COMO REINICIAR ARDUINO

Hay muchas formas de reiniciar. Acá presentamos Algunas, las más comunes.
Si queremos saber cuántas veces se ha reiniciado un Arduino, siempre podemos guardarlo en la EEPROM, o en un Archivo si disponemos de Lectora/Grabadora de Tarjetas SD

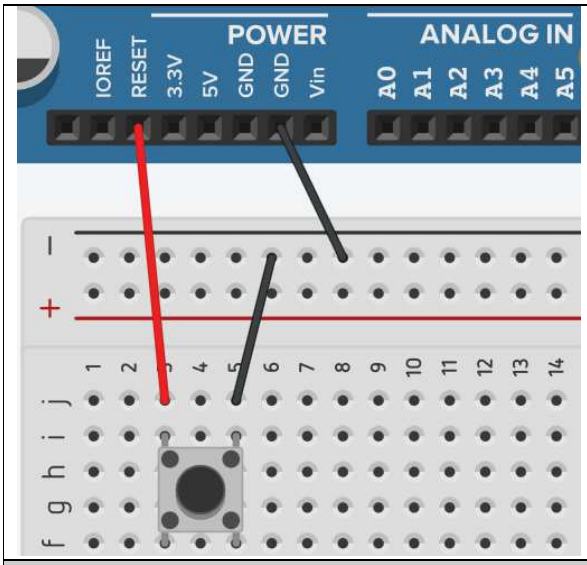
Reseteo por Hardware

1) Reseteo por Hardware - Botón Externo.

Casi todos los modelos de Arduino disponen de al menos un pin de Reset, destinado a este fin. Reiniciar Arduino con esta pin es muy simple. Solo es necesario llevar este Pin llevar a nivel LOW y se reiniciará el microcontrolador. Dicho de otra forma: Debes conectar el Pin "**Reset**" con el Pin "**GND**".
Para que esta simple conexión sea de de utilidad cada vez que nos haga falta, usaremos un botón, que al ser presionado (durante un mínimo de 2.5 µs) reiniciará nuestra placa Arduino. Y lo mejor de todo es que funciona sin una sola línea de código.

Pero como puedo ver que realmente se ha reiniciado la placa? Bueno te propongo un código muy Simple, que te permitirá apreciar este muy rápido reinicio del microprocesador. Insisto el código solo sirve para que puedas apreciar visualmente y através del puerto serie que Arduino se esta reindicando.

Circuito botón de reinicio externo Arduino



(Programa "850_Reset_Arduino_01_Boton")

```
1   int C;
2
3   void setup( ) {
4     Serial.begin(9600);
5     while (!Serial);
6     C = 0;
7     Serial.println("Inicia Programa.");
8   }
9
10  void loop( ) {
11    C++;
12    Serial.print(C);
13    Serial.println(" - Presiona Boton Para
14  Reiniciar");
15    delay(250);
16  }
```

El código solo sirve para que puedas apreciar visualmente y através del puerto serie que Arduino se esta reindicando

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Protoboard - 3 Cables de conexión Macho/Macho - 1 Botón Pulsador de 2 Patas o 4 patas - Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: Como se muestran en la imagen del circuito y el cable USB desde Arduino a La PC.

Reseteo por Software y Hardware

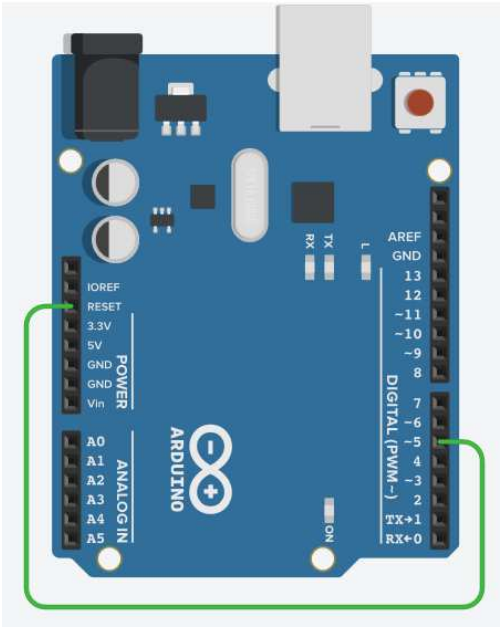
2) Reseteo Por Software y Hardware - Pin Reset.

