UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y CC. Computación
E.T.S.I. Informática

Software para Sistemas Empotrados y Dispositivos Móviles

Sistemas Empotrados. Arduino

Vicente Benjumea

Planificación

- **Práctica 1:** ejercicios 1–5.
- **Práctica 2:** ejercicios 6–10.
- **Práctica 3:** ejercicios 11–13.
- **Práctica 4:** ejercicios 14–16.
    - **Evaluación:** la realización de esta práctica es *evaluable*. Es necesario un trabajo *no presencial previo* por parte del alumno antes de su realización.

Material

Material utilizado en los ejercicios 1–15:

- 1 Arduino UNO
- 1 Placa de Prototipado
- 3 Resistencias de 470Ω (1/4W)
- 3 Resistencias de 1KΩ (1/4W)
- 3 Resistencias de 10KΩ (1/4W)
- 3 LEDs Rojos
- 1 LED Verde
- 3 Botón/Pulsador
- 1 Potenciómetro (10KΩ ó 100KΩ)
- 1 Sensor de Luz (LDR)
- 1 Sensor de temperatura (Thermistor NTC NTCLE100E3103JB0)
- 1 Transistor NPN (PN2222A)
- 1 Diodo (4004)
- 1 Relé (Bobina de 5V ó 6V, carga de 10A 240V-AC)
- 1 Manguera de 3 cables (tierra, fase, neutro) de 1.5mm con enchufes macho y hembra.
- 1 punto de luz de 230V.
- 1 Secador de pelo de mano de al menos 1000W.
- 1 Caja de plástico transparente de $57 \times 39 \times 28$ cm^3.
- Cables de conexión de prototipos.

Esta obra se encuentra bajo una licencia Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional (CC BY-NC-SA 4.0) de Creative Commons. Véase http://creativecommons.org/licenses/by-nc-sa/4.0/deed.es_ES

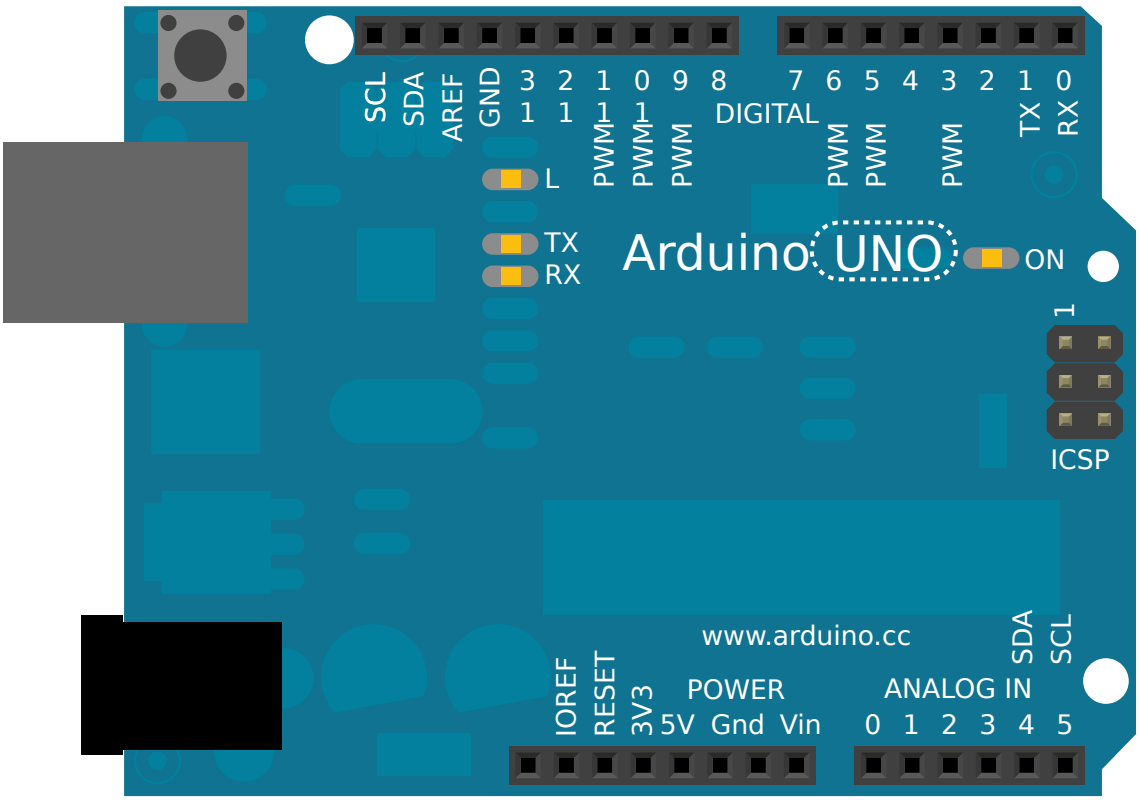
Recursos en la Web

- http://arduino.cc/
- http://arduino.cc/en/Reference/HomePage
- http://playground.arduino.cc/Code/AvoidDelay
- http://www.tiendaderobotica.com/download/Libro_kit_Basico.pdf
- Arduino Programming Notebook. Brian W. Evans.
  Trad.: J.M. Ruiz Gutiérrez. Adap.: J.M. Escuder Martínez.

  http://www.ardumania.es/wp-content/uploads/2011/10/Arduino_programing_notebook_ES.pdf

Introducción. Arduino UNO

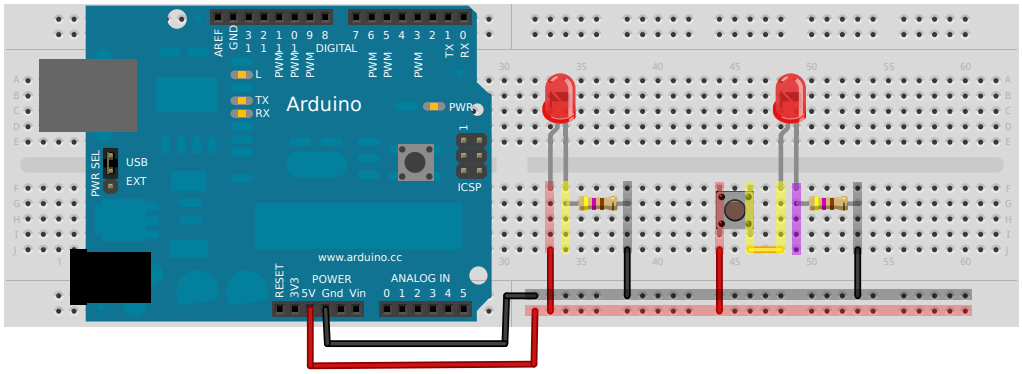
- Conocimiento de *Arduino UNO*.
  - Micro-controlador: ATmega328 de Atmel
  - Voltaje Operativo: 5V
  - Voltaje de Entrada (recomendado): 7-12V (limites): 6-20V
  - Pines de E/S Digital: 14 (6 proporcionan salida PWM)
  - Pines de Entrada Analógica: 6
  - Intensidad de Corriente por cada Pin de E/S: 40 mA
  - Intensidad de Corriente para el pin de 3.3V: 50 mA
  - Intensidad de Corriente para el pin de 5V: 200 mA
  - Memoria Flash: 32 KB (ATmega328) (0.5 KB es utilizada por el bootloader)
  - SRAM: 2 KB (ATmega328)
  - EEPROM: 1 KB (ATmega328)
  - Velocidad del Reloj: 16 MHz
  - Botón de Reset
- referencias:
  - `http://arduino.cc/en/Main/arduinoBoardUno`
  - `http://playground.arduino.cc/Main/ArduinoPinCurrentLimitations`



Introducción. Circuitos Electrónicos. Ley de Ohm (I)

Ley de Ohm	Ley de Joule	1ª Ley de Kirchoff (punto)	2ª Ley de Kirchoff (bucle)
$V = I \cdot R$	$P = I \cdot V$	$\sum_{k=1}^{n} I_k = 0$	$\sum_{k=1}^{n} V_k = 0$

- Placas de prototipado. Permiten la conexión y desconexión de dispositivos.
  - Conexiones internas de las filas de alimentación y de las columnas de conexión.
- Código de colores: Negro: 0V (Gnd) Rojo: 5V (Vcc)
- Código de colores: Azul: Salida (de Arduino) Verde: Entrada (a Arduino)
- Cualquier manipulación del circuito debe hacerse con la **ALIMENTACIÓN DESCONECTADA**.
- La conexión directa de 5V o de cualquier pin de salida de Arduino a Masa (*Gnd*, 0V) provocará un **CORTOCIRCUITO** que dañará el dispositivo.
- Las resistencias se caracterizan por ofrecer una determinada oposición/resistencia (medida en *Ohmios*) al paso de la corriente, con un límite máximo de corriente determinado por la potencia que puede disipar (medida en *Watios*) [usualmente $1/4W$, $1/2W$ $1W$].
- Un LED es un *diodo emisor de luz*, que deja pasar la corriente en un único sentido (el pin largo es el positivo/ánodo y el pin corto es el negativo/cátodo), con una caída de potencial de 2V y un máximo de 20mA de corriente. Si se supera dicho límite, el dispositivo podría dañarse.
- Un botón/pulsador permite conectar dos puntos de un circuito (cerrar un circuito) mientras se encuentra pulsado, estando desconectado (circuito abierto) en otro caso.



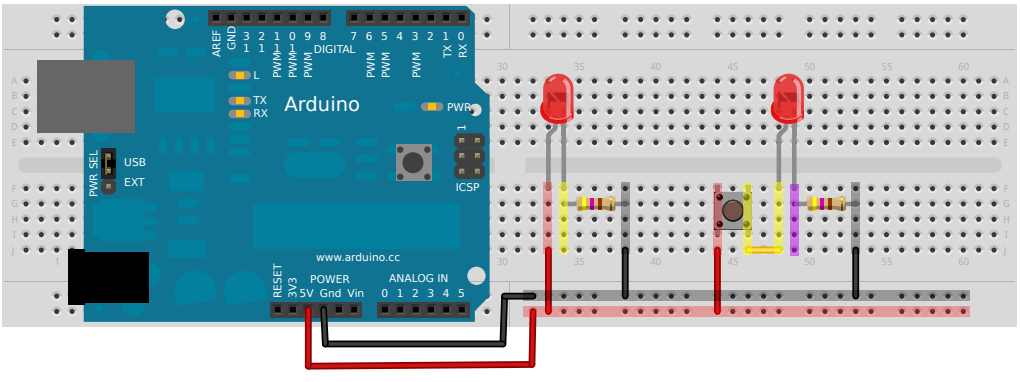
Ejercicio 1. Circuitos Electrónicos. Ley de Ohm (II)

Ley de Ohm	Ley de Joule	1ª Ley de Kirchoff (punto)	2ª Ley de Kirchoff (bucle)
$V = I \cdot R$	$P = I \cdot V$	$\sum_{k=1}^{n} I_k = 0$	$\sum_{k=1}^{n} V_k = 0$

- Cualquier manipulación del circuito debe hacerse con la **ALIMENTACIÓN DESCONECTADA**.
- **Circuito 1**: conecte el pin positivo de un LED directamente a 5V, y el pin negativo del LED a una resistencia de 470Ω, la cual se conecta a masa (Gnd) con el otro pin.
- **Funcionamiento 1**: el flujo de corriente pasa, desde los 5V, a través del LED y la resistencia hasta masa (Gnd), completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia.
- Pruebe el circuito 1 con resistencias de 470Ω, 1KΩ y 10KΩ, y analice su comportamiento:
  - La primera ley de Kirchoff nos dice que la intensidad de corriente que pasa por el LED es la misma que la que pasa por la resistencia, ya que no hay ninguna bifurcación.
  - La segunda ley de Kirchoff nos dice que si en un bucle hay un potencial de +5 Voltios, dicho potencial debe caer hasta cero al cerrar el bucle, por lo que entre el LED y la resistencia debe haber una caída de potencial de 5 Voltios. Si el LED proporciona, por su construcción, una caída de potencial de 2 Voltios, entonces habrá una caída de potencial de 3 Voltios en la resistencia.

Resistencia	Ley de Ohm ($V = I \cdot R$)	I_{LED}	Ley de Joule ($P = I \cdot V$)	P_{LED}	P_{Circ}
150Ω	$3V = 0.020A \cdot 150\Omega$	$0.020A$	$0.060W = 0.020A \cdot 3V$	$0.040W$	$0.100W$
470Ω	$3V = 0.006A \cdot 470\Omega$	$0.006A$	$0.018W = 0.006A \cdot 3V$	$0.012W$	$0.030W$
1KΩ	$3V = 0.003A \cdot 1000\Omega$	$0.003A$	$0.009W = 0.003A \cdot 3V$	$0.006W$	$0.015W$
10KΩ	$3V = 0.0003A \cdot 10000\Omega$	$0.0003A$	$0.0009W = 0.0003A \cdot 3V$	$0.0006W$	$0.0015W$

- **Circuito 2**: conecte el pin positivo de un LED a 5V a través de un botón, y el pin negativo del LED a una resistencia de 470Ω, la cual se conecta a masa (Gnd) con el otro pin.
- **Funcionamiento 2**: mientras el botón no esté pulsado, los dos puntos del circuito se encuentran desconectados, por lo que el circuito estará abierto (no tiene conexión cerrada con 5V), no habrá flujo de corriente eléctrica por él y no se encenderá el LED. Sin embargo, cuando el botón está pulsado, se cierra el circuito (conectando los puntos anteriormente desconectados), permitiendo el paso del flujo de corriente eléctrica desde los 5V, a través del botón, el LED y la resistencia hasta masa (Gnd), completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia.

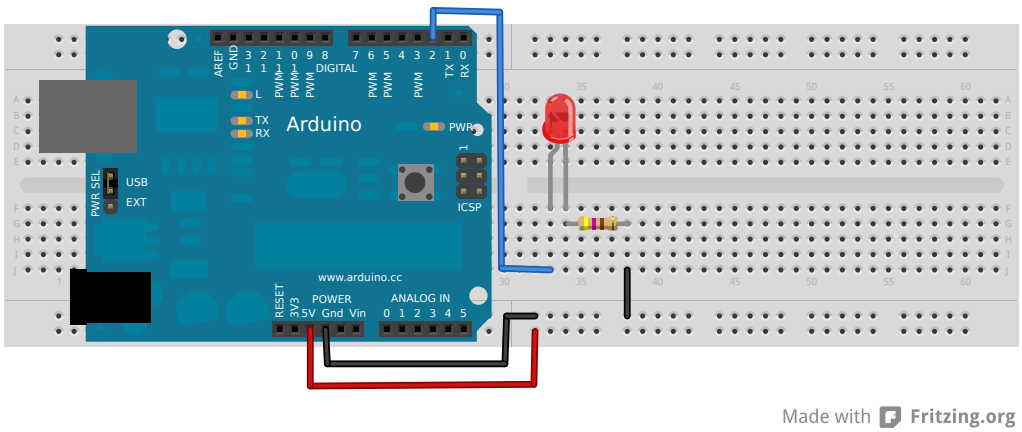


Ejercicio 2. Salida Digital. LED

- **Circuito LED**: conecte el pin positivo de un LED al pin digital número 2 de Arduino, y el pin negativo del LED a una resistencia de 470Ω, la cual se conecta a masa (*Gnd*) con el otro pin.
- **Funcionamiento LED**: cuando se active 5V (`HIGH`) en el pin digital de salida correspondiente de Arduino, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia. En caso de que se active 0V (`LOW`) en el pin digital de salida, entonces no habrá flujo de corriente y no se encenderá el LED.
- **Sketch**: diseñe un programa que encienda un LED conectado al pin digital de salida número 2.
- Entorno para el diseño de circuitos: Fritzing.
- Entorno de desarrollo: Arduino IDE .
  - Hay que especificar el tipo de Arduino y el puerto de conexión.
  - Lenguaje de programación *C* o *C++*. Librerías propias de Arduino.
- *Ayuda:*
  - Tipos simples, signo y tamaños en bytes

boolean	char	byte	short	u-short	int	u-int	word	long	u-long	float	double
1	±1	+1	±2	+2	±2	+2	+2	±4	+4	±4.6	±4.6

  - `setup() {...}` inicialización de todo el proceso. Se ejecuta una única vez al comienzo.
  - `loop() {...}` proceso iterativo que se ejecuta indefinidamente. Debe ejecutar de forma iterativa (cíclica) todas las tareas que componen el proceso.
  - Almacenamiento de datos en *variables globales* accesibles por `setup()` y `loop()`.
  - `pinMode(pin, INPUT/OUTPUT)`; especifica el modo (entrada/salida) en el que se utilizará un determinado pin digital de Arduino. Los pines digitales están en modo de entrada (INPUT) por defecto.
  - `digitalWrite(pin, LOW/HIGH)`; permite escribir un valor (`LOW` o `HIGH`) en un pin digital de salida, que corresponde a un voltaje de 0V o 5V. Dicho voltaje permanece asociado al pin hasta que se escribe otro valor.



