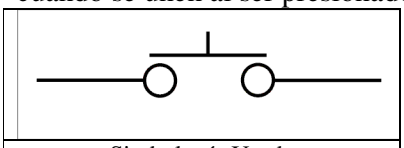
BOTON PULSADOR

Un pulsador es un tipo de interruptor eléctrico, que consta de dos contactos metálicos separados que cuando se unen al ser presionado el botón, permiten el paso de corriente.



Simbología Usada

Existen muchos modelos de pulsadores, en la foto podemos ver el modelo más común en casas de electrónica de la zona. Sin embrago podemos conseguir otros formatos y colores.

Imagen de un Botón Pulsador

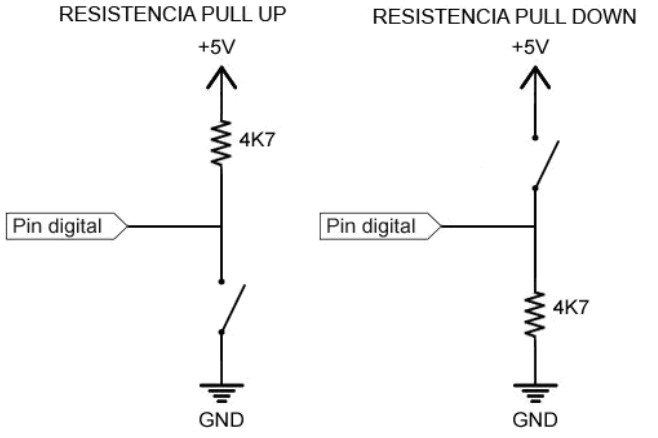
Los pulsadores son dispositivos que tienen un defecto llamado rebote. Cuando se presiona o se suelta el pulsador, se produce una fluctuación entre sus contactos internos, por lo tanto cuando se va a pasar de un 1 (HIGH) a un 0 (LOW) o viceversa, esas fluctuaciones son también leídas por el Arduino, produciendo un comportamiento inesperado en el funcionamiento de nuestros proyectos. Por ejemplo, el usuario puede presionar una sola vez el pulsador pero por culpa del rebote, o a la velocidad de proceso que tiene la placa Arduino, el programa podría interpretarlo como si se hubiese presionado varias veces.

Se puede crear una ***rutina anti-rebote*** (SoftWare) muy efectiva, aunque no infalible y con el hardware hacerla muy efectiva.

A continuación la rutina, que debes comprender y resultara muy simple, pero insisto, debe ser entendida y adaptada en cada caso en particular. (**Detectamos cuando dejamos de presionar el botón**)

```
int presionado = 0;
int PinDigital = 10; // Pin Digital donde conectaremos el Pulsador (Ver imagen)
void loop( ){
   if (digitalRead( pulsador ) == LOW) { //Pregunta si el pulsador está presionado
      presionado = 1;     //La variable cambia de valor
   }
   if ( digitalRead(pulsador) == HIGH && presionado == 1){ //Si ya NO está presionado
      //ACÁ debe realizar la acción deseada
      presionado = 0;    //La variable vuelve a su valor original-
   }
} // Final de la función loop
```

FORMAS DE REALIZAR CONEXIONES: Ahora Bien, al momento de hacer las conexiones, hay dos formas de hacerlo, mostramos las dos, solo depende de que lado colocamos GND y los 5V.



Esquema de Conexiones. **IMPORTANTE:** El denominado PIN_DIGITAL es el pin que usamos en Arduino para detectar que ha pasado con el Pulsador (Presionado o No Presionado)

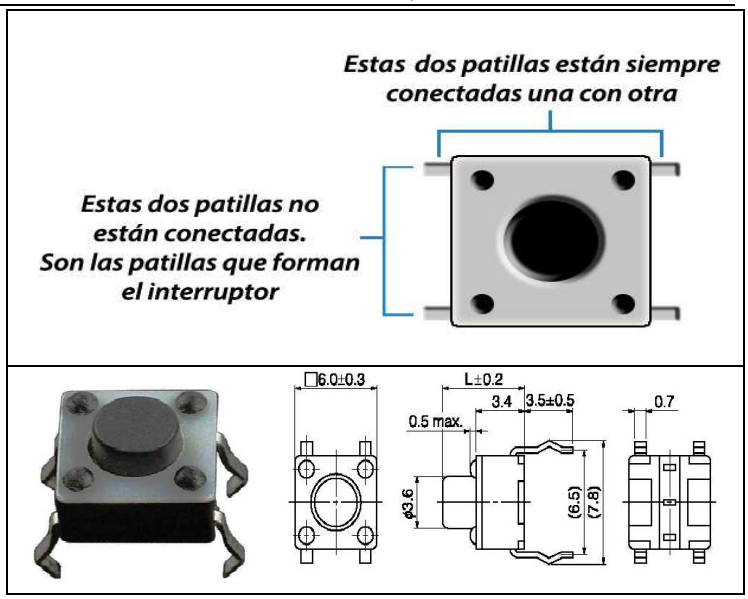
A cada tipo de conexión se le da un nombre, así podremos identificar bien cada caso:

- **Pull Up:** La resistencia "Pull Up" garantiza un valor HIGH a la entrada de Arduino cuando el pulsador está abierto. Cuando está cerrado el pulsador, la entrada digital estará a un nivel LOW.
- **Pull Down:** La resistencia "Pull Down" garantiza un valor LOW a la entrada de Arduino cuando el pulsador está abierto. Cuando está cerrado el pulsador, la entrada digital estará a un nivel HIGH.

Veamos ahora el programa que nos preemitirá detectar cuando el Botón Pulsador ha sido presionado.

Para conectar los de 2 patas es muy simple, por una pata ingresa la corriente y por la otra sale cuando el botón es presionado y se cierra el circuito. Pero cuando tiene 4 patas, solo hay que prestar un poco atención:

Primero veamos el botón de frente y encontraremos que hay dos patas que salen para atrás y dos patas que salen para adelante. Ahora viendo el botón de frente, las dos patas de la derecha están interconectadas (adelante y atrás), y tendrán la misma función, que podremos suponer "Entrada" y el par de patas de la derecha (adelante y atrás), también están interconectadas entre si y podremos decir que dejarán salir la corriente cuando el circuito se cierra (Botón Presionado).



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).

FUNCIONES QUE UTILIZAREMOS EN PROXIMO PROYECTO.

- **digitalRead(pin)**: Lee el valor desde del un **PIN** digital específico. Devuelve un valor HIGH o LOW. El número de **PIN** puede ser especificado con una variable o una constante.

1- Reconocimiento Del Estado De Un Botón Pulsador. Programación Multitareas.

Reconocer el estado de un Botón Pulsador, eliminando el rebote de señal. Esto es la base para futuros desarrollos. Usaremos en este caso el formato construido con resistencia **PULL UP**.

(Programa: Botón_01)

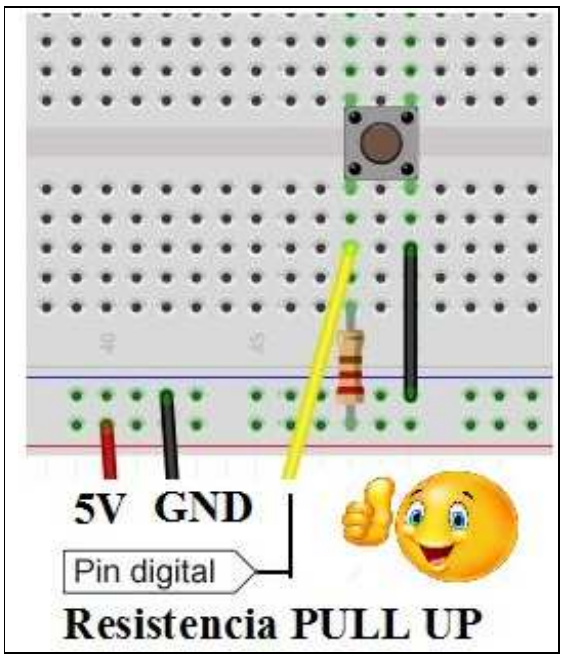
```
1   int PinDigital = 10;   // Pin Digital donde conectare el Pulsador
2   int EstadoBoton = 0;
3   int presionado = 0;
-
4   void setup( ) {
5     Serial.begin(9600);
6     pinMode(PinDigital, INPUT);
7   }
-
8   void loop( ){
-   /****** Detecto cuando se presiona Botón *********************/
9     EstadoBoton = digitalRead(PinDigital); // Leemos el PinDigital.
10    if (EstadoBoton == LOW) { //Pregunta si el pulsador está presionado
11       presionado = 1;     //La variable cambia de valor
12    }
-   /********* Detecto cuando Suelta Botón ***********************/
13    EstadoBoton = digitalRead(PinDigital); // Leemos el PinDigital.
14    if (EstadoBoton == HIGH && presionado == 1) {
15       presionado = 0;    //La variable vuelve a su valor original
16       Serial.println("Botón ------> ON");
17    } else { // Si el valor es bajo:
18       Serial.println("Botón OFF");
19    }
-
20    delay(200); // Detenemos el Programa (Solo para ver bien que pasa).
21  } // Finaliza función loop( )
```

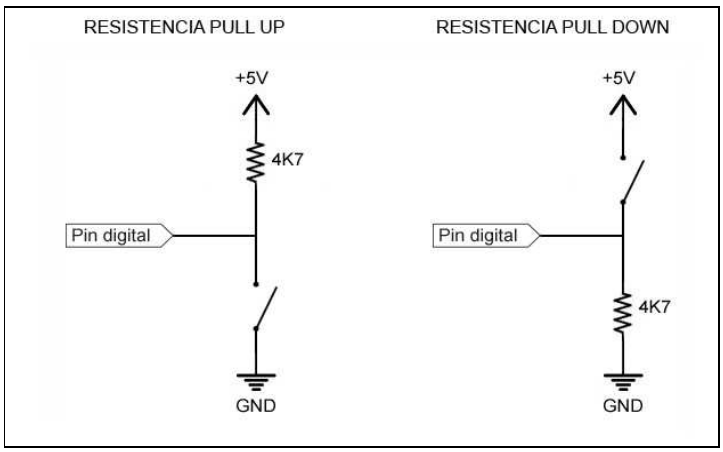
Explicación, Línea Por Línea	
1	Declaración de Variable, a la que asignamos el número de PIN, que usaremos en la placa Arduino.
2	Variable que usaremos para almacenar el estado del botón
3	Variable que usaremos para Eliminar el robote.
4	Comienza función de Configuración "**setup( )**".
5	Abro el puerto Serial. Esto me permite ver cuando se presiona el Botón.
6	Definimos el "**PinDigital**" como entrada. Para nuestro ejemplo será el Pin numero 10.
7	Llave que finaliza la función de Configuración "**setup( )**".
8	Cabecera e inicio de la Función "**loop( )**". Recordar que esta función, es quien permite que las acciones del programa se repitan indefinidamente.
9	Leemos el Pin configurado en la Variable "**PinDigital**".
10	Pregunto por el Estado "**EstadoBoton == LOW**" - La respuesta dependerá de cual circuito se haya implementado: "**Pull Up**" o "**Pull Down**"
11	Prendo Variable "**presionado**". (Recordar que la placa es muy veloz y en el tiempo que nosotros presionamos una vez el Botón, ella puede haber leído el pin al que esta conectado el Botón muchas veces)
12	Termina Pregunta de Estado del Botón, iniciada en línea 10.
13	Leemos nuevamente el Pin configurado en la Variable "**PinDigital**".
14	Preguntamos por el estado del botón (Leído en línea anterior) – Si ya No Esta presionado y además se pregunta si antes Estaba presionado – Esto es para asegurarnos que el botón se presiono (líneas 10/11/12) y ya se soltó – Eliminando así la posibilidad de múltiples lecturas de estado mientras se presiona o que se trata de un rebote.
15	Si Se cumplió la condición (línea 14) – Estoy seguro que el botón ha sido presionado, entonces la variable vuelve a su valor original (Habilito para la próxima).
16	Muestro en monitor, por Puerto Serie, mensaje botón esta "ON". (Botón Presionado)
17	Termina la Parte verdadera y comienza la parte Falsa del "if" que comenzó en línea 14
18	Muestro por el monitor del Puerto Serie, el mensaje que el botón esta "ON". (Botón Desactivado o NO Presionado)
19	Termina Parte Falsa que comenzó en línea 17
20	Detenemos el Programa por 200 Milisegundos (Para ver bien que pasa en el monitor).
21	Llave que marca el final de la función "**loop( )**". Que comenzó en la línea 7

Al construir nuestro proyecto con un "Botón Pulsador", siempre hay que recordar la resistencia!

