UNIT 5: LAS SEÑALES ANALÓGICAS

OBJETIVOS

Hasta ahora hemos considerado que todas las señales que maneja el Arduino son señales de carácter digital compuestas de niveles “1” o “0”. Las señales de entrada pueden venir de pulsadores, interruptores, detectores, etc…, pero siempre introducen un “1” o un “0”. Las señales de salida pueden activar o no a leds, relés, motores, etc…, pero siempre a base de aplicarles un “1” o un “0”.

Sin embargo en la naturaleza no todo tiene un carácter digital. Existe también lo que se llaman señales “analógicas”. Señales cuyo valor o tensión puede variar entre un mínimo y un máximo a lo largo del tiempo.

El controlador Arduino, como la mayor parte de los controladores modernos, es capaz de generar unas señales digitales moduladas en anchura. Reciben el nombre de señales “PWM” (“Pulse Width Modulation”) o, lo que es lo mismo, “Modulación de Anchura de Pulso”.

En esta unidad vas a aprender qué son, cómo se obtienen y para qué se pueden emplear. Te adelanto que están bastante relacionadas con la regulación y control de potencia.

TEORÍA

- INTRODUCCIÓN
- CONVERSIÓN DIGITAL
- RESOLUCIÓN
  - Ahora tú
- FUNCIONES DEL LENGUAJE ARDUINO
- PERIFÉRICOS ANALÓGICOS
  - Potenciómetros
  - Sensor de luz visible
  - Sensor IR de reflexión
  - Sensor de temperatura
- SALIDAS PSEUDO ANALÓGICAS

- Que son las señales PWM?
- Para qué sirven?
- Como se generan?
  - Función: analogWrite()

- OTRAS FUNCIONES DEL LENGUAJE ARDUINO
  - Función random()
  - Función randomSeed()

PRÁCTICA

- EJEMPLO 1: Conversión ADC
- EJEMPLO 2: Umbrales
- EJEMPLO 3: Comparador analógico
- EJEMPLO 4: Control de brillo
- EJEMPLO 5: Timón electrónico
- EJEMPLO 6: Fotómetro
- EJEMPLO 7: Control de alumbrado
- EJEMPLO 8: Measuring reflections
- EJEMPLO 9: Detectando colores
- EJEMPLO 10: Temperatura
- EJEMPLO 11: Climatizador
- EJEMPLO 12: Señal PWM
- EJEMPLO 13: Efectos ópticos
- EJEMPLO 14: Regulación manual
- EJEMPLO 15: Luces aleatorias

PRACTICE MATERIALS

- *Ordenador portátil o de sobremesa.*
- *Entorno de trabajo Arduino IDE que se incluye en el material complementario y que se supone instalado y configurado.*
- *Tarjeta controladora Arduino UNO*
- *Cable USB.*

TABLE OF CONTENTS

TEORÍA

1. INTRODUCCIÓN

Quizá a lo largo del curso, en alguna ocasión, haya salido la expresión de “señales analógicas de entrada”. Pero… ¿qué son?, ¿para qué sirven?, ¿dónde están?, ¿cómo se manejan? Es el momento de responder a estas preguntas. Ya sabes que en el mundo “digital” todo se contempla como si únicamente hubiera dos posibles valores o niveles: el nivel “1” y el nivel “0”. Un pulsador, un interruptor o un detector puede estar activado (a “1”) o no (a “0”). Un led, un relé o un motor se puede encender o apagar. El sonido no es ni más ni menos que una señal que transita de nivel “1” a nivel “0” a una determinada velocidad o frecuencia. La señal

PWM es una señal digital en la que la duración del nivel “1” o ciclo útil podemos variar y regular así la potencia aplicada. La comunicación serie no es más que la transferencias de bits con niveles “1” y “0”.

Cuando tenemos varios bits los agrupamos en bytes. Con ellos creamos números de diferentes tamaños, codificamos caracteres y formamos mensajes. En fin, que hasta ahora siempre te has movido en estos parámetros.

Sin embargo el mundo “real” es diferente. En la naturaleza nos podemos encontrar con magnitudes que pueden tener múltiples valores o matices. En definitiva, tú mejor que nadie sabes que no todo es blanco o negro, también hay grises.

Piensa por ejemplo en la temperatura ambiente. Se trata de una magnitud física que está variando constantemente. No hace la misma temperatura de madrugada, que al mediodía o que a la noche. Si tuviéramos un sensor capaz de medir la temperatura y ofrecernos una tensión proporcional a la misma, veríamos que esa tensión estaría variando constantemente en el tiempo, más o menos como se muestra en la figura.

La temperatura es una magnitud analógica. Gracias al sensor obtenemos una tensión de 2 V a las 5 h de la mañana. A las 9 h la temperatura aumenta y también lo hace por tanto la tensión, que sube a 4 V. A las 15 h la tension alcanza los 5 V y, a partir de las 16 h, la tensión va disminuyendo ya que la temperatura también lo hace. Únicamente nos hace falta buscar una relación entre la temperatura y la tensión que nos da el sensor.

Piensa ahora en la gran cantidad de magnitudes físicas analógicas que te rodean. Mediante los sensores o “transductores” apropiados podrás convertir esas magnitudes en sus tensiones analógicas equivalentes:

- ✓ Humedad y/o humedad relativa. Nos permite conocer la cantidad de vapor de agua que hay en la atmósfera.
- ✓ Presión atmosférica. Con el sensor apropiado podemos conocer la presión que ejerce el aire sobre la tierra.
- ✓ Peso. Podemos medir la fuerza con la que un cuerpo actúa sobre un punto en reposo.

- ✓ Velocidad. Si dispusieras del sensor adecuado podrías conocer la velocidad a la que se desplaza el aire, un vehículo o tú mismo.
- ✓ Luz. Se puede medir la luz ambiente o la luz que incide sobre un objeto. Hay sensores que detectan luz visible, luz infra roja, ultravioleta, etc…
- ✓ Sonido. Con los correspondientes sensores se puede medir, conocer y/o detectar ruidos, sonido ambiente, volumen, etc…

2. CONVERSIÓN DIGITAL

Te puedes imaginar que hoy en día dispones en el mercado de todo tipo de sensores y "transductores" capaces de medir y ofrecer una tensión analógica equivalente a la magnitud física que están midiendo. Sin embargo, ningún Sistema digital es capaz de manipular ni procesar directamente esas tensiones analógicas. Por desgracia el Arduino tampoco.

