Lección 1—Fundamentos de Arduino

Funcionamiento, instalación del software y manejo básico

El software Arduino (IDE) te permite escribir programas y subirlos a tu pizarra. En la página de Arduino Software encontrará dos opciones:

https://www.arduino.cc/en/Main/Software

1. Si tiene una conexión a Internet fiable, debe utilizar el IDE (Arduino Web Editor) en línea. Este le permitirá guardar sus bocetos en la nube, teniéndolos disponibles desde cualquier dispositivo y con una copia de seguridad. Siempre tendrá la versión más actualizada del IDE sin necesidad de instalar actualizaciones o librerías generadas por la comunidad.

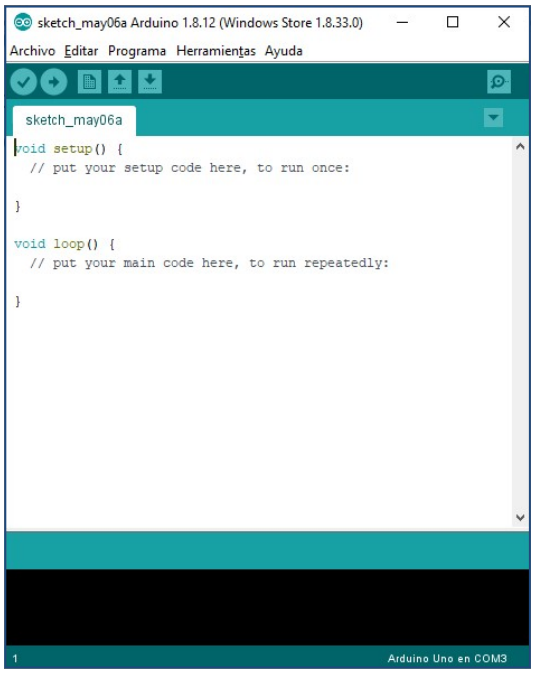
2. Si prefieres trabajar sin conexión, deberías usar la última versión del IDE de escritorio.

Instalar el IDE de Arduino Desktop

Descargue el software adecuado según su sistema operativo.

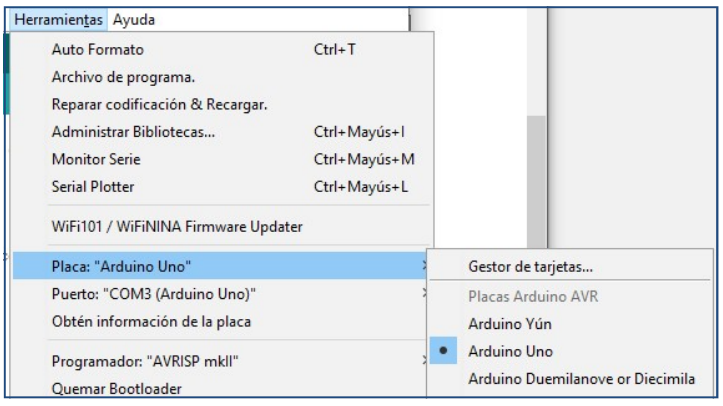
En Windows 10 se puede descargar directo de la store de Windows.

Instalamos el software y ejecutamos y veremos la pantalla principal del programa.

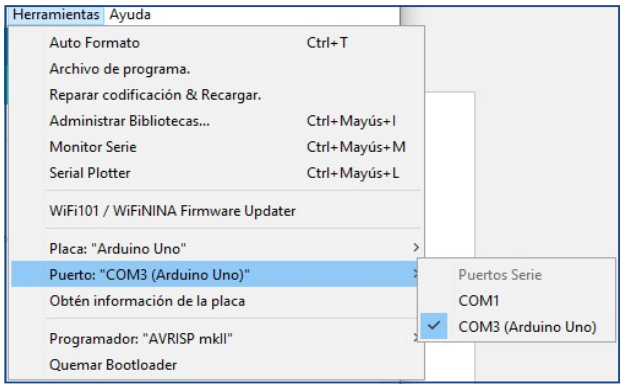


Conectamos el Arduino al Pc y configuramos el programa.

En la pestaña “Herramientas” vamos a la opción “Placa” y ahí elegimos nuestra placa, en este caso tenemos conectado la placa Arduino uno.