CIRCUITO PARA EL PROYECTO

Lista de Materiales: 1 Botón Pulsador de 2 Patas – 1 Resistencia de 4K7 Ω – 4 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.





El circuito fue construido con resistencia PULL UP, sin embargo, deberías, luego de realizar las pruebas, cambiar el diseño a PULL DOWN y analizar los resultados obtenidos. Se muestran ambos diseños. Es muy Buena oportunidad para ganar experiencia con estos circuitos.

PARA PENSAR: Si el circuito es PULL UP, **porque** la secuencia de conexión, es igual dibujo esquemático del circuito:

5V – Resistencia – Pin – PULSADOR - GND

Cuando ya este en funcionamiento, abrir el puerto Serial. Esto permite ver los mensajes que emitimos cuando se presiona el Botón. Recordar que para activar el Monitor del Puerto serie, hay que abrir en el IDE (donde escribió el programa que envió a la placa Arduino), en el Menú "**Herramientas**" y elegir la opción "**Monitor Serie**".

Si Intercambia en el Protoboard, el lugar donde conecta los cables de 5V y GND, en que se transforma el circuito?

2- Prender un LED cuando Botón Pulsador es Accionado (Primera Parte).

En este ejemplo, prenderemos un LED cuando detectemos que un interruptor ha sido presionado (accionado). Dicho en otras palabras, **cuando el botón es presionado, y mientras este presionado**, se prendera el LED (entonces en este caso, no hace falta controlar el Rebote).

(Programa 002_Boton_02_Led_Alarma_01)

```
1  int ledAlarmaPin = 5;      // Pin de salida para el LED
2  int interruptorPin =  3;   // Pin de entrada ( Acá está conectado el interruptor )
3  void setup( ) {
4    pinMode(ledAlarmaPin, OUTPUT);
5    pinMode(interruptorPin, INPUT);
6-  }
7  void loop( ) {
8    if (digitalRead( interruptorPin ) == HIGH){
9      digitalWrite(ledAlarmaPin, HIGH); // Enciende el LED
10     delay(250);                       // Pause de ¼ de segundo
11     digitalWrite(ledAlarmaPin, LOW);  // Apaga el LED
12     delay(250);                       // Pausa de ¼ de segundo
13   }
14 }  // Finaliza función loop( )
```

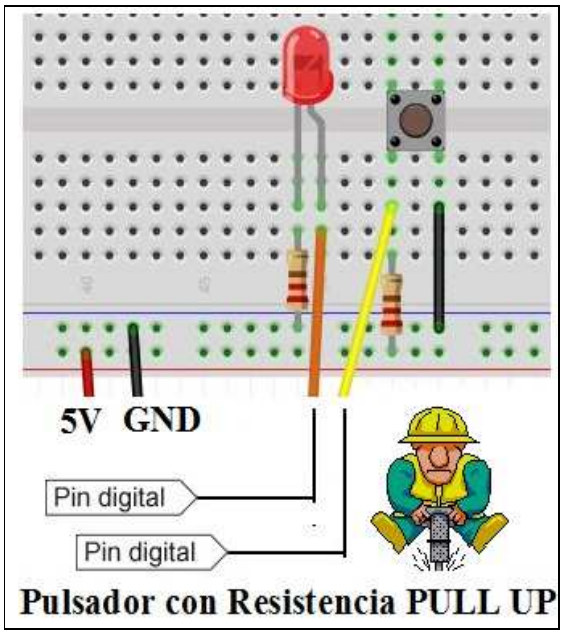
Explicación, Línea Por Línea	
1 y 2	Declaración de Variables y asignación de los valores que luego corresponderán a los PINES que usaremos. Las variables son: "**led**Alarma**Pin**" identificará el PIN al que se conectara el LED de alarma, y la variable "**interruptorPin**" que identifica al PIN en el que conectaremos el interruptor que luego controlaremos.
3	Comienza función de Configuración "**setup( )**".
4	En esta línea se configura el PIN identificado por la variable "**led**Alarma**Pin**", y por el saldrá la corriente que prendara el LED usado para avisar que el interruptor no esta dejando pasar corriente, se configura de salida "**OUTPUT**"
5	En esta línea se configura el PIN identificado por la variable "**interruptorPin**", en el que se conecta el interruptor que controlaremos. Se configura de entrada "**INPUT**".
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	En esta línea de código, la interpretación de la pregunta puede variar, según sea la conexión del interruptor (Resistencia PULL UP o Resistencia PULL DOWN). **Releer "FORMAS DE REALIZAR CONEXIONES" de un Botón Pulsador.** **Muy IMPORTANTE:** Si el botón se conecta con Resistencia "PULL UP", la pregunta será Verdadera cuando el Botón Pulsador **NO este Siendo Presionado** (Titila

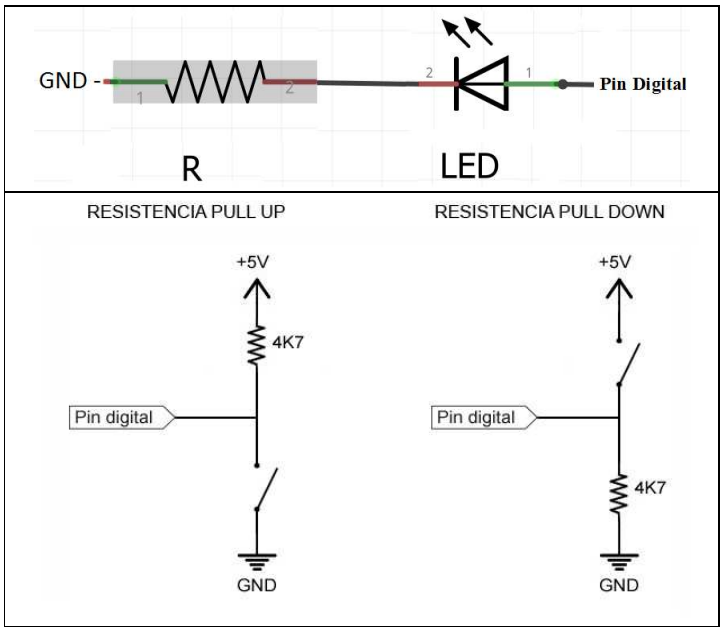
	el LED de Alarma). Por el contrario Si se conecta del Modo "PULL DOWN" el LED Titilara cuando el Botón Pulsador SI este Siendo Presionado.
9	Enciende el LED conectado al PIN indicado por la variable "**led**Alarma**Pin**"
10	Pausa de 1 segundo
11	Apaga el LED conectado al PIN indicado por la variable "**led**Alarma**Pin**"
12	Pausa de 1 segundo
13	Llave que cierra la parte verdadera de la pregunta realizada en línea 8.
14	Llave que marca el final de la función "**loop( )**". Que comenzó en la línea 7

Al construir un proyecto con "Botón Pulsador", siempre hay que recordar la resistencia!

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Botón Pulsador de 2 Patas – 1 LED - 1 Resistencia de 4K7 Ω (Para el Boton) – 1 Resistencia de 470Ω (Para el Led) – 5 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.





Muy Importante: Si el botón se conecta tal como se muestra en la imagen (Resistencia PULL UP), la pregunta de la línea 8 será Verdadera cuando el Botón Pulsador NO este Siendo Presionado (Titila el LED). Por el contrario Si se conecta del Modo "PULL DOWN" el LED Titilara cuando el Botón Pulsador SI este Siendo Presionado.

Si Intercambia en el Protoboard, los lugares donde conecta cables de 5V y GND, Transformará la Conexión en una "PULL DOWN". Hágalo y analice como responderá el LED?

Puede agregar mensajes para realizar un seguimiento del funcionamiento de su programa. Entonces cuando ya este en funcionamiento, podrá ver los mensajes que usted inserte, cuando se presiona el Botón. Recordar que para activar el Monitor del Puerto Serie, hay que abrir en el IDE (donde escribió el programa que envió a la placa Arduino), en el Menú "**Herramientas**" y elegir la opción "**Monitor Serie**".

3- Prender un LED cuando un Botón Pulsador es Accionado (Segunda Parte)

Haremos una variación del ejercicio anterior, esta vez, en caso de que el interruptor este dejando pasar corriente (el sistema esta Activo), se prenda un LED avisando que TODO ESTA BIEN Y OPERANDO. De lo contrario, otro LED avisando que El circuito no esta conectado (Inactivo). Entonces en este otro caso, tampoco hace falta controlar el Rebote, ya que es necesario detectar desde que comienza a presionar el Botón Pulsador, hasta que se deja de hacerlo.

(Programa 002_Boton_03_Led_Alarma_02)

```
1  int ledAlarmaPin = 5;      // Pin LED de Alarma - salida
2  int ledOkPin       = 7;    // Pin LED OK – salida
-
3  int interruptorPin = 3;    // Pin de entrada ( Acá está conectado el interruptor )
-
4  void setup( ) {
5    pinMode(ledAlarmaPin, OUTPUT);
6    pinMode(ledOkPin, OUTPUT);
7    pinMode(interruptorPin, INPUT);
8  }
-
9  void loop( ) {
10   if (digitalRead( interruptorPin ) == HIGH){
11     digitalWrite(ledAlarmaPin, HIGH);   // Enciende el LED
12     delay(250);                         // Pause de 1/4 segundo
13     digitalWrite(ledAlarmaPin, LOW);    // Apaga el LED
14     delay(250);                         // Pausa de 1/4 segundo
15   } else{
16     digitalWrite(ledOkPin, HIGH);       // Enciende el LED
17     delay(250);                         // Pause de 1/4 segundo
18     digitalWrite(ledOkPin, LOW);        // Apaga el LED
19     delay(250);                         // Pausa de 1/4 segundo
20   }
21 }  // Finaliza función loop( )
```

