PROGRAMACIÓN ARDUINO

La estructura básica del lenguaje de programación de Arduino es bastante simple y se compone de al menos dos partes o funciones. Estas dos funciones, encierran bloques que contienen declaraciones e instrucciones o estamentos. Ambas funciones son necesarias para que el programa trabaje.

Primero: La función "**setup( )**" es la encargada de contener las configuración (donde decimos que usaremos y como lo usaremos). Esta función se invoca (llamada o usada) una sola vez, cuando el programa empieza. Es cuando se inicializan los modos de trabajo de los PINs. Esta función debe ser incluida en un programa, aunque no haya declaración que ejecutar.

```
void setup( ) {
   Estamentos;
}
void loop( ) {
   Estamentos;
}
```

Segundo: la función "**loop( )**" es la que contienen el programa que se ejecutará cíclicamente (de ahí el termino loop -> bucle). Lo que posibilita que el programa este respondiendo continuamente ante los eventos que se produzcan en la tarjeta

FUNCIONES

Una función es un **bloque de código (conjunto de estamentos)** al que se le da un nombre. Este bloque de código es ejecutado cuando se llama a la función (se lo llama por el nombre). Como recién vimos, "**setup( )**" y "**loop( )**" son funciones.

El usuario (nosotros) puede escribir funciones para realizar tareas repetitivas y aportar claridad o reducir el tamaño de un programa. Las funciones pueden o no devolver algún dato o resultado. Para el caso de las dos primeras funciones que hemos analizado, "**setup( )**" y "**loop( )**", el tipo de dato o resultado devuelto es "void". Esto es el tipo de dato nulo o inexistente. Es decir, no devuelven nada.

Existen otros tipos de datos (por ejemplo int) pero esto lo veremos más adelante, cuando sea necesario usarlos. Resumiendo, si la función no devuelve ningún valor entonces se colocará delante la palabra "void".

LA LLAVES { }

En la programación que estamos estudiando, las llaves sirven para definir el principio y el final de un bloque de instrucciones. Podemos ver un ejemplo de esto en las funciones "setup( )" y "loop( )" que ya hemos visto. Una llave de apertura "{", siempre debe tener su complemento, una llave de cierre "}" cuando termina el bloque. Si no es así el programa dará errores.

El entorno de programación de Arduino, incluye una herramienta de gran utilidad. Para comprobar el complemento de una llave (ya sea que abre o que cierra) sólo tiene que hacer click en la llave y de inmediato, se marca la correspondiente llave complementaria de ese bloque.

EL PUNTO Y COMA ";"

El punto y coma se utiliza para dar por terminada una instrucción, y separar instrucciones en el lenguaje de programación de Arduino (Al igual que otros leguajes de programación).

Nota: Olvidarse de poner fin a una línea con un punto y coma se traducirá en un error de compilación. El texto de error no siempre es obvio, se referirá a la falta de una coma, o puede que no. Si se produce un error raro y de difícil detección lo primero que debemos hacer es comprobar que los puntos y comas están colocados al final de las instrucciones.

FUNCIONES QUE UTILIZAREMOS EN NUESTRO PRIMER PROGRAMA.

- **pinMode(pin, modo)**: Es usada dentro de "**setup( )**" para configurar el comportamiento de un PIN acorde a lo que se requiera en el programa: INPUT o OUTPUT. Por ejemplo: **pinMode(6 , OUTPUT);** este comando configurará el PIN número 6 como de salida.
- **digitalWrite(pin, valor)**: este es un comando que introduce un nivel alto (HIGH) o bajo (LOW) es decir, prende o Apaga, el PIN digital especificado. Por ejemplo, **digitalWrite(6, HIGH);** este comando emitirá por el PIN numero 6 un pulso de corriente. Es decir lo prende.

- **delay(tiempo):** inmoviliza y detiene por el lapso de tiempo especificado (en milisegundos) la ejecución del programa, manteniendo el estado del programa.

PROYECTOS CON LED

1- Prendido y apagado de un LED.

Hacer un programa y circuito, para que prenda y apague un LED a intervalos regulares. Usar el PIN 13 (Podría haberse elegido cualquiera de los otros PINES disponibles en la placa).

(Programa "001_Led_01")

```
1  void setup( ) {
2    pinMode(13, OUTPUT);
3  }
-
4  void loop( ) {
5    digitalWrite(13,HIGH);
6    delay(500);
7    digitalWrite(13,LOW);
8    delay(500);
9  }
```

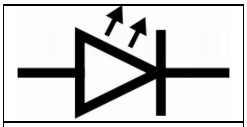
IMPORTANTE SABER: Un Pin, es una salida de la Placa Arduino, donde conectamos (enchufamos) un cable del circuito. En este ejemplo usamos dos pines de la Placa el GND (Negativo) y el Pin 13.

Explicación, Línea Por Línea	
1	Comienza función de Configuración
2	En esta línea se configura en que modo será usado el PIN. El PIN 13 será usado como una SALIDA DIGITAL.
3	Llave que finaliza la función de Configuración
4	Cabecera e inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
5	Enciende el LED conectado al PIN 13
6	Detiene el programa por 500 milisegundos – Mientras el LED esta Prendido.
7	Apaga el LED conectado al PIN 13
8	Detiene el programa por 500 milisegundos – Mientras el LED esta Apagado.
9	Llave que marca el final de la función "**loop( )**". Que comenzó en la línea 4

Ahora hay que construir el circuito, y para esto, primero deberemos ver unas cosas muy simples.. Así que a Leer un poquito.. ¡

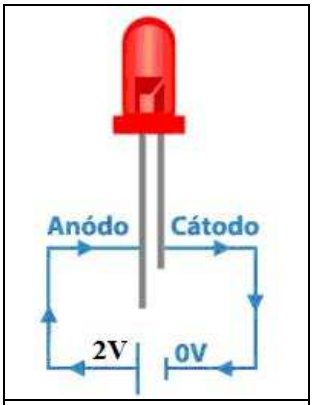
LED, LO QUE NECESITAS SABER.

LED: Sigla de la expresión inglesa *light-emitting diode*, 'diodo emisor de luz'. Este tipo de componentes, es muy particular, ya que solo dejan pasar la electricidad en un sentido.



Símbolo

Pero ¿cuál es ese sentido? Es muy sencillo y con la práctica, te acostumbrarás a utilizarlos. Los LED tienen una pata más larga que otra, esa pata larga, se conoce como **ánodo** (el polo positivo). Es por donde tiene que entrar la corriente. La patilla más pequeña se conoce como **cátodo** (polo negativo) y es por donde saldrá la corriente de electrones.

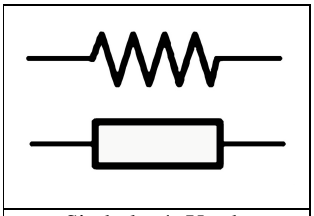


Polaridad

Teniendo claro esto ya podemos conectar LED a nuestro circuito. Ahora solo nos hace falta saber que resistencia poner. Para ello debemos hacer uso de la

Ley de Ohm. Esta Ley es la base fundamental sobre la que debemos construir nuestros circuitos.

RESISTENCIAS



Simbología Usada

Para conversar en clase: ¿Por qué debemos conectar una resistencia? Siempre tenemos que mirar la hoja de características técnicas de los componentes. Para un LED típico de 500 mm, el voltaje de operación está entre 1,8 V y 2,2 V. Pero nuestra fuente tiene 5V. Entonces debemos disminuí el voltaje que le llega, caso contrario podemos dañar el LED o como mínimo, disminuir la vida útil del componente. **Para esto es que usaremos las resistencias.** Por Ahora, Solo tenemos resistencias de un solo valor para usar con los LEDs. Así que no hace falta realizar el calculo, pero en breve si y para cada caso particular.

Imagen de una resistencia

CIRCUITO PARA NUESTRO PROYECTO

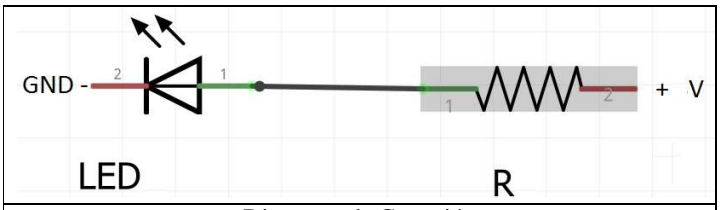


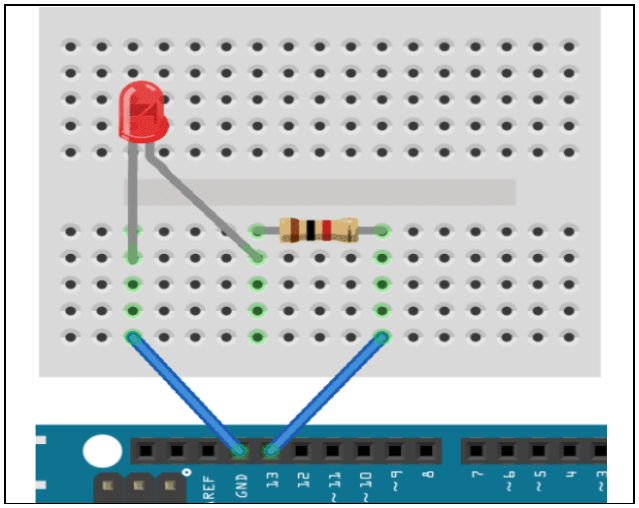
Diagrama de Conexión

Para irnos acostumbrando a esta muy sencilla simbología, veremos el circuito que usaremos con nuestro primer programa. Como puede apreciar es muy fácil de entender y construir. Ahora solo queda construir sobre la placa donde haremos nuestras pruebas (Protoboard). **IMPORTANTE:** GND es el PIN considerado como Negativo o Masa en la Placa Arduino. Si buscamos veremos que hay uno o varios pines así identificados (La cantidad dependerá del modelo de placa que estemos analizando).

Lista de Materiales

- 1 Placa Protoboard. (Ver apéndice A - Placa Protoboard)
- 1 Led.
- 1 Resistencia de 470Ω.
- 2 Cables Macho-Macho.
- Placa Arduino.
- Cable USB.

Recordatorio: Un Pin, es una salida de la Placa Arduino, donde conectamos (enchufamos) un cable del circuito. En este ejemplo usamos dos pines de la Placa el GND y el 13.



Muy Importante: Recordar que Los LED tienen una pata más larga que otra, esa pata larga, se conoce como **ánodo** (el polo positivo). Es por donde tiene que entrar la corriente. La pata más pequeña se conoce como **cátodo** (polo negativo).

Un Nuevo Concepto: Variables.

Técnicamente, una **variable** es un espacio en la memoria del ordenador, reservado para almacenar un dato determinado. Este dato debe ser de un "**tipo**" conocido y que el lenguaje de **programación** va soportar. Para declarar una variable siempre es necesario saber que tipo de dato que va a contener.

Lo anterior, dicho de otra manera: Una variable, puede ser pensada como una cajita del tamaño justo, para guardar algo que es necesario mantener guardado.

Una Variable, debe tener un nombre, así podremos identificarla, y ese nombre debe comenzar con una letra. Entonces podremos considerar nombres correctos para una variable:

A	PIN	SUMA	Resultado	Valor	Etc

Con respecto a los tipos de Variables, es decir que tipos de datos podemos guardad dentro de ellas:

Tipo de Dato	Comentario	Declarar una Variable
boolean	Tipo de datos lógico. Una Variable declarada de este tipo de Datos solo pueden tomar dos Valores "**TRUE**" o "**FALSE**"	bolean a;
byte	Byte almacena un valor numérico de 8 bits sin decimales. Tienen un rango entre 0 y 255	byte d;
char	**----**	char h;
class	Para definir una Clase de datos (Usada en la Programación Orientada a Objetos "POO")	**---**
double	**---**	double f;

float	El formato de dato del tipo "punto flotante" "float" se aplica a los números con decimales. Los números de punto flotante tienen una mayor resolución que los de 32 bits con un rango comprendido 3.4028235E +38 a +38-3.4028235E.
	float unaVariable = 3.14; // declara 'unaVariable' como tipo flotante
	Nota: Los números de punto flotante no son exactos, y pueden producir resultados extraños en las comparaciones. Los cálculos matemáticos de punto flotante son también mucho más lentos que los del tipo de números enteros, por lo que debe evitarse su uso si es posible.
		float c;

int	Los enteros son un tipo de dato primario, que almacenan valores numéricos de 16 bits sin decimales, comprendidos en el rango 32,767 a -32,768. **int unaVariable = 1500; // 'unaVariable' es variable del tipo entero** **Nota:** Las variables de tipo entero "int" pueden sobrepasar su valor máximo o mínimo como consecuencia de una operación. Por ejemplo, si x = 32767 y una posterior declaración agrega 1 a x, "**x = x + 1**" entonces el valor se x pasará a ser -32.768. (algo así como que el valor da la vuelta)	int a;
long	El formato de variable numérica de tipo extendido "long" se refiere a números enteros (tipo 32 bits) sin decimales que se encuentran dentro del rango -2147483648 a 2147483647.	long b;
uint8_t	Es un tipo de dato entero, cuyo valor puede oscilar entre 0 y 255	uint8_t c;
uint16_t	Es un tipo de dato entero, cuyo valor puede oscilar entre 0 y 32768	uint16_t d;
void	Tipo Nulo. Se usa para decir que una función no retorna resultado o que recibirá NADA.	--

IMPORTANTE: Todas las variables tienen que declararse antes de que puedan ser utilizadas. Para declarar una variable se comienza por definir su tipo como int (entero), long (largo), float (coma flotante), etc, asignándoles siempre un nombre, y, opcionalmente, un valor inicial. Esto sólo debe hacerse una vez en un programa, pero a una variable se le puede cambiar el valor cualquier momento que sea necesario hacerlo.

