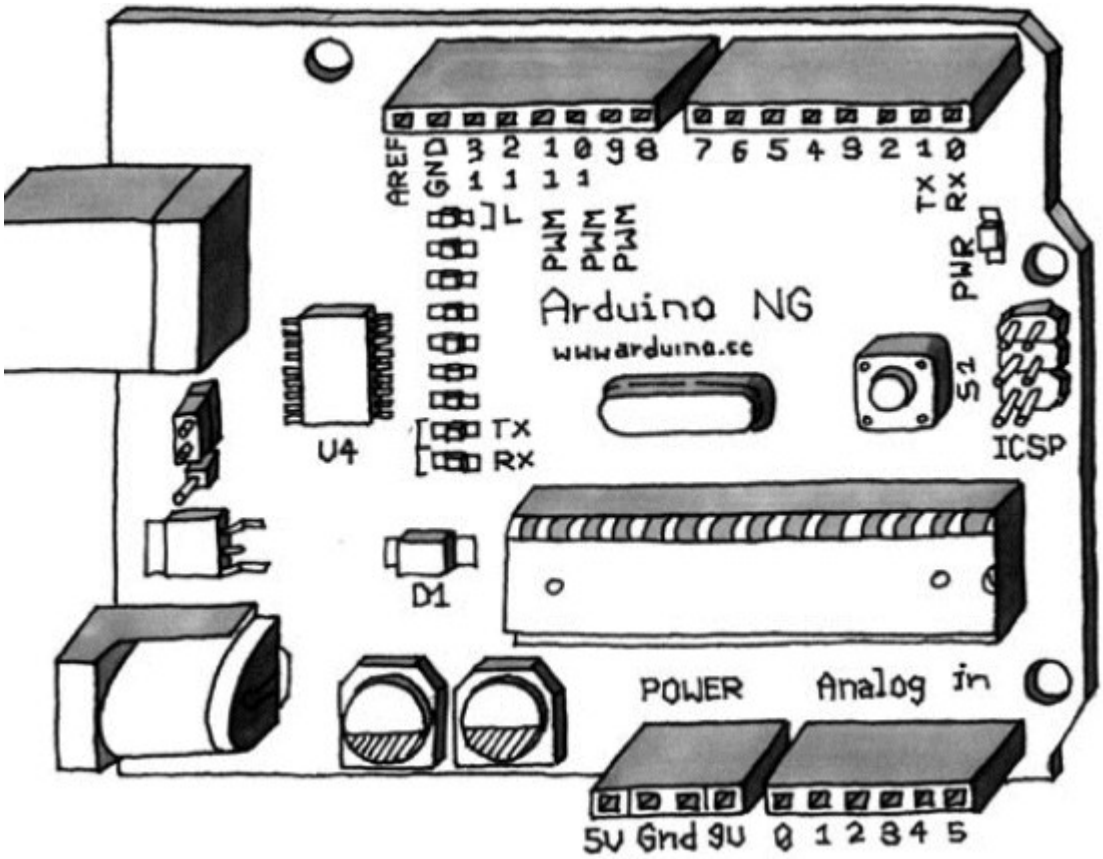
EJERCICIOS DE *ARDUINO* RESUELTOS

Grupo *Sabika*



Revisado: 11/11/2014

Instalar Entorno de Programación Arduino en Ubuntu (10.10, 10.04, 9.10 y 9.04)

Para la instalación de Arduino se requieren ciertos paquetes para su funcionamiento...

- librxtx-java *// Librería para comunicación serial*
- avr-libc & gcc-avr *// Paquete de compiladores para la programación de Microcontroladores Atmel con Lenguaje C*
- sun-java6-jre *// Motor Java*

1) Puede instalar estos paquetes desde Synaptic como sigue: Sistema > Administración > Gestor de Paquetes Synaptic En la ventana del Synaptic proceda a seleccionar cada uno de los paquetes mencionados ó desde una consola (terminal) escribiendo lo siguiente: *sudo apt-get install librxtx-java avr-libc gcc-avr sun-java6-jre*
2) Descargue arduino desde su pagina web en http://arduino.cc/ o abra una terminal y escriba lo siguiente:
Para versiones de 32 bits (i386) *wget http://arduino.googlecode.com/files/arduino-0021.tgz*
Para versiones de 64 bits (amd64) *wget http://files.arduino.cc/downloads/arduino-0021-2.tgz*
3) Descomprimalo...
Realizando doble clic sobre el archivador o en una terminal con *tar xvf arduino-0021.tgz*
4) Ubíquese en la carpeta...
Con el navegador de archivos o mediante una terminal *cd arduino-0021*
5) Ejecútelo …
Realizando doble clic sobre el archivo llamado *arduino* o mediante un terminal *./arduino*

Otra forma muy sencilla de Instalar Arduino en Ubuntu 9.10 y 10.04 es a través del repositorio de Arduino para Ubuntu, para ello seguimos los siguientes pasos en un terminal de Linux, «menú Aplicaciones > Accesorios > Terminal»:
1) «sudo add-apt-repository ppa:arduino-ubuntu-team». Añade el repositorio de ubuntu a las orígenes de software de tu equipo.
2) «sudo apt-get update». Actutaliza los orígenes de software de tu equipo y por tanto los repositorios.
3) «sudo apt-get install arduino». Instala Arduino con todas sus dependencias.
4) Arduino aparece en el «menú Aplicaciones > Programación > Arduino».

Nota: las ordenes que tengan «sudo» delante requieren permisos de administrador y por tanto pedirá la contraseña de adminsitrador.

En la actual Ubuntu 10.10 desde el «centro de software de Ubuntu» se instala directamente.

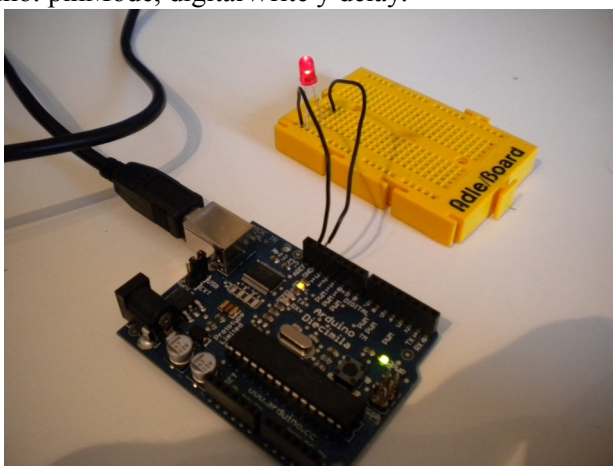
EJERCICIOS DE ARDUINO.

Led parpadeante.

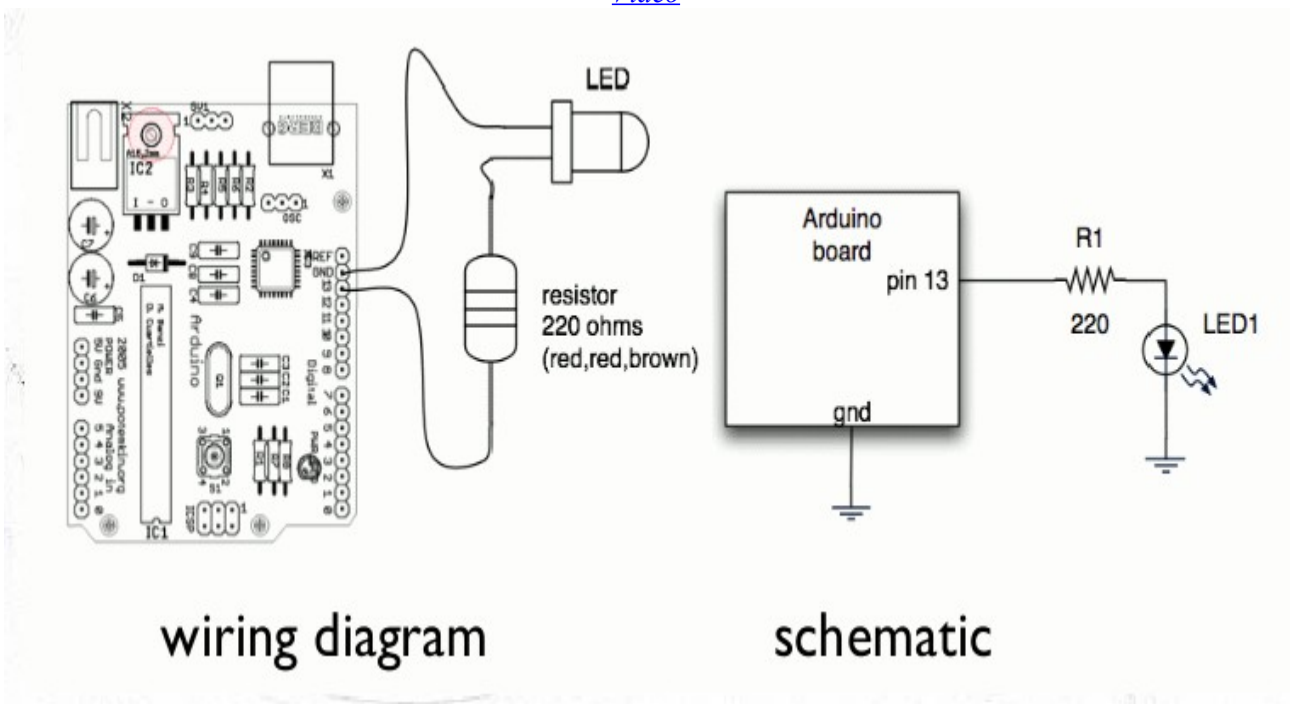
Se trata de conectar un led al pin13, haciendo que luzca durante 500 ms y que se apague durante 100 ms, este proceso se repetirá cíclicamente.

Objetivos:

- Reconocer partes de la placa.
- Aprender a conexionar leds a la placa.
- Familiarizarse con el entorno de programación.
- Reconocer las partes de un programa de arduino.
- Conocer órdenes como: pinMode, digitalWrite y delay.