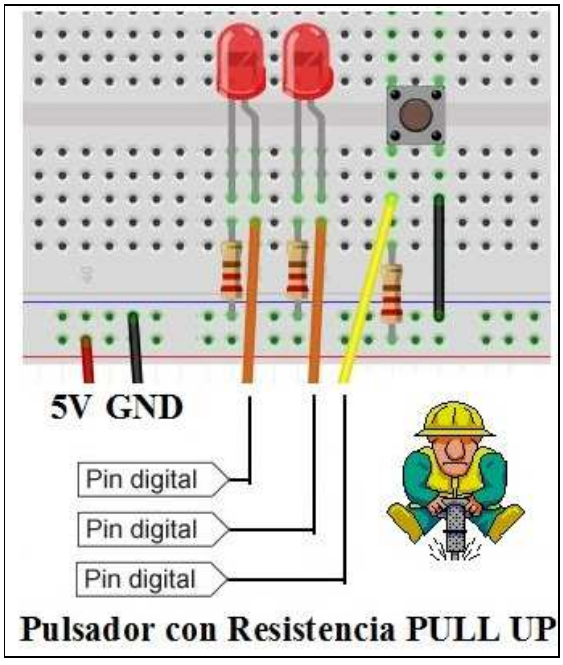
Explicación, Línea Por Línea

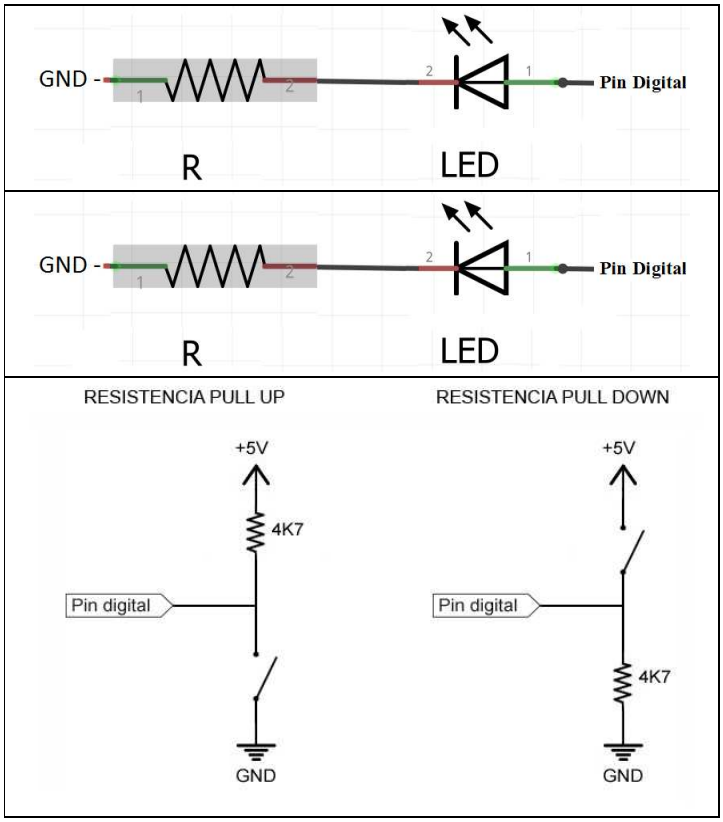
1 2 3	Declaración de Variables y asignación de los valores que luego corresponderán a los PINES que usaremos. Las variables son: "**led**Alarma**Pin**" identificará el PIN al que se conectara el LED de alarma, la Variable "ledOkPin" que indicara el PIN donde se conecta el LED que al prenderse indica que todo esta Bien y la variable "**interruptorPin**" que identifica al PIN en el que conectaremos el interruptor que luego controlaremos.
4	Comienza función de Configuración "**setup( )**".
5	En esta línea se configura el PIN identificado por la variable "**led**Alarma**Pin**", y por el saldrá la corriente que prendara el LED usado para avisar que el interruptor **NO** esta dejando pasar corriente, se configura de salida "**OUTPUT**"
6	En esta línea se configura el PIN identificado por la variable "**ledOkPin**", y por el saldrá la corriente que prendara el LED usado para indicar que el interruptor **SI** esta dejando pasar corriente, se configura de salida "**OUTPUT**"
7	En esta línea se configura el PIN identificado por la variable "**interruptorPin**", en el que se conecta el interruptor que controlaremos. Se configura de entrada "**INPUT**".
8	Llave que finaliza la función de Configuración
9	Cabecera e inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
10	En esta línea de código, la interpretación de la pregunta puede variar, según sea la

	conexión del interruptor (Resistencia PULL UP o Resistencia PULL DOWN). **Releer "FORMAS DE REALIZAR CONEXIONES" de un Botón Pulsador.** **Muy IMPORTANTE:** Si el botón se conecta con Resistencia "PULL UP", la pregunta será Verdadera cuando el Botón Pulsador **NO este Siendo Presionado** (Titila el LED de Alarma). Por el contrario, si el Botón Pulsador SI **esta Siendo Presionado** (Titila el LED de OK).
11	Enciende el LED conectado al PIN indicado por la variable "**led**Alarma**Pin**"
12	Pausa de ¼ de segundo
13	Apaga el LED conectado al PIN indicado por la variable "**led**Alarma**Pin**"
14	Pausa de ¼ de segundo
15	En esta línea esta la llave que cierra la parte verdadera, el "**else**" y la llave que abre la parte falsa de la pregunta realizada en línea 8.
16	Enciende el LED conectado al PIN indicado por la variable "**ledOkPin**"
17	Pausa de ¼ de segundo
18	Apaga el LED conectado al PIN indicado por la variable "**ledOkPin**"
19	Pausa de ¼ de segundo
20	Llave que cierra la parte verdadera de la pregunta realizada en línea 8.
21	Llave que marca el final de la función "**loop( )**". Que comenzó en la línea 7

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Botón Pulsador de 2 Patas – 2 LED - 1 Resistencia de 4K7 Ω (Para el Boton) – 2 Resistencia de 470Ω (Para Leds) – 6 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.





Puede agregar mensajes para realizar un seguimiento del funcionamiento de su programa. Entonces cuando ya este en funcionamiento, podrá ver los mensajes que usted inserte, cuando se presiona el Botón. Recordar que para activar el Monitor del Puerto Serie, hay que abrir en el IDE (donde escribió el programa que envió a la placa Arduino), en el Menú "**Herramientas**" y elegir la opción "**Monitor Serie**".

Y si Intercambiamos en el Protoboard, el lugar donde conecta los cables de 5V y GND, se transforma el circuito del Botón Pulsador, y como responderán los LED?

4- Encender y apagar un LED con el mismo botón pulsador.

Hacer un programa que permita detectar cuando un Botón Pulsador es presionado, y mediante esa detección prender y apagar el LED.

(Programa 002_Boton_04_Led_Pronde_Apaga_01)

```
1   const int PinLed    =  5;
2   const int PinBoton = 3;
-
3   int Estado              = 0; // Guardara el estado del botón
4   int state               = 0; // 0 LED apagado, mientras que 1 encendido
5   int Anterior_Estado = 0; // almacena el antiguo valor de Estado
-
6   void setup( ){  // definir si la variable es de entrada o salida.
7     pinMode(PinLed , OUTPUT); // establece PinLed como salida
8     pinMode(PinBoton , INPUT); // y PinBoton como entrada
9   }
-
10  void loop( ) {
11     Estado = digitalRead( PinBoton );   // lee el estado del Botón
12     if ( Estado == HIGH && Anterior_Estado == LOW) {
13        state= 1 - state;   // Oscilará entre Cero y Uno
14        //delay(10);
15     }
16     Anterior_Estado = Estado; // Guardo valor del antiguo estado
17     if (state==1){
18        digitalWrite(PinLed, HIGH); // enciende el LED
19     } else{
20        digitalWrite(PinLed,LOW); // apagar el LED
21     }
22  }  // Finaliza función loop( )
```

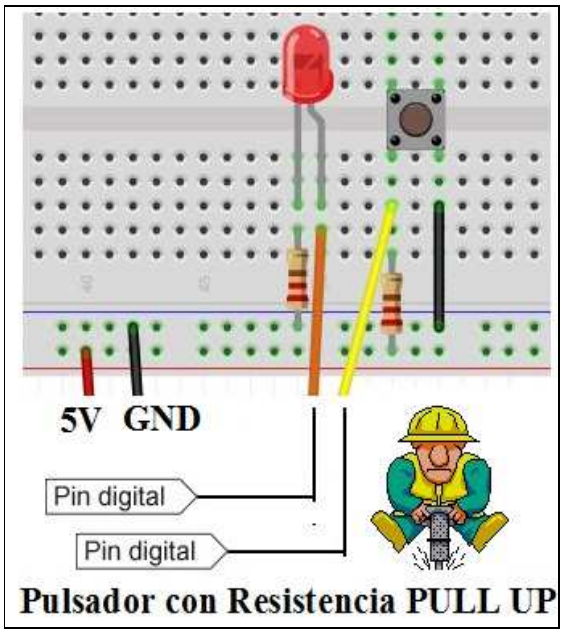
Explicación, Líneas Importantes	
10	Cabecera e inicio de la Función “**loop( )**”. Esta es quien permite que las acciones del programa se repitan indefinidamente.
11	Realizo una primer lectura del estado del Botón y guardo el resultado en la variable “Estado”
12	Verifico que el botón ha **cambiado de estado**. Es decir, si antas estaba LOW y ahora HIGH
13	Cambio de estado la Variable “state”. Preste atención, suele ser engañosa la sentencia
14	Instrucción deshabilitada. Si es de su agrado puede habilitarla y ver que pasa.
15	Termino “if”
16	Paso el estado actual como anterior
17	Si el contenido de la variable “state” es 1?
18	Prendo LED
19	De lo contrario
20	Apago LED
21	Fin del “**else**” que comenzó en línea 19
22	Final de la Función “**loop( )**”.

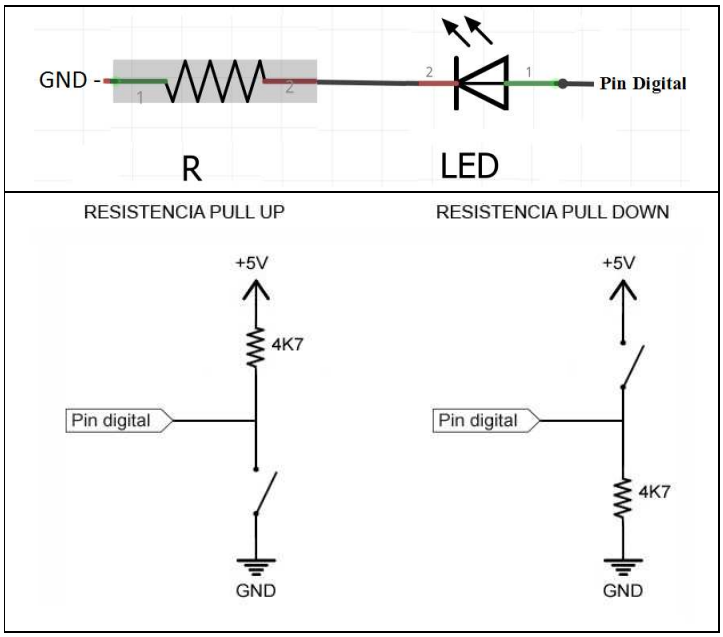
Puede agregar mensajes para realizar un seguimiento del funcionamiento de su programa (Recuerde abrir el puerto serie en la función “**setup( )**”). Entonces cuando ya este en funcionamiento, podrá ver los

mensajes que usted inserte, si se presiona el Botón. Recordar que para activar el Monitor del Puerto Serie, hay que abrir en el IDE (donde escribió el programa que envió a la placa Arduino), en el Menú "**Herramientas**" y elegir la opción "**Monitor Serie**".

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Botón Pulsador de 2 Patas – 1 LED - 1 Resistencia de 4K7 Ω (Para el Botón) – 1 Resistencia de 470Ω (Para el Led) – 5 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.





Si Intercambia en el Protoboard, el lugar donde conecta los cables de 5V y GND, en que se transforma el circuito del Botón Pulsador y como responderá el LED?

Qué Ocurre Cuando Usas La Función "delay( )" En Tu Programa

Quiero dejarte claro antes que nada que la función "delay( )" es un recurso importantísimo, y que en muchas ocasiones es imprescindible utilizar. Sin embargo, se trata de una herramienta que genera problemas de rendimiento en tus proyectos. Por eso, utilizar la función "delay( )" en Arduino.. ¡Sí, cuando sea necesario!

Cada vez que se ejecuta esta función en tu proyecto, el microcontrolador de tu Arduino se para, espera a que transcurra el tiempo que hayas establecido y ejecuta la siguiente instrucción. Esto provoca varios problemas.

1. Tu Arduino No Puede Detectar Eventos: Mientras tu placa Arduino está ejecutando un "delay( )" no puedes, por ejemplo, leer sensores. Esto no supone un gran problema siempre que no necesites una lectura muy frecuente de los mismos, pero en los proyectos que se "mueven" como puede ser un coche RC o un cuadricóptero, donde necesitas una respuesta rápida de tus dispositivos, el uso adecuado de delays es un factor clave (o deberás reconstruir tu modelo con mucha frecuencia).