Primero es imprescindible convertir las tensiones analógicas en sus equivalents valores digitales o binarios. Para ello se emplean ciertos circuitos electrónicos llamados "Convertidores Analógicos Digitales", o bien "Analog Digital Converters". Abreviadamente "ADC".

Fijate en la Figure 1. Muestra los components necesarios para convertir la información, en este caso la temperature.

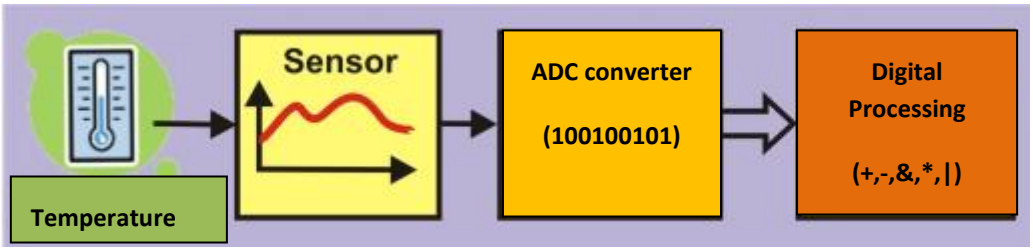

Figure 1

1. El sensor, en este caso de temperatura, capta la magnitud física a medir y para la que está diseñado. Lo más frecuente es que a su salida ofrezca una tension proporcional a esa magnitud, por ejemplo la temperatura.
2. Esta tensión se introduce al convertidor analógico/digital (ADC). Este circuito genera a su salida un valor binario equivalente a esa tensión de entrada.
3. El valor binario ahora puede ser leído por un controlador como puede ser el Arduino.
4. Dicho controlador puede realizar cualquier tipo de procesamiento con ese valor binario: registrarlo en memoria, realizar operaciones aritmético/lógicas, visualizarlo, transferirlo a otro sistema, etc… Just about all modern controllers including Arduino have an integrated ADC converter circuit. All you have to do is connect up the appropriate sensor or transducer to the analog input pin. The type of sensor or transducer you use will depend on the physical quantity you're going to measure.

Es más, es muy frecuente que esos controladores tengan varias patillas de entrada para señales analógicas. Observa la figura. Lo normal es que únicamente tengan un circuito convertidor ADC al que se le conectan diferentes patillas de entrada o “canales analógicos”. Arduino UNO tiene un único convertidor y seis patillas de entradas analógicas asociadas a él: A0-A5. Esto te permite tomar muestras de tensiones analógicas procedentes de hasta 6 sensores diferentes.

Naturalmente no puedes tomar varias muestras al mismo tiempo. Mediante las funciones apropiadas podrás hacer la conversión de un único canal analógico de entrada cada vez. Se dice que los canales de entrada están “multiplexados”.

Fijate en la Figure 2. Cada vez que ordenas realizar una conversión, el convertidor ADC toma una muestra de la tensión analógica presente en ese instante en el canal de entrada indicado. El resultado o el equivalente binario se obtiene al de unos 100 µS (0.0001”), que es el tiempo que se tarda en hacer la conversión. Aproximadamente Arduino puede hacer unas 10000 conversiones por segundo.

En el instante 1 la tensión analógica es de unos 2 V. El convertidor ADC genera un valor o número binario equivalente a esa tensión. En el instante 2 la tensión analógica es de unos 2.8 V, en el instante 3 es de 3.5 V, en el 4 de 4.2 V, y así sucesivamente. Ya ves cómo es una señal analógica. Fluctúa constantemente entre un mínimo (0 V en el ejemplo) y un máximo (5 V).

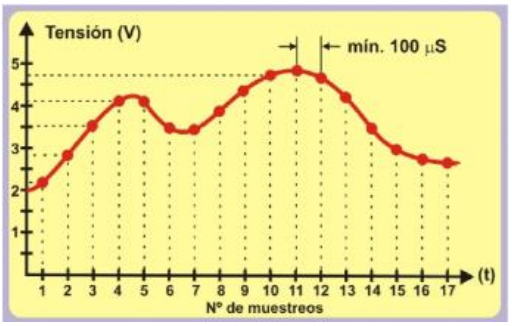

Figure 2

Ya he comentado que el tiempo que tarda Arduino UNO en hacer una conversión es de aproximadamente de 100 µS. Es bastante rápido, aunque los hay que son mucho más. Mira la figura. Representa una señal analógica con una frecuencia F de 1000 Hz. Su periodo T es de 1000 µS (1 mS). Esto quiere decir que el Arduino sería capaz de tomar 10 muestras de esa señal como máximo (1000 µS / 100 µS).

Si tuvieras una señal analógica cuya frecuencia F es de 500 Hz y su periodo T de 2000 µS, podrías tomar un total de 20 muestras de esa señal (2000 µS / 100 µS). Es decir, tendrías una “digitalización” más precisa que cuando la frecuencia de la señal analógica era de 1000 Hz.(Figure 3)

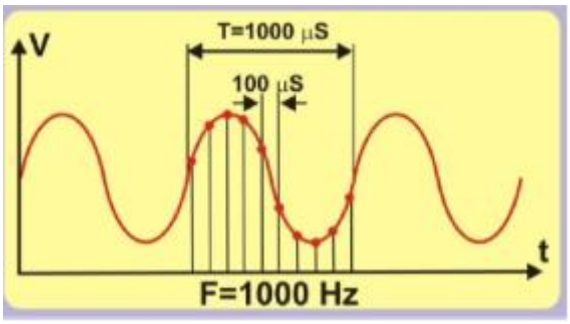

Figure 3

No siempre es necesario tomar tantas muestras por segundo. Piensa nuevamente en el sensor que mide la temperatura ambiente. Esta es una magnitud que varía muy lentamente. La temperatura no pasa de 10ºC a +15ºC en 0.0001 segundos. Lo mismo pasa, por ejemplo, con la luz natural. No pasamos de la noche al día en 100 µS. O con el peso. Nadie pesa 75 kg en un instante y 100 µS después su peso es de 31 Kg. Puedes seguir el mismo razonamiento con otras magnitudes: velocidad, presión atmosférica, humedad, etc…

¡¡ La velocidad de conversión de Arduino UNO es más que suficiente para medir la mayor parte de magnitudes físicas analógicas !!

3. RESOLUCIÓN

Además de la velocidad de conversión, otro factor que debes tener en cuenta en un circuito convertidor ADC es su precisión o "resolución". En el caso del convertidor integrado en Arduino UNO se dice que tiene una resolución de 10 bits. Esto es, el resultado que ofrece tras una conversión, puede tener 1024 valores binarios posibles (2^{10}).

