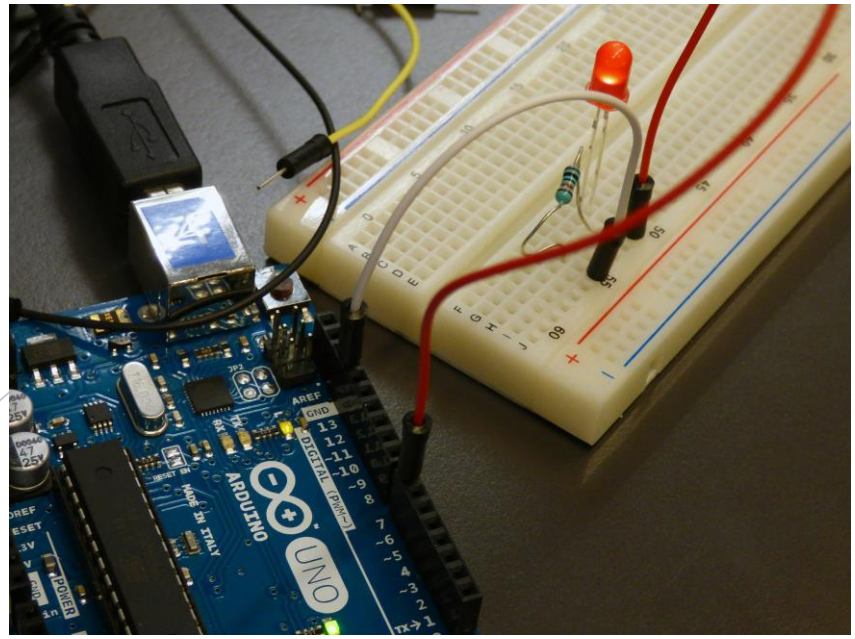
14-6-2020

Los Fundamentos de Arduino

El comienzo de una gran aventura



CREATIVIDAD AHORA

Capítulo 1: Fundamentos de Electrónica

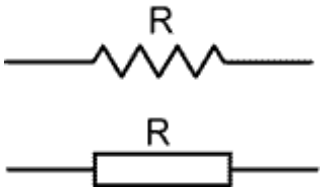
Si bien en esta lectura vamos a aprender Arduino, para ello es necesario tener algunos conocimientos básicos de electrónica y dispositivos electrónicos para poder hacer mas llevadero el aprendizaje, sin embargo, si el lector ya esta familiarizado con la electrónica podrá tranquilamente saltar este capitulo e ir al capitulo 2 donde ya vemos la parte de la programación de Arduino.

En este capitulo repasaremos los dispositivos electrónicos básicos, como resistencias, leds, potenciómetros y Protoboard, estos dispositivos se usarán en los primeros ejemplos por lo que es importante su aprendizaje, especialmente el Protoboard.

Resistencias

Es un dispositivo electrónico que se opone al paso de la corriente eléctrica, por lo que generara una caída tensión entre sus extremos, este dispositivo no tiene polaridad, es decir no tiene parte positiva ni negativa.

El símbolo de la resistencia se muestra a continuación



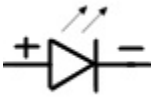
La resistencia tendrá un valor de R, estos identificados por un código de colores, siendo los que usaremos en este curso el color naranja-naranja-marrón que representa una resistencia de 330 Ohmios, también usaremos la resistencia de color marrón-negro-naranja que representa a 10 000 Ohmios, mas conocido como 10K.

330 ohmios 10K ohmios

Leds

Diodo Led es un foco que ilumina (se prende) cuando se polariza correctamente y recibe la corriente necesaria para funcionar, la cual por lo general es alrededor de 15mA, estos diodos leds tienen polaridad, es decir parte positivo y negativo siendo la patita más larga el positivo (+) y la patita más corta el negativo (-).

El símbolo del diodo led es el siguiente, la parte del + es la parte positiva y el – es el negativo.



Físicamente los diodos se ven de la siguiente manera, recordar que la patita mas larga es la parte positiva y la mas corta es la negativa. Los diodos los encontramos de diferentes colores.

Pulsador:

Es básicamente un dispositivo de 2 o 4 pines que tiene un botón en su parte superior, y cuando se presiona el botón une los 2 pines de este, si no se presiona los pines están separados.

El símbolo del pulsador es el mostrado.

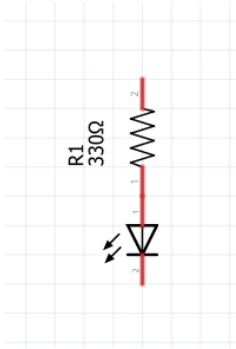
Físicamente se ven de la siguiente forma:

Potenciómetro:

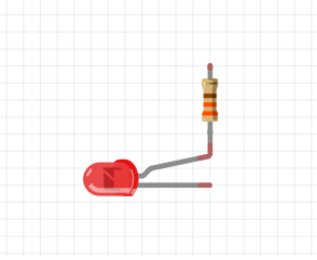
Dispositivo que consta de 3 pines y un girador, se le conoce como resistencia variable, lo que se hace para trabajar con él es energizar sus pines del extremo, uno con 5V y el otro extremo con GND (0 Voltios) y al variar el girador el pin del medio variara entre el voltaje de los extremos, es decir de 0 Voltios a 5 Voltios.

Protoboard

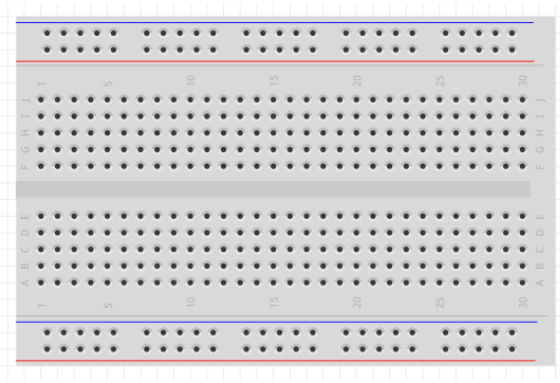
El componente mas importante, es el que nos permitirá hacer las conexiones entre diferentes componentes electrónicos. Para entender su funcionamiento veremos el intento de hacer la siguiente conexión, lo pondremos primero como esquema electrónico y luego físicamente.



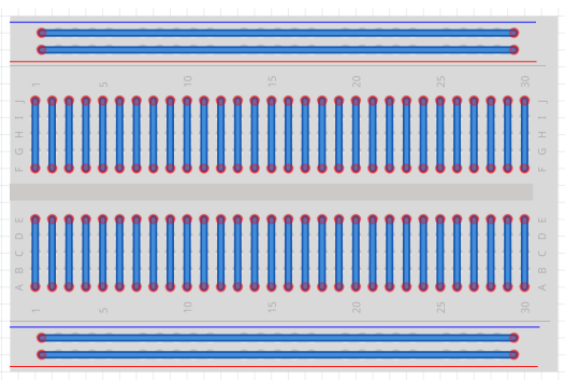
Lo que se pretende es unir una resistencia de 330 Ohmios con un Diodo Led, según el esquemático la parte positiva del Diodo Led (físicamente la patita mas larga) es la que esta unida a la resistencia, físicamente tendríamos lo siguiente:



Sin embargo, hacer esa unión en el aire no es muy practico dado la dificultad que presenta, para hacerlo de manera mas sencilla es el uso de protoboard, que contiene diversidad de agujeros para colocar los componentes pero están unidos internamente de una manera estratégica, básicamente están unidas de la siguiente forma:

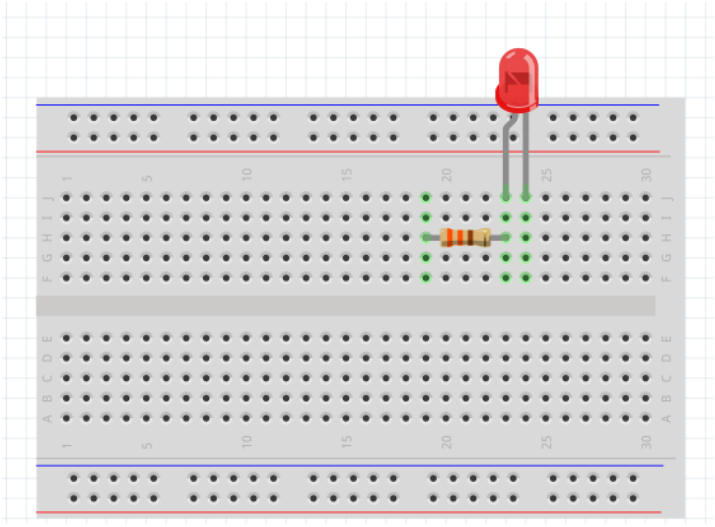


Protoboard Físicamente



Internamente están unidos las líneas

Donde están las líneas todos esos agujeros están unidos, basándonos en eso para hacer la conexión brindada será necesariamente hacer la siguiente conexión, recordar que la parte positiva del led (patita mas larga) es la que esta unida a la resistencia. El otro extremo del led y el otro extremo de la resistencia no indica a donde conectar por lo que en el protoboard podemos conectarlo a cualquier punto.