2. El Consumo De Tu Proyecto Aumenta: A ver, esto es una mentira a medias. Usar la función delay en Arduino no aumenta el consumo de tiempo de tu proyecto, pero si (a causa de los delays) tardas varias veces más en realizar la misma tarea, es lógico pensar que, en términos generales, el consumo de tu proyecto se habrá multiplicado. Además, utilizar un "delay( )" no significa dejar tu microcontrolador en standby, es decir, no reduces la carga de trabajo de tu placa. Mantienes el chip a **máximo rendimiento** pero simplemente diciéndole que no haga nada (mas que esperar). Por lo que, en caso de que tu proyecto requiera el uso de pausas, podrías pensar en sustituir los delays por funciones que pongan tu placa en standby.

3. Los Delays Dentro Del Loop Son Acumulativos: Como todas las instrucciones dentro del loop (que no es más que un bucle while(1)) se están ejecutando una y otra vez, todos los problemas que te genera el uso de la función "**delay**" en Arduino se van arrastrando (y por tanto multiplicando) con el transcurso del tiempo. Una forma reducir este problema consiste en utilizar esta función dentro de tus if( ) y switch( ), y evitar ponerla dentro de bucles for( ) o while( ) y, siempre que puedas, intentar poner tus delays en módulos y llamar a esos módulos desde el loop cuando sea necesario.
4. Cuanto Más Tiempo Duren los Delays, Peor: Utilizar delays con una duración excesiva es incluso peor que usar muchos (al menos entre "delay( )" y "delay( )", tu proyecto tiene algo más de flexibilidad). No estoy tratando de decir con esto que, si ibas a poner un "delay(200)" pongas dos "delay(100)", eso es absurdo. Lo que ocurre es que muchas veces utilizamos esta función para esperar a que se cargue algún dato en memoria EEPROM, se oriente correctamente un servomotor o simplemente se active un relé, sin tener en cuenta el tiempo real que esto conlleva, es decir, si el servomotor tarda en orientarse 20ms, ¿Por qué poner un delay(30)" ?
Supongo que a estas alturas estarás pensando: Bueno, ¿y cómo sé yo cuánto dura eso? Lo cierto es que muchas veces resulta imposible predecir cuánto va a tardar en ejecutarse alguna acción, por lo que deberás recurrir al ancestral método del ensayo y error, incrementando la duración del delay poco a poco hasta dar con el tiempo exacto. Sé por propia experiencia que no hay cosa más aburrida que cargar un sketch una y otra vez pero si quieres un resultado profesional, no te queda otra.
5. La Función "delay( )" Es Incompatible Con Las Interrupciones: Simplemente te diré que pueden surgirte fallos al utilizar "delay( )" con las interrupciones de tu placa Arduino y que una forma fácil de evitarlos es utilizando la función delayMicroseconds( ), lo que nos lleva al siguiente punto.
6. **Muhisimas veces** tu Sketch Mejora Simplemente Cambiando "delay( )" Por DelayMicroseconds. ¿Cuánto crees que tarda un Arduino funcionando a 16MHz en ejecutar una instrucción? Estoy seguro de que en muchos de tus proyectos, tu Arduino pierde más tiempo esperando entre "delay( )" y "delay( )" que ejecutando el resto del código.
7. **Siempre Que Puedas Elige La Función "millis( )":** La función "millis( )" no es la panacea que curará todos los males de tu proyecto, de hecho, la función "millis( )" también puede provocarte problemas. Aun así, es lo más cerca que estarás de trabajar de forma paralela con Arduino, es decir, lo más cerca que estarás de hacer dos o más cosas a la vez.
Gracias a la función "millis( )", en vez de esperar un tiempo sin hacer nada (sin modificar ningún parámetro), puedes contar el tiempo que está transcurriendo y ejecutar instrucciones en función de eso. La ventaja de este método es que mientras tu Arduino está contando internamente el tiempo que pasa, puede seguir ejecutando instrucciones.

5- Encender y Apagar un LED por medio de 3 botones (en forma indistinta). Con Programación Multitareas.

El programa permite prender o Apagar un LED por medio de 3 botones (prende desde cualquiera y apaga desde cualquiera de ellos).

(Programa "002_Boton_05_Led_Prende_Con_3_Botones)

```
1   const int PinBoton_01 = 3,
2             PinBoton_02 = 5,
3             PinBoton_03 = 9,
4             PinLed      = 7;
-
5   int Boton_01_Valor,  // Variable botón 01
6       Boton_02_Valor,  // Variable botón 02
7       Boton_03_Valor,  // Variable botón 03
8       EstadoLed,       // Estado del LED (Prendido o Apagado)
9       Presionado_01,   // Para saber si Este Botón Fue Presionado
10      Presionado_02,   // Para saber si Este Botón Fue Presionado
11      Presionado_03;   // Para saber si Este Botón Fue Presionado
-
12  void setup( ){
```

```
13    Serial.begin (9600);
14    pinMode(PinBoton_01, INPUT_PULLUP);
15    pinMode(PinBoton_02, INPUT_PULLUP);
16    pinMode(PinBoton_03, INPUT_PULLUP);
17    pinMode(PinLed, OUTPUT);
18    Presionado_01 = 0;
19    Presionado_02 = 0;
20    Presionado_03 = 0;
21    EstadoLed  = LOW;    // Comienza en Apagado
22  }
 -
23  void loop ( ){
24    Boton_01_Valor = digitalRead(PinBoton_01); // Leemos PinBoton_01.
25    Boton_02_Valor = digitalRead(PinBoton_02); // Leemos PinBoton_02.
26    Boton_03_Valor = digitalRead(PinBoton_03); // Leemos PinBoton_03.
27    if(Boton_01_Valor == LOW){
28      Presionado_01 = 1;     //Activo Botón
29    }
30    if(Boton_02_Valor == LOW){
31      Presionado_02 = 1;     //Activo Botón
32    }
33    if(Boton_03_Valor == LOW){
34      Presionado_03 = 1;     //Activo Botón
35    }
36    Boton_01_Valor = digitalRead(PinBoton_01); // Leemos PinBoton_01.
37    Boton_02_Valor = digitalRead(PinBoton_02); // Leemos PinBoton_02.
38    Boton_03_Valor = digitalRead(PinBoton_03); // Leemos PinBoton_03.
 -
39    if( (Boton_01_Valor == HIGH && Presionado_01 == 1 ||
40                  Boton_02_Valor == HIGH && Presionado_02 == 1 ||
41                        Boton_03_Valor == HIGH && Presionado_03 == 1)){
 -
42      Serial.println("------> Preciono Boton");
43      Presionado_01 = 0;    //La variable vuelve a su valor original
44      Presionado_02 = 0;    //La variable vuelve a su valor original
45      Presionado_03 = 0;    //La variable vuelve a su valor original
 -
46      EstadoLed = digitalRead(PinLed);  // Leo Estado del LED
47      if(EstadoLed == LOW){
48        Serial.println("---------> PRENDO LED");
49      }else{
50        Serial.println("---------> APAGO LED");
51      }
52      EstadoLed = 1 - EstadoLed;              // Invierte el estado
53      digitalWrite( PinLed, EstadoLed );     // Prende o apaga
54      Serial.println( digitalRead(PinLed) );  // Muestro Por Pantalla Valor del PIN
55    }
56    // Acá Realizo Otras Tareas
57    // Acá Realizo Otras Tareas
58  }  // Finaliza función loop( )
```

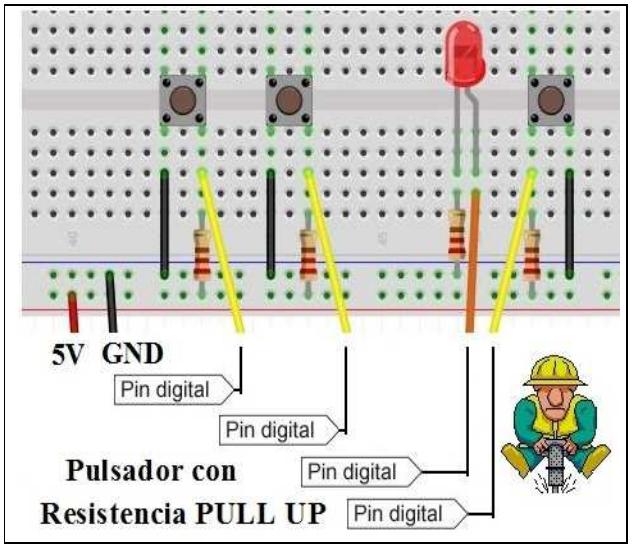
Explicación, Líneas Importantes

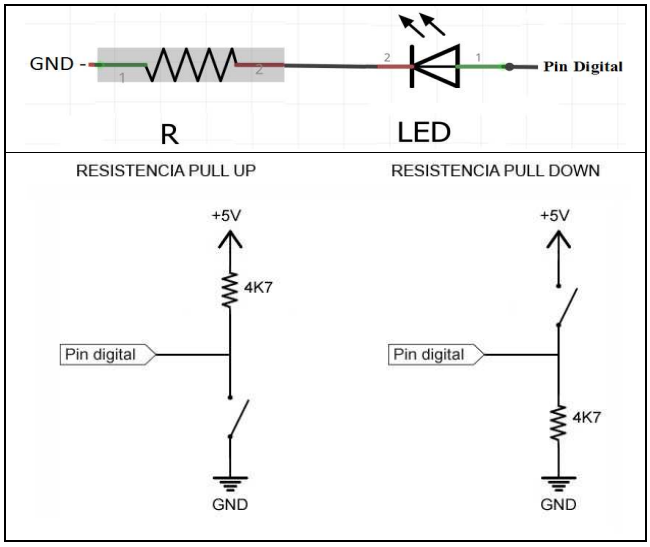
23	Cabecera e inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
24	Leo estado del Botón 01 y Guardo respuesta en variable destinada para eso.

25	Leo estado del Botón 02 y Guardo respuesta en variable destinada para eso.
26	Leo estado del Botón 03 y Guardo respuesta en variable destinada para eso.
29 a 29	Controlo si se ha Presionado el Botón 01, y de ser así, prendo la señal, y poder saber luego cuando se dejó de presionar.
30 a 32	Controlo si se ha Presionado el Botón 02, y de ser así, prendo la señal, y poder saber luego cuando se dejó de presionar.
33 a 35	Controlo si se ha Presionado el Botón 03, y de ser así, prendo la señal, y poder saber luego cuando se dejó de presionar.
36 a 38	Nuevamente leo estado de todos los botones.
39 a 41	Comienza "**if**" - Acá se pregunta si botón esta libre y antes había sido presionado. Si es verdad, realizará las acciones contenidas entres las líneas 42 a la 55.
42	Mensaje por Puerto Serie a la PC.
43 a 45	Apago variables que había activado al presionar Botón, de esta forma ya quedarían listas para detectar la próxima vez que sea presionado algún botón.
46	Leo el estado del LED. Y a continuación podré enviar mensajes para saber el estado en la PC y **poder invertir el estado** – Si esta prendido lo apago y si esta apagado lo prendo.
47 a 51	Mensajes a la PC.
52	Invierto estado de la Variable
53	Prendo o apago LED, según sea el nuevo estado
54	Envió Mensaje a la PC.
56 a 57	Indico al Programador que ahí podría insertar otros procesos, que nuestro Robot podrá ejecutarlos paralelamente sin ningún inconveniente. Siempre y cuando, estos nuevos procedimientos estén programados bajo el paradigma de Multiprocesamiento
58	Finaliza la Función "**loop** ( )"

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 3 Botones Pulsador de 2 Patas – 1 LED - 3 Resistencia de 4K7 Ω (Para el Botón) – 1 Resistencia de 470Ω (Para el Led) – 9 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.





Si Intercambia en el Protoboard, el lugar donde conecta los cables de 5V y GND, en que se transforma el circuito del Botón Pulsador y como responderá el LED?

FUNCIONES QUE UTILIZAREMOS EN NUESTRO PROGRAMA.