Vídeo



Solución:

```
void setup() { //comienza la configuracion
pinMode(13, OUTPUT); //configura el pin 13 como de salida
} //termina la configuracion

void loop() { //comienza el bucle principal del programa
digitalWrite(13, HIGH); //envia 5V al pin (salida) 13
delay (500); //espera 500 ms pin 13 con 5V
```

```
digitalWrite(13, LOW); //envia 0V al pin (salida) 13
delay (100); //espera 100 ms pin 13 con 0V
}
```

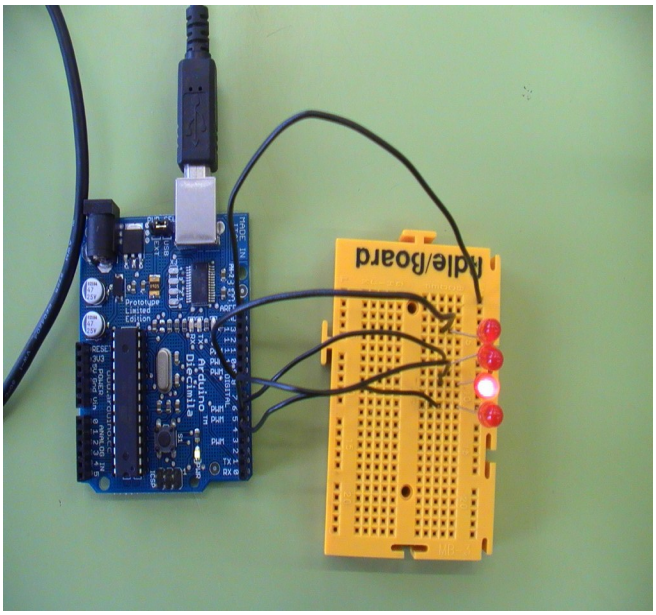
Secuencia de leds.

Se trata de encender y apagar 4 leds secuencialmente. Los leds deben estar conectados a los pines 5,6,7 y 8. Se deben encender y posteriormente apagar los leds desde el pin 5 al 8, con un tiempo de duración de encendido y apagado de 200 milisegundos.

Nota: en una segunda soución la secuencia principal del programa debe estar reproducida en una función a la que llamará el programa principal.

Objetivos:

- Familiarizarse con el entorno de programación.
- Aprender a declarar variables y variables tipo lista de valores.
- Aprender a declarar una función y llamarla cuando sea necesario.



Video

Solución 1:

```
int tiempo=200; //declara una variable como entero y de valor 200

void setup() { //comienza la configuracion
pinMode(5,OUTPUT);
pinMode(6,OUTPUT);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
}

void loop() { //comienza el bucle principal del programa
digitalWrite(5,HIGH);
delay(tiempo);
digitalWrite(5,LOW);
delay(tiempo);
digitalWrite(6,HIGH);
delay(tiempo);
digitalWrite(6,LOW);
delay(tiempo);
digitalWrite(7,HIGH);
delay(tiempo);
```

```
digitalWrite(7,LOW);
delay(tiempo);
digitalWrite(8,HIGH);
delay(tiempo);
digitalWrite(8,LOW);
delay(tiempo);
}
```

Solución 2:

```
int tiempo=200;
int n;

void setup() { //comienza la configuracion
for (n=5;n<9;n++) {
pinMode (n, OUTPUT);
}
}

void secuencia() {
for (n=5;n<9;n++) {
digitalWrite (n, HIGH);
delay (tiempo);
digitalWrite (n, LOW);
delay (tiempo);
}
}

void loop() {
secuencia();
}
```

Solución 3:

```
int leds[]={5,6,7,8};// Declara variables tipo lista de valores
int tiempo=200;
int n=0;
void setup() { //comienza la configuracion
for (n=0;n<4;n++) {
pinMode (leds[n], OUTPUT);
}
}
void secuencia() {
for (n=0;n<4;n++) {
digitalWrite (leds[n], HIGH);
delay (tiempo);
digitalWrite (leds[n], LOW);
delay (tiempo);
}
}
void loop() {
secuencia();
}
```

Solución 4:

```
int leds[]={5,6,7,8};// Declara variables tipo lista de valores
int n=0;
```

```
void setup() { //comienza la configuracion
for (n=0;n<4;n++) {
pinMode (leds[n], OUTPUT);
}
}
void secuencia(int tiempo) {
for (n=0;n<4;n++) {
digitalWrite (leds[n], HIGH);
delay (tiempo);
digitalWrite (leds[n], LOW);
delay (tiempo);
}
}
void loop() {
secuencia(200);
}
```

Cruce de semáforos.

Se trata de un cruce de semáforos controlado por arduino, para ello utilizaremos en el primer semáforo los pines 3 (led rojo), 4 (led ambar), 5 (led verde), en el segundo semáforo utilizaremos los pines 6 (led rojo), 7 (led ambar) y 8 (led verde). La secuencia de funcionamiento debe ser : rojo 1 – verde 2 durante 3 segundos, rojo 1 – ambar 2 durante 500 ms, verde 1 – rojo 2 durante 3 segundos, ambar 1 - , rojo 2 durante 500 ms.

Objetivos:

- Familiarizarse con el entorno de programación.
- Aprender a declarar variables tipo lista de valores.

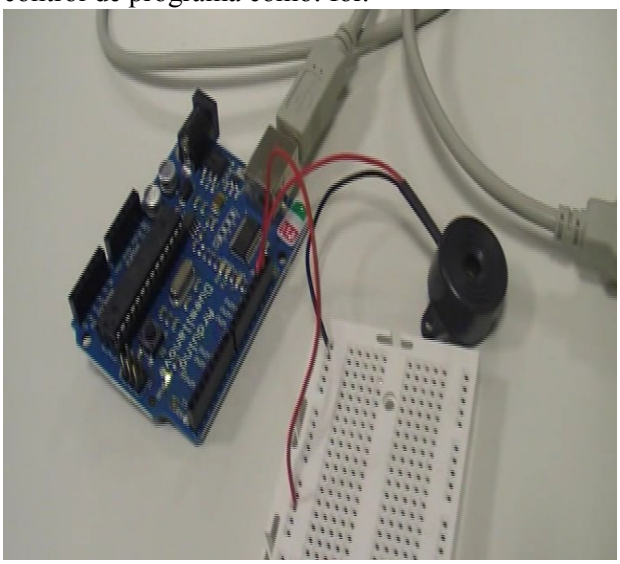
Solución:

```
int leds[]={3,4,5,6,7,8);
int tiempo1=3000;
int tiempo2=500;
int n;

void setup() {
for (n=0;n<6;n++) {
pinMode (leds[n],OUTPUT);
}
}

void loop () {
digitalWrite (leds[0],HIGH);
digitalWrite (leds[5],HIGH);
delay (tiempo1);
digitalWrite (leds[5],LOW);
digitalWrite (leds[4],HIGH);
delay (tiempo2);
difitalWrite[leds[0],LOW);
digitalWrite (leds[2],HIGH);
digitalWrite (leds[4],LOW);
digitalWrite (leds[3],HIGH);
delay (tiempo1);
digitalWrite (leds[2],LOW);
digitalWrite(leds[1],HIGH);
delay (tiempo2);
}
```

SOS con zumbador.
Se trata de un zumbador que en código morse (pitidos largos/cortos) especifica una palabra, en nuestro caso SOS. Para el que no lo sepa, la S son tres señales acústicas de corta duración y la O tres señales acústica de larga duración.
El zumbador debe estar conectado al pin 13, los pitidos cortos tendrán una duración de 100 ms y los largos 300 ms. Entre letra y letra debe pasar un tiempo de 300 ms y entre SOSs debe haber un tiempo de 1000 ms.
Nota: Debes usar variables para guardar los tiempos que vas a usar.

Objetivos:

- Reconocer partes de la placa.
- Aprender a conexionar un zumbador a la placa.
- Familiarizarse con el entorno de programación.
- Reconocer las partes de un programa de arduino.
- Aprender a como declarar variables.
- Conocer órdenes de control de programa como: for.



Video

Solución:

```
int corto=100; //Declara la variable de argumento entero "corto" y la inicializa con el valor 100 (letra S)
int pausa=300;//tiempo entre letra y letra
int largo=300; //variable de argumento entero "largo" y la inicializa con el valor 300 (letra O)
int espera=1000;//variable argumento entero "espera" y la inicializa con el valor 1000 (tiempo entre SOS - SOS)
int n=0;
int zumb=13; //PIN digital al que conectamos el zumbador

void setup(){ //comienza la configuracion
pinMode(zumb,OUTPUT);
}

void loop(){
for(n=0;n<3;n++){ //Iteracion en la que la variable n comienza con el valor 0
digitalWrite(zumb, HIGH); // y va aumentando en 1 en cada ciclo hasta que toma el valor 2,
delay(corto); // con lo que las instrucciones comprendidas entre los corchetes
digitalWrite(zumb,LOW); // se repiten 3 veces
delay(corto);
}
delay(pausa); //Tiempo entre letras
for(n=0;n<3;n++){ //Aqui esta la O
```

```
digitalWrite(zumb, HIGH);
delay(largo);
digitalWrite(zumb,LOW);
delay(largo);
}
delay(pausa);
for(n=0;n<3;n++){
digitalWrite(zumb, HIGH);
delay(corto);
digitalWrite(zumb,LOW);
delay(corto);
}
delay(espera); //Tiempo hasta repetir SOS de nuevo
}
```

Solución 2:

```
int tcorto=100;
int tlargo=300;
int pausa=300;
int espera=1000;
int n=0;

void setup(){ //comienza la configuracion
pinMode(13,OUTPUT);
}

void s(){ //comienza el bucle para la letra S
  for(n=0;n<3;n++) {
    digitalWrite (13,HIGH);
    delay (tcorto);
    digitalWrite (13,LOW);
    delay (tcorto);
}
}

void o(){ //comienza el bucle para la letra O
  for(n=0;n<3;n++) {
    digitalWrite (13,HIGH);
    delay (tlargo);
    digitalWrite (13,LOW);
    delay (tlargo);
}
}

void loop(){ //se ejecuta el bucle principal en el orden siguiente
  s();
  delay(pausa);
  o();
  delay(pausa);
  s();
  delay(espera);
}
```

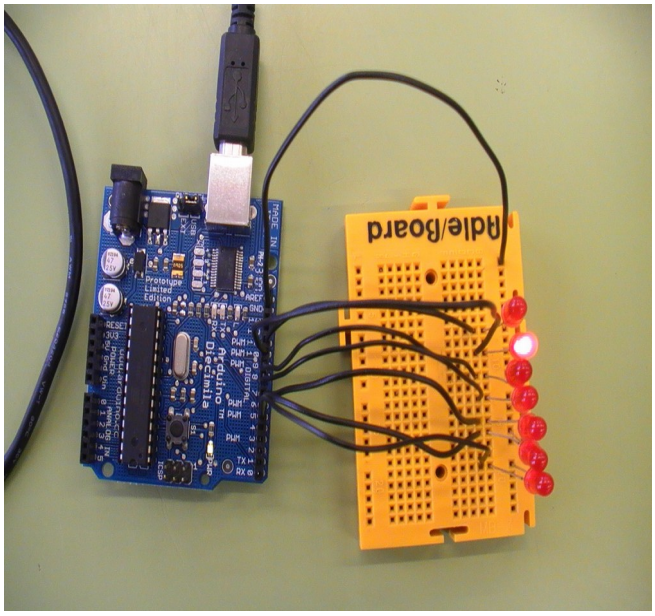
Coche Fantástico.