Solución

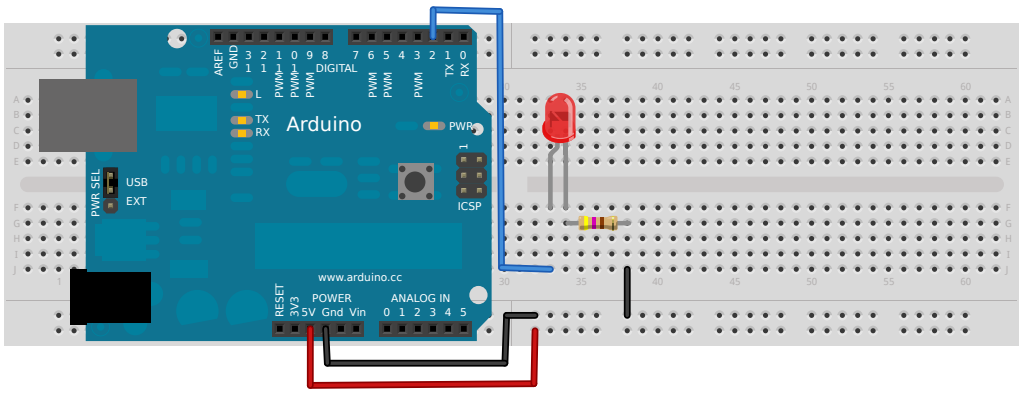
```
/*** Salida Digital. LED ***/
const byte LED1PIN = 2;
void setup()
{
    pinMode(LED1PIN, OUTPUT);
    digitalWrite(LED1PIN, HIGH);
}
void loop()
{
    /* vacio */
}
```

Solución Alternativa

```
/*** Salida Digital. LED ***/
const byte LED1PIN = 2;
void setup()
{
    pinMode(LED1PIN, OUTPUT);
}
void loop()
{
    digitalWrite(LED1PIN, HIGH);
}
```

Ejercicio 3. Salida Digital. Control de Tiempo (I)

- **Circuito LED**: conecte el pin positivo de un LED al pin digital número 2 de Arduino, y el pin negativo del LED a una resistencia de 470Ω, la cual se conecta a masa (*Gnd*) con el otro pin.
- **Funcionamiento LED**: cuando se active 5V (`HIGH`) en el pin digital de salida correspondiente de Arduino, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia. En caso de que se active 0V (`LOW`) en el pin digital de salida, entonces no habrá flujo de corriente y no se encenderá el LED.
- **Sketch**: diseñe un programa que haga parpadear un LED, conectado al pin digital de salida número 2, con un periodo de 1 sg, debiendo estar la mitad del periodo (500 ms) encendido, y la otra mitad (500 ms) apagado. El programa deberá controlar el periodo de activación de la tarea que controla el LED, así como el estado del LED.
- *Ayuda:*
  - `pinMode(pin, INPUT/OUTPUT)`; `digitalWrite(pin, LOW/HIGH)`;
  - `delay(ms)`; pausa la ejecución del programa por la cantidad de milisegundos (`unsigned long`) especificada como parámetro.
  - Hay que considerar que la utilización de `delay()` tiene numerosas desventajas, ya que pausa la ejecución de la mayoría de las tareas. En la mayoría de las situaciones, la función `millis()` proporciona un esquema más adecuado para la gestión del tiempo.



Solución

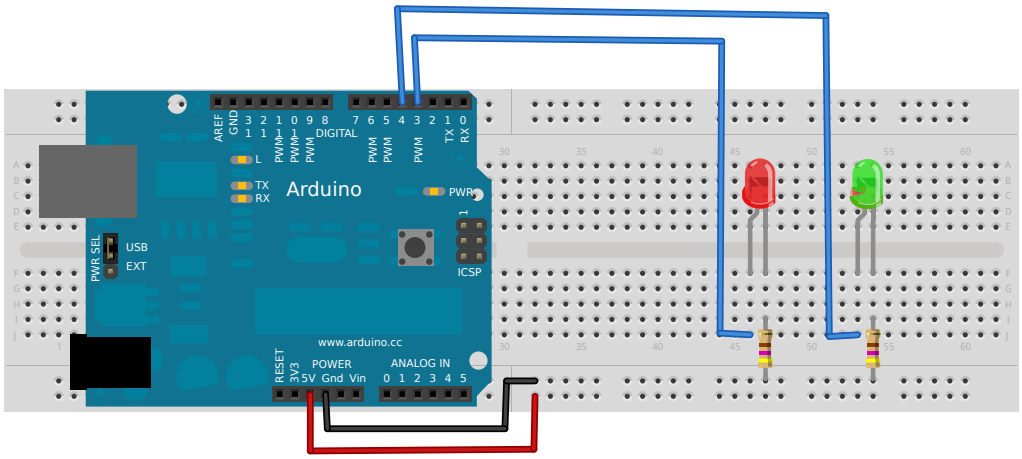
```
/*** Salida Digital. Control de Tiempo (I) ***/
const byte LED1PIN = 2;
const unsigned long LED1PERIODO = 1000;
const unsigned long LED1PERIODO2 = (LED1PERIODO/2);
void setup()
{
    pinMode(LED1PIN, OUTPUT);
}
void loop()
{
    digitalWrite(LED1PIN, HIGH);
    delay(LED1PERIODO2);
    digitalWrite(LED1PIN, LOW);
    delay(LED1PERIODO2);
}
```

Solución Alternativa

```
/*** Salida Digital. Control de Tiempo (I) ***/
/*
 * Solucion alternativa almacenando el estado del LED en una variable
 */
const byte LED1PIN = 2;
const unsigned long LED1PERIODO = 1000;
const unsigned long LED1PERIODO2 = (LED1PERIODO/2);
byte estado = LOW;
void setup()
{
    pinMode(LED1PIN, OUTPUT);
}
void loop()
{
    estado = (estado == LOW ? HIGH : LOW);
    digitalWrite(LED1PIN, estado);
    delay(LED1PERIODO2);
}
```

Ejercicio 4. Salida Digital. Control de Tiempo (II)

- **Circuito LED**: conecte el pin positivo de un LED rojo al pin digital número 3 de Arduino, y el pin negativo del LED a masa (*Gnd*) a través de una resistencia de 470Ω. Repita la misma conexión para un LED verde conectado al pin digital número 4 de Arduino.
- **Funcionamiento LED**: cuando se active 5V (`HIGH`) en el pin digital de salida correspondiente de Arduino, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia. En caso de que se active 0V (`LOW`) en el pin digital de salida, entonces no habrá flujo de corriente y no se encenderá el LED.
- **Sketch**: diseñe un programa que haga parpadear un LED rojo conectado al pin digital de salida número 3 y un LED verde conectado al pin digital de salida número 4, con un periodo de 1024 ms y 1458 ms respectivamente, debiendo cada uno estar la mitad del periodo encendido, y la otra mitad apagado. El programa deberá controlar el periodo de activación de las tareas que controlan los LEDs, así como el estado de los LEDs.
- Desde ahora, la utilización de la función `delay(ms)` no está permitida salvo justificación adecuada.
- Programación como **MÁQUINAS DE ESTADO**.
- *Ayuda:*
  - `pinMode(pin, INPUT/OUTPUT)`; `digitalWrite(pin, LOW/HIGH)`;
  - `ms = millis()`; devuelve la cantidad de milisegundos (`unsigned long`) que han transcurrido desde que el programa comenzó su ejecución (desbordamiento a los 50 días).
  - Hay que considerar que la utilización de `delay()` tiene numerosas desventajas, ya que pausa la ejecución de la mayoría de las tareas. En la mayoría de las situaciones, la función `millis()` proporciona un esquema más adecuado para la gestión del tiempo.



Solución

```
/*** Salida Digital. Control de Tiempo (III) ***/
/*
 * En Arduino, no se deben comparar tiempos directamente, ya que
 * existe la posibilidad de *desbordamiento*. En vez de ello, se deben
 * comparar *intervalos* de tiempo (menores que el limite de
 * desbordamiento).
 */
const byte LED1PIN = 3;
const byte LED2PIN = 4;
const unsigned long LED1PERIODO = 1024;
const unsigned long LED1PERIODO2 = (LED1PERIODO/2);
const unsigned long LED2PERIODO = 1458;
const unsigned long LED2PERIODO2 = (LED2PERIODO/2);
byte led1_estado = LOW;
byte led2_estado = LOW;
unsigned long led1_last_ms = 0;
unsigned long led2_last_ms = -LED2PERIODO2;
void setup()
{
    pinMode(LED1PIN, OUTPUT);
    pinMode(LED2PIN, OUTPUT);
}
void loop()
{
    unsigned long curr_ms = millis();
    if (curr_ms - led1_last_ms >= LED1PERIODO2) {
        led1_last_ms += LED1PERIODO2;
        led1_estado = (led1_estado == LOW ? HIGH : LOW);
        digitalWrite(LED1PIN, led1_estado);
    }
    if (curr_ms - led2_last_ms >= LED2PERIODO2) {
        led2_last_ms += LED2PERIODO2;
        led2_estado = (led2_estado == LOW ? HIGH : LOW);
        digitalWrite(LED2PIN, led2_estado);
    }
}
```

Solución Alternativa

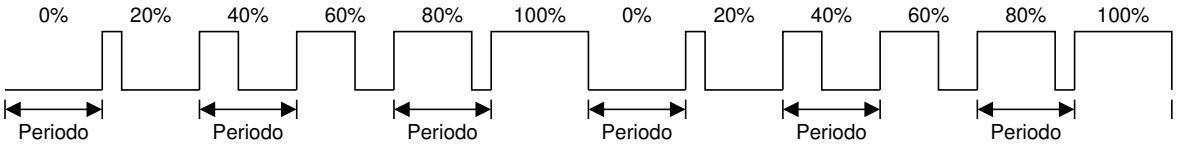
```
/*** Salida Digital. Control de Tiempo (III) ***/
/*
 * En Arduino, no se deben comparar tiempos directamente, ya que
 * existe la posibilidad de *desbordamiento*. En vez de ello, se deben
 * comparar *intervalos* de tiempo (menores que el limite de
 * desbordamiento).
 */
const byte LED1PIN = 3;
const byte LED2PIN = 4;
const unsigned long LED1PERIODO = 1024;
const unsigned long LED1PERIODO2 = (LED1PERIODO/2);
const unsigned long LED2PERIODO = 1458;
const unsigned long LED2PERIODO2 = (LED2PERIODO/2);
//-----------------------------------------
struct LED {
    unsigned long period_ms;
    unsigned long last_ms;
    byte pin;
    byte estado;
};
void setup_led(struct LED& led, byte pin, unsigned long per)
{
    led.period_ms = per;
    led.last_ms = -led.period_ms;
    led.pin = pin;
    led.estado = LOW;
    pinMode(led.pin, OUTPUT);
}
void loop_led(struct LED& led, unsigned long curr_ms)
{
    if (curr_ms - led.last_ms >= led.period_ms) {
        led.last_ms += led.period_ms;
        led.estado = (led.estado == LOW ? HIGH : LOW);
        digitalWrite(led.pin, led.estado);
    }
}
//-----------------------------------------
struct LED led_1;
struct LED led_2;
void setup()
{
    setup_led(led_1, LED1PIN, LED1PERIODO2);
    setup_led(led_2, LED2PIN, LED2PERIODO2);
}
void loop()
{
    unsigned long curr_ms = millis();
    loop_led(led_1, curr_ms);
    loop_led(led_2, curr_ms);
}
```

Ejercicio 5. Ciclo de Trabajo y Monitorización

- **Circuito LED**: conecte el pin positivo de un LED rojo al pin digital número 3 de Arduino, y el pin negativo del LED a masa (*Gnd*) a través de una resistencia de 470Ω. Repita la misma conexión para un LED verde conectado al pin digital número 4 de Arduino.

- **Funcionamiento LED**: cuando se active 5V (`HIGH`) en el pin digital de salida correspondiente de Arduino, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia. En caso de que se active 0V (`LOW`) en el pin digital de salida, entonces no habrá flujo de corriente y no se encenderá el LED.

- **Sketch**: diseñe un programa que haga parpadear un LED verde conectado al pin digital de salida número 4 con un periodo de 800 ms. Además, debe controlar el **ciclo de trabajo**[1], con un periodo de 5000 ms, de un LED rojo conectado al pin digital de salida número 3, de tal forma que el LED rojo se encontrará durante el periodo, y de forma cíclica, en cada una de las siguientes cargas del ciclo de trabajo { 0 %, 20 %, 40 %, 60 %, 80 % y 100 % }.



- **Monitorización**: con un periodo de 1000 ms, se enviará el estado del LED rojo, *(0)* para apagado y *(1)* para encendido, a través del puerto *USB* para que sea mostrado en el terminal de control. Utilice para ello la biblioteca *Serial*.

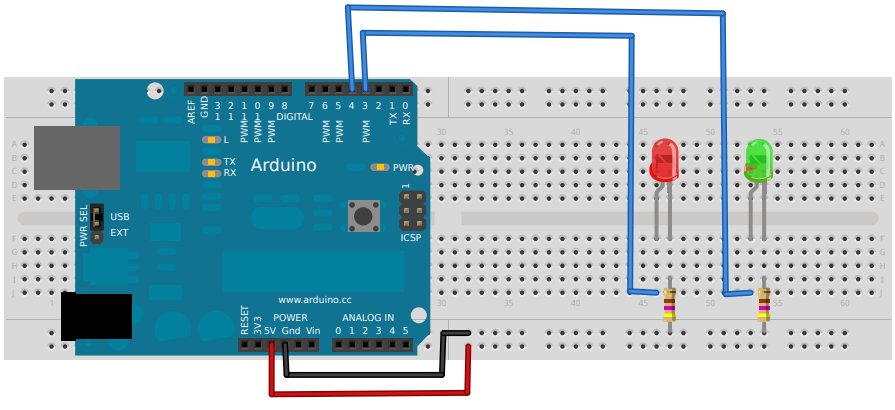
- Se deberá enviar al puerto *USB* al inicio del programa las siguientes líneas de control:

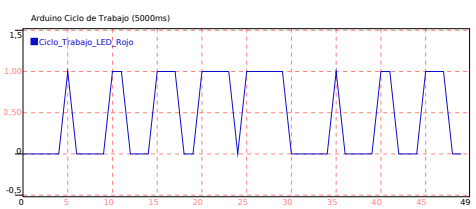
```
#> Arduino Ciclo de Trabajo (5000ms)
#$ -y-1:2 -tCiclo_Trabajo_LED_Rojo
```

- Nótese que no es posible utilizar los pines digitales 0 y 1 si se utiliza el puerto *USB*.

- *Ayuda:*

  - `pinMode(pin, OUTPUT)`; `digitalWrite(pin, LOW/HIGH)`; `ms = millis()`;
  - `Serial.begin(9600)`; `if(Serial){}`; `Serial.println(data)`; `Serial.flush()`;
  - `http://en.wikipedia.org/wiki/Duty_cycle`





[1] Ciclo de Trabajo: el porcentaje de tiempo del periodo de control que el dispositivo se encuentra activo.

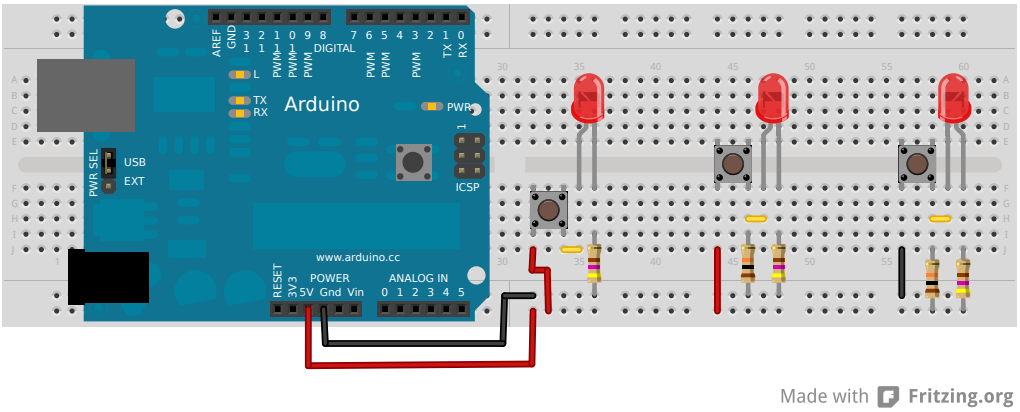
Solución

```
/*** Ciclo de Trabajo y Monitorizacion ***/
const byte LEDR_PIN = 3;
const byte LEDV_PIN = 4;
const unsigned long LEDR_PERIODO_CONTROL = 5000;
const unsigned long LEDV_PERIODO = 800;
const unsigned long LEDV_PERIODO2 = (LEDV_PERIODO/2);
const unsigned long MONITOR_PERIODO = 1000;
const byte CICLO_TRABAJO_INC = 20;
const byte CICLO_TRABAJO_MAX = 100;

//--------------------------------------
struct LED_Verde {
    unsigned long last_ms;
    byte pin;
    byte estado;
};
void setup_ledv(struct LED_Verde& led, byte pin)
{
    led.last_ms = -LEDV_PERIODO2;
    led.pin = pin;
    led.estado = LOW;
    pinMode(led.pin, OUTPUT);
}
void loop_ledv(struct LED_Verde& led, unsigned long curr_ms)
{
    if (curr_ms - led.last_ms >= LEDV_PERIODO2) {
        led.last_ms += LEDV_PERIODO2;
        led.estado = (led.estado == LOW ? HIGH : LOW);
        digitalWrite(led.pin, led.estado);
    }
}
//--------------------------------------
struct LED_Rojo {
    unsigned long last_ms;
    unsigned long duty_cicle_ms;
    byte pin;
    byte estado;
    byte ciclo_trabajo;
};
void setup_ledr(struct LED_Rojo& led, byte pin)
{
    led.last_ms = -LEDR_PERIODO_CONTROL;
    led.duty_cicle_ms = LEDR_PERIODO_CONTROL;
    led.pin = pin;
    led.estado = LOW;
    pinMode(led.pin, OUTPUT);
    led.ciclo_trabajo = 0;
}
void loop_ledr(struct LED_Rojo& led, unsigned long curr_ms)
{
    if (curr_ms - led.last_ms >= LEDR_PERIODO_CONTROL) {
        led.last_ms += LEDR_PERIODO_CONTROL;
        led.duty_cicle_ms = led.ciclo_trabajo * LEDR_PERIODO_CONTROL / 100;
        led.estado = (led.duty_cicle_ms == 0 ? LOW : HIGH);
        digitalWrite(led.pin, led.estado);
        led.ciclo_trabajo += CICLO_TRABAJO_INC;
        if (led.ciclo_trabajo > CICLO_TRABAJO_MAX) {
            led.ciclo_trabajo = 0;
        }
    }
    if (curr_ms - led.last_ms >= led.duty_cicle_ms) {
        led.duty_cicle_ms = LEDR_PERIODO_CONTROL;
        led.estado = LOW;
        digitalWrite(led.pin, led.estado);
    }
}
//--------------------------------------
struct Monitor {
    unsigned long last_ms;
};
void setup_monitor(struct Monitor& mtr)
{
    mtr.last_ms = -MONITOR_PERIODO;
```

```
    Serial.begin(9600);
    if (Serial) {
      Serial.println("#> Arduino Ciclo de Trabajo (5000ms)");
      Serial.println("#$ -y-1:2 -tCiclo_Trabajo_LED_Rojo");
    }
}
void loop_monitor(struct Monitor& mtr, unsigned long curr_ms,
                  const struct LED_Rojo& led)
{
    if (curr_ms - mtr.last_ms >= MONITOR_PERIODO) {
        mtr.last_ms += MONITOR_PERIODO;
        if (Serial) {
            Serial.println(led.estado == LOW ? 0 : 1);
        }
    }
}
//-------------------------------------
struct Monitor   monitor;
struct LED_Verde ledv;
struct LED_Rojo  ledr;

void setup()
{
    setup_ledv(ledv, LEDV_PIN);
    setup_ledr(ledr, LEDR_PIN);
    setup_monitor(monitor);
}

void loop()
{
    unsigned long curr_ms = millis();
    loop_ledv(ledv, curr_ms);
    loop_ledr(ledr, curr_ms);
    loop_monitor(monitor, curr_ms, ledr);
}
```

