Tutorial_03.- Arduino: Entradas y salidas analógicas

Contenido

Material necesario

- Placa Arduino
- Cable micro USB
- Placa Protoboard
- Cables Jumper-Wire
- Componentes:
  - LEDs
  - Resistencias
  - Pulsadores
  - Potenciómetro

Conceptos teóricos

- Corriente y tensión directa de un diodo LED
- Cálculo del valor de una resistencia limitadora (Ley de Ohm)
- Resistencia de Pull-up y Pull-Down
- Filtrado de rebotes de un pulsador
- Funcionamiento de un potenciómetro
- Divisor de tensión
- Convertidor analógico digital (ADC)
- Modulación de ancho de pulso (PWM)

Estructuras de programación en Arduino (C/C++)

- Funciones setup() y loop() de Arduino
- Condicional *If*
- Bucle *For*

Funciones propias de Arduino

- pinMode()
- digitalWrite()
- digitalRead()
- analogRead()
- analogWrite()
- Serial.begin()
- Serial.println()
- map()

IMPORTANTE:

Comprobaremos que nuestra placa **Arduino** está **desconectada** y sin energía, puesto de no ser así podría dañarse tanto la placa, como el equipo. Una vez hemos realizado esta comprobación, pasaremos a realizar el montaje.

1. Entradas analógicas

Contenido teórico

Entradas analógicas

En Arduino se denominan entradas analógicas a aquellos pines del microcontrolador conectados a cualquiera de los canales del convertidor analógico-digital (ADC) del que dispone.

Estas entradas son usadas habitualmente para conectar sensores cuya lectura será transformada a un valor digital gracias al ADC.

Convertidor analógico-digital (ADC)

Un convertidor A/D es un circuito que lee un valor analógico de tensión y devuelve un valor digital con un valor binario.

Estos convertidores cuentan con dos señales de entrada llamadas Vref+ y Vref- que determinan el rango en el cual se convertirá una señal de entrada. En la placa Arduino Vref-=0 y Vref+=5V (por defecto).

El convertidor A/D establece una relación entre su entrada (señal analógica) y su salida (digital) dependiendo de su resolución. Esta resolución se puede obtener a partir del valor máximo de tensión a la entrada del ADC y el número de bits con que cuenta.

A manera de ejemplo, el convertidor análogico incluido en el microcontrolador de la placa Arduino Leonardo es de 10 bits y tiene la capacidad de convertir una muestra analógica de entre 0 y hasta 5 voltios:

```
Resolución = valor analógico / (2^(nºbits)-1)
Resolución = 5 V / 1023 →
     1024 valores representables, siendo el máximo valor 1023
Resolución = 0.00488 o 4.88mV
```

Lo anterior quiere decir que por cada 4.88 milivoltios que aumente el nivel de tensión en la entrada al A/D, la salida de éste aumentará en una unidad (siempre sumando en forma binaria bit a bit). Por ejemplo:

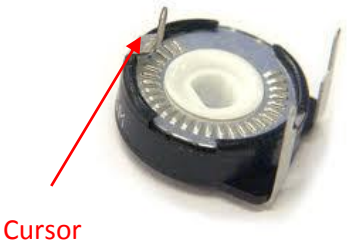
```
Entrada - Salida

0 V          - 00 0000 0000    (0 decimal)

0.00488 V    - 00 0000 0001

0.00976 V    - 00 0000 0010

…

5 V          - 11 1111 1111    (1023 decimal)
```

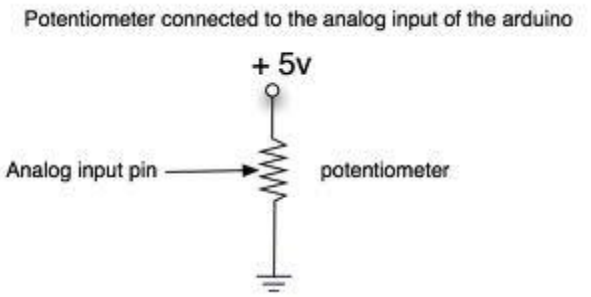
Potenciómetro

Un potenciómetro es un elemento pasivo con un eje que proporciona una resistencia variable. Cuenta con tres terminales, uno conectado a tierra, otro a la tensión Vcc y un tercero que es el cursor (pin central)

En Arduino conectaremos el cursor del potenciómetro a la entrada analógica A0.