Variable global es aquella que puede ser vista y utilizada por cualquier función y estamento de un programa. Esta variable se declara al comienzo del programa, antes de **setup( )** o en una librería (tema que veremos un poco mas adelante).

Variable local es aquella que se define dentro de una función o como parte de un bloque de programa, por ejemplo dentro de un bucle. Sólo es visible y puede utilizarse dentro de la función o bloque en la que se declaró.

2- Prendido y Apagado de un LED – USANDO VARIABLES

Modificamos el programa anterior, y esta vez usamos una variable para identificar el PIN al que conectaremos nuestro LED. Esto nos permitirá cambiar el PIN en uso casi en forma instantánea, simplemente cambiando el número en un solo lugar.

(Programa "001_Led_02")

```
1   int ledPin = 13;
-
2   void setup( ) {
3     pinMode(ledPin, OUTPUT);
4   }
5   void loop( ) {
6     digitalWrite(ledPin, HIGH);
7     delay(500);
8     digitalWrite(ledPin, LOW);
9     delay(500);
10  }
```

El Circuito y Conexiones son iguales al Problema Anterior.

Lista de Materiales

- 1 Placa Protoboard.
- 1 Led.
- 1 Resistencia de 470Ω.
- 2 Cables Macho-Macho.
- Placa Arduino.
- Cable USB.

Explicación, Línea Por Línea

1	Declaración de una variable del tipo entera (int) a la que llamamos "**ledPin**" y le asignamos el número 5, que luego haremos corresponder con el PIN 13 – (Podría haber sido cualquier otro valor que se corresponda con un PIN de la placa).
2	Comienza función de Configuración
3	En esta línea se configura en que modo será usado el PIN. Para identificar cual es el PIN configurados usaremos el número que indique la variable "**ledPin**". En este ejemplo la variable contiene el valor 13. Entonces el PIN 13 será usado como una SALIDA DIGITAL.
4	Llave que finaliza la función de Configuración
5	Cabecera e inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
6	Enciende el LED conectado al PIN indicado por la variable "**ledPin**" que para nuestro ejemplo es 13.
7	Detiene el programa por 500 milisegundos – Mientras el LED esta Prendido.
8	Apaga el LED conectado al PIN indicado por la variable "**ledPin**" que para nuestro ejemplo es 13.
9	Detiene el programa por 500 milisegundos – Mientras el LED esta Apagado.
10	Llave que marca el final de la función "**loop( )**". Que comenzó en la línea 5

No hace falta construir un nuevo circuito, ya que usaremos nuevamente el que construimos para nuestro primer ejemplo.

COMO INSERTAR COMENTARIOS DENTRO DEL CÓDIGO

Este lenguaje de programación, al igual que otros, nos permite escribir comentarios en el Código, que no serán tenidos en cuenta, pero a nosotros como programadores, nos ayudan a entender lo que escribimos, sobre todo cuando re-leemos un código después de algún tiempo y no recordamos lo que hicimos.
Existen dos formas de escribir comentarios.

El primer tipo de comentario inicia al colocar dos barras // y comenzamos a escribir el comentario que termina cuando cambiamos de renglón.

El segundo tipo de Comentario, es el que comienza con /* y continua, sin importar cuantos renglones escriba, hasta que lo cierre con los caracteres */

Resumiendo: comentarios, son áreas de texto ignorados por el programa que se utilizan para las descripciones del código o comentarios que ayudan a comprender el programa.

Ejemplo:

```
// Este es un comentarios del primer tipo. Abarca TODO el renglón…
/* Este es un bloque de comentario, correspondiente al segundo tipo
no se debe olvidar cerrar los comentarios, porque …….
estos deben estar equilibrados */
```

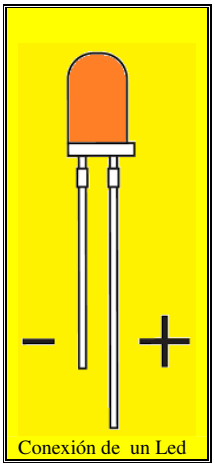
Importante: Dentro de una misma línea de un bloque de comentarios no se puede escribir otro bloque de comentarios (usando /* .. */)

3- Prender 2 LED – Alternando el prendido y apagado.

Pender dos LED alternándolos cada medio segundo - Cuando uno se prende el otro se apaga.

(Programa: "001_Led_03")

```
1   int PinLed_01 = 12;  // Declaración Primer Variable y guardamos el valor 12.
2   int PinLed_02 = 13;  // Declaración Segunda Variable y guardamos el valor 13.
-
3   void setup( ) { // Acá comienza la Función setup( )
4   pinMode(PinLed_01, OUTPUT);
5   pinMode(PinLed_02, OUTPUT);
6   }
-
7   void loop( ) {  // Acá comienza la Función loop( )
8    digitalWrite(PinLed_01, HIGH);
9    digitalWrite(PinLed_02, LOW);
10   delay(500);
-
11   digitalWrite(PinLed_01, LOW);
12   digitalWrite(PinLed_02, HIGH);
13   delay(500);
14  }
```

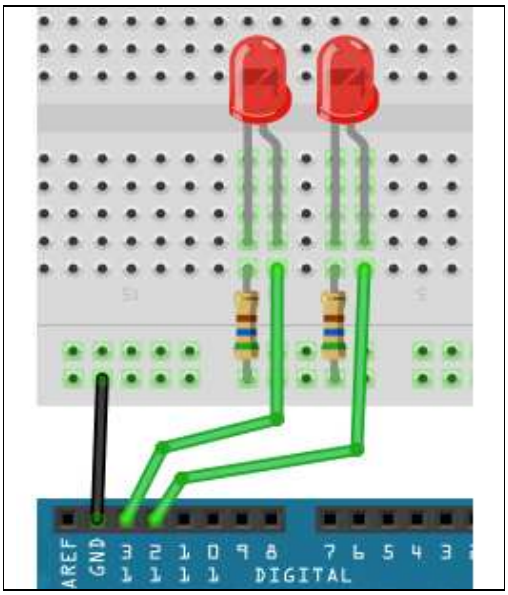


Conexión de un Led

Explicación, Línea Por Línea

1 y 2	Declaración de Variables a las que asignamos los valores, que luego corresponderán a los PINES que usaremos.
3	Comienza función de Configuración "**setup( )**".
4 y 5	En esta línea se configuran en que modo serán usados los PINes. Usaremos el número de PIN que indiquen las variables "**PinLed_01**" y "**PinLed_02**". Entonces serán usados como SALIDAS DIGITALES. En este ejemplo, las variables contienen los valores 12 y 13.
6	Llave que finaliza la función de Configuración
7	Cabecera e inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
8	Enciende el LED conectado al PIN indicado por la variable "**PinLed_01**".
9	Apaga el LED conectado al PIN indicado por la variable "**PinLed_02**".
10	Detiene el programa por 500 milisegundos.
11	Apaga el LED conectado al PIN indicado por la variable "**PinLed_01**".
12	Enciende el LED conectado al PIN indicado por la variable "**PinLed_02**".
13	Detiene el programa por 500 milisegundos – Mientras el LED esta Apagado
14	Llave que marca el final de la función "**loop( )**". Que comenzó en la línea 7

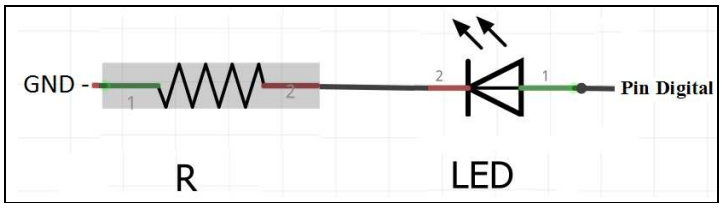
Construcción Del Circuito



Lista de Materiales:

1 Placa Protoboard.
2 Led.
2 Resistencia de 470Ω.
3 Cables Macho-Macho.
Placa Arduino.
Cable USB

Un Pin, es una salida de la Placa Arduino, donde conectamos (enchufamos) un cable del circuito. En este ejemplo usamos Tres pines de la Placa el GND y los pines 12 y 13.



Esta vez, usaremos dos veces el mismo diagrama de conexión, solo que estarán conectados a dos pines distintos (Tanto en la Placa Arduino como en el Protoboard). Es decir, el circuito del led 1 estará conectado a "**PinLed_01**" y el led 2 a "**PinLed_02**". Pero Ambos al Mismo GND.

Muy Importante: Recordar que Los LED tienen una pata más larga que otra, esa pata larga, se conoce como **ánodo** (el polo positivo). Es por donde tiene que entrar la corriente. La pata más pequeña se conoce como **cátodo** (polo negativo).

4- Prender 3 LED en Secuencia y Luego Apagarlos en la misma Secuencia.

Conectar tres LEDs a los pines (3,5 y 7) y penderlos en secuencia con una demora de medio segundo (500 Mili Segundos), luego apagarlos en idéntica secuencia con igual intervalo de tiempo.

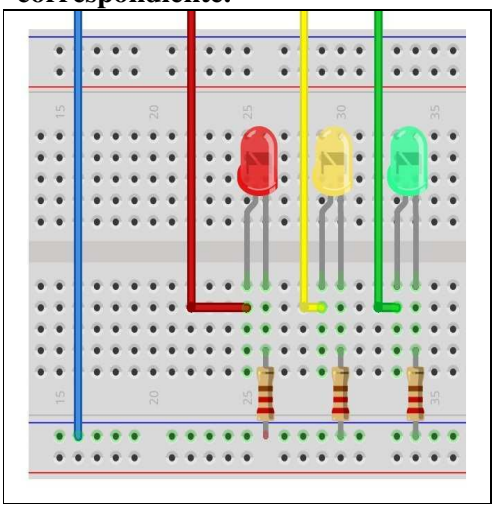
RECORDAR: GND es el PIN considerado como Negativo o Masa en la Placa Arduino. Si buscamos veremos que hay uno o varios pines así identificados (La cantidad dependerá del modelo de placa que estemos analizando).

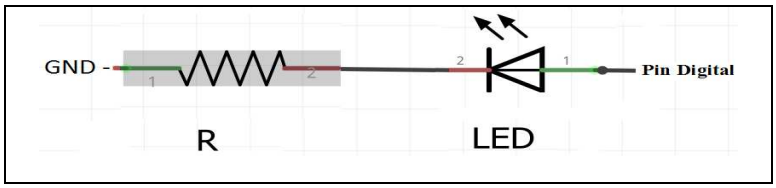
(Programa "001_Led_04")

```
1  int PinLed_01 =  3;
2  int PinLed_02 =  4;
3  int PinLed_03 =  7;
-
4  int TiempoLed01=1000;
5  int TiempoLed02=1000;
6  int TiempoLed03=1000;
7  int TiempoEntreLed = 500;
-
8  void setup( ) {
9   pinMode(PinLed_01,OUTPUT);
10  pinMode(PinLed_02,OUTPUT);
11  pinMode(PinLed_03,OUTPUT);
12  }
13 void loop( ) {
14   digitalWrite(PinLed_01,HIGH);
15   delay(TiempoLed01);
16   digitalWrite(PinLed_02,HIGH);
17   delay(TiempoLed02);
18   digitalWrite(PinLed_03,HIGH);
19   delay(TiempoEntreLed);
20   digitalWrite(PinLed_01,LOW);
21   delay(TiempoLed01);
22   digitalWrite(PinLed_02,LOW);
23   delay(TiempoLed02);
24   digitalWrite(PinLed_03,LOW);
25   delay(TiempoLed03);
26 }
```

Lista de Materiales: 1 Placa Protoboard - 3 LED - 3 Resistencia de 470Ω - 4 Cables Macho-Macho -Placa Arduino - Cable USB

En Cuanto al circuito, repetiremos tres veces el mismo, conectando cada LED al PIN correspondiente.





Los Cables deben ser conectados de la Siguiente Forma: el Cable Azul a GND, y los otros cables que pertenecen a los led, conectar a los Pines configurados en el comienzo del programa en las variables "**PinLed_01**", "**PinLed_02**" y "**PinLed_03**".

Como modificaría el programa, para que los LED se apaguen en el orden inverso al que se prendieron?