¿Cómo establecer una relación entre la tensión analógica y el valor binario? Para ello necesitas conocer una constante llamada "Tensión de referencia" (V_{REF}). Es la tensión que emplea el circuito convertidor para hacer sus operaciones internas. La resolución por bit se calcula según la siguiente ecuación, y depende tanto de la V_{REF} como del número de bits del convertidor. Suponiendo una V_{REF} = 5V:

$$Resolución = \frac{V_{REF}}{2^{10}} = \frac{5}{1024} = 0.0048V/Bit \cong 0.005V = 5mV$$

INPUT VOLTAGE	OUTPUT			OUTPUT VOLTAGE	OUTPUT		
	BINARY	DEC.	HEX.		BINARIO	DEC.	HEX.
0.000	0000000000	0	0x000	**2.500**	1000000000	512	0x200
0.005	0000000001	1	0x001	**3.000**	1001100110	614	0x266

0.010	0000000010	2	0x002	**4.000**	1100110011	819	0x333
0.015	0000000011	3	0x003	**4.985**	1111111100	1020	0x3FC
0.020	0000000100	4	0x004	**4.990**	1111111101	1021	0x3FD
1.000	0011001100	204	0x0CC	**4.995**	1111111110	1022	0x3FE
2.000	0110011000	408	0x198	**5.000**	1111111111	1023	0x3FF

Te basta con dividir la tensión analógica de entrada entre la resolución por bit del convertidor, en este caso 0.0048.

- **Tu turno**

Suponiendo que la tensión de referencia VREF es de 3 V y el convertidor tiene 8 bits de resolución, completa la siguiente tabla para las diferentes tensiones analógicas de entrada.

BIT RESOLUTION							
INPUT VOLTAGE	OUTPUT			**OUTPUT VOLTAGE**	OUTPUT		
	BINARY	**DEC.**	**HEX.**		**BINARIO**	**DEC.**	**HEX.**
0.00				**1.50**			
0.01				**1.67**			
0.50				**1.85**			
0.65				**2.12**			
0.83				**2.57**			
0.95				**2.93**			
1.15				**3.00**			

.

¡¡ IMPORTANTE !! La tensión analógica de entrada que vas a medir NUNCA debe ser superior la tensión de referencia VREF. Por ejemplo, si V_{REF} = 5 V, la tensión analógica de entrada será también de 5 V como máximo.

4. FUNCTIONS IN ARDUINO LANGUAGE

El uso del convertidor ADC que integra el Arduino UNO no puede ser más sencillo. Únicamente se necesitan dos funciones del lenguaje Arduino de programación.

- **Función: analogReference()**

Permite establecer el valor de la tensión de referencia (V_{REF}) que debe emplear el circuito convertidor ADC, para realizar la conversión de una tensión analógica en su equivalente digital o binario.

Esto es importante, ya que una vez conocida podrás calcular la resolución por bit como hiciste anteriormente.

La tensión V_{REF} no debe ser mayor que la tensión general que alimenta al controlador Arduino UNO (5 V). Por otra parte, la tensión analógica de entrada que vas a convertir, no debe ser mayor que V_{REF}.

Sintaxis:

analogReference(type);

type: Selecciona el tipo de V_{REF} que va a emplear el controlador.

DEFAULT: Es la tensión de referencia por defecto. V_{REF} es la misma que la tensión con la que se alimenta el controlador: 5 V (3.3 según modelo).

INTERNAL: Es una tensión V_{REF} fija que se genera internamente en el controlador. En el caso de Arduino UNO es de 1.1 V

EXTERNAL: La tensión V_{REF} deseada se aplica por la patilla A_{REF} del controlador.

Ejemplo:

analogReference(INTERNAL); *// Se usa la tensión interna de 1.1 V*

- **Función analogRead()**

Es la sentencia que vas a emplear cuando quieras realizar una conversion analógico/digital. Cada vez que se ejecuta toma una muestra de la tensión presente en la patilla o canal analógico indicado, y realiza la conversión de la misma.**Sintaxis:**

analogRead(pin);

pin: Es el número de la patilla correspondiente al canal analógico que deseas convertir. En el caso de Arduino UNO puede ser de A0 a A5.

Ejemplo:

int V;

V = analogRead(A2); *// Realiza la conversión de la tensión presente en A2*

- **Función map()**

Aunque esta función no está pensada expresamente para la conversión ADC, nos puede venir muy bien para el caso que nos ocupa. Permite mapear, reasignar o acotar un valor comprendido entre un mínimo y un máximo, en otro.

Verás, ya sabes que el convertidor ADC de Arduino tiene una resolución de 10 bits y puede expresar un valor comprendido entre 0 y 1023 (2^{10}). Por otra parte los números con los que trabaja Arduino se empaquetan en bytes (8 bits) o múltiplos de bytes (int, unsigned int, long y unsigned long). A veces te puede interesar utilizar el resultado de una conversión (10 bits) como si fuera un byte comprendido entre 0 y 255 (8 bits) o un int (16 bits), etc… Los bytes o los int son formatos mucho más estándar.

Sintaxis:

map(value, fromLow, fromHigh, toLow, toHigh);

value: Es el valor que se desea reasignar. Normalmente es de tipo int (16 bits) o de tipo long (32 bits). No puede usarse con valores de tipo float.

fromLow: Expresa el límite mínimo del valor actual.

fromHigh: Expresa el límite máximo del valor actual.

toLow: Expresa el límite mínimo del nuevo valor.

toHigh: Expresa el límite máximo del nuevo valor.

Ejemplo:

```
int V;

V = analogRead(A2);          // Convierte la tensión presente en la patilla A2

V = map(V,0,1023,0,255);     // Reasigna el valor leído convirtiéndolo en un byte
```

El valor analógico leído desde la patilla A2 está comprendido entre 0 y 1023 y se almacena en la variable “V”. Dicho valor se reasigna en otro equivalente comprendido entre 0 y 255. Según Arduino, y por pura curiosidad, esta función responde al siguiente cálculo matemático:

$$Resultado = \frac{(valor - min) * (Nmax - Nmin)}{(max - min) + Nmin}$$

5. ANALOG PERIPHERALS

En el mercado puedes encontrar multitud de sensores capaces de proporcionar una tensión analógica, comprendida entre un mínimo y un máximo, en función de una determinada magnitud física. Los puedes considerar como periféricos analógicos.