Se trata de encender y apagar 7 leds secuencialmente. Los leds deben estar conectados a los pines 5,6,7,8,9,10 y 11.

Se deben encender y apagar los leds desde el pin 5 al 11, con un tiempo de encendido y apagado de 50 ms, más tarde se deben encender y apagar los leds desde el pin 11 al 5, con un tiempo de encendido y apagado de 50 ms. La secuencia se debe repetir indefinidamente.
El efecto del programa es el de las luces delanteras de nuestro querido "Coche fantástico".

Objetivos:

- Familiarizarse con el entorno de programación.
- Repasar declaración de variables tipo lista de valores.
- Repasar órdenes de control de programa como: for.



Video

Solución:

```
int leds[]={5,6,7,8,9,10,11};
int n=0;
int tiempo=50;

void setup() { //comienza la configuración
  for (n=0;n<7;n++) {
    pinMode(leds[n],OUTPUT);
  }
}

void loop() {
    for (n=0;n<7;n++) {
    digitalWrite (leds[n],HIGH);
    delay(tiempo);
    digitalWrite (leds[n],LOW);
    delay(tiempo);
  }
  for (n=6;n>=0;n--) {
    digitalWrite (leds[n],HIGH);
    delay(tiempo);
    digitalWrite (leds[n],LOW);
    delay(tiempo);
  }
}
```

Solución 2 (sin variable de listas de valores (array)):

```
int n=0;
int tiempo=50;

void setup() { //comienza la configuración
 for (n=5;n<12;n++) {
   pinMode(n,OUTPUT);
  }
}

void loop() {
   for (n=5;n<12;n++) {
   digitalWrite (n,HIGH);
   delay(tiempo);
   digitalWrite (n,LOW);
   delay(tiempo);
  }
 for (n=11;n>=5;n--) {
   digitalWrite (n,HIGH);
   delay(tiempo);
   digitalWrite (n,LOW);
   delay(tiempo);
  }
}
```

Solución 3 (Mejorando el efecto visual):

```
int leds[]={5,6,7,8,9,10,11};
int n=0;
int tiempo=30;

void setup() { //comienza la configuración
 for (n=0;n<7;n++) {
   pinMode(leds[n],OUTPUT);
  }
}

void loop() {
   for (n=0;n<7;n++) {
   digitalWrite (leds[n],HIGH);
   delay(tiempo);
   digitalWrite(leds[n+1],HIGH);
   delay(tiempo);
   digitalWrite (leds[n],LOW);
   delay(tiempo*2);
  }
 for (n=6;n>=0;n--) {
   digitalWrite (leds[n],HIGH);
   delay(tiempo);
   digitalWrite(leds[n-1],HIGH);
   delay(tiempo);
   digitalWrite (leds[n],LOW);
   delay(tiempo*2);
  }
}
```

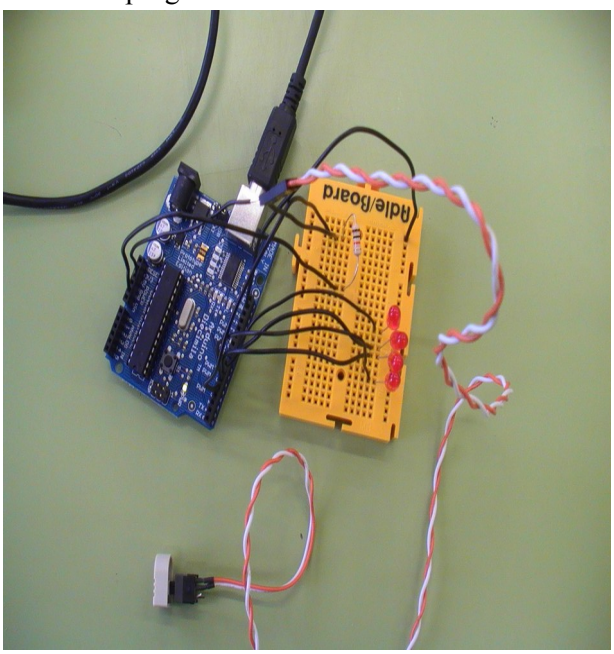
Secuencia de leds con pulsador.

Se trata de encender y apagar 4 leds secuencialmente al accionar un pulsador. El pulsador debe estar conectado al pin 4, y los leds a los pines 5,6,7 y 8.

Se deben encender y posteriormente apagar los leds desde el pin 5 al 8, con un tiempo de duración de encendido y apagado de 200 milisegundos.
Nota: la secuencia principal del programa debe estar reproducida en una función a la que llamará el programa principal.

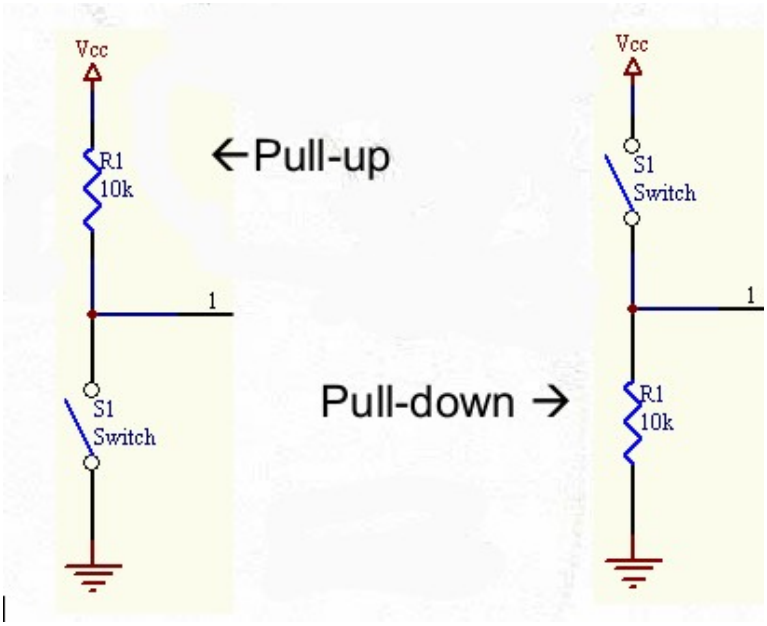
Objetivos:

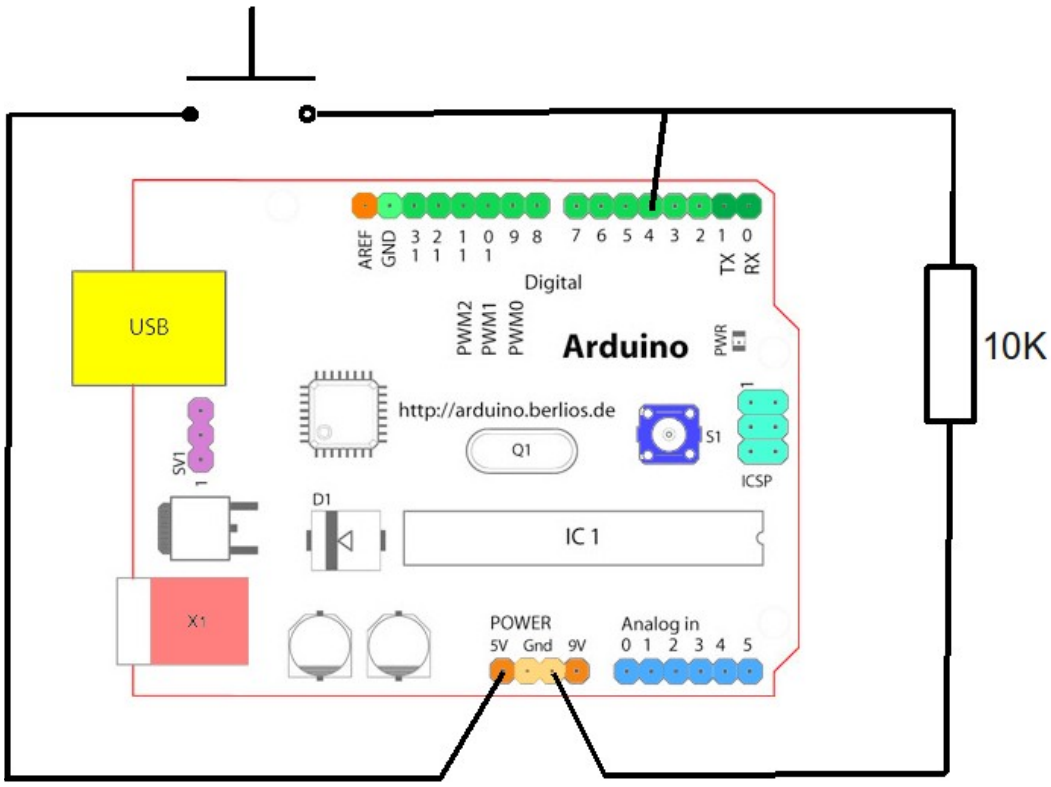
- Familiarizarse con el entorno de programación.
- Aprender a conectar una entrada digital a arduino (pulsador).
- Aprender a declarar variables tipo lista de valores.
- Aprender a declarar una función y llamarla cuando sea necesario.
- Conocer órdenes como: digitalRead.
- Conocer órdenes de control de programa como: If.



