Universidad Europea

LAUREATE INTERNATIONAL UNIVERSITIES

ENTRADAS Y SALIDAS ANALÓGICAS

Índice

Presentación

En este recurso vamos a presentar algunas **funcionalidades específicas para microcontroladores**, que no son habituales en microprocesadores de uso general.

Los microcontroladores están pensados para trabajar como **dispositivos empotrados**, controlando sistemas en un amplio espectro de aplicaciones: juguetes, robótica, domótica, automatización de procesos industriales, electrónica de consumo (lavadoras, TV, estaciones meteorológicas), etc. Por tanto, deben incluir **circuitería específica** para realizar tareas como:

- **Minimizar el consumo energético**, dado que a menudo se operan desde baterías. Se posibilita la activación o desactivación selectiva de módulos funcionales (se retira la alimentación de aquellos módulos del micro que no sean necesarios), y se implementan diferentes modos de reposo (*sleep modes*) para evitar consumo eléctrico innecesario.

- Si funcionan de manera aislada, deben disponer de un **sistema de reinicio automático** si un programa se cuelga, dado que no hay nadie que lo reinicie manualmente. Este mecanismo se conoce como *watchdog* (perro guardián).

- **Interactuar con otros dispositivos digitales**, empleando entradas y salidas digitales, donde solo hay dos posibles niveles de tensión significativos (alto o bajo).

- **Interactuar con dispositivos analógicos** (típicamente sensores y actuadores), donde las tensiones pueden tomar cualquier valor significativo entre un mínimo y un máximo (por ejemplo, 0.7 V, 2.3 V, 4.1 V, o cualquier otro valor).

- **Controlar motores u otros actuadores** (iluminación, electroimanes, etc.), mediante señales en Modulación de Ancho de Pulso (PWM o Pulse Width Modulation).

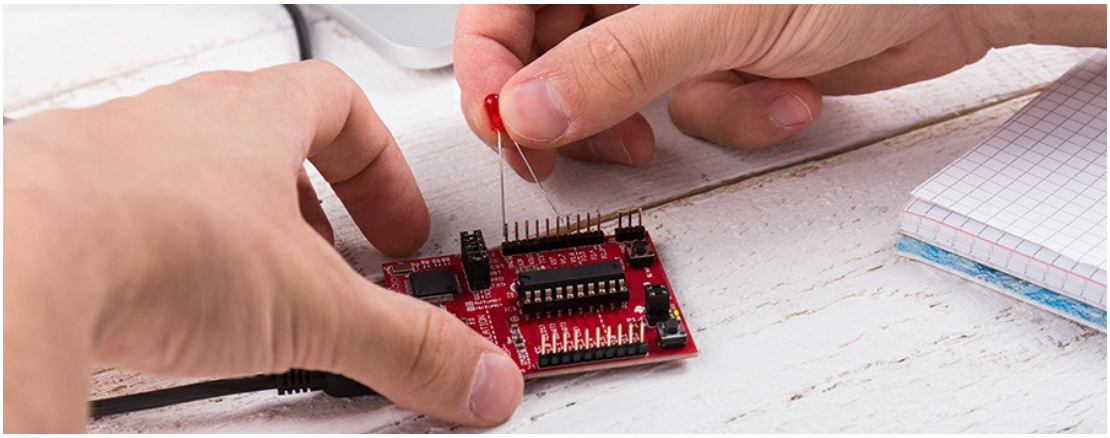
En este recurso nos centraremos en estas dos últimas características.

Los **objetivos** que se pretenden alcanzar con este recurso son los siguientes:

- Identificar algunas **funcionalidades** específicas para microcontroladores.

- Estudiar la **interconexión** con las señales analógicas de los procesadores ATmega *, tanto entradas como salidas.

- Revisar algunos **dispositivos analógicos**, tanto sensores como motores u otros **actuadores**.



* *Esta marca es una marca registrada que se cita para uso exclusivamente docente.*

Conversión de tensión analógica/digital (A/D)

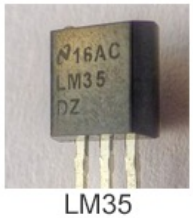
Una **tensión analógica** es un valor continuo (como estos ejemplos), por lo que es necesario digitalizarla para que un micro pueda conocer su valor. Esta operación se realiza empleando un circuito llamado **conversor analógico/digital**:

Algunos ejemplos de señales de tensión analógicas

- **Sensores de temperatura**: termopares, RTD (Resistor Temperature Detector), termistores, circuitos integrados como el LM35, LM135 o AD580.