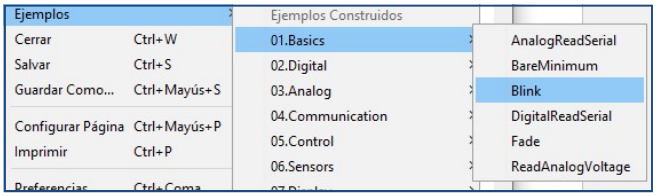
Luego elegimos el puerto de conexión en la misma pestaña de “Herramientas” entramos ala opción “Puerto” y elegimos el indicado.



Este puerto puede cambiar cada vez que abrimos el programa o reiniciamos el PC, tomar eso en cuenta.

Abrir primer ejemplo

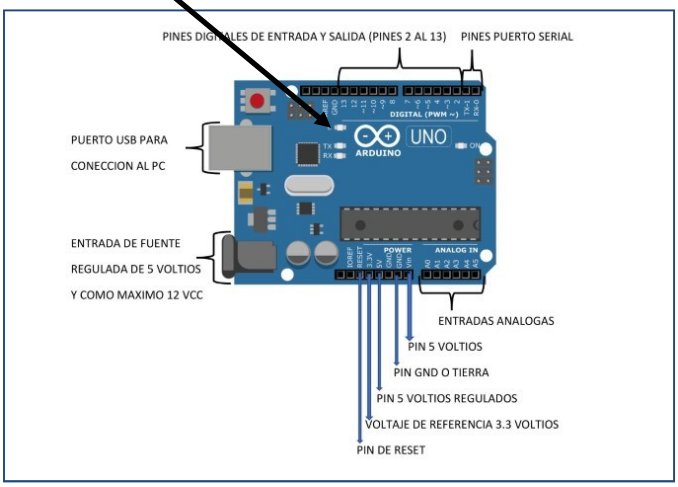
En la pestaña “Archivo” en la opción “Ejemplos” entramos a 01. Basics—Blink



El archivo que abrimos es el siguiente :

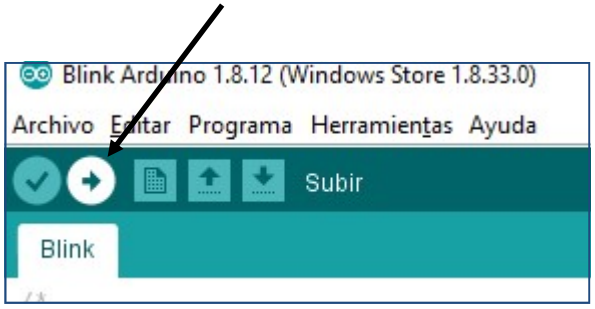
```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);

  delay(1000);                      // wait for a second

}
```

Lo que realiza este código es hacer parpadear el led que se encuentra en la placa en el PIN13

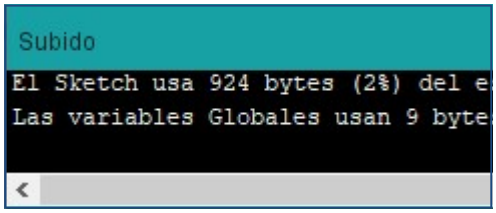


Para subir el programa nuestra placa hacemos lo siguiente:

Presionamos la opción “Subir”



Primero “Compilara” , y luego subirá el código, posterior indicara “Subido” y con eso ya tenemos nuestra placa programada.