Video

Solución:






```
int cadenaleds[]={5,6,7,8};
int pulsador=4;
int tiempo=200;
int n=0;

void setup() {
for(n=0;n<4;n++) {
pinMode (cadenaleds[n],OUTPUT);
}
pinMode (pulsador,INPUT);
}

void flash() {
for (n=0;n<4;n++) {
digitalWrite (cadenaleds[n],HIGH);
delay (tiempo);
digitalWrite (cadenaleds[n],LOW);
delay (tiempo);
}
}

void loop() {
if (digitalRead(pulsador)==HIGH) {
flash ();
}
}
```

Solución 2:

```
int leds[]={5,6,7,8};
int tiempo=200;
int pulsador=4;
int n=0;
int valorpulsador=0;
```

```
void setup(){
  for(n=0;n<4;n++){
    pinMode(leds[n],OUTPUT);
  }
  pinMode(pulsador,INPUT);
  Serial.begin(9600);
}

void monitoriza(){
  Serial.print("El valor del pulsador es ...");
  Serial.println(valorpulsador);
  delay(1000);
}

void secuencia(){
  for(n=0;n<4;n++){
    digitalWrite(leds[n],HIGH);
    delay(tiempo);
    digitalWrite(leds[n],LOW);
    delay(tiempo);
  }
}

void loop(){
  valorpulsador=digitalRead(pulsador);
  monitoriza();
  if (valorpulsador==1){
    secuencia();
  }
}
```

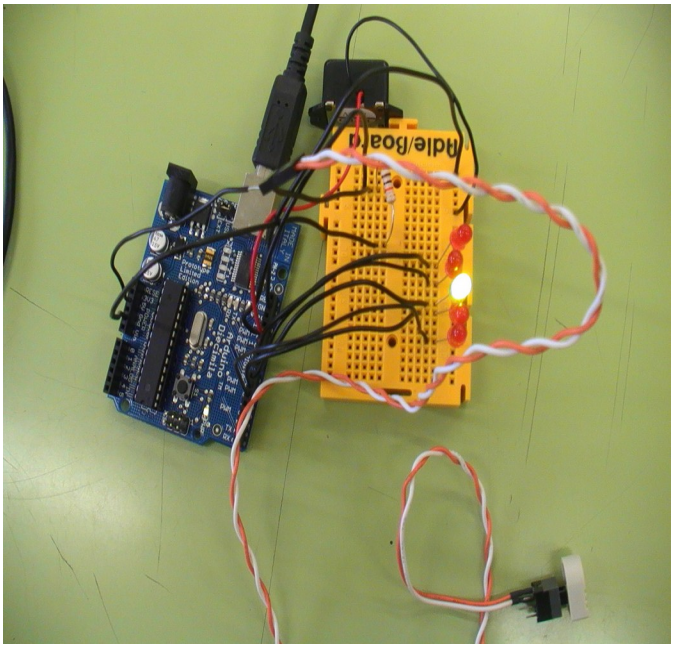
Ruleta de la fortuna.

Se trata de cinco leds que se van encendiendo y apagando formando una secuencia, el jugador debe dar al pulsador cuando el led intermedio se enciende, si acierta funciona un zumbador y la velocidad de la secuencia aumenta.

Los leds deben estar conectados de los pines 5 a 9 (inclusives), el zumbador al pin 10, el pulsador al pin 11. El tiempo inicial entre encendido y encendido de leds debe ser 200 ms, si se acierta se decrementa el tiempo en 20 ms, si el tiempo entre encendidos llegase a 10 ms, se devuelve el tiempo a 200 ms.

Objetivos:

- Repaso de conexión de entrada digital a arduino (pulsador).
- Repaso de variables tipo lista de valores.
- Repaso de declarar una función y llamarla cuando sea necesario.
- Repaso de órdenes como: digitalRead.
- Repaso de órdenes de control de programa como: For, If.



Vídeo

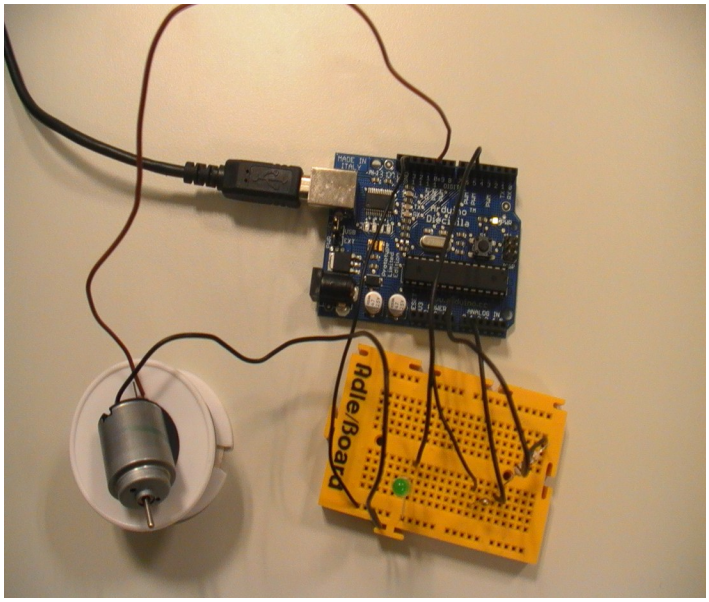
Solución:

```
int leds[]={5,6,7,8,9};
int n=0;
int tiempo=200;
int zumbador=10;
int pulsador=11;

void setup (){
for(n=0;n<5;n++) {
pinMode(leds[n],OUTPUT);
}
pinMode(zumbador,OUTPUT);
pinMode(pulsador,INPUT);
}

void compruebaacierto(){
if(digitalRead(pulsador)==HIGH && n==2) {
digitalWrite(zumbador,HIGH);
delay (1000);
digitalWrite(zumbador,LOW);
tiempo=tiempo-20;
if(tiempo<10){
tiempo=200;}
}
}

void loop () {
for(n=0;n<5;n++) {
digitalWrite(leds[n],HIGH);
delay(tiempo);
compruebaacierto();
digitalWrite(leds[n],LOW);
delay(tiempo);
}
}
```

Termostato.
Se trata de un dispositivo que haga funcionar un motor y un led cuando la temperatura supera cierto umbral. Para ello conectaremos una ntc a la entrada analógica 0, un led al pin 5 y un motor de corriente continua al pin 10. Cuando la temperatura llegue a cierto umbral de voltaje (entre 0 y 1024) que nosotros decidamos, se conectarán a la vez el diodo led y el motor que puede tener unas aspas de ventilador en su eje para enfriar la ntc. Además se deberá visionar el valor de voltaje en la entrada analógica (valor entre 0 y 1024) en una consola en el PC.

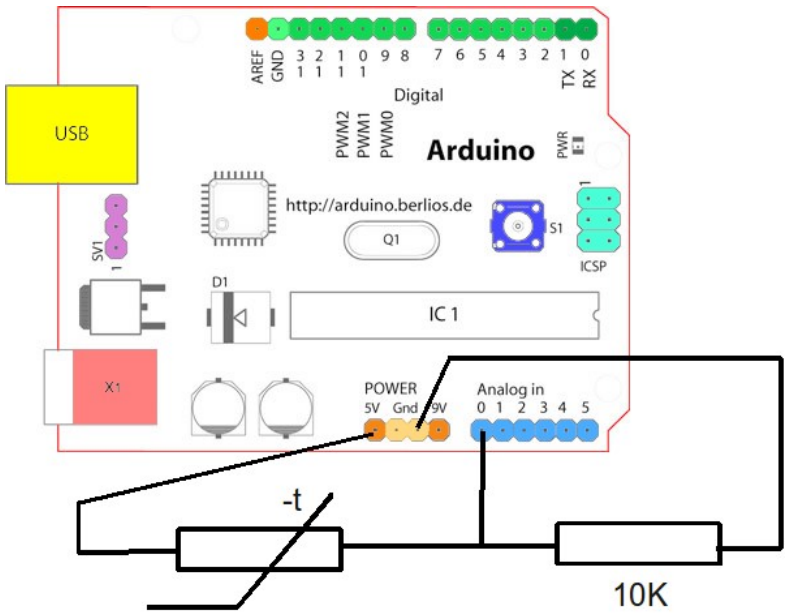
Objetivos:

- Conexión de entrada analógica a arduino (ntc).
- Órdenes como: analogRead.
- Visualizar datos en consola de puerto serie, con órdenes como: Serial.begin, Serial.print.
- Repaso de órdenes de control de programa como: If else.



Vídeo

Solución:



```
int led=5;
int ntc=0;
int motor=10;
int medida=0;
int nivel=700; //variable que guarda el límite de temperatura al que se activa el ventilador

void setup(){
pinMode(led,OUTPUT);
pinMode(motor,OUTPUT);
Serial.begin(9600);
}

void monitoriza(){ //procedimiento que envía al puerto serie, para ser leído en el monitor,
Serial.print("La medida es ...");
Serial.println(medida);
Serial.print();
delay(1000); //para evitar saturar el puerto serie
}

void loop(){
medida=analogRead(ntc);
monitoriza();
if(medida>nivel){ //si la señal del sensor supera el nivel marcado:
digitalWrite(led,HIGH); //se enciende un aviso luminoso
digitalWrite(motor,HIGH); //arranca el motor
}
else{ // si la señal está por debajo del nivel marcado
digitalWrite(led,LOW);
digitalWrite(motor,LOW); // el motor se para
}
}
```

Aumentar y disminuir intensidad luminosa de led (fading).

Se trata aumentar y disminuir la luminosidad de un led usando la capacidad de ofrecer una tensión variable que da una salida analógica. Para ello se conecta un led al pin 11 y se provoca que su luminosidad pase de mínima a máxima, para luego ir de máxima a mínima. Los valores de salidas analógicas van del mínimo 0 al máximo 255.

Objetivos:

- Conexionado de salidas analógicas (power with module pwm).
- Conocer órdenes como analogWrite.

