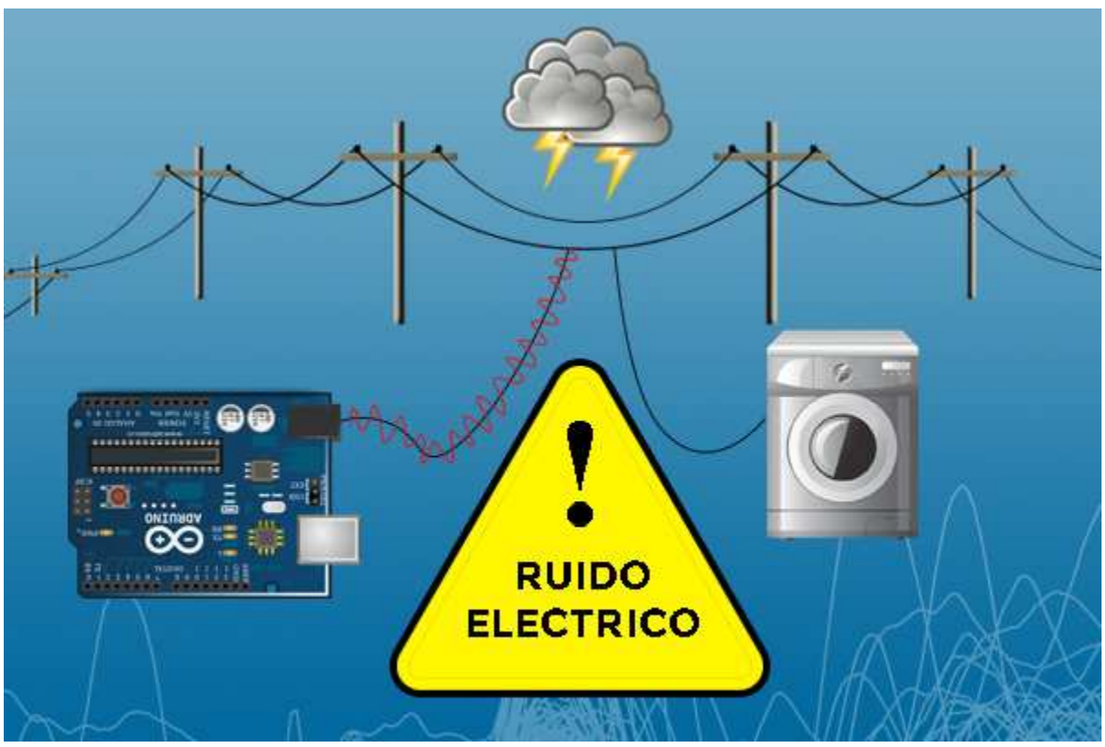
Eliminar el ruido en Arduino, controlando la señal

Fuente: Luis del Valle Hernández *(Revisión: Prof;: Bolaños 28-8-19)*

Cómo eliminar el ruido en Arduino.

Tarde o temprano te verás expuesto a esta señal parasitaria que se acopla a tu señal útil y perturba tus mediciones. Por eso es necesario saber qué es el ruido, cuáles son las fuentes que lo producen y como podemos atajarlo.

Al igual que en muchos factores involucrados en la electrónica y en la programación, no hay una fórmula mágica para la eliminación del ruido, pero si que podemos aplicar ciertas técnicas que nos ayuden a mitigarlo en la medida de lo posible.



¿Qué es el ruido?, conocerlo nos ayudará a eliminar el ruido en Arduino

Podemos encontrar muchos artículos, libros e incluso vídeos sobre el ruido en Internet. El objeto de este capítulo no es profundizar hasta tal detalle, solo quiero dejar claro cómo afecta a nuestros proyectos y cómo podemos eliminar el ruido en Arduino.
El ruido no deja de ser una señal eléctrica, al igual que las señales de Arduino, normalmente una señal continua de 5V, y de otros sistemas eléctricos que trabajan con señales alternas que pueden ir, por ejemplo, de 220V a -220V.

Por lo tanto, el ruido es una señal de interferencia que se añade o se suma a nuestra señal útil produciendo efectos perjudiciales. Vamos a verlo con un ejemplo. Imagínate un sensor de temperatura cuya relación temperatura-voltaje es de 1º-10mV. Esto quiere decir que cada grado, se incrementa en 10mV la tensión obtenida. Si medimos la temperatura en algún lugar donde sabemos que hay 20ºC exactamente, el resultado será 200mV. Al conectar el sensor a Arduino comprobamos que el voltaje obtenido es de 220mV, lo que equivale a una temperatura de 22ºC. Esos 20mV de más ¿qué son? Precisamente eso es el ruido, una señal parasitaria que se añade a nuestra señal útil y por lo tanto la altera.

