

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 $1K\Omega$ (1/4W)
- 3 Resistencias de $10K\Omega$ (1/4W)
- 3 LEDs Rojos
- 1 LED Verde
- 3 Botón/Pulsador
- 1 Potenciómetro ($10K\Omega$ ó $100K\Omega$)
- 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 $57 \times 39 \times 28$ cm³.
- Cables de conexión de prototipos.

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

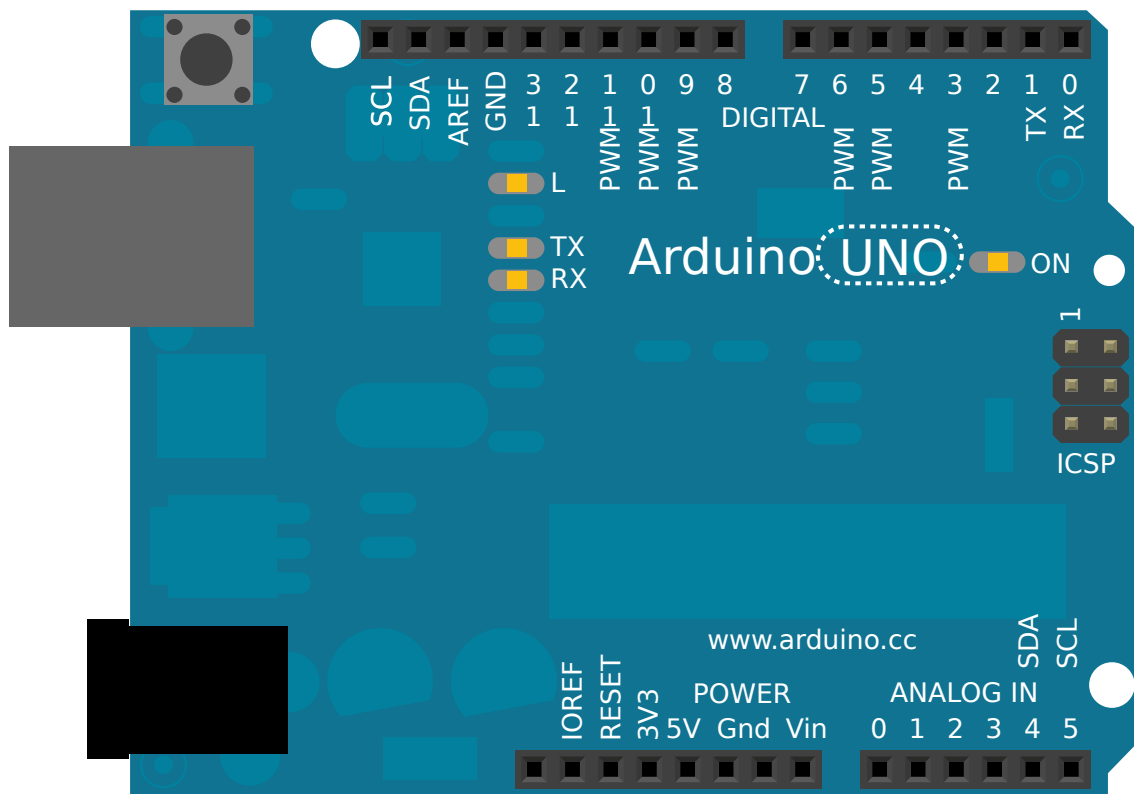
Introducción. Arduino UNO

■ 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:

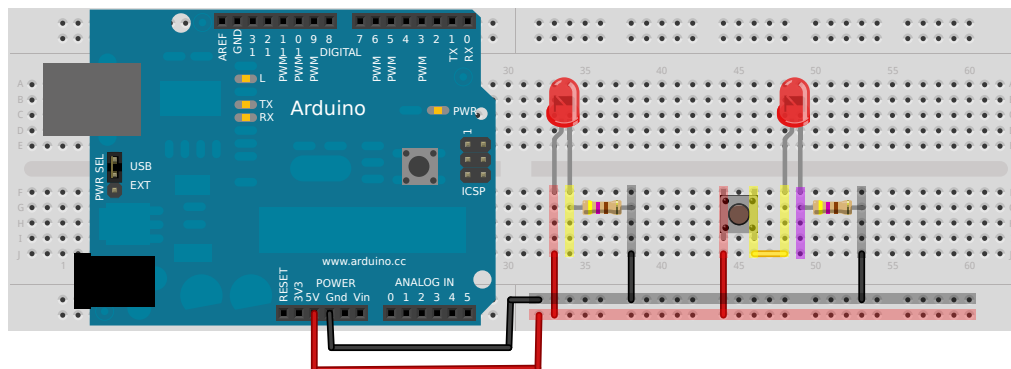
- <http://arduino.cc/en/Main/arduinoBoardUno>
- <http://playground.arduino.cc/Main/ArduinoPinCurrentLimitations>



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 *Wattios*) [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.



Made with  Fritzing.org

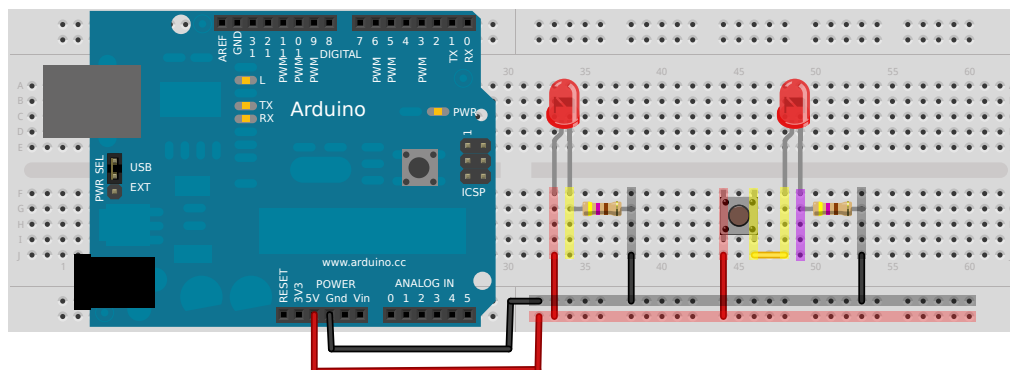
Ejercicio 1. Circuitos Electrónicos. Ley de Ohm (II)

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.



Made with  Fritzing.org

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 .