En este caso, no es necesario apretar botones, pero necesitamos un pin exclusivo para este proceso, y podemos reiniciar desde nuestro programa, cada vez que sea necesario, haciendo lo mismo que hacíamos cuando apretamos el botón. Llevar el Pin "**Reset**" a nivel LOW.
Para el caso del Ejemplo usamos el Pin 5, para activar el reinicio de Arduino, pero podríamos usar cualquier otro que este libre. Recordar que ese pin Debe ser exclusivo para activa el reinicio.

IMPORTANTE: algunas placas Arduino pueden presentar el problema que, no permiten que se actualice su programa desde la PC mientras se encuentre conectado el cable al Pin Reset. El problema desaparece desconectando el pin reset durante la actualización. Luego se ser actualizada la placa Arduino se puede conectar y todo funciona bien.

(Programa "**850_Reset_Arduino_02_A_Software_y_Hardware**")

```
1  int ResetPin = 12;
2
3  void setup( ) {
4  /*** Estas tres líneas deben estar en este orden o no funcionara ***/
5    digitalWrite(ResetPin, HIGH);
6    delay(200);
7    pinMode(ResetPin, OUTPUT);
8  /*******************************************************/
9    Serial.begin(9600); //Inicializo Puerto Serie
10   while (!Serial);      // Esperamos que el puerto serie este abierto.
11
12   Serial.println("Iniciando Arduino..");
13   delay(1000);
14 }
15
16 void loop( ) {
17   Serial.println("Reiniciando Arduino en 2 Segundos");
18   delay(2000);
19
20   digitalWrite(ResetPin, LOW);  // Reinicia Arduino
21
22   delay(1000);
23   Serial.println("Esto Nunca Pasa, porque Arduino se Reinicio");
24 }
```



CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Cable de conexión Macho/Macho - Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: Como se muestran en la imagen del circuito, desde el pin 5 directo al Pin Reset y el cable USB desde Arduino a La PC.

3) Reseteo Por Software y Hardware - Pin Reset, cada intervalos regulares de tiempo.

En este caso, no es necesario apretar botones, pero necesitamos un pin exclusivo para este proceso, y podemos reiniciar desde nuestro programa, cada vez que sea necesario, haciendo lo mismo que hacíamos cuando apretamos el botón. Llevar el Pin "**Reset**" a nivel LOW.

Para el caso del Ejemplo usamos el Pin 5, para activar el reinicio de Arduino, pero podríamos usar cualquier otro que este libre. Recordar que ese pin Debe ser exclusivo para activa el reinicio.

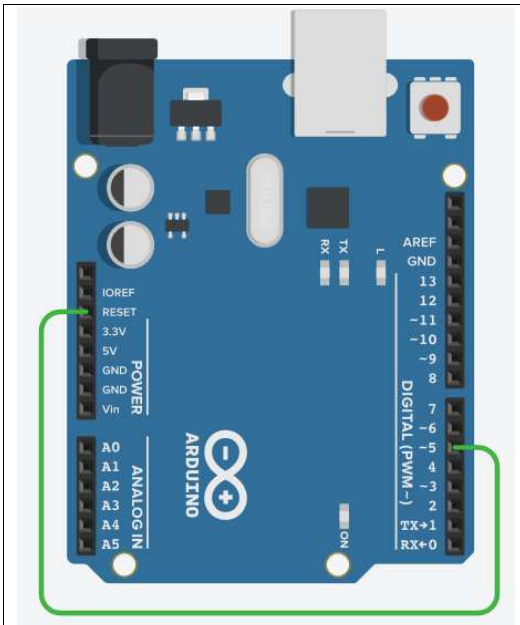
IMPORTANTE: algunas placas Arduino pueden presentar el problema que, no permiten que se actualice su programa desde la PC mientras se encuentre conectado el cable al Pin Reset. El problema desaparece desconectando el pin reset durante la actualización. Luego se ser actualizada la placa Arduino se puede conectar y todo funciona bien.