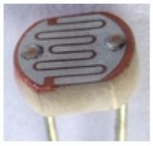
RTD

Termistor

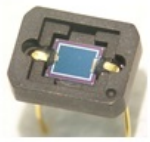


LM35
Fuente: wikimedia Commons (2014)

- **Sensores de iluminación**: LDR (Light Dependent Resistors), fotodiodos, fototransistores).

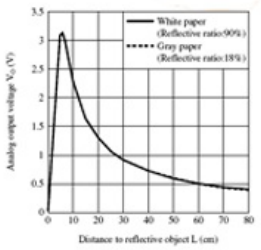


LDR
Fuente: Wikimedia Commons (2014)



Fotodiodo
Fuente: Wikimedia Commons (2008)

Sensor de distancia GP2Y0A21YK
Fuente: Wikimedia Commons (2012)



- **Potenciómetros ajustables** (de giro o desplazamiento).

Potenciómetro de giro
Fuente: Wikipedia (2016)

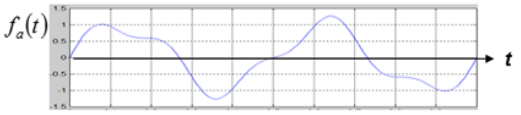
Potenciómetro deslizante
Fuente: Wikipedia (2016)

El proceso para **digitalizar** una tensión analógica es como sigue:

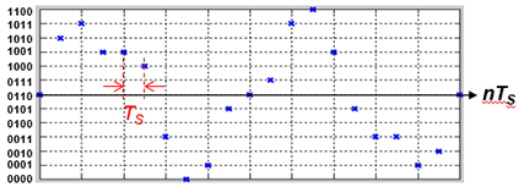
- **Seleccionar un canal de entrada**. La tensión se mide entre ese canal y la referencia de tensiones GND.
- **Iniciar la conversión** tomando una muestra de la tensión, empleando un circuito de muestreo y retención (*sample&hold*) que mantiene la medida estable mientras se hace la digitalización (que tarda un cierto tiempo).
- Completada la digitalización, el **valor digital correspondiente está listo para ir al bus de datos**.

El proceso anterior permite tomar una medida instantánea (**muestra**), empleando **m bits** para codificarla en binario. Si la medida se actualiza periódicamente, el tiempo entre muestras se denomina **periodo de muestreo T_S (seg.).** Su inversa es el número de muestras por segundo, denominada **frecuencia de muestreo f_S en Herzios (Hz):**

$$f_s = \frac{1}{T_S}$$



Señal analógica (amplitud en voltios)



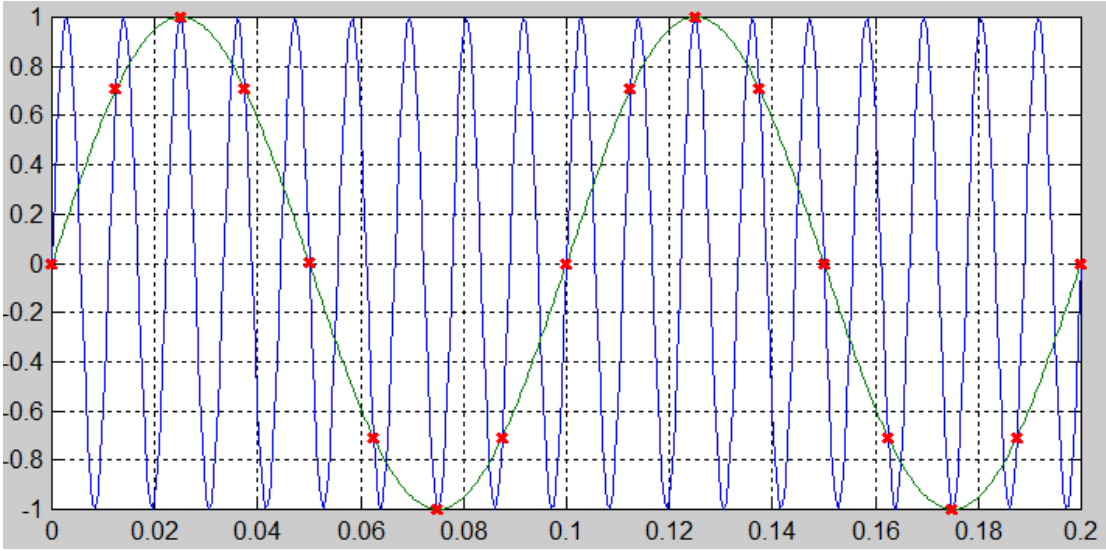
Señal digitalizada (con 4 bits)

Conversión analógica/digital (A/D): cuantificación, muestreo y *aliasing*

Cuando se digitaliza una señal analógica **se pierde información** por dos razones:

- La codificación con un número finito (m) de bits implica que las amplitudes **se truncarán al valor binario más cercano** (cuantificación). El error que introduce esta codificación se reduce incrementando el número de bits empleado, pero en contrapartida aumenta el tiempo de medida, el tamaño de los datos, y el coste.