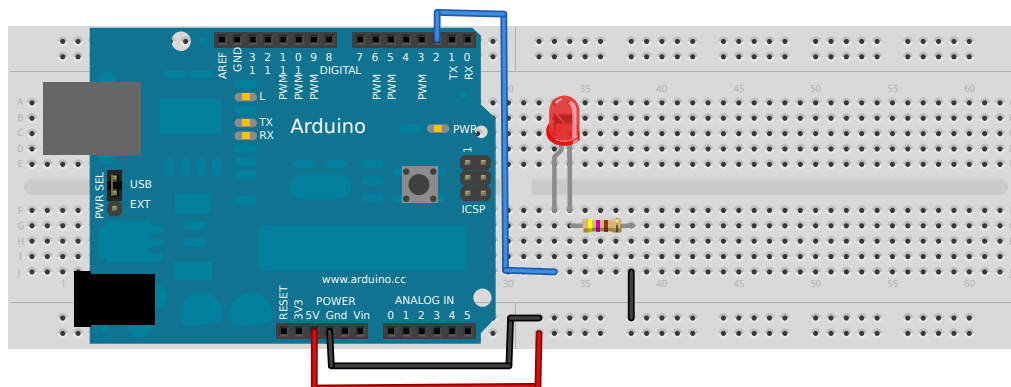
- Hay que especificar el tipo de Arduino y el puerto de conexión.
- Lenguaje de programación *C* o *C++*. Librerías propias de Arduino.

- *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

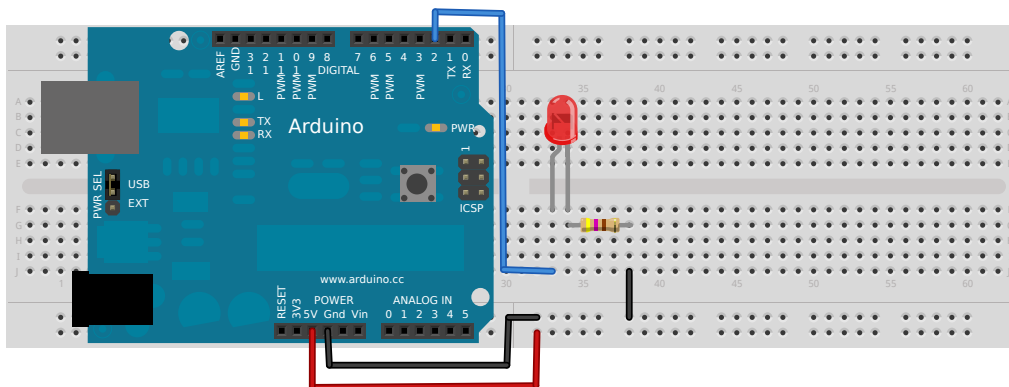
- `setup() { ... }` inicialización de todo el proceso. Se ejecuta una única vez al comienzo.
- `loop() { ... }` proceso iterativo que se ejecuta indefinidamente. Debe ejecutar de forma iterativa (cíclica) todas las tareas que componen el proceso.
- Almacenamiento de datos en *variables globales* accesibles 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.



Made with Fritzing.org

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

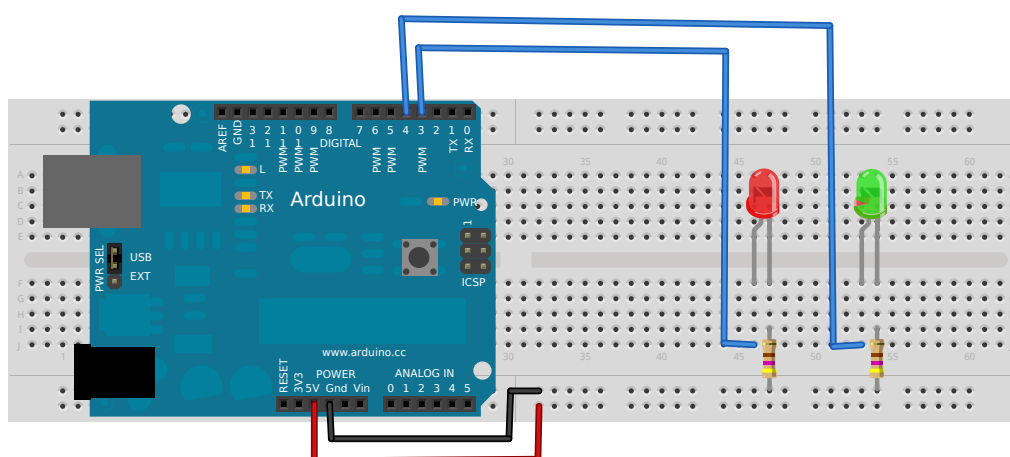
- **Circuito LED:** conecte el pin positivo de un LED al pin digital número 2 de Arduino, y el pin negativo del LED a una resistencia de 470Ω , la cual se conecta a masa (*Gnd*) con el otro pin.
- **Funcionamiento LED:** cuando se active 5V (HIGH) en el pin digital de salida correspondiente de Arduino, 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.



Made with  Fritzing.org

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

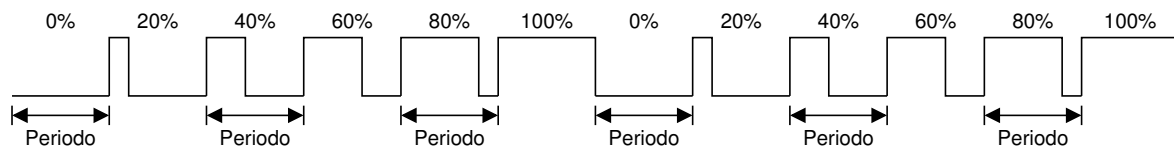
- **Circuito LED:** conecte el pin positivo de un LED rojo al pin digital número 3 de Arduino, y el pin negativo del LED a masa (*Gnd*) a través de una resistencia de 470Ω . Repita la misma conexión para un LED verde conectado al pin digital número 4 de Arduino.
- **Funcionamiento LED:** cuando se active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



Made with  Fritzing.org

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 % }.

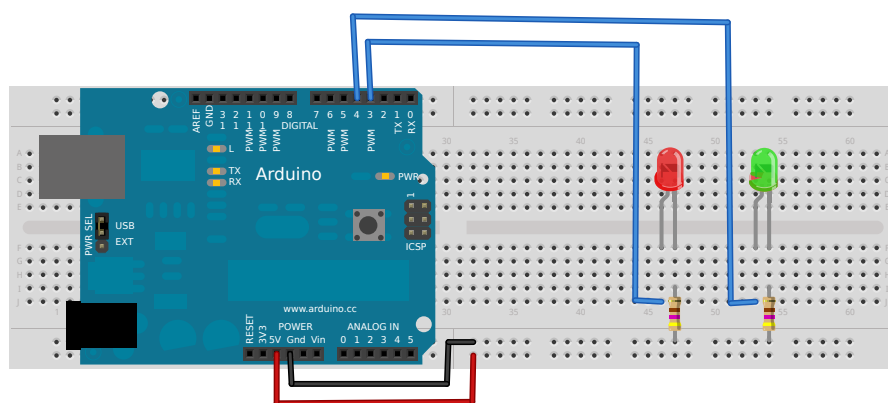


- **Monitorización:** con un periodo de 1000 ms, se enviará el estado del LED rojo, (0) para apagado y (1) para encendido, a través del puerto *USB* para que sea mostrado en el terminal de control. Utilice para ello la biblioteca *Serial*.
- Se deberá enviar al puerto *USB* al inicio del programa las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