INTERESANTE - Arduino y el puerto serie

Veamos algo nuevo. Enviar mensajes desde la Placa Arduino a la PC y viceversa.
Prácticamente todas las placas Arduino disponen al menos de un Puerto Serie. Las placas Arduino UNO y Mini Pro disponen de uno que opera a nivel TTL 0V / 5V, por lo que son directamente compatibles con la conexión USB. Por su parte, Arduino Mega y Arduino Due disponen de 4 TTL 0V / 5V.

Los puertos serie están físicamente unidos a distintos pines de la placa Arduino. Lógicamente, mientras usamos los puertos de serie no podemos usar como entradas o salidas digitales los pines asociados con el puerto serie en uso. (Ver mapa de pines en nuestra página - **PinOut**).

En Arduino UNO y Mini Pro, los pines empleados son 0 (llamado RX) y el pin 1 (llamado TX). En el caso de Arduino Mega y Arduino Due, que tienen cuatro puertos serie, el puerto serie 0 está conectado a los pines 0 (RX) y 1 (TX), el puerto serie 1 a los pines 19 (RX) y 18 (TX) el puerto serie 2 a los pines 17 (RX) y 16 (TX), y el puerto serie 3 a los pines 15 (RX) y 14 (TX).

También debe saber que en todas los modelos de placas arduino, es posible Definir nuevos puertos mediante librarías (SoftwareSerial.h) pero esto lo veremos mas adelante.

Muchos modelos de placas Arduino disponen de un conector USB o Micro USB conectado a uno de los puertos de serie, lo que simplifica el proceso de conexión con un ordenador. Sin embargo algunas placas, como por ejemplo la Mini Pro, prescinden de este conector por lo que la única forma de conectarse a las mismas es directamente a través de los pines correspondientes.

Todo esto lo veremos con más detalle (para cada modelo de placa) un poco más adelante, pero ya podemos comenzar a trabajar con el puerto serie.

FUNCIONES QUE UTILIZAREMOS EN NUESTRO PRÓXIMO PROGRAMA.

- **Serial.begin(9600)**: Abrimos el Puerto Serie para poder enviar y recibir mensajes, a través del cable de comunicación, mientras se esta ejecutando el programa en nuestra Placa Arduino.
  El numero, indica la velocidad de conexión, por ahora solo usaremos 9600. Para ver estos mensajes, cuando se este ejecutando el programa en la Placa, abrir en el "**IDE**" (donde escribió el programa que envió a la placa Arduino), en el Menú "**Herramientas**", elegir la opción "**Monitor Serie**". **Nota Importante**: Cuando se utiliza la comunicación serie, los PINs digitales 0 (RX) y 1 (TX) no pueden utilizarse al mismo tiempo.

- **Serial.println(Cadena ó Variable ó Función que retorne un Valor)**: Imprime los datos en el puerto serie, seguido por un retorno de carro automático y salto de línea.
- **Serial.print ( ):**, igual que "**Serial.println( )"** pero No baja de renglón.

```
Serial.println( Valor); // Envía contenido de "Valor" al puerto - Lo veremos en la PC
```

El ejemplo mostrara en la PC, cada 1 segundo, el siguiente Mensaje: **Enviamos el número: 5**

```
1   int Numero = 5;
2   void setup( ) {
3      Serial.begin(9600); // configura el puerto serie a 9600bps
4      Serial.println("Acá ya puedo Enviar mensajes" ); // Envío Texto
5   }
-
6   void loop( ) {
7     Serial.print( "Enviamos el número: ");  // Envío texto
8     Serial.println(Numero);   // Envío contenido de la variable Número
9     delay(1000); // espera 1 segundo
10  }
```

3	Abro el canal de comunicaciones o puerto serie
4	Envío mensaje, que solo será enviado una vez, ya que la función "**setup( )**" no se repite.
7 y 8	Envío Mensajes de texto y numérico. Esto se repetirá muchas veces, ya que esta dentro de la función "**loop( )**" que se repite permanentemente

5- Informar en el Puerto Serie Cuando se Prende o Apaga un LED.

Hacer un programa y circuito, que prenda y apague un LED y se informe en la Pantalla de la PC por medio del Puerto Serie cada vez que el Led se prenda o apague.

(Programa "750_Puerto_Serie_01_Prendo_Led_01")

```
1   int PinLed = 13;
-
2   void setup( ) {
3     pinMode(PinLed, OUTPUT);
4     Serial.begin(9600); // configura el puerto serie a 9600bps
-
5     Serial.print( "Led Conectado en el PIN "); // Envío Mensaje de Texto
6     Serial.println( PinLed ); // Envío Mensaje numérico
7   }
-
8   void loop( ) {
9     digitalWrite(PinLed,HIGH);
10    Serial.println( "Led Prendido"); // Envío Texto
11    delay(500);
-
12    digitalWrite(PinLed,LOW);
13    Serial.println( "Led Apagado");  // Envío Texto
14    delay(500);
15  }
```

Explicación, Línea Por Línea	
1	Declaro Variable que contendrá el Nro de Pin al que conectaré el LED.
2	Comienza función de Configuración "**setup( )**"

3	En esta línea se configura en que modo será usado el PIN. El PIN 13 será usado como una SALIDA DIGITAL.
4	Apertura del Puerto Serie, y se establece en 9600 Baudios la Velocidad de Comunicación. Esto nos permitirá mandar mensajes desde la placa y verlos en la computadora. Para ver estos mensajes, cuando se este ejecutando el programa en la Placa, abrir en el IDE (donde escribió el programa que envió a la placa Arduino), en el Menú "**Herramientas**", elegir la opción "**Monitor Serie**".
5	Envío o escribo el mensaje de texto en el puerto serie "**Led Conectado en el PIN**". Este mensaje se mostrará en la PC UNA sola vez, ya que se envía desde la función "**setup( )**", y esta solo se ejecuta UNA vez.
6	Envío o escribo mensaje numérico, es decir envío el contenido de la variable por el puerto serie. Al igual que el mensaje anterior, este se mostrará en la PC UNA sola vez, ya que se envía desde la función "**setup( )**", y esta solo se ejecuta UNA vez.
7	Llave que finaliza la función de Configuración
8	Cabecera e inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
9	Enciende el LED conectado al PIN 13
10	Envío o escribo el mensaje de texto en el puerto serie. (Se verá en la PC).
11	Detiene el programa por 500 milisegundos – Mientras el LED esta Prendido.
12	Apaga el LED conectado al PIN 13
13	Envío o escribo el mensaje de texto en el puerto serie. (Se verá en la PC).
14	Detiene el programa por 500 milisegundos – Mientras el LED esta Apagado.
15	Llave que marca el final de la función "**loop( )**". Que comenzó en la línea 4

CIRCUITO PARA NUESTRO PROYECTO

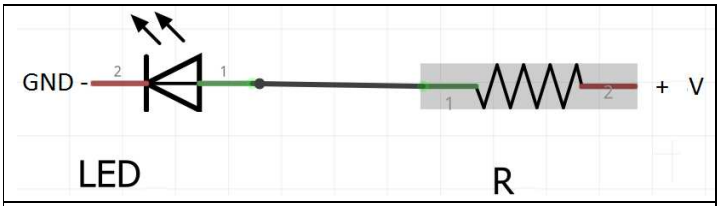


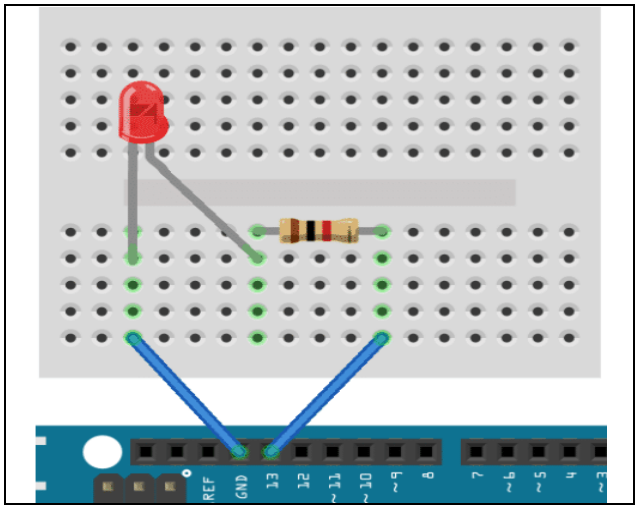
Diagrama de Conexión

Para irnos acostumbrando a esta muy sencilla simbología, veremos el circuito que usaremos con nuestro primer programa. Como puede apreciar es muy fácil de entender y construir.
Ahora solo queda construir sobre la placa donde haremos nuestras pruebas (Protoboard).
IMPORTANTE: GND es el PIN considerado como Negativo o Masa en la Placa Arduino. Si buscamos veremos que hay uno o varios pines así identificados (La cantidad dependerá del modelo de placa que estemos analizando).

Lista de Materiales

- 1 Placa Protoboard. (Ver apéndice A - Placa Protoboard)
- 1 Led.
- 1 Resistencia de 470Ω.
- 2 Cables Macho-Macho.
- Placa Arduino.
- Cable USB.

Recordatorio: Un Pin, es una salida de la Placa Arduino, donde conectamos (enchufamos) un cable del circuito. En este ejemplo usamos dos pines de la Placa el GND y el 13.



6- Prender 3 LED en Secuencia y Luego Apagarlos en la misma Secuencia - Informando las acciones por el Puerto Serie.

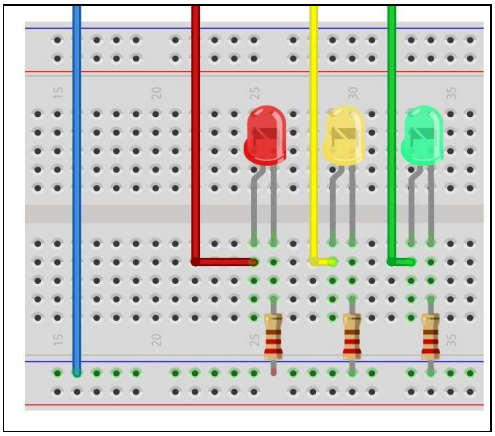
Informa a través del puerto serie, en que puertos se han configurados los LEDs y cada vez que se prenda o apague un Led.

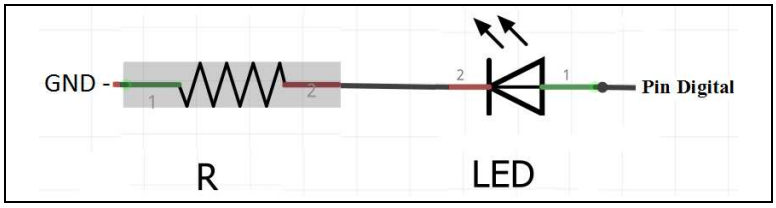
(Programa "750_Puerto_Serie_01_Prendo_Led_02")

```
1  int PinLed_01 =  3;
2  int PinLed_02 =  4;
3  int PinLed_03 =  7;
-
4  int TiempoLed01=1000;
5  int TiempoLed02=1000;
6  int TiempoLed03=1000;
7  int TiempoEntreLed = 500;
-
8  void setup() {
9   Serial.begin(9600); // configura a 9600bps
10  pinMode(PinLed_01,OUTPUT);
11  pinMode(PinLed_02,OUTPUT);
12  pinMode(PinLed_03,OUTPUT);
-
13  Serial.print( "Se configuraron LEDs en PINES: "); // Mensaje de Texto
14  Serial.print( PinLed_01 ); // Envío Mensaje numérico
15  Serial.print( " - "); // Envío Mensaje de Texto
16  Serial.print( PinLed_02 ); // Envío Mensaje numérico
17  Serial.print( " - "); // Envío Mensaje de Texto
18  Serial.println( PinLed_03 ); // Envío Mensaje numérico
19  }
-
20  void loop() {
21   digitalWrite(PinLed_01,HIGH);
22   Serial.print( "Prendió LED conectado al PIN: "); // Mensaje de Texto
23   Serial.println( PinLed_01 ); // Envío Mensaje numérico
24   delay(TiempoLed01);
-
25   digitalWrite(PinLed_02,HIGH);
26   Serial.print( "Prendió LED conectado al PIN: "); // Mensaje de Texto
27   Serial.println( PinLed_02 ); // Envío Mensaje numérico
28   delay(TiempoLed02);
-
29   digitalWrite(PinLed_03,HIGH);
30   Serial.print( "Prendió LED conectado al PIN: "); // Mensaje de Texto
31   Serial.println( PinLed_03 ); // Envío Mensaje numérico
-
32   delay(TiempoEntreLed);
-
33   digitalWrite(PinLed_01,LOW);
34   Serial.print( "Apago LED conectado al PIN: "); // Mensaje de Texto
35   Serial.println( PinLed_01 ); // Envío Mensaje numérico
36   delay(TiempoLed01);
-
37   digitalWrite(PinLed_02,LOW);
38   Serial.print( "Apago LED conectado al PIN: "); // Mensaje de Texto
39   Serial.println( PinLed_02 ); // Envío Mensaje numérico
40   delay(TiempoLed02);
```

```
-
41    digitalWrite(PinLed_03,LOW);
42    Serial.print( "Apago LED conectado al PIN: "); // Mensaje de Texto
43    Serial.println( PinLed_03 ); // Envío Mensaje numérico
44    delay(TiempoLed03);
45    Serial.println( "============================="); // Mensaje de Texto
46  }
```

Lista de Materiales: 1 Placa Protoboard - 3 LED - 3 Resistencia de 470Ω - 4 Cables Macho-Macho -Placa Arduino - Cable USB

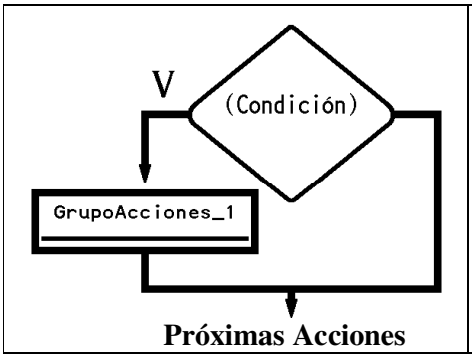




Los Cables deben ser conectados de la Siguiente Forma: el Cable Azul a GND, y los otros cables que pertenecen a los led, conectar a los Pines configurados en el comienzo del programa en las variables "PinLed_01", "PinLed_02" y "PinLed_03".