- Los valores de la tensión analógica son **desconocidos entre muestra y muestra** . Esto es, solo conocemos cuánto vale la tensión en los momentos en que se mide. Los **cambios en la tensión** que se produzcan entre muestras **no se conocen**. Por lo tanto, si el muestreo es demasiado lento, los cambios de la señal que se han perdido pueden incluir información importante. Por ejemplo, en la siguiente figura la **onda de color azul** se muestrea demasiado lentamente: solamente conocemos de ella las muestras en color rojo:

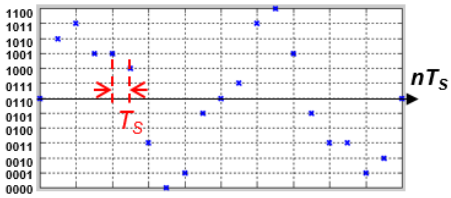


La señal en **color verde** (más lenta) produciría exactamente las mismas muestras. Por tanto como solo conocemos las muestras, identificaremos la señal verde que la representan, pero no podemos adivinar la azul que las originó en realidad. La señal verde (errónea) se denomina un **alias** de la señal azul (real), y este efecto se denomina ***aliasing***. Solo puede evitarse empleando un **muestreo lo bastante rápido**. En general, si la frecuencia más alta de la señal a digitalizar es $\mathbf{f_{MAX}}$, se debe usar una frecuencia de muestreo f_S que cumpla:

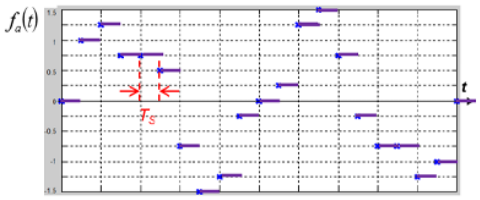
$$f_s > 2f_{MAX}\ (teorema\ de\ muestreo)$$

Conversión de tensión digital/analógica (D/A)

Para ofrecer una **salida de tensión analógica**, se emplea un conversor digital/analógico, que genera como salida una tensión continua correspondiente al valor digital de entrada (en **m bits**). Este conversor mantiene además la tensión de salida constante hasta que se actualice con una nueva entrada.



Secuencia de valores digitales (4 bits)



Señal analógica reconstruida (valores en voltios)

Los micros de Atmel* no disponen de **conversor D/A**, en su lugar ofrecen varias salidas de modulación de ancho de pulso (PWM).

Ejemplo Conversor A / D especializado

Conversor A/D especializado

Un *laptop* ordinario suele contar con un **conversor A/D especializado** de un solo canal: la entrada de micrófono. La señal de un micrófono es **analógica**, en un rango típico de -0.2 a +0.2 V. Esta entrada es digitalizada para poder grabar sonido. Asimismo, incluye una salida analógica especializada de dos (o más) canales: las **salidas de altavoces/cascos**. Genera la tensión analógica que hace vibrar una membrana en los altavoces que es la que origina las **ondas de sonido en el aire**. Con ciertas precauciones eléctricas se pueden emplear para digitalizar (o generar) otras señales también.

* *Esta marca es una marca registrada que se cita para uso exclusivamente docente.*

Entrada analógica (A/D): bits de conversión, rango de medida y resolución

La tensión analógica de entrada debe encontrarse dentro de un cierto rango de medida [V_{min}, V_{max}]. Si se emplean **m bits** para realizar la conversión, el número de niveles en que se divide el rango de medida es 2 m. Por lo tanto, la variación mínima que se puede codificar en la tensión de entrada (lo que se conoce como **resolución**) es:

$$\text{Resolución} = \frac{\text{Rango de medida}}{\text{Nº de niveles}} = \frac{V_{max} - V_{min}}{2^m}$$

En el caso de los microcontroladores ATmega328, el conversor es de 10 bits (1024 posibles niveles). El valor mínimo es V_{min}=0, mientras que el valor máximo V_{max}=**AREF** (un valor de referencia seleccionable). Este valor es común para todos los canales de entrada (de 6 a 8 según el chip).

Por lo tanto, la salida del conversor será 0000000000b (0x000) para una tensión de entrada de 0 V, y 1111111111b (0x3FF) para una tensión igual a AREF:

- Eligiendo AREF=5 V (AVCC), la resolución es 5 V/210 = 0.005 V por bit (5 mV/bit).
- Eligiendo AREF=1.1 V, la resolución es 1.1 V/210 = 0.001 V por bit (1 mV/bit).
- También puede imponerse AREF como una tensión de referencia externa.