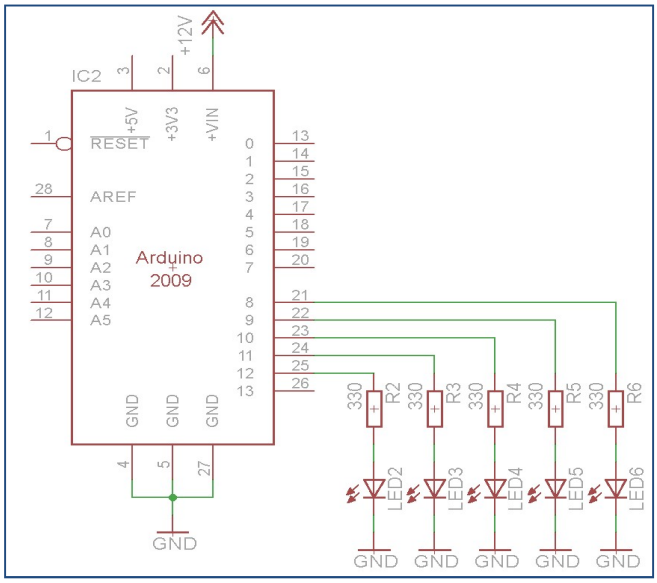


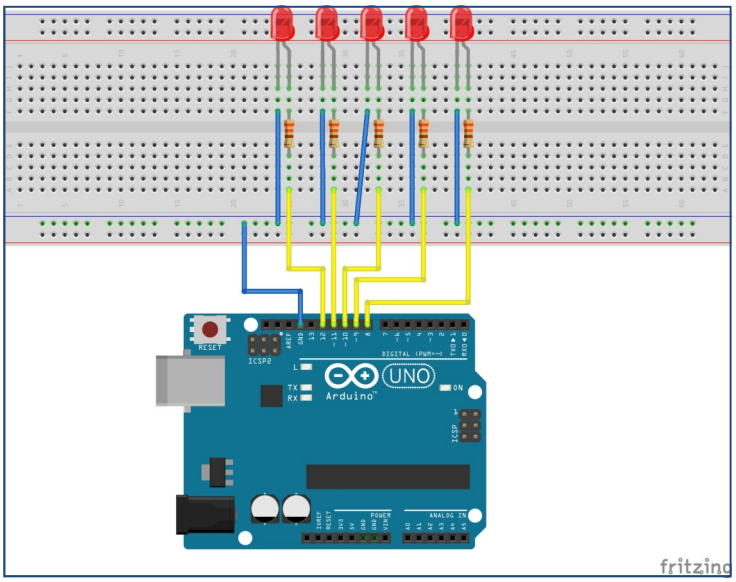
Ya debería estar parpadeando el led integrado, este es el programa mas básico, sirve bastante para probar el estado de nuestras placas en el futuro.

Practica # 1 – Encender LEDS

Conectar Leds a los pines digitales 8, 9, 10, 11 y 12 y hacer que se enciendan de forma secuencial cada segundo.

Luego que se apaguen todos y vuelva a inicial el programa.





Practica # 1 – Código

```
/*  Practica1 Conectar Leds a los pines 12 hasta el 8 y hacer que se enciendan de forma secuencial cada segundo.  Luego que se apaguen todos y vuelva a inicial el programa. */

// la función de configuración se ejecuta una vez al pulsar reinicio
void setup() {

  pinMode(12, OUTPUT);// Inicia el Pin12 como salida
  pinMode(11, OUTPUT);// Inicia el Pin11 como salida
  pinMode(10, OUTPUT);// Inicia el Pin10 como salida
  pinMode(9, OUTPUT);// Inicia el Pin9 como salida
  pinMode(8, OUTPUT);// Inicia el Pin8 como salida
}

// la función de bucle se ejecuta una y otra vez para siempre
void loop() {
  digitalWrite(12, HIGH);   // Encender  LED  (HIGH nivel de voltaje alto 5V)
  delay(1000);              // Retardo 1 segundo
  digitalWrite(11, HIGH);   // Encender  LED  (HIGH nivel de voltaje alto 5V)
  delay(1000);              // Retardo 1 segundo
  digitalWrite(10, HIGH);   // Encender  LED  (HIGH nivel de voltaje alto 5V)
  delay(1000);              // Retardo 1 segundo
  digitalWrite(9, HIGH);   // Encender  LED  (HIGH nivel de voltaje alto 5V)
  delay(1000);              // Retardo 1 segundo
  digitalWrite(8, HIGH);   // Encender  LED  (HIGH nivel de voltaje alto 5V)
  delay(1000);              // Retardo 1 segundo

  digitalWrite(8, LOW);   // Apagar LED
  digitalWrite(9, LOW);   // Apagar LED
  digitalWrite(10, LOW);   // Apagar LED
  digitalWrite(11, LOW);   // Apagar LED
  digitalWrite(12, LOW);   // Apagar LED
  delay(1000);              // Retardo 1 segundo

}
```

setup()

La función setup() se invoca una sola vez al comienzo del programa. Esta función se usa para realizar la configuración inicial, dentro de esta configuración podemos establecer el modo de trabajo de los pines o inicializar la comunicación serie entre otras cosas.

```
void setup() {
  // put your setup code here, to run once:
}
```

loop()

La función loop() es la función principal dentro del programa. Esta función se va a ejecutar continuamente de manera cíclica, ejecutando todas las instrucciones que se encuentren en su interior.

```
void loop() {
  // put your main code here, to run repeatedly:
}
```

Llaves

Las llaves definen el principio y el final de un bloque de instrucciones. Se usan para delimitar el inicio y fin de funciones como setup()o para delimitar el alcance de los bucles y condicionales del programa.

```
funcion()
{
Estamentos o instrucciones;
}
```

Punto y coma

El punto y coma **";"** se utiliza para definir el final de una instrucción y separarla de la siguiente. Si no colocamos punto y coma, el programa va a interpretar mal las instrucciones y se va a producir un error de compilación.

```
DigitalWrite (10,HIGH);
```

El error más común a la hora de programar suele ser olvidar poner punto y coma al final de la instrucción.

Línea de comentarios

La línea de comentarios tienen la misma función que los bloques de comentarios, la única diferencia es que las líneas de comentarios suelen usarse para comentar instrucciones ya que solo afectan a una línea.

```
int  x = 10;      //declara la variable 'x' como tipo entero de
```

Bloque de comentarios /*….*/

Los bloques de comentarios son áreas de texto que nos ayudan a describir o comentar un programa, estos bloques serán ignorados a la hora de compilar el programa en nuestro Arduino.

```
/*El bloque de comentario ayuda al programador
  a describir el programa */
```

Se pueden introducir todas las líneas de texto que se deseen siempre que se encuentren entre los caracteres /*...*/

Se recomienda el uso de bloques de comentarios siempre que se pueda, ya que ayudan a la comprensión del programa a personas ajenas al mismo, además, estos comentarios no van a ocupar espacio de programa, ya que son ignorados a la hora de compilar.

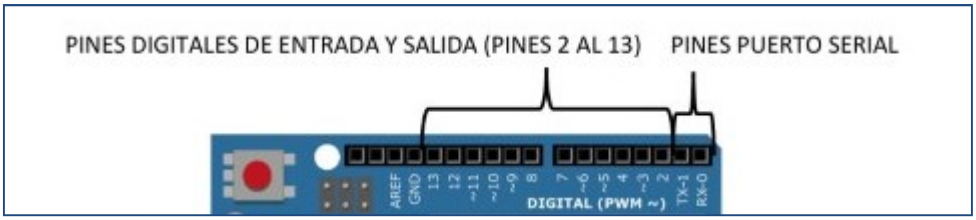
Nombrar pines

Es un habito bastante bueno nombrar los pines que se usaran en el program al inicio para luego solo utilizar su nombre. Por ejemplo nombramos el pin 12 como LED1 y el pin 11 como LED2

```
/* Practica
Nombrar el pin 12 como "LED1" y el pin 11 como "LED2" y hacer que parpadee   LED1   en
forma inversa al LED2. */

int LED1 = 12;  //Pin12 nombrado como LED1
int LED2 = 11;  //Pin11 nombrado como LED2

// la función de configuración se ejecuta una vez al pulsar reinicio
void setup() {

  pinMode(12, OUTPUT);// Inicia el Pin12 como salida
  pinMode(11, OUTPUT);// Inicia el Pin11 como salida
}

// la función de bucle se ejecuta una y otra vez para siempre
void loop() {
  digitalWrite(LED1, HIGH);   // Encender  LED  (HIGH nivel de voltaje alto 5V)
  digitalWrite(LED2, LOW);   // Apagar LED
  delay(1000);                // Retardo 1 segundo

  digitalWrite(LED1, LOW);   // Encender  LED  (HIGH nivel de voltaje alto 5V)
  digitalWrite(LED2, HIGH);   // Apagar LED
  delay(1000);                // Retardo 1 segundo
}
```

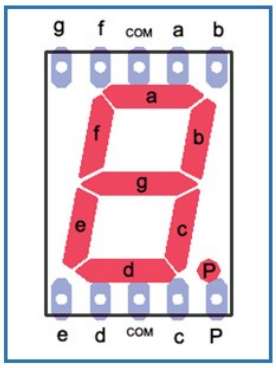
Entradas y salidas digitales

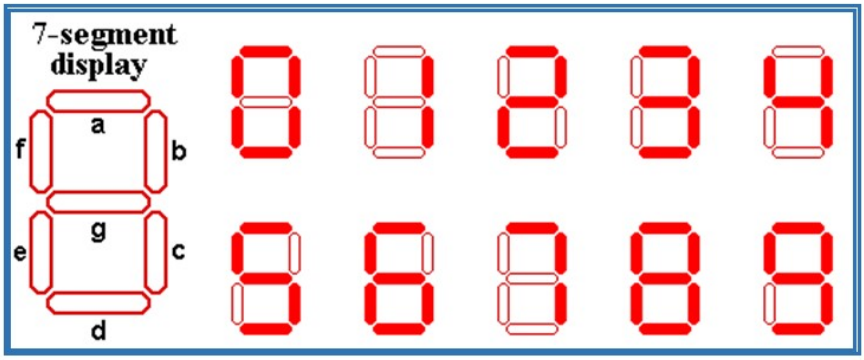
Están situadas en la parte de arriba de la placa, van desde el pin 0 hasta el pin 13, este último pin lleva una resistencia interna incluida. La señal digital puede estar o encendida o apagada (LOW o HIGH). Los pines cero y uno se pueden utilizar para cargar el programa en la placa.

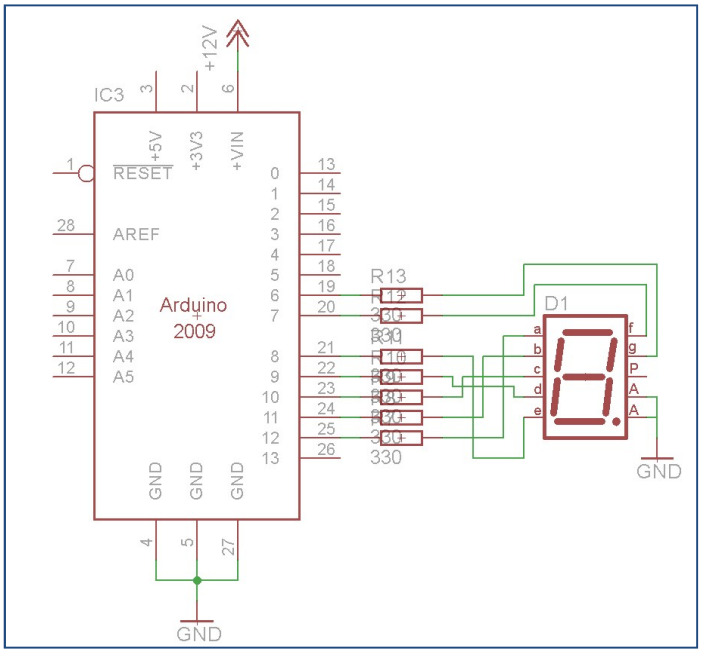


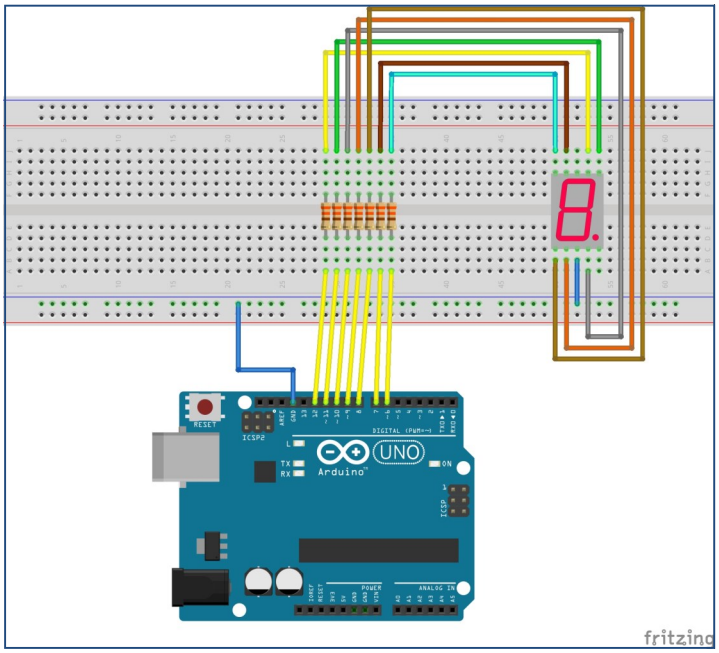
Practica # 2 – Manejo de display catodo común 7 segmentos

Encender el display del numero 0 al 5.