- **analogWrite(pin, value)**: Escribe un valor pseudo-analógico usando modulación por ancho de pulso (PWM) en un PIN de salida marcado como PWM. Esta función está activa para los pines 3, 5, 6, 9, 10, 11. Ej analogWrite(9, v); // escribe 'v' en el PIN 9 (analógico). Puede especificarse un valor de ve que oscile entre 0 - 255. Un valor 0 genera 0V en el PIN, y especificado 255 genera 5V. (NOTA: Esta Función ya la habíamos explicado ampliamente en proyectos anteriores)

6- Aumentar/reducir intensidad LED con 2 Botones.

Este programa permite Subir y/o Bajar la intensidad con que brilla un LED. Botón para Subir y Botón para Bajar.

(Programa 002_Boton_06_Led_Intensidad_Con_2_Botones)

```
1  const int PinBotonSuma = 5, //botón que sumara 5 al ciclo de trabajo
2            PinBotonResta = 6, //botón que restara 5 al ciclo de trabajo
3            PinLed        = 3; //Puerto con salida PWM
-
4  int botonSumaValue,   //Variable botón de suma
5     botonRestaValue,   //Variable botón de resta
6     IntensidadLed,     //Variable Indica Brillo Led
7     Incrementos =10;   // Cantidad de unidades que se incrementará
-
8  void setup( ){
9   Serial.begin (9600);
10  pinMode(PinLed, OUTPUT);
11  pinMode(PinBotonSuma, INPUT_PULLUP);
12  pinMode(PinBotonResta, INPUT_PULLUP);
13 }
-
14 void loop ( ){
15  botonSumaValue= digitalRead(PinBotonSuma);
16  botonRestaValue= digitalRead(PinBotonResta);
17  if (botonSumaValue == HIGH && IntensidadLed <=245){
18    IntensidadLed +=Incrementos;
19  }
20  if (botonRestaValue == HIGH && IntensidadLed >=10){
21    IntensidadLed -=Incrementos;
22  }
23  Serial.print("Intensidad: ");
24  Serial.println(IntensidadLed);
25  analogWrite(PinLed, IntensidadLed);
26  delay(100);  // Puede eliminarse – Esta solo para permitir ver los mensajes.
27   // Acá Realizo Otras Tareas
28   // Acá Realizo Otras Tareas
29 }  // Finaliza función loop( )
```

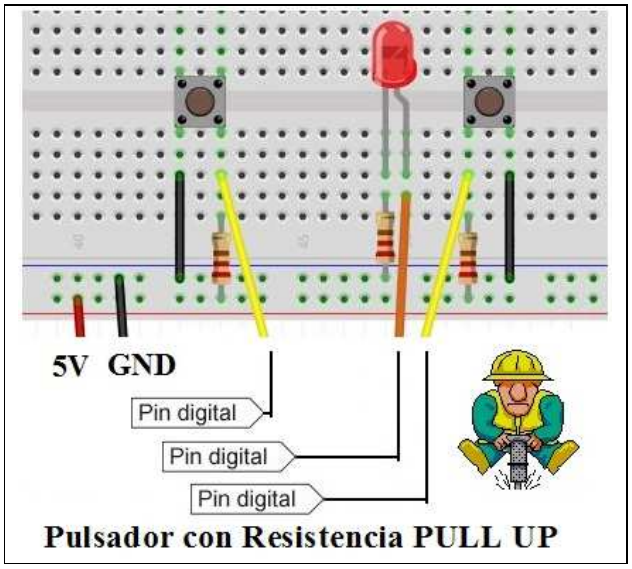
Explicación, Líneas Importantes

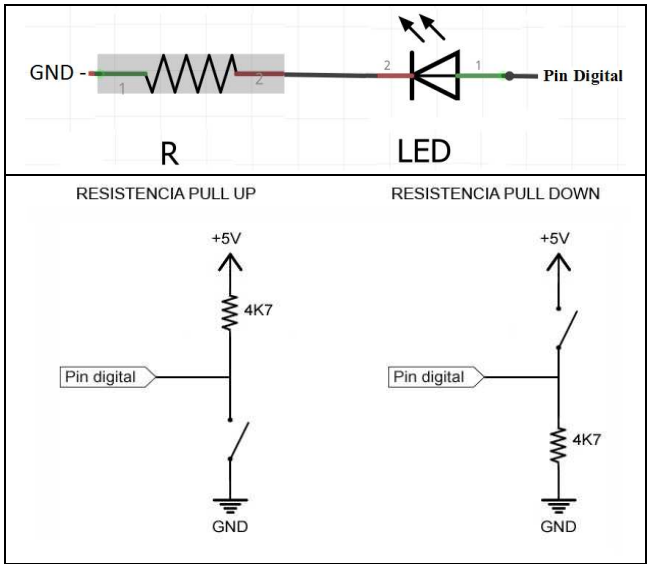
8	Cabecera o inicio de la Función "**setup( )**". Configuración de Puertos y Algunas Acciones que se deben hacer una sola vez y al principio del programa
9	Abro el puerto serie para iniciar comunicación con la Placa Arduino
14	Cabecera o inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
15 y 16	Leo el estado de los botones que serán usados para subir y bajar intensidad del

	LED.
17 a 19	Pregunto se el Botón de "**subir**" ha sido presionado y si puedo continuar subiendo, si esto se cumple, entonces Sumo el incremento deseado, a la variable que usare para indicar la intensidad del LED.
20 a 22	Pregunto se el Botón de "**bajar**" ha sido presionado y si puedo continuar bajando, si esto se cumple, entonces Resto el incremento deseado, a la variable que usare para indicar la intensidad del LED.
23 y 24	Mensajes a la PC a través del puerto serie.
25	Doy la orden de Prender el LED con la intensidad indicada en la variable "**IntensidadLed**"
26	Hago un pequeño "**delay**" (Puede eliminarse) para permitir una visualización adecuada de los mensajes enviados por el Puerto Serie..
27 y 28	Acá se pueden agregar todas las tareas que sean necesaria realizar mientras el LED sube o baja la intensidad (Esto lo permite la multitarea)
29	Final de la función "**loop( )**"

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 2 Botones Pulsador de 2 Patas – 1 LED - 2 Resistencia de 4K7 Ω (Para el Botón) – 1 Resistencia de 470Ω (Para el Led) – 7 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.





Si Intercambia en el Protoboard, el lugar donde conecta los cables de 5V y GND, en que se transforma el circuito del Botón Pulsador y como responderá el LED?

7- Semáforo (02) inteligente de una sola calle.

Semáforo para Autos y Peatones. Funciona Permanentemente, pero mediante botón detiene a los autos y acciona semáforo de Peatones.

(Programa 002_Boton_07_Semaforo_03)

```
1   int Verde     = 1;
2   int Amarillo = 2;
3   int Rojo      = 3;
-
4   int Procesando;
-
```

```
5   long TiempoVerdeAuto     = 7000;  // Milisegundos que estará Verde
6   long TiempoAmarilloAuto = 2000;  // Milisegundos que estará Amarillo
7   long TiempoRojoAuto      = 7000;  // Milisegundos que estará Rojo
-
8   const int PinVerdeAutos     =  5;
9   const int PinAmarilloAutos  =  7;
10  const int PinRojoAutos      =  9;
11  const int PinBoton          = 11;
-
12  const int PinVerdePeaton    = 13;
13  const int PinRojoPeaton     = 17;
-
14  boolean   SI       = true;
15  boolean   NO       = false;
16  boolean   SaltoNormal;
-
17  long TiempoInicial  = 0;
-
18  void setup( ) {
19    Serial.begin(9600);
20    pinMode(PinVerdeAutos, OUTPUT);        //Semáforo para Autos
21    pinMode(PinAmarilloAutos, OUTPUT);    //Semáforo para Autos
22    pinMode(PinRojoAutos, OUTPUT);         //Semáforo para Autos
23    pinMode(PinBoton, INPUT_PULLUP);     //Botón Cruce Peatón
24    pinMode(PinVerdePeaton, OUTPUT);      //Semáforo Peatón
25    pinMode(PinRojoPeaton, OUTPUT);        //Semáforo Peatón
26    TiempoInicial = millis( );      // Tiempo Inicial
27    Procesando = Verde;             // Para que el primer led en prender sea el Verde
28    SaltoNormal     = SI;           // No Hay Botón presionado al comenzar Programa
29  }
-
30  boolean ApretaBoton(int Color){
31    if (digitalRead( PinBoton ) == LOW){  // si botón apretado (Pasa Peatón)
32      //Serial.println("Apretó el botón");  // Habilite este mensaje!
33      TiempoInicial = millis( );  // La Cuenta Inicia de Cero Para Cruce de Peatones
34      if(Color != Rojo){  // Si Apretaron Botón cuando esta en rojo, ahí lo dejo
35        Procesando = Amarillo; // Apretaron Botón, y configuro a donde debe saltar
36      }
37      SaltoNormal = NO;
38      return SI;
39    }else return NO;
40  }
-
41  void AutoVerde( ){  // Función que maneja el color Verde del Semáforo para Autos.
42    if(Procesando == Verde){
43       SaltoNormal = SI;
44      if(!ApretaBoton(Procesando) && (millis( ) - TiempoInicial <= TiempoVerdeAuto)){
45        digitalWrite(PinVerdeAutos, HIGH); // Enciende el LED Verde Autos
46        digitalWrite(PinRojoPeaton, HIGH); // Enciende el  ROJO PEATON
47      }else{
48        digitalWrite(PinVerdeAutos, LOW); // Apaga el LED Verde Autos
49        digitalWrite(PinRojoPeaton, LOW); // Apaga el LED ROJO PEATON
50        if(SaltoNormal) Procesando = Amarillo;
51        TiempoInicial = millis( );
```

```
52        }
53      }
54    }
-
55    void AutoAmarillo( ){ // Función que maneja el color Amarillo del Semáforo para Autos.
56      if(Procesando == Amarillo){
57        SaltoNormal = SI;
58        if(!ApretaBoton(Procesando) && ( ( millis( ) - TiempoInicial) <= TiempoAmarilloAuto)){
59          digitalWrite(PinAmarilloAutos, HIGH);  // Enciende el LED Amarillo Autos
60          digitalWrite(PinRojoPeaton, HIGH);       // Enciende el  Rojo PEATON
61        }else{
62          digitalWrite(PinAmarilloAutos, LOW); // Apaga el LED Amarillo Autos
63          digitalWrite(PinRojoPeaton, LOW);      // APAGA el  Rojo PEATON
64          if(SaltoNormal) Procesando = Rojo;
65          TiempoInicial = millis( );
66        }
67      }
68    }
-
69    void AutoRojo( ){  // Función que maneja el color Rojo del Semáforo para Autos.
70      if(Procesando == Rojo){
71         SaltoNormal = SI;
72        if(!ApretaBoton(Procesando) && ( ( millis( ) - TiempoInicial) <= TiempoRojoAuto)){
73          digitalWrite(PinRojoAutos, HIGH);    // Enciende el LED Rojo Autos
74          digitalWrite(PinVerdePeaton, HIGH); // Enciende el LED Verde Peatón
75        }else{
76          digitalWrite(PinRojoAutos, LOW);     // Apaga el LED Rojo Autos
77          digitalWrite(PinVerdePeaton, LOW);  // Apaga el LED Verde Peatón
78          if(SaltoNormal) Procesando = Verde;
79          TiempoInicial = millis( );
80        }
81      }
82    }
-
83    void loop( ) {
84      AutoVerde( );
85      AutoAmarillo( );
86      AutoRojo( );
87    }  // Finaliza función loop( )
```