- ✓ **Potenciómetros analógicos.** Moviendo los ejes de los mismos introduces una tensión analógica variable entre 0 y 5 V.
- ✓ **Sensor de luz.** Proporciona una tensión en función de la luz ambiente que incide sobre él.
- ✓ **Sensor IR.** Se trata de un sensor de reflexión de luz infra roja (IR) no visible. Puede medir la luz IR reflejada que incide sobre él.
- ✓ **Sensor de temperatura.** Mide la temperatura ambiente y genera una tension proporcional a la misma.

A. POTENTIOMETERS

Son resistencias variables cuyo valor puedes modificar moviendo un eje o mando llamado “cursor”. Estoy seguro que, quizá sin saberlo, los has utilizado en infinidad de ocasiones como puede ser el mando que usas para ajustar el volumen de la radio, el TV, equipo de música, etc… Son los periféricos analógicos más económicos y simples que puedes encontrar. Pueden tener diferentes formas, aspectos y tamaños. En la figura tienes los potenciómetros usados.

Dispone de tres terminales o patillas. La 1 y la 2 se corresponden con los extremos de la resistencia. Representan el valor total de la misma. La patilla 3 es el cursor. Está mecánicamente unido al eje o mando que nos permite deslizarlo por el cuerpo de la resistencia y variar así el valor de la misma. Si lo colocaras justo en el centro del recorrido, entre las patillas 1 y 3 tendrás la mitad de la resistencia. Entre las patillas 3 y 2 la otra mitad.

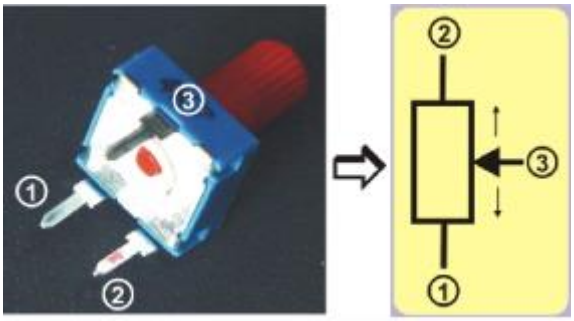

Figure 4

B. SENSOR DE LUZ VISIBLE

Se basa en un pequeño dispositivo llamado “fototransistor”. Sin entrar en tecnicismos, basta decir que se trata de un componente que deja pasar a su través más o menos intensidad eléctrica, en

función de la luz que incide sobre él. Los hay que son sensibles a la luz visible o ambiente, o bien que son sensibles a luz no visible como puede ser la infra roja (IR). Los hay de diferentes formas y tamaños.

Como se muestra en la figura, podemos emplear una linterna para que el sensor reciba más o menos luz. Con ello varía la intensidad eléctrica I que lo atraviesa. Esta intensidad I circula a través de una resistencia R, provocando una tensión V que también varía de forma directamente proporcional: V = R * I. En resumidas cuentas: al variar la luz, se varía la intensidad I y esta a su vez varía la tensión V entre 0 y 5 V.

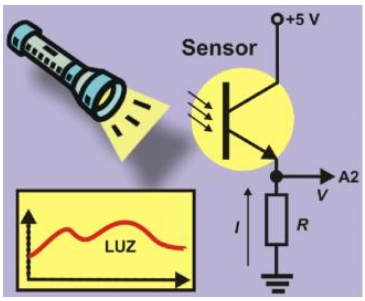

Figure 5

C. SENSOR IR DE REFLEXIÓN

Se trata de otro sensor de luz, pero de luz infra roja (IR) no visible por el ojo humano. Detecta la cantidad de luz que incide sobre él al ser reflejada por un objeto. El sensor está compuesto de dos componentes. El diodo emisor (E), alimentado con +5 V, está constantemente emitiendo luz IR. Cuando esta se refleja sobre un objeto, rebota y es recibida por el fototransistor receptor (R) de luz IR. Según la cantidad de luz reflejada, el receptor deja pasar más o menos intensidad I a través de la resistencia R, lo que provoca una mayor o menor tensión analógica V. Recuerda que: V = R * I. Resumiendo. Cuanto más próximo esté el objeto, mayor es la cantidad de luz reflejada. La intensidad I es mayor y por tanto la tensión V también.

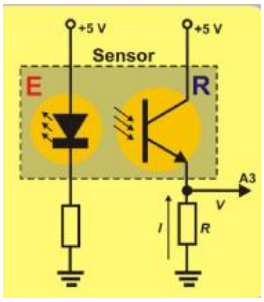

Figure 6

D. TEMPERATURE SENSOR

El conocido dispositivo LM35 Este componente tiene tres patillas. Dos de ellas se conectan con la tensión de alimentación de 0 y +5 V. Por la tercera patilla circula una intensidad I proporcional a la temperatura. Dicha intensidad atraviesa la resistencia R, lo que crea en ella una tensión V (V = R * I). Se conecta con la patilla de entrada analógica A4.

Según el fabricante de este dispositivo, la resolución del sensor es de 10 mV/ºC. Para calcular la temperatura ambiente en grados centígrados (ºC), a partir de la tensión analógica (Va), puedes emplear la siguiente ecuación:

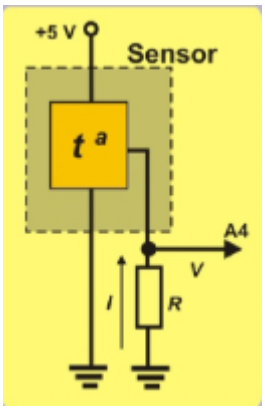

Figure 7

$$^{o}C = \frac{5 * Va * 100}{1024} = \frac{Va * 500}{1024}$$

6. PSEUDO ANALOG OUTPUT

Como ya has visto Arduino se compone de las entradas y salidas digitales, como sabrás la tarjeta Arduino UNO disponía de 14 patillas que pueden configurarse como entradas o salidas digitales. De hecho, la mayor parte de ellas, las has usado en los múltiples ejemplos que has realizado hasta el momento.

Quizá recuerdes que seis de esas patillas podían actuar también como patillas de salida de señales PWM. Por si acaso, mira la siguiente figura. Me estoy refiriendo a las patillas 3, 5, 6, 9,10 y 11, precedidas del signo “~” y resaltadas con una flecha.

A. QUE SON LAS SEÑALES PWM?

La abreviatura “PWM” viene del inglés: “Pulse Width Modulation” , que quieredecir algo así como “Modulación de Anchura del Pulso”. Se trata de una señal digital(de “1”s y “0”s), periódica (que se repite constantemente), de una determinadafrecuencia F (en un segundo se repiten X ciclos), pero “asimétrica”. Esto es, el tiempoen que la señal está a nivel “1”, puede ser totalmente diferente del tiempo en que esaseñal está a nivel "0".