La selección del valor para AREF se realiza escribiendo los bits 7 y 6 del **registro ADMUX**, denominados por ello REFS1 y REFS0 (*REFerece Selection*), con alguna de las siguientes combinaciones:

Table 24-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

Fuente: Bits de selección de la referencia de tensión AREF. Cortesía de Atmel

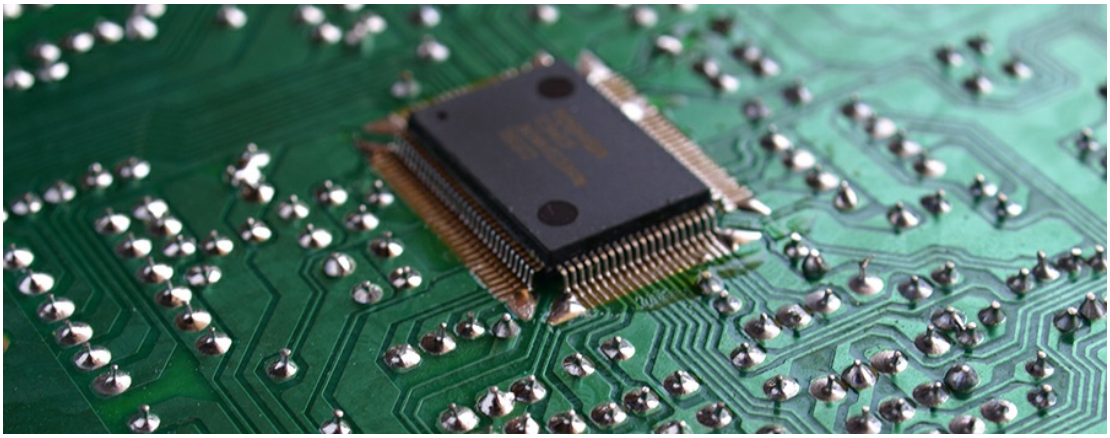
La **máxima frecuencia de muestreo** es 15 kHz (15000 muestras/s). Se puede llegar a 77 kHz bajando la resolución a 8 bits.

Relación entre la tensión de entrada analógica y el valor digital

Conocida la resolución del conversor, podemos saber qué valor digital corresponde a una tensión dada con la siguiente expresión, donde el resultado debe truncarse para ser un valor entero:

$$valor\ digital = truncar\left(\frac{tensión\ de\ entrada}{resolución}\right)$$

Ejemplo 1. Valor digital para una tensión analógica	¿Cuál es el valor digital correspondiente a una tensión de 3.57 V, en el rango de medida dado por AREF=5 V? En este caso la resolución (Voltios por bit) es 5 V/210. Por lo tanto 3.57 V se convertirá a un valor en bits: $value = \frac{3.57\,V}{\frac{5}{2^{10}}\,V/bit} = 731 \rightarrow 1011011011_b = 0x2DB$ Del mismo modo, a partir de un valor digitalizado podemos conocer la tensión medida correspondiente: $tensión = resolución\ x\ valor\ digital$
Ejemplo 2. Tensión analógica correspondiente a un valor digital	¿A qué tensión corresponde un valor 0x0DF=0011011111b con rango de medida AREF=1.1 V? La resolución (voltios por bit) es 1.1 V/210. Por lo tanto un valor 0011011111b = 223 corresponde a una tensión real de: $tensión = \frac{1.1V}{2^{10}}223 = 0.239V$



Configuración del conversor A/D en el ATmega328

Antes de poder comenzar las conversiones A/D es necesario **configurar** varios parámetros de la electrónica, como el fondo de escala (V_{max}), AREF. Como es habitual en micros, esto se hace escribiendo combinaciones predefinidas en los bits de ciertos **registros de configuración**. Los bits de configuración que debemos escribir son los siguientes:

Registro PRR (Power Reduction Register)	El bit PRADC debe ser escrito a 0 para **deshabilitar el ahorro de energía** (y por lo tanto alimentar la electrónica del conversor). Si el conversor no se va a usar, este bit se debe escribir a 1 para ahorrar baterías.
Registro ADMUX (AD Multiplexer)	• Los 4 bits MUX3:0 se usan para **seleccionar el canal de entrada** desde el que convertir (0000=canal0, 0001=canal1,…). • Los dos bits REFS1:0 se usan para **elegir la tensión máxima** (fondo de escala) AREF.
Registro ADCSRA (Analog to Digital Converter Status Register A)	• El bit ADEN (A/D enable) debe ser escrito a 1 para habilitar el conversor A/D. • El bit ADSC (A/D Start Conversion) debe ser escrito a 1 para disparar una conversión individual (desde programa). Cuando la conversión se haya completado, este bit se devuelve a 0 automáticamente. • El bit ADIE (A/D Interrupt Enable) activa/desactiva la interrupción de "conversión completada". • Los 3 bits ADPS2:0 (A/D prescaler select) ajusta la velocidad de reloj del conversor A/D (que debe quedar entre 50 kHz y 200 kHz), en base a dividir la frecuencia de reloj principal por un factor de preescalado. Por ejemplo para la frecuencia de reloj usual de 16 MHz, el preescalado ha de ser 128 (ADPS2:0= 111) de modo que 16 MHz/128=125 kHz. • El bit ADATE (A/D Auto Triggering Enable) activa/desactiva el autodisparo del conversor con una señal externa (como por ejemplo un temporizador).

Registro DIDR0: (Digital Input Disable Register)	Dado que los conectores de las entradas analógicas 0 a 5 tienen también la funcionalidad de entradas digitales, los bits 0 a 5 de este registro pueden escribirse a 1 para **deshabilitar la funcionalidad digital** (en los pines que se usen para entrada analógica) con el fin de ahorrar energía.

Registro ADCSRB (Analog to Digital Converter Status Register B)

Los 3 bits ADTS2:0 permiten elegir la señal externa para realizar conversiones automáticas (si se habilita autodisparo), según la tabla siguiente:

Table 24-6. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Fuente: Bits de selección de la señal de disparo del conversor A/D. Cortesía de Atmel

Esta tabla necesita una explicación adicional. La activación de cada conversión A/D puede realizarse de **dos maneras**:

- **Por programa** (escribiendo un 1 en el bit ADSC del registro **ADCSRA**. A continuación nos mantenemos leyendo ese bit hasta que la conversión acabe, lo que provoca su cambio automático a 0. Entonces podemos proceder a leer el resultado de la conversión. Si se va a usar este modo de trabajo, se debe seleccionar el modo "*free running*" en el registro ADCSRB (caso 000 de la tabla).
- **Automáticamente**, usando como señal de disparo una señal externa. Esta señal puede ser:
  - La **línea de interrupción 0** (caso 010 de la tabla), donde podemos conectar cualquier señal digital externa y que esta dispare la medida.
  - **Señales de salida** de los temporizadores/contadores del micro (resto de casos de la tabla). Esto permite temporizar conversiones periódicas.

En el funcionamiento automático el programa no necesita preocuparse de disparar la conversión. Para saber cuándo hay una conversión completada y lista para ser leída, el conversor lanza la **interrupción** "conversión completada" (vector de interrupción 0x003A) que debemos haber permitido configurando el bit ADIE del registro **ADCSRA**. De este modo la carga de la CPU es mínima (solo ejecutamos código cuando hay una conversión lista).

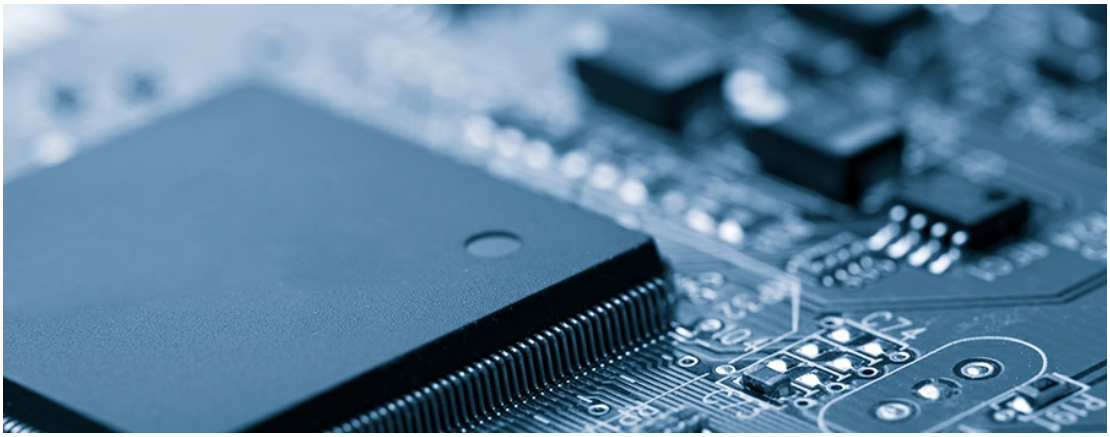
En detalle

Cómo leer el resultado de una conversión A / D

Cómo leer el resultado de una conversión A/D

Una vez completada la conversión (el bit ADSC cambia a 0, y se lanza la interrupción "conversión completada" si está habilitada), la pareja de registros **ADCH:ADCL** guardan el resultado de la última conversión A/D.

