizando el sensor de línea izquierdo y el sensor de choque, programar el **Arduino** para lograr el siguiente comportamiento:

- El robot deberá alternar entre las dos configuraciones de los motores: **Girando izquierda** y **Girando derecha** para poder seguir la línea. Se inicia en **Girando derecha** y se mantiene hasta que el sensor detecte un cambio (y produzca una interrupción). En ese momento debe cambiar a **Girando izquierda** hasta la próxima interrupción. Y así sucesivamente.
- El pulsador debe comportarse como interruptor ON/OFF, es decir, cuando el pulsador esté apretado el robot debe detenerse (configurado como **Detenido**) y cuando se suelta el robot debe seguir en el estado que estaba (siguiendo la línea).

A continuación se muestra la máquina de estados que reflejan el comportamiento esperado (ver Fig. 4):

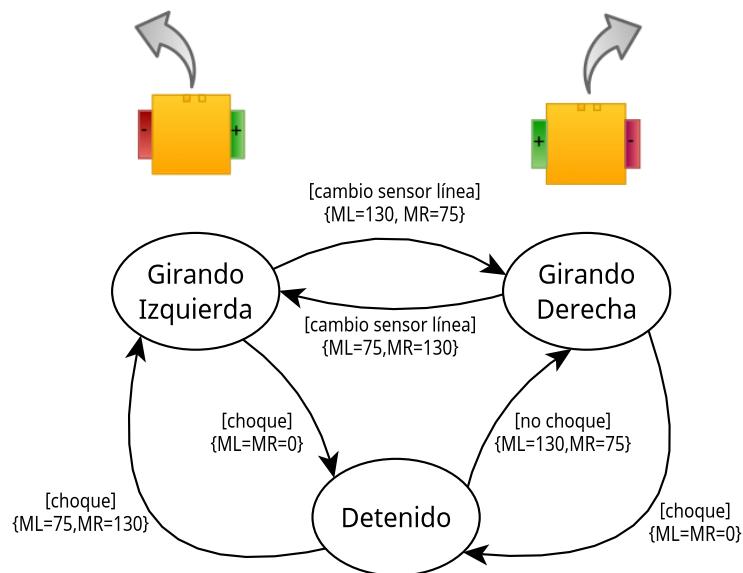


Figura 4: Comportamiento 1

**Comportamiento 2** Utilizando los dos sensores de línea y el sensor de choque programar el **Arduino** para lograr el siguiente comportamiento:

- El robot arrancará configurado en **Avanzando**. Cuando detecte que se desvía de la línea deberá corregir su rumbo. Para ello, si el sensor izquierdo empieza a sensar negro (lo cual produce una interrupción), se debe modificar a **girando izquierda** hasta que el mismo sensor vuelve a sensar blanco (produciendo nuevamente una interrupción). Lo mismo sucede si el sensor que provoca una interrupción es el sensor derecho.

- el pulsador debe comportarse como una pausa de 3 segundos, es decir, que cuando el pulsador es activado el robot configurarse como **Detenido** por 3 segundos y luego continuar con su comportamiento anterior. No se puede utilizar la función `delay()`, pero para saber el paso del tiempo se puede utilizar la función `millis()`.

A continuación se muestra las dos máquinas de estados (ver Fig. 5) que reflejan el comportamiento esperado.

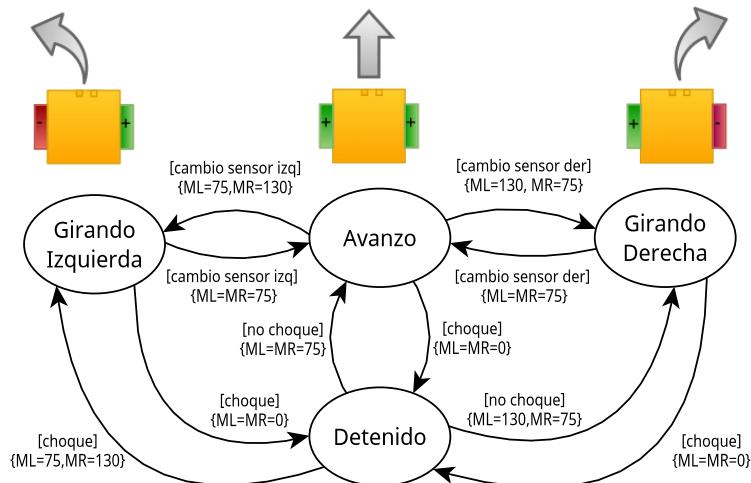


Figura 5: Comportamiento 2

**Comportamiento 3 (Optativo)** Modificar el comportamiento 2 para lograr que en las rectas el robot aumente su velocidad. Para lograr esto establezca límites de velocidad máximos y mínimos, tomando como mínimo el valor actual en el estado **Avanzando** (potencia 75 en cada motor). Determine una constante mediante la cual el robot aumente su velocidad mientras se mantenga en el estado **Avanzando**. Cuando el robot cambie de estado, o se encuentre en otro estado que no sea **Avanzando**, deberá repetir el comportamiento 2.