Fíjate un poco en la señal de la siguiente Figure 8 y que ahora mismo vamos a estudiar. Es una señal digital con un period T de 2 mS. Es decir, en un segundo caben 500 ciclos idénticos (1000 mS/2). La frecuencia F por tanto es de 500 Hz (F=1/T).

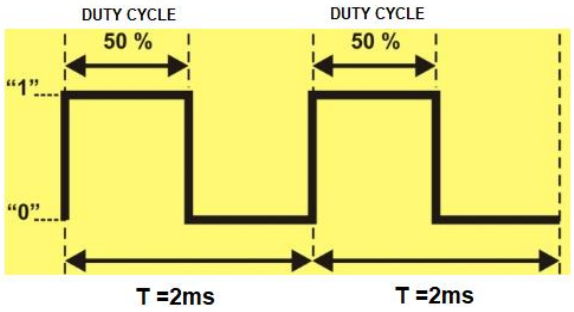

Figure 8

En este caso el nivel "1" de cada ciclo dura el mismo tiempo que el nivel "0": 1 mS cada uno (en total ambos suman los 2 mS del periodo T). Se dice que es una señal "simétrica". El tiempo en que la señal vale nivel "1" recibe el nombre de "ciclo útil". En el ejemplo su valor es del 50% (1 mS) del valor total del periodo (2 mS). Se dice que es una señal PWM al 50%.

Vamos ahora a analizar las 4 señales PWM mostradas en la figura. Todas tienen en común el mismo periodo T de 2 mS, o lo que es igual, una frecuencia F de 500 Hz. Sin embargo la duración del nivel "1", es decir, el ciclo útil varía en todas ellas.

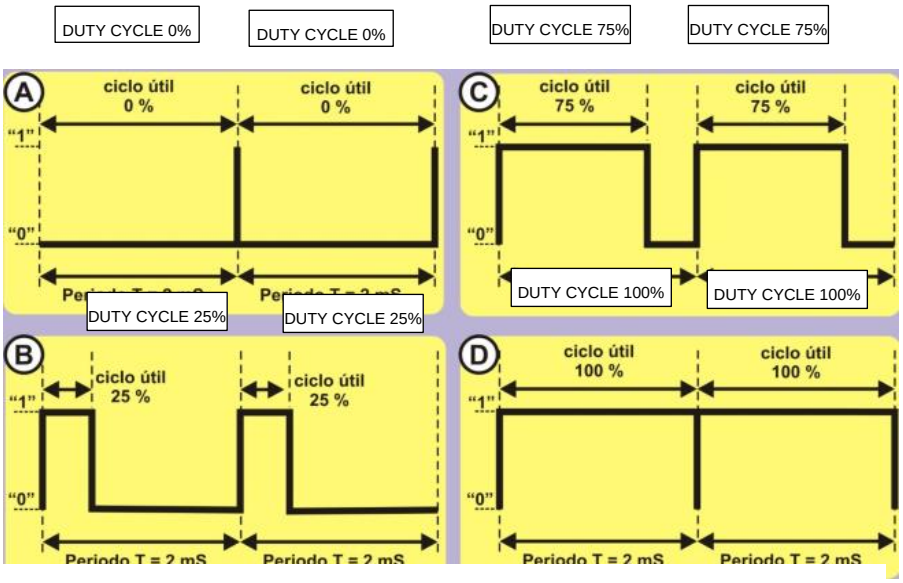

Figure 9

- ✓ **La señal A.** Es una señal PWM con un ciclo útil del 0% de la duración del periodo. Es decir, está a nivel "1" durante 0 mS (0% * 2 / 100), y a nivel "0" durante el 100% del tiempo restante, es decir, 2 mS (100% * 2 / 100).

- ✓ **La señal B.** Es una señal PWM con un ciclo útil del 25% de la duración del periodo. Es decir, está a nivel “1” durante 0.5 mS (25% * 2 / 100), y a nivel “0” durante el 75% del tiempo restante, es decir, 1.5 mS (75% * 2 / 100).

- ✓ **La señal C**. Es una señal PWM con un ciclo útil del 75%. Está a nivel “1” durante 1.5 mS (75% * 2 / 100) y a nivel “0” el 25% del tiempo restante, es decir, 0.5 mS (25% * 2 / 100).

- ✓ **La señal D**. Es una señal PWM con un ciclo útil del 100%. Está a nivel “1” durante 2 mS (100% * 2 / 100) y a nivel “0” el 0% del tiempo restante, es decir, 0 mS (0% * 2 / 100).

Ya te he comentado que la mayor parte de los controladores modernos son capaces de generar una o varias señales PWM por algunas de sus patillas. En el caso del Arduino UNO las patillas son: 3, 5, 6, 9, 10 y 11. Cuando empleas cualquiera de ellas como salidas PWM, debes conocer la frecuencia F de las señales que genera Arduino.Mira la tabla:

PIN Nº	FREQUENCY, F	PERIOD, T
5 and 6	**980 Hz**	**1.02 mS or 1020 µS**
3, 9, 10 and 11	**490 Hz**	**2.04 mS or 2040 µS**

B. ¿PARA QUE SIRVEN?

Voy a tratar de evitar tecnicismos complejos. Dado que las señales PWM permiten ajustar el tiempo en que se mantiene el nivel “1” (ciclo útil) para cada periodo, las puedes usar para controlar y regular la potencia que aplicas a ciertos periféricos de salida como pueden ser una bombilla, un led, un motor, un servo, etc…

Imagina una señal PWM como la mostrada en la figura, y que está conectada a un led. Como el ciclo útil es del 50%, la mitad del tiempo el led estará encendido y la otra mitad apagado. El led brillará a la mitad de su potencia en cada periodo, ni más ni menos.

Según este mismo razonamiento, si se le aplica un ciclo útil del 0%, la señal estará permanentemente a nivel “0”. El led no se iluminará. Si el ciclo útil es del 25% el led brillará la cuarta parte de su potencia y con una señal PWM del 75% brillará las tres cuartas partes. Con una ciclo útil

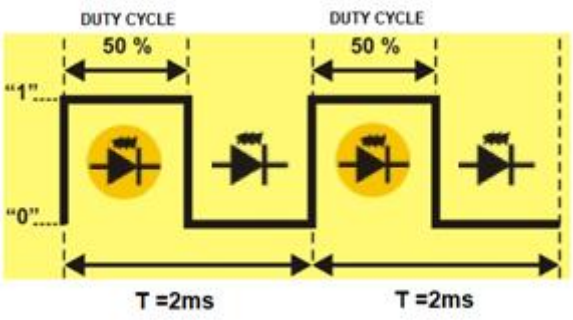

Figure 10

del 100% la señal se mantiene a nivel “1” permanentemente. El led brillará por tanto al máximo de su potencia. Tenemos así un rango que va del 0% al 100% de la señal PWM para ajustar el brillo deseado.