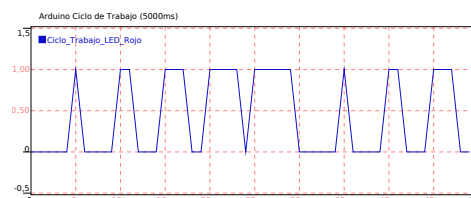
```
#> Arduino Ciclo de Trabajo (5000ms)
#\$ -y-1:2 -tCiclo_Trabajo_LED_Rojo
```

- 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



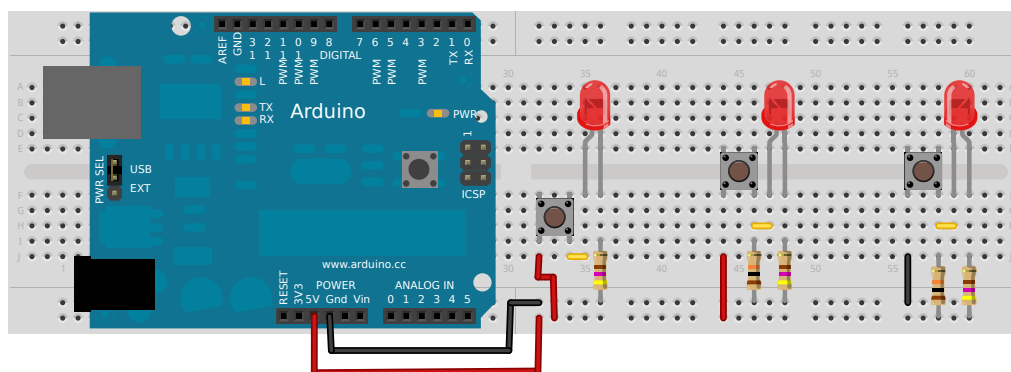
Made with Fritzing.org



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

Introducción. Control de Botones

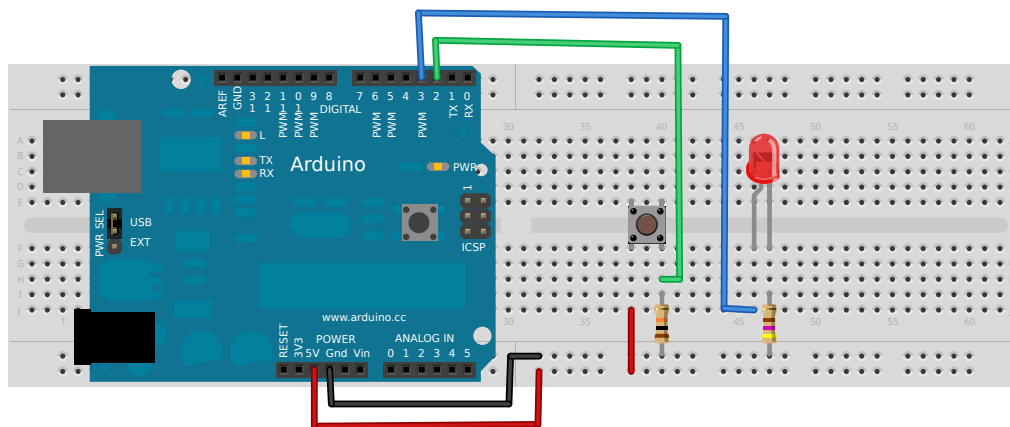
- **Circuito 1 y funcionamiento 1:** la descripción del primer circuito y su funcionamiento corresponden al *circuito-2* del *ejercicio 1* de esta misma relación de ejercicios.
- **Circuito 2:** conecte el pin positivo de un LED a 5V a través de un botón, y el pin negativo del LED a una resistencia de 470Ω , la cual se conecta a masa (*Gnd*) con el otro pin. Además, conecte una resistencia *Pull-Down* de $10K\Omega$ 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 $10K\Omega$, 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 $10K\Omega$, 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 $10K\Omega$ 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 $10K\Omega$, 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 $10K\Omega$, 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.



Made with  Fritzing.org

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 $10K\Omega$. 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 $10K\Omega$, 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.



Made with  Fritzing.org

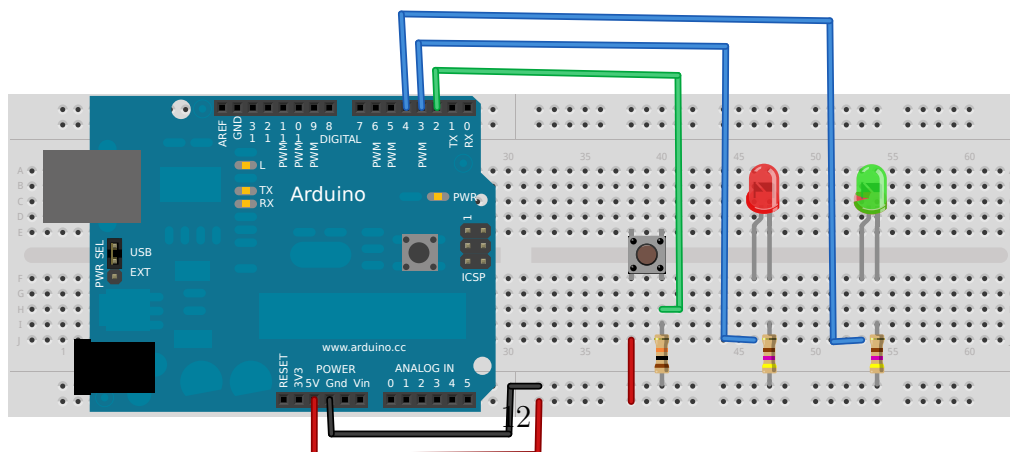
Ejercicio 7. Entrada y Salida Digital. Efecto Rebote. Depuración

- **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 $10K\Omega$. 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 $10K\Omega$, 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();`