Solución:

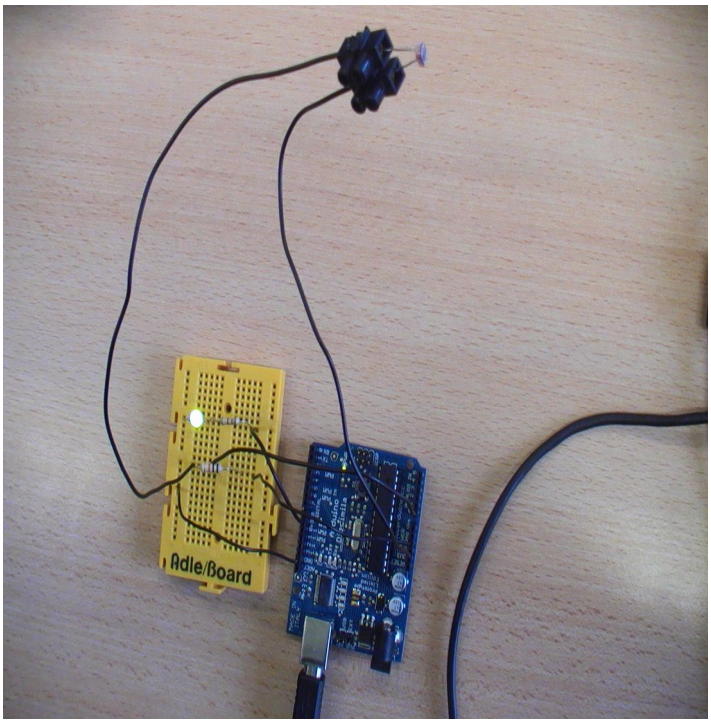
```
int luminosidad = 0;  // variable para asignar la luminosidad al led
int led = 11; // pin del led

void setup()
{
  // en el setup no hay que configurar nada
}

void loop()
{
  for (luminosidad = 0 ; luminosidad <= 255; luminosidad=luminosidad+3) // fade in (from min to max)
  {
    analogWrite(led, luminosidad);  // ilumina el led con el valor asignado a luminosidad (entre 0 y 255)
    delay(30); // espera 30 ms para que se vea el efecto
```

```
 }
 for (luminosidad = 255; luminosidad >=0; luminosidad=luminosidad-3)   // fade out (from max to min)
 {
   analogWrite(led, luminosidad);
   delay(30);
 }
}
```

Luz de led en función de la luz.
Se trata de un dispositivo que haga lucir un led más o menos en función de la luz externa. Para ello conectaremos una ldr a la entrada analógica 0 y un led al pin 9. Cuando la luz se encuentre entre 0 y 512 el led debe colocarse en el nivel de potencia máxima (255), si la luz se encuentra entre valores 512 y 1024 el debe lucir al nivel de potencia 64. Además se deberá visionar el valor de voltaje en la entrada analógica (valor entre 0 y 1024) en una consola en el PC.

Objetivos:

- Repaso conexión de entrada analógica a arduino (ldr).
- Conexionado de salidas analógicas.
- Órdenes como: analogWrite.
- Repaso de visualizar datos en consola de puerto serie, con órdenes como: Serial.begin, Serial.print.
- Repaso de órdenes de control de programa como: If else.



Vídeo

Solución:

```
int led=9;
int ldr=0;
int luz=0;

void setup(){
pinMode(9,OUTPUT);
Serial.begin(9600);
}
```

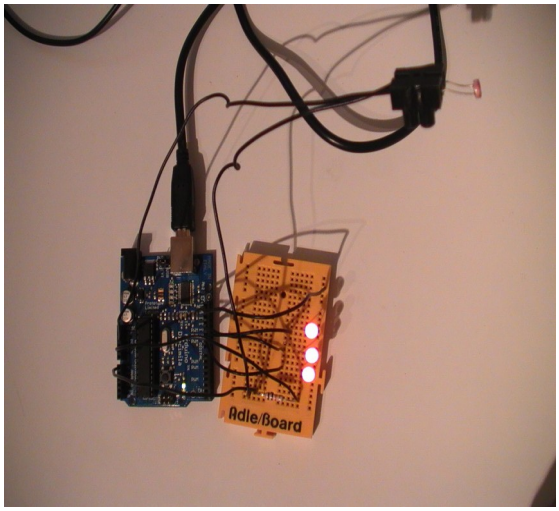
```
void monitoriza(){
Serial.print("El valor de luz es ...");
Serial.println(luz);
delay(1000);
}

void loop(){
luz=analogRead(ldr);
monitoriza();
if(luz<512 && luz>=0){
analogWrite(led,255);
}
if(luz>=512 && luz<=1024) {
analogWrite(led,64);
}
}
```

Luz de leds en función de la luz. Versión 2.

Se trata de un dispositivo que haga lucir tres leds más o menos en función de la luz externa. Para ello conectaremos una ldr a la entrada analógica 0 y los leds a los pines 9,10 y 11. Cuando la luz se encuentre entre 768 y 1023 los leds debe colocarse en el nivel de potencia 64, si la luz se encuentra entre valores 512 y 767 los leds deben lucir al nivel de potencia 127, si la luz se encuentra entre valores 256 y 511 los leds deben lucir al nivel de potencia 191, si la luz se encuentra entre valores 0 y 255 los leds deben lucir al nivel de potencia 255. Además se deberá visionar el valor de voltaje en la entrada analógica (valor entre 0 y 1024) en una consola en el PC.

Objetivos:

- Repaso conexión de entrada analógica a arduino (ldr).
- Repaso conexionado de salidas analógicas.
- Repaso órdenes como: analogWrite.
- Repaso de visualizar datos en consola de puerto serie, con órdenes como: Serial.begin, Serial.print.
- Repaso de órdenes de control de programa como: If else.



Vídeo

Solución:

```
int leds[]={9,10,11};
int tiempo=300;
int ldr=0;
int n=0;
int luz=0;
```

```
void setup(){
for(n=0;n=2;n++) {
pinMode(leds[n],OUTPUT);
}
Serial.begin(9600);
}

void monitoriza() {
Serial.print("El valor de la luz es ...");
Serial.println(luz);
delay(1000);
}

void loop(){
luz=analogRead(ldr);
monitoriza();
if (luz<=1023 && luz>=768) {
for (n=0;n=2;n++) {
analogWrite(leds[n],64);
delay(tiempo);
}
}
if (luz<=767 && luz>=512) {
for (n=0;n=2;n++) {
analogWrite(leds[n],127);
delay(tiempo);
}
}
if (luz<=511 && luz>=256) {
for (n=0;n=2;n++) {
analogWrite(leds[n],191);
delay(tiempo);
}
}
if (luz<=255 && luz>=0) {
for (n=0;n=2;n++) {
analogWrite(leds[n],255);
delay(tiempo);
}
}
}
```

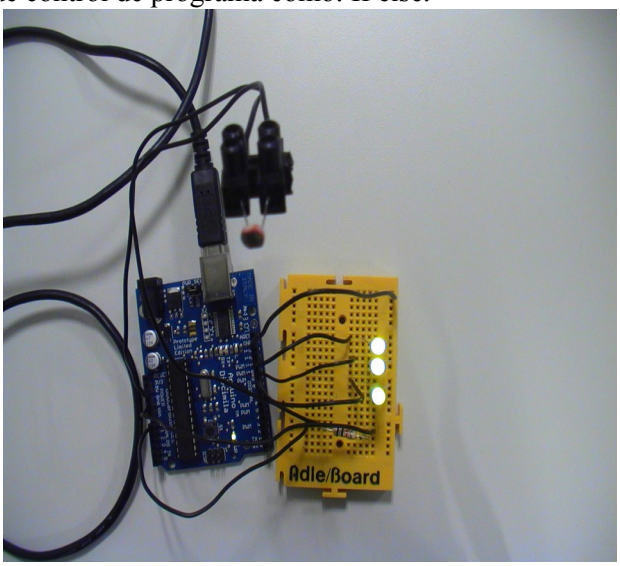
Luz de leds en función de la luz. Versión 3.

Se trata de un dispositivo que haga lucir tres leds más o menos en función de la luz externa. Para ello conectaremos una ldr a la entrada analógica 0 y los leds a los pines 9,10 y 11. El valor de la entrada analógica 0 está comprendido entre 0 y 1024, y el valor de la luminosidad de los leds entre 0 y 255. Los leds deben lucir entre 0 y 255 en función del valor de la entrada analógica 0, siendo su valor inversamente proporcional al valor de la entrada analógica 0 (de 0 a 1024), o sea a más luz menor intensidad luminosa de los leds.

Objetivos:

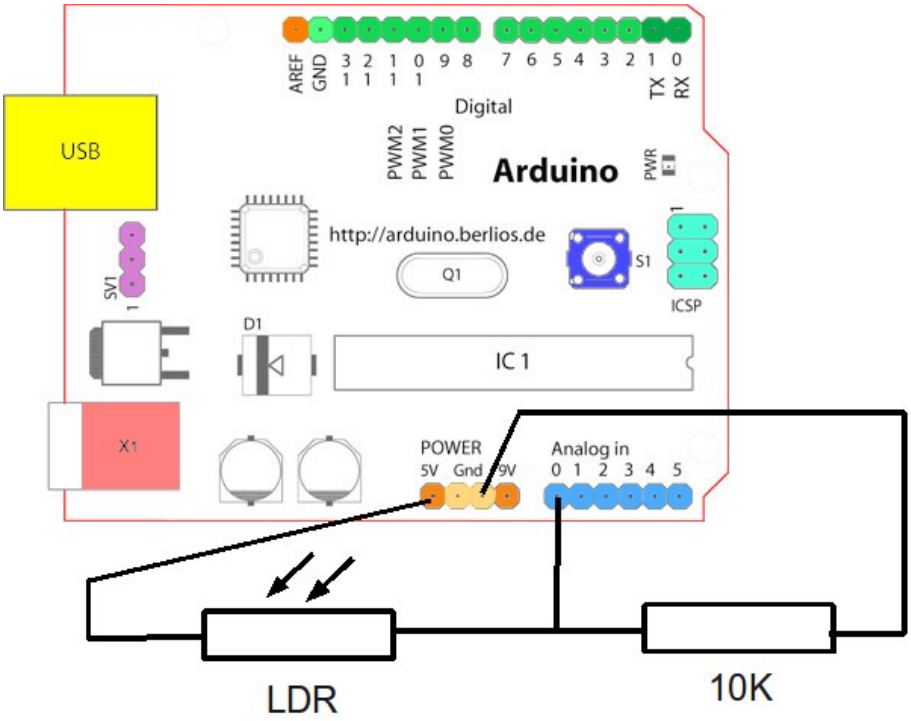
- Repaso conexión de entrada analógica a arduino (ldr).
- Repaso conexionado de salidas analógicas.
- Repaso órdenes como: analogWrite.
- Repaso de visualizar datos en consola de puerto serie, con órdenes como: Serial.begin, Serial.print.

- Repaso de órdenes de control de programa como: If else.



Vídeo

Solución:




```
int ldr=0;
int leds[]={9,10,11};
int n=0;
int medida=0;
int luzled=0;

void setup(){
  for (n=0;n<3;n++) {
    pinMode(leds[n],OUTPUT);
  }
  Serial.begin(9600);
}
```

```
void monitoriza(){
  Serial.print("La medida de luz es ...");
  Serial.println(medida);
  Serial.print("La luz a dar en los leds es ...");
  Serial.println(luzled);
  delay(1000);
}

void loop(){
medida=analogRead(ldr);
luzled=255-(medida/4);
monitoriza();
for (n=0;n<3;n++){
    analogWrite(leds[n],luzled);
    delay(200);
  }
}
```

Termostato con velocidad de motor variable.