Diseñamos con casos ideales que no contemplan el ruido en nuestros proyectos

Cuando estamos diseñando nuestro proyecto, utilizamos la típica Ley de Ohm o Leyes de Kirchhoff. Estas leyes se basan en desarrollos matemáticos que no tienen en cuenta el efecto del ruido. Podemos probar a crear un circuito muy sencillo con una resistencia y un LED. Si hacemos los cálculos y obtenemos los voltajes, comprobaremos con un multímetro que aunque se parezcan, el resultado no es el mismo. Esto es debido al ruido inherente de la propia placa de Arduino, los componentes e incluso el multímetro. En consecuencia debemos contemplar esta señal perjudicial y por lo tanto eliminar el ruido en Arduino.

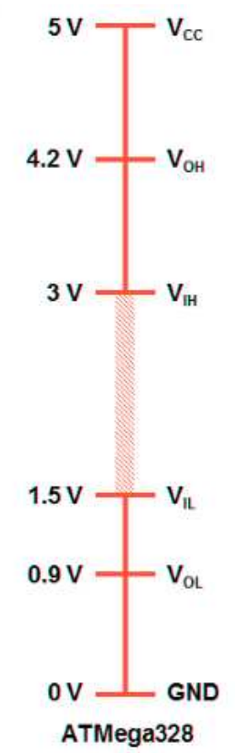
La señal de ruido no afecta de igual manera a las señales analógicas que digitales

Ya hemos visto el ejemplo de un sensor de temperatura con una señal analógica. Pero no afecta en igual medida a las señales digitales, dependerá de la amplitud del ruido. En términos generales, **la señales analógicas son más sensibles al ruido**.
Si en el ejemplo anterior del sensor de temperatura, la señal de ruido tuviera 0.3mV en vez de 20mV, no afectaría a nuestras medidas. Por lo tanto la amplitud de la señal es importante. En el caso de las señales digitales, la amplitud también es importante, pero solo afectará si dicha amplitud es lo suficientemente grande para cambiar de estado de alto a bajo o de bajo a alto.

Cómo funcionan las señales digitales

Hagamos un paréntesis para ver cómo funcionan las señales digitales en Arduino. Estamos acostumbrados a hablar de alto y bajo y relacionarlo con 0V y 5V. Pero eso no es realmente lo que sucede dentro de la placa. Existen unos márgenes para identificar si la señal está en estado alto o bajo.

Precisamente este es uno de los puntos fuertes de Arduino, gracias a la amplitud de ese rango la señal digital soporta una gran cantidad de ruido. A continuación puedes ver la imagen donde se representan esos márgenes para el microcontrolador ATmega328 del Arduino UNO.



Como comprobamos, de 0V a 1,5V se considera un estado bajo. De 3V a 5V se considera un estado alto. Por último de 1,5V a 3V se considera una indeterminación. Supongamos que tenemos un nivel bajo en la señal digital con un valor de 0,5V. Para que se produzca un cambio a una indeterminación necesitaremos que la amplitud del ruido sea de más de 1V y para que cambie a estado alto más de 2,5V.

Por lo tanto, **el ruido no afecta en la misma medida a las señales digitales** eso sí, cuando afecta los resultados pueden ser «catastróficos». Vamos a verlo con un ejemplo de la vida real

En el año 2010 se produjo el apagón analógico, se pasó de emitir la televisión en analógico a digital (en España)

Recordarás que antes la televisión se veía de diferente manera. Antes del año 2010, se emitía con una señal analógica. A partir de ese año se empezó a emitir en digital a través de la TDT (Televisión Digital Terrestre).
Con la televisión analógica, cuando había interferencias por ruido, la señal se distorsionaba y se producía el típico efecto nieve. Aún así se conseguía ver la televisión, aunque sin calidad. Con la TDT se consiguieron varias cosas entre ellas una mejor calidad, la necesidad de menor potencia

para emitir y la optimización del ancho de banda. El gran problema de este tipo de señales es que si le afecta el ruido, la televisión se deja de ver. Seguramente te habrá pasado alguna vez donde la televisión se ha quedado en negro. Esto sucede por lo anteriormente comentado, la señal de ruido tiene tanta amplitud que hace cambiar de estado a los bit digitales.
En resumen, el ruido es otra señal que interfiere con nuestra señal útil. Afecta en mayor medida a señales analógicas que digitales, pero cuando estas últimas son afectadas, su recuperación es complicada. Si queremos eliminar el ruido en Arduino, debemos conocer las fuentes y factores que lo producen.