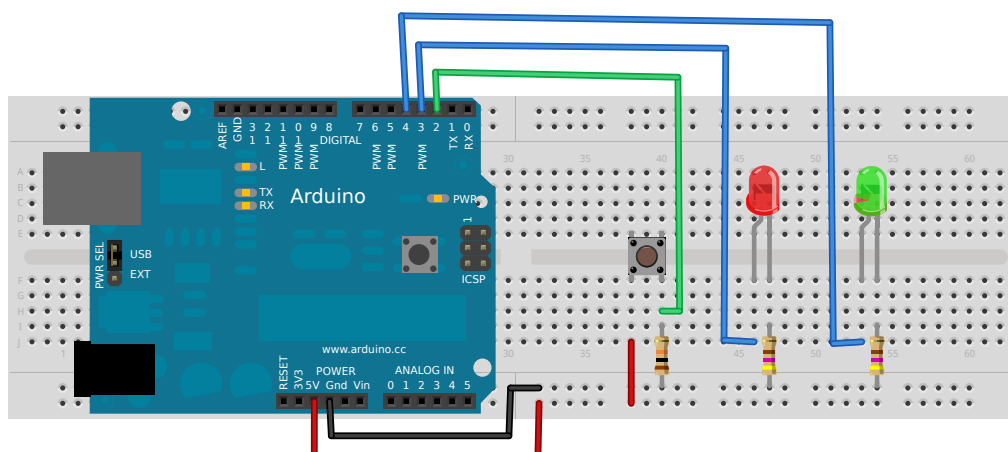


Ejercicio 8. Entrada y Salida Digital. Anular el Efecto Rebote SW

- **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 $10K\Omega$. 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 $10K\Omega$, 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();`

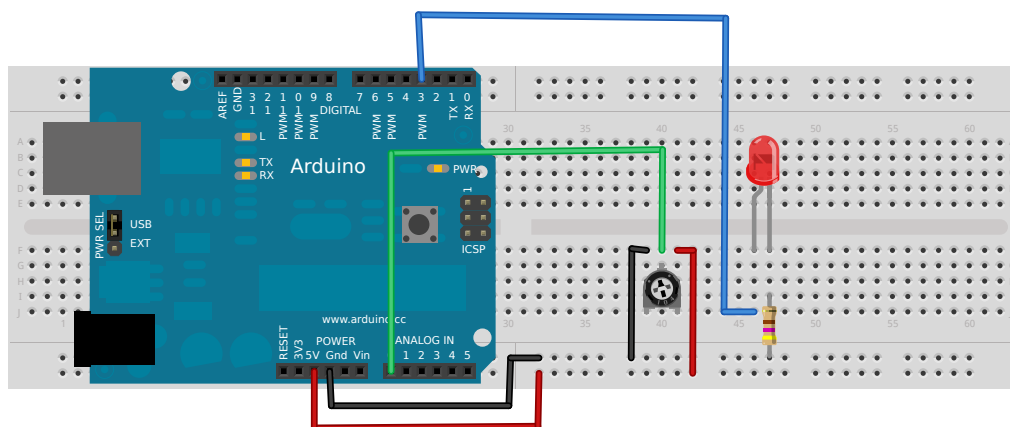


Ejercicio 9. Entrada Analógica (Pot) y Salida PWM

- **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.

$$5V \text{ --- } R_1 \text{ --- } V_A \text{ --- } R_2 \text{ --- } 0V \quad V_A = 5V \times \frac{R_2}{R_1 + R_2}$$

- **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.



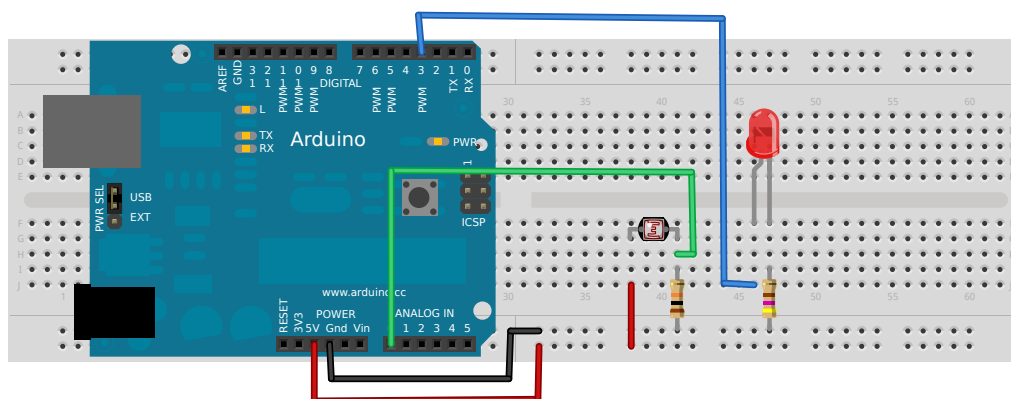
Made with  Fritzing.org

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 $10K\Omega$. 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.

$$5V \text{ --- } R_1 \text{ --- } V_A \text{ --- } 10K\Omega \text{ --- } 0V \quad V_A = 5V \times \frac{10000}{R_{LDR} + 10000}$$

- 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>



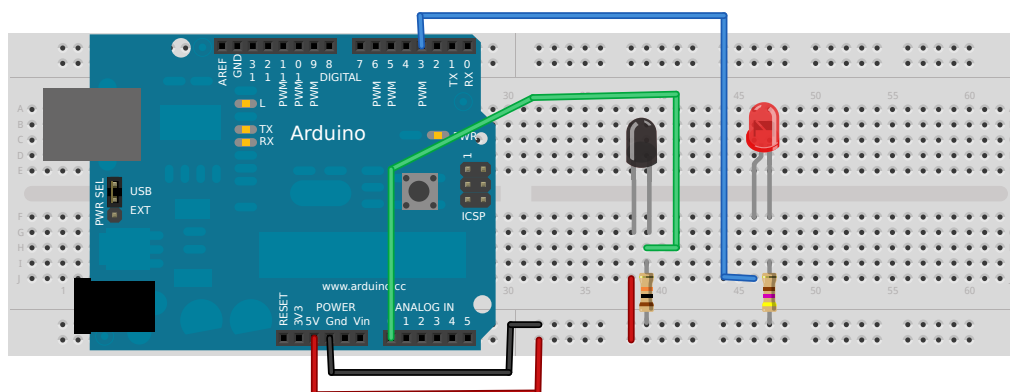
Made with  Fritzing.org

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 $10K\Omega$. 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.

$$5V \text{ --- } R_1 \text{ --- } V_A \text{ --- } 10K\Omega \text{ --- } 0V \quad V_A = 5V \times \frac{10000}{R_{TMP} + 10000}$$

- 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>



Made with Fritzing.org

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 $10K\Omega$. 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.

$$5V \text{ --- } R_1 \text{ --- } V_A \text{ --- } 10K\Omega \text{ --- } 0V \quad V_A = 5V \times \frac{10000}{R_{TMP} + 10000}$$

- **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:

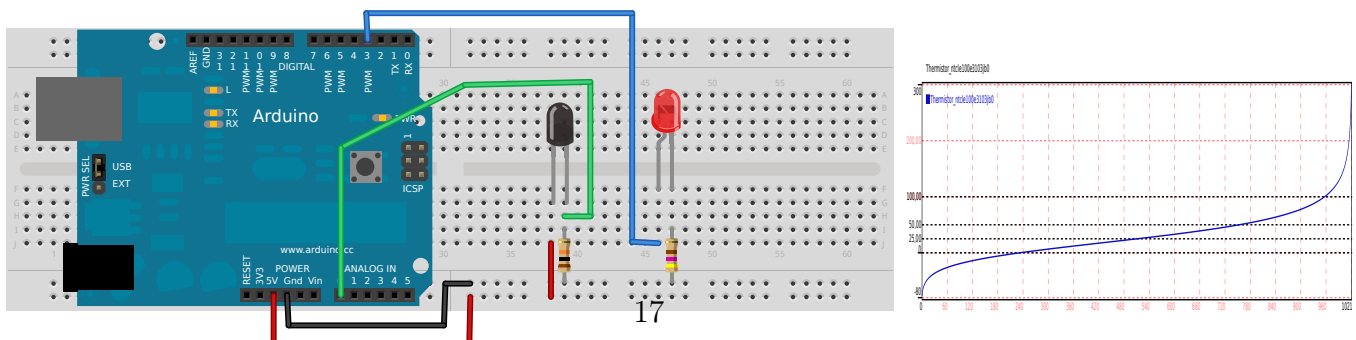
```
#> Arduino Sensor de Temperatura
#$ -y20:45 -l36 -l40 -tTemperatura -tLED
```

■ Ayuda:

- Se definen los siguientes parámetros para los siguientes termistores:

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>



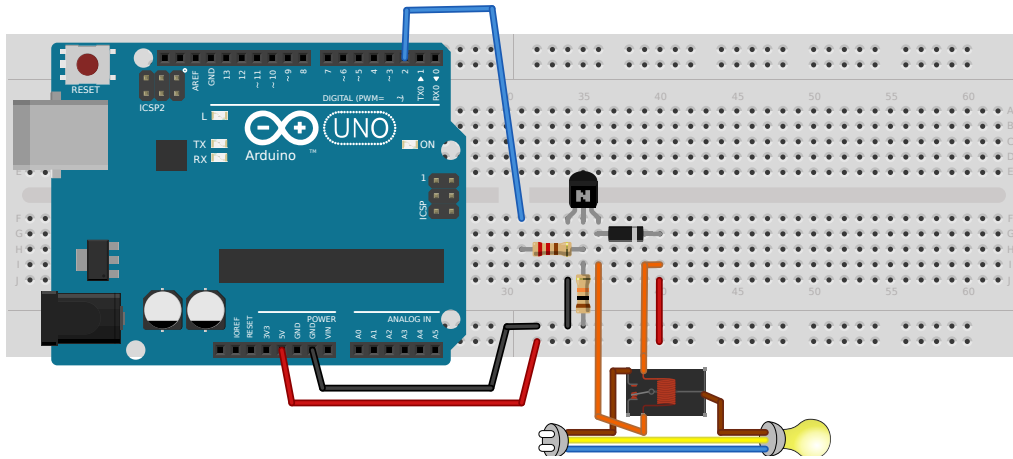
Ejercicio 13. Control de Carga Externa: Transistor NPN y Relé

- **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 $10K\Omega$.
- **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 periódicamente, 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

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 periódicamente 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 $^\circ\text{C}$: $\tau = \frac{1}{\left(\frac{1}{(T_0 + 273.15)} + \frac{1}{B} \ln\left(\frac{R_{TMP}}{R_0}\right)\right)} - 273.15$