Capítulo 2: La Plataforma Arduino

Inicialmente diremos que Arduino es una Plataforma que esta basada en un dispositivo programable como un Microcontrolador, eso depende de la placa que usaremos, en esta lectura se vera la placa Arduino UNO, sin embargo, tenemos diferentes modelos siendo los mas populares el Arduino UNO, Arduino Leonardo, Arduino Mega, Arduino DUE entre muchos otros.

Como se menciono en esta lectura estudiaremos el modelo Arduino UNO R3, el cual esta basado en un microcontrolador Atmega 328, por lo que nos interesara conocer primero el funcionamiento de un microcontrolador para ver lo que el Arduino puede hacer:

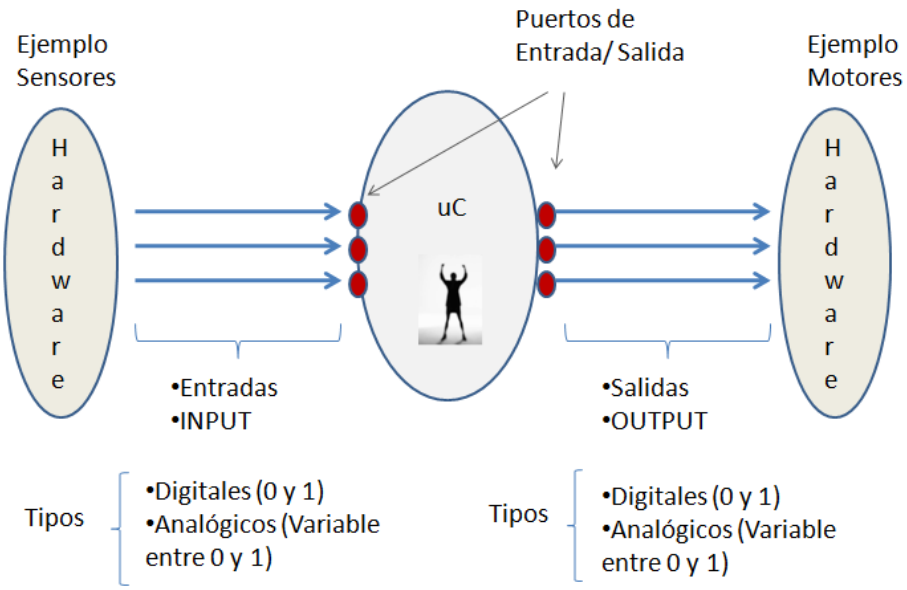
Funcionamiento de un Microcontrolador

Describiremos de la forma más sencilla su funcionamiento, para esto asumiremos que nosotros somos el microcontrolador, dado que nosotros somos los que lo programaremos y decidiremos su actuar.

La forma de trabajo de un microcontrolador (uC) es que recibe información de elementos externos (por ejemplo Sensores), procesa la información recibida y luego envía

un resultado producto de ese procesamiento hacia otros elementos(por ejemplo hacia Motores), la información que procesa puede ser digital(0 y 1) o analógica (un valor variable entre 0 y 1), por donde ingresa la información al microcontrolador se conoce comúnmente como entradas(INPUT) y por donde sale la información se conoce como salidas(OUTPUT).

El siguiente Grafico describe el funcionamiento general de un Microcontrolador cualquiera, notar que los puntos por donde ingresa y sale la información se conoce como puertos de entrada y salida.



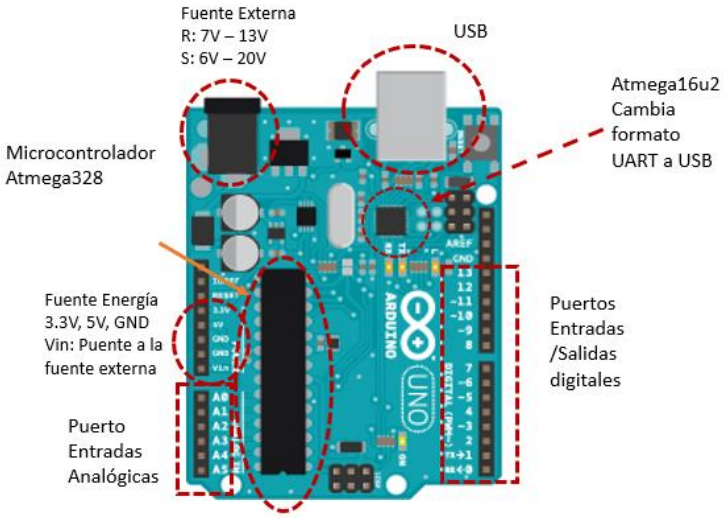
Observación 1: Un Bit es un estado Lógico que puede tener solamente 2 valores, estos valores son el “1” (uno) o el “0” (cero).

Observación 2: Teniendo en cuenta la Observación Anterior un valor Digital está compuesto de Bits por lo que solo tendrá 2 valores (0 y 1), sin embargo, un valor Analógico es un valor Variable entre 0 y 1.

Descripción de la Placa Arduino UNO

Vamos a describir las partes más importantes que posee la placa Arduino UNO, para ellos veamos en el siguiente grafico su descripción.

Podemos ver en el grafico que disponemos de 14 pines (pin 0 al 13) para entradas y salidas digitales, es decir por esos pines ingresara o saldrá información digital (0 y 1), adicionalmente tenemos 6 pines de entrada Analógica (A0 al A5) por lo que por esos pines ingresara la información analógica (un valor variable entre 0 y 1), para energizar el Arduino lo recomendado es una fuente externa de 7V a 13V, sin embargo el regulador de la placa puede soportar de 6V hasta 20V, se energiza la placa por fuente externa o por cable USB.

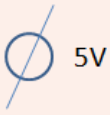


EL Arduino UNO está basado en un Microcontrolador Atmega 328, este microcontrolador no tiene comunicación USB, solamente Serial (UART) por lo que inicialmente no se podría comunicar con la Computadora, sin embargo, la placa incorpora un microcontrolador Atmega16u2 que se encarga de convertir el formato Serial a UART y viceversa, con esto el Arduino ya puede comunicarse con la PC.

Funciones digitales Básicas

Hasta el momento tenemos preparado el terreno para empezar a programar, es el momento de conocer las herramientas (sentencias) con las que vamos a desarrollar el código, empezaremos viendo las que trabajan netamente en los pines digitales (0 al 13).

Es importante en este punto aclarar que el Arduino Maneja Niveles de Voltaje, sin embargo, en Programación nosotros manejamos lógica digital (0 y 1) por lo que la siguiente equivalencia es usada en la interacción de la parte física con la parte lógica.

Lógico	Físico	Símbolo
0	0V , GND, Tierra	
1	5V, fuente	 5V

Notas que las funciones digitales son las sentencias que vamos a usar para desarrollar nuestro programa, estamos comenzando con 4 de ellas que son las más básicas

y 3 de ellas hacen referencia principalmente a funciones digitales, considerar que es importante el manejo de mayúsculas y minúsculas, dado que el código Arduino es sensible a ellos, también notar el punto y coma (;) al final de cada sentencia.

- pinMode(#pin, valor);
  - #pin del Arduino que se va a configurar, varia de 0 a 13.
  - Valor que se va a configurar, puede ser OUTPUT que significa Salida o INPUT que significa Entrada
- digitalWrite(#pin, valor);
  - #pin del Arduino que se va a colocar su valor, varia de 0 a 13.
  - Valor que se va a colocar, puede ser HIGH (el cual significa 1) o LOW (el cual significa 0).
- digitalRead(#pin);
  - #pin del Arduino que se va a leer su estado, esto nos va a devolver un HIGH o un LOW.
- delay(t);
  - tiempo "t" esta expresado en milisegundos, indica la cantidad de tiempo que el programa quedara en stop sin hacer nada.

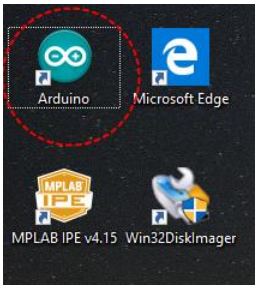
Con estas 4 sentencias pinMode, digitalWrite, digitalRead y delay ejecutaremos los primeros ejemplos para ir familiarizándonos con nuestro código.