Estructura Condicional "if"

El "**if**( )" es una estructura usada para probar o verificar si una determinada condición se ha cumplido, por ejemplo: comprobar si un valor es igual a un cierto número, entonces si esta condición se cumple, se ejecuta una serie de declaraciones o acciones. Para nuestro ejemplo "GrupoAccines_1".



Si cumple (condición) hacer {
GrupoAcciones_1;
}

Próximas_Acciones;

Y si la condición fuera falsa (no se cumple), el programa salta (no ejecuta el Grupo de Acciones) y continua con las próximas Acciones.

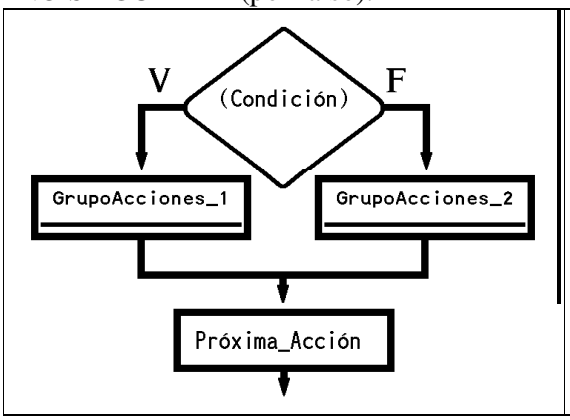
Dicho de otra forma, la condición (operador lógico) puede comparar una variable que contiene un valor, con otra variable o constante. Si la comparación, o la condición entre paréntesis se cumple (es verdadera), las declaraciones dentro de los corchetes se ejecutan. Si no es así, el programa salta sobre ellas (hasta la llave que cierra el bloque) y continúa con "**PróximasAcciones**".

Esta sentencia, debe codificarse de la siguiente forma (Para que nuestro IDE lo entienda). Y es muy importante saber que las llaves Abren y Cierran el Bloque de acciones. y el Punto y coma, da por terminada una acción.

```
if (  Condición ) {
  GrupoAcciones_1;
}
próximasAcciones;
```

Estructura Condicional "if_else"

Una extensión de la estructura condicional "**if**", es agregar la opción de hacer cosas cuando la condición NO SE CUMPLE (por falso).




```
if ( Condición ) {
  GrupoAcciones_1;
} else {
  GrupoAcciones_2;
}

PróximaAcción;
```

En este ejemplo, la condición (operador lógico) puede comparar una variable que contiene un valor, con otra variable o constante. Y si la condición entre paréntesis se cumple (es verdadera), se ejecutaran las operaciones dentro de los corchetes "**GrupoAcciones_1**".
De lo contrario (si no se cumple la condición), el programa salta sobre ellas hasta el grupo de acciones que están en la parte Falsa: "**GrupoAcciones_2**".
Una vez ejecutado uno de los dos bloques de acciones (No se ejecutan los dos bloques), continua con "PróximaAcción".

Tabla De operadores Lógicos que demos usar

Operador	Descripción	Operador	Descripción
==	Igual - (A igual a B)	&&	Y - (A y B)
||	or - (A ó B)	!	No . Es la Negación (Lo opuesto)
>	Mayor que – (A Mayor que B)	<	Menor que – (A Menor que B)
>=	Mayor o Igual (A Mayor o Igual que B)	<=	Menor o Igual – (A Menor o igual que B)
!=	Distinto – (A distinto de B)		

FUNCIONES QUE UTILIZAREMOS EN NUESTRO PRÓXIMO PROGRAMA.

- **millis( )**: Devuelve el número de milisegundos transcurrido desde el inicio del programa, hasta el momento actual. Normalmente será un valor grande (dependiendo del tiempo que esté en marcha la aplicación, después de cargada o después de la última vez que se pulsó el botón "reset" de la tarjeta). **Nota:** Este número se desbordará (si no se resetea de nuevo a cero), después de aproximadamente 9 horas de trabajo continuo.

```
valor = millis( ); // La Variable "valor" Guardará el número de milisegundos
```

- **analogWrite(Pin,Valor)**: Esta función (instrucción) sirve para escribir un pseudo-valor analógico, utilizando el procedimiento de modulación por ancho de pulso (PWM) a uno de los pin´s de Arduino marcados como "pin PWM". El más reciente Arduino, que implementa el chip ATmega168, permite habilitar como salidas analógicas tipo PWM los pines 3, 5, 6,9, 10 y 11. Los modelos de Arduino más antiguos que implementan el chip ATmega8, solo tiene habilitadas para esta función los pines 9, 10 y 11. El valor que se puede enviar a estos pines de salida analógica puede darse en forma de variable o constante, pero siempre dentro del intervalo 0-255.

```
analogWrite(pin, valor); // escribe 'valor' en el 'pin' definido como analógico.
```

Si enviamos el valor 0 genera una salida de 0 voltios en el pin especificado; un valor de 255 genera una salida de 5 voltios de salida en el pin especificado. Para valores de entre 0 y 255, el pin saca tensiones intermedias entre 0 y 5 voltios - **el valor HIGH de salida equivale a 5v (5**

voltios). Debido a que esta es una función de hardware, en el pin de salida analógica (PWN) se generará una onda constante después de ejecutada la instrucción analogWrite, hasta que se llegue a ejecutar otra instrucción analogWrite (o una llamada a digitalRead o digitalWrite en el mismo pin). Nota: Las salidas analógicas a diferencia de las digitales, no necesitan ser declaradas como INPUT u OUTPUT. Pero es muy buena la costumbre declararlas para no rehusarlas por error..

7- Prender y Apagar un LED sin usar la función "delay( )".

Este programa es una variante de otro que ya construimos anteriormente, pero es muy importante, ya que controlamos el tiempo sin la función "delay( )". Esta forma de programar la pondremos en práctica siempre que sea posible, ya que es el primer paso a la Programación Multitarea.

(Programa "001_Led_05")

```
1   int ledPin = 5;
-
2   long TiempoInicial,
3        TiempoEspera = 500;
4   int   EstadoLed;
-
5   void setup( ) {
6    pinMode(ledPin, OUTPUT);
7    TiempoInicial = millis( );
8    EstadoLed = HIGH;
9   }
10  void loop( ) {
11   if(millis( ) - TiempoInicial < TiempoEspera){
12      digitalWrite(ledPin, EstadoLed);
13   }else{
14      EstadoLed = 1 - EstadoLed; // Invierte el estado
15      TiempoInicial = millis( );      // Toma nuevo tiempo inicial. Próxima espera
16   }
17  }
```

Una Recomendación: Detente en este ejemplo hasta que realmente lo comprendas, ya que resulta muy importante y el modelo de control de tiempo sin usar la función delay( ).

Si te resulta complicado consulta a tu profesor en Próxima Clase.

Explicación, Línea Por Línea

1	**Declaración de una constante**, la que usaremos para identificar el número de Pin al que conectaremos el LED.
2	Variable en donde tomamos el tiempo inicial, cada vez que comenzamos a medir el tiempo de espera.
3	Variable en la que configuramos el tiempo (en milisegundos) que debemos esperar después de cada cambio de estado, para hacer otro cambio. (PRENDER - APAGAR). Le asigno en el mismo acto de la declamación el valor 500 Milisegundos.
4	Variable que indicara el estado en que debe estar el LED (Prendido o apagado)
5	Comienza función de Configuración "**setup( )**". Acá configuramos que hacer con cada PIN (Entrada o salida). También podremos abrir el Puerto Serie, o hacer tareas que se hacen una sola vez y al comienzo del programa.
6	Configuro el PIN "**PinLed**" como Salida, para poder prender y apagar el LED que conectaremos ahí.
7	Tomo el Tiempo y lo guardo como valor inicial, para luego poder saber cuantos milisegundos transcurrieron
8	Configuro el estado inicial del LED, en este caso, debe comenzar prendido.
9	Finaliza función de Configuración "**setup( )**".
10	Cabecera e inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
11	Pregunto si debo continuar esperando.

12	Si continuo esperando, pongo el LED en el estado que diga la variable "**EstadoLed**"
13	Si ya termino el tiempo de espera.
14	Cambio o invierto el estado que indica la variable "**EstadoLed**"
15	Tomo el Nuevo tiempo Inicial, y así poder esperar el nuevo lapso de tiempo.
16	Termina la parte falsa del "**if**"
17	Llave que finaliza la función "**loop( )**".

CIRCUITO PARA NUESTRO PROYECTO

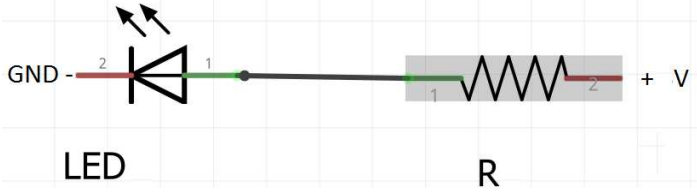
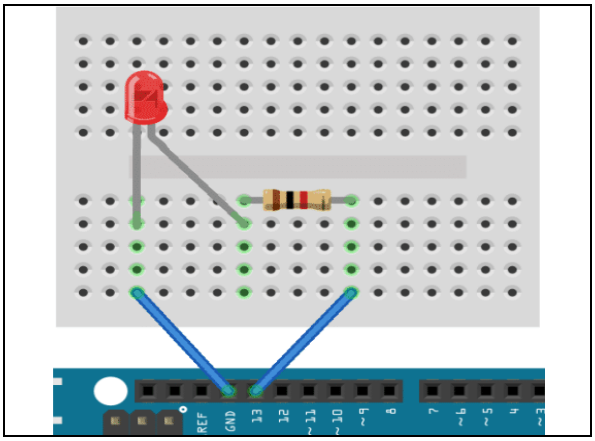


Diagrama de Conexión

Como puede apreciar es muy fácil de entender y construir. Aparte ya lo hemos construido!

Lista de Materiales: 1 Placa Protoboard, 1 Led, 1 Resistencia de 470Ω, 2 Cables Macho-Macho, Placa Arduino, Cable USB



Muy Importante: Recordar que Los LED tienen una pata más larga que otra, esa pata larga, se conoce como **ánodo** (el polo positivo). Es por donde tiene que entrar la corriente. La pata más pequeña se conoce como **cátodo** (polo negativo).

8- Prender 2 LED – Alternando el prendido y apagado. Sin usar Función "delay( )"

Pender dos LED alternándolos cada medio segundo - Cuando uno se prende el otro se apaga. Este programa es una variante de otro que ya construimos anteriormente, pero es muy importante, ya que controlamos el tiempo sin la función "delay( )". Esta forma de programar la pondremos en práctica siempre que sea posible, ya que es el primer paso a la Programación Multitarea.