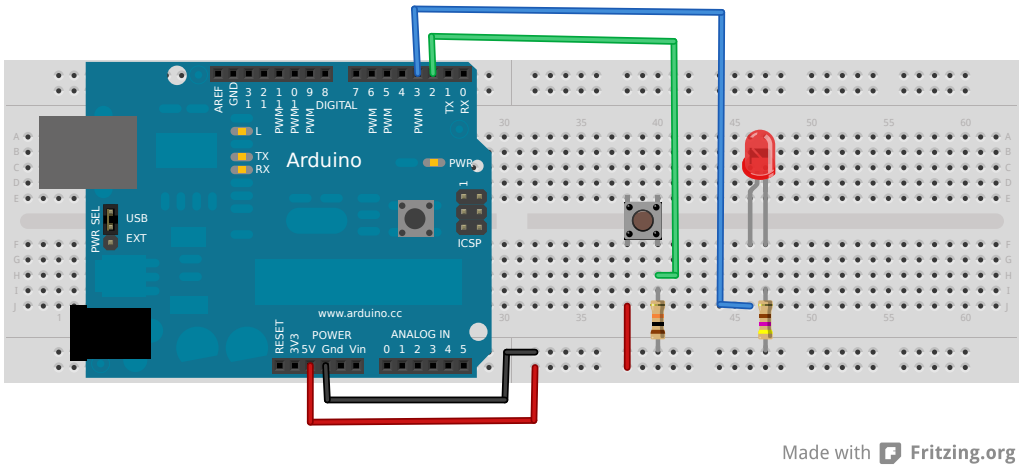
Introducción. Control de Botones

- **Circuito 1** y **funcionamiento 1**: la descripción del primer circuito y su funcionamiento corresponden al *circuito-2* del *ejercicio 1* de esta misma relación de ejercicios.
- **Circuito 2**: conecte el pin positivo de un LED a 5V a través de un botón, y el pin negativo del LED a una resistencia de 470Ω, la cual se conecta a masa (*Gnd*) con el otro pin. Además, conecte una resistencia *Pull-Down* de 10KΩ desde el pin del botón que se conecta al LED, hasta masa (*Gnd*).
- **Funcionamiento 2**: mientras el botón no esté pulsado, el circuito no tiene conexión cerrada con 5V y el LED tiene conectados ambos pines a masa (*Gnd*) a través de sendas resistencias de 470Ω y 10KΩ, por lo que no habrá flujo de corriente eléctrica por él y no se encenderá el LED. Sin embargo, cuando el botón está pulsado, se cierra el circuito, permitiendo el paso del flujo de corriente eléctrica desde los 5V, a través del botón, el LED y la resistencia hasta masa (*Gnd*), completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia. Además, también hay un pequeño paso de corriente por la resistencia *Pull-Down* de 10KΩ, pero al ser la resistencia muy grande, el flujo de corriente es despreciable ($0.0005A$).
- **Circuito 3**: conecte el pin negativo de un LED a masa (*Gnd*) a través de un botón, y el pin positivo del LED a una resistencia de 470Ω, la cual se conecta a 5V con el otro pin. Además, conecte una resistencia *Pull-Up* de 10KΩ desde el pin del botón que se conecta al LED, hasta 5V.
- **Funcionamiento 3**: mientras el botón no esté pulsado, el circuito no tiene conexión cerrada con masa (*Gnd*) y el LED tiene conectados ambos pines a 5V a través de sendas resistencias de 470Ω y 10KΩ, por lo que no habrá flujo de corriente eléctrica por él y no se encenderá el LED. Sin embargo, cuando el botón está pulsado, se cierra el circuito, permitiendo el paso del flujo de corriente eléctrica desde los 5V, a través de la resistencia, el LED y el botón hasta masa (*Gnd*), completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia. Además, también hay un pequeño paso de corriente por la resistencia *Pull-Up* de 10KΩ, pero al ser la resistencia muy grande, el flujo de corriente es despreciable ($0.0005A$).
- La ventaja que se obtiene con la utilización de resistencias *Pull-Down* y *Pull-Up* es que no hay terminales desconectados, por lo que el circuito es más resistente al ruido electromagnético y señales espúreas. Esto es especialmente importante cuando se utiliza un botón como señal de entrada para un pin de Arduino.



Ejercicio 6. Entrada y Salida Digital. Botón-PullDown

- **Circuito LED**: conecte el pin positivo de un LED al pin digital número 3 de Arduino, y el pin negativo del LED a masa (*Gnd*) a través de una resistencia de 470Ω.
- **Funcionamiento LED**: cuando se active 5V (`HIGH`) en el pin digital de salida correspondiente de Arduino, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia. En caso de que se active 0V (`LOW`) en el pin digital de salida, entonces no habrá flujo de corriente y no se encenderá el LED.
- **Circuito Botón-PullDown**: conecte un pin del botón a 5V, y el otro pin del botón a masa (*Gnd*) a través de una resistencia *Pull-Down* de 10KΩ. Además, conecte el pin del botón conectado a la resistencia *Pull-Down* al pin digital número 2 de Arduino.
- **Funcionamiento Botón-PullDown**: mientras el botón no está pulsado, el pin digital de entrada de Arduino está conectado a masa (*Gnd*) a través de una resistencia de 10KΩ, por lo que proporciona una señal de entrada de 0V (`LOW`). Sin embargo, cuando el botón está pulsado, el pin digital de entrada de Arduino se conecta directamente a 5V, por lo que proporciona una señal de entrada de 5V (`HIGH`).
- **Sketch**: diseñe un programa que encienda un LED conectado al pin digital de salida número 3 si el botón conectado al pin digital de entrada número 2 está pulsado, y lo apague en caso contrario.
- Pruebe también el circuito de botón anterior eliminado la resistencia *Pull-Down*.
- La ventaja que se obtiene con la utilización de resistencias *Pull-Down* y *Pull-Up* es que no hay terminales desconectados, por lo que el circuito es más resistente al ruido electromagnético y señales espúreas. Esto es especialmente importante cuando se utiliza un botón como señal de entrada para un pin de Arduino.
- *Ayuda:*
  - `pinMode(pin, INPUT/OUTPUT)`; `digitalWrite(pin, LOW/HIGH)`;
  - `digitalRead(pin)`; permite leer un valor (`LOW` o `HIGH`) de un pin digital de entrada, que depende del voltaje ($\leq 2V$ ó $\geq 3V$) de la señal que haya en dicho pin.



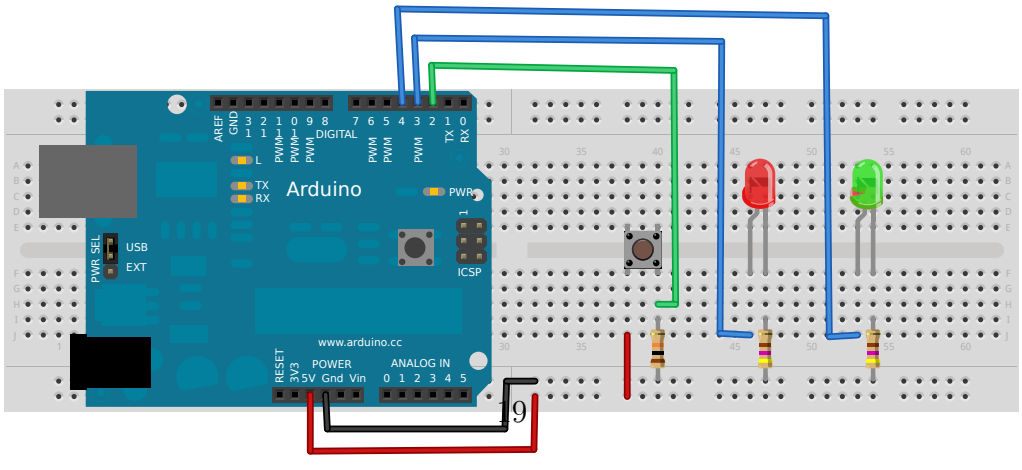
Solución

```
/*** Entrada y Salida Digital. Boton-PullDown ***/
const byte BOTON1PIN = 2;
const byte LED1PIN = 3;
void setup()
{
    pinMode(BOTON1PIN, INPUT);
    pinMode(LED1PIN, OUTPUT);
}
void loop()
{
    int est_bot = digitalRead(BOTON1PIN);
    digitalWrite(LED1PIN, est_bot);
}
```

Ejercicio 7. Entrada y Salida Digital. Efecto Rebote. Depuración

- **Circuito LED**: conecte el pin positivo de un LED rojo al pin digital número 3 de Arduino, y el pin negativo del LED a masa (*Gnd*) a través de una resistencia de 470Ω. Repita la misma conexión para un LED verde conectado al pin digital número 4 de Arduino.
- **Funcionamiento LED**: cuando se active 5V (`HIGH`) en el pin digital de salida correspondiente de Arduino, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia. En caso de que se active 0V (`LOW`) en el pin digital de salida, entonces no habrá flujo de corriente y no se encenderá el LED.
- **Circuito Botón-PullDown**: conecte un pin del botón a 5V, y el otro pin del botón a masa (*Gnd*) a través de una resistencia *Pull-Down* de 10KΩ. Además, conecte el pin del botón conectado a la resistencia *Pull-Down* al pin digital número 2 de Arduino.
- **Funcionamiento Botón-PullDown**: mientras el botón no está pulsado, el pin digital de entrada de Arduino está conectado a masa (*Gnd*) a través de una resistencia de 10KΩ, por lo que proporciona una señal de entrada de 0V (`LOW`). Sin embargo, cuando el botón está pulsado, el pin digital de entrada de Arduino se conecta directamente a 5V, por lo que proporciona una señal de entrada de 5V (`HIGH`).
- **Sketch**: diseñe un programa que encienda el LED rojo conectado al pin digital de salida número 3 si el botón conectado al pin digital de entrada número 2 está pulsado, y lo apague en caso contrario. Además, cuando se pulsa el botón, también se conmuta el estado (de encendido a apagado y viceversa) del LED verde conectado al pin digital de salida número 4, el cual inicialmente está apagado.
- Se debe comprobar el correcto funcionamiento del sistema, especialmente del LED verde, el cual deberá fallar de forma aleatoria debido al efecto *REBOTE* (*bouncing*) que se produce en todos los dispositivos mecánicos que abren o cierran circuitos, produciendo múltiples *pulsos* espúreos de corta duración.
- **Depure** el programa, para ello, lleve una cuenta del número de flancos de subida y envíelos a través del puerto *USB* para que sean mostrados en el terminal de control del ordenador. Utilice para ello la biblioteca *Serial*.

- Nótese que no es posible utilizar los pines digitales 0 y 1 si se utiliza el puerto *USB*.
- *Ayuda:*
  - `Serial.begin(9600); if(Serial){}; Serial.println(data); Serial.flush();`



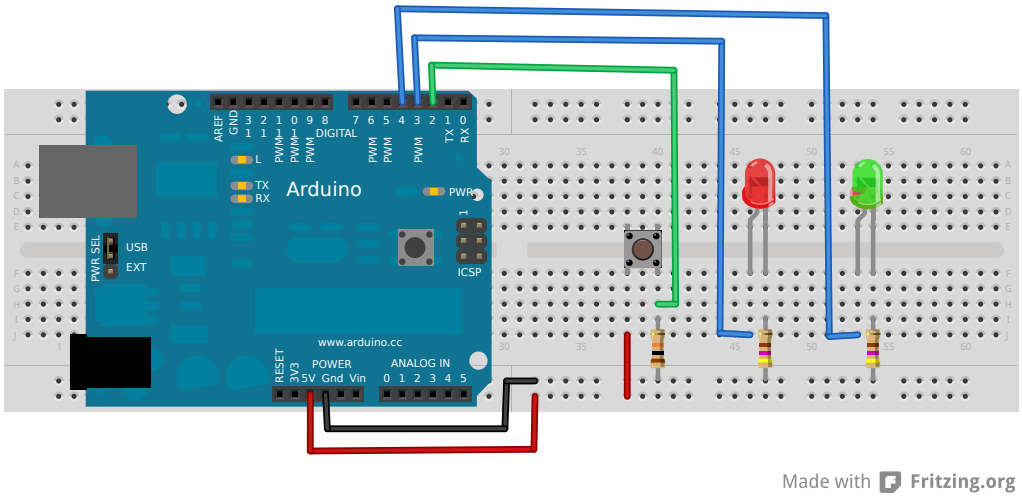
Solución

```
/*** Entrada y Salida Digital. Efecto Rebote. Depuracion ***/
const byte BOTON1PIN = 2;
const byte LED1PIN = 3;
const byte LED2PIN = 4;
int est_bot;
int est_ledv = LOW;
int cnt_flancos = 0;
void setup()
{
    Serial.begin(9600);
    pinMode(BOTON1PIN, INPUT);
    pinMode(LED1PIN, OUTPUT);
    pinMode(LED2PIN, OUTPUT);
    est_bot = digitalRead(BOTON1PIN);
}
void loop()
{
    int est = digitalRead(BOTON1PIN);
    if ((est == HIGH)&&(est_bot == LOW)) {
        est_ledv = (est_ledv == LOW ? HIGH : LOW);
        ++cnt_flancos;
        if (Serial) {
            Serial.print("Cuenta flancos: ");
            Serial.println(cnt_flancos);
        }
    }
    est_bot = est;
    digitalWrite(LED1PIN, est_bot);
    digitalWrite(LED2PIN, est_ledv);
}
```

Ejercicio 8. Entrada y Salida Digital. Anular el Efecto Rebote SW

- **Circuito LED**: conecte el pin positivo de un LED rojo al pin digital número 3 de Arduino, y el pin negativo del LED a masa (*Gnd*) a través de una resistencia de 470Ω. Repita la misma conexión para un LED verde conectado al pin digital número 4 de Arduino.
- **Funcionamiento LED**: cuando se active 5V (`HIGH`) en el pin digital de salida correspondiente de Arduino, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia. En caso de que se active 0V (`LOW`) en el pin digital de salida, entonces no habrá flujo de corriente y no se encenderá el LED.
- **Circuito Botón-PullDown**: conecte un pin del botón a 5V, y el otro pin del botón a masa (*Gnd*) a través de una resistencia *Pull-Down* de 10KΩ. Además, conecte el pin del botón conectado a la resistencia *Pull-Down* al pin digital número 2 de Arduino.
- **Funcionamiento Botón-PullDown**: mientras el botón no está pulsado, el pin digital de entrada de Arduino está conectado a masa (*Gnd*) a través de una resistencia de 10KΩ, por lo que proporciona una señal de entrada de 0V (`LOW`). Sin embargo, cuando el botón está pulsado, el pin digital de entrada de Arduino se conecta directamente a 5V, por lo que proporciona una señal de entrada de 5V (`HIGH`).
- **Sketch**: diseñe un programa que encienda el LED rojo conectado al pin digital de salida número 3 si el botón conectado al pin digital de entrada número 2 está pulsado, y lo apague en caso contrario. Además, cuando se pulsa el botón, también se conmuta el estado (de encendido a apagado y viceversa) del LED verde conectado al pin digital de salida número 4, el cual inicialmente está apagado.
- **Anule** el efecto rebote del botón *por software*, para ello, sólo serán considerados aquellos cambios de estado (flancos) del botón que ocurran después de un tiempo determinado (20 ms) desde el último cambio de estado válido.