Fuentes de ruido en los sistemas eléctricos

Conocer las fuentes de ruido nos va permitir atacar la raíz para evitar la perturbación en la señal útil. Como veremos más adelante, en ocasiones no es posible y debemos convivir con esta interferencia. Existen diferentes tipos de ruido electrónico que podemos clasificar en dos grandes grupos dependiendo de su origen.

Ruido interno o inherente

Es el producido en el interior de los dispositivos electrónicos como consecuencia de su naturaleza física. A su vez se divide en tres tipos:

- **Ruido térmico o ruido blanco o ruido de Johnson-Nyquist**
  Movimiento de los electrones dentro de un conductor causado por la agitación térmica.

- **Ruido de disparo**
  Fluctuaciones aleatorias de la corriente eléctrica a través de un conductor.

- **Ruido de tiempo de disparo**
  Es el producido, por ejemplo, en los transistores por su naturaleza física.

Este ruido es aleatorio es decir, no tiene un patrón definido. Por lo tanto es muy complicado eliminar el ruido en Arduino. Influirá, en mayor o menor medida, en todos nuestros proyectos como consecuencia de la utilización de componentes electrónicos como conductores, cables, resistencias, protoboard, etc…

Ruido externo o interferencias

Al contrario que el ruido interno, éste se produce fuera de los dispositivos. Son interferencias producidas por acoplamiento eléctrico y/o magnético. A su vez, se divide en dos tipos:

- **Ruido generado por sistemas creados por el hombre**
  En este tipo se clasifican todos los aparatos eléctricos producidos por los seres humanos. En mayor o menor medida, producen interferencias que pueden afectar a nuestra señal. Entre ellos se encuentran por ejemplo las lavadoras, televisores, ordenadores e incluso la fuente de alimentación con la que alimentamos nuestro Arduino. En todos ellos se produce algún tipo de acoplamiento eléctrico y/o magnético.

- **Ruido generado por sistemas naturales**
  Este tipo de ruido es más difícil controlarlo ya que es complicado predecir su comportamiento y su aparición. Los rayos o cargas electrostáticas son un ejemplo.

El ruido interno puede ser periódico, intermitente o aleatorio, haciendo difícil eliminarlo. En muchos casos no hay una conexión directa con la fuente, incluso puede estar a cientos de

metros y propagarse a través del tendido eléctrico. Efectos como una lavadora en funcionamiento, un rayo cerca de tu vivienda o el propio ordenador pueden ser fuentes de ruido.
En resumen, debemos de ser capaces de eliminar el ruido en Arduino. Para ello necesitamos conocer las fuentes de ruido. El ruido interno siempre lo tendremos presente, por la naturaleza física de los componentes que utilizamos, y el ruido externo será más difícil de controlar debido a que en ocasiones no sabremos a ciencia cierta que está introduciendo ruido en nuestro sistema.

Técnicas, procedimientos y consejos para eliminar el ruido en Arduino

Ya hemos visto qué es el ruido y las fuentes que lo pueden generar. Ahora vamos a ver algunas técnicas y consejos que nos serán útiles a la hora de eliminar el ruido en nuestros proyectos. Antes de nada, debemos ser conscientes que es imposible eliminar el ruido completamente. Pero esto no implica que debamos pasarlo por alto. Debemos de ser capaces de minimizar lo máximo posible sus efectos.
Podemos atacarlo desde dos puntos de vista diferentes, reduciendo el ruido en su fuente o propagación y reduciendo el ruido a base de filtros y promediando la señal. Veamos cuales son las características principales de estas dos técnicas.

Reducción del ruido en su fuente o propagación

Estas técnicas son, en ocasiones, poco eficaces y difíciles de implementar. Eso sí, si las conseguimos aplicar de una manera óptima, es la manera más efectiva de reducir el ruido. Esto es debido a que atacamos la fuente o la propagación de la distorsión. Por lo tanto evitamos que se mezcle con la señal útil evitando así su degradación.
Se suele utilizar, sobre todo, para el ruido externo.