(Programa "001_Led_06")

```
1   int PinLed_01 = 3;
2   int PinLed_02 = 5;
-
3   long TiempoInicial,
4        TiempoEspera;
5   int  EstadoLed;
-
6   void setup( ) {
7     pinMode(PinLed_01, OUTPUT);
8     pinMode(PinLed_02, OUTPUT);
9     TiempoInicial = millis( );
10    TiempoEspera = 250;
11    EstadoLed = HIGH;  // Inicializo como prendido
12  }
```

IMPORTANTE RECORDAR:
Un Pin, es una salida de la Placa Arduino, donde conectamos (enchufamos) un cable del circuito.

```
-
13  void loop( ) {
14    if( millis( ) < TiempoInicial + TiempoEspera ){
15      digitalWrite(PinLed_01, EstadoLed);
16      digitalWrite(PinLed_02, !EstadoLed);  // Ver Explicación
17    }else{
18      EstadoLed = 1 - EstadoLed; // Ver Explicación
19      TiempoInicial = millis( );
20    }
21  }
```

Explicación, Línea Por Línea

1 y 2	**Declaración de Variables**, que usaremos para identificar el número cada uno de los PINs usados para conectaremos los LEDs.
3	Variable en donde tomamos el tiempo inicial, cada vez que comenzamos a medir el tiempo de espera.
4	Variable en la que configuramos el tiempo (en milisegundos) que debemos esperar después de cada cambio de estado, para hacer otro cambio. (PRENDER - APAGAR)
5	Variable que indicara el estado en que debe estar el LED (Prendido o apagado)
6	Comienza función de Configuración "**setup( )**". Acá configuramos que hacer con cada PIN (Entrada o salida). También podremos abrir el Puerto Serie, o hacer tareas que se hacen una sola vez y al comienzo del programa.
7 y 8	Configuro los Pines "**PinLed_01**" y "**PinLed_02**" como Salida, para poder prender y apagar los LEDs que conectaremos ahí.
9	Tomo el Tiempo y lo guardo como valor inicial, para luego poder saber cuantos milisegundos transcurrieron
10	Configuro el tiempo de espera entre estado y estado (Prendido - Apagado)
11	Configuro el estado inicial de un LED, en este caso, debe comenzar prendido.
12	Finaliza función de Configuración "**setup( )**".
13	Cabecera e inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
14	Pregunto si debo continuar esperando.
15 y 16	**Si continuo esperando, pongo "**PinLed_01**" en el estado que diga la variable "**EstadoLed**" y "**PinLed_02**" en el estado negado (Opuesto o invertido) que tenga la variable "**EstadoLed**"**
17	Si ya termino el tiempo de espera. "**else**" la parte falsa del "**if**"
18	Cambio o invierto el estado que indica la variable "**EstadoLed**" También podría haber puesto, con igual resultado: **EstadoLed = !EstadoLed;** Recordar que el signo de Admiración "**!**" significa NO (negación o invertido)
19	Tomo el Nuevo tiempo Inicial, y así poder esperar el nuevo lapso de tiempo.
20	Termina la parte falsa del "if"
21	Llave que finaliza la función "**loop( )**".

CIRCUITO PARA NUESTRO PROYECTO

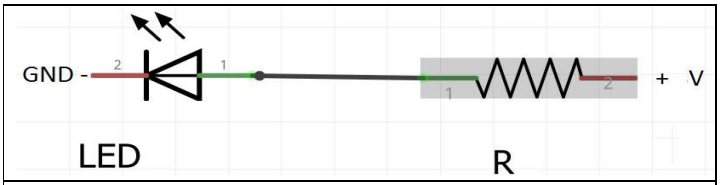


Diagrama de Conexión

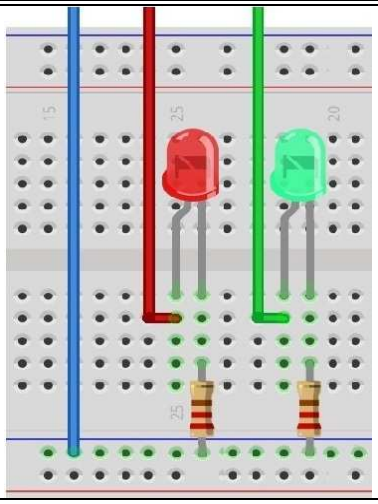
Para construir este circuito, repita para cada LED, y ambos debe conectarlos al mismo GND.

Como puede apreciar es muy fácil de entender y construir. Aparte ya lo hemos construido!

Lista de Materiales

1 Placa Protoboard, 2 Led, 2 Resistencia de 470Ω, 3 Cables Macho-Macho, Placa Arduino, Cable USB

Conectar Cable Azul a GND, los otros dos cables (Rojo y Verde) a los pines en que se hayan configurado las variables "**PinLed_01**" y "**PinLed_02**" en el programa.



Muy Importante: Recordar que Los LED tienen una pata más larga que otra, esa pata larga, se conoce como **ánodo** (el polo positivo). Es por donde tiene que entrar la corriente. La pata más pequeña se conoce como **cátodo** (polo negativo).

9- Prendido y apagado de un LED en Forma Gradual (Variando la intensidad). Sin usar la función delay( ).

Hacer un programa y circuito eléctrico que prenda gradualmente un LED, subiendo la intensidad desde su mínimo al máximo, y luego se apague gradualmente, desde el máximo al mínimo.

(Programa "001_Led_05_Subo_Bajo_Brillo_Led_01")

```
1   const int PinLed   =  9;
-
2   int Sube = 1;
3   int Baja = 2;
4   int Procesando;
-
5   long TiempoInicial,
6        TiempoEspera  = 40;  // Cantidad de Milisegundos entre Incrementos
7   int  Incremento    =  5,
8        Intensidad;
-
9   void setup( ) {
10    Serial.begin(9600);
11    pinMode(PinLed, OUTPUT);
12    TiempoInicial = millis( );
13    Intensidad    = 0;
14    Procesando    = Sube;
15  }
16  void loop( ) {
17   CalculaIntensidad( );
18   analogWrite(PinLed, Intensidad);
19   Serial.println(Intensidad);
20   // Mientras Procesa la Intensidad, acá puede hacer otras cosas..!!!
21  }
22  void CalculaIntensidad( ){
23    if(Procesando == Sube){
24      if( Intensidad < 255 - Incremento){    // Controla cuanto sube
25        if ( millis( ) - TiempoInicial > TiempoEspera){
26          Intensidad = Intensidad + Incremento;
27          TiempoInicial = millis( );
28        }
29      }else{
```

```
30          Procesando = Baja;
31          Serial.println("Bajando");
32        }
33      }
34        if(Procesando == Baja){
35        if(Intensidad > Incremento ){     // Controla Cuanto Baja
36          if ( millis( ) - TiempoInicial > TiempoEspera){
37            Intensidad = Intensidad - Incremento;
38            TiempoInicial = millis( );
39          }
40        }else{
41            Procesando = Sube;
42            Serial.println("Subiendo");
43        }
44      }
45  }
```

Explicación, Línea Por Línea

1	**Declaración de una constante**, la que usaremos para identificar el número de Pin al que conectaremos el LED.
2 y 3	Declaración de dos variables, que usaremos para identificar dos procesos. Uno para incrementar (subir el brillo) y la otra para decrementar (bajar el brillo) el valor con el que asociamos el brillo del LED.
4	Variable que usaremos para indicar cual de los procesos debemos ejecutar.
5	Variable en donde tomamos el tiempo inicial, cada vez que comenzamos a medir el tiempo de espera
6	Variable en la que configuramos el tiempo (en milisegundos) que debemos esperar después de cada cambio, para hacer otro cambio, en la intensidad del LED.
7	Cantidad de unidades que "subiré" o "bajare" la intensidad del LED.
8	Intensidad con que haré Brillar el LED. Recordar que los valores con que se puede hacer brillar un LED, son los comprendidos en el intervalo 0 y 255.
9	Comienza función de Configuración "**setup( )**".
10	Apertura del Puerto Serie, y se establece en 9600 Baudios la Velocidad de Comunicación. Esto nos permitirá mandar mensajes desde la placa y verlos en la computadora. Para ver estos mensajes, cuando se este ejecutando el programa en la Placa, abrir en el IDE (donde escribió el programa que envió a la placa Arduino), en el Menú "**Herramientas**", elegir la opción "**Monitor Serie**".
11	Configuro la Variable "**PinLed**" como Salida, para poder prender y apagar el LED que conectaremos ahí.
12	Tomo El tiempo inicial para comenzar.
13	La intensidad del LED, con la que iniciara todo el proceso.
14	Y configuro el primer proceso que realizare, en este caso, comenzare a subir la intensidad, desde el valor inicial configurado en la línea 13.
15	Llave que finaliza la función de Configuración: "**setup( )**".
16	Cabecera e inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
17	Llamado a la función "**CalculaIntensidad( )**" donde realizaremos los cálculos de intensidad con la que brillara el LED. Recordar que los valores con que se puede hacer brillar un LED, son los comprendidos en el intervalo 0 y 255.
18	Prendo el LED, con la intensidad calculada y guardada en la variable "**Intensidad**"
19	Escribo en el puerto serie, el valor contenido por la Variable "**Intensidad**". Para ver estos mensajes, cuando se este ejecutando el programa en la Placa, abrir en el "**IDE**" (donde escribió el programa que envió a la placa Arduino), en el Menú "**Herramientas**", elegir la opción "**Monitor Serie**".

20	Un Comentario del Código, pero ahí podría poner cualquier otra acción, para hacer otras cosas, además de hacer brillar el LED y Apagarlo
21	Llave que finaliza la función "**loop( )**".
22	Cabecera o inicio de la Función "**CalculaIntensidad( )**"
23	Identificamos que debemos hacer, "**Subir**" o "**Bajar**" – En esta Línea "**Subir**".
24	Pregunto si Puedo continuar Subiendo la Intensidad (Con la que brillará el LED).
25	Acá pregunto si ya paso el tiempo de espera, entre cada variación de Intensidad.
26	Subo la Intensidad con la que prenderé el LED. Simplemente sumando la Constante declarada para tal fin.
27	Una Vez Incrementada la Intensidad, Tomo el Nuevo tiempo Inicial, y así poder esperar el nuevo lapso de tiempo.
28	Final de la parte Verdadera del Control de Tiempo.
29	Finaliza de la parte verdadera del control de Proceso, en este caso "Bajar" y comienzo da la parte falsa, es decir, cuando termina de Bajar.
30	Como Ya termino lo que estaba haciendo, ahora configura el cambio o próxima actividad. En teste Caso "**Baja**r".
31	Muestro en el Puerto Serie, el mensaje de lo que me preparo a Realizar, en este Caso "**Bajar**". Recordar que para ver estos mensajes, cuando se este ejecutando el programa en la Placa, abrir en el IDE (donde escribió el programa que envió a la placa Arduino), en el Menú "**Herramientas**", elegir la opción "**Monitor Serie**".
32	Termino la Parte Falsa del Control de Actividad.
33	Fin del "**if**" que comenzó en la línea 23.
34	Identificamos que debemos hacer, "**Subir**" o "**Bajar**" – En esta Línea "**Bajar**".
35	Pregunto si Puedo continuar Bajando la Intensidad. (Con la que brillará el LED).
36	Acá pregunto si ya paso el tiempo de espera, entre cada variación de Intensidad.
37	Bajo la Intensidad con la que prenderé el LED. Simplemente sumando la Constante declarada para tal fin.
38	Una Vez Disminuida la Intensidad, Tomo el Nuevo tiempo Inicial, y así poder esperar el nuevo lapso de tiempo.
39	Final de la parte Verdadera del Control de Tiempo.
40	Finaliza de la parte verdadera del control de Proceso, en este caso "Bajar" y comienzo da la parte falsa, es decir, cuando termina de Bajar.
41	Como Ya termino lo que estaba haciendo, ahora configura el cambio o próxima actividad. En teste Caso "**Subir**"
42	Muestro en el Puerto Serie, el mensaje de lo que me preparo a Realizar, en este Caso "**Subir**". Recordar que para ver estos mensajes, cuando se este ejecutando el programa en la Placa, abrir en el IDE (donde escribió el programa que envió a la placa Arduino), en el Menú "**Herramientas**", elegir la opción "**Monitor Serie**".
43	Termino la Parte Falsa del Control de Actividad.
44	Fin del "**if**" que comenzó en línea 34.
45	Finaliza la Función "**CalculaIntensidad( )**" que comenzó en línea 22.

CIRCUITO PARA NUESTRO PROYECTO

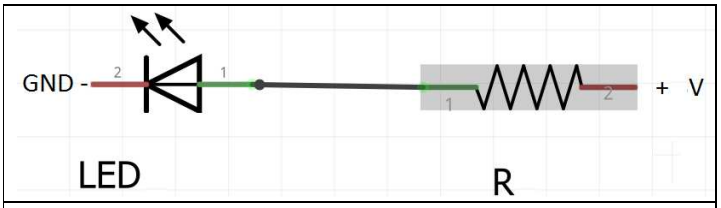


Diagrama de Conexión

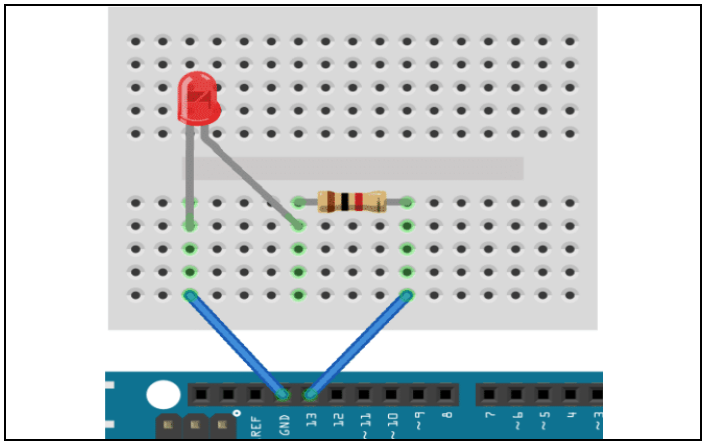
Nuevamente, Para acostumbrando a esta muy sencilla simbología, veremos el circuito que usaremos con este programa. Como puede apreciar es muy fácil de entender y construir.
Ahora solo queda hacer el circuito sobre la placa Protoboard, donde haremos nuestras pruebas.