Se trata de diseñar un dispositivo que haga lucir un led y funcionar el motor de un ventilador cuando la temperatura llegue a cierto valor umbral (entre 0 y 1024). Para ello conectaremos una ntc a la entrada analógica 0, el led al pin 13 y el motor al pin 9. El motor debe funcionar a cierto nivel de potencia a elegir entre 0 y 255. Además se deberá visionar el valor de voltaje en la entrada analógica (valor entre 0 y 1024) en una consola en el PC.

Objetivos:

- Repaso conexión de entrada analógica a arduino (ntc).
- Repaso conexionado de salidas analógicas.
- Repaso órdenes como: analogWrite.
- Repaso de visualizar datos en consola de puerto serie, con órdenes como: Serial.begin, Serial.print.
- Repaso de órdenes de control de programa como: If else.

Solución:

```
int motor=9;
int led=13;
int ntc=0;
int temperatura=0;

void setup(){
pinMode(led,OUTPUT);
pinMode(motor,OUTPUT);
Serial.begin(9600);
}

void monitoriza(){
Serial.print("El valor de temperatura es ...");
Serial.println(temperatura);
delay(1000);
}

void loop(){
temperatura=analogRead(ntc);
monitoriza();
if(temperatura>530){
digitalWrite(led,HIGH);
```

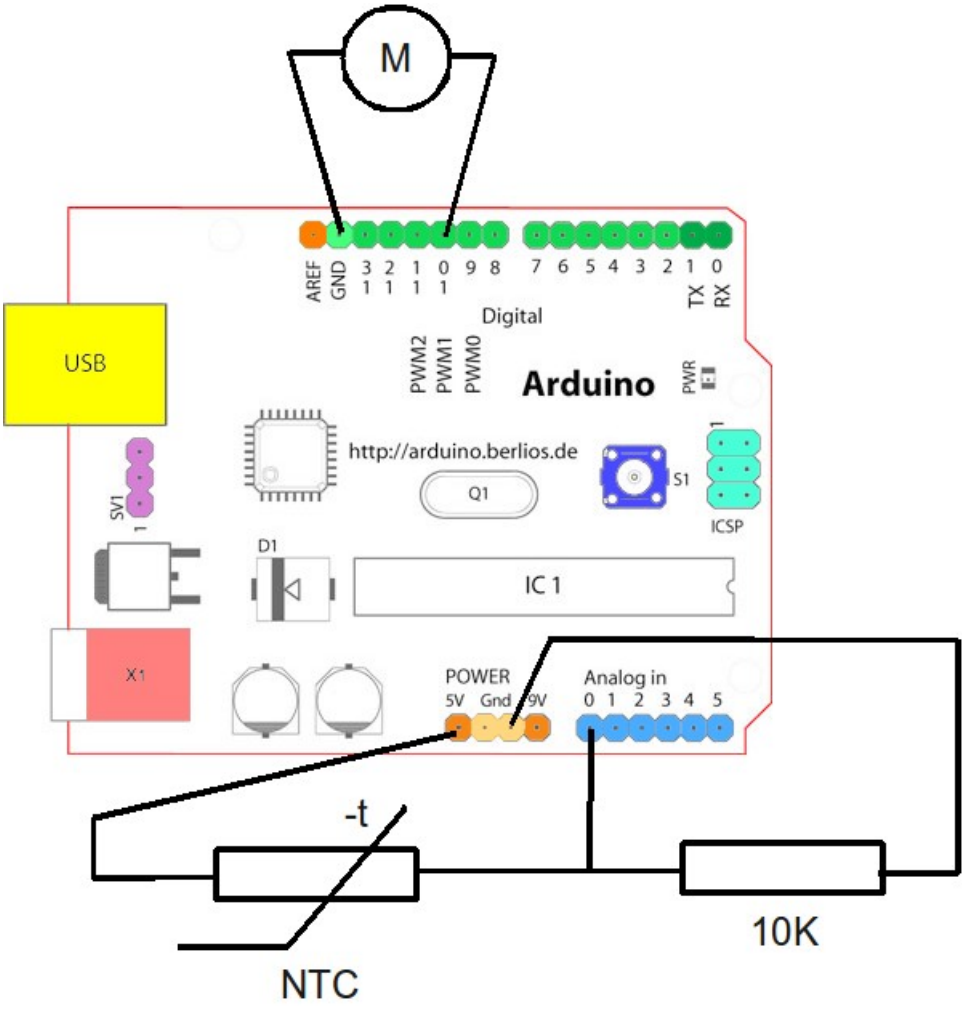
```
analogWrite(motor,200);
}
else {
digitalWrite(led,LOW);
digitalWrite(motor,LOW);
}
}
```

Termostato con velocidad de motor variable (Versión 2).
Se trata de un dispositivo que haga girar un motor más o menos rápido en función de la temperatura. Para ello conectaremos una ntc a la entrada analógica 0 y un led al pin 9 y el motor al pin 10. El valor de la entrada analógica 0 está comprendido entre 0 y 1024, y el valor del la tensión del pin 10 entre 0 y 5 voltios (entre 0 y 255). El motor debe girar a una velocidad entre 0 y 255 en función del valor de la entrada analógica 0, siendo su valor directamente proporcional al valor de la entrada analógica 0 (de 0 a 1024), o sea a más temperatura más velocidad del motor. Además el led del pin 9 debe encenderse.

Objetivos:

- Repaso conexión de entrada analógica a arduino (ntc).
- Repaso conexionado de salidas analógicas.
- Repaso órdenes como: analogWrite.
- Repaso de visualizar datos en consola de puerto serie, con órdenes como: Serial.begin, Serial.print.
- Repaso de órdenes de control de programa como: If else.

Solución:



```
int ntc=0;
int led=13;
int motor=9;
int n=0;
int temperatura=0;
int velocidadmotor=0;

void setup(){
  pinMode(led,OUTPUT);
  pinMode(motor,OUTPUT);
  Serial.begin(9600);
}

void monitoriza(){
  Serial.print("El valor de la temperatura es ...");
  Serial.println(temperatura);
  delay(1000);
}

void loop(){
  temperatura=analogRead(ntc);
  monitoriza();
  velocidadmotor=temperatura/4;
  digitalWrite(led,HIGH);
  analogWrite(motor,velocidadmotor);
}
```

Aumentar luminosidad de led con pulsador (fading).

Se trata de aumentar la luminosidad de un diodo led conectado al pin 11 a través de la activación de un pulsador. El pulsador debe estar conectado al pin 2. Mientras el pulsador está conectado aumenta la luminosidad del led hasta llegar a su valor máximo (255), si el pulsador se desactiva se mantendrá su luminosidad hasta que el valor de luminosidad llegue a su máximo (255) pulsando nuevas veces, si esto ocurre la luminosidad pasará a valor nulo (0).

Objetivos:

- Repaso de conexionado de entradas digitales.
- Repaso de órden digitalRead.
- Repaso de conexionado de salidas analógicas.
- Repaso de órden analogWrite.

Solución:

```
int led = 11; // elegimos el pin del led
int pulsador = 2; // elegimos el pin del pulsador
int x=0; // configuramos la variable para incrementar el valor de luminosidad

void setup()
{
  pinMode(led, OUTPUT); // declaramos led como salida
  pinMode(pulsador, INPUT); // declaramos pulsador como entrada
}
void loop()
{
while (digitalRead(pulsador) == HIGH && x<=255) // chequea si el pulsador está pulsado y x es menor
de 255
```

```
  {
    analogWrite(led,x); // aumenta la luminosidad del led en función del tiempo de activación de pulsador
    delay(20);
    x=x+3;
   }
if (x>255) {
  x=0; // asigna el valor 0 a x
  analogWrite(led, 0); // apaga el led
}
}
```

Control de motor de Corriente Contínua (DC) con Arduino.

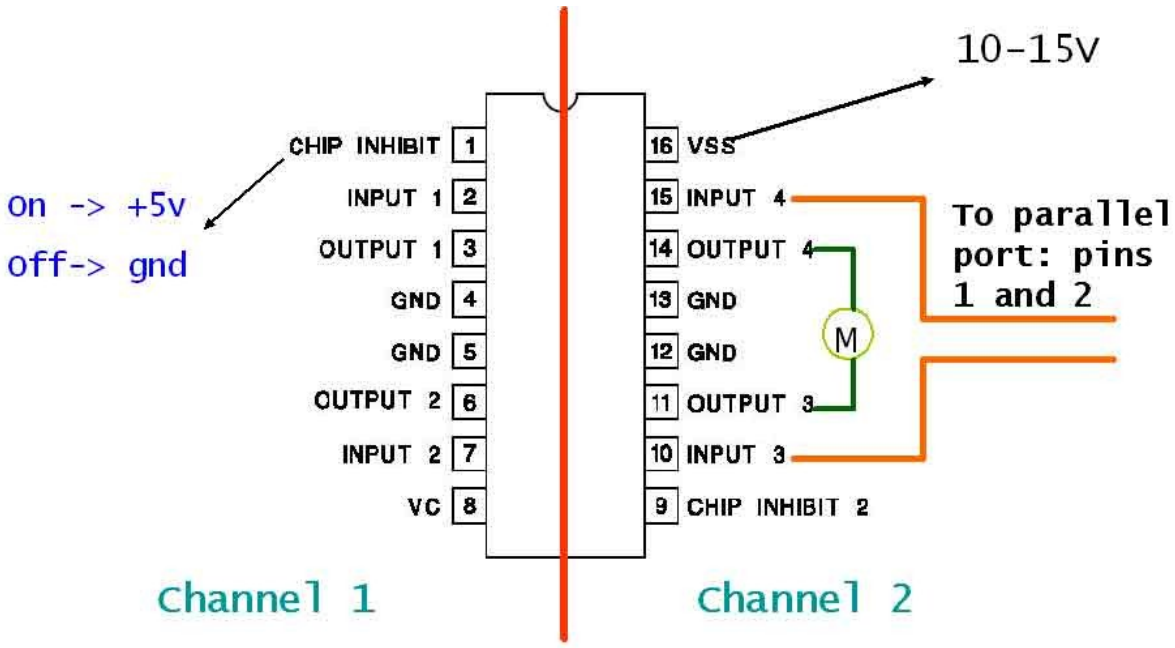
Se trata de controlar el encendido, sentido de giro y potencia de un motor de corriente contínua. El motor debe girar en un sentido al tope de potencia durante cinco segundos, se detendrá durante dos segundos y volverá a girar en sentido contrario a tope de potencia durante cinco segundos deteníendose más tarde. Como segundo paso de la práctica debe realizar la maniobra anterior y más tarde repetirla a mitad de potencia. El motor debe estar conectado a los pines 7 y 8 para su control de giro y usaremos el pin 5 para realizar el control de potencia de arduino usándolo como salida analógica (PWM).
Nota: para esta práctica vamos a utilizar el chip LD293D.