Cuerpo del Programa

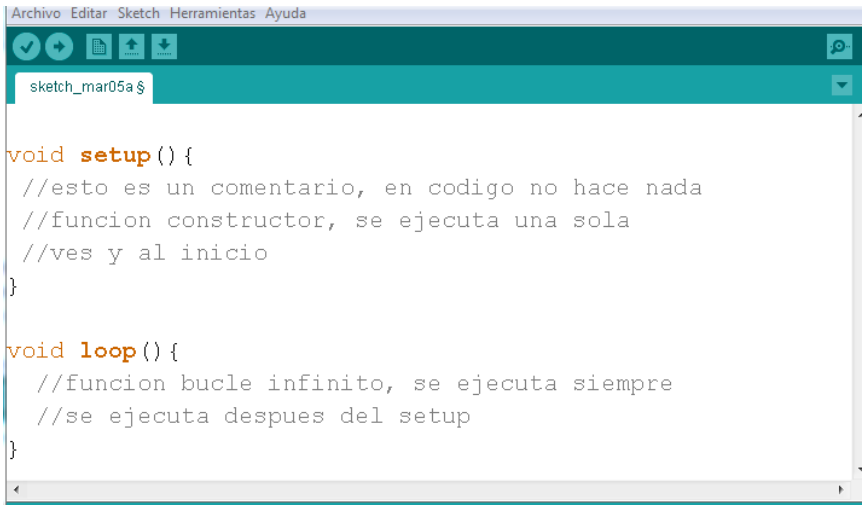
Es el momento de desarrollar y hacer nuestros primeros ejemplos, para ello necesitamos instalar el Arduino el cual se puede descargar de la web oficial de Arduino el cual es el siguiente:

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

Ahí puede descargar e instalar para cualquier sistema Operativo, sea Windows, Linux o Mac, una vez instalado abrir el Arduino el cual tendrá el siguiente icono:

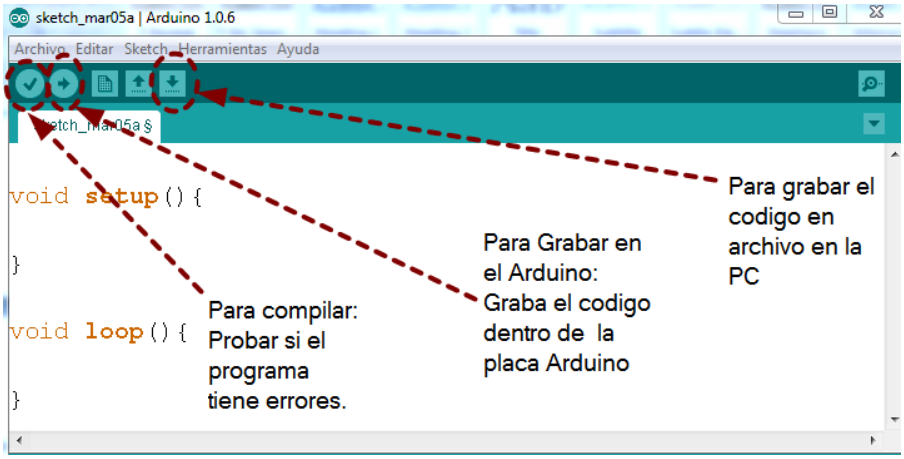


Luego se nos abre la ventana de Programación, en ella podemos observas las 2 funciones básicas que siempre tendrá el código, el void setup y el void loop, el código (sentencias) se colocara dentro de estas funciones, lo que se coloque dentro del setup se ejecutara una sola vez y al inicio, luego de ejecutar el setup se ejecutara el loop, el cual lo hará infinitas veces, en el código para adicionar un comentario es con doble //.



Observación: En la programación de Arduino es sensible a las mayúsculas y minúsculas, hay que tener mucho cuidado al momento de programar.

Botones: A continuación, se muestran las funcionalidades de los Botones:

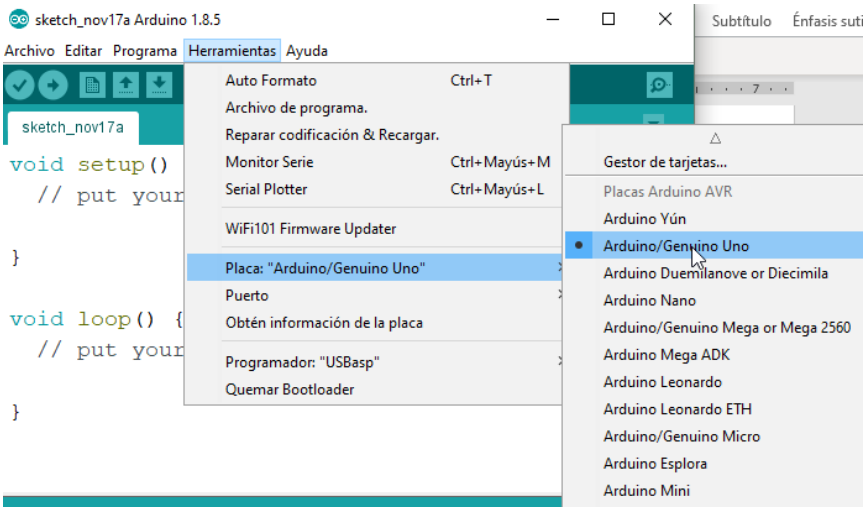


- En el Primer Botón  probamos si nuestro código tiene algún error en la escritura, esto solo es de verificación, no graba el código hecho en ningún Lado.

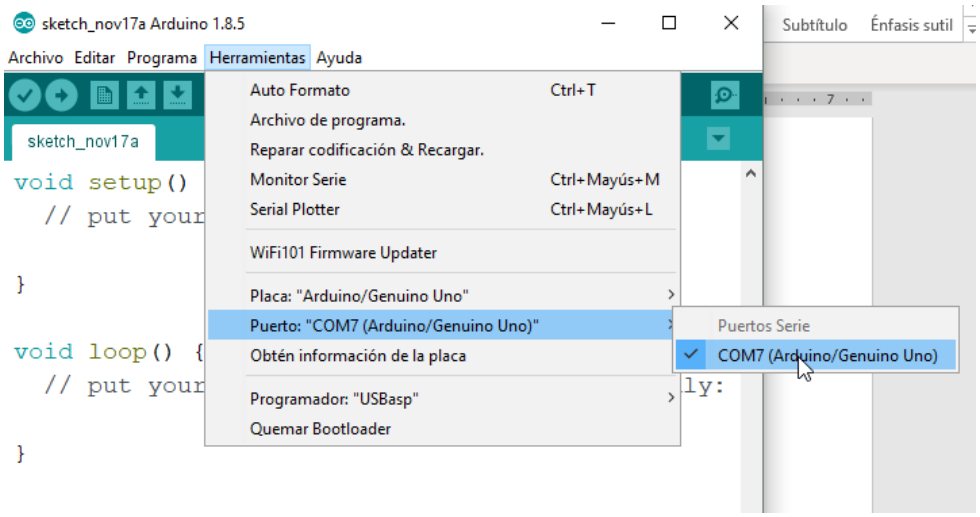
- En el Segundo Botón  grabamos el código hecho dentro de la placa Arduino, con esto Arduino ya estaría haciendo exactamente lo que deseamos que haga según el código.
- En el Tercer Botón  grabamos el código que hemos hecho en una carpeta de la PC, esto con el propósito de luego poder continuar programándolo o hacerle alguna modificación.

Con esto ya tenemos todo lo necesario para ir a programar y probar nuestros ejemplos, para ello conectemos nuestro Arduino por el cable USB a la PC y verifiquemos lo siguiente:

Conectar la Placa Arduino y seleccionar el modelo de Arduino a usar, en nuestro caso será el Arduino UNO como muestra el grafico. Si usted tiene otro modelo como el Arduino Leonardo o Mega escoger el modelo de la lista brindada.