RECORDAR: GND es el PIN considerado como Negativo o Masa en la Placa Arduino. Si buscamos veremos que hay uno o varios pines así identificados (La cantidad dependerá del modelo de placa que estemos analizando).

Lista de Materiales

- 1 Placa Protoboard.
- 1 Led.
- 1 Resistencia de 470Ω.
- 2 Cables Macho-Macho.
- Placa Arduino.
- Cable USB.

IMPORTANTE SABER: Un Pin, es una salida de la Placa Arduino, donde conectamos (enchufamos) un cable del circuito.



RECORDAR: GND es el PIN considerado como Negativo o Masa en la Placa Arduino. Si buscamos veremos que hay uno o varios pines así identificados (La cantidad dependerá del modelo de placa que estemos analizando).

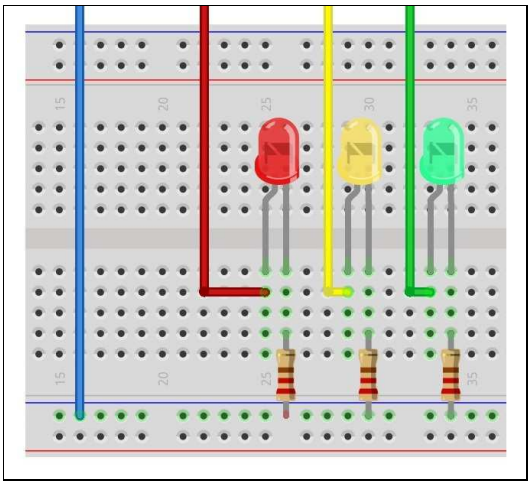
10- Semáforo Simple. Este programa es muy parecido al de los semáforos reales, que solo controlan las luces**.**

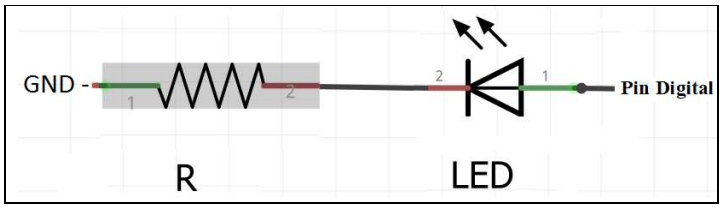
(Programa "001_Led_06_Semaforo_00")

```
1  long TiempoVerdeAuto    = 7000;  // Cantidad de Milisegundos en Verde
2  long TiempoAmarilloAuto = 2000;  // Cantidad de Milisegundos en Amarillo
3  long TiempoRojoAuto     = 7000;  // Cantidad de Milisegundos en Rojo
-
4  const int PinVerdeAutos    =  5;
5  const int PinAmarilloAutos =  7;
6  const int PinRojoAutos     =  9;
-
7  void setup( ) {
8    pinMode(PinVerdeAutos, OUTPUT);    //Semáforo Auto
9    pinMode(PinAmarilloAutos, OUTPUT); //Semáforo Auto
10   pinMode(PinRojoAutos, OUTPUT);     //Semáforo Auto
11 }
-
12 void loop( ) {
13   AutoVerde( );
14   delay(TiempoVerdeAuto);
15   AutoAmarillo( );
16   delay(TiempoAmarilloAuto);
17   AutoRojo( );
18   delay(TiempoRojoAuto);
19 }
-
20 void AutoVerde( ){
21    digitalWrite(PinVerdeAutos, HIGH); // Enciende el LED Verde Autos
22    digitalWrite(PinAmarilloAutos, LOW); // Apaga el LED Amarillo Autos
```

```
23    digitalWrite(PinRojoAutos, LOW);    // Apaga el LED Rojo Autos
24  }
-
25  void AutoAmarillo( ){
26    digitalWrite(PinVerdeAutos, LOW);
27    digitalWrite(PinAmarilloAutos, HIGH); // Enciende el LED Amarillo Autos
28    digitalWrite(PinRojoAutos, LOW);
29  }
-
30  void AutoRojo( ){
31    digitalWrite(PinVerdeAutos, LOW);
32    digitalWrite(PinAmarilloAutos, LOW);
33    digitalWrite(PinRojoAutos, HIGH);
34  }
```

Construcción Del Circuito





Lista de Materiales:

- 1 Placa Protoboard.
- 3 Led.
- 3 Resistencia de 470Ω.
- 4 Cables Macho-Macho.
- Placa Arduino.
- Cable USB

En Cuanto al circuito, los cables deben ser conectados de la siguiente forma, el Cable Azul a GND, y los otros cables que pertenecen a los LED, conectar a los Pines configurados en el comienzo del programa en las CONSTANTES: "**PinVerdeAutos**", "**PinAmarilloAutos**", "**PinRojoAutos**".

PROGRAMACIÓN MULTITAREA

La multitarea es la característica de los sistemas operativos modernos de permitir que varios procesos o aplicaciones se ejecuten aparentemente al mismo tiempo, compartiendo los recursos (uno o más procesadores). Permitiendo la ejecución de muchos más programas.

En esta categoría también se encuentran todos los sistemas que cumplen simultáneamente las necesidades de dos o más usuarios, llamados "sistemas multiusuarios", que compartan los mismos recursos. Este tipo de sistemas se emplea especialmente en redes. En resumen, se trata de fraccionamiento del tiempo (timesharing en inglés).

El concepto de este tipo de procesamiento es:

- Cada proceso tiene un turno y un tiempo para hacer sus cosas, y cuando se termina ese tiempo, el proceso debe suspender la tarea y esperar el próximo turno.
- No se permiten tiempos inactivos (Esperando que algo suceda, o que se desocupe un recurso). Si hay una espera, automáticamente le toca el turno al próximo proceso.

Cuando nos iniciamos en la programación con Arduino, la función "**delay( )**" es muy útil y nos permite visualizar y/o comprender algunas tareas o conceptos, sin embargo, una muy buena técnica en la programación, es evitar usarla, o directamente erradicarla, puesto que mientras "**delay( )**" se esta ejecutando (mientras esperamos) queda inmovilizado nuestro programa, y es incapaz de hacer otras tareas.

Un buen comienzo de la Programación Multitarea, es usar cierta lógica, que permita continuar con otras tareas mientras esperamos, y/o, usar funciones de temporización, que cuenten el tiempo y ejecuten la operación cuando el temporizador se dispare, permitiendo que nuestro programa haga varias tareas a la vez.

El concepto del **Paradigma de Programación Multitarea**, será usado frecuentemente en estas páginas.

11- Semáforo Simple. Programación Multitareas - No usa la función delay( ).

Implementar un Semáforo, sin usar función "**delay( )**". Permitir que el programa haga otras cosas mientras esta funcionando normalmente el semáforo.

(Programa "001_Led_06_Semaforo")

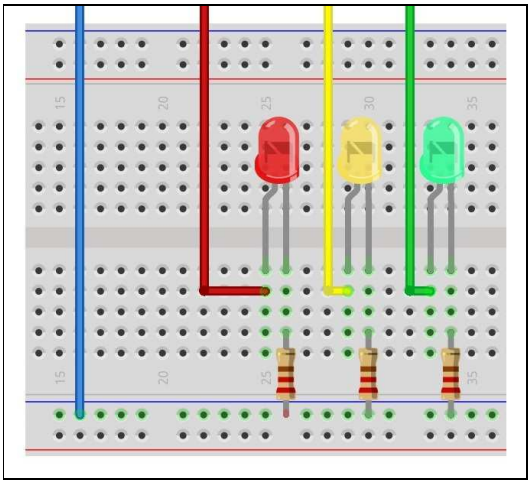
```
1   int Verde      = 1;   //Color a Procesar en el Semáforo
2   int Amarillo = 2;  //Color a Procesar en el Semáforo
3   int Rojo       = 3;    //Color a Procesar en el Semáforo
-
4   int Procesando;    // Indica que Color se esta procesando (prendido)
-
5   long TiempoVerdeAuto      = 5000;   // Cantidad de Milisegundos en Verde
6   long TiempoAmarilloAuto = 2000; // Cantidad de Milisegundos en Amarillo
7   long TiempoRojoAuto       = 5000;    // Cantidad de Milisegundos en Rojo
-
8   const int PinVerdeAutos      =  5;  // Constante que Configura Pin Luz Verde
9   const int PinAmarilloAutos  =  7; // Constante que Configura Pin Luz Amarilla
10  const int PinRojoAutos       =  9; // Constante que Configura Pin Luz Roja
-
11  long TiempoInicial = 0;   // Momento en el que se comienza a Controlar el Tiempo
-
12  void setup( ) {
13    pinMode(PinVerdeAutos, OUTPUT);      //Semáforo Auto
14    pinMode(PinAmarilloAutos, OUTPUT); //Semáforo Auto
15    pinMode(PinRojoAutos, OUTPUT);       //Semáforo Auto
16    TiempoInicial = millis( );    // Se toma el Tiempo Inicial
17    Procesando = Verde;          // Color por el que comenzará el Semáforo
18  }
-
19  void loop( ) {
20   AutoVerde( );
21   AutoAmarillo( );
22   AutoRojo( );
23  }
-
24  void AutoVerde( ){
25    if(Procesando == Verde){
26      if(millis( ) - TiempoInicial <= TiempoVerdeAuto){
27        digitalWrite(PinVerdeAutos, HIGH); // Enciende el LED Verde Autos
```

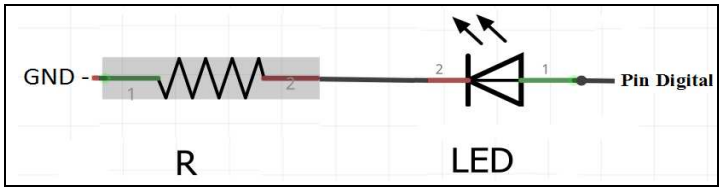
```
28        }else{
29          digitalWrite(PinVerdeAutos, LOW);  // Apaga el LED Verde Autos
30          Procesando = Amarillo;      // Cambia Próximo Color
31          TiempoInicial = millis( );    // Tiempo inicial próximo Color
32        }
33      }
34    }
-
35    void AutoAmarillo( ){
36      if(Procesando == Amarillo){
37        if( millis( ) - TiempoInicial <= TiempoAmarilloAuto){
38          digitalWrite(PinAmarilloAutos, HIGH); // Enciende el LED Amarillo Autos
39        }else{
40          digitalWrite(PinAmarilloAutos, LOW);  // Apaga el LED Amarillo Autos
41          Procesando = Rojo;          // Cambia Próximo Color
42          TiempoInicial = millis( );  // Tiempo inicial próximo Color
43        }
44      }
45    }
-
46    void AutoRojo( ){
47      if(Procesando == Rojo){
48        if( millis( ) - TiempoInicial <= TiempoRojoAuto){
40          digitalWrite(PinRojoAutos, HIGH);   // Enciende el LED Rojo Autos
50        }else{
51          digitalWrite(PinRojoAutos, LOW);    // Apaga el LED Rojo Autos
52          Procesando = Verde;         // Cambia Próximo Color
53          TiempoInicial = millis( );    // Tiempo inicial próximo Color
54        }
55      }
56    }
```

Explicación General del Funcionamiento

Este programa tiene el mismo funcionamiento que el Anterior, solo que en vez de dos procesemos (Subir y Bajar) este tiene tres, que controlan los colores del semáforo.
Analice bien a fondo el funcionamiento de este programa, ya que más adelante reutilizaremos el código en un semáforo inteligente (Activación mediante un botón para el cruce de peatones).

Construcción Del Circuito





Lista de Materiales:

- 1 Placa Protoboard.
- 3 Led.
- 3 Resistencia de 470Ω.
- 4 Cables Macho-Macho.
- Placa Arduino.
- Cable USB

En Cuanto al circuito para prender un LED, repetiremos tres veces el mismo, conectar el Cable Azul a GND, y los otros cables que pertenecen a los LED, conectar a los Pines configurados en el comienzo del programa en las CONSTANTES: "**PinVerdeAutos**", "**PinAmarilloAutos**", "**PinRojoAutos**".

RECORDAR: GND es el PIN considerado como Negativo o Masa en la Placa Arduino. Si buscamos veremos que hay uno o varios pines así identificados (La cantidad dependerá del modelo de placa que estemos analizando).

12- Semáforo para Autos y Peatones. Programación Multitarea - Sin usar la función delay( ).

Implementación de un semáforo para autos y peatones. Más adelante implementaremos este mismo semáforo pero incluyendo un botón que otorga la prioridad a Peatones y el sonido de alerta para No Videntes.