Objetivos:

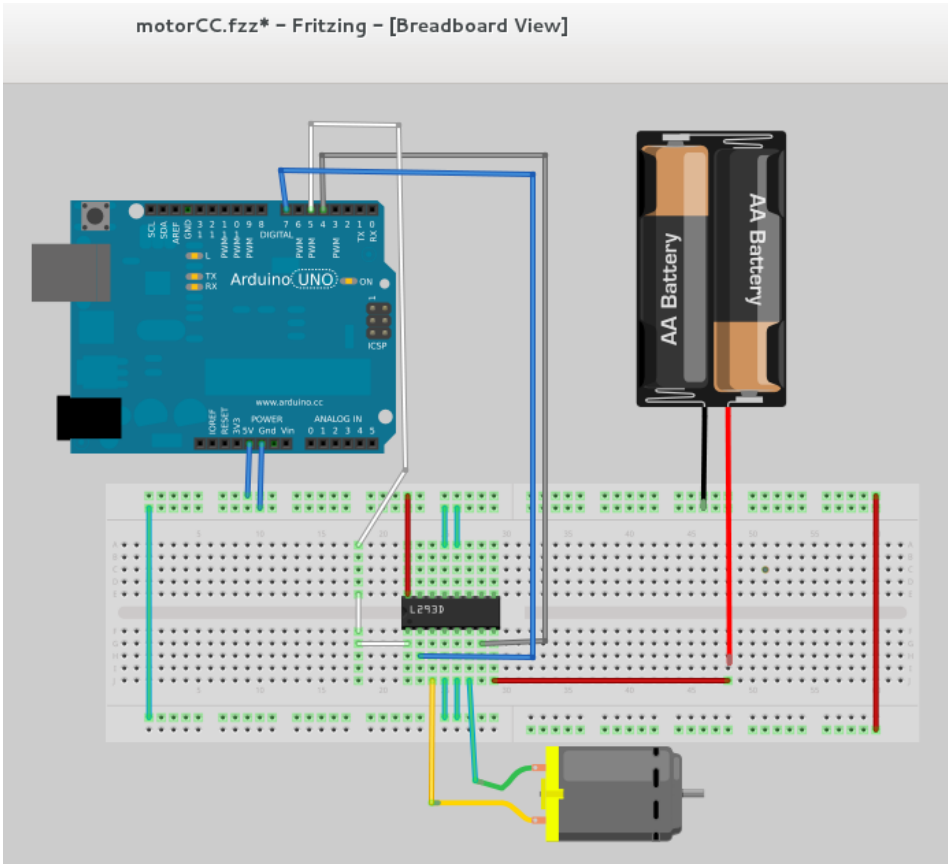
- Conocer chip L293D: conexionado y control mediante pines.
- Repaso a órdenes como analogWrite y digitalWrite.

Antes de realizar el programa vamos a dar información sobre el chip LD293D y realizar algunas consideraciones sobre el mismo.
Si invertimos la polaridad de un motor de continua (motor DC) conseguimos que gire hacia el lado contrario. Para poder hacer esto y para poder suministrar la alimentación adecuada a este tipo de motores, la solución ideal es utilizar lo que se conoce como un driver de motores DC. Este driver es un chip que alimenta los motores a un voltaje diferente (3V en nuestro caso), y mediante 2 entradas digitales controlaremos hacia donde gira el motor.
El chip que utilizaremos (L293D o L293NE) nos permite controlar 2 motores de continua conectando 4 salidas digitales de nuestra placa (2 para cada motor). De estas dos salidas de cada motor si ponemos la primera a HIGH y la segunda a LOW girará hacia un lado y si lo hacemos a la inversa girará hacia el lado contrario.

Descripción Chip L293D/B(puente H):
Es un circuito integrado o chip, que puede ser utilizado para controlar simultáneamente la velocidad y dirección de dos motores de continua (contiene dos puentes H). La diferencia entre el modelo L393D y L293B, es que el primero viene con diodos de protección que evita los daños producidos por los picos de voltaje que puede producir el motor..



Contiene 4 pines digitales (2,7,10, 15) son los que ponen INPUT en el dibujo, para controlar la dirección de los motores, hay dos por cada motor a controlar. Para controlar su sentido de giro tengo que poner un pin a «HIGH» y el otro a «LOW», si pongo los dos a «LOW» el motor se detiene.
Los pines (1,9) admiten como entrada una señal PWM, y se utiliza para controlar la velocidad de los motores con la técnica de modulación de ancho de pulso, salidas analógicas.
Los motores van conectados entre los pines (3, 6) y (11,14), son los que ponen OUTPUT en el gráfico.
Las patillas 8 y 16 son las de alimentación y se tienen que poner ambas entre 10 y 15 voltios a una fuente de alimentación, así como, debemos conectar las patillas nombradas como GND (tierra) a 0 voltios de nuestra fuente de alimentación.

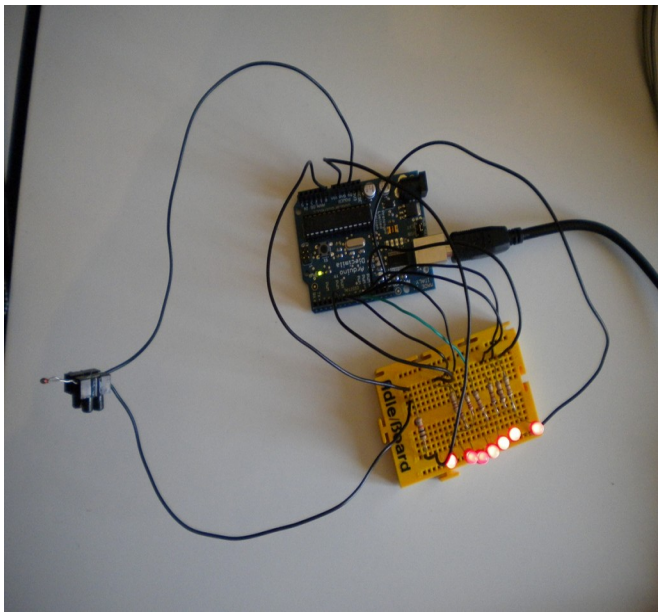


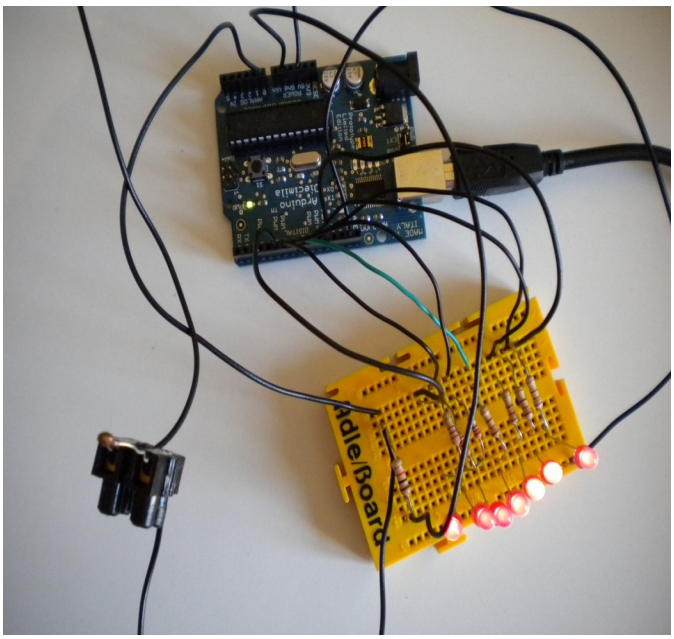
Solución:

```
void setup()
{
  pinMode(7, OUTPUT); // declaramos el pin 7 como salida
  pinMode(8, OUTPUT); // declaramos el pin 8 como salida
}
void secuencia() // función que realiza la secuencia de movimiento del motor
{
digitalWRite (7, HIGH);
digitalWrite (8;LOW);
delay (5000);
digitalWRite (7, LOW);
digitalWrite (8;LOW);
delay (2000);
digitalWRite (7, LOW);
digitalWrite (8;HIGH);
delay (5000);
digitalWRite (7, LOW);
digitalWrite (8;LOW);
delay (2000);
}

void loop()
{
analogWrite (5, 255);
secuencia();
analogWrite (5,128);
secuencia();
}
```

Termómetro de leds.

8 leds lucen o se apagan ejerciendo de escala termométrica. En función de la temperatura lucen más cantidad de leds o menos. También se monitoriza en tiempo real (cada segundo), el valor de la temperatura en grados Celsius.

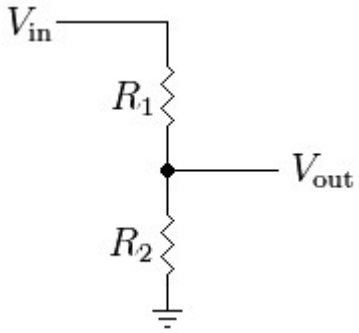




Vídeo

Antes del programa algunas consideraciones:
a) Tras hacer varias medidas sobre la variación de temperatura y resistencia de la ntc, Lorenzo Olmo extrajo la siguiente ecuación empírica, que pone en relación ambas magnitudes de la ntc
$T = -28{,}9 \times lnRntc + 224{,}55$, ayudado de OpenOffice Calc, su horno, su frigorífico y su buen hacer.
b) Hemos conexionado la NTC a las entradas analógicas de la siguiente manera realizando un divisor de tensión.