(Programa "**850_Reset_Arduino_02_B_Software_y_Hardware**")

```
   int ResetPin = 5;

   unsigned long IntervaloSegundos,
                 Inicia_Segundos,
                 Pasaron,
                 RegresivaSegundos,
                 Anterior_RegresivaSegundos,
                 Queda;

   void Inicia_Intervalo(unsigned int H, unsigned long M, unsigned long S){
      IntervaloSegundos = H*3600 + M*60 + S;
      Inicia_Segundos = millis( )/1000;
      Serial.print("Iniciando Intervalo de ");
      Serial.print(IntervaloSegundos);
      Serial.println(" Segundos");
   }

   unsigned long Intervalo( ){
1     Pasaron =  millis( )/1000 - Inicia_Segundos;
2     Anterior_RegresivaSegundos = RegresivaSegundos;  // Para saber cuando Mostrar el Tiempo
3     RegresivaSegundos = IntervaloSegundos - Pasaron;
4     if( Pasaron >= IntervaloSegundos){
5        Serial.println("Reiniciando Arduino en 2 Segundos");
6        delay(2000);
7        digitalWrite(ResetPin, LOW);  // Reinicia Arduino
8        delay(1000);
9     }
10    return (RegresivaSegundos);
11 }
12
13 int Mostrar_Tiempo( ){
14   int ok=0;
15   if(RegresivaSegundos != Anterior_RegresivaSegundos){
16     ok=1;
17   }
18   return (ok);
19 }
20
   void setup( ) {
     Serial.begin(9600); //Inicializo Puerto Serie
     while (!Serial);       // Esperamos que el puerto serie este abierto.

     Serial.println("Iniciando Arduino..");
     Serial.print("Pin para Reiniciar: ");
     Serial.println(ResetPin);

     digitalWrite(ResetPin, HIGH);     //Estas tres líneas deben estar en este orden o no funcionará
     delay(250);                       //Estas tres líneas deben estar en este orden o no funcionará
     pinMode(ResetPin, OUTPUT);    //Estas tres líneas deben estar en este orden o no funcionará

     Inicia_Intervalo(0,1,5);
     delay(1000);
   }

   void loop( ) {
     Queda = Intervalo( );
```

```
  if(Mostrar_Tiempo( )){
     Serial.println(Queda);
  }
}
```

-o-



CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Protoboard - 3 Cables de conexión Macho/Macho - 1 Botón Pulsador de 2 Patas o 4 patas - Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: Como se muestran en la imagen del circuito, desde el pin 5 directo al Pin Reset y el cable USB desde Arduino a La PC.

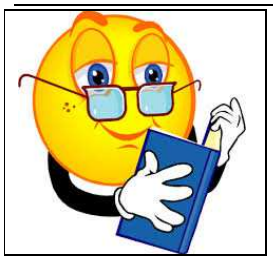
Reseteo por Software

4) Reseteo Por Software - Función "resetfunc( )"

Acá veremos como reiniciar Arduino por medio del software (desde nuestro programa y sin cables de conexión). Así que si usted está buscando un manos libres o automático, debe saber que Arduino tiene incorporada una función llamada "**resetFunc( )**" que debemos declarar en la dirección 0 (cero) y cuando ejecutamos esta función, Arduino se reinicia automáticamente. Por lo tanto, no es necesario hacer nada en el hardware y simplemente escribir un breve código y cargarlo en su placa Arduino.

(Programa "**850_Reset_Arduino_03_A_Software**")

```
1   void setup( ) {
2     Serial.begin(9600); //Inicializo Puerto Serie
3     while (!Serial);        // Esperamos que el puerto serie este abierto.
4
5     Serial.println("Iniciando Arduino..");
6     delay(1000);
7   }
8
9   void(* resetFunc) (void) = 0; //Se Declara la Función reset con dirección 0 (cero)
10
11  void loop( ) {
12
13    Serial.println("Reiniciando Arduino en 2 Segundo");
14    delay(2000);
15
16    resetFunc( );  //Llamo a Función reset - Reinicia Arduino
17    delay(100);
18
19    Serial.println("Esto Nunca Pasa, porque Arduino se Reinicia");
20  }
```



CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: En este ejemplo no hay cables que conectar, salvo el USB desde Arduino a La PC.

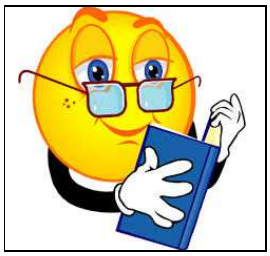
5) Reseteo Por Software - Función "resetfunc( )". El reseteo debe hacerse a Intervalos regulares.

Acá veremos como reiniciar Arduino por medio del software (desde nuestro programa y sin cables de conexión). Así que si usted está buscando un manos libres o automático, debe saber que Arduino tiene incorporada una función llamada "**resetFunc( )**" que debemos declarar en la dirección 0 (cero) y cuando ejecutamos esta función, Arduino se reinicia automáticamente. Por lo tanto, no es necesario hacer nada en el hardware y simplemente escribir un breve código y cargarlo en su placa Arduino.

(Programa "850_Reset_Arduino_03_B_Software_A_Intervalos_Regulares")

```
1   struct Intervalo{
2     unsigned long IntervaloSegundos,
3                   Pasaron,
4                   RegresivaSegundos,
5                   AnteriorRegresivaSegundos,
6                   Inicia_Segundos;
7
8     void Inicia(unsigned long S);
9     void Inicia(unsigned int H, unsigned long M, unsigned long S);
10    unsigned long CuentaRegresiva( );
11    int PuedeMostrar( );
12  };
13
14   void Intervalo::Inicia(unsigned long S){
15     Inicia(0,0,S);
16  }
17
18  void Intervalo::Inicia(unsigned int H, unsigned long M, unsigned long S){
19     Inicia_Segundos =  millis( )/1000;
20     IntervaloSegundos =   H*3600 + M*60 + S;
    }

    unsigned long Intervalo::CuentaRegresiva( ){
       Pasaron = millis( )/1000 - Inicia_Segundos;
       AnteriorRegresivaSegundos = RegresivaSegundos;
       RegresivaSegundos = IntervaloSegundos - Pasaron;
       if( Pasaron >= IntervaloSegundos){
          Inicia_Segundos = millis( )/1000;
          RegresivaSegundos = 0; // Aseguro que retorne Cero
       }
       return(RegresivaSegundos);
    }

    int Intervalo::PuedeMostrar( ){
      int ok=0;
      if( AnteriorRegresivaSegundos != RegresivaSegundos){
         ok=1;
      }
      return(ok);
    }

    Intervalo A,B,C; // Declaro Varios Intervalos, aunque solo necesito y usaré el A
```

```
void(* resetFunc) (void) = 0; //Se Delara la Funcion reset con direccion 0 (cero)

void setup( ){
   Serial.begin(9600);
   while(!Serial);

   Serial.println("Iniciando Arduino...");

   A.Inicia(65);       // Configuro intervalo de reinicio en 65 segundo
//  A.Inicia(0, 1, 5);  // También podría haberlo configurado así (función Sobrecargada)
}

unsigned long T1;

void loop( ){
  T1 = A.CuentaRegresiva( );
  if( A.PuedeMostrar( )){
    Serial.print("T1: ");
    Serial.println(T1);
  }
  if(T1 == 0){
    Serial.println("Reiniciando Arduino en 2 Segundo");
    delay(2000);
    resetFunc( );  //Llamo a Función reset - Reinicia Arduino
    delay(100);
  }
}
```



CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: En este ejemplo no hay cables que conectar, salvo el USB desde Arduino a La PC.

Reseteo por Medio De Interrupciones

6) Reinicio Por Medio De Interrupciones - Arduino Watchdog.

Esta forma de reiniciar Arduino, será explicada muy superficialmente, y no la desarrollaremos, ya que es tema para mas adelante.

En electrónica, un perro guardián (en inglés watchdog) es un mecanismo de seguridad que provoca un reset del sistema en caso de que éste se haya bloqueado.

Consiste en un temporizador que irá continuamente decrementando un contador, inicialmente con un valor relativamente alto. Cuando este contador llegue a cero, se reiniciará el sistema, así que se debe diseñar una subrutina en el programa de manera que refresque o reinicie al perro guardián antes de que provoque el reset. Si el programa falla o se bloquea, al no actualizar el contador del perro guardián a su valor de inicio, éste llegará a decrementarse hasta cero y se reiniciará el sistema mediante una interrupción.

El watchdog es un timer que va dentro de la arquitectura del microcontrolador y es independiente del resto de timers.
La interrupción del watchdog es la más prioritaria en la tabla de los vectores de interrupción.

El Watchdog Timer en el microprocesador del Arduino solo tiene una fuente de reloj que es su propio oscilador interno de 128kHz (a diferencia de los temporizadores internos de 8/16bit, que pueden usar el reloj del sistema de 16Mhz o un reloj externo). Es este oscilador separado permite que el WDT funcione en el modo de potencia más baja: SLEEP_MODE_PWR_DOWN .

01000101 01101100 00100000 01010011 01100001 01100010 01101001 01101111 00100000
01010000 01101001 01110100 11000011 10100001 01100111 01101111 01110010 01100001
01110011 00100000 01100100 01100101 00100000 01010011 01100001 01101101 01101111
01110011 00100000 01100100 01101001 01101010 01101111 00100000 01100001 01101100
01100111 01110101 01101110 01100001 00100000 01110110 01100101 01111010 00111010
00100000 00100010 01000101 01101100 00100000 01110000 01110010 01101001 01101110
01100011 01101001 01110000 01101001 01101111 00100000 01100101 01110011 00100000
01101100 01100001 00100000 01101101 01101001 01110100 01100001 01100100 00100000
01100100 01100101 01101100 00100000 01110100 01101111 01100100 01101111 11100010
10000000 10011101 00101110

Si tienes algunas Correcciones y/o Sugerencias, por favor contáctame.