(Programa "001_Led_07_Semaforo_02")

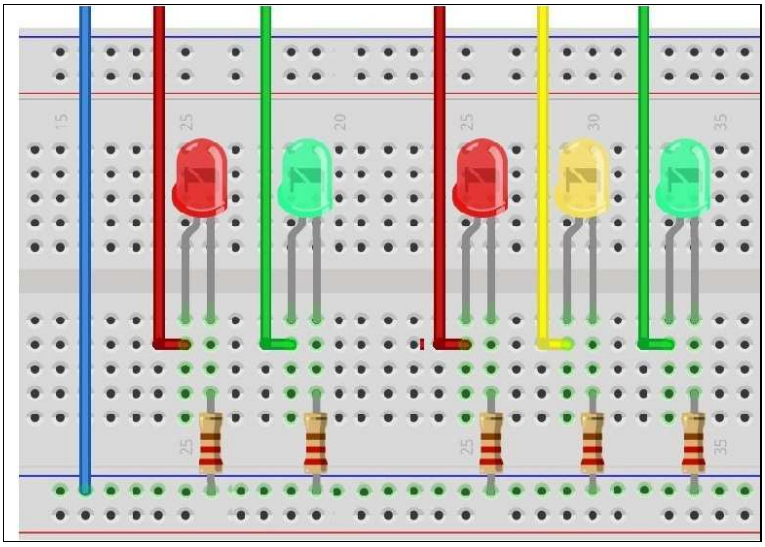
```
1   const int PinVerdeAutos      =  5;
2   const int PinAmarilloAutos =  7;
3   const int PinRojoAutos       =  9;
-
4   const int PinVerdePeaton    =  14;
5   const int PinRojoPeaton      =  16;
-
6   int Verde     = 1;
7   int Amarillo = 2;
8   int Rojo       = 3;
9   int Procesando;
-
10  long TiempoVerdeAuto     = 7000;  // Milisegundos en Verde
11  long TiempoAmarilloAuto = 2000;  // Milisegundos en Amarillo
12  long TiempoRojoAuto      = 7000;  // Milisegundos en Rojo

13  boolean SI   =  true;
14  boolean NO  = false;
-
15  long TiempoInicial = 0;
-
16  void setup( ) {
17    Serial.begin(9600); // Abre el Puerto Serie Para Enviar y Recibir Mensajes.
18    pinMode(PinVerdeAutos, OUTPUT);      //Semáforo Auto
19    pinMode(PinAmarilloAutos, OUTPUT);  //Semáforo Auto
20    pinMode(PinRojoAutos, OUTPUT);       //Semáforo Auto
21    pinMode(PinVerdePeaton, OUTPUT);     //Semáforo Peatón
22    pinMode(PinRojoPeaton, OUTPUT);      //Semáforo Peatón
23    TiempoInicial = millis( );
24    Procesando = Verde;
25  }
26  void AutoVerde( ){
27    if(Procesando == Verde){
28      if(millis( ) - TiempoInicial <= TiempoVerdeAuto){
29        digitalWrite(PinVerdeAutos, HIGH); // Enciende LED Verde Autos
30        digitalWrite(PinRojoPeaton, HIGH); // Enciende ROJO PEATON
31      }else{
32        digitalWrite(PinVerdeAutos, LOW); // Apaga LED Verde Autos
33        digitalWrite(PinRojoPeaton, LOW); // Enciende LED ROJO PEATON
34        Procesando = Amarillo;
```

```
35          TiempoInicial = millis( );
36        }
37      }
38    }
39    void AutoAmarillo( ){
40      if(Procesando == Amarillo){
41        if( millis( ) - TiempoInicial <= TiempoAmarilloAuto){
42          digitalWrite(PinAmarilloAutos, HIGH);  // Enciende LED Amarillo Autos
43          digitalWrite(PinRojoPeaton, HIGH); // Enciende LED AMARILLO Peatón
44        }else{
45          digitalWrite(PinAmarilloAutos, LOW); // Apaga LED Amarillo Autos
46          digitalWrite(PinRojoPeaton, LOW); // APAGA AMARILLO Peatón
47          Procesando = Rojo;
48          TiempoInicial = millis( );
40        }
50      }
51    }
52    void AutoRojo( ){
53      if(Procesando == Rojo){
54        if( millis( ) - TiempoInicial <= TiempoRojoAuto){
55          digitalWrite(PinRojoAutos, HIGH); // Enciende LED Rojo Autos
56          digitalWrite(PinVerdePeaton, HIGH); // Enciende LED Verde Peatón
57        }else{
58          digitalWrite(PinRojoAutos, LOW); // Apaga LED Rojo Autos
59          digitalWrite(PinVerdePeaton, LOW); // Apaga LED Verde Peatón
60          Procesando = Verde;
61          TiempoInicial = millis( );
62        }
63      }
64    }
-
65    void loop( ) {
66      AutoVerde( );
67      AutoAmarillo( );
68      AutoRojo( );
69    }
```

Explicación General del Funcionamiento

Este programa tiene el mismo funcionamiento que el Anterior, solo que se agrega el manejo de las luces para peatón. Analice bien a fondo el funcionamiento de este programa, ya que más adelante reutilizaremos el código en un semáforo inteligente (Activación mediante un botón para el cruce de peatones y Sonido para No Videntes).

Construcción Del Circuito

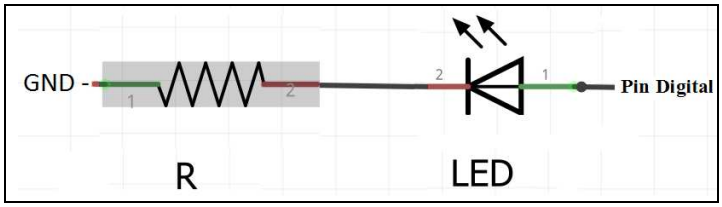


Lista de Materiales:

- 1 Placa Protoboard.
- 5 Led.
- 5 Resistencia de 470Ω.
- 6 Cables Macho-Macho.
- Placa Arduino.
- Cable USB

De izuierda a derecha: Cable Azul (Conectar a GND), Luego los cables del Semáforo, es decir el rojo y Verde del Peatón, y finalmente Rojo Amarillo y verde de los Autos, conectar a los Pines configurados al comienzo del programa en las CONSTANTES: "**PinVerdeAutos**", "**PinAmarilloAutos**", "**PinRojoAutos**", "**PinVerdePeaton**" y "**PinRojoPeaton**".

En Cuanto al circuito para prender un LED, repetiremos Cinco veces el mismo Circuito.



RECORDAR: GND es el PIN considerado como Negativo o Masa en la Placa Arduino. Si buscamos veremos que hay uno o varios pines así identificados (La cantidad dependerá del modelo de placa que estemos analizando).

13- Interactuando con la PC – (Primera Parte) - Dar la orden desde la PC para que la Placa Prenda un LED (Puerto Serie). Programación Multitarea.

Comando	Prende	Apaga
A	Led 01	---------
B	---------	Led 01
C	Led 02	---------
D	---------	Led 02
E	Led 03	---------
F	---------	Led 03

Enviar comandos desde la Computadora que permitan Seleccionar, Prender o Apagar un LED entre tres. Ver Comandos. En caso de recibir un Comando Erróneo, se debe informar a través un mensaje por el puerto Serie.
Esta forma de trabajar, nos abre una importantísima vía de comunicación con la placa en el momento que el programa se esta ejecutando. Analícela detenidamente ya que nos resultará muy útil.

(Programa "001_Led_08_Leo_Puerto_Serie_01")

```
1   const int PinLed_01 = 4;
2   const int PinLed_02 = 6;
3   const int PinLed_03 = 8;
4   char dato;
-
5   void setup ( ){
6     Serial.begin (9600);       //Iniciamos la comunicación serial.
7     pinMode(PinLed_01,OUTPUT); //Inicializamos los pines que utilizaremos.
8     pinMode(PinLed_02,OUTPUT); //Inicializamos los pines que utilizaremos.
9     pinMode(PinLed_03,OUTPUT); //Inicializamos los pines que utilizaremos.
10  }
-
11  void loop ( ){
12    dato=Serial.read( ); //Se lee un dato enviado por Puerto Serie y guarda en "dato".
```

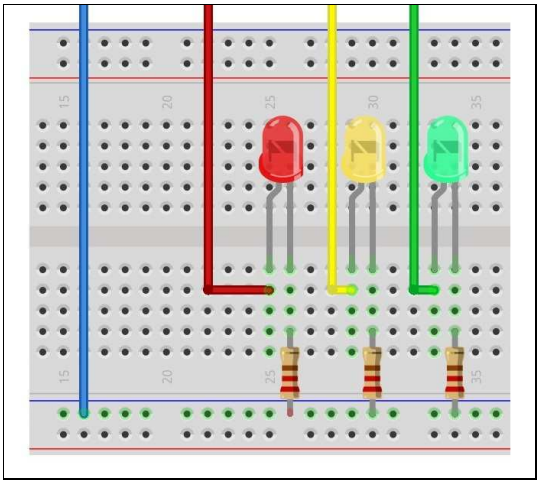
```
13  if(dato > 0){ // Si Se pudo leer Algo (Todo caractér tiene Código > Cero)
14    if(dato == 'A' || dato == 'a'){
15          digitalWrite(PinLed_01,HIGH);
16          Serial.println("Led 01 encendido");
17    }else if(dato == 'B' || dato == 'b'){
18          digitalWrite(PinLed_01,LOW);
19          Serial.println("Led 01 apagado");
20    }else if(dato == 'C' || dato == 'c'){
21          digitalWrite(PinLed_02,HIGH);
22          Serial.println("Led 02 encendido");
23    }else if(dato == 'D' || dato == 'd'){
24          digitalWrite(PinLed_02,LOW);
25          Serial.println("Led 02 Apagado");
26    }else if(dato == 'E' || dato == 'e'){
27          digitalWrite(PinLed_03,HIGH);
28          Serial.println("Led 03 encendido");
29    }else if(dato == 'F' || dato == 'f'){
30          digitalWrite(PinLed_03,LOW);
31          Serial.println("Led 03 Apagado");
32    }else {
33          int d = dato;
34          Serial.print("Comando Incorrecto: ");
35          Serial.print(d);
36          Serial.print(" - ");
37          Serial.println(dato);
38    } // Finaliza "else" del ultimo "if"
39   } // Finaliza "if" - Donde se controla si se ha leído algo
40   // Otras Tareas Distintas
41   // Otras Tareas Distintas
42  }
```

IMPORTANTE RECORDAR: Un Pin, es una salida de la Placa Arduino, donde conectamos (enchufamos) un cable del circuito.

Explicación Líneas Importantes

6	Iniciamos la comunicación serial. Esto nos permitirá, mientras el programa se esta ejecutando, recibir datos o enviarle comando.
12	Leo el Puerto Serie.
13	Si Se leyó algo del puerto serie, se retorna un carácter con código mayor que cero, de lo contrario, en la variable habrá un carácter con código **menos uno** "-1"
14	Desde acá, analizo el dato leído. En este caso pregunto si se trata de una "A".
15	Prendo el LED conectado a "**PinLed_01**".
16	Envió un mensaje por puerto serie a la Computadora.
33 al 38	Declaro una Variable entera "d" y le asigno el carácter leído, esto hace que se guarde el código del carácter, luego lo mostrare por el puerto serie. Notar que al momento de enviar el comando uso "**Serial.print**(" sin el "**ln**" al final. Esto hace que no baje de renglón al mostrarlo. Y en conjunto con los comandos que siguen, armo una línea. Ejecute el programa y analice el resultado que visualiza al enviar un comando erróneo.
39	Finaliza "**else**" del ultimo "**if**"
40	Finaliza "if" - Donde se controla si se ha leído algo desde el Puerto serie
41 y 42	Otras Tareas Distintas
43	Fin de la Función "**loop( )**"

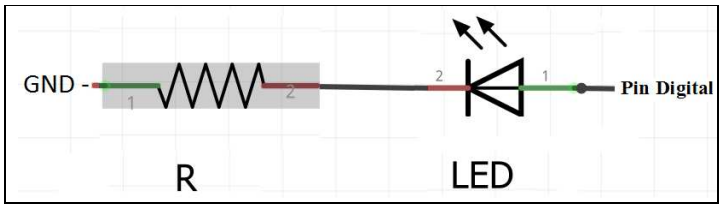
CONSTRUCCIÓN DEL CIRCUITO



Lista de Materiales:

- 1 Placa Protoboard.
- 3 Led.
- 3 Resistencia de 470Ω Para LEDs.
- 4 Cables Macho-Macho.
- Placa Arduino.
- Cable USB

De izuierda a derecha: Cable Azul (Conectar a GND), Luego los cables Rojo, Amarillo y Verde, se deben conectar a los Pines configurados al comienzo del programa en las CONSTANTES: "**PinLed_01**", "**PinLed_02**" y "**PinLed_03**".