Teniendo en cuenta que un divisor de tensión genérico tiene el siguiente conexionado,y atiende a la expresión: $Vout = \frac{R2}{R1+R2} \times Vin$



c) Operando en nuestro caso:

$$Vout = \frac{10000}{10000 + Rntc} \times 5$$

$$Vout = analogRead(ntc) \times \frac{1}{1024} \times 5$$

$$temperatura = analogRead(ntc)$$

sustituyendo Vout por su valor:

$$\frac{temperatura \times 5}{1024} = \frac{50000}{10000 + Rntc}$$

$$Rntc = \frac{10240000}{temperatura} - 10000$$

sustituyendo Rntc por su valor:

$$temperatura3(real) = (-28{,}9) \times Ln(\frac{10240000}{temperatura} - 10000) + 224{,}55$$

Solución 1:

```
int zumbador=4; //asigna a la variable zumbador el valor 4
int leds[]={5,6,7,8,9,10,11,12}; //define variable en forma de lista llamada leds con 8 valores
int nleds; //define el número de leds que se encenderán
int ntc=0; //asigna a la variable ntc el valor 0
int temperatura=0; //asigna a la variable temperatura el valor 0
int n=0; //asigna a la variable n el valor 0
int m=0; //asigna a la variable m el valor 0
float temperatura2=0;
float temperatura3=0;

void setup(){ //comienza la configuración
  for (n=0;n<8;n++) {
    pinMode(leds[n],OUTPUT);
  } //hace que los valores de la lista de la variable leds del 0 al 7 (del 5 al 12) lo asigne a los pines y los
declara como de salida
    pinMode(zumbador,OUTPUT); //la variable zumbador (4) es el pin que pone como de salida
  Serial.begin(9600); // hace que comience la comunicación con el puerto serie (pantalla de salida)
}

void monitoriza(){ //función monitoriza sirve para mostrar la temperatura de la NTC en valores desde 0 a
1024
  Serial.print("El valor de temperatura en grados Celsius es ...");
  //Serial.println(temperatura);
  //Serial.println(temperatura2);
  Serial.println(temperatura3);
  delay(1000);
}

void apagaleds(){ // función que sirve para apagar todos los leds
  for (m=0;m<8;m++) {
    digitalWrite(leds[m],LOW);
  }
}
```

```
int variable;
int led;
int n;

void setup() {
for (n=3;n<13;n++) {
  pinMode (n,OUTPUT);
  }
 Serial.begin (9600);
}

void loop() {
variable=analogRead (0);
Serial.println (variable);
led=map(variable,0,1203,3,12);

for (n=3;n<(led+1);n++) {
  digitalWrite (n,HIGH);
  }
```

```
delay (200);
for (n=3;n<(led+1);n++) {
  digitalWrite (n,LOW);
  }
}
```

```
void loop(){ //bloque principal del programa, que se repite hasta el infinito y más alla
  temperatura=analogRead(ntc); // lee el valor entre 0 y 1024 de la entrada analógica analógica 0 (valor de la variable ntc)
  temperatura2=(10240000/temperatura)-10000;
  temperatura3=(-28.9)*log(temperatura2)+224.55;
  monitoriza(); // llama a la función monitoriza
  apagaleds(); // llama a la función apagaleds


if(temperatura3<12){ //si la temperatura es menor a 12 grados, apaga todos los leds
  apagaleds();
  }
  if(temperatura3>12&&temperatura3<=13){ // si la temperatura se encuentra entre 12 y 13 grados enciende el led primero (salida digital 5)
     digitalWrite(leds[0],HIGH);
  }
  if (temperatura3>13&temperatura3<=14) { // si la temperatura se encuentra entre 13 y 14 grados enciende los leds 1 y 2 (salidas digitales 5 y 6)
    digitalWrite(leds[0],HIGH);
    digitalWrite(leds[1],HIGH);
  }
  if (temperatura3>14&&temperatura3<=15) { // si la temperatura se encuentra entre 14 y 15 grados enciende los leds 1,2 y 3 (salidas digitales 5,6 y 7)
    digitalWrite(leds[0],HIGH);
    digitalWrite(leds[1],HIGH);
    digitalWrite(leds[2],HIGH);
  }
  if (temperatura3>15&&temperatura3<=16) { // si la temperatura se encuentra entre 15 y 16 grados enciende los leds 1,2,3 y 4 (salidas digitales 5,6,7 y 8)
    digitalWrite(leds[0],HIGH);
    digitalWrite(leds[1],HIGH);
    digitalWrite(leds[2],HIGH);
    digitalWrite(leds[3],HIGH);
  }
  if (temperatura3>16&&temperatura3<=17) { // si la temperatura se encuentra entre 16 y 17 grados enciende los leds 1,2,3,4 y 5 (salidas digitales 5,6,7,8 y 9)
    digitalWrite(leds[0],HIGH);
    digitalWrite(leds[1],HIGH);
    digitalWrite(leds[2],HIGH);
    digitalWrite(leds[3],HIGH);
    digitalWrite(leds[4],HIGH);
  }
  if (temperatura3>17&&temperatura3<=18) { // si la temperatura se encuentra entre 17 y 18 grados enciende los leds 1,2,3,4,5 y 6 (salidas digitales 5,6,7,8,9 y 10)
    digitalWrite(leds[0],HIGH);
    digitalWrite(leds[1],HIGH);
    digitalWrite(leds[2],HIGH);
    digitalWrite(leds[3],HIGH);
    digitalWrite(leds[4],HIGH);
    digitalWrite(leds[5],HIGH);
```

```
 }
 if (temperatura3>18&&temperatura3<=19) { // si la temperatura se encuentra entre 18 y 19 grados enciende los leds 1,2,3,4,5,6 y 7 (salidas digitales 5,6,7,8,9,10 y 11)
   digitalWrite(leds[0],HIGH);
   digitalWrite(leds[1],HIGH);
   digitalWrite(leds[2],HIGH);
   digitalWrite(leds[3],HIGH);
   digitalWrite(leds[4],HIGH);
   digitalWrite(leds[5],HIGH);
   digitalWrite(leds[6],HIGH);
 }
if (temperatura3>19&&temperatura3<=20) { // si la temperatura se encuentra entre 19 y 20 grados enciende los leds 1,2,3,4,5,6,7 y 8 (salidas digitales 5,6,7,8,9,10,11 y 12)
   digitalWrite(leds[0],HIGH);
   digitalWrite(leds[1],HIGH);
   digitalWrite(leds[2],HIGH);
   digitalWrite(leds[3],HIGH);
   digitalWrite(leds[4],HIGH);
   digitalWrite(leds[5],HIGH);
   digitalWrite(leds[6],HIGH);
   digitalWrite(leds[7],HIGH);
 }

 if (temperatura3>20) { // si la temperatura es mayor de 20 grados enciende todos los leds 1,2,3,4,5,6,7 y 8 (salidas digitales 5,6,7,8,9,10,11 y 12)
   digitalWrite(leds[0],HIGH);
   digitalWrite(leds[1],HIGH);
   digitalWrite(leds[2],HIGH);
   digitalWrite(leds[3],HIGH);
   digitalWrite(leds[4],HIGH);
   digitalWrite(leds[5],HIGH);
   digitalWrite(leds[6],HIGH);
   digitalWrite(leds[7],HIGH);
 }
 delay(1000); // espera un segundo
 }
```

Grabadora de ritmos.

Se trata de una caja dónde se dan unos golpes con una cadencia, y posteriormente el zumbador piezoeléctrico del sistema repite los golpes dados.

Solución:

```
int led=13;
int piezoelectrico=1;
int medida=0;
int nivel = 5; //valor de la señal analógica que establecemos como límite de la sensibilidad del sensor
int zumbador=10;
int numerogolpes = 0; //variable que lleva la cuenta del nº de golpes
int tiempo = 0; //variable que lleva la cuenta del tiempo entre cada par de golpes
int n=0;
int cadenatiempos[]= {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,}; //cadena en la que se guardarán los tiempos entre golpes
int cadenagolpes[]={2,3,4,5,6,7,8,9,10,11,12,13,14,15}; //cadena en la que llevamos la cuenta de los golpes
```

```
void setup() {
pinMode(led,OUTPUT);
pinMode(zumbador,OUTPUT);
Serial.begin(9600); //inicia la comunicación con el puerto serie del ordenador y
} //establece la velocidad de transferencia

void guardatiempos(){ //procedimiento que guarda los tiempos entre golpes en una cadena
for (n=0 ;n<14 ;n++){
if (numerogolpes==cadenagolpes[n]){ //compara el valor del contador de golpes con los valores de la
cadena
cadenatiempos[n] =tiempo; //para asociar el primer tiempo con 2 golpes, el segundo con 3 golpes y así
sucesivamente
} //a través de la posición que ocupan en las cadenas
}
}

void sifinrepro(){ //procedimiento que reproduce, con una se cuencia de pitidos la secuencia de golpes
delay(500); //las tres primeras líneas producen el primer pitido
digitalWrite(zumbador, HIGH);
delay(50);
digitalWrite(zumbador, LOW);

for(n=0 ;n<(numerogolpes-1); n++){ //iteración que produce los pitidos a partir del segundo
delay(cadenatiempos[n]*100); //incorporando los tiempos que se han guardado, el multiplicar por 100 es
un ajuste empírico
digitalWrite(zumbador, HIGH); //que tiene en cuenta los tiempos que pasan mientras se ejecuta el
programa
delay(50);
digitalWrite(zumbador, LOW);
} //cuando termina la reproducción:
numerogolpes=0; //se reinician las variables para poder comenzar con otra secuencia
tiempo=0;
}

void loop() {

medida = analogRead(piezoelectrico); //actualizamos el valor de la señal del sensor

if (medida >= nivel) { //si la señal supera el límite
digitalWrite(led,HIGH); //se enciende el LED
Serial.print("TOC! "); //se envían caracteres por el puerto serie
Serial.println(medida);
delay(100);
numerogolpes = numerogolpes+1; //aumenta en uno el contador de golpes
guardatiempos(); //guardamos el tiempo en su cadena
tiempo = 0; //cada vez que se guarda un tiempo el contador se pone a 0 para empezar a contar el siguiente
delay(100); //espera para no obtener una lectura múltiple de un solo golpe
digitalWrite(led,LOW);
} //si no hay una señal de intensidad suficiente:

delay(100); //espera para no saturar el puerto serie
tiempo = tiempo +1; //aumenta el contador de tiempo
if((numerogolpes>0)&&(tiempo >=30)){ //si pasa mucho tiempo después de al menos un golpe:
digitalWrite(led,HIGH); //se considera que se ha terminado y comienza la reproducción
sifinrepro();
digitalWrite(led,LOW);
}
```

```
}
```

Fuentes de conocimiento utilizadas:

- http://www.arduino.cc/
- http://www.arduino.cc/es/
- http://arduino.cc/es/Secundaria/Secundaria
- http://arduino-ubuntu.blogspot.com/
- http://visualp5.net/visualp5net-taller-arduino.html
- https://sites.google.com/a/divinechildhighschool.org/electronics/Home/Arduino-Lessons
- Libro “Computación Física en Secundaria” de *Marco Antonio Rodríguez Fernández* Libro publicado bajo licencia CreativeCommons-Attribution-Share Alike 2.0: