Práctica 6: Semáforo con petición de peatón

**Descripción:**

   Realizar un programa que simule el funcionamiento de un semáforo con pulsador de peatón. (solo luces vehículo).

***Entradas/salidas:***

* Led rojo - pin 10
* Led ambar - pin 12
* Led verde - pin 13
* Pulsador - pin 2

***Funcionamiento:***

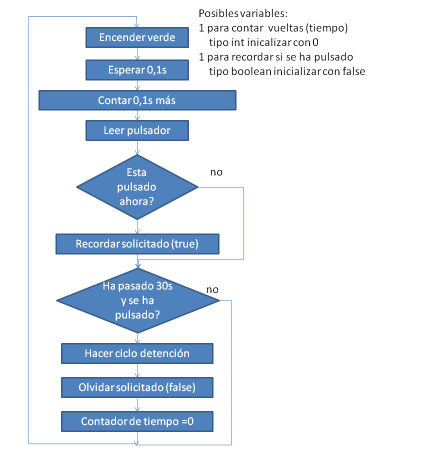
   Ciclos-tiempos si no hay pulsación:

      Verde (constantemente)

    Si hay pulsación:

       Seguirá en verde si no ha transcurrido un tiempo mínimo de 30s. - Ambar (2s) - Rojo (10s) - Ambar intermitente (5s)

       y se considerará final del efecto de pulsación.



Práctica 7: Control de un led con pulsador

**Descripción:**

   Realizar un programa que controle el encendido y apagado de un led mediante la pulsación de un pulsador.

***Entradas/salidas:***

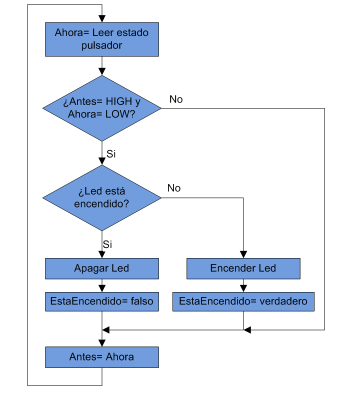
* Pulsador - pin 2
* Led - pin 13

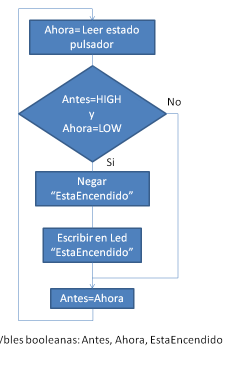
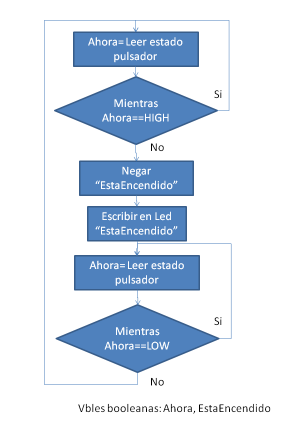
***Funcionamiento:***

* **Una pulsación enciende, la siguiente apaga, la siguiente enciende .....**

**Pistas:**

* **Hay que diferenciar entre pulsaciones, no estado del pulsador...**
* **e puede realizar la acción de encendido y apagado en el momento de la pulsación o de la liberación**(Transiciones HIGH-LOW o LOW-HIGH)...
* Se necesitarán variables para diferenciar el valor anterior del actual.



Práctica 7\_2: Control de un led con dos pulsadores

**Descripción:**

   Realizar un programa que controle el encendido y apagado de un led mediante la pulsación de dos pulsadores.