Explicación, Líneas Importantes

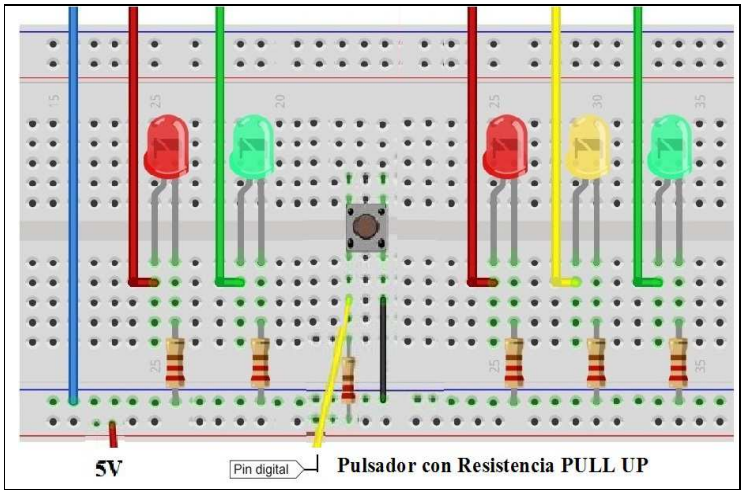
18	Cabecera o inicio de la Función "**setup( )**". Configuración de Puertos y Algunas Acciones que se deben hacer una sola vez y al principio del programa
19	Abro el puerto serie para iniciar comunicación con la Placa Arduino
26	Tomo el primer tiempo inicial. Para comenzar.
27	Configuro cual será la primera luz que prenda el semáforo al comenzar
28	Configuro que, comienza normalmente, no se presiono el botón para peatones.
30	Inicio o cabecera de la Función "**ApretaBoton( )**" que controla cuando se aprieta el botón y reconfigura el semáforo de autos y Peatones.
31	Verifico si ha sido presionado el botón que dará prioridad de paso a un Peatón.
32	Envío Mensaje por Puerto Seria a la PC, informando que se ha presionado el Botón (Puede activar esta línea eliminando los caracteres //)
33	Tomo el tiempo inicial para el proceso que se esta iniciando (permitir que pasen los Peatones)

34 a 36	Configuro la próxima acción a realizar será prender la luz amarilla para los autos, a menos que ya estuviera en rojo, y de ser así, dejo que continúe en rojo, ya que los peatones podrán pasar
37	Configuro que rompa la secuencia normal de luces.
38	Termino la Función, retornando un "**SI**" es decir que Si fue presionado el Botón de Peatones
39	Termino la Función, retornando un "**NO**" es decir que No fue presionado el Botón de Peatones
41	Inicio o cabecera de la Función "**AutoVerde( )**" que controla el funcionamiento de luz Verde en semáforo de autos y más.
42	Verifico si es este el color que se debe procesar. Si es así continuo, de lo contrario termino la función.
43	Regreso a su estado normal la variable "**SaltoNormal**" (no se que se proceso antes).
44	Si no Se ha apretado el botón para Peatones, y el tiempo de la Luz que estoy procesando, todavía no termino, procederé a mantener prendidas las luces.
45 y 46	Prendo lunes del semáforo de Autos y Peatones.
47	"**else**" Parte falsa del "**if**". Es decir, que se apretó el botón de peatones o se acabo el tiempo que deben estar prendidas estas luces, entonces, procederé a apagaras y configurar cual será la próxima acción que se debe ejecutar.
48 a 52	Apago luces y configuro cual será la próxima acción que se debe ejecutar.
53	Termina parte falsa del "**if**".
54	Termina Función.
55	Inicio o cabecera de la Función "**AutoAmarillo( )**" que controla el funcionamiento de la luz Amarilla en semáforo de autos y más.
56	Verifico si es este el color que se debe procesar. Si es así continuo, de lo contrario termino la función.
57	Regreso a su estado normal la variable "**SaltoNormal**" (no se que se proceso antes).
58	Si no Se ha apretado el botón para Peatones, y el tiempo de la Luz que estoy procesando, todavía no termino, procederé a mantener prendidas las luces.
59 y 60	Prendo lunes del semáforo de Autos y Peatones.
61	"**else**" Parte falsa del "**if**". Es decir, que se apretó el botón de peatones o se acabo el tiempo que deben estar prendidas estas luces, entonces, procederé a apagaras y configurar cual será la próxima acción que se debe ejecutar.
62 a 66	Apago luces y configuro cual será la próxima acción que se debe ejecutar.
67	Termina parte falsa del "**if**".
68	Termina Función.
69	Inicio o cabecera de la Función "**AutoRojo( )**" que controla el funcionamiento de la luz Roja en semáforo de autos y más.
	Verifico si es este el color que se debe procesar. Si es así continuo, de lo contrario termino la función.
	Regreso a su estado normal la variable "**SaltoNormal**" (no se que se proceso antes).
	Si no Se ha apretado el botón para Peatones, y el tiempo de la Luz que estoy procesando, todavía no termino, procederé a mantener prendidas las luces.
	Prendo lunes del semáforo de Autos y Peatones.
	"**else**" Parte falsa del "**if**". Es decir, que se apretó el botón de peatones o se acabo el tiempo que deben estar prendidas estas luces, entonces, procederé a apagaras y configurar cual será la próxima acción que se debe ejecutar.
	Apago luces y configuro cual será la próxima acción que se debe ejecutar.
80	Termina parte falsa del "**if**".
81	Termina el **if(Procesando == Rojo){**
82	Termina Función.
83	Cabecera e inicio de la Función "**loop( )**". Esta es quien permite que las acciones del programa se repitan indefinidamente.
84	Llamado a la función "**AutoVerde( )**".

85	Llamado a la función "**AutoAmarillo( )**".
86	Llamado a la función "**AutoRojo( )**".
87	Finaliza función "**loop( )**".

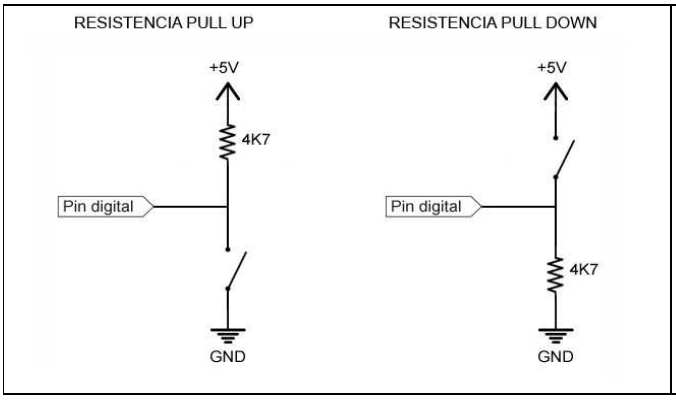
CIRCUITO PARA NUESTRO PROYECTO

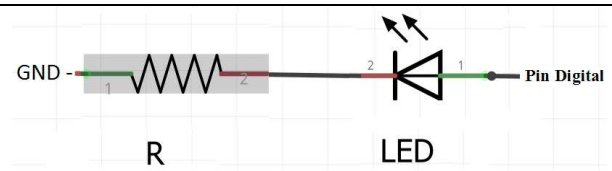
Lista de Materiales: 1 Botón Pulsador de 2 Patas – 5 LED - 1 Resistencia de 4K7 Ω (Para el Botón) – 5 Resistencias de 470Ω (Para LEDs) – 9 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.



Arriba, 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**".

Abajo, de Izquierda a Derecha: Cable Rojo conectar a 5V y cable amarillo al Pin digital configurado en la constante "**PinBoton**" al comienzo del programa.





Repetir esta conexión para cada uno de los LEDs. Debes conectar todos al mismo GND (cable azul).

FUNCIONES QUE UTILIZAREMOS EN NUESTRO PROGRAMA.

Obtiene números aleatorios y los muestra por el Monitor Serie

```
long randomNumber;
void setup( ) {
  Serial.begin(9600);
  Serial.println("Ini. sec. núm. aleatorios");
}
void loop( ) {
 //Genera numero aleatorio entre 1 y 100
  randomNumber = random(1,100);

  Serial.print("El numero aleatorio es = ");
  Serial.println(randomNumber);
  delay(1000);
}
```

- **analogRead(pin)**: Lee el valor desde el pin analógico especificado con una resolución de 10 bits. Esta función solo funciona en los pines analógicos (0-5). El valor resultante es un entero de 0 a 1023. Los pines analógicos, a diferencia de los digitales no necesitan declararse previamente como INPUT o OUTPUT.

- **Random( )**: Emula el azar en un programa, por ejemplo, obtener resultados aleatorios de un dado para un juego. Para ello existe la función esta función, que está presente en todos los lenguajes de programación de alto nivel. Con Arduino, tenemos dos alternativas:

1- random(max): Obtiene un numero "aleatorio" desde 0 hasta cualquier valor < max.
2- random(min, max): Obtiene un numero "aleatorio" desde "min" hasta cualquier valor < "max".

Sin embrago, a pesar se obtener numero aleatorios, siempre obtendremos la misma secuencia. Podemos solucionar esto usando la llamada "**semilla"** y que también existe en todos los lenguajes que usan la función "**random**( )".

La semilla es un valor por defecto que toma la función "**random**" para inicializarse, si cambiamos esta semilla la secuencia de datos cambiara cada vez que se use. Y es acá donde aparece en escena la función "**randomSeed( )".**

- **randomSeed( ):** inicializa el generador de números pseudo aleatorios, haciendo que se inicie en un punto arbitrario en su secuencia aleatoria.
  Entonces, en el caso que sea importante que, la secuencia de valores generados por "**random( )**" difiera, en sucesivas ejecuciones de un programa, use randomSeed( ) para inicializar el generador de números aleatorios. Con Arduino podemos usar cono parámetro de entrada, el valor devuelto por la función "**analogRead( )**" con el pin desconectado. Es decir: **randomSeed(analogRead(0)).**
  Por otro lado, ocasionalmente puede ser útil usar secuencias pseudo aleatorias que se repinta exactamente.
  Esto se puede hacer llamando a randomSeed( ) con un número fijo, antes de iniciar la secuencia aleatoria. Es decir, por ejemplo: **randomSeed(4)**

Ejemplo uso de las Funciones ramdom( ) y randomSeed(analogRead(0));

```
long NroAleatorio;

void setup( ) {
   Serial.begin(9600);
   randomSeed(analogRead(0));
}
void loop( ) {
     NroAleatorio  = random(300);
    Serial.println( NroAleatorio );
    delay(50);
}
```

8- Simulación de un dado al ser lanzado. Nuestro circuito, simulará un DADO, donde cada LED será uno de los puntitos que nos muestra la cara del dado. En el Circuito, los LEDs deben ser ordenados tal como se nuestra.

IzquierdaArriba		DerechaArriba
IzquierdaMedio	CentralMedio	DerechaMedio
IzquierdaAbajo		DerechaAbajo

7 Led en Total

Posición de LEDs en el Circuito y Nombres que se asignan a las variables en programa

El numero de PIN que esté asociado a la Constante, realmente no importa, pero si hay que respetar las posiciones de cada LEDs.