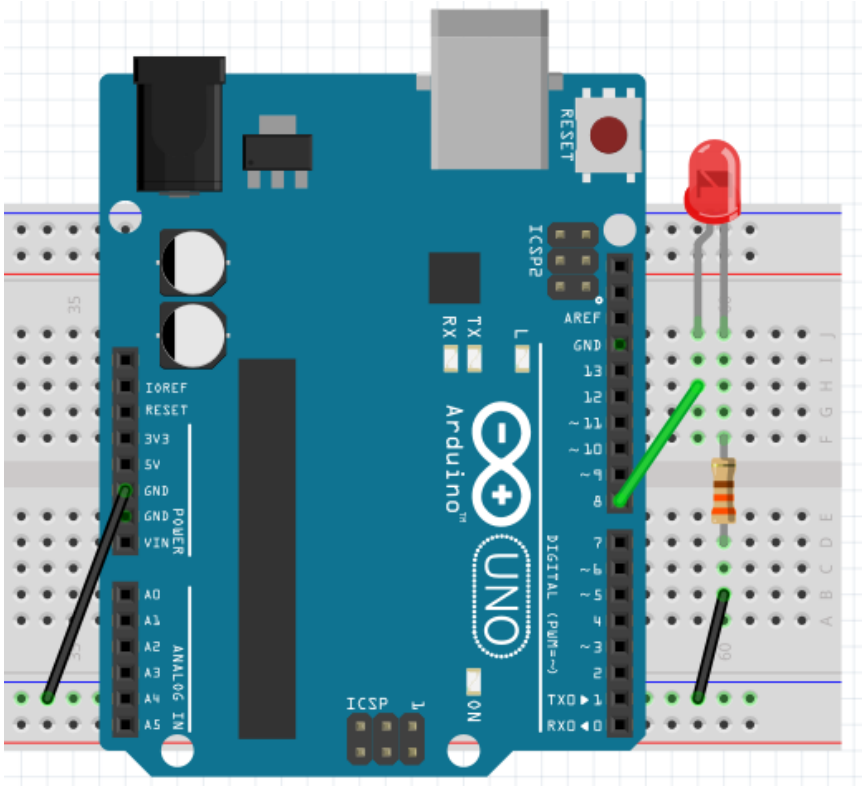


Luego al conectar el Arduino por primera vez (y solo la primera vez) tomara unos segundos para que se instale el driver en la placa, una vez instalado el Arduino obtendrá un Puerto COM de la lista dada, seleccionar el que diga Arduino UNO, si posee varios puertos COM y no esta seguro cual es, puede desconectar el Arduino y fijarse en la lista de Puertos COM, luego conectar y ver cual se adiciona, considerar que la primera vez tomara unos segundos en instalarse el driver y por lo tanto en aparecer el puerto COM del Arduino.

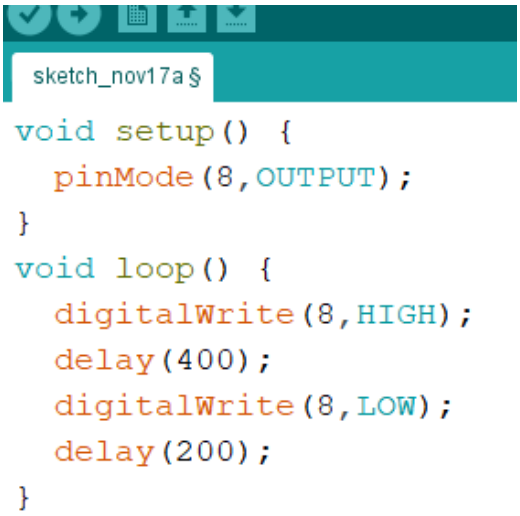


Haciendo estos 2 pasos adicionales ya podemos empezar con los ejemplos, los cuales indicamos a continuación:

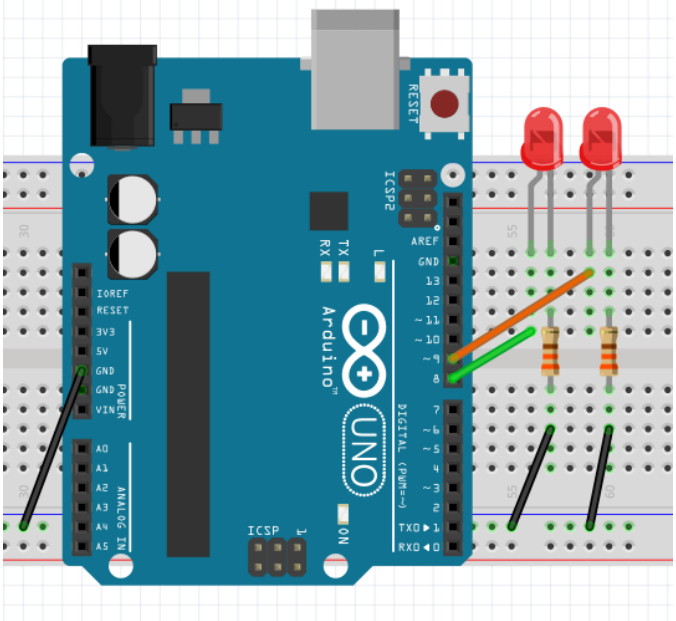
Ejemplo 01: Conectar un Led con una resistencia de 330 al pin 8 del Arduino. Hacer que el Led parpadee infinitamente (prenda y apague) con tiempo de encendido de 400ms y tiempo de apagado de 200ms. Hacemos notar que la parte positiva del led va hacia el pin 8 del Arduino, la parte negativa del led se junta con un extremo de la resistencia, el otro extremo de la resistencia va conectada a Tierra, es decir a un GND del Arduino (el Arduino tiene 3 GNDs).



Solución: A continuación, brindamos el código de la solución, considerar que se borraron los comentarios, una vez terminado el código se da click en compilar , en eso nos pedirá guardar el código, le damos un nombre cualquier (Ejemplo01 por decir), luego le damos click en grabar el código para ya subirlo a la placa , con esto ya el código debería reflejarse en nuestro circuito. Hacer este procedimiento para los demás ejemplos.



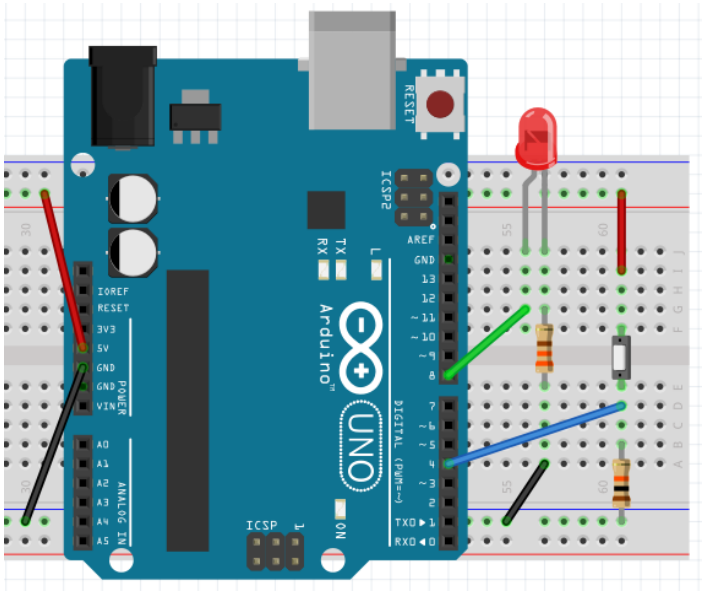
Ejemplo 02: Del Problema anterior adicionar un led con su resistencia al pin 9 del Arduino, el led del pin 8 parpadea infinitamente con tiempo de encendido de 400ms y tiempo de apagado de 200ms, cuando el led del pin 8 este prendido el led del pin 9 debe estar apagado, y cuando el led del pin 8 está apagado, el led del pin 9 debe estar encendido, es decir el pin 9 hace todo lo opuesto al pin 8.



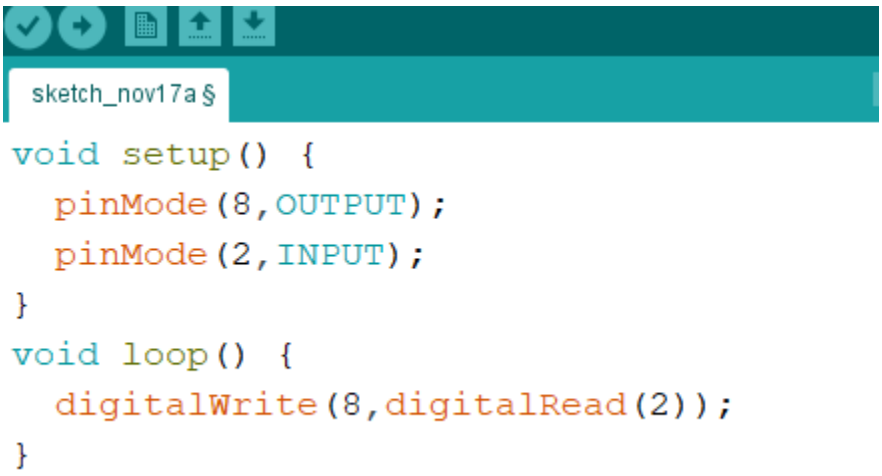
sketch_nov17a §

```
void setup() {
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
}
void loop() {
  digitalWrite(8,HIGH);
  digitalWrite(9,LOW);
  delay(400);
  digitalWrite(8,LOW);
  digitalWrite(9,HIGH);
  delay(200);
}
```

Ejemplo 03: Conectar un Led al pin 8 y un pulsador al pin 2 del Arduino según el grafico. Hacer que cuando este presionado el pulsador el led debe encender y cuando no está presionado el pulsador el led debe estar apagado, es decir apenas presiona led enciende, mantiene presionado sigue encendido y suelta el pulsado el led se apaga.