De la misma manera que puedes regular el brillo de un led o de una lámpara, también podrás regular la velocidad de un motor o el posicionamiento del eje de un servo. El principio es el mismo y en breve podrás experimentar con ello. Que sepas que hay un buen número de periféricos que se controlan mediante señales PWM.

C. ¿CÓMO SE GENERAN?

Los controladores que son capaces de generar señales PWM integran en su interior una serie de circuitos electrónicos más o menos complejos como osciladores, contadores, comparadores, registros, etc… Haciéndolos trabajar a todos ellos en equipo y de forma sincronizada, se consigue generar una o varias señales de este tipo. Sin embargo hay que tener bastantes conocimientos técnicos para poner todo eso en marcha.

- **Función: analogWrite()**

Permite ajustar la duración del ciclo útil de una señal PWM de salida.

Sintaxis:

analogWrite(pin, value);

pin: Hace referencia a la patilla por la cual se va a generar la señal PWM. Recuerda que en el caso de nuestro controlador Arduino UNO pueden ser: 3, 5, 6, 9, 10 y 11.

value Determina la duración del ciclo útil. Es un número de un byte comprendido entre 0 y 255. El valor 0 representa el 0% del ciclo útil, 127 representa un ciclo útil del 50% y 255 corresponde a una señal PWM con un ciclo útil del 100%.

Ejemplo:

```
byte Voltage = 25;                  // Indica el % de la potencia deseada
int Cycle= Voltage*255/100;         // Calcula la duración del ciclo entre 0 y 255
analogWrite(6,Cycle);               // Genera la señal PWM al 25% por la patilla 6
```

La patilla elegida para sacar la señal PWM no necesita ser configurada como salida. Va implícito en la propia función. La señal PWM de salida se mantiene indefinidamente presente en la patilla indicada hasta que ejecutes de Nuevo analogWrite() con un valor diferente, o bien ejecutes una función digitalRead() o digitalWrite() sobre esa misma patilla. En cualquiera de los casos la salida de la señal PWM queda anulada.

7. OTRAS FUNCIONES DEL LENGUAJE ARDUINO

Aunque son funciones que no tienen nada que ver con las señales PWM, te las voy a comentar porque así amplias tus conocimientos sobre el propio lenguaje y además te pueden dar bastante juego.

Están relacionadas con la producción de números aleatorios. Es decir, números que pueden ser de cualquier valor dentro de unos límites, números al azar.

- **The random() function**

Genera un número aleatorio entre un mínimo y un máximo. El valor que devuelve es de tipo entero de 32 bits con signo (long).

Sintaxis:

random(min, max);

min: Establece el valor mínimo (él incluido) del número aleatorio a obtener. Es opcional. Si no se indica el valor mínimo es 0.

max: Establece el valor máximo (él excluido) del número aleatorio a obtener.

Ejemplo:

// Emular una tirada de dos dados

byte Die_1;

byte Die_2;

Die_1 = random(1,7); *// Genera un número aleatorio entre 1 y 6*

Die_2 = random(1,7); *// Genera un número aleatorio entre 1 y 6*

- **The randomSeed() function**

Es sistema que emplea Arduino para generar números aleatorios consiste en una secuencia o lista muy larga de números diferentes, pero siempre es la misma secuencia. Con esta función puedes iniciar el sistema generador haciéndolo empezar en un punto arbitrario de esa secuencia o lista de números.

Sintaxis:

randomSeed(value);

value: Es un valor cualquiera, entero de 16 bits (int) o largo de 32 (long). Se emplea como “semilla” con la que se inicia de forma arbitraria la secuencia de números aleatorios. Cuanto más aleatorio sea a su vez esta “semilla”, más arbitrario será el inicio de la secuencia

PRÁCTICA

8. EJEMPLO 1: Conversión AD

Aquí va el primer ejemplo. Es lo más sencillo que se puede plantear. Simplemente consiste en leer el valor analógico que introduces mediante el potenciómetro conectado con la patilla A0. La conversión se realiza cada vez que acciones el pulsador conectado con la patilla 4. Con objeto de que asientes conocimientos adquiridos en las unidades anteriores, el resultado de la conversión se va a transmitir vía serie al PC y en diferentes formatos

Lo realmente novedoso del ejemplo lo tienes resaltado en la Figure 11. La función analogRead(A0) lee y convierte la señal analógica de la entrada A0 en su equivalente valor binario y lo guarda en la variable “AN0”.

Dicho valor se convierte en tensión multiplicándolo por la constante 0.0048 (la resolución por bit) y se guarda en la variable “V”. Finalmente los resultados se transmiten vía serie en formato decimal, binario, hexadecimal y en tensión. En la Figure 12 se han hecho varias conversiones. Empieza con el potenciómetro en su tope izquierdo (0 V) y lo vas moviendo sucesivamente hasta el tope derecho (5 V).

```
while(digitalRead(4));
delay(20);                  //Esperar a recibir un pulso desde D4

AN0=analogRead(A0);         //Realiza la conversión de A0 (potenciome
V=AN0*0.00488;              //Calcula la tensión equivalente

Serial.print(AN0);          //Transmite la medida en decimal
Serial.print("\t");         //Tabulación
```

Figure 11

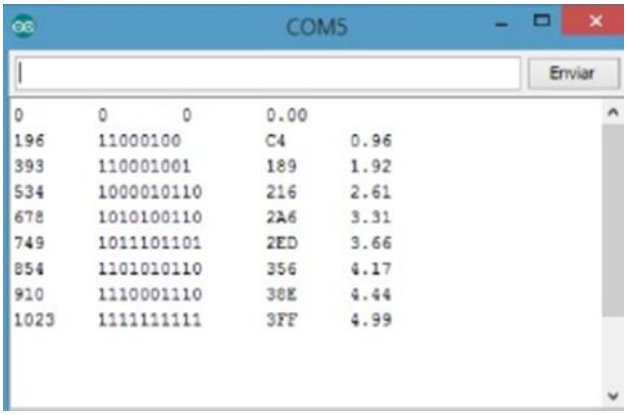

Figure 12

9. EJEMPLO 2: Umbrales

Por supuesto que con el valor binario que se obtiene en una conversión, puedes realizar cualquier tipo de operación. En este ejemplo se realizan dos conversiones, la del potenciómetro

conectado con la entrada analógica A0 y el conectado con A1. Si el resultado de convertir la entrada A0 es superior a 4 V, se activa el led rojo. Si el resultado de la entrada A1 es superior a los 2.5 V, se activa el led blanco. Si no se da ninguna de esas condiciones ambos leds permanecen apagados.