***Entradas/salidas:***

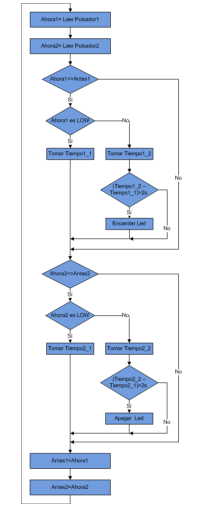
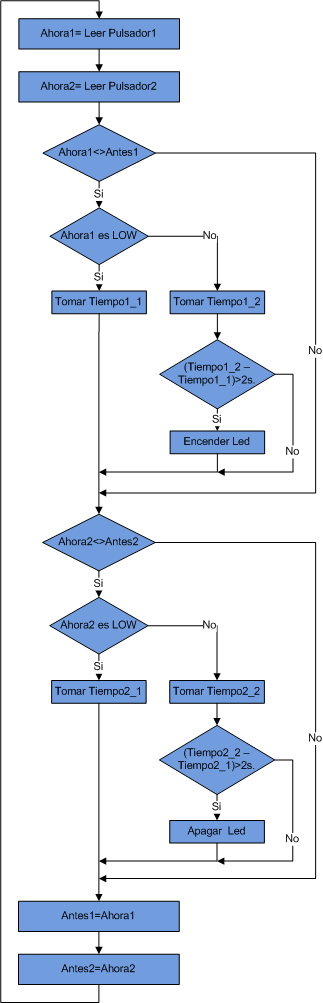
* Pulsador encendido - pin 2
* Pulsador apagado  - pin 3
* Led - pin 13

***Funcionamiento:***

* El led se enciende si se presiona durante al menos 2 segundos el pulsador de encendido.
* El led se apaga si se presiona durante al menos 2 segundos el pulsador de apagado.
* Si cualquier pulsación se libera antes de 2 segundos no debe cambiar el estado del led.

.

Pista: Diagrama de flujo dos versiones (1 decisión al liberar pulsador - 2 cuando cumple 2s.)

Práctica 8: Medición del tiempo de pulsación

**Descripción:**

   Realizar un programa que mida y comunique al ordenador mediante el objeto Serial, la duración de una pulsación.

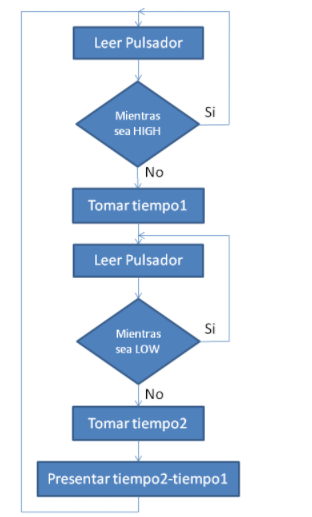
***Entradas/salidas:***

* Pulsador - pin 2

***Funcionamiento:***

   El monitor serial del IDE de Arduino debe mostrar el tiempo de cada pulsación en segundos o ms (lo más adecuado).

   Se utilizará la función millis()



Práctica 9: Control gradual de un led

**Descripción:**

   Realizar un programa que encienda y apague un led mediante un pulsador de forma gradual.

***Entradas/salidas:***

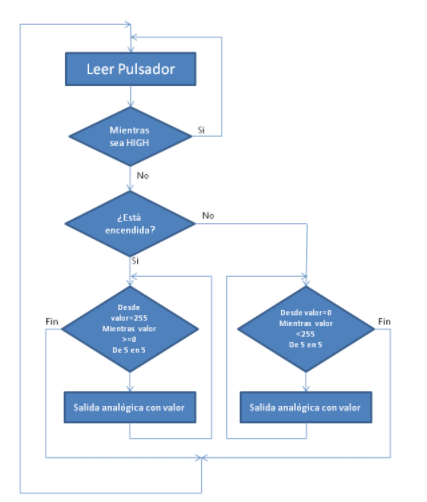
* Pulsador - pin 4
* Led - pin 11 ~

***Funcionamiento:***

   El encendido debe realizarse de forma gradual desde completamente apagado a completamente encendido.

   El apagado debe realizarse de forma gradual desde completamente encendido a completamente apagado.

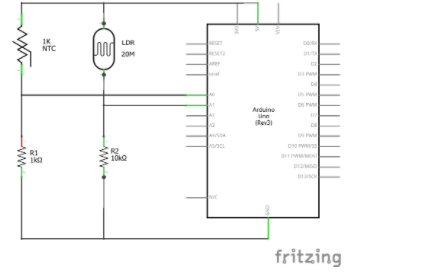
   Se utilizará la función analogWrite() sobre una salida válida para PWM (salida marcada con ~).



## Práctica 10: Lectura de entradas analógicas

**Descripción:**

  Realizar un programa que lea las entradas analógicas A0 y A1 y presente por medio del monitor serial los valores tanto núméricos (convertido) como voltaje presente en la entrada (cálculado).



***Entradas/salidas:***

* LDR - pin A1
* NTC - pin A0

***Funcionamiento:***

   Las entradas analógicas en arduino (A0 - A5) son accesibles desde un conversor analógico-digital de 10 bits que realiza la conversión del voltaje presente en una entrada (de 0 a 5V) a un número binario de 10 bits (desde 0 a 1023). De lo que se obtiene:

(Un incremento de  en la entrada implica un incremento de una unidad en el número convertido)

   El muestreo, cuantificación y codificación de una tensión analógica presente en una entrada se realiza a través de la función: analogRead(pin) que devuelve el **número entero** (0 a 1023) correspondiente.

   La información presentada en el monitor serial debe tener la forma:

LDR: *número* - *tensión* V / NTC: *número* - *tensión*V

Nota: Para calcular la tensión se necesitará usar el tipo **float**

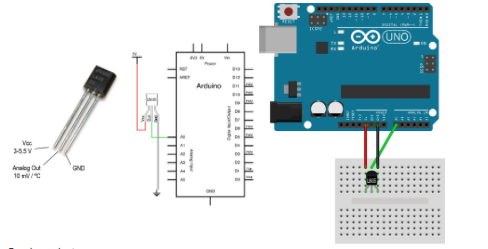
Práctica 11: Termómetro y termostato

**Descripción:**

  Realizar un programa que lea la temperatura por medio del sensor LM35 y sirva como termómetro y termostato.

***Entradas/salidas:***

* LM35 - pin A0



***Funcionamiento:***

   El dispositivo LM35 es un sensor de temperatura barato, sencillo y fiable que podemos usar con arduino de forma fácil a través de las entradas analógicas.

   Dicho sensor, bien alimentado, entrega en su salida una tensión de 10mV/ºC

 - Funcionamiento como termómetro: Enviar por medio del monitor serial la temperatura en ºC (con 1 decimal).

 - Funcionamiento como termostato: Encender un led (p.ej. pin 13 incluido en arduino) solo cuando la temperatura esté por debajo de 20ºC.

El termostato se suele realizar con histéresis, p.ej. se enciende el led cuando baje de 20ºC y se apaga cuando suba por encima de 22ºC.

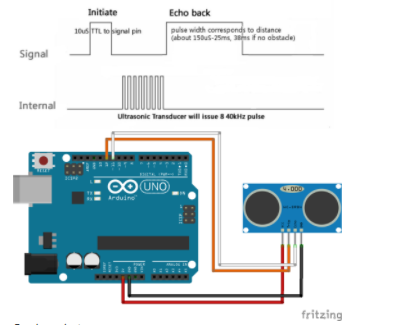
Práctica 12: Ultrasonidos, HC-SR04 y arduino.

**Descripción:**

  Realizar un programa que mida la distancia a un objeto y lo muestre por medio del monitor serial.

***Entradas/salidas:***

* Trig - salida (elección libre)
* Echo - entrada (elección libre)



***Funcionamiento:***

   El **sensor HC-SR04** es un módulo que incorpora un par de transductores de ultrasonido que se utilizan de manera conjunta para determinar la distancia del sensor con un objeto colocado enfrente de este. Un transductor (emisor) emite un pulso de ultrasonidos que se propaga a la velocidad Vs=343m/s aprox. Cuando este sonido "choca" con un objeto, parte del sonido es reflejado en dirección contraria, de manera que un transductor (receptor) reacciona ante el "rebote", pudiendo medir de esta forma el tiempo transcurrido en el viaje ida-vuelta del pulso ultrasónico.

Con cálculos sencillos se puede, por tanto, obtener la distancia a la que se encuentra el objeto.

***Ejemplo básico de uso:***

void **setup**(){

  pinMode(Trig, OUTPUT); /\*activación del pin "Trigger" como salida: para el pulso ultrasónico\*/

  pinMode(Echo, INPUT); /\*activación del pin "Echo" como entrada: tiempo del rebote del ultrasonido\*/  
}

void **loop**(){

  digitalWrite(Trig,LOW); /\* Por cuestión de estabilización del sensor\*/

  delayMicroseconds(5);

  digitalWrite(Trig, HIGH); /\* envío del pulso ultrasónico\*/

  delayMicroseconds(10);

  tiempo=pulseIn(Echo, HIGH); /\* Función para medir la longitud del pulso entrante. Mide el tiempo  
 que transcurrido entre el envío del pulso ultrasónico y cuando el sensor recibe el rebote, es decir:  
  desde que el pin Echo empieza a recibir el rebote, HIGH, hasta que deja de hacerlo, LOW, la longitud del pulso entrante\*/  
....

**tiempo** tendrá el valor expresado en microsegundos que ha tardado en ir y volver el pulso de sonido (doble trayecto).

con dicho valor calcular la distancia a la que está el objeto y expresarlo de la mejor forma posible (cm, m, ....)

## Práctica 12\_2: Asistencia al aparcamiento

Genera un aviso luminoso o acústico variable según distancia al objeto (rango desde 1m a 5 cm como mínimo).

Práctica 13: Sensores de gas y Co2

Comprueba y visualiza en el monitor serial, la medida de los sensores de gas y CO2. (MQ-2 y MQ-7)

* El MQ-4 es sensible al gas natural (metano).
* El MQ-7 es sensible al monóxido de carbono (CO)
* Buen tutorial sobre los sensores de gas y arduino, en el que se detalla como hallar la expresión matématica para calcular la medeida de concentración de gas (p.p.m.). <https://naylampmechatronics.com/blog/42_Tutorial-sensores-de-gas-MQ2-MQ3-MQ7-y-MQ13.html>
* Tienen cuatro pines, 2 de alimentación (VCC o 5V y GND), una salida analógica y una salida digital.
* La salida digital responde con un LOW, o un HIGH, cuando la concentración del gas detectado esté por debajo o por encima de un valor predeterminado por medio de la resistencia ajustable integrada en el módulo. (Para Saber que concentración es esa, sería necesario comparar con un detector calibrado o medir la tensión analogica en el momento del cambio de la digital y obtener la concentración por medio de la gráfica suministrada por el fabricante).

## Práctica 14: Sensores de sonido

Comprueba y visualiza en el monitor serial, la medida del sensor de sonido.

Tiene cuatro pines, 2 de alimentación (VCC o 5V y GND), una salida analógica y una salida digital.

La salida digital responde con un LOW, o un HIGH, cuando el nivel de sonido detectado está por debajo o por encima de un valor predeterminado por medio de la resistencia ajustable integrada en el módulo. (Para saber que nivel es ese, sería necesario comparar con un sonómetro calibrado o medir la tensión analogica en el momento del cambio de la digital ).

- Elimina cualquier texto adicional enviado por el Objeto Serial, y deja solo los envíos del nº obtenido en la conversión ADC, sin retardos entre envíos (ningún delay). Prueba en el IDE de Arduino la opción Herramientas -> Serial Plotter.

## Práctica 15: Módulo RFID/NFC, tarjetas sin contacto.

Lee el apartado Módulo RFID/NFC, y reliza las operaciones descritas para incluir la librería y modificar los archivos .h y .cpp necesarios para poder funcionar con arduino.

Prueba del módulo RFID, consigue leer y visualizar en el monitor serial, el código ID de las tarjetas/llaveros por medio del ejemplo readMifareTargetID (menú: ***Archivo->Ejemplos->PN532\_SPI->readMifareTargetID***). Puedes obtener incluso el ID de tu tarjeta de transportes.

Apunta el ID de, por lo menos, 4 llaveros.

Crea un programa con un array de números ID (tantos como hayas apuntado) y un array de nombres de las mismas dimensiones. El programa basado en el ejemplo anterior, usado para leer los ID, deberá funcionar como un control de acceso de manera que, después de cada lectura, se busque en el array de números y si encuentra coincidencia presente un saludo al "nombre" del array de nombres que esté en la misma posición y active una salida digital (abrepuertas) durante un segundo. Si no se encuentra el ID leido en el array de números debe presentar un mensaje de "no autorizado".