- **Depure** el programa, para ello, lleve una cuenta del número de flancos de subida controlados y envíelos a través del puerto *USB* para que sean mostrados en el terminal.
- *Ayuda:*
  - `ms = millis();`



Solución

```
/*** Entrada y Salida Digital. Anular el Efecto Rebote SW ***/
const unsigned long UMBRAL_DEBOUNCE_MS = 20;
const byte BOTON1PIN = 2;
const byte LED1PIN = 3;
const byte LED2PIN = 4;
int est_ledv = LOW;
int est_bot;
unsigned long boton_cambio_ms = 0;
int cnt_flancos = 0;
void setup()
{
    Serial.begin(9600);
    pinMode(BOTON1PIN, INPUT);
    pinMode(LED1PIN, OUTPUT);
    pinMode(LED2PIN, OUTPUT);
    est_bot = digitalRead(BOTON1PIN);
}
void loop()
{
    unsigned long curr_ms = millis();
    int est = digitalRead(BOTON1PIN);
    if ((est != est_bot)&&(curr_ms - boton_cambio_ms >= UMBRAL_DEBOUNCE_MS)) {
        est_bot = est;
        boton_cambio_ms = curr_ms;
        if (est_bot == HIGH) {
            est_ledv = (est_ledv == LOW ? HIGH : LOW);
            //------------
            ++cnt_flancos;
            if (Serial) {
                Serial.print("Cuenta flancos: ");
                Serial.println(cnt_flancos);
            }
        }
    }
    digitalWrite(LED1PIN, est_bot);
    digitalWrite(LED2PIN, est_ledv);
}
```

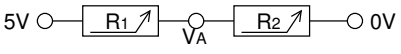
Solución Alternativa

```
/*** Entrada y Salida Digital. Anular el Efecto Rebote SW ***/
const unsigned long UMBRAL_DEBOUNCE_MS = 20;
const byte BOTON1PIN = 2;
const byte LED1PIN = 3;
const byte LED2PIN = 4;
//---------------------------------------
struct Boton {
    unsigned long cambio_ms;
    byte pin;
    byte estado;
};
void setup_boton(struct Boton& boton, byte pin)
{
    boton.cambio_ms = 0;
    boton.pin = pin;
    boton.estado = LOW;
    pinMode(boton.pin, INPUT);
}
void loop_boton(struct Boton& boton, unsigned long curr_ms)
{
    int est = digitalRead(boton.pin);
    if ((est != boton.estado)&&(curr_ms - boton.cambio_ms >= UMBRAL_DEBOUNCE_MS)) {
        boton.cambio_ms = curr_ms;
        boton.estado = est;
    }
}
//---------------------------------------
struct LEDB {
    byte pin;
    byte estado;
};
void setup_ledb(struct LEDB& led, byte pin)
{
    led.pin = pin;
    led.estado = LOW;
    pinMode(led.pin, OUTPUT);
}
void loop_ledb(struct LEDB& led, unsigned long curr_ms, const struct Boton& boton)
{
    if (curr_ms == boton.cambio_ms) {
        led.estado = boton.estado;
        digitalWrite(led.pin, led.estado);
    }
}
//---------------------------------------
struct LEDC {
    byte pin;
    byte estado;
};
void setup_ledc(struct LEDC& led, byte pin)
{
    led.pin = pin;
    led.estado = LOW;
    pinMode(led.pin, OUTPUT);
}
void loop_ledc(struct LEDC& led, unsigned long curr_ms, const struct Boton& boton)
{
    if ((curr_ms == boton.cambio_ms)&&(boton.estado == HIGH)) {
        led.estado = (led.estado == LOW ? HIGH : LOW);
        digitalWrite(led.pin, led.estado);
    }
}
//---------------------------------------
struct Monitor {
    int cnt_flancos;
};
void setup_monitor(struct Monitor& mtr)
{
    Serial.begin(9600);
    mtr.cnt_flancos = 0;
}
void loop_monitor(struct Monitor& mtr, unsigned long curr_ms, const struct Boton& boton)
{
```

```
    if (Serial && (curr_ms == boton.cambio_ms)&&(boton.estado == HIGH)) {
        ++mtr.cnt_flancos;
        Serial.print("Cuenta flancos: ");
        Serial.println(mtr.cnt_flancos);
    }
}
//-------------------------------------
struct Boton boton;
struct LEDB ledb;
struct LEDC ledc;
struct Monitor monitor;

void setup()
{
    setup_boton(boton, BOTON1PIN);
    setup_ledb(ledb, LED1PIN);
    setup_ledc(ledc, LED2PIN);
    setup_monitor(monitor);
}
void loop()
{
    unsigned long curr_ms = millis();
    loop_boton(boton, curr_ms);
    loop_ledb(ledb, curr_ms, boton);
    loop_ledc(ledc, curr_ms, boton);
    loop_monitor(monitor, curr_ms, boton);
}
```

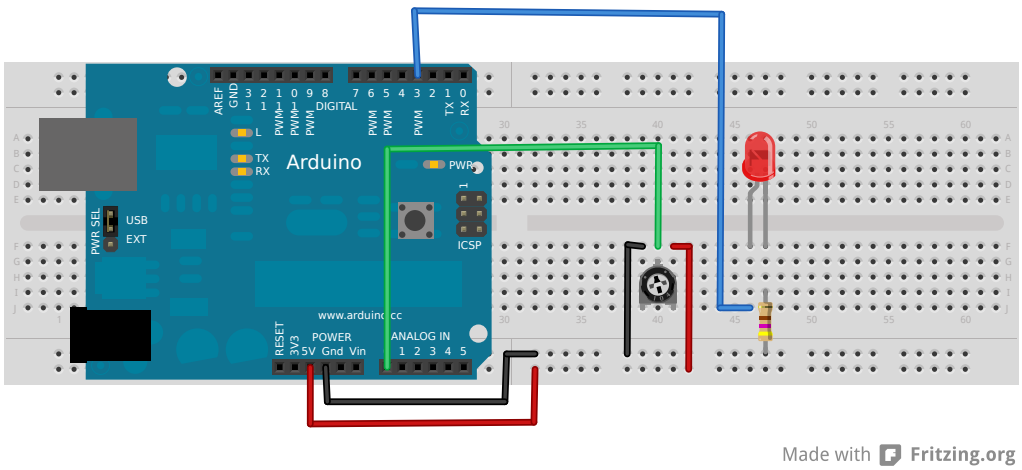
Ejercicio 9. Entrada Analógica (Pot) y Salida PWM

- **Circuito LED-PWM**: conecte el pin positivo de un LED al pin *PWM* número 3 de Arduino, y el pin negativo del LED a masa (*Gnd*) a través de una resistencia de 470Ω.
- **Funcionamiento LED-PWM**: cuando se activa una señal *PWM* de valor entre 0 y 255, se activa un voltaje entre 0V y 5V equivalente en el pin *PWM* de salida correspondiente, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por el voltaje activado en el pin *PWM* y por la resistencia, de tal forma que a mayor voltaje, mayor intensidad de corriente y mayor brillo en el LED. Puede ser conveniente calibrar el LED (calcular el rango de valores útiles).
- **Circuito Potenciómetro**: conecte el pin izquierdo del potenciómetro a masa (*Gnd*), el pin derecho a 5V, y el pin central al pin analógico número 0 de Arduino.
- **Funcionamiento Potenciómetro**: a medida que se gira a la derecha el control del potenciómetro, la resistencia del punto central aumenta respecto a 0V y disminuye respecto a 5V, por lo que el voltaje en el punto central va aumentando desde 0V hasta 5V según gira a la derecha, o disminuye según gira a la izquierda, proporcionando una señal continua al pin analógico de Arduino.



$$V_A = 5V \times \frac{R_2}{R_1 + R_2}$$

- **Sketch**: diseñe un programa que encienda un LED conectado al pin *PWM* de salida número 3 con un brillo (intensidad) proporcional a la posición de giro del control del potenciómetro conectado al pin analógico 0.
- *Ayuda:*
  - `pinMode(pin, OUTPUT);`
  - `analogWrite(pin, pwm)`; permite escribir un valor (entre 0 y 255) en un pin *PWM* de salida, que corresponde a un voltaje *simulado* entre 0V y 5V. Dicho voltaje permanece asociado al pin hasta que se escribe otro valor.
  - `analogRead(pin)`; permite leer un valor (entre 0 y 1023) de un pin analógico de entrada, que depende del voltaje (entre 0V y 5V) de la señal que haya en dicho pin.



Solución

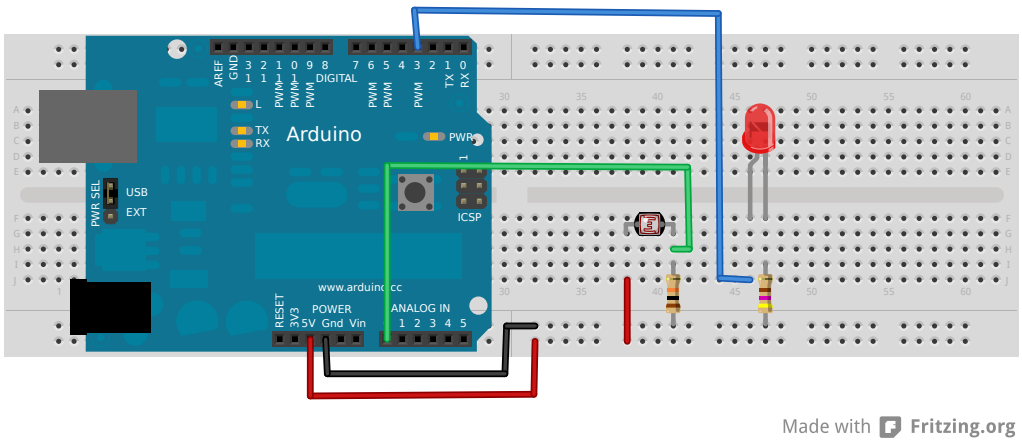
```
/*** Entrada Analogica (Pot) y Salida PWM ***/
const byte POT1PIN = 0;
const byte LED1PIN = 3;
const int MINPOT = 0;          /* calcular valor para el potenciometro minimo */
const int MAXPOT = 1023;       /* calcular valor para el potenciometro maximo */
const byte MINLED_470 = 0;     /* calcular valor a partir del cual el LED comienza a brillar */
const byte MAXLED_470 = 255;   /* calcular valor de maximo brillo del LED */
inline long map2c(long x, long in_min, long in_max, long out_min, long out_max)
{
    x = x < in_min ? in_min : x > in_max ? in_max : x;
    return out_min + (x - in_min) * (out_max - out_min + 1) / (in_max - in_min + 1);
}
void setup()
{
    pinMode(LED1PIN, OUTPUT);
}
void loop()
{
    int val = analogRead(POT1PIN);
    analogWrite(LED1PIN, map2c(val, MINPOT, MAXPOT, MINLED_470, MAXLED_470));
}
```

Ejercicio 10. Entrada Analógica (LDR-PullDown) y Salida PWM

- **Circuito LED-PWM**: conecte el pin positivo de un LED al pin *PWM* número 3 de Arduino, y el pin negativo del LED a masa (*Gnd*) a través de una resistencia de 470Ω.
- **Funcionamiento LED-PWM**: cuando se activa una señal *PWM* de valor entre 0 y 255, se activa un voltaje entre 0V y 5V equivalente en el pin *PWM* de salida correspondiente, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por el voltaje activado en el pin *PWM* y por la resistencia, de tal forma que a mayor voltaje, mayor intensidad de corriente y mayor brillo en el LED. Puede ser conveniente calibrar el LED (calcular el rango de valores útiles).
- **Circuito LDR-PullDown**: conecte un pin del sensor LDR a 5V, y el otro pin del sensor LDR a masa (*Gnd*) a través de una resistencia *Pull-Down* de 10KΩ. Además, conecte el pin del sensor LDR conectado a la resistencia *Pull-Down* al pin analógico número 0 de Arduino.
- **Funcionamiento LDR-PullDown**: a medida que aumenta la luz disminuye la resistencia del sensor LDR por lo que el voltaje en el punto conectado al pin analógico de Arduino va aumentando desde 0V hasta 5V, y viceversa, proporcionando una señal continua al pin analógico de Arduino.

$$V_A = 5V \times \frac{10000}{R_{LDR}+10000}$$

- El rango real del voltaje proporcionado depende de las condiciones ambientales y del dispositivo, por lo que es conveniente **calibrar** el sensor para las condiciones actuales.
- **Sketch**: diseñe un programa que encienda un LED conectado al pin *PWM* de salida número 3 con un brillo (intensidad) inversamente proporcional a la luminosidad del ambiente (cuanto menos luminosidad, mayor intensidad del LED), medida por un sensor LDR conectado al pin analógico 0.
- *Ayuda:*
  - `pinMode(pin, OUTPUT)`; `analogWrite(pin, pwm)`; `analogRead(pin)`;
  - `http://www.advancedphotonix.com/ap_products/pdfs/pdv-p9203.pdf`
  - `http://en.wikipedia.org/wiki/Photoresistor`



Solución

```
/*** Entrada Analogica (LDR-PullDown) y Salida PWM ***/
const byte LDR1PIN = 0;
const byte LED1PIN = 3;
const int MINLDR = 0;     /* calcular valor para el LDR minimo */
const int MAXLDR = 1023; /* calcular valor para el LDR maximo */
const byte MINLED_470 = 0;     /* calcular valor a partir del cual el LED comienza a brillar */
const byte MAXLED_470 = 255;  /* calcular valor de maximo brillo del LED */
inline long map2c(long x, long in_min, long in_max, long out_min, long out_max)
{
    x = x < in_min ? in_min : x > in_max ? in_max : x;
    return out_min + (x - in_min) * (out_max - out_min + 1) / (in_max - in_min + 1);
}
void setup()
{
    pinMode(LED1PIN, OUTPUT);
}
void loop()
{
    int val = analogRead(LDR1PIN);
    /* notese la inversion de rango debido a proporcion inversa */
    analogWrite(LED1PIN, map2c(val, MINLDR, MAXLDR, MAXLED_470, MINLED_470));
}
```

Solución Alternativa

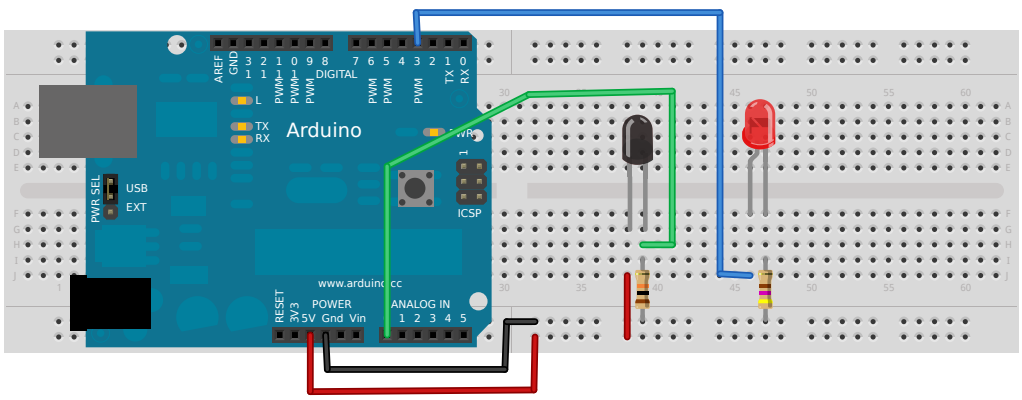
```
/*** Entrada Analogica (LDR-PullDown) y Salida PWM ***/
/*
 * Solucion alternativa con calibracion del sensor
 */
const byte LDR1PIN = 0;
const byte LED1PIN = 3;
const byte MINLED_470 = 0;     /* calcular valor a partir del cual el LED comienza a brillar */
const byte MAXLED_470 = 255;   /* calcular valor de maximo brillo del LED */
const unsigned long CALIBR_MS = 30000;

struct Calibrar {
    int min, max;
};
void calibrar(struct Calibrar& cal, byte pin, byte ledpin, unsigned long t_ms)
{
    if (t_ms == 0) {
        cal.min = 0;
        cal.max = 1023;
    } else {
        byte led = LOW;
        unsigned niter = t_ms / 100;
        cal.max = cal.min = analogRead(pin);
        for (unsigned i = 0; i < niter; ++i) {
            int val = analogRead(pin);
            if (val < cal.min) {
                cal.min = val;
            } else if (val > cal.max) {
                cal.max = val;
            }
            if (ledpin < 14) {
                led = (led == LOW ? HIGH : LOW);
                digitalWrite(ledpin, led);
            }
            delay(100);
        }
    }
}
inline long map2c(long x, long in_min, long in_max, long out_min, long out_max)
{
    x = x < in_min ? in_min : x > in_max ? in_max : x;
    return out_min + (x - in_min) * (out_max - out_min + 1) / (in_max - in_min + 1);
}
struct Calibrar ldr;
void setup()
{
    pinMode(LED1PIN, OUTPUT);
    calibrar(ldr, LDR1PIN, LED1PIN, CALIBR_MS);
}
void loop()
{
    int val = analogRead(LDR1PIN);
    /* notese la inversion de rango debido a proporcion inversa */
    analogWrite(LED1PIN, map2c(val, ldr.min, ldr.max, MAXLED_470, MINLED_470));
}
```

Ejercicio 11. Entrada Analógica (Thermistor-PullDown) y Salida PWM

- **Circuito LED-PWM**: conecte el pin positivo de un LED al pin *PWM* número 3 de Arduino, y el pin negativo del LED a masa (*Gnd*) a través de una resistencia de 470Ω.
- **Funcionamiento LED-PWM**: cuando se activa una señal *PWM* de valor entre 0 y 255, se activa un voltaje entre 0V y 5V equivalente en el pin *PWM* de salida correspondiente, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por el voltaje activado en el pin *PWM* y por la resistencia, de tal forma que a mayor voltaje, mayor intensidad de corriente y mayor brillo en el LED. Puede ser conveniente calibrar el LED (calcular el rango de valores útiles).
- **Circuito Thermistor-PullDown**: conecte un pin del sensor Thermistor a 5V, y el otro pin del sensor Thermistor a masa (*Gnd*) a través de una resistencia *Pull-Down* de 10KΩ. Además, conecte el pin del sensor Thermistor conectado a la resistencia *Pull-Down* al pin analógico número 0 de Arduino.
- **Funcionamiento NTC-Thermistor-PullDown**: a medida que aumenta la temperatura disminuye la resistencia del sensor NTC-Thermistor por lo que el voltaje en el punto conectado al pin analógico de Arduino va aumentando desde 0V hasta 5V, y viceversa, proporcionando una señal continua al pin analógico de Arduino.

$$V_A = 5V \times \frac{10000}{R_{TMP}+10000}$$

- El rango real del voltaje proporcionado depende de las condiciones ambientales y del dispositivo, por lo que es conveniente **calibrar** el sensor para las condiciones actuales.
- **Sketch**: diseñe un programa que encienda un LED conectado al pin *PWM* de salida número 3 con un brillo (intensidad) directamente proporcional a la temperatura del ambiente (cuanto más temperatura, mayor intensidad del LED), medida por un sensor Thermistor conectado al pin analógico 0.
- *Ayuda:*
  - `pinMode(pin, OUTPUT)`; `analogWrite(pin, pwm)`; `analogRead(pin)`;
  - `http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/ntcle100.pdf`
  - `http://en.wikipedia.org/wiki/Thermistor`



Solución

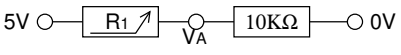
```
/*** Entrada Analogica (Thermistor-PullDown) y Salida PWM ***/
const byte TMP1PIN = 0;
const byte LED1PIN = 3;
const int MINTMP = 0;     /* calcular valor para la temperatura ambiente minima */
const int MAXTMP = 1023; /* calcular valor para la temperatura ambiente maxima */
const byte MINLED_470 = 0;     /* calcular valor a partir del cual el LED comienza a brillar */
const byte MAXLED_470 = 255;  /* calcular valor de maximo brillo del LED */
inline long map2c(long x, long in_min, long in_max, long out_min, long out_max)
{
    x = x < in_min ? in_min : x > in_max ? in_max : x;
    return out_min + (x - in_min) * (out_max - out_min + 1) / (in_max - in_min + 1);
}
void setup()
{
    pinMode(LED1PIN, OUTPUT);
}
void loop()
{
    int val = analogRead(TMP1PIN);
    /* proporcion directa y Pull-Down */
    analogWrite(LED1PIN, map2c(val, MINTMP, MAXTMP, MINLED_470, MAXLED_470));
}
```