Cuando cargues el programa comprueba que para que el led rojo se encienda, es necesario girar el eje del potenciómetro prácticamente hasta la derecha. Sin embargo para encender el led blanco, basta que gires el potenciómetro aproximadamente la mitad de su recorrido. Imagina que un potenciómetro se comporta como una especie de joystick en el que puedes detectar la posición en que se encuentra dentro de su recorrido.

10. EJEMPLO 3: Comparador analógico

Como consecuencia de lo anterior, podemos hacer programas que sean capaces de comparar dos tensiones o señales analógicas y determinar quién es mayor, menor, o si ambas son iguales. Esto es lo que se propone hacer en el ejemplo.

Se comparan las tensiones analógicas V1 y V2 conectadas con las patillas A0 y A1, y procedentes de ambos potenciómetros. Los leds de salida se activan de acuerdo a la siguiente tabla.

SI...	LED
V1 = V2	Ambar
V1 > V2	Rojo
V1 < V2	Verde

11. EJEMPLO 4: Control de brillo

Seguro que has visto alguna vez un sistema para el control de la iluminación de una habitación. Mediante algún tipo de mando, el brillo aumenta o disminuye creando así diferentes ambientes. Esto mismo lo vas a hacer con este ejemplo.

La tensión analógica aplicada por la entrada A0, procedente de uno de los potenciómetros, va a servir para generar una señal PWM que regula la potencia aplicada al led blanco conectado en la patilla 6.

Observa el fragmento correspondiente al cuerpo principal del programa. ¡¡ Es muy fácil !! La función analogRead(A0) lee el valor analógico en la patilla A0.

Como ya sabes, el resultado de la conversión puede estar comprendido entre 0 y 1023. Mediante la función map() redondeas ese valor a otro equivalente comprendido entre 0 y 255, que es el parámetro que necesita la función analogWrite() para generar una señal PWM por la salida 6 (led blanco).

Carga el programa y asombra a tus familiares y amigos. Al mover el potenciómetro de izquierda a derecha o viceversa, el brillo del led aumenta o disminuye.

```
void loop()

  AN0=analogRead(A0);
  AN0=map(AN0,0,1023,0,255);

  analogWrite(6,AN0);
```

Figure 13

12. EJEMPLO 5: Timón electrónico

Vamos a seguir el mismo razonamiento que en el ejemplo anterior. Imagina que estás navegando. Con el movimiento del potenciómetro vas a mover el eje del servo al que está asociado el timón de la embarcación.

Si estudias el programa, verás que la función map() redondea el resultado de la conversión a un valor comprendido entre 0 y 180, que son los grados que puede girar el eje del servo.

13. EJEMPLO 6: Fotómetro

Vamos ahora a trabajar con el sensor de luz emulando el funcionamiento de un fotómetro o instrumento para medir la luz ambiente. Cada vez que se acciona el pulsador conectado en la patilla 4, se realiza la conversión analógica de la tensión que ofrece el sensor de luz conectado en la patilla A2.

El programa mide el valor analógico en la entrada A2 en función de la luz que incide en el sensor y calcula la tensión. Los resultados se transmiten vía serie. Considera a este ejemplo como un programa experimental. Puedes realizar varias medidas como en la figura. Empiezas con el sensor a oscuras y finalizas cuando en el sensor incide la máxima luz.

También puedes buscar una relación entre esas medidas y las proporcionadas por un instrumento comercial. A partir de ahí tú también puedes diseñar tu propio instrumento para medir la luz en luxes o en lúmenes.

14. EJEMPLO 7: Control de alumbrado

El ejemplo no aporta grandes novedades, pero puede tener una utilidad práctica como puede ser el control de un sistema de alumbrado público. Seguro que lo tienes en las calles de tu pueblo o ciudad.

Cuando se va haciendo de noche y la luz va disminuyendo, las farolas se encienden. En este ejemplo, el sensor de luz conectado con la entrada analógica A2, mide la luz ambiente. Seguro que en tu calle, en algún lugar escondido y protegido, hay un sensor similar. El led conectado con la patilla 6 simula a la farola.

Quizá sería de destacar que nuevamente se ha creado una función, Medir_luz(). Toma un número de muestras de la luz ambiente cada cierto tiempo y calcula y devuelve la media aritmética de todas ellas.

Con esto se “filtra” la señal analógica que da el sensor de luz, evitando que muy pequeñas variaciones provoquen que la farola se encienda o apague inútilmente.

El programa principal llama a nuestra función Medir_luz() y compara la media con el valor mínimo establecido para activar o no al led blanco.

- **Tu turno**

Considera el ejemplo. Es de noche y las farolas están apagadas. ¿Qué pasa si en una noche de tormenta se produce un rayo o un relámpago cuya luz incide sobre el sensor? Seguramente las farolas se apagarán ya que en ese instante nuestro programa piensa que es de día. Luego se volverán a encender. Esto es un caso real y puede ocurrir, pero ¿es deseable? Encuentra una solución que evite estas situaciones y mejora así el programa..

15. EJEMPLO 8: Medir la reflexión

El ejemplo es muy sencillo y no aporta grandes novedades. Se trata de medir la luz reflejada que recibe el sensor infra rojo (IR) conectado con la entrada analógica A3. El resultado se transmite vía serie.

- **Tu turno**

Cuando grabes el programa abre también el monitor serie para que veas las diferentes medidas. Como el convertidor tiene una resolución de 10 bits, los valores estarán comprendidos entre 0 y 1023 (2^{10}). Haz diferentes comprobaciones:

- ✓ Si no pones ningún objeto frente al sensor, verás que el valor medido es más bien pequeño. La luz IR emitida se dispersa, no rebota y sólo se recibe por tanto una pequeña cantidad de la misma.
- ✓ Si pones un objeto de color claro frente al sensor, a unos 10 mm de distancia, verás que el valor medido aumenta notablemente. Los colores claros reflejan mejor la luz IR, haciendo que reflexione y rebote hacia el sensor en un mayor cantidad
- ✓ Puedes probar ahora a la misma distancia pero con un objeto oscuro o negro. El valor leído disminuye. Los colores oscuros absorben más la luz IR y el sensor recibe menos cantidad de luz rebotada.