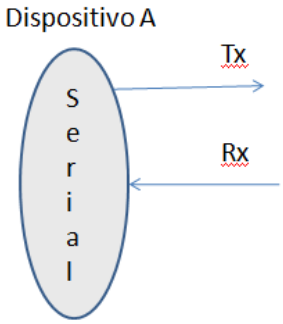


Solución: Notar que el digitalRead(#pin) es un HIGH o LOW, si el pin esta conectado a 5Voltios entonces digitalRead(#pin) = HIGH.

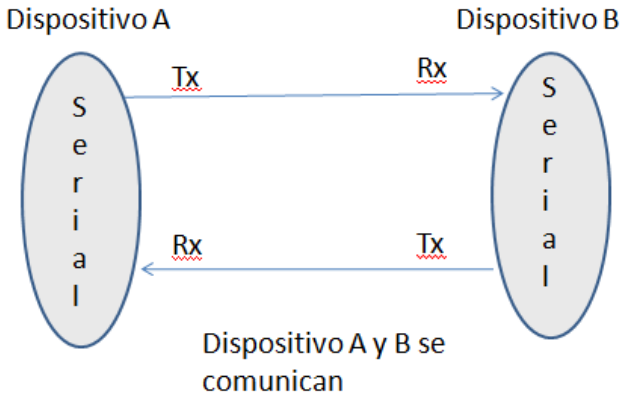


Capítulo 3: Uso del Serial

Así como los seres humanos nos comunicamos mediante un idioma, siendo uno de ellos el castellano, los dispositivos electrónicos también se comunican entre ellos mediante un lenguaje (protocolo), siendo los más populares el I2C, SPI y UART(Serial), en este momento comentaremos sobre el Serial, que es una forma de comunicación de 2 dispositivos electrónicos. La característica de la comunicación Serial es que el dispositivo tiene 2 hilos de comunicación, uno para Transmitir (Tx) y otro para Recibir (Rx).

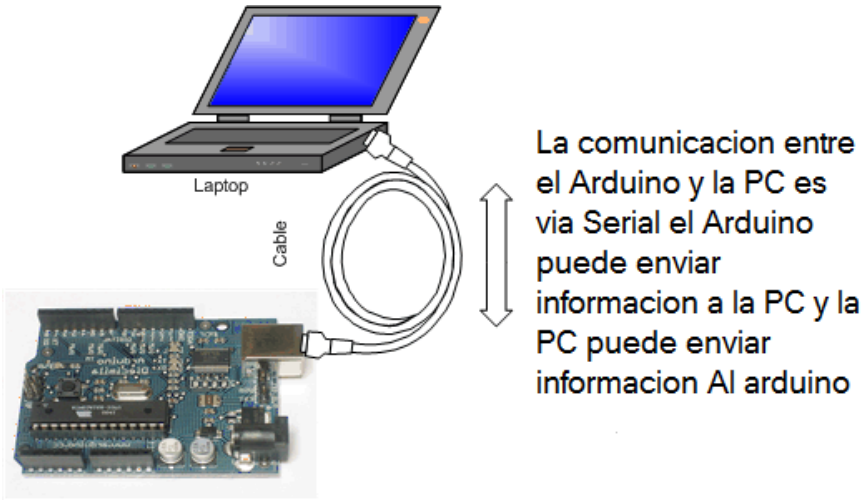


Este será capaz de comunicarse con cualquier otro dispositivo que también soporte comunicación Serial, considerar que el otro dispositivo al soportar comunicación Serial también debe tener 2 Hilos, uno para Tx y otro Rx. El conexionado de 2 dispositivos que soportan Serial debe ser la Tx de uno con la Rx de otro, la siguiente figura muestra el detalle.



Ahora tenemos que el Arduino soporta comunicación Serial (Pin 0 es Rx y pin 1 es Tx), por lo que se puede comunicar con cualquier otro dispositivo vía Serial, uno de estos dispositivos es la Computadora, recordar que el Microcontrolador del Arduino (Atmega328) no soporta comunicación USB por lo que no podría comunicarse con la PC, sin embargo el Arduino tiene el microcontrolador de conversión de formato USB a UART, por lo cual para

el Arduino se estaría comunicando con la PC vía Serial y ya otro elemento se encarga de convertir ese formato.



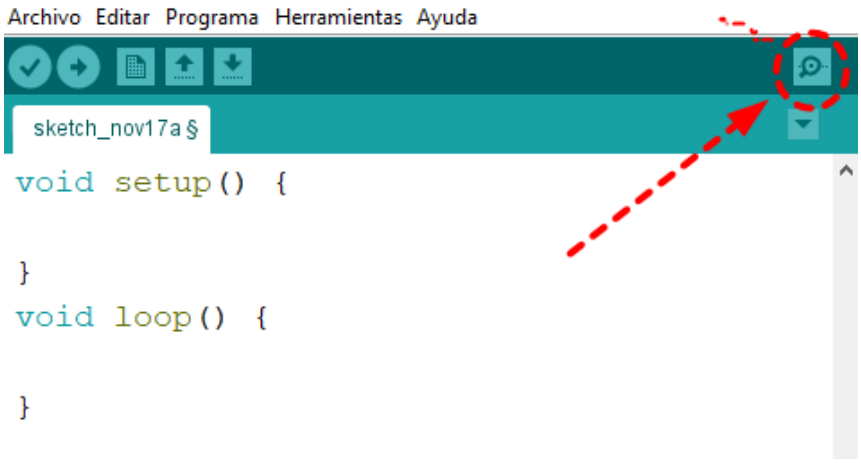
Funciones del Serial

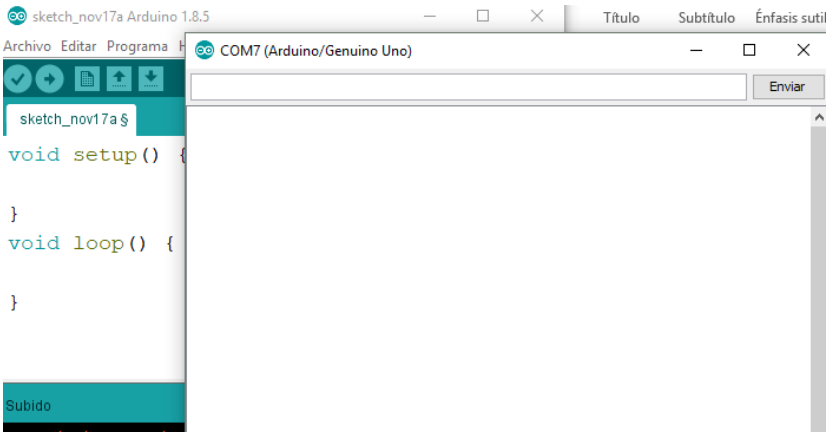
Veamos ahora las sentencias en Arduino para enviar y recibir información a la PC.

- Serial.begin(9600);
  - Con esto iniciamos la comunicación Serial en el Arduino, por defecto la comunicación Serial esta deshabilitada.
- Serial.print(valor);
  - Con esto iniciamos la comunicación Serial en el Arduino, por defecto la comunicación Serial esta deshabilitada.
- Serial.println(valor);
  - Con esto enviamos información del Arduino a la PC con salto de Línea, es decir la siguiente información a enviar será en otra línea diferente a la enviada actualmente.
- Serial.available();
  - Con esto obtenemos el número de bytes o caracteres que han llegado al buffer del Serial y están esperando ser leídos, por ejemplo, si recibimos la palabra “Hola” entonces habrá llegado 4 bytes, por lo que Serial.available() nos devolverá el valor de 4.
- Serial.read();
  - Con esto leemos un byte o carácter del buffer, por lo que si recibimos “Hola” y ejecutamos Serial.read() entonces nos devolverá la letra ‘h’, si lo volvemos a ejecutar Serial.read() nos devolverá la letra o, y así sucesivamente.

Monitor Serial

Hasta el momento tenemos cubierto la parte del Arduino, es decir tenemos las sentencias a usar para poder enviar y recibir información via Serial, sin embargo necesitamos el otro extremo para completar la comunicación, si deseamos comunicarnos con la PC via Serial necesitaremos de un programa en la PC que abra un puerto Serial, que pueda enviar y recibir información via Serial, ese programa tendría que desarrollarse en algún lenguaje de programación en el Serial, sin embargo el IDE de Arduino ya tiene incorporado un programa que hace exactamente eso, ese programa se llama Monitor Serial y se abre haciendo click en el icono mostrado en la figura.