- **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 1

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 -136 -138 -140 -tTemperatura -tCalefactor
```

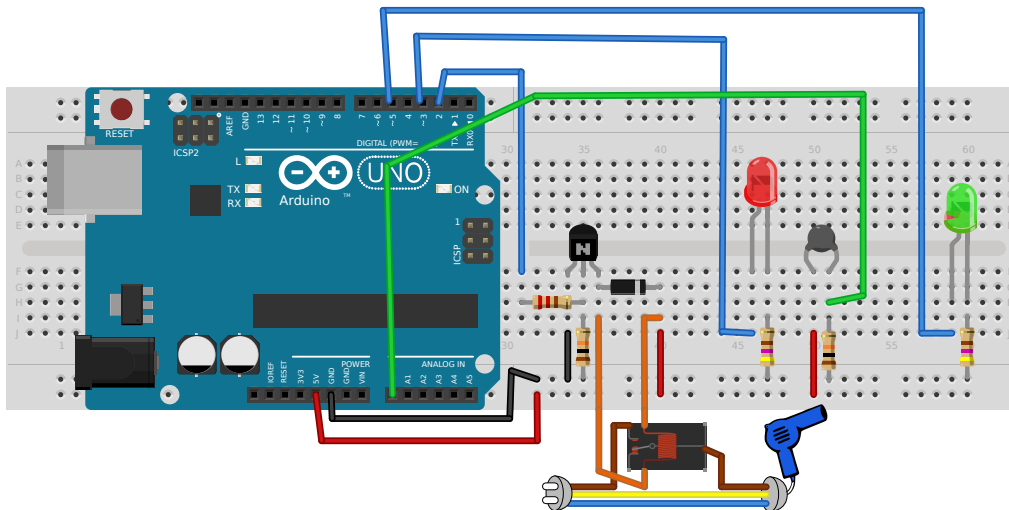
Continúa en la siguiente página

■ *Ayuda:*

- Se definen los siguientes parámetros para los siguientes termistores:

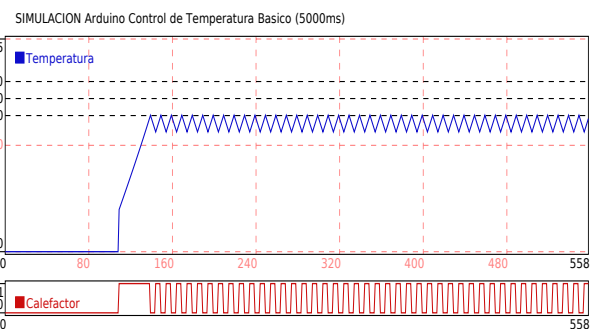
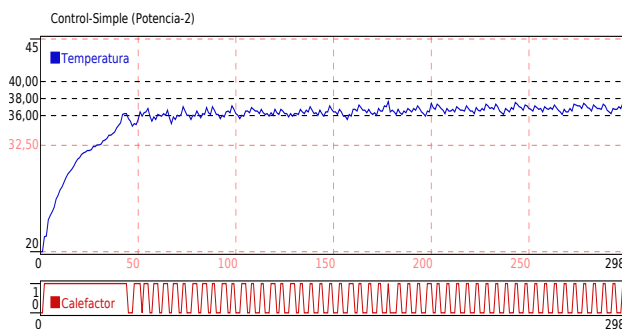
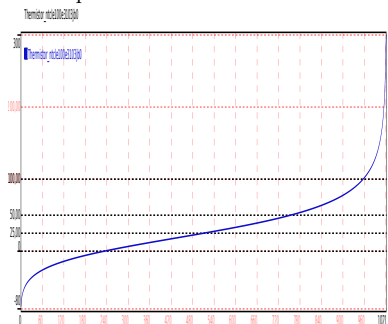
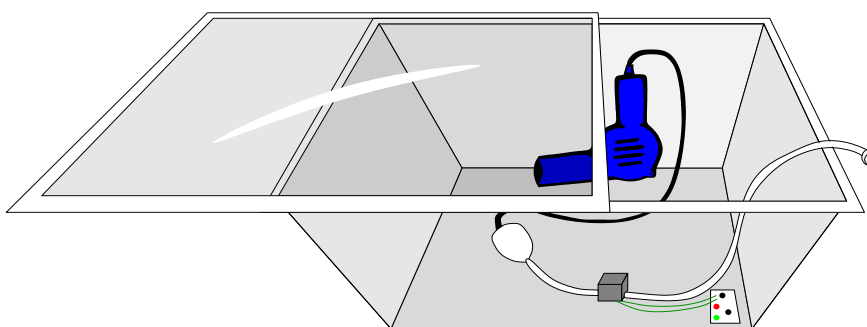
Thermistor	T_0	R_0	B
NTCLE100E3103JB0	25.0	10000.0	3977.0

- <http://dlmmh9ip6v2uc.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



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 periódicamente 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 $^\circ\text{C}$: $\tau = \frac{1}{\left(\frac{1}{T_0 + 273.15}\right) + \frac{1}{B} \ln\left(\frac{R_{TMP}}{R_0}\right)} - 273.15$

- **Control:** con un periodo de 5000 ms, calcula el *error* ($\varepsilon \in \mathbb{R}$) como la diferencia entre la temperatura deseada y la actual ($\varepsilon = 38^\circ\text{C} - \tau$), 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:

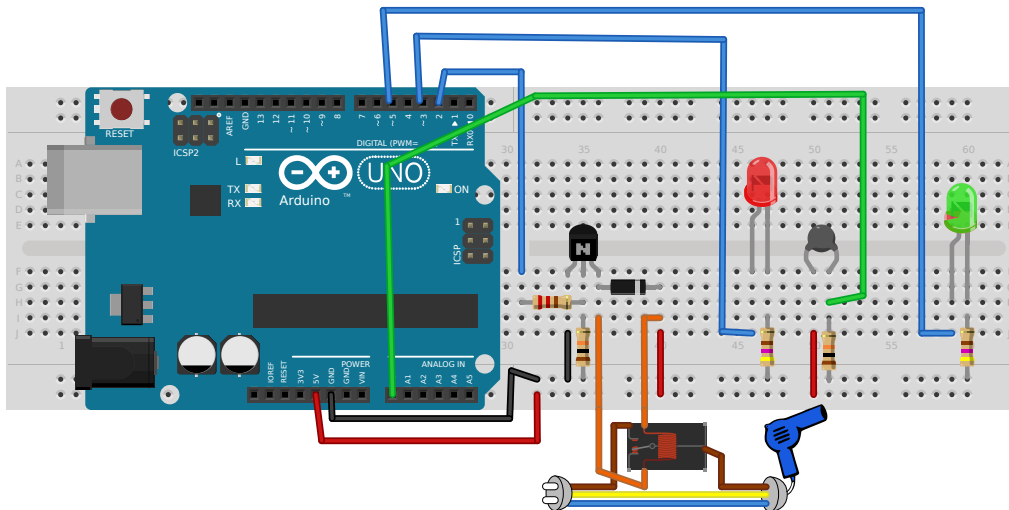
```
#> Arduino Control de Temperatura Ciclo Trabajo (5000ms)
## -y20:45 -w4 -136 -138 -140 -tTemperatura -tCalefactor
```

■ *Ayuda:*

- Se definen los siguientes parámetros para los siguientes termistores:

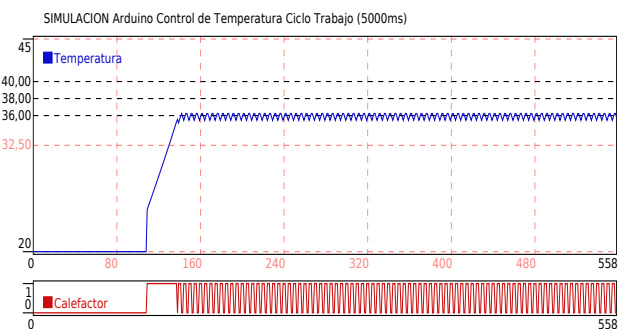
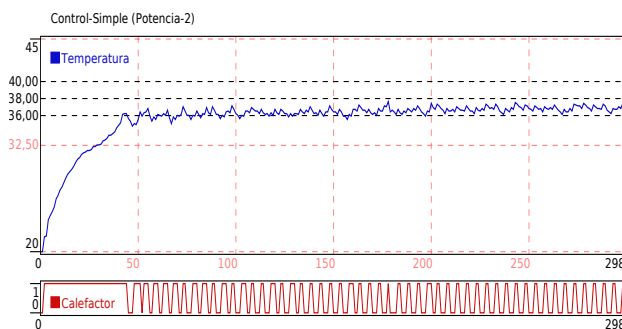
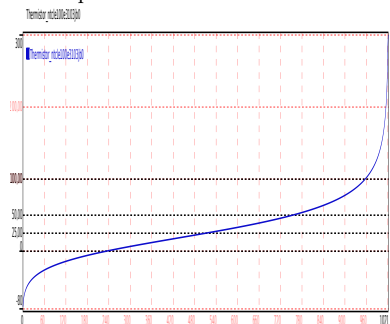
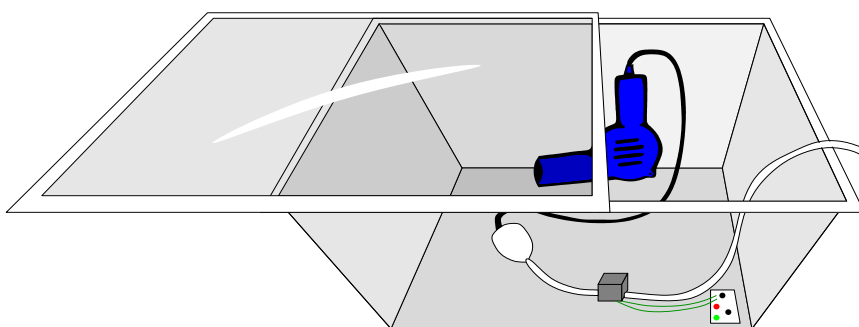
Thermistor	T_0	R_0	B
NTCLE100E3103JB0	25.0	10000.0	3977.0

- <http://dlmmh9ip6v2uc.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



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 periódicamente 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 $^\circ\text{C}$: $\tau = \frac{1}{\left(\frac{1}{(T_0 + 273.15)} + \frac{1}{B} \ln\left(\frac{R_{TMP}}{R_0}\right)\right)} - 273.15$

- **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 -136 -138 -140 -tTemperatura -tCalefactor
```