Al girar el eje del potenciómetro, cambiamos el valor de la resistencia entre los pines de los extremos y el pin central del potenciómetro. Esto cambia la "cercanía" relativa del pin central a 5 Voltios y a masa, ofreciendo diferentes valores analógicos de entrada (tensión).





Terminal serie

Todos los microcontroladores de las placas Arduino tienen al menos un puerto serie (también conocido como UART o USART). El puerto serie permite la comunicación entre la placa Arduino y un ordenador u otros dispositivos (Raspberry, p.e.).

El microcontrolador de Arduino Leonardo dispone de dos puertos serie:

- "Serial": correspondiente al puerto serie USB.
- "Serial1": correspondiente al puerto serie localizado en los pines 0 (RX1) y 1 (TX1).
-

El microcontrolador de Arduino Uno solo dispone de un puerto serie:

- "Serial": correspondiente al puerto serie USB y también a los pines 0 (RX0) y 1 (TX1).

En el apartado referencia de la web de Arduino puedes ver que existen varias funciones para trabajar con el puerto serie.

El entorno de trabajo de Arduino integra un "Monitor Serial", disponible en las herramientas, mediante el cual se monitorizan los valores recibidos por el puerto serie. Es muy importante que la velocidad de transmisión establecida en el "Monitor Serial" coincida con la configurada para el puerto serie de Arduino. Por defecto, este valor es de 9.600 baudios para el "Monitor Serial".

Proyecto03_01: Control del Encendido/Apagado de un LED con un potenciómetro. Visualización en el terminal serie.

En este proyecto vamos a aprender utilizar las entradas analógicas de nuestro Arduino. Conectaremos un potenciómetro a la entrada analógica A0 y mostraremos el valor de tensión leído mediante el terminal serie incluida en el IDE de Arduino. Además, si el valor leído en el potenciómetro supera a 500, lucirá un LED.

Material

- 1 Potenciómetro
- 1 diodo LED
- 1 resistencia limitadora para el LED

Funciones de arduino

analogRead(pinSensor)

http://arduino.cc/en/Reference/AnalogRead

Esta instrucción nos permite leer cualquier sensor que conectemos al pin analógico establecido. Por defecto, Arduino realizará una conversión analógico-digital para toda señal de 0v a 5v con una resolución de 10 bits, lo que nos da 2^{10} (1024) valores para nuestro programa, correspondiendo 0 a 0v y 1023 a +5v.

Serial.begin(<velocidad>)

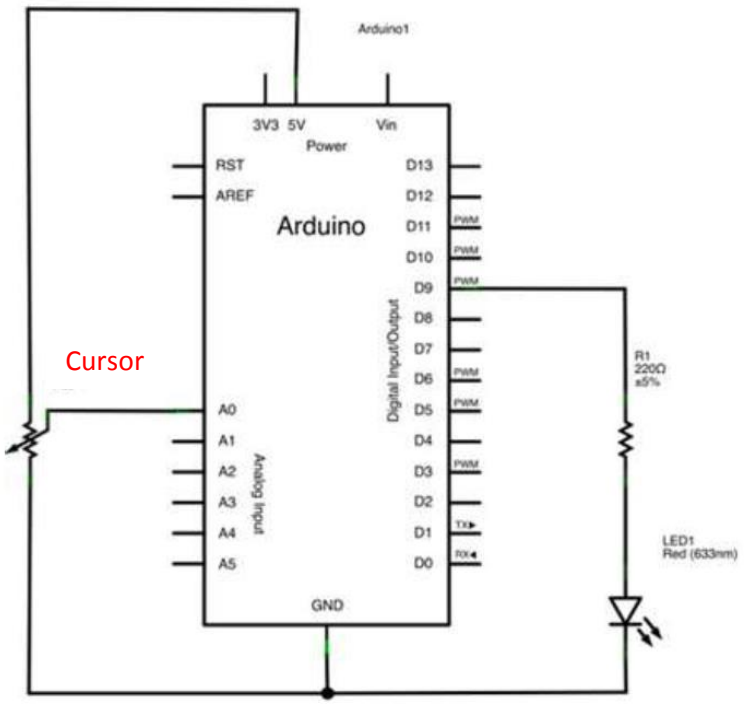
Inicia la comunicación serie con el microcontrolador a la velocidad establecida. Esta velocidad debe ser la misma que esté establecida en el Terminal Serie

Serial.println (<valor>)

Imprime el valor especificada en la terminal serie del IDE y a continuación un retorno de carro.

http://arduino.cc/en/Reference/Serial

Esquema



Código

```
const int LedPin= 9;
int   AnalogValue=0;

void setup() {
  pinMode(LedPin,OUTPUT);

  //Open and set the baudrate for the Serial port (9600 baudios)
  Serial.begin(9600);
}

void loop() {
  //Read the potentiometer
  AnalogValue=analogRead(A0);

  //Show the read data in the console
  Serial.println(AnalogValue);

  //Check value to switch on or off the LED
  if (AnalogValue>=500)
    {
     digitalWrite(LedPin,HIGH);
    }
  else
    {
     digitalWrite(LedPin,LOW);
    }
  delay(200);
}
```

2. Salidas analógicas

Contenido teórico

Modulación de ancho de pulso (PWM)

PWM (http://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos)

Técnica en la que se modifica el ciclo de trabajo de una señal periódica (senoidal o cuadrada), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga (p.e. un motor).

Se denomina **ciclo de trabajo (δ)** de una señal periódica a la relación entre el tiempo que dura la parte positiva de la señal y su período. Expresado matemáticamente:

$$D = \frac{\tau}{T}$$

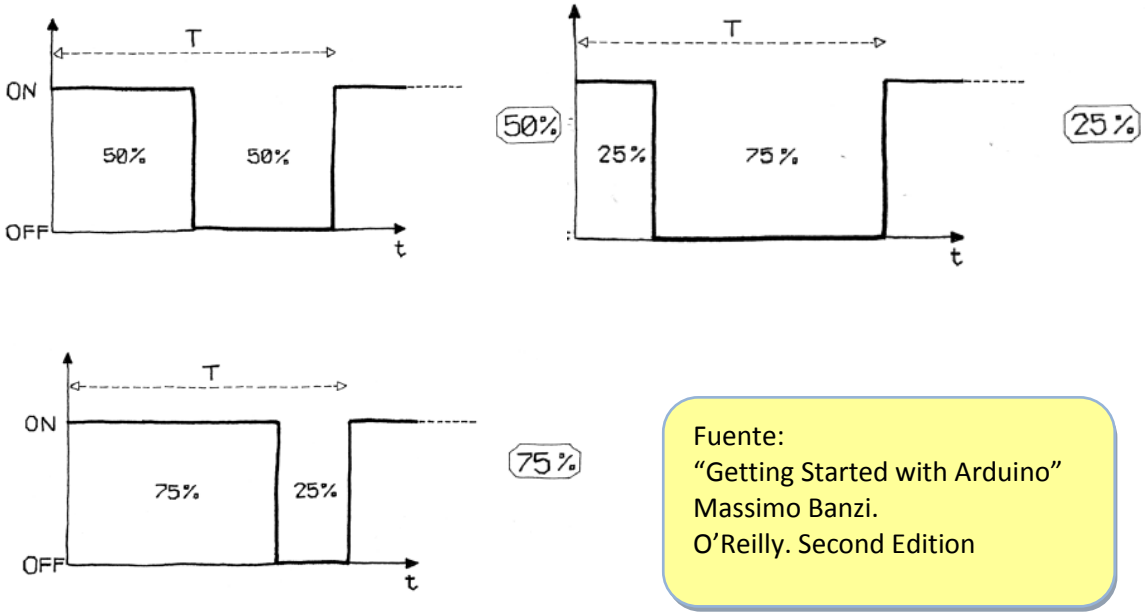
D es el ciclo de trabajo

τ es el tiempo en que la función es positiva (ancho del pulso)

T es el período de la función

Aplicaciones del PWM:

- Variar la intensidad de brillo de un LED
- Variar la velocidad motor DC manteniendo su par (fuerza) constante.
- Variar la posición en grados de un servomotor.
- Etc.



Fuente:
"Getting Started with Arduino"
Massimo Banzi.
O'Reilly. Second Edition

Arduino Uno cuenta con tres salidas conectadas a los pines del microcontrolador ATMEL que proporciona una función de PWM. Esas salidas están serigrafiadas en la placa con el símbolo ~.

Para obtener una salida con PWM está disponible la función analogWrite() que más abajo se detalla.