(Programa 002_Boton_09_Dado_01)

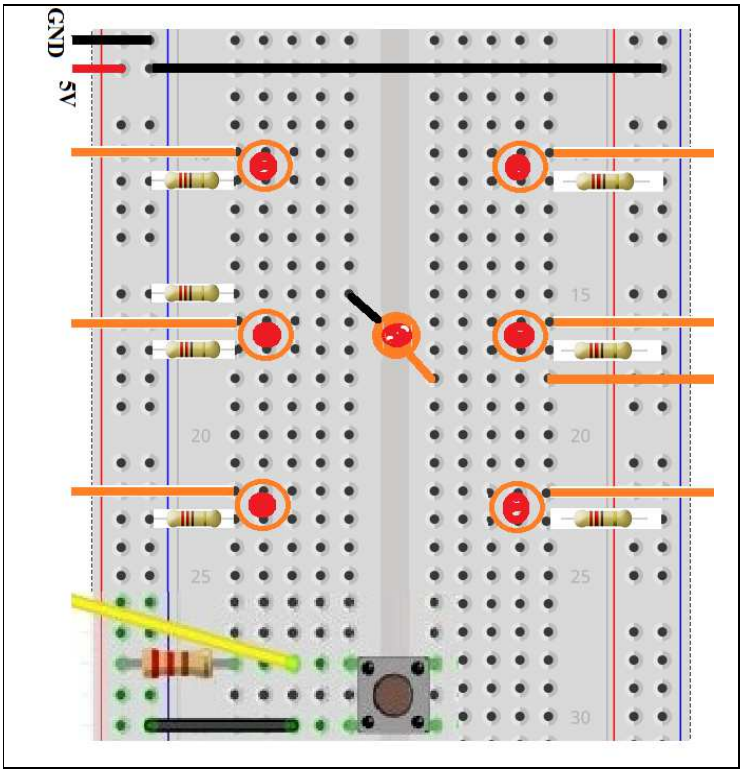
```
1   int const PinBoton       = 2;
2   int const PinAnalogico = 0; // Pin Usado para Generar la Raíz
-
3   int const IzquierdaAbajo   = 3;
4   int const IzquierdaMedio  = 4;
5   int const IzquierdaArriba  = 5;
6   int const CentralMedio     = 6;
7   int const DerechaAbajo    = 7;
8   int const DerechaMedio   = 8;
9   int const DerechaArriba   = 9;
10  long NroAleatorio;
-
11  void setup( ){
12    pinMode(IzquierdaAbajo, OUTPUT);
13    pinMode(IzquierdaMedio, OUTPUT);
```

```
14    pinMode(IzquierdaArriba, OUTPUT);
15    pinMode(CentralMedio, OUTPUT);
16    pinMode(DerechaAbajo, OUTPUT);
17    pinMode(DerechaMedio, OUTPUT);
18    pinMode(DerechaArriba, OUTPUT);
19    pinMode(PinBoton, INPUT);
20    Serial.begin(9600);
21    randomSeed(analogRead(PinAnalogico));
22  }
23   void loop( ){
24     if (digitalRead(PinBoton) == LOW){
25        NroAleatorio = random(1, 7);
26        Serial.println( NroAleatorio );
27        if (NroAleatorio == 6)  Seis( );
28        if (NroAleatorio == 5)  Cinco( );
29        if (NroAleatorio == 4)  Cuatro( );
30        if (NroAleatorio == 3)  Tres( );
31        if (NroAleatorio == 2)  Dos( );
32        if (NroAleatorio == 1)  Uno( );
33        delay(4000);  // Esta espera es puesta para que tengas tiempo de ver
34        LimpiaTodo( );
35     }
36  }  // Finaliza función loop( )
37   void Seis( ){
38     digitalWrite(IzquierdaAbajo, HIGH);
39     digitalWrite(IzquierdaMedio, HIGH);
40     digitalWrite(IzquierdaArriba, HIGH);
41     digitalWrite(DerechaAbajo, HIGH);
42     digitalWrite(DerechaMedio, HIGH);
43     digitalWrite(DerechaArriba, HIGH);
44  }
45  void Cinco( ){
46     digitalWrite(IzquierdaArriba, HIGH);
47     digitalWrite(IzquierdaAbajo, HIGH);
48     digitalWrite(CentralMedio, HIGH);
49     digitalWrite(DerechaArriba, HIGH);
50     digitalWrite(DerechaAbajo, HIGH);
51  }
52  void Cuatro( ){
53     digitalWrite(IzquierdaArriba, HIGH);
54     digitalWrite(IzquierdaAbajo, HIGH);
55     digitalWrite(DerechaArriba, HIGH);
56     digitalWrite(DerechaAbajo, HIGH);
57  }
58  void Tres( ){
59     digitalWrite(IzquierdaArriba, HIGH);
60     digitalWrite(CentralMedio, HIGH);
62     digitalWrite(DerechaAbajo, HIGH);
63  }
64  void Dos( ){
65     digitalWrite(DerechaAbajo, HIGH);
66     digitalWrite(IzquierdaArriba, HIGH);
67  }
68  void Uno( ){
```

```
69      digitalWrite(CentralMedio, HIGH);
70  }
71  void LimpiaTodo( ){
72      digitalWrite(IzquierdaAbajo, LOW);
73      digitalWrite(IzquierdaMedio, LOW);
74      digitalWrite(IzquierdaArriba, LOW);
75      digitalWrite(CentralMedio,LOW);
76      digitalWrite(DerechaAbajo, LOW);
77      digitalWrite(DerechaMedio, LOW);
78      digitalWrite(DerechaArriba, LOW);
79  }
```

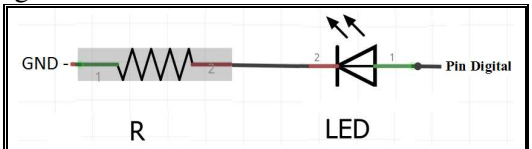
CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales: 1 Botón Pulsador de 2 Patas – 7 LED - 1 Resistencia de 4K7 Ω (Para el Botón) – 7 Resistencias de 470Ω (Para LEDs) – 12 Cables Macho/Macho - 1 Placa Protoboard - Placa Arduino y 1 Cable USB.



Izquierda, de Arriba hacia abajo: Cable Negro a GND, Rojo a 5V, Naranjas, se conectan al Pin Digital corresponderte, según lo configurado en programa "**IzquierdaArriba**", "**IzquierdaMedio**" e "**IzquierdaAbajo**". Cable Amarillo, al Pin Digital reservado en el programa para el Botón.

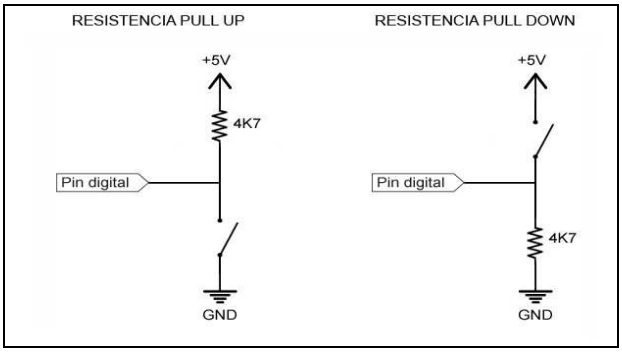
Derecha, de Arriba hacia abajo: Cables Naranjas, que corresponden a las constantes usadas en el programa: "**DerechaArriba**", "**DerechaMedi**o", "**CentralMedio**" y "**DerechaAbaj**o" se conectan a lo Pines corresponderte, según lo configurado en programa.



Repetir conexión para cada uno de los LEDs. Todos al mismo GND.

Recordar el LED central, Son 7 en total.

Respecto a la Conexión del Botón, en este caso estamos aplicando PULL UP (Con la Secuencia: 5V-Resistencia-Pin-Boton-GND).

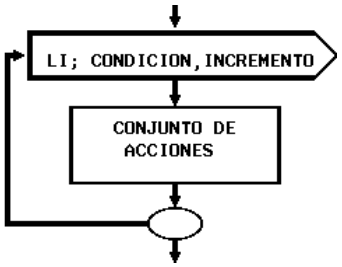


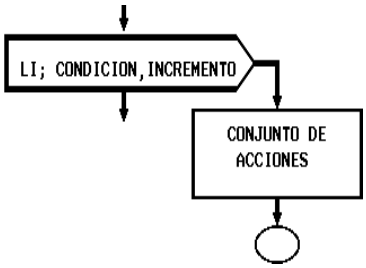
Si aplicáramos la otra configuración en la resistencia del Botón, este seria ignorado permanentemente.
Una buena práctica, que nos dará experiencia, es realizar la prueba y analizar los resultados con la otra configuración del Botón.

A este proyecto regresaremos más adelante cuando prendamos un Display, con un Dígito Numérico. El concepto es el mismo que hemos utilizado acá.

Estructura Repetitiva for( )

A continuación presentamos el **símbolo usado para la diagramación**, y su **traducción o codificación** al lenguaje de programación. Debido a que diversos autores representan esta estructura de varias formas, hemos tomado aquí las dos más frecuentes. Es conveniente aclarar que ambas simbologías tienen igual codificación.-






```
for( i=0; i<Max; i++ ) {
  accion1
  accion2
   .
 }
Siguiente Acción
```

La estructura **for( ),** podríamos decir que es, una manera de describir un bucle. La estructura esta compuesta por la palabra reservada "**for**", seguida por un conjunto de expresiones entre paréntesis. Estas expresiones son **dos** o **tres** campos separados entre si por punto y coma.

El primer campo contiene la inicialización o valor inicial de la variable. Podría agregar que no hay límite para estas expresiones, pero un buen estilo de programación recomienda no excederse. Varios mandatos de inicialización pueden ubicarse en este campo, separados por punto y coma.

```
void setup( ) {
  Serial.begin(9600);
}
void loop( ) {
  for(int i=0;  i<10;  i++) {
    Serial.println(i);
    delay(1000);
  }
}
```
Ejemplo simple uso del "for"

El segundo campo, en este caso conteniendo, es la condición que se evalúa al principio de cada vuelta del bucle o mejor dicho "**la condición que debe cumplir para continuar repitiendo**"; caso contrario se detendrá, y continuara con la "**Siguiente Acción**" fuera del buche. Es una expresión que puede evaluar verdadero o falso.

La expresión contenida en el 3er. campo se ejecuta cada vez que el bucle se ejecuta, pero no se ejecuta hasta que todos los mandatos contenidos en el bucle se ejecutan en su totalidad. Este campo, al igual que el primero, puede contener varios operandos separados por punto y coma.

Como siempre, el bloque de acciones debe comenzar con una llave, y finalizado con otra que cierra. En caso de haber solo una instrucción, podremos omitir las llaves, pero considero una buena técnica, el poner siempre la llave que abre y la llave que cierra el bloque.

9- Prender y Apagar Gradualmente un LED (Subiendo su brillo desde cero a Máximo y desde el Máximo a cero). Uso de la Estructura Repetitiva "for".

(Programa "001_Led_05_Subo_Bajo_Brillo_Led_02_for")

```
1  int ledPin = 13; // Pin controlado por PWM
2  int i;        // Variable usada para ciclo repetitivo
-
3  void setup( ){
4    pinMode(ledPin, OUTPUT);
5  }
6  void loop( ) {
7    for (i=0; i<=255; i++){
8      analogWrite(ledPin, i);   // Establece el brillo a i
9      delay(10);              // Detiene el Programa por 10 ms
```

Este Proyecto lo la habíamos hecho, pero ahora estamos viendo una nueva estructura de la Programación. El "**for**"

```
10    }
11    for (i=255; i>=0; i--) {
12      analogWrite(ledPin, i);
13      delay(10);              // Detiene el Programa por 10 ms
14    }
15  }  // Finaliza función loop( )
```

CIRCUITO PARA NUESTRO PROYECTO

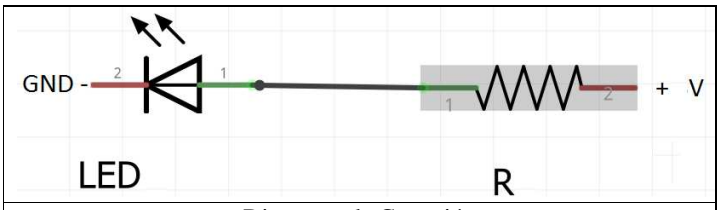


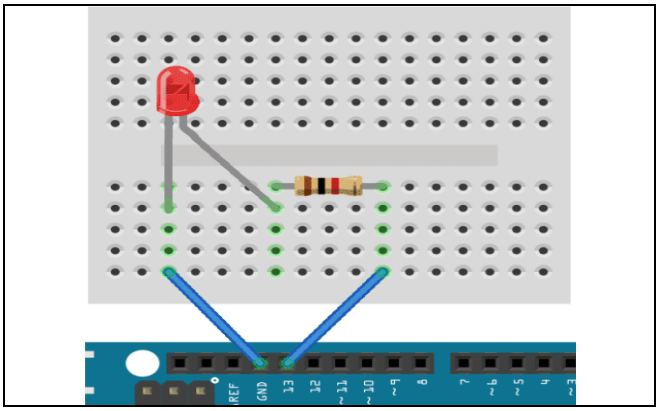
Diagrama de Conexión

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).



10- Prender y Apagar 6 LED en secuencia, como Luces de navidad. Uso de la Estructura Repetitiva "for".