■ *Ayuda:*

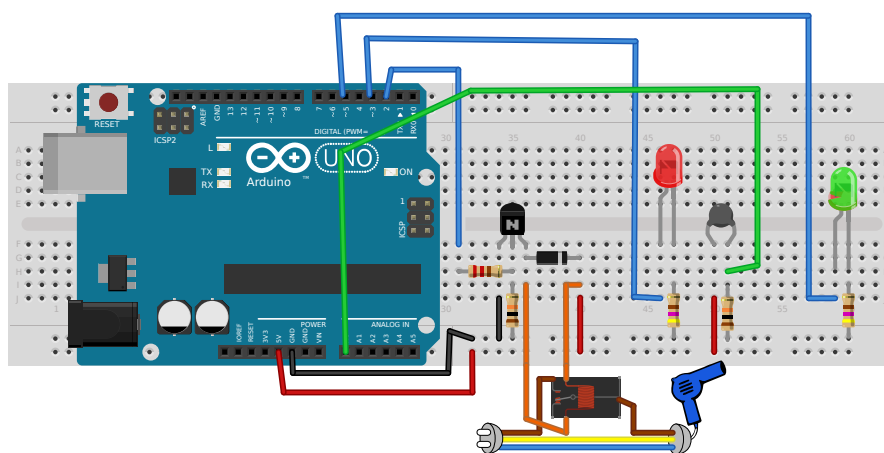
- Se definen los siguientes parámetros para los siguientes termistores:

Thermistor	T_0	R_0	B
NTCLE100E3103JB0	25.0	10000.0	3977.0

- <http://dlmh9ip6v2uc.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

$$\rho(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

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



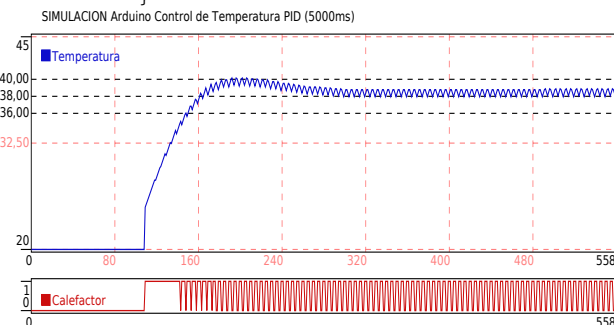
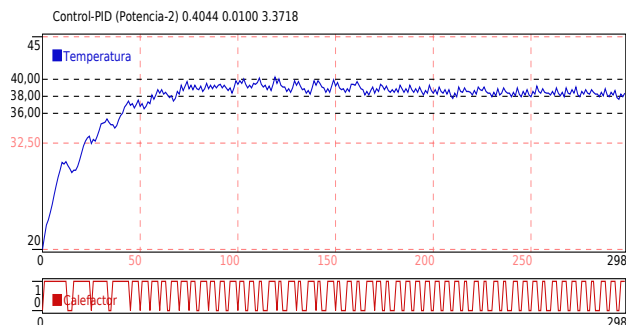
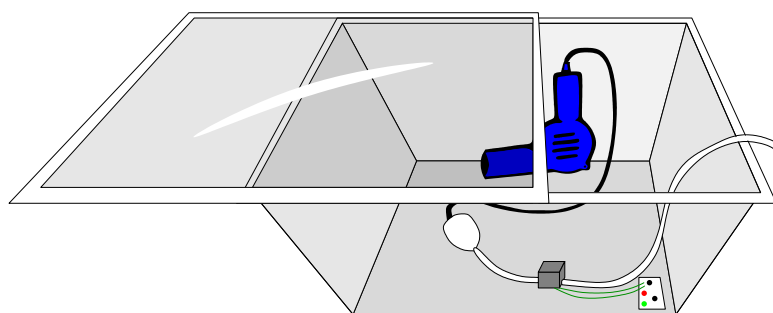
Transistor **PN2222A**: conexión **igual** al esquema. Transistor **BC547CG**: conexión **inversa** al esquema