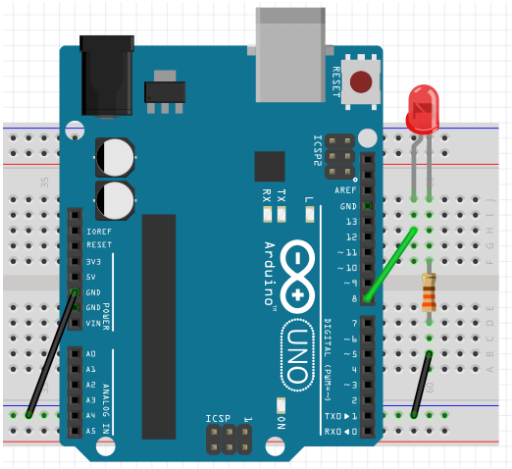
Con esto ya podemos seguir con los ejemplos, sin embargo, usaremos solamente donde el Arduino envía información, dado que para la recepción necesitaremos de una condicional if que veremos más adelante.

Ejemplo 04: Hacer que se imprima en el Monitor Serial del Arduino el mensaje “Hola Arduino” cada 1 segundo. Este ejemplo no requiere esquemático.

sketch_nov17a §

```
void setup() {
  Serial.begin(9600);
}
void loop() {
 Serial.println("Hola Arduino");
 delay(1000);
}
```

Ejemplo 05: Repetir el Ejemplo 01, con el led parpadeando, adicionar cuando el Led Prenda aparezca en el Monitor Serial la palabra “Led Prendido” y cuando el led se apaga “Led Apagado” en el Monitor Serial.



sketch_nov17a §

```
void setup() {
  pinMode(8,OUTPUT);
  Serial.begin(9600);
}
void loop() {
 digitalWrite(8,HIGH);
 Serial.println("Led Prendido");
 delay(400);
 digitalWrite(8,LOW);
 Serial.println("Led Apagado");
 delay(200);
}
```

Capítulo 3: Uso de Variables

Las variables se utilizan para guardar información de manera temporal en la memoria, por ejemplo, si tenemos el número 98 y deseamos almacenarlo en algún lugar, este lugar donde se almacenara se conoce como variable, las variables tienen un tipo dependiendo de la información que van a almacenar, por ejemplo no es lo mismo almacenar un "Hola" que el número 234, tampoco es lo mismo almacenar la letra 'H' que el número 2.84, por ellos existen diferentes tipos de variables dado que existen diferentes tipos de información.

Para usar las variables hay que declararlas, y en el momento de declarar indicarle el tipo de información que guardará, esto se definirá con el tipo de variable a crear.

Declaración de Variables

Las variables se declaran de cualquiera de las 2 siguientes formas:

- tipoVariable nombreVariable;
- tipoVariable nombreVariable = valorInicial;

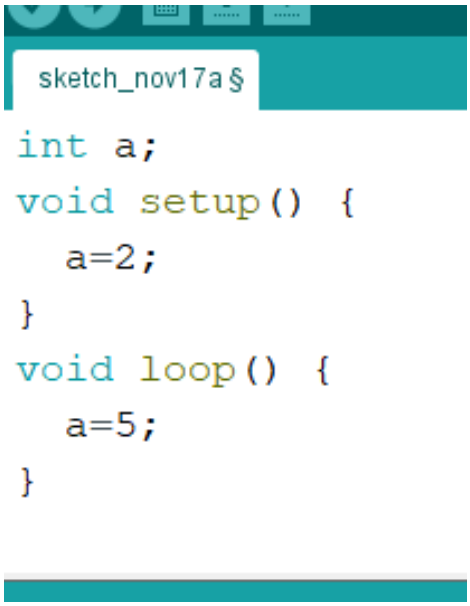
En ambos casos uno define que nombre colocar a la variable, los tipos de Variables se basan según el tipo de información que almacenaran y puede ser cualquier de los siguientes casos.

tipoVariable	Tamaño	Ejemplo de uso	Comentario
byte	8 bits	byte c = 28;	Entero de 8 bits, maximo 255
int	16 bits	int mk = 1023;	Entero de 16 bits
long	32 bits	long p = 203132;	Entero de 32 bits
boolean	8 bits	boolean m = false;	Valor booleano
char	8 bits	char m = 'k'	Para letras
float	32 bits	float v = 2.84;	Para puntos decimales
String		String m = "Hola"	Para cadenas de texto

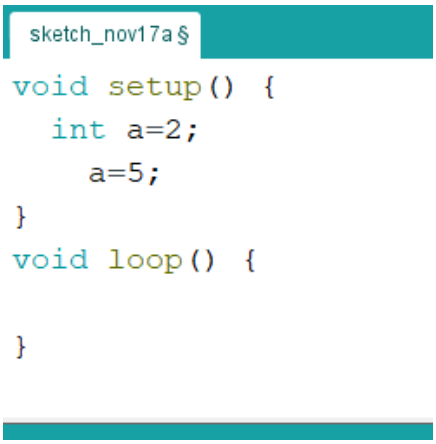
Variables Globales y Locales

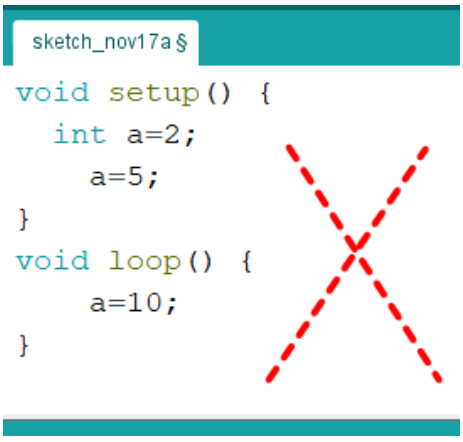
Dependerá de donde definamos la variable para que una variable sea Global o Local, se dirá que es Global cuando la variable existe en toda la hoja de la programación, sin embargo, será Local cuando solo exista dentro de una función como en el setup o en el Loop.

A continuación, mostramos una Variable Global, que se define fuera del Setup y Loop, por lo tanto, existe en toda la hoja y puede usarse en cualquier función.



A continuación, mostramos una variable Local, que esta definido dentro del setup y solamente existe dentro del setup, si intentamos usarlo en el loop nos saldrá un error de que no existe la variable, dado que esta declarado en el setup y solamente existe dentro de ella.





Operaciones con Variables

Con las variables si son del tipo numérico podemos realizar diferentes operaciones, como la suma, resta entre otras, una vez definida la variable podemos usarla en cualquier operación matemáticamente siempre que la variable exista.

Operaciones Comunes:

- Suma: c = a + b;
- Resta: c = a – b;
- Multiplicación: c = a*b;
- División: c= a/b;

Adicionalmente podemos hacer diversas agrupaciones con ellas, siempre se ejecutará lo que este encerrado por paréntesis.

- d = a+b+c;
- e = (a+b) * (c+d);
- e = ((a + b + c) /d) – (d-a) * (a+c);

Ejemplo 06: Hacer que en el Monitor Serial aparezca un contador de 0 1 2 3 4 … y así sucesivamente cada 1 segundo. Este ejemplo no requiere esquemático.

sketch_nov17a §

```
int a = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println(a);
  a=a+1;
  delay(1000);
}
```

Ejemplo07: Repetir el ejemplo Anterior, con la adicional de que el número este precedido por la palabra “Contador =”, es decir los valores a mostrar serán:

Contador = 0
Contador = 1
Contador = 2
Contador = 3
Contador = 4
…

sketch_nov17a §

```
int a = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.print("Contador = ");
  Serial.println(a);
  a=a+1;
  delay(1000);
}
```

Capítulo 5: Función Analógica analogRead()

Arduino posee 6 entradas Analógicas, en esas entradas analógicas puede ingresar un valor variable entre 0 y 1, es decir un valor variable entre 0 Voltios y 5 Voltios. A continuación, describimos la función que hace lectura de la entrada.