Para asegurar una lectura correcta se deben **leer primero los 8 bits bajos (ADCL),** dado que esto bloquea automáticamente nuevas conversiones. A continuación, leer los 2 bits más altos (leyendo el registro ADCH, teniendo en cuenta de que sólo los bits 1 y 0 son parte de la medida), lo que habilita de nuevo las conversiones automáticamente.



Ejemplo de configuración del módulo de entrada analógica

La siguiente rutina escribe los **registros de configuración** para hacer funcionar el conversor A/D de la siguiente manera:

- Digitalización del canal 1 (AI1).
- Rango de medida AREF=1.1 V.
- Adquisición automática disparada periódicamente por el temporizador0 (en la siguiente unidad veremos cómo se configura a su vez).

Se recomienda analizar el código con detalle, teniendo en cuenta que las etiquetas usadas para las direcciones físicas de los distintos registros, y la posición de cada bit dentro de su registro, están definidos en el fichero **m328def.inf** de Atmel*. A partir del entorno Atmel Studio 7.0 este fichero queda incluido automáticamente.

La operación "1<<n" hace que el traductor a código máquina desplace el 1 n posiciones (escribiendo un 1 en ese bit del registro):

```
1.  config_ADC:   ;configure A/D converter
2.      ; ADC enabled, conversion completed interrupt enabled, auto-triggering enabled,
    A/D clock prescaler 128 (ADPS2:0=111)
3.      LDI R16, (1<<ADEN))|(1<<ADIE)|(1<< ADATE) |(1<< ADPS2) |(1<< ADPS1) |(1<<
    ADPS0)
4.      STS ADCSRA, R16
5.      ;trigger source selected as Timer0 compare match A
6.      LDI R16, (0<< ADTS2) |(1<< ADTS1) |(1<< ADTS0)
7.      STS ADCSRB, R16
8.      ;input channel 1: MUX3:0=0001;  AREF=internal 1.1V reference (REFS1:0=11)
9.      LDI R16, (1<<MUX0)|(1<<REFS1) |(1<<REFS0)
10.     STS ADMUX, R16
11.     ;disable digital input circuitry for channel 1 (saves energy)
12.     LDI R16, (1<<ADC1D)
13.     STS DIDR0, R16
14.     ;be sure to disable the power reduction saving for the ADC circuitry
15.     LDI R16, (0<<PRADC)
16.     STS PRR, R16
17.     RET
```

* *Esta marca es una marca registrada que se cita para uso exclusivamente docente.*

Salidas PWM (Pulse Width Modulation)

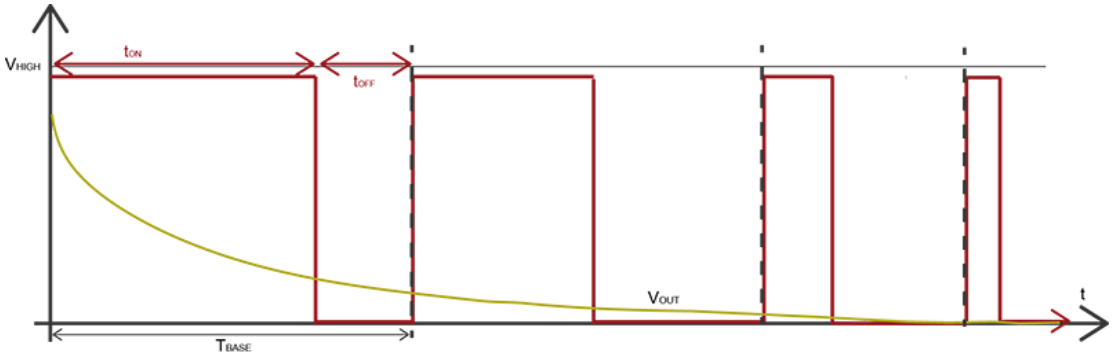
Las salidas PWM (siglas de Modulación de Ancho de Pulso) son una alternativa barata y eficiente a una salida analógica real. Básicamente, consiste en utilizar una salida digital (0 a 5 V) para crear una forma de onda con un **valor medio ajustable.** Si el dispositivo al que se le envíe esta señal es **mucho más lento** que los cambios en la **forma de onda**, éste dispositivo sólo "verá" el valor medio de la tensión (que es un valor continuo).

Por **ejemplo**, si una luz parpadea a más de 25 Hz, el ojo humano no es capaz de apreciar el parpadeo, sino que solo ve la luz medio encendida (esto se aprovecha para reproducir videos, si se reproduce una secuencia de imágenes con más de 25 imágenes por segundo, el ojo ve movimiento continuo).