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 ...
  }
}
```

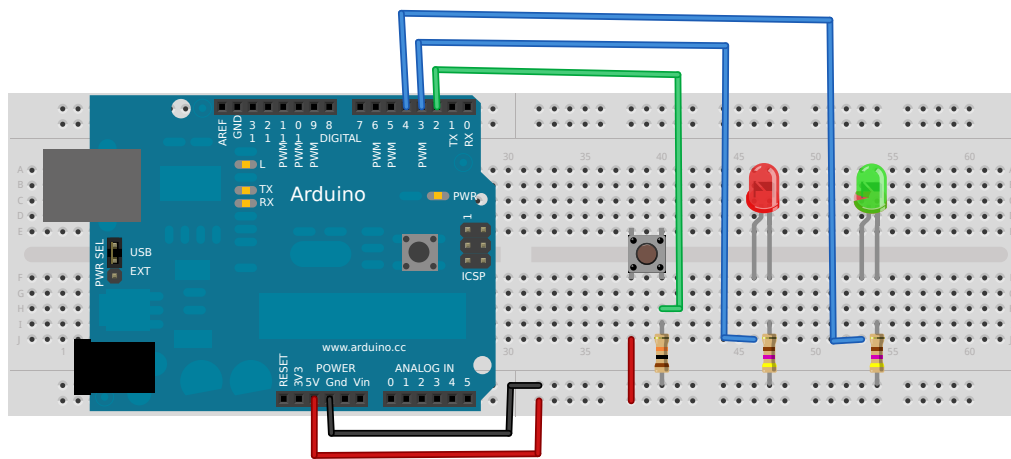


Ejercicio 17. Creación de Bibliotecas

- 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



Made with  Fritzing.org

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();

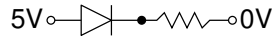
#include <EEPROM.h>
EEPROM.write(address, value);
byte b = EEPROM.read(address);

#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

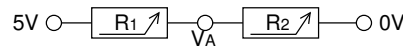
■ Cálculo del valor de la resistencia en un LED:

- 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: $R_{LED} = V/I = (5 - 2)/0.003 = 1K\Omega$



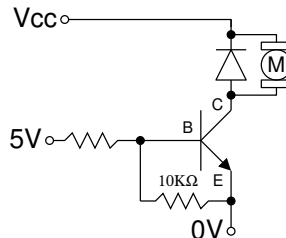
■ Cálculos en un *divisor de voltaje*:

- Voltaje en *voltios* a partir del valor analógico leído *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 leído *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: $R_{B(sat)} = (V_B - V_{BE(sat)})/I_{B(sat)} = (5 - 1)/0.033 = 121\Omega$

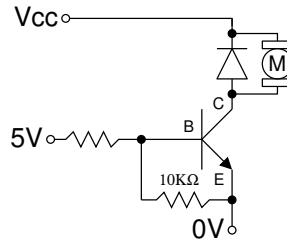


■ Cálculo del valor de la intensidad de colector en un transistor NPN (PN2222A) para una resistencia de base dada:

- 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: 1KΩ.
 - 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: $R_{B(sat)} = (V_B - V_{BE(sat)}) / I_{B(sat)} = (5 - 1) / 0.006 = 666\Omega$

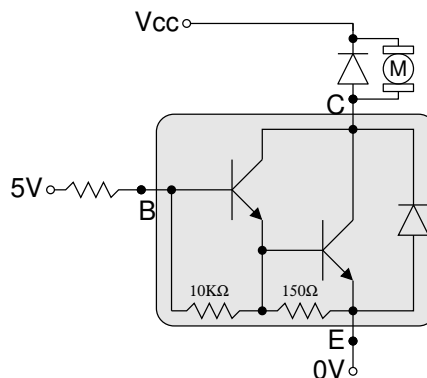


■ Cálculo del valor de la intensidad de colector en un transistor NPN (2N3904) para una resistencia de base dada:

- 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: 1KΩ.
 - 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: $R_{B(sat)} = (V_B - V_{BE(sat)}) / I_{B(sat)} = (5 - 2.5) / 0.013 = 187\Omega$

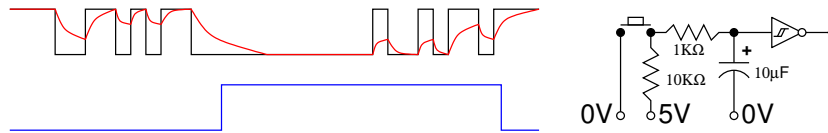


■ Cálculo del valor de la intensidad de colector en un transistor Darlington NPN (BDX33C) para una resistencia de base dada:

- 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: 1KΩ.

- 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 ($V_i = 5V$, $V_{T-} = [0.5..1.0]$): $V_{cap} = V_i \times \left(e^{\frac{-t}{RC}} \right) \Rightarrow t = -RC \ln \left(\frac{V_{cap}}{V_i} \right)$
 $0.0016sg = -1K\Omega \times 1\mu F \times \ln \left(\frac{1V}{5V} \right)$
- Carga ($V_f = 5V$, $V_{T+} = [1.4..1.9]$): $V_{cap} = V_f \times \left(1 - e^{\frac{-t}{RC}} \right) \Rightarrow t = -RC \ln \left(1 - \frac{V_{cap}}{V_f} \right)$
 $0.0036sg = -11K\Omega \times 1\mu F \times \ln \left(1 - \frac{1.4V}{5V} \right)$



- 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

<div> <div> 0 1 2 3 4 5 6 7 8 9 </div> <div> 0 Black 1 Brown 2 Red 3 Orange 4 Yellow 5 Green 6 Blue 7 Purple 8 Grey 9 White </div> </div> <div> <div> ±1% ±2% ±5% ±10% </div> <div> Brown Red Gold Silver </div> </div>	<div> <div> ±1% ±2% ±5% ±10% </div> <div> 27K EXAMPLE </div> <div> 0 X1 1 1 X10 2 2 X100 3 3 X1000 4 4 X10000 5 5 X100000 6 6 X1000000 7 7 X10 8 8 X100 9 9 </div> </div>	<div> <div> ±1% ±2% ±5% ±10% </div> <div> 15K EXAMPLE </div> <div> 0 0 X1 1 1 1 X10 2 2 2 X100 3 3 3 X1000 4 4 4 X10000 5 5 5 X100000 6 6 6 X1000000 7 7 7 X100 8 8 8 X1000 9 9 9 </div> </div>	<div> <div> ±1% ±2% ±5% ±10% </div> <div> 100 50 25 15 10 5 1 620K EXAMPLE </div> <div> 0 0 X1 1 1 1 X10 2 2 2 X100 3 3 3 X1000 4 4 4 X10000 5 5 5 X100000 6 6 6 X1000000 7 7 7 X100 8 8 8 X1000 9 9 9 </div> </div>
Color Codes	4 Band Resistors	5 Band Resistors	6 Band Resistors

ROW	GOLD	BLACK	BROWN	RED	ORANGE	YELLOW	GREEN
1-	1R0	10R	100R	1K0	10K	100K	1M0
2-	1R1	11R	110R	1K1	11K	110K	1M1
3-	1R2	12R	120R	1K2	12K	120K	1M2
4-	1R3	13R	130R	1K3	13K	130K	1M3
5-	1R5	15R	150R	1K5	15K	150K	1M5
6-	1R6	16R	160R	1K6	16K	160K	1M6
7-	1R8	18R	180R	1K8	18K	180K	1M8
8-	2R0	20R	200R	2K0	20K	200K	2M0
9-	2R2	22R	220R	2K2	22K	220K	2M2
10-	2R4	24R	240R	2K4	24K	240K	2M4
11-	2R7	27R	270R	2K7	27K	270K	2M7
12-	3R0	30R	300R	3K0	30K	300K	3M0
13-	3R3	33R	330R	3K3	33K	330K	3M3
14-	3R6	36R	360R	3K6	36K	360K	3M6
15-	3R9	39R	390R	3K9	39K	390K	3M9
16-	4R3	43R	430R	4K3	43K	430K	4M3
17-	4R7	47R	470R	4K7	47K	470K	4M7
18-	5R1	51R	510R	5K1	51K	510K	5M1
19-	5R6	56R	560R	5K6	56K	560K	5M6
20-	6R2	62R	620R	6K2	62K	620K	6M2
21-	6R8	68R	680R	6K8	68K	680K	6M8
22-	7R5	75R	750R	7K5	75K	750K	7M5
23-	8R2	82R	820R	8K2	82K	820K	8M2
24-	9R1	91R	910R	9K1	91K	910K	9M1
							10M
							BLUE

COLOR CODES FOR THE WHOLE E12/E24 RANGE OF RESISTORS

The twelve odd rows - 1, 3, 5... - represent values available in the E12 range only, plus 10M