**Opcional:** para evaluar cada comportamiento en el robot, deberán crear un archivo en blanco, cuyo nombre deben ser los apellidos de los alumnos, seguido por el número de comportamiento y con extensión (.ino), por ejemplo (PerezGomez1.ino). En estos archivos deberá copiar el código del simulador. Pasar dichos archivos a un pendrive y entregarlo a algún docente.

Debido a que el robot utiliza un **ArduinoMega** y la simulación un **ArduinoUno**, se deberán realizar los siguientes cambios en la asignación de pines para su ejecución en el robot:

pin	Conexión
4	Potencia motor derecho
5	Masa motor derecho
7	Masa motor izquierdo
6	Potencia motor izquierdo
14	Sensor de choque

### 3. Anexo

#### 3.1. Código de ejemplo

```
1 /* Constantes que no cambian y son usadas para setear el numero de los pines */
2 #define motorPinA 5           // el numero del pin de uno de los cables de motor
3 #define motorPinB 6           // el numero del pin del otro cable del motor
4 #define sensorInt 3          // el numero del pin del sensor que interrumpe
5 #define sensorPoll 2          // el numero del pin del sensor sobre el cual se hace polling
6
7
8 /* variables que cambian su valor: */
9 bool andando = true;
10 bool nitro = false;
11
12 /* setup inicial*/
13 void setup() {
14     pinMode(sensorPoll, INPUT_PULLUP); // inicializo el pin como una entrada (recordar usar
15     // INPUT_PULLUP)
16     pinMode(motorPinA, OUTPUT);      // inicializo el pin A del motor como una salida
17     digitalWrite(motorPinA,0);       // pongo en 0 el pin A del motor
18     pinMode(motorPinB, OUTPUT);      // inicializo el pin B del motor como una salida
19     Serial.begin(9600);            // configuro e inicializo la comunicacion serial a 9600
20     // baudios
21     // configuro la interrupcion del pin sensorInt para que llame a la funcion meInterrumpieron
22     // cuando se produce un cambio
23     attachInterrupt(digitalPinToInterrupt(sensorInt), meInterrumpieron, CHANGE);
24 }
25
26 /* loop del programa */
27 void loop(){
28     if(andando){                // Pregunto si estoy en estado andando
29         if (digitalRead(sensorPoll)){ // Pregunto si el sensorPoll esta apretado
30             nitro = true;           // Si estoy en estos estados seteo el estado nitro
31         }
32         else{                     // Si el sensorPoll no esta apretado
33             nitro = false;          // pongo nitro como false
34         }
35     actualizarEstado();          // llamo a la funcion que manda potencia a los motores
36     // dependiendo del estado definido
37 }
38
39 /* funcion que actualiza la velocidad de los motores dependiendo del estado*/
40 void actualizarEstado(){
41     if(andando){                // Si el estado es andando
42         if(nitro){               // y nitro es verdadero
43             digitalWrite(motorPinB,250); // Seteo la velocidad de el motor a 250
44             Serial.println("estado: andando nitro"); // mando por serial el estado
45         }
46         else{                   // Si nitro == 0 entonces pongo velocidad de 30
47             digitalWrite(motorPinB,30); // a los motores
48             Serial.println("estado: andando sin nitro"); // mando por serial el estado
49         }
50     }
51     else{                      // Si andando no es verdadero
52         digitalWrite(motorPinB,0); // freno el motor poniendo velocidad 0
53         Serial.println("estado: detenido"); // mando por serial el estado
54     }
55 }
56
57 /* funcion que se llama cuando se produce una interrupcion */
58 void meInterrumpieron(){
59     if(andando){                // si estoy andando
60         andando=false;          // entonces freno
61     }
62     else{                      // Si estoy frenado
63         andando = true;         // comienzo a andar
64     }
65 }
```