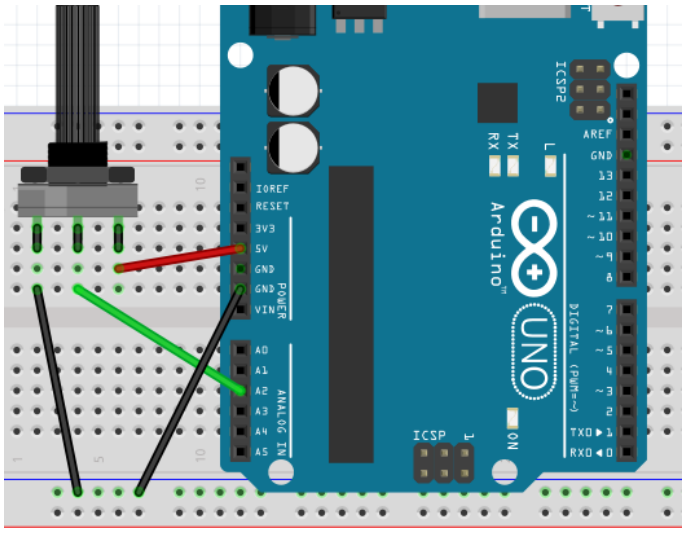
- analogRead(#pin);
    - Donde #pin varía entre 0 a 5 (A0 a A5), en total 6 entradas analógicas.
    - El valor que devuelve esta lectura es un entero que varía entre 0 a 1023.
    - Al hacer esta lectura conviene muchas veces almacenar el valor leído en una variable, por ejemplo en la siguiente sentencia: int a = analogRead(0);
    - En el ejemplo anterior hacemos lectura de la entrada Analógica A0 y el valor leído (varia 0 a 1023) lo almacenamos en la variable a.

La Función Map

La función map nos permite cambiar el rango de una variable a otra variable, es decir si tenemos una variable que tiene un rango, pero quisiéramos otro, la función map nos permite crear una nueva variable con el rango deseado, que depende de la variable inicial

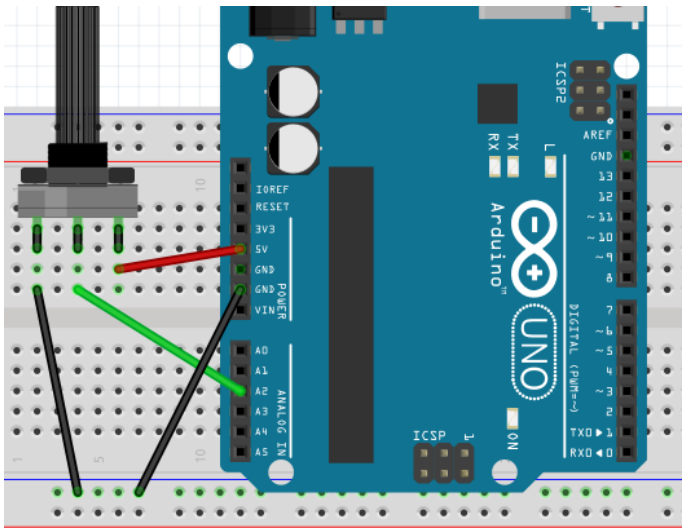
- Int b = map(a,RangoInicialInicial, RangoInicialFinal, RangoFinalInicial, RangoFinalFinal);
    - Donde a es la variable inicial con el rango no deseado.
    - Donde b es la variable con el rango deseado.

Ejemplo 08: Conectar un Potenciómetro al pin Analógico A2 del Arduino, hacer que al girar el potenciómetro aparezca en el Monitor Serial un Numero que varía de 0 a 1023 conforme variamos la posición del Potenciómetro, es decir si el potenciómetro está en un extremo mostrara el Monitor Serial 0, y si el Potenciómetro está en el otro extremo el Monitor Serial mostrara el 1023.




```
sketch_nov17a §
int a;
void setup() {
  Serial.begin(9600);
}
void loop() {
  a = analogRead(2);
  Serial.println(a);
  delay(200);
}
```

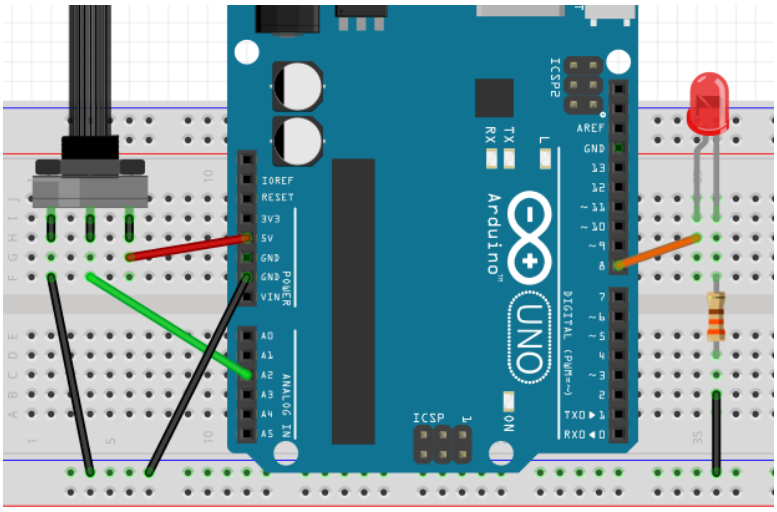
Ejemplo 09: Mismo esquemático que el problema anterior, hacer que en el Monitor Serial muestre los valores de entrada del Potenciómetro, es decir de 0.0 Voltios a 5.0 Voltios, ya no de 0 a 1023, estos valores variaran conforme giremos el potenciómetro.



sketch_nov17a §

```
int a;
void setup() {
  Serial.begin(9600);
}
void loop() {
  a = analogRead(2);
  float v = a*5.0/1023;
  Serial.println(v);
  delay(200);
}
```

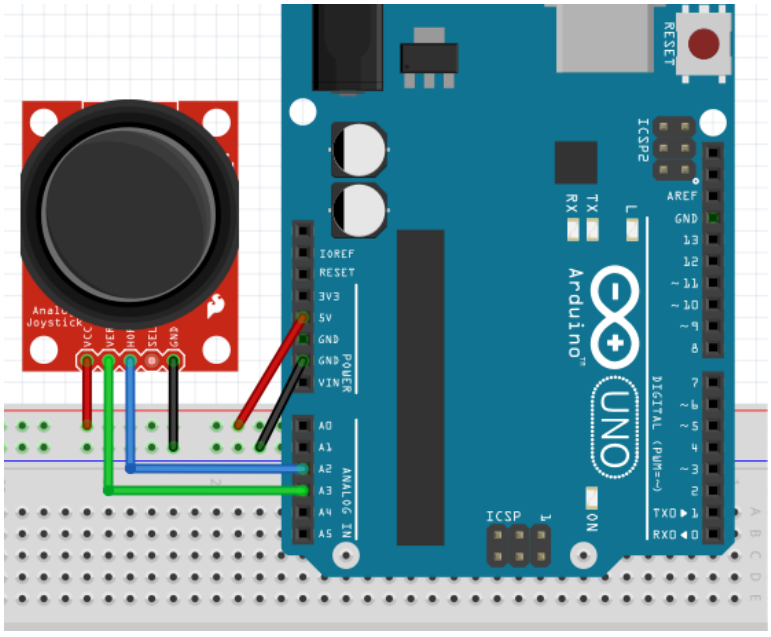
Ejemplo 10: Mismo esquemático que el anterior, pero adicionar un led al pin 8, hacer que el led parpadee infinitamente (similar al problema 01), pero en esta ocasión el tiempo de encendido igual al tiempo de apagado (Ton=Toff) y este tiempo varia con el potenciómetro de 50ms a 250ms, es decir cuando el potenciómetro está en un extremo el Ton es 50ms y cuando está el potenciómetro en el otro extremo el Ton es 250ms.



sketch_nov17a §

```
void setup() {
  Serial.begin(9600);
  pinMode(8,OUTPUT);
}
void loop() {
  int a = analogRead(0);
  int t = map(a,0,1023,50,250);
  digitalWrite(8,HIGH);
  delay(t);
  digitalWrite(8,LOW);
  delay(t);
}
```

Ejemplo11: Conectar el Joystick al Arduino, puede tener varios modelos de Joystick, pero todos tienen 5V, GND, Ver (y) y Hor (x), el eje x e y (Vertical y Horizontal) son salidas analógicas que representarían a los potenciómetros internos del Joystick, conectar estos al pin A2 y A3 e imprimir los valores analógicos en el Monitor Serial.



sketch_nov18a §

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int x = analogRead(0);
  int y = analogRead(1);
  Serial.print("x=");
  Serial.print(x);
  Serial.print(" ,y=");
  Serial.println(y);
  delay(200);
}
```

Capítulo 6: Control de Flujo

En muchas situación en nuestro programa tendremos la necesidad de tomar decisiones, es justamente uno de puntos importantes del funcionamiento de un Microcontrolador que describimos en el capitulo 2, el microcontrolador recibe información, procesa (tomar decisiones) y envía un resultado, el control de flujo se utiliza para la toma de decisiones, tenemos el if, while y for, y todas actúan en base a una condición que se puede cumplir como no, a continuación describimos las condiciones mas usuales que encontraremos.

Condición

Es la comparación de 2 cantidades que solamente produce 2 resultados, la condición es verdad o la condición es falsa, también podemos decir que la condición se cumple o no se cumple, entre estas comparaciones tenemos:

- a > b ; a mayor que b
- a < b ; a menor que b
- a == b ; a idéntico a b (en valor)
- a ¡= b ; a diferente de b (en valor)
- a >= b ; a mayor o igual a b
- a <= b ; a menor o igual a b

Condicional if

En la condicional if tenemos 3 variantes o formas de usarlas, es importante notar que sea cual sea el caso el if siempre abrirá y cerrara llaves ({}), a lo que va dentro de las llaves se le denomina un bloque de código.

if: Analiza una condición, si la condición es verdadera ejecuta un bloque de código (encerrado por llaves), si la condición es falsa no ejecuta el bloque de código.