16. EJEMPLO 9: Detectando colores

Se trata de un ejemplo puramente experimental y susceptible de ser mejorado. Consiste en detectar el color de un objeto. Te puedes basar en los valores que obtuviste en la tabla del ejemplo anterior. En este caso vamos a distinguir entre los colores blanco y negro.

El programa espera a que acciones el pulsador conectado en la patilla 12, para realizar la medida del sensor IR conectado con la entrada analógica A3. Si esta es menor de 300 se transmite, vía serie, el mensaje “NEGRO”. Si es igual o mayor se transmite el mensaje “BLANCO”.

Como se muestra en la figura, con una sencilla cartulina puedes construir un objeto de color blanco y negro. Lo pones frente al sensor a una distancia aproximada de unos 15 mm y pulsas D12 para realizar la medida. En el monitor serie aparecerá el color del objeto.

17. EJEMPLO 10: Temperatura

Ejemplo muy parecido a los anteriores. Consiste en medir y mostrar la temperatura ambiente que capta el sensor conectado con la patilla de entrada analógica A4.

La medida se realiza cuando acciona el pulsador conectado en la patilla D4. Vía serie se transmite el resultado de la conversión (AN4), su equivalente en tensión (V = AN4 *0.0048) y la temperatura en ºC (T = AN4 * 500 / 1024).

18. EJEMPLO 11: Climatizador

Otro ejemplo con una evidente utilidad práctica. Vamos a simular el funcionamiento de un sencillo sistema de climatización. Con la figura te puedes hacer una idea.

Cuando la temperatura supera un valor establecido en la variable “Max”, se activa el led verde (patilla 9) simulando la puesta en marcha de un sistema de refrigeración. Si la temperatura ambiente está por debajo del valor establecido en la variable “Min”, se activa el led rojo (patilla 11) simulando la puesta en marcha de un calefactor. Si la temperatura medida se encuentra entre “Max” y “Min”, tanto el refrigerador como el calefactor se desactivan. Se supone que es la temperatura de confort.

19. EJEMPLO 12: A PWM SIGNAL

No se puede hacer un ejemplo más sencillo y es un buen momento para que uses la función analogWrite(). Se trata de generar una señal PWM por la patilla 6 que, en la tarjeta de experimentación, está conectada con el led blanco.

En la variable “Potencia” debes indicar el porcentaje del ciclo útil o potencia deseada. En la variable “Ciclo” se calcula el valor del ciclo útil equivalente, que está comprendido entre 0 y 255.

Observa que la función setup() está vacía, pero es obligatorio ponerla. En la función loop() tienes el cuerpo principal del programa. Mediante analogWrite() se genera una señal PWM por la patilla 6 y con el ciclo útil que hayas indicado.

Un detalle importante. Observa la función while(1);. Forma un bucle infinito sobre sí misma ya que la condición (1) siempre es “verdadera”. Es decir, el controlador no ejecuta ninguna otra función.

Sin embargo, la señal PWM sigue presente en la patilla 6, ¿por qué? Porque las señales PWM en las patillas 3, 5, 6, 9, 10 y 11 son generadas por los circuitos electrónicos internos del controlador. Una vez puestas en marcha mediante la función analogWrite(), se mantienen indefinidamente hasta que digas lo contrario. Se dice que son señales generadas por “Hardware”

20. EJEMPLO 13: Efectos ópticos

Con este ejemplo sí que podrás apreciar claramente el efecto óptico que se produce en el brillo del led blanco cuando le aplicas una señal PWM cuyo ciclo útil va aumentando y disminuyendo gradualmente.

El primer bucle for() aumenta el ciclo útil desde 0 hasta 255 en pasos de 5 en 5. El brillo del led irá aumentando del mínimo al máximo.

El segundo bucle for() disminuye el ciclo útil desde 255 hasta 0 en pasos de 5 en 5. El brillo del led irá disminuyendo desde el máximo al mínimo.

Tras una temporización de 0.5” se vuelve a repetir todo el proceso.

21. EJEMPLO 14: Regulación manual

Aquí sí que tienes un buen ejemplo de aplicación práctica. Vas a regular manualmente el brillo del led blanco conectado en la patilla 6. Mediante el pulsador 4 aumentarás el brillo aumentando el ciclo útil de la señal PWM en incrementos de 5. Mediante el pulsador 7 disminuirás el brillo acortando la duración del ciclo útil en decrementos de 5.

Te recuerdo que este mismo principio de regulación lo puedes aplicar, por ejemplo, a un motor eléctrico. Con ello conseguirías controlar su velocidad de giro.

22. EJEMPLO 15: LUCES ALEATORIAS

Considera este ejemplo como una simple curiosidad. Quizá te sirva como modelo para decorar tu árbol de navidad, escaparate, habitación, jardín, o lo que se te ocurra. Se van a generar simultáneamente 4 señales PWM por las patillas 6, 9, 10 y 11 que, como sabes, están conectadas con los leds blanco.

Aprovechando la función random() que hemos estudiado en esta misma unidad, la duración de los ciclos útiles de cada señal PWM va a ser aleatoria, por lo que la potencia aplicada a los leds también.

REFERENCIAS

BOOKS

[1]. **EXPLORING ARDUINO**, Jeremy Blum, Ed.Willey
[2]. Practical ARDUINO, Jonathan Oxer & Hugh Blemings, Ed.Technology in action
[3]. Programing Arduino, Next Steps, Simon Monk
[4]. Sensores y actuadores, Aplicaciones con ARDUINO, Leonel G.Corona Ramirez, Griselda S. Abarca Jiménez, Jesús Mares Carreño, Ed.Patria
[5]. ARDUINO: Curso práctico de formación, Oscar Torrente Artero, Ed.RC libros
[6]. 30 ARDUINO PROJECTS FOR THE EVIL GENIUS, Simon Monk, Ed. TAB
[7]. Beginning C for ARDUINO, Jack Purdum, Ph.D., Ed.Technology in action
[8]. ARDUINO programing notebook, Brian W.Evans

WEB SITES

[1]. https://www.arduino.cc/
[2]. https://www.prometec.net/
[3]. http://blog.bricogeek.com/
[4]. https://aprendiendoarduino.wordpress.com/
[5]. https://www.sparkfun.com