Proyecto03_02: Control de la intensidad luminosa de un LED. Modulación PWM.

En este proyecto, el diodo LED variará su intensidad lumínica en función de un valor que inicialmente se incrementa y posteriormente se decrementa, es el llamado ***efecto fading*** *(desvanecimiento). Puede observarse que se ha elegido la salida número 9 compatible con la opción PWM.*

Estructura de programación

for

Este código incluye una estructura de bucle.

http://arduino.cc/en/Reference/for#.UyFymD9WLzE

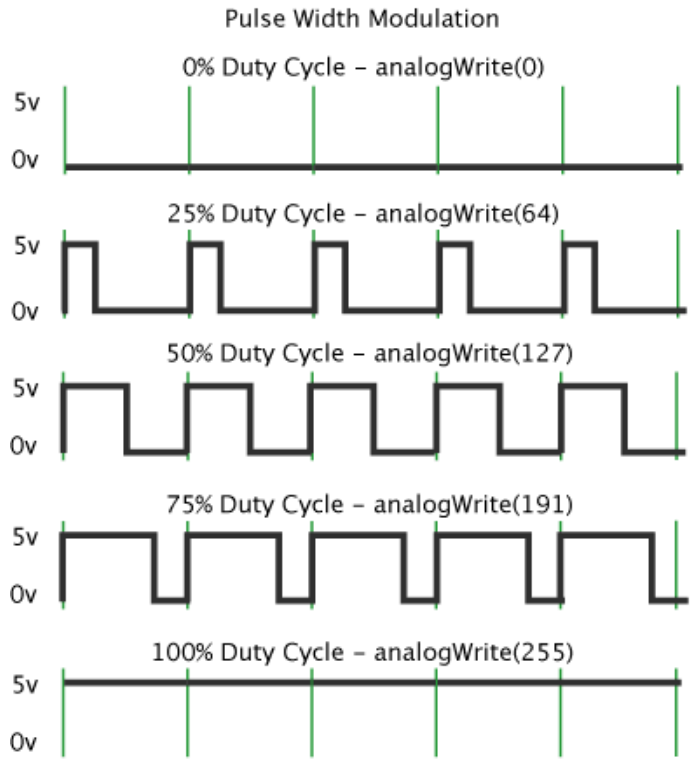
Funciones Arduino

analogWrite(pin,valor)

Después de llamar a la función **analogWrite()**, el pin generará una onda cuadrada estable con el ciclo de trabajo especificado. La frecuencia de la señal PWM será de aproximadamente 500 Hz.

- `pin`: Es el pin en el cual se quiere generar la señal PWM.
- `valor`: El ciclo de trabajo deseado comprendido entre 0 (siempre apagado) y 255 (siempre encendido).

http://arduino.cc/en/Reference/AnalogWrite



Observe las formas de ondas de la figura, la tensión promedio que proporciona cada una está relacionada con el ciclo de trabajo. Así, con un ciclo de trabajo del 25% tendremos una tensión promedio de 1.25V, con 50%, 2.5V y así con el resto de valores del ciclo de trabajo.

Al aplicar una señal de PWM a un diodo LED, por ejemplo, en función del ciclo de trabajo establecido estaremos aplicando una mayor o menor tensión a éste, y por tanto su luminosidad será mayor o menor.

Código

```
// Fade an LED in and out

const int LedPin = 11;  // LED
int i = 0;              // To count up and down
void setup() {
pinMode(LedPin, OUTPUT);      // setup LedPin as an output
}

void loop(){
for (i = 0; i < 255; i++) {    // loop from 0 to 254 (fade in)
analogWrite(LedPin, i);         // set the LED brightness
delay(10);                      // Wait 10ms because analogWrite
                                // is instantaneous and we would
                                // not see any change
}
for (i = 255; i > 0; i--) {   // loop from 255 to 1 (fade out)
analogWrite(LedPin, i);             // set the LED brightness
delay(10);                    // Wait 10ms
}
}
```

Ejercicios

1. Construye un circuito y el código correspondiente que permita al usuario regular la intensidad luminosa de un LED mediante un potenciómetro. El valor introducido mediante el potenciómetro y el enviado al PWM deberán mostrarse en el terminal serie.

NOTA: Deberás usar PWM y la función de Arduino "map()" http://arduino.cc/es/Reference/map