```
if (condición) {
   //sentencia 1
   //sentencia 2
   //.....
   //sentencia n
}
```

If – else : Analiza una condición, si la condición es verdadera ejecuta el bloque de código (encerrado por llaves), si la condición es falsa ejecuta el bloque de código encerrado por el else (entre llaves), en este caso tenemos 2 bloques de código y siempre se ejecutara uno de ellos, pero nunca los dos.

```
if (condición) {
   //sentencia 1
   //sentencia 2
   //.....
   //sentencia n
} else {
   //sentencia n+1
   //sentencia n+2
   //...
   //sentencia n+m
}
```

else if : Se utiliza para anidar condicionales, es decir hacer un condición con if, y luego de eso analizar otra condición, se hacer múltiples condiciones con el else if, también conocido como if anidado.

```
if (condición){
    .....
} else if(condición 2){
    .....
} else if (condición 3) {
    .....
} else {
    ......
}
```

while :

Otro control de flujo bien útil es el while, este al igual que el if analiza una condición, si la condición de cumple ejecuta lo que esta entre llaves, sino salta y ya no ejecuta, la diferencia entre el if y while es que el if solamente ejecuta lo que esta entre llaves una sola vez, sin embargo el while lo ejecutara constantemente mientras la condición se cumple, es decir si la condición siempre se cumple entonces el while ejecutara lo que está entre llaves infinitamente.

```
while (condición) {
  //sentencia 1
  //sentencia 2
  // sentencia 3
  //....
  //sentencia n
}
```

El while analiza la condición, si es verdad ejecutara todo el bloque de código (sentencia 1 hasta sentencia n), luego vuelve a analizar la condición, si se cumple nuevamente entonces otra vez ejecutara el bloque de código y asi sucesivamente siempre y cuando la condición se cumpla, si la condición ya no se cumple entonces ya no ejecutara el bloque, salta y continua el código.

for:

Otro control de flujo también para bucles al igual que el while muy usado es el for, en realidad si forma de trabajo es similar por no decir idéntico al while, la diferencia radica en que el for usa 3 argumentos, el cual uno de ellos es la condición, adicionalmente tiene la sentencia única inicial así como la sentencia adicional, veamos su estructura para tener una mejor explicación.

```
for (sentencia única inicial ; condición; sentencia adicional) {
  // sentencia 1
  // sentencia 2
  // sentencia 3
  //....
  // sentencia n
}
```

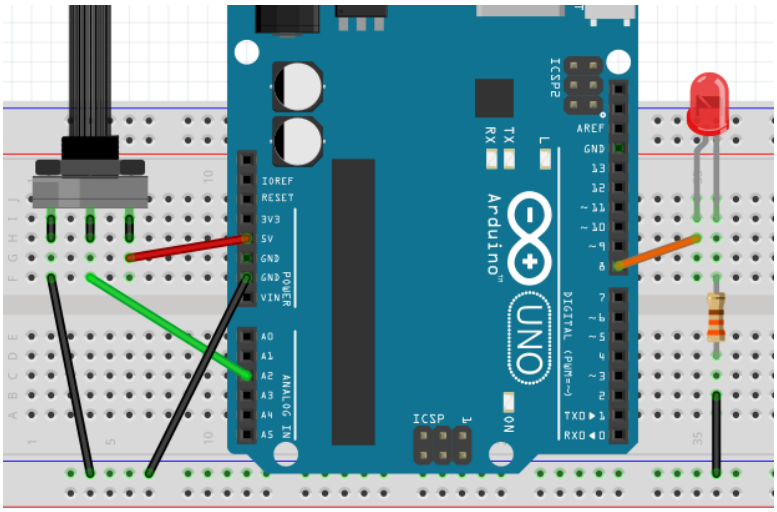
Como se observa, lo primero que hará el for es ejecutar la sentencia única inicial, una sola vez y al comienzo, luego analiza la condición, si la condición es verdad ejecutara el bloque de código (de la sentencia 1 hasta la sentencia n), luego ejecutara la sentencia adicional y ahí termina el primer bucle, luego vuelve a analizar la condición (ya no ejecuta la sentencia única inicial), si la condición se cumple nuevamente ejecuta el bloque de código y después la sentencia adicional, así sucesivamente hasta que la condición no se cumpla, cuando llegue esto saltara y continua el código.

Podemos hacer una analogía entre el while y el for, básicamente lo siguiente demuestra 2 códigos idénticos en funcionalidad.

```
S0;
while (condición) {
  S1;
  S2;
  S3;
   ...
  SN;
  SA;
}
```

```
for (S0 ; condición; SA) {
  S1;
  S2;
  S3;
  ....
  SN;
}
```

Ejemplo 12: El esquemático es el mismo que el ejemplo 10, hacer que al girar de un extremo del potenciómetro a la mitad del potenciómetro el led este apagado, y al girar de la mitad del potenciómetro hacia el otro extremo el led debe estar encendido, es decir girando la mitad led apagado y girando la otra mitad el led encendido.

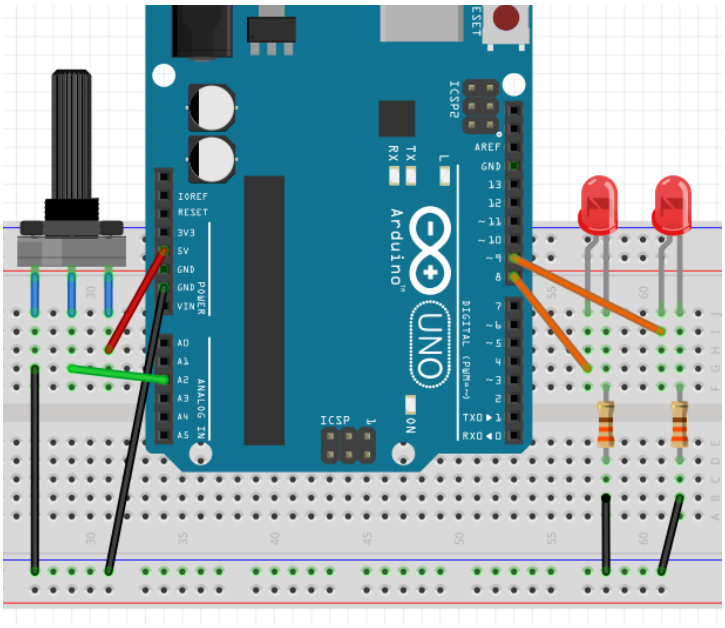


sketch_nov17a §

```
void setup() {
  pinMode(8,OUTPUT);
}
void loop() {
  int a = analogRead(0);
  if(a<512){
    digitalWrite(8,HIGH);
  }else{
    digitalWrite(8,LOW);
  }
}
```

Ejemplo 13: Al esquema anterior adicionar un led al pin 9, hacer que ambos leds parpadeen infinitamente con Ton = Toff y este Ton varia de 50ms a 250ms (similar al ejemplo 10), al girar el potenciómetro de un extremo hacia la mitad varia el led del pin 8 de 50ms a 250ms mientras el led del pin 9 está apagado, al llegar a la mitad del potenciómetro empieza a

variar el pin 9 de 50ms a 250ms hasta el otro extremo del potenciómetro mientras el led del pin 8 está apagado.



sketch_nov1 / a §

```
void setup() {
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
}
void loop() {
  int a = analogRead(0);
  if(a<512){
    int t = map(a,0,511,50,250);
    digitalWrite(8,HIGH);
    delay(t);
    digitalWrite(8,LOW);
    delay(t);
  }else{
    int t = map(a,512,1023,50,250);
    digitalWrite(9,HIGH);
    delay(t);
    digitalWrite(9,LOW);
    delay(t);
  }
}
```