(Programa "001_Led_09_6_Led_Como_luces_Navidad_for")

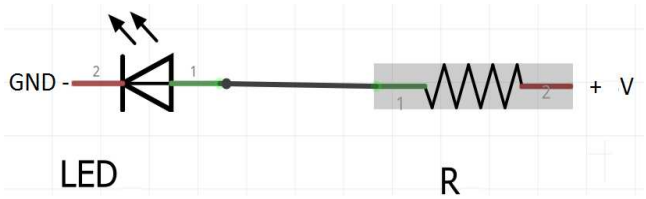
```
1   int TiempoEspera = 100,  // entre cada Led
2       MenorPin  =  2,      // Pina al Que conectare El primer Led
3       MaximoPin = 12,      // Pin al que conectare ultimo LED
4       EstePin;
-
5   void setup( ) {
6     for (EstePin = MenorPin; EstePin <= MaximoPin; EstePin=EstePin + 2) {
7       pinMode(EstePin, OUTPUT);
8     }
9   }
10  void loop( ) {
11    for (EstePin = MenorPin; EstePin <= MaximoPin; EstePin = EstePin + 2) {
12      digitalWrite(EstePin, HIGH);
13      delay(TiempoEspera);
14      digitalWrite(EstePin, LOW);
15    }
16    for (EstePin = MaximoPin; EstePin >= MenorPin; EstePin = EstePin - 2) {
```

```
17      digitalWrite(EstePin, HIGH);
18      delay(TiempoEspera);
19      digitalWrite(EstePin, LOW);
20    }
21  }  // Finaliza función loop( )
```

CIRCUITO PARA NUESTRO PROYECTO

Lista de Materiales

1 Placa Protoboard, 6 Led, 6 Resistencia de 470Ω, 12 Cables Macho-Macho, Placa Arduino, Cable USB



Usar para cada Led, repetir este diagrama, y todos conectados a un mismo GND (Cable negro a la derecha de la imagen). Tal como se aprecia en la Imagen.

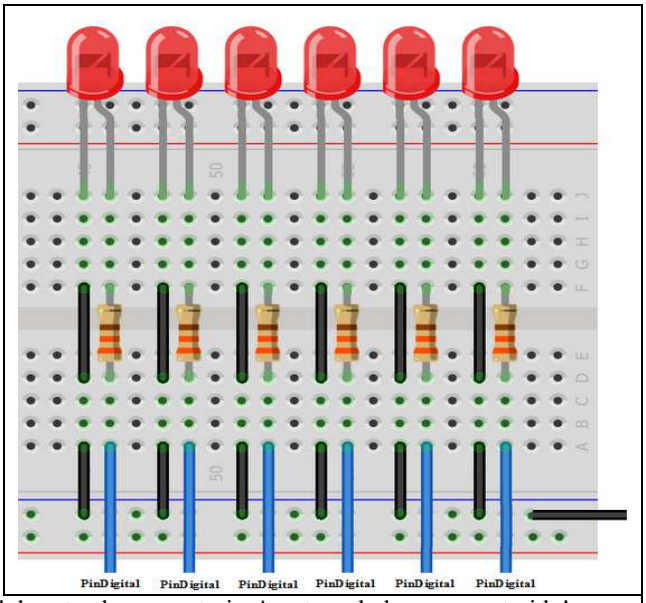


Diagrama de Conexión - Como puede apreciar es muy fácil de entender y construir. Aparte ya lo hemos construido!

11- Selección de opciones, entre 5 secuencias distintas. Mediante el botón seleccionar una de 5 secuencias rítmicas con los LEDs. En este programa, podrás ver, que pasa cuando usas "delay( )" en bucles que solo hacen el bucle, es decir, no utilizas la programación multitarea. Cuando este proyecte este funcionando, veras, que en algunas oportunidades, debes mantener apretado unos segundos el botón para que el programa te detecte. Más adelante, veremos como solucionar este inconveniente.

(Programa "002_Boton_10_Control_de_Opciones_01")

```
1   int TiempoEspera = 100,   // entre cada Led
2      Cantidad_Led =   6,
3      MenorPin  =  2,      // Pina al Que conectare El primer Led
4      MaximoPin =  MenorPin + Cantidad_Led -1,      // Pin al que conectare ultimo LED
5      PinDigitalBoton = 9, //Definimos la entrada del pulsador en el pin 2
6      EstePin,
7      EstadoBoton,
8      presionado,
9      NumeroPulsaciones;
-
10  void setup( ) {
11    pinMode (PinDigitalBoton, INPUT_PULLUP );
12    Serial.begin (9600);
13     for (EstePin = MenorPin; EstePin <= MaximoPin; EstePin=EstePin + 1) {
14       pinMode(EstePin, OUTPUT);
15       Serial.print("Pin Led Habilitado: ");
16       Serial.println(EstePin);
```

```
17    }
18    NumeroPulsaciones = 0;
19    presionado = 0;
20  }
-
21  void loop ( ){
22    EstadoBoton = digitalRead(PinDigitalBoton); // Leemos el PinDigital.
23    if (EstadoBoton == LOW) { //Pregunta si el pulsador está presionado
24       presionado = 1;     //La variable cambia de valor
25       Serial.println("Apreta ");
26    }
27    EstadoBoton = digitalRead(PinDigitalBoton); // Leemos el PinDigital.
28    if (EstadoBoton == HIGH && presionado == 1) {
29       presionado = 0;    //La variable vuelve a su valor original
30       NumeroPulsaciones++;
31       Serial.print("Opciom: ");
32       Serial.println(NumeroPulsaciones);
33    }
34    switch (NumeroPulsaciones){
35      case 0:{ // Apaga Todo
36        ApagaLed_ID(0);
37      }break;
38      case 1:{ // Prende Todo
39        PrendeLed_ID(0);
40      }break;
41      case 2:{ // Prende y apaga
42        ApagaLed_ID(0);
43        delay(TiempoEspera);
44        PrendeLed_ID(0);
45        delay(TiempoEspera);
46      }break;
47      case 3:{ // Prende y Apaga en secuencia de Izquierda a Derecha
48        ApagaLed_ID( TiempoEspera );
49        PrendeLed_ID( TiempoEspera );
50      }break;
51      case 4:{ // Prende  y Apaga en secuencia de Derecha a Izquierda
52        ApagaLed_DI( TiempoEspera );
53        PrendeLed_DI( TiempoEspera );
54      }break;
55      case 5:{ // Prende y apaga en secuencia
56      VaViene(TiempoEspera);
57      }break;
58      default:{ // Reinicia secuencia de Acciones
59         NumeroPulsaciones = 0;
60         Serial.println("Apagado..");
61      }break;
62    }
63  }
-
64  void ApagaLed_ID(int Espera){
65    for (EstePin = MenorPin; EstePin <= MaximoPin; EstePin = EstePin + 1) {
66      digitalWrite(EstePin, LOW);
67      delay(Espera);
68    }
69  }
-
70  void PrendeLed_ID(int Espera){
71    for (EstePin = MenorPin; EstePin <= MaximoPin; EstePin = EstePin + 1) {
72      digitalWrite(EstePin, HIGH);
```

```
73        delay(Espera);
74      }
75    }
-
76    void ApagaLed_DI(int Espera){
77      for (EstePin = MaximoPin; EstePin >= MenorPin; EstePin = EstePin - 1) {
78        digitalWrite(EstePin, LOW);
79        delay(Espera);
80      }
81    }
-
82    void PrendeLed_DI(int Espera){
82      for (EstePin = MaximoPin; EstePin >= MenorPin; EstePin = EstePin - 1) {
84        digitalWrite(EstePin, HIGH);
85        delay(Espera);
86      }
87    }
-
88    void VaViene(int Espera){
89       for (EstePin = MenorPin; EstePin <= MaximoPin; EstePin = EstePin + 1) {
90         digitalWrite(EstePin, HIGH);
91         delay(Espera);
92         digitalWrite(EstePin, LOW);
93       }
94      for (EstePin = MaximoPin; EstePin >= MenorPin; EstePin = EstePin - 1) {
95         digitalWrite(EstePin, HIGH);
96         delay(Espera);
97         digitalWrite(EstePin, LOW);
98       }
99    }
```

CIRCUITO PARA NUESTRO PROYECTO

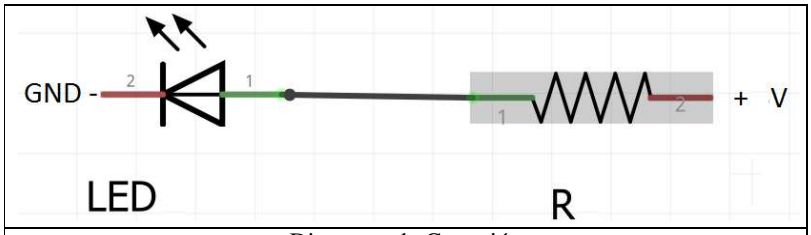


Diagrama de Conexión

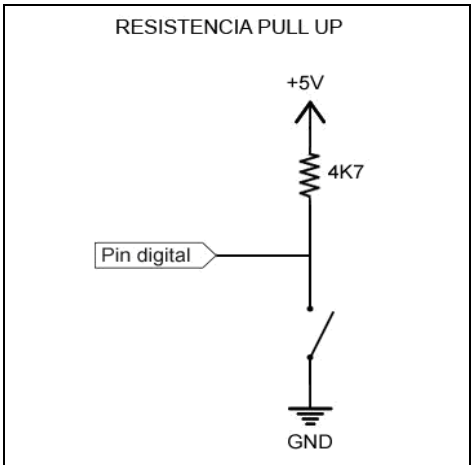


Diagrama de conexión del Botón

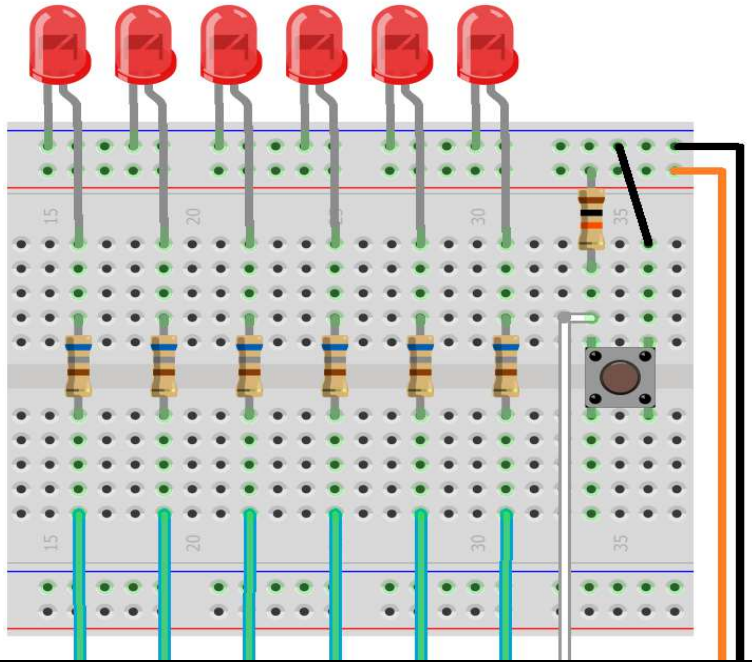
Como puede apreciar, las conexiones son muy fáciles de entender y construir.
Para cada Led, repetir este diagrama, y todos conectados a un mismo GND (Cable negro).
Cable Rojo debe conectarse a 5V, y el cable Blando, se conecta al PIN configurado para el BOTON, y los Cables Verdes, a cada PIN de LED, configurados en el Programa.

<u>Lista de Materiales</u>

1 Placa Protoboard, 6 Led, 6 Resistencia de 470Ω (Para Led), 1 Resistencia de 4K7 Ω (Para el Botón), 10 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).

Notar que el botón, tiene una conexión del tipo "**Pull_Up**" (5V-Resistencia-PinControl-Boton-GND).



01000110 01100101 01101100 01101001 01100011 01101001 01110100 01100001 01100011 01101001 01101111 01101110 01100101 01110011 00101110 00100001

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