Reducción del ruido filtrando y promediando la señal

Estas técnicas se centran en aplicar filtros y promediados en la señal una vez ha sido contaminada con el ruido. Al contrario que las técnicas que atacan a la fuente, éstas técnicas son más comunes y efectivas.
Su objetivo es amortiguar el ruido frente a la señal útil. Para ello se pueden aplicar estas técnicas a través del software o del hardware (electrónica). Por ejemplo, podemos eliminar las interferencias que se producen en una imagen aplicando los famosos filtros de suavizado. Son técnicas muy conocidas y que consisten en calcular el promediado de un píxel con sus vecinos.
El ruido puede provenir de los dispositivos de captura o de efectos adversos de la iluminación.

No existe una solución única que podamos aplicar para la reducción de ruido

Dependerá de cada caso, de cada proyecto, de los dispositivos involucrados el elegir unas técnicas u otras. Es necesario una fase de investigación que nos permitan identificar las fuentes de ruido que afectan a nuestro sistema.
Por ejemplo, el propio Arduino ya contempla alguna de estas medidas. Esto no implica que no podamos adoptar las mismas u otras medidas para combatir el efecto del ruido. Para que te hagas una idea, puedes ver el esquema eléctrico de la placa de Arduino UNO y comprobar las protecciones que llevan integradas.

Consejos prácticos para eliminar el ruido en Arduino

Como ya te he comentado, no existe una fórmula mágica que elimine el ruido en nuestros sistemas. Pero siguiendo ciertas prácticas podremos minimizarlo al máximo. A continuación te dejo unos consejos que deberías aplicar en todos tus proyectos.

- **Desacoplos de todas las fuentes de alimentación con condensadores de Bypass.**
- **Apantallar cables que transmitan datos analógicos. Recuerda que estas señales son más sensibles al ruido.**
- **Utilizar tomas de tierra separadas para las etapas analógica, digital y para el propio Arduino.**
- **En la medida de lo posible, utilizar los reset de la placa por hardware en vez de por software.**
- **Utilizar una fuente independiente para el microcontrolador.**
- **No utilizar cables largos, reducirlos al máximo.**
- **Utilizar componentes de calidad, con un diseño de bajo ruido.**
- **En sensores analógicos utilizar filtros paso bajo.**
- **Por software, haciendo el promediado de la señal.**
- **Con electrónica, utilizando una resistencia y un condensador.**
- **Blindaje de cables y dispositivos con materiales no conductores.**
- **Intentar controlar el ruido en la fase de diseño de nuestro proyecto. Muchos proyectos han tenido que ser rediseñados una vez terminados.**

Ejemplos prácticos donde se aplica la reducción de ruido

Por último te voy a hablar de dos ejemplos prácticos donde he aplicado el filtrado que me ha permitido eliminar el ruido en Arduino. El primer es el caso del sensor LM35 en el curso Crear un dispositivo del IoT con Arduino MKR1000. En una de las fases, pudimos comprobar como este sensor fluctuaba en ocasiones hasta dos grados.

La solución, eliminar el ruido en Arduino aplicando un promediado de la señal de entrada por el pin analógico. Este promediado consiste en calcular la media de un número de muestras. Un factor a tener en cuenta es la cantidad de muestras que se utilizan para hacer la media.

Cuantas más muestras, mejor será la media pero más tiempo tarda en detectar cambios en la señal. Por el contrario si cogemos un rango muestral bajo, la media será peor pero tardará menos tiempo en adaptarse a los cambios. Por lo tanto, debemos encontrar un equilibrio entre precisión y rapidez cogiendo el número de muestras ideal para nuestra situación.

El segundo proyecto donde he tenido que aplicar un filtrado de ruido es en el Sensor de nivel de agua con Arduino. En este proyecto apliqué la misma técnica que en el caso anterior pero sobre la distancia obtenida a través del sensor ultrasónico.

Además de aplicar un filtrado, Fram Mart, un alumno del Campus de Programarfacil, me aconsejó utilizar dos tubos conectados al Trigger y Echo del sensor de ultrasonidos, para conseguir más precisión. Otra técnica diferente para reducir un ruido externo.

Conclusión

No hay fórmulas mágicas, como ya te he dicho, para eliminar el ruido en Arduino. Debemos conocer muy bien tanto las fuentes que nos puedan producir ruido, como los componentes que tengamos conectados.

Hay que poner énfasis en la alimentación de nuestros circuitos, señales analógicas, componentes electrónicos y cableado del sistema. Solo así conseguiremos ser inmunes al ruido.