Conversor digital- analogico

En el caso de un arduino Uno, el valor de 0 voltios analógico es expresado en digital como B0000000000 (0) y el valor de 5V analógico es expresado en digital como B1111111111 (1023).

Por lo tanto todo valor analógico intermedio es expresado con un valor entre 0 y 1023, es decir, sumo 1 en binario cada 4,883 mV.

Como hemos dicho Arduino Uno tiene entradas analógicas que gracias a los conversores analógico digital puede entender ese valor el microcontrolador, pero no tiene salidas analógicas puras y para solucionar esto, usa la técnica de PWM.

Las Salidas PWM (Pulse Width Modulation) permiten generar salidas analógicas desde pines digitales. Arduino Uno no posee salidas analógicas puras.

Como hemos dicho Arduino Uno tiene entradas analógicas que gracias a los conversores analógico digital puede entender ese valor el microcontrolador, pero no tiene salidas analógicas puras y para solucionar esto, usa la técnica de PWM.

Las Salidas PWM (Pulse Width Modulation) permiten generar salidas analógicas desde pines digitales. Arduino Uno no posee salidas analógicas puras.

Permite hasta 48 pines de Arduino. Usa el [TI 74HC4067](http://www.ti.com/lit/ds/symlink/cd74hc4067.pdf) para la funcionalidad de multiplexación de 16 canales y [TI 74HC595](http://www.ti.com/lit/ds/symlink/sn74hc595.pdf) para los registros de cambio (registro de desplazamiento de tres estados). Tiene múltiples modos: entrada y salida digital y entrada analógica

Ventajas de la señal digital

Sistema digital-analógico.

1. Cuando una señal digital es atenuada o experimenta perturbaciones leves, puede ser reconstruida y amplificada mediante sistemas de [regeneración](https://es.wikipedia.org/wiki/Regenerador) de señales.
2. Cuenta con sistemas de [detección y corrección de errores](https://es.wikipedia.org/wiki/Detecci%C3%B3n_y_correcci%C3%B3n_de_errores), que se utilizan cuando la señal llega al receptor; entonces comprueban (uso de redundancia) la señal, primero para detectar algún error, y, algunos sistemas, pueden luego corregir alguno o todos los errores detectados previamente.
3. Facilidad para el [procesamiento](https://es.wikipedia.org/wiki/Procesamiento_digital_de_se%C3%B1ales) de la señal. Cualquier operación es fácilmente realizable a través de cualquier software de edición o procesamiento de señal.
4. La señal digital permite la [multigeneración](https://es.wikipedia.org/w/index.php?title=Multigeneraci%C3%B3n&action=edit&redlink=1" \o "Multigeneración (aún no redactado)) infinita sin pérdidas de calidad.
5. Es posible aplicar técnicas de compresión de datos sin pérdidas o técnicas de compresión con pérdidas basados en la codificación perceptual mucho más eficientes que con señales analógicas.

Inconvenientes de la señal digital

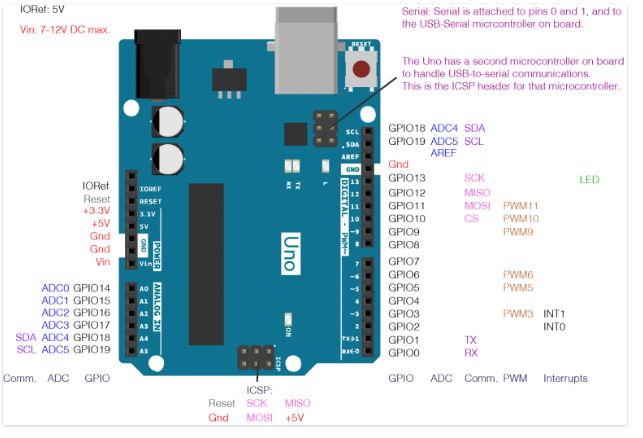
1. Se necesita una conversión analógica-digital previa y una decodificación posterior, en el momento de la recepción.
2. Si no se emplean un número suficiente de niveles de cuantificación en el proceso de digitalización, la relación señal ruido ([SNR](https://es.wikipedia.org/wiki/Relaci%C3%B3n_se%C3%B1al/ruido)) de la señal resultante se verá reducida.
3. El contenido en frecuencia de la señal digital viene limitado por la [frecuencia de Nyquist](https://es.wikipedia.org/wiki/Frecuencia_de_muestreo), de forma que la componente máxima se corresponde con la mitad de la [tasa de muestreo](https://es.wikipedia.org/wiki/Tasa_de_muestreo). Además, por cuestiones prácticas, se debe dejar un margen de seguridad desde la frecuencia de Nyquist y el límite de la banda de interés. Así por ejemplo, en los [CD](https://es.wikipedia.org/wiki/Disco_compacto), cuya frecuencia de muestreo es 44,1 kHz, la componente máxima es 22,05 kHz y la banda de interés se limita a los 20 kHz (margen del 10%).

Conversor analógico digital

Los 6 pines correspondientes a las entradas analógicas A0..A5 del Arduino UNO corresponde al puerto C. Estos 6 pines se pueden usar como entradas y salidas digitales como cualquier otro puerto digital como hemos visto en el apartado anterior. Además de denominarlas A0..A5, también es posible llamarlas como pines 14..19.