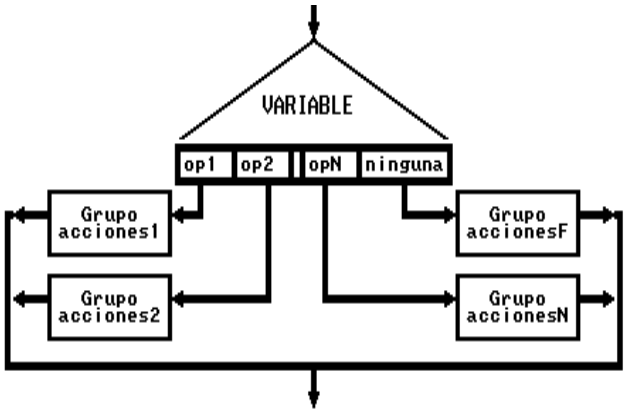
En Cuanto al circuito para prender un LED, repetiremos Tres veces el mismo Circuito.

RECORDAR: GND es el PIN considerado como Negativo o Masa en la Placa Arduino. Si buscamos veremos que hay uno o varios pines así identificados (La cantidad dependerá del modelo de placa que estemos analizando).

Estructura Condicional Switch( )

La programación Arduino, incorpora una sentencia de selección con múltiples opciones: **switch( )** , que compara sucesivamente el valor de una expresión, con una lista de opciones**.** Las opciones deben ser del tipo enteras "**int**" o de cadenas "**string**". Cuando encuentra una correspondencia, se ejecuta el bloque de acciones asociadas a la constante. A continuación presentamos el **símbolo usado para la diagramación**, y su **traducción o codificación** al lenguaje.

El **switch** permite la selección de una posibilidad entre un abanico de varias opciones. Se comprueba el valor de la *expresión*, por orden, con los valores de las constantes especificadas en las sentencias **case**. Cuando se encuentra una correspondencia, como se dijo antes, se ejecuta la secuencia de sentencias asociadas con el **case,** hasta la primera sentencia **break**; en caso de no encontrarla, se ejecutará hasta el final de la estructura **switch**.




```
switch (variable) {
     case op1:{ bloque de acciones1
       }break;
     case op2:{ bloque de acciones2
       }break;
           .
           .
    case opN: { bloque de accionesN
       }break;
     default: { bloque de accionesF
       }break;
 } /* fin del switch */
```

La sentencia **default,** se ejecuta si no se ha encontrado ninguna correspondencia (Representa el "**else**" de la estructura "**if**"). Esta sentencia es opcional y si no está presente, no se ejecutará ninguna acción al fallar

todas las opciones. Debe quedar claro que este tipo de estructura también puede anidarse (Tener otras estructuras en cada una de las opciones, incluso un "**switch**"), dependiendo de las necesidades del programa.

Veamos un ejemplo. Cargue el programa y confeccione el circuito y vea como funciona. Ya que lo estudiaremos:

14- Interactuando con la PC – (Segunda Parte) - Dar la orden desde la PC para prender o apagar un LED (Puerto Serie). Ejemplo estructura switch( ).

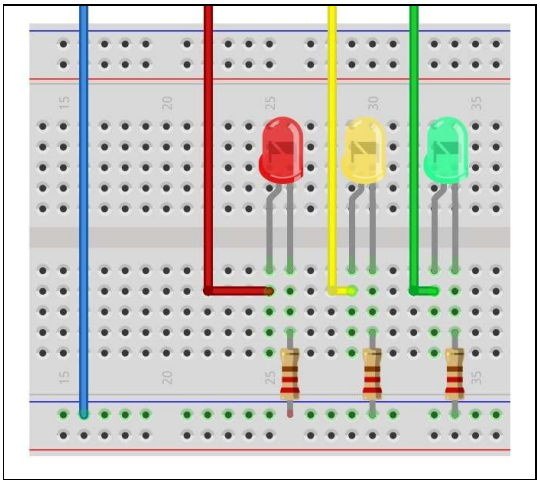
Este ejercicio, ya lo habíamos resuelto, pero usando "if". Enviar comandos desde la Computadora que permitan Seleccionar, Prender o Apagar un LED entre tres. En caso de recibir un Comando Erróneo, el programa debe informar mediante un mensaje por el puerto Serie.

(Programa "001_Led_08_Leo_Puerto_Serie_02_switch)

```
1   const int PinLed_01 = 4;
2   const int PinLed_02 = 6;
3   const int PinLed_03 = 8;
-
4   char dato;
-
5   void setup ( ){
6     Serial.begin (9600);       //Iniciamos la comunicación serial.
7     pinMode(PinLed_01,OUTPUT); //Inicializamos los pines que utilizaremos.
8     pinMode(PinLed_02,OUTPUT); //Inicializamos los pines que utilizaremos.
9     pinMode(PinLed_03,OUTPUT); //Inicializamos los pines que utilizaremos.
10  }
-
11  void loop ( ){
12    dato=Serial.read( ); //Se lee la variable enviada por el monitor del puerto Serie.
13    if(dato > 0){ // Si Se pudo leer Algo
14      switch (dato){ //Seleccionamos el caso, dependiendo del caractér recibido.
15        case 'A': {
16          digitalWrite(PinLed_01,HIGH);
17          Serial.println("Led 01 encendido");
18        } break;
19        case 'B': {
20          digitalWrite(PinLed_01,LOW);
21          Serial.println("Led 01 apagado");
22        } break;
23        case 'C': {
24          digitalWrite(PinLed_02,HIGH);
25          Serial.println("Led 02 encendido");
26        } break;
27        case 'D': {
28          digitalWrite(PinLed_02,LOW);
29          Serial.println("Led 02 Apagado");
30        } break;
31        case 'E': {
32          digitalWrite(PinLed_03,HIGH);
33          Serial.println("Led 03 encendido");
34        } break;
35        case 'F': {
36          digitalWrite(PinLed_03,LOW);
```

```
37              Serial.println("Led 03 Apagado");
38            } break;
39            default: {
40              int d=dato;
41              Serial.print("Comando Incorrecto: ");
42              Serial.print(d);
43              Serial.print(" - ");
44              Serial.println(dato);
45            } break;
46        } // Finaliza switch
47    } // Finaliza if - Donde se controla si se ha leído algo
48    // Otras Tareas Distintas
49  }
```

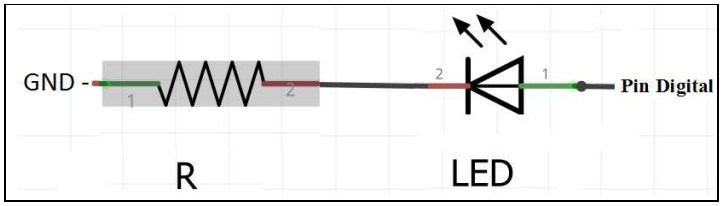
CONSTRUCCIÓN DEL CIRCUITO



Lista de Materiales:

- 1 Placa Protoboard.
- 3 Led.
- 3 Resistencia de 470Ω Para LEDs.
- 4 Cables Macho-Macho.
- Placa Arduino.
- Cable USB

De izuierda a derecha: Cable Azul (Conectar a GND), Luego los cables Rojo, Amarillo y Verde, se deben conectar a los Pines configurados al comienzo del programa en las CONSTANTES: "**PinLed_01**", "**PinLed_02**" y "**PinLed_03**".



En Cuanto al circuito para prender un LED, repetiremos Tres veces el mismo Circuito.

RECORDAR: GND es el PIN considerado como Negativo o Masa en la Placa Arduino. Si buscamos veremos que hay uno o varios pines así identificados (La cantidad dependerá del modelo de placa que estemos analizando).

15- Como detectar desde Arduino, cuando tenemos DATOS para leer que ingresan por el puerto Serie.

Serial.available( ): Obtiene el número de bytes (caracteres) disponibles para su lectura desde el puerto serie. Se trata de datos que ya llegaron y se almacenaron en el buffer de recepción del puerto serie (tiene 64 bytes) a la espera de ser leídos.

Parámetros de la Función: Ninguno

Retornos de la Función: bytes disponibles para leer, pueden ser Caracteres o números.

(Programa "100_Puerto_Serie_Available")

```
1   int ByteEntrante = 0;   // Para los datos de entrada
-
2   void setup( ) {
3     Serial.begin(9600);     // abre el puerto Serie, configura los datos a 9600 bps
4   }
-
```

```
5   void loop( ) {
6     if (Serial.available( ) > 0) {
7       ByteEntrante = Serial.read( );      // lee el byte de entrada:
8       Serial.print("Recibido: ");
9       Serial.println(ByteEntrante, DEC);
10      // Acá proceso la información recibida
11    }
12  }
```

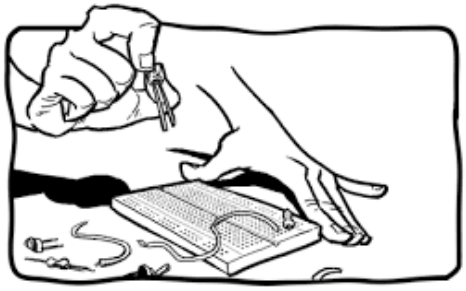
16- RECEPCIÓN DE NÚMEROS ENTEROS (PC envía Arduino recibe).

El código para recibir Números eneros, es similar al de recepción de Letras, sin embrago, hago notar que al enviarse los datos caracteres ASCII, debemos restar el valor ‘0’ al dato (numero como caractér) recibido para recuperar el valor numérico.

(Programa “100_Puerto_Serie_Recepcion_Numeros_01”)

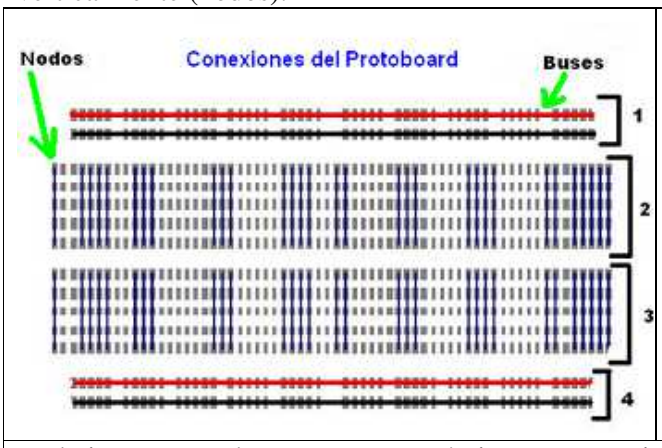
```
1   int Numero;
2   char Caracter;
-
3   void setup( ){
4     Serial.begin(9600);
5   }
-
6   void loop( ){
7     if (Serial.available( ) > 0){  //si existe información pendiente de ser leída
8       Caracter = Serial.read( );    //leemos Puerto
9       if (Caracter >= '1' && Caracter <= '9'){ //Si es Numero – Caractér esta entre '1' y '9'
10        Numero = Caracter - '0'; //restamos el valor '0' para obtener el numero enviado
-
11        Serial.print("Recibido Caractér: ");
12        Serial.print(Caracter);
13        Serial.print(" - Nro Transformado: ");
14        Serial.println(Numero);
15        // ACA HAGO LO QUE DEBO HACER
16        // ACA HAGO LO QUE DEBO HACER
17        // ACA HAGO LO QUE DEBO HACER
-
18      }
18    }
20  }
```

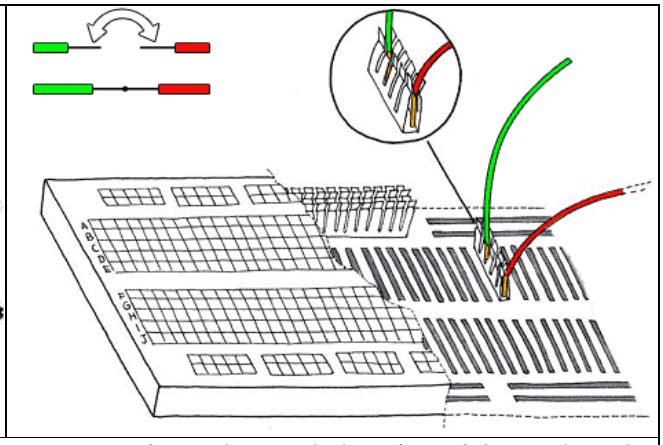
Apéndice A - Como usar una Placa PROTOBOARD



Acá construiremos los circuitos que, conectados a nuestra placa Arduino, controlaremos por medio del programa.
Mas adelante usaremos placas pre impresas, por ahora, mientras aprendemos usaremos las placas Protoboard. Primero entandamos como usarlas.
Protoboard o también “Bread Borrad”, es una placa de pruebas en la que no se utiliza soldadura para conectar los componentes electrónicos. Las conexiones se hacen por medio de láminas metálicas que ejercen presión sobre los terminales de los componentes, que abarcan zonas específicas en el

mismo. Del lado superior se encuentran una serie de agujeros dispuestos horizontal (buses) y verticalmente (nodos).





En la imagen puede verse como están internamente interconectados cada uno de los pines del Protoboard

01000001 01101110 01101001 01101101 01101111 00100001

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