LISTA DE COMPONENTES

MODULOS

M1 = Arduino UNO

SEMICONDUCTORES

LED 1—LED5 = Led común 5mm

D1 = Display Catodo común

RESISTENCIAS

R1 - R7 = 330Ω

VARIOS

Kit de cables de conexión

Protoboard

Practica # 2 – Código

```
/* Practica 2 El problema es encender el display
del 0 al 4 de forma continua con retardo de 1
segundo, utilizar los pines   12…6. . */

int SEG_A = 12;  //Pin12 nombrado como SEG_A del display
int SEG_B = 11;  //Pin10 nombrado como SEG_B del display
int SEG_C = 10;  //Pin11 nombrado como SEG_C del display
int SEG_D = 9;   //Pin9 nombrado como SEG_D del display
int SEG_E = 8;   //Pin8 nombrado como SEG_E del display
int SEG_F = 7;   //Pin7 nombrado como SEG_F del display
int SEG_G = 6;   //Pin6 nombrado como SEG_G del display

// la función de configuración se ejecuta una vez al pulsar reinicio
void setup() {

  pinMode(12, OUTPUT);  // Inicia el Pin12 como salida
  pinMode(11, OUTPUT);  // Inicia el Pin11 como salida
  pinMode(10, OUTPUT);  // Inicia el Pin10 como salida
  pinMode(9, OUTPUT);   // Inicia el Pin9 como salida
  pinMode(8, OUTPUT);   // Inicia el Pin8 como salida
  pinMode(7, OUTPUT);   // Inicia el Pin7 como salida
  pinMode(6, OUTPUT);   // Inicia el Pin6 como salida
}

// la función de bucle se ejecuta una y otra vez para siempre
void loop() {
  // Numero 0
  digitalWrite(SEG_A, HIGH);   // Encender
  digitalWrite(SEG_B, HIGH);   // Encender
  digitalWrite(SEG_C, HIGH);   // Encender
  digitalWrite(SEG_D, HIGH);   // Encender
  digitalWrite(SEG_E, HIGH);   // Encender
  digitalWrite(SEG_F, HIGH);   // Encender
  digitalWrite(SEG_G, LOW);   //  Apagar
  delay(1000);                // Retardo 1 segundo

   // Numero 1
  digitalWrite(SEG_A, LOW);   //  Apagar
  digitalWrite(SEG_B, HIGH);   // Encender
  digitalWrite(SEG_C, HIGH);   // Encender
  digitalWrite(SEG_D, LOW);   //  Apagar
  digitalWrite(SEG_E, LOW);   //  Apagar
  digitalWrite(SEG_F, LOW);   //  Apagar
  digitalWrite(SEG_G, LOW);   //  Apagar
  delay(1000);                // Retardo 1 segundo

    // Numero 2
  digitalWrite(SEG_A, HIGH);   // Encender
  digitalWrite(SEG_B, HIGH);   // Encender
  digitalWrite(SEG_C, LOW);   //  Apagar
  digitalWrite(SEG_D, HIGH);   // Encender
  digitalWrite(SEG_E, HIGH);   // Encender
  digitalWrite(SEG_F, LOW);   //  Apagar
  digitalWrite(SEG_G, HIGH);   // Encender
  delay(1000);                // Retardo 1 segundo
```

```
// Numero 3
digitalWrite(SEG_A, HIGH);   // Encender
digitalWrite(SEG_B, HIGH);   // Encender
digitalWrite(SEG_C, HIGH);   // Encender
digitalWrite(SEG_D, HIGH);   // Encender
digitalWrite(SEG_E, LOW);    //  Apagar
digitalWrite(SEG_F, LOW);    //  Apagar
digitalWrite(SEG_G, HIGH);   // Encender
delay(1000);                 // Retardo 1 segundo

    // Numero 4
digitalWrite(SEG_A, LOW);    // Encender
digitalWrite(SEG_B, HIGH);   // Encender
digitalWrite(SEG_C, HIGH);   // Encender
digitalWrite(SEG_D, LOW);    //  Apagar
digitalWrite(SEG_E, LOW);    // Apagar
digitalWrite(SEG_F, HIGH);   // Encender
digitalWrite(SEG_G, HIGH);   // Encender
delay(1000);                 // Retardo 1 segundo
 }
```

Puerto USB

Este puerto tiene la función de cargar el programa en la memoria del arduino, más aún si se requieren que salgan datos para que sean leídos en un ordenador, este ara la función de lectura y escritura, de allí la importancia del puerto usb.

Fuente de alimentacion del Arduino

La tarjeta puede funcionar con un suministro externo de 6 a 20 voltios. Si se suministra menos de 7 VDC la corriente puede ser inestable. Si se utiliza más de 12 V, el regulador de voltaje se puede sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

En concreto acabamos de analizar el funcionamiento de un arduino uno como son sus voltajes, sus entradas y salidas, recuerden que siendo el arduino un sistema digital, este se le deben respetar sus voltajes y sus retroalimentaciones ya que de lo contrario lo dañara.