La forma de manejar con registros las entradas analógicas correspondientes al puerto C con PORT, DDR y PIN es para usar esos pines como I/O digitales, puesto que los pines de los microcontroladores son multipropósito como se ha dicho anteriormente.

En las entradas analógicas entran en juego los conversores Analógico Digital (ADC)



El datasheet de ATmega advierte de hacer lecturas rápidas entre pines analógicos (analogRead). Esto puede causar ruido eléctrico e introducir jitter en el sistema analógico. Se aconseja que después de manipular pines analógicos (en modo digital), añadir un pequeño retraso antes de usar analogRead () para leer otros pines analógicos.

Un microcontrolador solo entiende señales digitales (1’s y 0’s), por lo tanto para poder leer señales analógicas necesitamos los convertidores Analógico a Digital (ADC). Esta conversión consiste en la transcripción de [señales analógicas](https://es.wikipedia.org/wiki/Se%C3%B1al_anal%C3%B3gica) en [señal digital](https://es.wikipedia.org/wiki/Se%C3%B1al_digital), con el propósito de facilitar su procesamiento (codificación, compresión, etcétera) y hacer la señal resultante (digital) más inmune al ruido y otras interferencias a las que son más sensibles las señales analógicas.

Aunque el ATmega328P tiene 6 pines que son capaces de ser utilizados como pines de entrada analógicos (Port C), sólo hay un ADC en el microcontrolador, pero entre el ADC y los pines hay un multiplexor analógico, esto permite que podamos elegir qué pin está conectado al ADC, es decir, aunque podemos utilizar todos los pines, sólo se puede leer el valor de uno de ellos a la vez, para casi todas las aplicaciones esto es más que suficiente, pero en algunos casos limitados que necesitan lecturas ADC de alta velocidad se podría necesitar el uso de ADC externos. **En el caso de la ATmega328P los pines que se pueden utilizar una entrada analógica son todos los del puerto C**.

También se puede cambiar la tensión máxima (siempre por debajo de Vcc) que utiliza el ADC como referecia, es la llamada tensión de referencia y es la tensión contra la que todas las entradas analógicas hacen las conversiones. Esta tensión de referencia se toma del pin AREF. Reducir el voltaje máximo del ADC tiene sentido para mejorar la resolución del ADC. Con 5V la resolución es de 5/1023 = 4,88 mV para cada valor, pero para un sensor que no pasa de 3.3V la resolución es de 3.3/1023 = 3.22mV.

También tenemos que tener cuidado de la frecuencia máxima de trabajo del ADC, este valor se especifica en la ficha técnica y es de 200 kHz, este es el valor del reloj interno de la circuitería del ADC y se genera dividiendo el reloj principal ATmega, que en el caso del UNO es 16 MHz, este divisor del reloj se realiza mediante pre-escaladores y sólo hay un rango limitado de valores, por lo que la frecuencia máxima que podemos utilizar y estar dentro de la frecuencia máxima de trabajo es 125 kHz. El siguiente pre-escalador supone usar el ADC a 250 kHz, en este caso no se puede garantizar la resolución de 10 bits, pero si una resolución de 8 bits. De todas formas en caso de necesitar un ADC más rápido se podría usar uno externo.

**El ADC en microcontroladores AVR utiliza una técnica conocida como aproximación sucesiva** mediante la comparación de la tensión de entrada con la mitad de la tensión de referencia generada internamente. La comparación continúa dividiendo de nuevo la tensión y actualizando cada bit del registro ADC a 1 si el voltaje es HIGH en la comparación o 0 en el otro caso. Este proceso se realiza 10 veces (por cada bit de resolución del ADC) y genera como resultado la salida binaria.

Una de las características claves del convertidor, es su número de bits, que define la resolución con la que podemos cuantificar esa conversión a digital.  En el caso de Arduino, son 10 los bits que maneja en la puertas analógicas, lo que significa que su resolución es 1.024 posibles valores. Cuanto mayor sea esta resolución mejor es la capacidad de aproximación al valor real cuya conversión buscamos.

Como las señales que normalmente manejamos en Arduino están alrededor de los 5V, comparar contra 5V es lo razonable, porque además la industria tiene una gama completa de sensores cuyo valor máximo devuelve 5V. Pero cada vez más, la industria produce electrónica de 3,3V o podemos encontrar sensores con una sensibilidad de menos de 3V, y si usamos el ADC para digitalizar señales de pico 3,3V o menos, estamos perdiendo precisión y resolución, porque estamos desperdiciando una parte de las posibles comparaciones.

En el caso de un sensor a 3.3V, al ser 3,3V el máximo de la tensión de entrada compararlo contra 5V supone que nunca tendremos lecturas mayores de 1.024 \* 3,3 /5 = 675 y seguiremos teniendo escalones de entrada de 5mV. Como el ADC es un comparador de tensiones, si pudiéramos cambiar el valor de tensión contra el que comparamos por una de 3,3V, los escalones serian de 3,3V/1024 = 0,00322265625 o sea 3,2 mV.

En los microcontroladores AVR, el PWM está disponible con todos los timers. Timer 0 y timer 2 dan una resolución de 8 bit mientras que el timer 1 ofrece una resolución de 16 bits. Con 8 bits hay 256 pasos individuales y en 16 bit hay una resolución de 65536 pasos.

El multiplexador tiene 9 canales posibles según el datasheet