Solución Alternativa

```
/*** Entrada Analogica (Thermistor-PullDown) y Salida PWM ***/
/*
 * Solucion alternativa con calibracion del sensor
 */
const byte TMP1PIN = 0;
const byte LED1PIN = 3;
const byte MINLED_470 = 0;     /* calcular valor a partir del cual el LED comienza a brillar */
const byte MAXLED_470 = 255;   /* calcular valor de maximo brillo del LED */
const unsigned long CALIBR_MS = 30000;

struct Calibrar {
    int min, max;
};
void calibrar(struct Calibrar& cal, byte pin, byte ledpin, unsigned long t_ms)
{
    if (t_ms == 0) {
        cal.min = 0;
        cal.max = 1023;
    } else {
        byte led = LOW;
        unsigned niter = t_ms / 100;
        cal.max = cal.min = analogRead(pin);
        for (unsigned i = 0; i < niter; ++i) {
            int val = analogRead(pin);
            if (val < cal.min) {
                cal.min = val;
            } else if (val > cal.max) {
                cal.max = val;
            }
            if (ledpin < 14) {
                led = (led == LOW ? HIGH : LOW);
                digitalWrite(ledpin, led);
            }
            delay(100);
        }
    }
}
inline long map2c(long x, long in_min, long in_max, long out_min, long out_max)
{
    x = x < in_min ? in_min : x > in_max ? in_max : x;
    return out_min + (x - in_min) * (out_max - out_min + 1) / (in_max - in_min + 1);
}
struct Calibrar tmp;
void setup()
{
    pinMode(LED1PIN, OUTPUT);
    calibrar(tmp, TMP1PIN, LED1PIN, CALIBR_MS);
}
void loop()
{
    int val = analogRead(TMP1PIN);
    /* proporcion directa y Pull-Down */
    analogWrite(LED1PIN, map2c(val, tmp.min, tmp.max, MINLED_470, MAXLED_470));
}
```

Ejercicio 12. Sensor de Temperatura (Thermistor) en °Celsius

- **Circuito LED**: conecte el pin positivo de un LED al pin digital número 3 de Arduino, y el pin negativo del LED a masa (*Gnd*) a través de una resistencia de 470Ω.
- **Funcionamiento LED**: cuando se active 5V (`HIGH`) en el pin digital de salida correspondiente de Arduino, el flujo de corriente pasará a través del LED y la resistencia hasta *Gnd*, completando el circuito. El brillo del LED depende de la intensidad de la corriente que pasa por él, y está regulada por la resistencia. En caso de que se active 0V (`LOW`) en el pin digital de salida, entonces no habrá flujo de corriente y no se encenderá el LED.
- **Circuito Thermistor-PullDown**: conecte un pin del sensor Thermistor a 5V, y el otro pin del sensor Thermistor a masa (*Gnd*) a través de una resistencia *Pull-Down* de 10KΩ. Además, conecte el pin del sensor Thermistor conectado a la resistencia *Pull-Down* al pin analógico número 0 de Arduino.
- **Funcionamiento NTC-Thermistor-PullDown**: a medida que aumenta la temperatura disminuye la resistencia del sensor NTC-Thermistor por lo que el voltaje en el punto conectado al pin analógico de Arduino va aumentando desde 0V hasta 5V, y viceversa, proporcionando una señal continua al pin analógico de Arduino.



$$V_A = 5V \times \frac{10000}{R_{TMP}+10000}$$

- **Sketch**: diseñe un programa que, con un periodo de 1000 ms, toma muestras del sensor de temperatura, cuando la temperatura se encuentre dentro de un rango adecuado (36°C–40°C), el LED estará encendido, y apagado en otro caso.

  Cálculo de Temperatura: considerando que $V_A = val \times \frac{5.0}{1023.0}$ y $V_A = 5 \times \frac{10000.0}{R_{TMP}+10000}$, entonces $R_{TMP} = \frac{1023.0 \times 10000.0}{val} - 10000$, donde *val* es el valor del pin analógico leído por Arduino. Por la ecuación del parámetro B, la temperatura en °C: $\tau = \frac{1}{\frac{1}{(T_0+273.15)} + \frac{1}{B}\ln\left(\frac{R_{TMP}}{R_0}\right)} - 273.15$

  Monitorización: en el mismo periodo, se enviará el valor de la temperatura actual (en °C) y el estado del LED a través del puerto *USB* para que sea mostrado en el terminal de control.

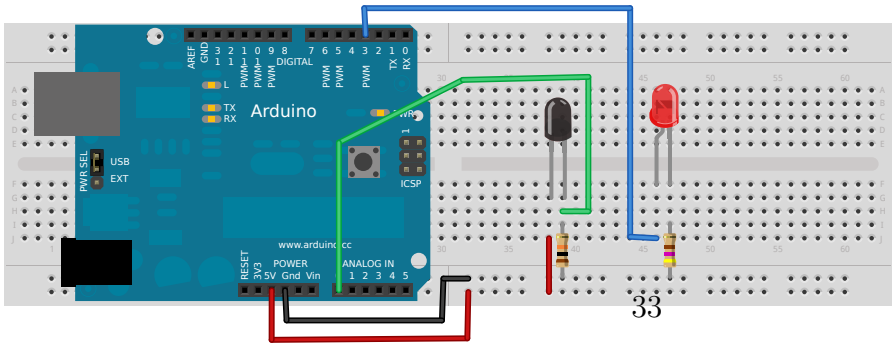
  Se deberá enviar al puerto *USB* al inicio del programa las siguientes líneas de control:

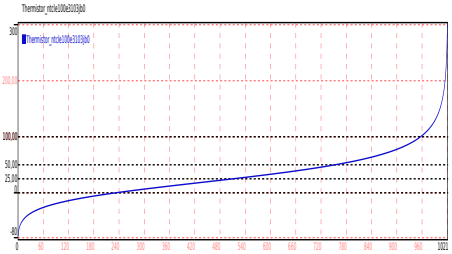
```
#> Arduino Sensor de Temperatura
#$ -y20:45 -l36 -l40 -tTemperatura -tLED
```

- *Ayuda:*
  - Se definen los siguientes parámetros para los siguientes thermistores:

Thermistor	T_0	R_0	B
NTCLE100E3103JB0	25.0	10000.0	3977.0

  - `http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/ntcle100.pdf`
  - `http://en.wikipedia.org/wiki/Thermistor`





Solución

```
/*** Sensor de Temperatura (Thermistor) en grados Celsius ***/
#include <math.h>
/*
 * Thermistor-PullDown: 5V o--THERMISTOR--o--10KOhm--o 0V
 * val valor analogico correspondiente al voltaje Va [0..1023]
 * T0 temperatura referencia en grados celsius
 * R0 resistencia a la temperatura de referencia
 * B coeficiente
 */
inline float val2tmp(int val, float T0, float R0, float B) {
    float t, r;
    r = ((1023.0 * 10e3) / float(val)) - 10e3;
    T0 += 273.15;        /* celsius . kelvin */
    t = 1.0 / ((1.0 / T0) + (log(r / R0) / B));
    return t - 273.15;   /* kelvin . celsius */
}
/*-------------------------------------*/
const float THERMISTOR_T0 = 25.0;
const float THERMISTOR_R0 = 10000.0;
const float THERMISTOR_B = 3977.0;
/*-------------------------------------*/
const unsigned long PERIODO_TMP_MS = 1000;
const byte TMP1PIN = 0;
const byte LED1PIN = 3;
const float TMP_MIN = 36.0;
const float TMP_MAX = 40.0;
unsigned long tmp_last_ms = -PERIODO_TMP_MS;
void setup()
{
    Serial.begin(9600);
    if (Serial) {
      Serial.println("#> Arduino Sensor de Temperatura");
      Serial.println("#$ -y20:45 -l36 -l40 -tTemperatura -tLED");
    }
    pinMode(LED1PIN, OUTPUT);
}
void loop()
{
    unsigned long curr_ms = millis();
    if (curr_ms - tmp_last_ms >= PERIODO_TMP_MS) {
        tmp_last_ms += PERIODO_TMP_MS;
        int val = analogRead(TMP1PIN);
        float tmp = val2tmp(val, THERMISTOR_T0, THERMISTOR_R0, THERMISTOR_B);
        byte led = (TMP_MIN <= tmp && tmp <= TMP_MAX) ? HIGH : LOW;
        digitalWrite(LED1PIN, led);
        if (Serial) {
            Serial.print(tmp);
            Serial.print(" ");
            Serial.println(led == LOW ? 0 : 1);
        }
    }
}
```

Solución Alternativa

```
/*** Sensor de Temperatura (Thermistor) en grados Celsius ***/
#include <math.h>
/*
 * Thermistor-PullDown: 5V o--THERMISTOR--o--10KOhm--o 0V
 * val valor analogico correspondiente al voltaje Va [0..1023]
 * T0 temperatura referencia en grados celsius
 * R0 resistencia a la temperatura de referencia
 * B coeficiente
 */
inline float val2tmp(int val, float T0, float R0, float B) {
    float t, r;
    r = ((1023.0 * 10e3) / float(val)) - 10e3;
    T0 += 273.15;       /* celsius . kelvin */
    t = 1.0 / ((1.0 / T0) + (log(r / R0) / B));
    return t - 273.15;  /* kelvin . celsius */
}
/*-------------------------------------*/
const float THERMISTOR_T0 = 25.0;
const float THERMISTOR_R0 = 10000.0;
const float THERMISTOR_B = 3977.0;
/*-------------------------------------*/
const unsigned long PERIODO_TMP_MS = 1000;
const byte TMP1PIN = 0;
const byte LED1PIN = 3;
const float TMP_MIN = 36.0;
const float TMP_MAX = 40.0;
//-------------------------------------
struct Tmp {
    unsigned long last_ms;
    float val;
    byte pin;
};
void setup_tmp(struct Tmp& tmp, byte pin)
{
    tmp.last_ms = -PERIODO_TMP_MS;
    tmp.pin = pin;
    tmp.val = 0;
}
void loop_tmp(struct Tmp& tmp, unsigned long curr_ms)
{
    if (curr_ms - tmp.last_ms >= PERIODO_TMP_MS) {
        tmp.last_ms += PERIODO_TMP_MS;
        int val = analogRead(tmp.pin);
        tmp.val = val2tmp(val, THERMISTOR_T0, THERMISTOR_R0, THERMISTOR_B);
    }
}
//-------------------------------------
struct LED {
    unsigned long last_ms;
    byte pin;
    byte estado;
};
void setup_led(struct LED& led, byte pin)
{
    led.last_ms = -PERIODO_TMP_MS;
    led.pin = pin;
    led.estado = LOW;
    pinMode(led.pin, OUTPUT);
}
void loop_led(struct LED& led, unsigned long curr_ms, const struct Tmp& tmp)
{
    if (curr_ms - led.last_ms >= PERIODO_TMP_MS) {
        led.last_ms += PERIODO_TMP_MS;
        led.estado = (TMP_MIN <= tmp.val && tmp.val <= TMP_MAX) ? HIGH : LOW;
        digitalWrite(led.pin, led.estado);
    }
}
//-------------------------------------
struct Monitor {
    unsigned long last_ms;
};
void setup_monitor(struct Monitor& mtr)
{
```

```
    mtr.last_ms = -MONITOR_PERIODO;
    Serial.begin(9600);
    if (Serial) {
      Serial.println("#> Arduino Sensor de Temperatura");
      Serial.println("#$ -y20:45 -l36 -l40 -tTemperatura -tLED");
    }
}
void loop_monitor(struct Monitor& mtr, unsigned long curr_ms,
                  const struct Tmp& tmp, const struct LED& led)
{
    if (Serial && (curr_ms - mtr.last_ms >= MONITOR_PERIODO)) {
        mtr.last_ms += MONITOR_PERIODO;
        Serial.print(tmp.val);
        Serial.print(" ");
        Serial.println(led.estado == LOW ? 0 : 1);
    }
}
//---------------------------------------
struct Tmp tmp;
struct LED led;
Struct Monitor monitor;
void setup()
{
    setup_tmp(tmp, TMP1PIN);
    setup_led(led, LED1PIN);
    setup_monitor(monitor);
}
void loop()
{
    unsigned long curr_ms = millis();
    loop_tmp(tmp, curr_ms);
    loop_led(led, curr_ms, tmp);
    loop_monitor(monitor, curr_ms, tmp, led);
}
```

Solución Alternativa

```
/*** Sensor de Temperatura (Thermistor) en grados Celsius ***/
#include <math.h>
/*
 * Thermistor-PullDown: 5V o--THERMISTOR--o--10KOhm--o 0V
 * val valor analogico correspondiente al voltaje Va [0..1023]
 * T0 temperatura referencia en grados celsius
 * R0 resistencia a la temperatura de referencia
 * B coeficiente
 */
inline float val2tmp(int val, float T0, float R0, float B) {
    float t, r;
    r = ((1023.0 * 10e3) / float(val)) - 10e3;
    T0 += 273.15;        /* celsius a kelvin */
    t = 1.0 / ((1.0 / T0) + (log(r / R0) / B));
    return t - 273.15;   /* kelvin a celsius */
}
/*-------------------------------------*/
const float THERMISTOR_T0 = 25.0;
const float THERMISTOR_R0 = 10000.0;
const float THERMISTOR_B = 3977.0;
/*-------------------------------------*/
const unsigned long PERIODO_TMP_MS = 100;
const unsigned long PERIODO_LED_MS = 100;
const unsigned long PERIODO_MTR_MS = 100;
const byte TMP1PIN = 0;
const byte LED1PIN = 3;
const float TMP_MIN = 36.0;
const float TMP_MAX = 40.0;
//-------------------------------------
struct Tmp {
    unsigned long period_ms;
    unsigned long last_ms;
    float val;
    byte pin;
};
void setup_tmp(struct Tmp& tmp, byte pin, unsigned long p)
{
    tmp.period_ms = p;
    tmp.last_ms = -tmp.period_ms;
    tmp.pin = pin;
    tmp.val = 0;
}
void loop_tmp(struct Tmp& tmp, unsigned long curr_ms)
{
    if (curr_ms - tmp.last_ms >= tmp.period_ms) {
        tmp.last_ms += tmp.period_ms;
        int val = analogRead(tmp.pin);
        tmp.val = val2tmp(val, THERMISTOR_T0, THERMISTOR_R0, THERMISTOR_B);
    }
}
//-------------------------------------
struct LED {
    unsigned long period_ms;
    unsigned long last_ms;
    float tmp_min;
    float tmp_max;
    byte pin;
    byte estado;
};
void setup_led(struct LED& led, byte pin, unsigned long p, float n, float x)
{
    led.period_ms = p;
    led.last_ms = -led.period_ms;
    led.tmp_min = n;
    led.tmp_max = x;
    led.pin = pin;
    led.estado = LOW;
    pinMode(led.pin, OUTPUT);
}
void loop_led(struct LED& led, unsigned long curr_ms, const struct Tmp& tmp)
{
    if (curr_ms - led.last_ms >= led.period_ms) {
        led.last_ms += led.period_ms;
```

```
        led.estado = (led.tmp_min <= tmp.val && tmp.val <= led.tmp_max) ? HIGH : LOW;
        digitalWrite(led.pin, led.estado);
    }
}
//-------------------------------------
struct Monitor {
    unsigned long period_ms;
    unsigned long last_ms;
};
void setup_monitor(struct Monitor& mtr, unsigned long p)
{
    mtr.period_ms = p;
    mtr.last_ms = -mtr.period_ms;
    Serial.begin(9600);
    if (Serial) {
      Serial.println("#> Arduino Sensor de Temperatura");
      Serial.println("#$ -y20:45 -l36 -l40 -tTemperatura -tLED");
    }
}
void loop_monitor(struct Monitor& mtr, unsigned long curr_ms,
                  const struct Tmp& tmp, const struct LED& led)
{
    if (Serial && (curr_ms - mtr.last_ms >= mtr.period_ms)) {
        mtr.last_ms += mtr.period_ms;
        Serial.print(tmp.val);
        Serial.print(" ");
        Serial.print(led.estado == LOW ? 0 : 1);
        Serial.println();
    }
}
//-------------------------------------
struct Tmp tmp;
struct LED led;
Struct Monitor monitor;
void setup()
{
    setup_tmp(tmp, TMP1PIN, PERIODO_TMP_MS);
    setup_led(led, LED1PIN, PERIODO_LED_MS, TMP_MIN, TMP_MAX);
    setup_monitor(monitor, PERIODO_MTR_MS);
}
void loop()
{
    unsigned long curr_ms = millis();
    loop_tmp(tmp, curr_ms);
    loop_led(led, curr_ms, tmp);
    loop_monitor(monitor, curr_ms, tmp, led);
}
```

Ejercicio 13. Control de Carga Externa: Transistor NPN y Relé

- **Control**: Arduino puede controlar directamente, a través de sus salidas digitales y *PWM*, dispositivos electrónicos que requieran poca potencia (un máximo de 40mA). Sin embargo, para controlar dispositivos que requieran una mayor potencia, tales como motores, relés, solenoides, etc., se puede utilizar un control indirecto, mediante un *transistor bipolar NPN*, de un circuito externo que proporcione la potencia necesaria (hasta 200mA/1A).

- **Circuito Transistor-NPN**: conecte el emisor del transistor NPN a masa (*Gnd*), la base al pin digital número 2 de Arduino a través de una resistencia de 220Ω y el colector a 5V a través de un *diodo en posición inversa*. Además, opcionalmente conecte la base a masa (*Gnd*) a través de una resistencia de *Pull-Down* de 10KΩ.

- **Funcionamiento Transistor-NPN**: cuando se active 5V (`HIGH`) en el pin digital de salida correspondiente de Arduino, el flujo de corriente pasará a través de la resistencia de 220Ω, y la base del transistor hasta el emisor y a *Gnd*, completando el circuito. Así mismo, la corriente que pasa desde el colector hasta el emisor del transistor es proporcional a la corriente que pasa desde la base al emisor, actuando de esta forma como un interruptor electrónico, cerrando el circuito externo, cuya carga a controlar estará conectada entre el colector del transistor y el diodo conectado a los 5V.

  El diodo en posición inversa es un mecanismo de seguridad, ya que ciertos dispositivos como motores, relés, solenoides, etc. acumulan energía, que en caso de *desconexión brusca* puede provocar choques y descargas eléctricas que dañen los dispositivos, por lo que el diodo en posición inversa introduce un circuito de retorno que permite descargar dicha energía acumulada de forma adecuada.

  La resistencia opcional de *Pull-Down* garantiza un voltaje de 0V en caso de que la conexión de la base quede abierta.

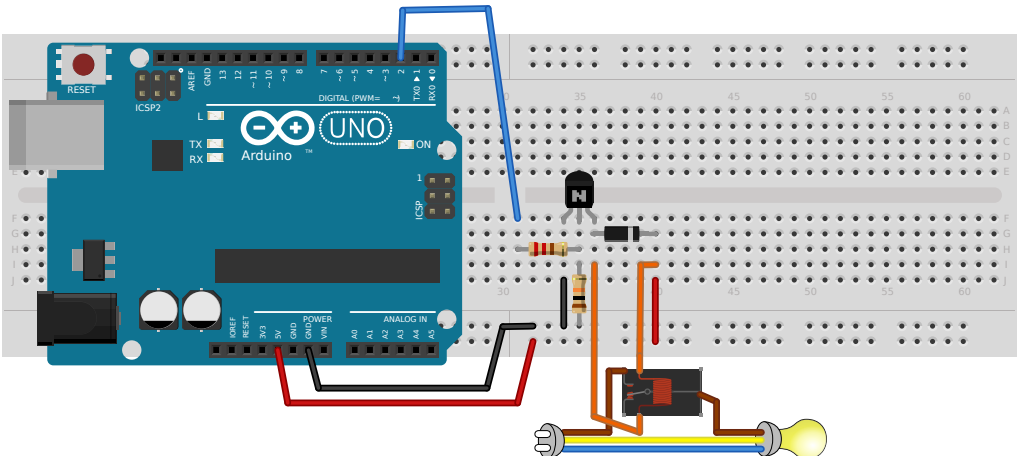
- **Circuito Relé**: conecte el relé como la carga controlada por el transistor, es decir, un pin de control del relé al colector y el otro pin de control al pin del diodo conectado a 5V. Además, conecte los pines del interruptor de carga del relé a la fase del circuito de corriente alterna (230VAC) a controlar (bombilla).

- **Funcionamiento Relé**: el relé es un interruptor de un circuito de alta potencia (*DC* o *AC*) controlado electro-magnéticamente por un circuito de corriente continua de baja potencia independiente. Cuando se activa el paso de corriente por el electro-imán, se induce un efecto magnético que conmuta el interruptor y cierra el circuito de alta potencia controlado. Si la corriente no fluye, no hay efecto magnético y el circuito permanece abierto.

- **Sketch**: diseñe un programa que conmute periodicamente, cada 5 sg, el estado (de desconectado a conectado y viceversa, el cual inicialmente está desconectado) del relé controlado por el transistor NPN conectado al pin digital de salida número 2.

- *Ayuda:*

  - `pinMode(pin, INPUT/OUTPUT)`; `digitalWrite(pin, LOW/HIGH)`; `ms = millis()`;
  - `www.fairchildsemi.com/ds/PN/PN2222A.pdf`
  - `www.finder-reles.net/es/finder-reles-serie-40.pdf`



Transistor **PN2222A**: conexión **igual** al esquema. Transistor **BC547CG**: conexión **inversa** al esquema

Solución

```
/*** Control de Carga Externa: Transistor NPN y Rele ***/
const unsigned long PERIODO_MS = 5000;
const byte RELE1PIN = 2;
int est_rele = LOW;
unsigned long last_ms = 0;
void setup()
{
    pinMode(RELE1PIN, OUTPUT);
}
void loop()
{
    unsigned long curr_ms = millis();
    if (curr_ms - last_ms >= PERIODO_MS) {
        last_ms += PERIODO_MS;
        est_rele = (est_rele == LOW ? HIGH : LOW);
        digitalWrite(RELE1PIN, est_rele);
    }
}
```

Ejercicio 14. Control Automático Básico (Thermistor)

- **Circuito LED-rojo-verde**: conecte un LED rojo al pin digital número 3 de Arduino y un LED verde al pin digital número 5 de Arduino, según el esquema del ejercicio 4.
- **Circuito Thermistor-PullDown**: conecte un sensor de temperatura al pin analógico número 0 de Arduino, según el esquema del ejercicio 12.
- **Circuito Transistor-NPN**: conecte un transistor NPN al pin digital número 2 de Arduino, según el esquema del ejercicio 13.
- **Circuito Relé**: conecte un relé al transistor NPN, según el esquema del ejercicio 13, controlando un calefactor.
- Disponga todos los circuitos y dispositivos especificados anteriormente en un entorno independiente semicerrado de volumen pequeño (aprox. $1/2m^3$).
- **Sketch**: diseñe un programa que controle la temperatura del *entorno independiente* para que se encuentre el mayor tiempo posible dentro de un rango adecuado (36°C–40°C). Para ello ejecuta periodicamente las siguientes tareas *(en orden de prioridad)*:
- **Sensor de Temperatura**: con un periodo de 1000 ms, toma muestras del sensor de temperatura, cuando la temperatura se encuentre dentro del rango adecuado, el LED verde estará encendido, y apagado en otro caso.

  Cálculo de Temperatura: considerando que $V_A = val \times \frac{5.0}{1023.0}$ y $V_A = 5 \times \frac{10000.0}{R_{TMP}+10000}$, entonces $R_{TMP} = \frac{1023.0 \times 10000.0}{val} - 10000$, donde val es el valor del pin analógico leído por Arduino. Por la ecuación del parámetro B, la temperatura en °C: $\tau = \frac{1}{\frac{1}{(T_0+273.15)} + \frac{1}{B}\ln\left(\frac{R_{TMP}}{R_0}\right)} - 273.15$
- **Control**: con un periodo de 5000 ms, si la temperatura actual es menor que la temperatura central del rango (38°C) entonces encenderá el calefactor durante todo el periodo de control. En otro caso, el calefactor estará apagado durante todo el periodo de control:

$$\kappa = \begin{cases} \text{Si } \tau < 38 \text{ entonces:} & \text{Calefactor encendido todo el periodo} \\ \text{En otro caso:} & \text{Calefactor apagado todo el periodo} \end{cases}$$

  Además, cuando el calefactor esté funcionado, el LED rojo estará encendido, y apagado en otro caso.
- **Monitorización**: con un periodo de 1000 ms, se enviará el valor de la temperatura actual y el estado del calefactor *(0 ó 1)* a través del puerto *USB* para que sean mostrados en el terminal de control. Por ejemplo: `25.6` *espacio* `1` *salto-línea*

  En el `setup` del programa, se deberán enviar al puerto *USB* las siguientes líneas de control:

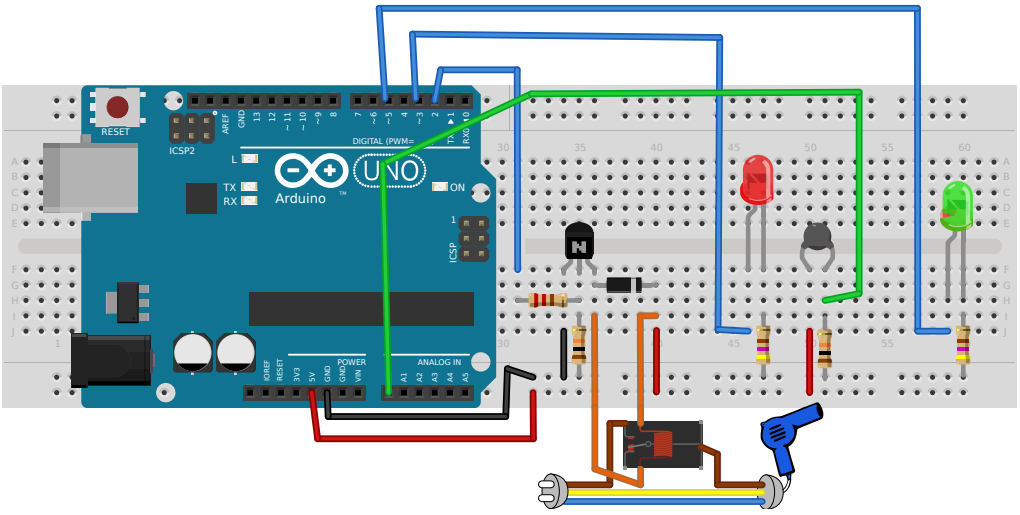
```
#> Arduino Control de Temperatura Basico (5000ms)
#$ -y20:45 -w4 -l36 -l38 -l40 -tTemperatura -tCalefactor
```

Continúa en la siguiente página

- *Ayuda:*
  - Se definen los siguientes parámetros para los siguientes thermistores:

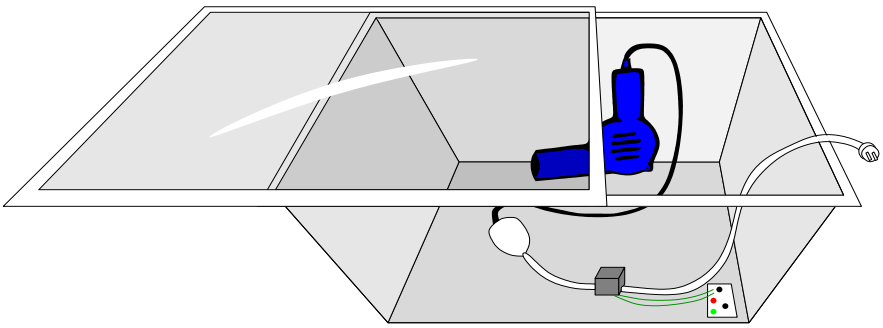
Thermistor	T_0	R_0	B
NTCLE100E3103JB0	25.0	10000.0	3977.0

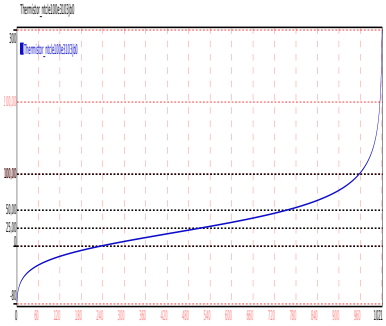
  - `http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/ntcle100.pdf`
  - `http://en.wikipedia.org/wiki/Thermistor`

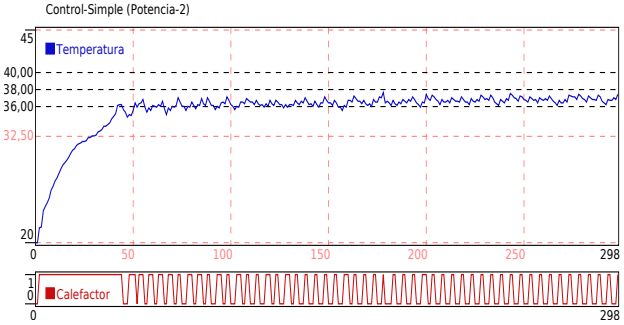


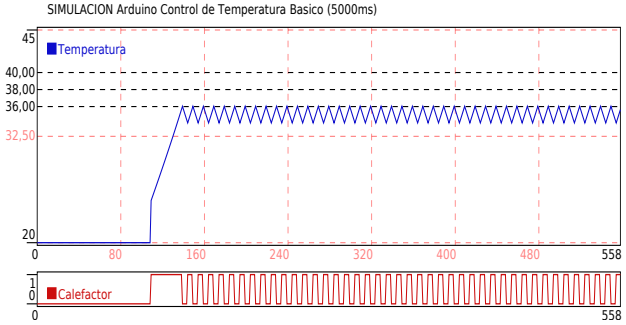
Transistor **PN2222A**: conexión **igual** al esquema. Transistor **BC547CG**: conexión **inversa** al esquema

El sensor de temperatura debe estar lo más **aislado** posible del resto de componentes









Solución

```
/*** Control Automatico Basico (Thermistor) ***/
#include <math.h>
/*
 * Thermistor-PullDown: 5V o--THERMISTOR--o--10KOhm--o 0V
 * val valor analogico correspondiente al voltaje Va [0..1023]
 * T0 temperatura referencia en grados celsius
 * R0 resistencia a la temperatura de referencia
 * B coeficiente
 */
inline float val2tmp(int val, float T0, float R0, float B) {
    float t, r;
    r = ((1023.0 * 10e3) / float(val)) - 10e3;
    T0 += 273.15;       /* celsius . kelvin */
    t = 1.0 / ((1.0 / T0) + (log(r / R0) / B));
    return t - 273.15;  /* kelvin . celsius */
}
/*-------------------------------------*/
const float THERMISTOR_T0 = 25.0;
const float THERMISTOR_R0 = 10000.0;
const float THERMISTOR_B = 3977.0;
/*-------------------------------------*/
const unsigned long PERIODO_MTR_MS = 1000;
const unsigned long PERIODO_TMP_MS = 1000;
const unsigned long PERIODO_RELE_MS = 5000;
const byte RELE1PIN = 2;
const byte LEDRPIN = 3;
const byte LEDVPIN = 5;
const byte TMP1PIN = 0;
const float TMP_REF = 38.0;
const float TMP_MIN = 36.0;
const float TMP_MAX = 40.0;
//-------------------------------------
struct Tmp {
    unsigned long last_ms;
    float val;
};
void setup_tmp(unsigned long curr_ms, struct Tmp& tmp)
{
    pinMode(LEDVPIN, OUTPUT);
    tmp.last_ms = curr_ms - PERIODO_TMP_MS;
    tmp.val = 0;
}
void tarea_tmp(unsigned long curr_ms, struct Tmp& tmp)
{
    if (curr_ms - tmp.last_ms >= PERIODO_TMP_MS) {
        tmp.last_ms += PERIODO_TMP_MS;
        int val = analogRead(TMP1PIN);
        tmp.val = val2tmp(val, THERMISTOR_T0, THERMISTOR_R0, THERMISTOR_B);
        byte est_tmp = (TMP_MIN <= tmp.val && tmp.val <= TMP_MAX) ? HIGH : LOW;
        digitalWrite(LEDVPIN, est_tmp);
    }
}
//-------------------------------------
struct Rele {
    unsigned long last_ms;
    byte estado;
};
void setup_rele(unsigned long curr_ms, struct Rele& rele)
{
    pinMode(RELE1PIN, OUTPUT);
    pinMode(LEDRPIN, OUTPUT);
    rele.last_ms = curr_ms - PERIODO_RELE_MS;
    rele.estado = LOW;
}
void tarea_rele(unsigned long curr_ms, float tmp_val, struct Rele& rele)
{
    if (curr_ms - rele.last_ms >= PERIODO_RELE_MS) {
        rele.last_ms += PERIODO_RELE_MS;
        rele.estado = (tmp_val < TMP_REF) ? HIGH : LOW;
        digitalWrite(RELE1PIN, rele.estado);
        digitalWrite(LEDRPIN, rele.estado);
    }
}
```

```
//-------------------------------------
struct Monitor {
    unsigned long last_ms;
};
void setup_monitor(unsigned long curr_ms, struct Monitor& mtr)
{
    Serial.begin(9600);
    if (Serial) {
      Serial.println("#> Arduino Control de Temperatura Basico (5000ms)");
      Serial.println("#$ -y20:45 -w4 -l36 -l38 -l40 -tTemperatura -tCalefactor");
    }
    mtr.last_ms = curr_ms - PERIODO_MTR_MS;
}
void tarea_monitor(unsigned long curr_ms, struct Monitor& mtr,
                   const struct Tmp& tmp, const struct Rele& rele)
{
    if (curr_ms - mtr.last_ms >= PERIODO_MTR_MS) {
        mtr.last_ms += PERIODO_MTR_MS;
        if (Serial) {
            Serial.print(tmp.val);
            Serial.print(" ");
            Serial.println(rele.estado);
        }
    }
}
//-------------------------------------
struct Monitor mtr;
struct Tmp tmp;
struct Rele rele;
void setup()
{
    unsigned long curr_ms = millis();
    setup_tmp(curr_ms, tmp);
    setup_rele(curr_ms, rele);
    setup_monitor(curr_ms, mtr);
}
void loop() {
    unsigned long curr_ms = millis();
    tarea_tmp(curr_ms, tmp);
    tarea_rele(curr_ms, tmp.val, rele);
    tarea_monitor(curr_ms, mtr, tmp, rele);
}
```

Ejercicio 15. Control Automático (Thermistor) en Ciclo de Trabajo

- **Circuito LED-rojo-verde**: conecte un LED rojo al pin digital número 3 de Arduino y un LED verde al pin digital número 5 de Arduino, según el esquema del ejercicio 5.
- **Circuito Thermistor-PullDown**: conecte un sensor de temperatura al pin analógico número 0 de Arduino, según el esquema del ejercicio 12.
- **Circuito Transistor-NPN**: conecte un transistor NPN al pin digital número 2 de Arduino, según el esquema del ejercicio 13.
- **Circuito Relé**: conecte un relé al transistor NPN, según el esquema del ejercicio 13, controlando un calefactor.
- Disponga todos los circuitos y dispositivos especificados anteriormente en un entorno independiente semicerrado de volumen pequeño (aprox. $1/2m^3$).
- **Sketch**: diseñe un programa que controle la temperatura del *entorno independiente* para que se encuentre el mayor tiempo posible dentro de un rango adecuado (36°C–40°C). Para ello ejecuta periodicamente las siguientes tareas *(en orden de prioridad)*:
- **Sensor de Temperatura**: con un periodo de 1000 ms, toma muestras del sensor de temperatura, cuando la temperatura se encuentre dentro del rango adecuado, el LED verde estará encendido, y apagado en otro caso.

  Cálculo de Temperatura: considerando que $V_A = val \times \frac{5.0}{1023.0}$ y $V_A = 5 \times \frac{10000.0}{R_{TMP}+10000}$, entonces $R_{TMP} = \frac{1023.0 \times 10000.0}{val} - 10000$, donde *val* es el valor del pin analógico leído por Arduino. Por la ecuación del parámetro B, la temperatura en °C: $\tau = \frac{1}{\frac{1}{(T_0+273.15)} + \frac{1}{B}\ln\left(\frac{R_{TMP}}{R_0}\right)} - 273.15$
- **Control**: con un periodo de 5000 ms, calcula el *error* ($\varepsilon \in \mathbb{R}$) como la diferencia entre la temperatura deseada y la actual ($\varepsilon = 38°\text{C} - \tau$), y encenderá o apagará el calefactor según la siguiente ecuación del ciclo de trabajo:

$$\kappa = \begin{cases} \text{Si } \varepsilon \leq 0 \text{ entonces:} & \text{Calefactor apagado todo el periodo } (0\,\%) \\ \text{Si } 0 < \varepsilon < 5 \text{ entonces:} & \text{Calefactor encendido el } (20 \times \varepsilon)\,\% \text{ del periodo } (\equiv 1000 \times \varepsilon \text{ ms}) \\ \text{En otro caso:} & \text{Calefactor encendido todo el periodo } (100\,\%) \end{cases}$$

  Nótese que en el caso de $(0 < \varepsilon < 5)$, se deberá planificar una *subtarea* que apague el calefactor en el tiempo calculado (véase el ejercicio 5 de ciclo de trabajo).

  Además, cuando el calefactor esté funcionado, el LED rojo estará encendido, y apagado en otro caso.
- **Monitorización**: con un periodo de 1000 ms, se enviará el valor de la temperatura actual y el estado del calefactor *(0 ó 1)* a través del puerto *USB* para que sean mostrados en el terminal de control. Por ejemplo: `25.6` *espacio* `1` *salto-línea*

  En el `setup` del programa, se deberán enviar al puerto *USB* las siguientes líneas de control:

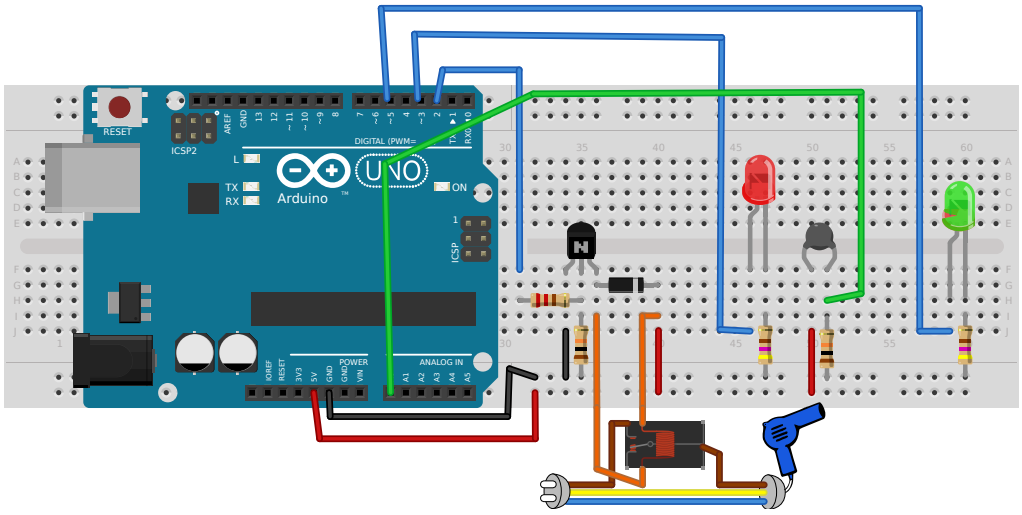
```
#> Arduino Control de Temperatura Ciclo Trabajo (5000ms)
#$ -y20:45 -w4 -l36 -l38 -l40 -tTemperatura -tCalefactor
```

Continúa en la siguiente página

- *Ayuda:*
  - Se definen los siguientes parámetros para los siguientes thermistores:

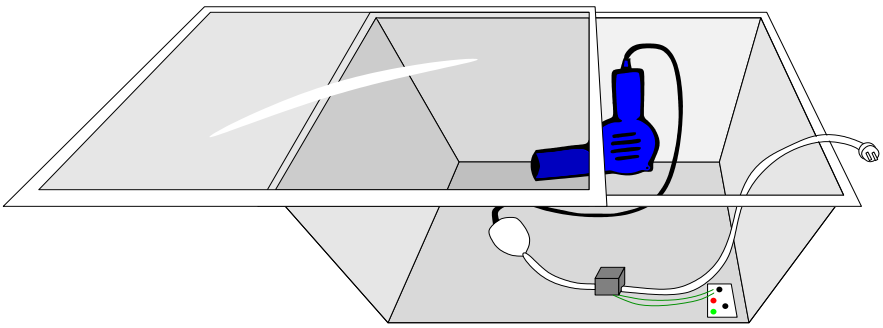
Thermistor	T_0	R_0	B
NTCLE100E3103JB0	25.0	10000.0	3977.0

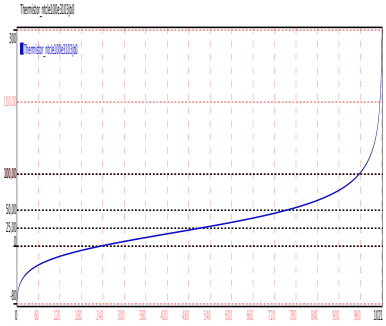
  - `http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/ntcle100.pdf`
  - `http://en.wikipedia.org/wiki/Thermistor`

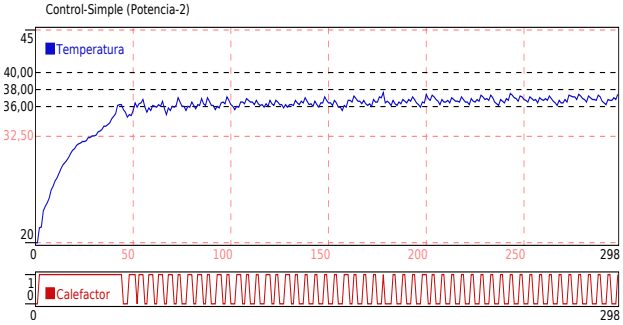


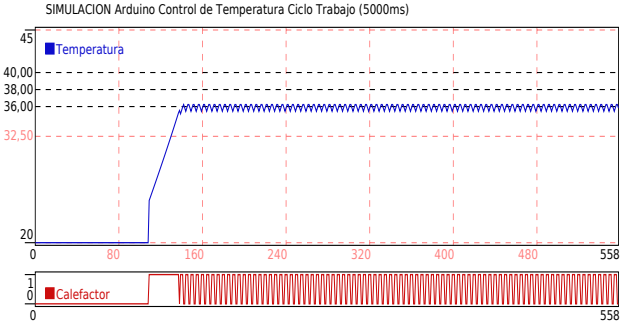
Transistor **PN2222A**: conexión **igual** al esquema. Transistor **BC547CG**: conexión **inversa** al esquema

El sensor de temperatura debe estar lo más **aislado** posible del resto de componentes









Solución

```
/* A realizar por el alumno como practica evaluable */
```

Ejercicio 16. Control Automático (Thermistor) en Ciclo de Trabajo con PID

- **Circuito LED-rojo-verde**: conecte un LED rojo al pin digital número 3 de Arduino y un LED verde al pin digital número 5 de Arduino, según el esquema del ejercicio 5.
- **Circuito Thermistor-PullDown**: conecte un sensor de temperatura al pin analógico número 0 de Arduino, según el esquema del ejercicio 12.
- **Circuito Transistor-NPN**: conecte un transistor NPN al pin digital número 2 de Arduino, según el esquema del ejercicio 13.
- **Circuito Relé**: conecte un relé al transistor NPN, según el esquema del ejercicio 13, controlando un calefactor.
- Disponga todos los circuitos y dispositivos especificados anteriormente en un entorno independiente semicerrado de volumen pequeño (aprox. $1/2m^3$).
- **Sketch**: diseñe un programa que controle la temperatura del *entorno independiente* para que se encuentre el mayor tiempo posible dentro de un rango adecuado (36°C–40°C). Para ello ejecuta periodicamente las siguientes tareas *(en orden de prioridad)*:
- **Sensor de Temperatura**: con un periodo de 1000 ms, toma muestras del sensor de temperatura, cuando la temperatura se encuentre dentro del rango adecuado, el LED verde estará encendido, y apagado en otro caso.

  Cálculo de Temperatura: considerando que $V_A = val \times \frac{5.0}{1023.0}$ y $V_A = 5 \times \frac{10000.0}{R_{TMP}+10000}$, entonces $R_{TMP} = \frac{1023.0 \times 10000.0}{val} - 10000$, donde *val* es el valor del pin analógico leído por Arduino. Por la ecuación del parámetro B, la temperatura en °C: $\tau = \frac{1}{\frac{1}{(T_0+273.15)} + \frac{1}{B}\ln\left(\frac{R_{TMP}}{R_0}\right)} - 273.15$
- **Control PID**: con un periodo de 5000 ms, el *control PID* (con los siguientes coeficientes para un periodo expresado en *segundos*, $K_p = 0.4044$, $K_i = 0.0100$ y $K_d = 3.3718$) proporciona el valor ($\rho \in \mathbb{R}$) a partir de la temperatura deseada (38°C) y la temperatura actual (τ), y encenderá o apagará el calefactor según la siguiente ecuación del ciclo de trabajo:

  $$\kappa = \begin{cases} \text{Si } \rho \leq 0 \text{ entonces:} & \text{Calefactor apagado todo el periodo } (0\,\%) \\ \text{Si } 0 < \rho < 5 \text{ entonces:} & \text{Calefactor encendido el } (20 \times \rho)\,\% \text{ del periodo } (\equiv 1000 \times \rho \text{ ms}) \\ \text{En otro caso:} & \text{Calefactor encendido todo el periodo } (100\,\%) \end{cases}$$

  Nótese que en el caso de $(0 < \rho < 5)$, se deberá planificar una *subtarea* que apague el calefactor en el tiempo calculado (véase el ejercicio 5 de ciclo de trabajo).

  Además, cuando el calefactor esté funcionado, el LED rojo estará encendido, y apagado en otro caso.
- **Monitorización**: con un periodo de 1000 ms, se enviará el valor de la temperatura actual y el estado del calefactor *(0 ó 1)* a través del puerto *USB* para que sean mostrados en el terminal de control. Por ejemplo: `25.6` *espacio* `1` *salto-línea*

  En el `setup` del programa, se deberán enviar al puerto *USB* las siguientes líneas de control:

```
#> Arduino Control de Temperatura PID 0.4044 0.0100 3.3718 (5000ms)
#$ -y20:45 -w4 -l36 -l38 -l40 -tTemperatura -tCalefactor
```

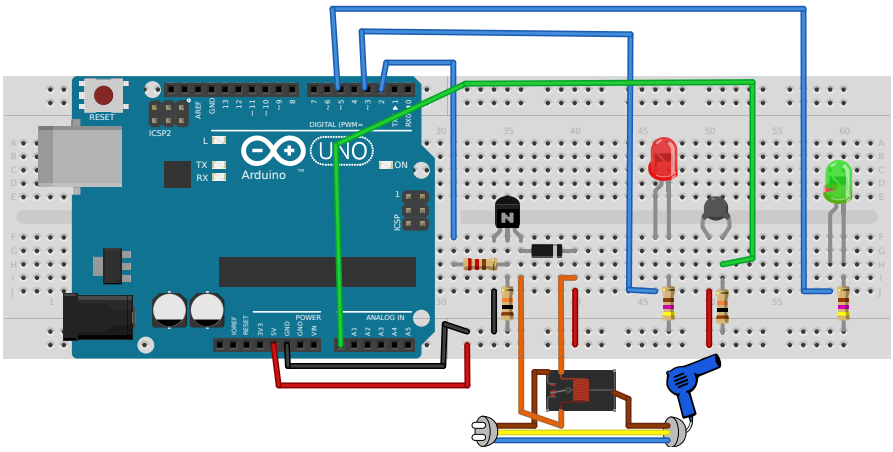

Continúa en la siguiente página

- *Ayuda:*
  - Se definen los siguientes parámetros para los siguientes thermistores:

Thermistor	T_0	R_0	B
NTCLE100E3103JB0	25.0	10000.0	3977.0

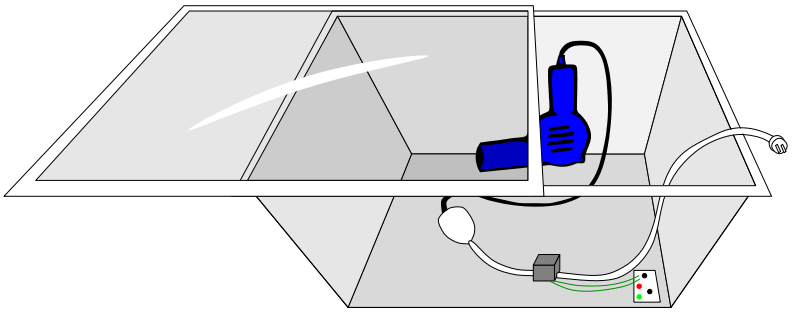
  - `http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/ntcle100.pdf`
  - Véase ejercicio de simulación de control de temperatura con PID en las prácticas de Posix-TR.
  - `http://en.wikipedia.org/wiki/PID_controller`

$\rho(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{d}{dt} e(t)$	$\rho_t = K_p e_t + K_i \sum_{\tau=0}^{t} e_\tau \times T + K_d \frac{e_t - e_{(t-1)}}{T}$



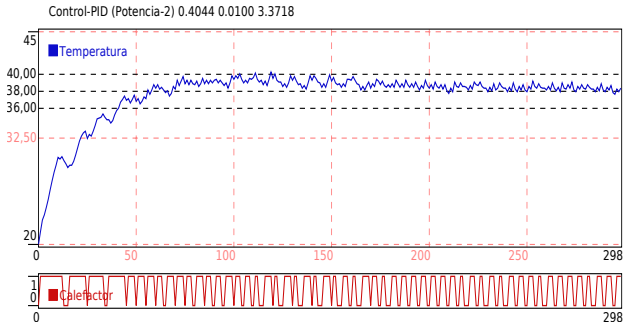
Transistor **PN2222A**: conexión **igual** al esquema. Transistor **BC547CG**: conexión **inversa** al esquema

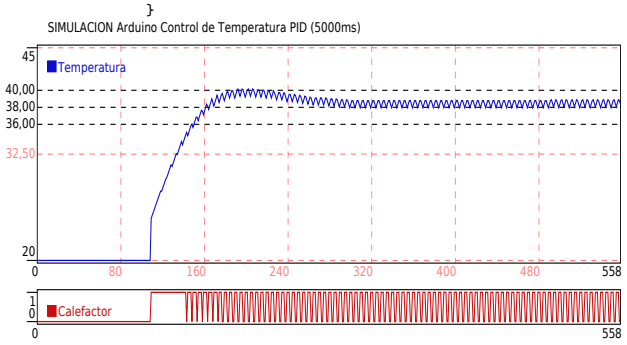
El sensor de temperatura debe estar lo más **aislado** posible del resto de componentes



Pseudocódigo para Control PID

```
void setup() {
  last_ms = -PERIOD_MS;
  prev_error = 0;
  integral = 0;
}
void loop () {
  // ...
  if (curr_ms - last_ms >= PERIOD_MS) {
    last_ms += PERIOD_MS;
    error = target_value - current_value;
    integral = integral + (error * PERIOD_MS * 1e-3);
    derivative = (error - prev_error) / (PERIOD_MS * 1e-3);
    PID = Kp * error + Ki * integral + Kd * derivative;
    prev_error = error;
    // ... control PID ...
  }
}
```





Solución

```
/* A realizar por el alumno como practica evaluable */
```

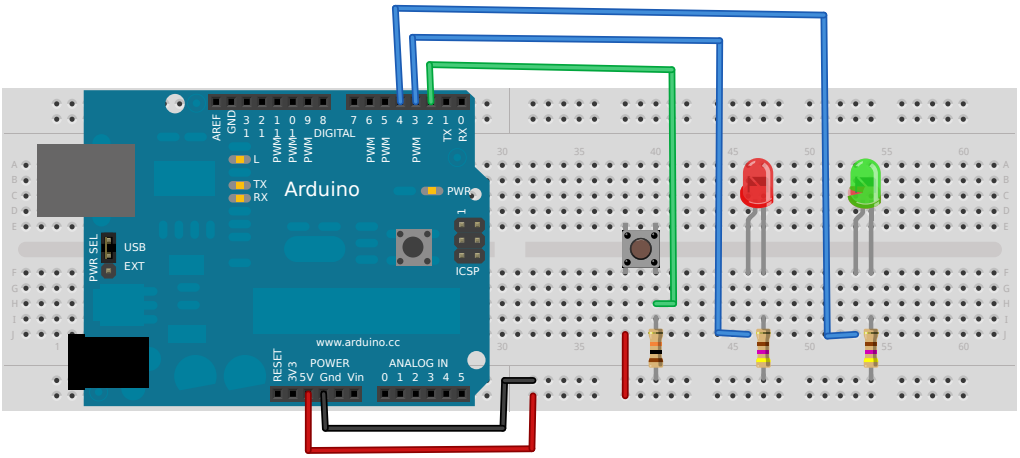
```
/*
 * PID: Ajuste manual de constantes
 *
 * Si el sistema debe mantenerse online, un método de ajuste consiste
 * en establecer primero los valores de I y D a cero. A continuación,
 * incremente P hasta que la salida del lazo oscile. Luego establezca
 * P a aproximadamente la mitad del valor configurado
 * previamente. Después incremente I hasta que el proceso se ajuste en
 * el tiempo requerido (aunque subir mucho I puede causar
 * inestabilidad). Finalmente, incremente D, si se necesita, hasta que
 * el lazo sea lo suficientemente rápido para alcanzar su referencia
 * tras una variación brusca de la carga.
 */
```

Ejercicio 17. Creación de Bibliotecas

- Desarrolle una biblioteca (`DLed`) para controlar el estado de un LED, conectado a un determinado pin de salida digital.
- Desarrolle una biblioteca (`Boton`) para controlar el estado de un botón, conectado a un determinado pin de entrada digital, anulando el efecto rebote con una duración umbral determinada.
- Creación de bibliotecas utilizando el lenguaje de programación *C++*.
  - Fichero de cabecera (`.h`): con *guardas*, definición de constantes, tipos, prototipos y clases. inclusión de `"Arduino.h"` si necesario.
  - Fichero de implementación (`.cpp`): debe incluir su fichero de cabecera y `"Arduino.h"`, así como implementar las funciones y métodos declarados en el fichero de cabecera.
  - Fichero `keywords.txt` donde se especifican las palabras a resaltar por el IDE de Arduino.
  - Hacer un fichero *zip* conteniendo un directorio con el nombre de la biblioteca, que incluya el fichero de cabecera y de implementación de la biblioteca. Por ejemplo para la biblioteca `boton.zip`:

```
alumno@debian:lib$ zip -r boton.zip boton
alumno@debian:lib$ unzip -l boton.zip
Archive: boton.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2013-10-24 11:52   boton/
    12172  2013-10-24 18:14   boton/boton.cpp
     2748  2013-10-24 18:44   boton/boton.h
      632  2013-10-24 18:45   boton/keywords.txt
---------                     -------
    14920                     2 files
```

- Incorporar las nuevas bibliotecas desde el menú del IDE de Arduino `Sketch.Importar Librería.AddLibrary`.
- **Circuito**: considere el mismo circuito y funcionamiento que en el ejercicio 8.
- **Sketch**: considere la misma especificación del programa que en el ejercicio 8.
- Utilice las bibliotecas (`DLed` y `Boton`) creadas anteriormente. `Sketch.Importar Librería.DLed` y `Sketch.Importar Librería.Boton`


Arduino
www.arduino.cc
USB
EXT
PWR SEL
ICSP
POWER
ANALOG IN
DIGITAL
Made with Fritzing.org

Solución

```
/*** Creacion de Bibliotecas en C++ ***/
/*------------------------------------*/
/*- dled/keywords.txt ----------------*/
/*------------------------------------*/
DLed        KEYWORD1
getEstado   KEYWORD2
setEstado   KEYWORD2
conmutar    KEYWORD2
setup       KEYWORD2
loop        KEYWORD2
/*------------------------------------*/
/*- dled/dled.h ----------------------*/
/*------------------------------------*/
#ifndef dled_h_
#define dled_h_
#include "Arduino.h"

class DLed {
    byte pin;
    byte estado;
public:
    DLed(byte p = 0) {
        setup(p);
    }
    byte getEstado() const {
        return estado;
    }
    void setEstado(byte e) {
        estado = e;
    }
    void conmutar() {
        estado = (estado == LOW ? HIGH : LOW);
    }
    void setup(byte p) {
        pin = p;
        estado = LOW;
        pinMode(p, OUTPUT);
        digitalWrite(pin, estado);
    }
    void loop() const {
        digitalWrite(pin, estado);
    }
};
#endif
/*------------------------------------*/
/*- boton/keywords.txt ---------------*/
/*------------------------------------*/
Boton       KEYWORD1
getEstado   KEYWORD2
getCambio   KEYWORD2
setup       KEYWORD2
loop        KEYWORD2
/*------------------------------------*/
/*- boton/boton.h --------------------*/
/*------------------------------------*/
#ifndef boton_h_
#define boton_h_
#include "Arduino.h"

class Boton {
    unsigned long cambio_ms;
    unsigned long debounce_ms;
    byte pin;
    byte estado;
public:
    Boton(byte p = 0, unsigned long dms = 20) {
        setup(p, dms);
    }
    byte getEstado() const {
        return estado;
    }
    unsigned long getCambio() const {
        return cambio_ms;
```

```
    }
    void setup(byte p, unsigned long dms = 20);
    void loop(unsigned long curr_ms);
};
#endif
/*--------------------------------------*/
/*- boton/boton.cpp ------------------*/
/*--------------------------------------*/
#include "boton.h"

void Boton::setup(byte p, unsigned long dms)
{
    cambio_ms = -dms;
    debounce_ms = dms;
    pin = p;
    pinMode(p, INPUT);
    estado = digitalRead(p);
}

void Boton::loop(unsigned long curr_ms)
{
    byte est = digitalRead(pin);
    if ((est != estado)&&(curr_ms - cambio_ms >= debounce_ms)) {
        estado = est;
        cambio_ms = curr_ms;
    }
}
/*--------------------------------------*/
/*- sketch ---------------------------*/
/*--------------------------------------*/
#include <dled.h>
#include <boton.h>

class Tarea1 {
    Boton b1;
    DLed  l1;
    DLed  l2;
public:
    Tarea1(byte pb1, byte pl1, byte pl2) {
        setup(pb1, pl1, pl2);
    }
    void setup(byte pb1, byte pl1, byte pl2);
    void loop(unsigned long curr_ms);
private:
    void control(unsigned long curr_ms);
};
void Tarea1::setup(byte pb1, byte pl1, byte pl2)
{
  b1.setup(pb1);
  l1.setup(pl1);
  l2.setup(pl2);
}
void Tarea1::loop(unsigned long curr_ms)
{
    b1.loop(curr_ms);
    control(curr_ms);
    l1.loop();
    l2.loop();
}
void Tarea1::control(unsigned long curr_ms)
{
    if (b1.getCambio() == curr_ms) {
        l1.setEstado(b1.getEstado());
        if (b1.getEstado() == HIGH) {
            l2.conmutar();
        }
    }
}
/*--------------------------------------*/
const byte BOTON1PIN = 2;
const byte LED1PIN = 3;
const byte LED2PIN = 4;

Tarea1 t1;
```

```
void setup()
{
    t1.setup(BOTON1PIN, LED1PIN, LED2PIN);
}

void loop()
{
    unsigned long curr_ms = millis();
    t1.loop(curr_ms);
}
/*-------------------------------------*/
```

Apéndice I. Operaciones Comunes en Arduino

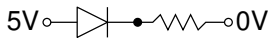
```
boolean char u-char byte short u-short int u-int word long u-long float double

pinMode(pin, mode);         // INPUT/OUTPUT
digitalWrite(pin, value);   // LOW/HIGH
int v = digitalRead(pin);   // LOW/HIGH

analogReference(type);      // DEFAULT, INTERNAL, EXTERNAL
analogWrite(pin, value);    // [0..255] PWM
int v = analogRead(pin);    // [0..1023]

delay(ms);
delayMicroseconds(us);
unsigned long ms = millis();
unsigned long us = micros();

shiftOut(dataPin, clockPin, bitOrder, value);    // MSBFIRST, LSBFIRST
byte b = shiftIn(dataPin, clockPin, bitOrder);   // MSBFIRST, LSBFIRST

unsigned long us = pulseIn(pin, value);
unsigned long us = pulseIn(pin, value, us_timeout);

tone(pin, frequency, duration);
tone(pin, frequency);
noTone(pin);

// int0->pin2, int1->pin3 ; LOW, CHANGE, RISING, FALLING
attachInterrupt(digitalPinToInterrupt(pin), void (*f)(), mode);
detachInterrupt(digitalPinToInterrupt(pin));
noInterrupts();
interrupts();

randomSeed(seed);
int r = random(max);           // [0..max)
int r = random(min, max);      // [min..max]
int v = min(x, y);             // (x < y ? x : y)
int v = max(x, y);             // (x > y ? x : y)
int v = constrain(x, a, b);    // (x < a ? a : (x > b ? b : x))
int y = abs(x);                // (x < 0 ? -x : x)
double p = pow(b, e);
double s = sqrt(x);
double s = sin(rad);
double c = cos(rad);
double t = tan(rad);

/* map2 PRECONDICION: (in_min <= in_max) */
long map2(long x, long in_min, long in_max, long out_min, long out_max)
{
    return out_min + (x - in_min) * (out_max - out_min + 1) / (in_max - in_min + 1);
}
/* map2c PRECONDICION: (in_min <= in_max) */
long map2c(long x, long in_min, long in_max, long out_min, long out_max)
{
    x = x < in_min ? in_min : x > in_max ? in_max : x;
    return out_min + (x - in_min) * (out_max - out_min + 1) / (in_max - in_min + 1);
}
```

Apéndice II. Operaciones Comunes en Arduino

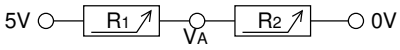
```
byte b = lowByte(x);       // (x & 0x00FF)
byte b = highByte(x);      // ((x >> 8) & 0x00FF)
byte b = bitRead(x, n);    // (!!(x & (1 << n)))
bitWrite(x, n, b);         // (x = (x & ~(1 << n)) | ((b & 1) << n))
bitSet(x, n);              // (x |= (1 << n))
bitClear(x, n);            // (x &= ~(1 << n))
byte b = bit(n);           // (1 << n) equiv a pow(2, n)

if (Serial) {/*...*/}
Serial.begin(rate);              // 9600
Serial.println(data);
Serial.print(data, data_type); // DEC, OCT, BIN, HEX, BYTE , ASCII
Serial.write(val);
Serial.write(str);
Serial.write(buf, len);
Serial.flush();
int nb = Serial.available();
int b = Serial.read();
int b = Serial.peek();
int i = Serial.parseInt();
float f = Serial.parseFloat();
Serial.readBytes(buffer, length);
Serial.readBytesUntil(character, buffer, length);
Serial.setTimeout(ms_timeout);
void serialEvent(){}
Serial.end();

#include <EEPROM.h>
EEPROM.write(address, value);
byte b = EEPROM.read(address);

#include <Wire.h>
Wire.begin();         // Master
Wire.begin(address); // Slave
byte nb = Wire.requestFrom(address, quantity);
byte nb = Wire.requestFrom(address, quantity, stop);
Wire.beginTransmission(address);
byte st = Wire.endTransmission();
byte st = Wire.endTransmission(stop);
byte nb = Wire.write(value);
byte nb = Wire.write(string);
byte nb = Wire.write(data, length);
byte nb = Wire.available();
byte b = Wire.read();
Wire.onReceive(void (*f)(int));
Wire.onRequest(void (*f)());
```

Apéndice III. Cálculo de Valores de Resistencias en Circuitos

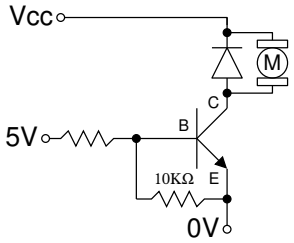
- Cálculo del valor de la resistencia en un LED:
  - Voltaje: 5V. Caída de tensión en el LED: 2V. Máxima intensidad en LED: 20mA.
    - Resistencia mínima: $R_{LED} = V/I = (5-2)/0.020 = 150\Omega$
    - Resistencia máxima: $R_{LED} = V/I = (5-2)/0.003 = 1K\Omega$



- Cálculos en un *divisor de voltaje*:
  - Voltaje en *voltios* a partir del valor analógico leido *val*: $V_A = val \times \frac{5.0}{1023.0}$
  - Voltaje en *voltios* a partir del valor de las resistencias: $V_A = 5V \times \frac{R_2}{R_1+R_2}$
  - Valor de la resistencia R_1 a partir de R_2 y V_A: $R_1 = R_2 \times \frac{5}{V_A} - R_2$
  - Valor de la resistencia R_1 a partir de R_2 y del valor analógico leido *val*: $R_1 = \frac{1023 \times R_2}{val} - R_2$.

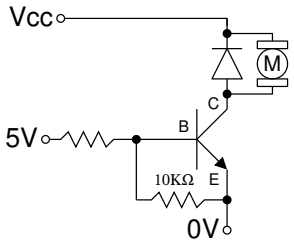


- Cálculo del valor de la resistencia de base en un transistor NPN (PN2222A):
  - Voltaje: 5V. Caída de tensión base-emisor: 1V-2V. Ganancia: 30. Máxima intensidad Colector-Emisor: 1000mA.
    - Intensidad de base de saturación: $I_{B(sat)} = I_C/H_{FE} = 1.000/30 = 0.033A$
    - Resistencia mínima: $R_{B(sat)} = (V_B - V_{BE(sat)})/I_{B(sat)} = (5-1)/0.033 = 121\Omega$

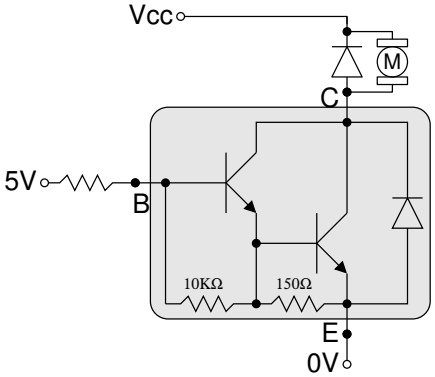


- Cálculo del valor de la intensidad de colector en un transistor NPN (PN2222A) para una resistencia de base dada:
  - Voltaje: 5V. Caída de tensión base-emisor: 1V-2V. Ganancia: 30. Resistencia de base: 220Ω.
    - Intensidad de base: $I_B = (V_B - V_{BE(sat)})/R_B = (5-1)/220 = 0.018A$
    - Intensidad de colector: $I_C = H_{FE} \times I_B = 30 \times 0.018 = 0.540A$
  - Voltaje: 5V. Caída de tensión base-emisor: 1V-2V. Ganancia: 30. Resistencia de base: $1K\Omega$.
    - Intensidad de base: $I_B = (V_B - V_{BE(sat)})/R_B = (5-1)/1000 = 0.004A$
    - Intensidad de colector: $I_C = H_{FE} \times I_B = 30 \times 0.004 = 0.120A$

- Cálculo del valor de la resistencia de base en un transistor NPN (2N3904):
  - Voltaje: 5V. Caída de tensión base-emisor: 1V. Ganancia: 30. Máxima intensidad Colector-Emisor: 200mA.
    - Intensidad de base de saturación: $I_{B(sat)} = I_C/H_{FE} = 0.200/30 = 0.006A$
    - Resistencia mínima: $R_{B(sat)} = (V_B - V_{BE(sat)})/I_{B(sat)} = (5-1)/0.006 = 666\Omega$



- Cálculo del valor de la intensidad de colector en un transistor NPN (2N3904) para una resistencia de base dada:
  - Voltaje: 5V. Caída de tensión base-emisor: 1V. Ganancia: 30. Resistencia de base: 470Ω.
    - Intensidad de base: $I_B = (V_B - V_{BE(sat)})/R_B = (5-1)/470 = 0.008A$
    - Intensidad de colector: $I_C = H_{FE} \times I_B = 30 \times 0.008 = 0.255A$
  - Voltaje: 5V. Caída de tensión base-emisor: 1V. Ganancia: 30. Resistencia de base: $1K\Omega$.
    - Intensidad de base: $I_B = (V_B - V_{BE(sat)})/R_B = (5-1)/1000 = 0.004A$
    - Intensidad de colector: $I_C = H_{FE} \times I_B = 30 \times 0.004 = 0.120A$
- Cálculo del valor de la resistencia de base en un transistor Darlington NPN (BDX33C):
  - Voltaje: 5V. Caída de tensión base-emisor: 2.5V. Ganancia: 750. Máxima intensidad Colector-Emisor: 10A.
    - Intensidad de base de saturación: $I_{B(sat)} = I_C/H_{FE} = 10/750 = 0.013A$
    - Resistencia mínima: $R_{B(sat)} = (V_B - V_{BE(sat)})/I_{B(sat)} = (5-2.5)/0.013 = 187\Omega$



- Cálculo del valor de la intensidad de colector en un transistor Darlington NPN (BDX33C) para una resistencia de base dada:
  - Voltaje: 5V. Caída de tensión base-emisor: 2.5V. Ganancia: 750. Resistencia de base: 470Ω.
    - Intensidad de base: $I_B = (V_B - V_{BE(sat)})/R_B = (5-2.5)/470 = 0.005A$
    - Intensidad de colector: $I_C = H_{FE} \times I_B = 750 \times 0.005 = 4A$
  - Voltaje: 5V. Caída de tensión base-emisor: 2.5V. Ganancia: 750. Resistencia de base: $1K\Omega$.

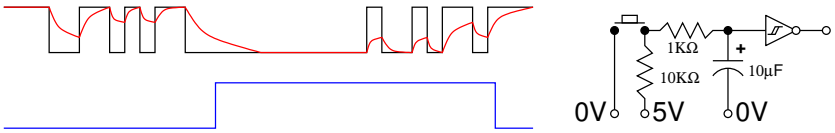
- Intensidad de base: $I_B = (V_B - V_{BE(sat)})/R_B = (5 - 2.5)/1000 = 0.0025A$
- Intensidad de colector: $I_C = H_{FE} \times I_B = 750 \times 0.0025 = 1.875A$

- Cálculo del tiempo de carga y descarga del condensador en un circuito para eliminar por hardware el rebote de un botón (`http://www.ganssle.com/debouncing.htm`):

  - Considerando:
    - V_i: Voltaje inicial del condensador antes de la descarga.
    - V_{cap}: Voltaje en condensador después de t segundos de carga/descarga.
    - V_f: Voltaje final del condensador después de la carga.
    - Inversor Schmitt Trigger (SN74LS14): $V_{T-} = [0.5..1.0]$ $V_{T+} = [1.4..1.9]$
  - Descarga ($V_i = 5\text{V}$, $V_{T-} = [0.5..1.0]$): $V_{cap} = V_i \times \left(e^{\frac{-t}{RC}}\right) \Rightarrow t = -RC \ln\left(\frac{V_{cap}}{V_i}\right)$

    $0.0016\text{sg} = -1K\Omega \times 1\mu\text{F} \times \ln\left(\frac{1\text{V}}{5\text{V}}\right)$
  - Carga ($V_f = 5\text{V}$, $V_{T+} = [1.4..1.9]$): $V_{cap} = V_f \times \left(1 - e^{\frac{-t}{RC}}\right) \Rightarrow t = -RC \ln\left(1 - \frac{V_{cap}}{V_f}\right)$

    $0.0036\text{sg} = -11K\Omega \times 1\mu\text{F} \times \ln\left(1 - \frac{1.4\text{V}}{5\text{V}}\right)$



- Control *PID* (`http://en.wikipedia.org/wiki/PID_controller`):

$\rho(t) = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{d}{dt} e(t)$	$\rho_t = K_p e_t + K_i \sum_{\tau=0}^{t} e_\tau \times T + K_d \frac{e_t - e_{(t-1)}}{T}$

Apéndice III. Códigos de Color de Resistencias