Esta forma de onda se construye definiendo un tiempo base T_{BASE} para la misma (la frecuencia de la onda será $f_{BASE}=1/T_{BASE}$). Dentro de este tiempo, la señal se mantiene a nivel alto V_{HIGH} (5 V) durante un tiempo t_{ON}, y a nivel bajo (0 V) durante el resto del tiempo $t_{OFF}=T_{BASE}-t_{ON}$. Se denomina ciclo de trabajo, d (*duty cycle*), al ratio entre el tiempo en ON y el tiempo base.

El valor medio de la onda (V_{OUT}) es entonces:

$$V_{OUT} = V_{HIGH} \frac{t_{ON}}{T_{BASE}} = V_{HIGH} \cdot d$$



Por lo tanto, hay que escoger un **tiempo base significativamente menor** que el tiempo de respuesta del dispositivo a manejar (se recomienda que sea al menos la décima parte). Por ejemplo, si un motor tarda 0.5 s en acelerar desde parado hasta su velocidad nominal, tendremos que escoger un tiempo base inferior a 0.5/10 = 0.05 s = 50 ms ($f_{BASE}\geq 20$ Hz). Si lo hacemos así, el motor no "verá" el tren de pulsos que forma la onda PWM, sino únicamente su valor medio, VOUT, que podemos elegir cambiando el ciclo de trabajo **d**.

Siguiendo este ejemplo, si mantenemos la salida a 5 V durante 35 ms (y a nivel bajo durante los 15 ms restantes), el ciclo de trabajo vale $d=t_{ON}/T_{BASE}=35$ ms/50 ms=0.7, y la tensión media generada será $V_{OUT}=V_{HIGH}.d=5x0.7=3.5$ V.

Las salidas PWM en el ATmega328

Cualquier salida digital puede usarse para **generar una onda PWM por programa**: escribimos un 1, esperamos a que transcurra el tiempo t_{ON}, escribimos un 0, esperamos a que transcurra t_{OFF}, repetimos.

No obstante, para evitar **cargar al microprocesador**, los microcontroladores ofrecen la posibilidad de emplear circuitos temporizadores separados para generar automáticamente estas ondas, una vez configurados. El micro ATmega328 dispone de 3 circuitos temporizadores / contadores que pueden usarse para esta tarea:

- Timer/Counter0, de 8 bits. V_{OUT} puede ajustarse en incrementos de 5 V/28=19.5 mV.
- Timer/Counter1 y Timer/Counter2, de 16 bits. V_{OUT} puede ajustarse en incrementos de 5 V/216=0.07 mV.

Su empleo como temporizadores y su configuración como salidas PWM están muy relacionados. Vamos a indicar cómo configurar el Timer/Counter0; la configuración es análoga para los demás Timer/Counters, simplemente cambiando los registros de configuración.

El temporizador/contador0 (8 bits) como salida PWM

La **salida PWM** es el bit **OC0A** (conectado físicamente al bit 6 del PORTD del micro, y a uno de los pines del mismo). La **configuración** consiste en los siguientes *settings*:

Configurar pin como salida	**Configurar como salida el pin correspondiente** al bit OC0A, con la instrucción SBI PORTD, 6.
Configurar modo de trabajo como FastPWM	Configurar el modo de trabajo como FastPWM: en el Registro **TCCR0A**, escribir los bits WGM02:0 a 011. En este modo, el temporizador cuenta pulsos de reloj desde al valor mínimo BOTTOM (= 0) al valor máximo TOP (=0xFF), entonces se resetea y vuelve a empezar a contar automáticamente. El **temporizador** compara el **valor de cuenta** con el contenido del **registro OCR0A**, y cambia su estado (según configuremos) cuando ambos valores se igualan.

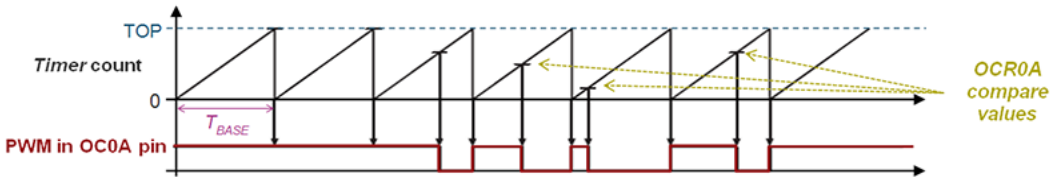
Establecer el valor objetivo para la cuenta de pulsos	**Guardar en el registro OCR0A** el valor de cuenta de pulsos correspondiente t_{ON} (teniendo en cuenta que el máximo es 0xFF). Por ejemplo, para un ciclo de trabajo del 20% (d=0.2) el valor a usar sería 0.2*0xFF=0x33. En función del valor que guardemos en este registro, el ciclo de trabajo es d = OCR0A/255, y por tanto la tensión media de salida será V_{OUT} = 5x(OCR0A/255).
Configurar el comportamiento de la salida	Elegir el modo **"borrar cuando se cumple la comparación"** (*clear on compare match*), escribiendo los bits COM0A1:0 al valor 10 en el registro TCCR0A. Esto hace que la salida PWM (bit OC0A) sea 1 mientras que la cuenta de tiempo no llegue al valor guardado en OCR0A (t_{ON}), y cambie a 0 una vez iguala este valor, manteniéndose así hasta el reseteo del temporizador (que inicia una nueva cuenta).

Funcionando de esta manera, podemos cambiar el **ciclo de trabajo d** cuando queramos escribiendo un nuevo valor en el registro OCR0A.

En detalle

Forma de onda generada en la salida OC0A

Forma de onda generada en la salida OC0A



El tiempo base T_{BASE} se establece cambiando la **frecuencia de los pulsos de reloj que cuenta el temporizador**. Para una frecuencia del reloj principal clkI/O=16MHz, el tiempo base cumple:

T_BASE=preescalado/16MHz

Donde el valor del preescalado se selecciona con los bits CS02:0 en el registro de configuración **TCCR0B** según la tabla siguiente:

Table 15-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Fuente: Bits de selección del preescalado de la frecuencia de reloj. Cortesía de Atmel

Resumen

En este recurso, se han establecido las ideas básicas sobre algunas funcionalidades avanzadas de los microcontroladores, **entradas y salidas de tensión analógica** y **salidas PWM**, que necesitan un hardware especializado (circuitos electrónicos).

En particular se estudia el caso del **microcontrolador ATmega328**, describiendo los registros de configuración que deben escribirse, con una combinación de bits establecida por el fabricante, para permitir el funcionamiento adecuado de cada módulo. Pero cualquier otro microprocesador (o microcontrolador) que implemente unas funcionalidades similares tendrá una estructura similar, con unos ciertos registros de configuración que permitan seleccionar los modos de trabajo.

El papel de los programas conocidos comúnmente como **drivers** (usados para poder utilizar el hardware de los computadores desde programa) es precisamente encapsular este diálogo con los registros de configuración y control del hardware, de manera que el programador pueda abstraerse del detalle de funcionamiento, y simplemente utilizar las prestaciones de hardware. Esto aplica no solo a hardware integrado en el procesador (como en este caso) sino al que se conecte a la **placa base** o a **puertos de expansión** de un computador.

Las **librerías en lenguaje C para Arduino** encapsulan los procedimientos de configuración en determinadas funciones. En un caso general, los fabricantes de hardware suelen también proporcionar librerías (en C o Python) para encapsular este laborioso procedimiento. Aun así, los programadores que crean estas librerías están obligados en última instancia a "conversar" en ensamblador con los registros del hardware. También, los programadores que deseen crear drivers para un procesador o un sistema operativo diferente a los que el fabricante proporcione.

¡Enhorabuena! Has finalizado con éxito.

Referencias bibliográficas

- Manual detallado (datasheet) del fabricante del procesador ATmega328 (Atmel). Páginas 93-110, y 237-252. Disponible en: <http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf> [Consultado el 20 de abril de 2016].

- Wikimedia Commons (2008). Photodiode close-up. Disponible en: <https://id.wikipedia.org/wiki/Berkas:Photodiode-closeup.jpg> [Consultado el 15 de Abril del 2016].

- Wikimedia Commons (2012). Sharp GP2Y0A21YK infrared proximity sensor.- LDR. Disponible en: <https://commons.wikimedia.org/wiki/File:Sharp_GP2Y0A21YK_IR_proximity_sensor_cropped.jpg> [Consultado el 15 de Abril del 2016].

- Wikimedia Commons (2014). LM35 temperature sensor, semiconductor thermometer. Measures 0-100 degrees C. Disponible en: <https://commons.wikimedia.org/wiki/File:LM35_temperature_sensor_semiconductor_thermometer_1480374_5_6_HDR_enhancer.jpg#metadata> [Consultado el 15 de Abril del 2016].

- Wikimedia Commons (2014). Photoresistor - LDR. Disponible en: <https://commons.wikimedia.org/wiki/File:LDR_1480405_6_7_HDR_Enhancer_1.jpg> [Consultado el 15 de Abril del 2016].

- Wikipedia (2016). Potenciómetro. Disponible en: < https://es.wikipedia.org/wiki/Potenci%C3%B3metro> [Consultado el 15 de Abril del 2016].

