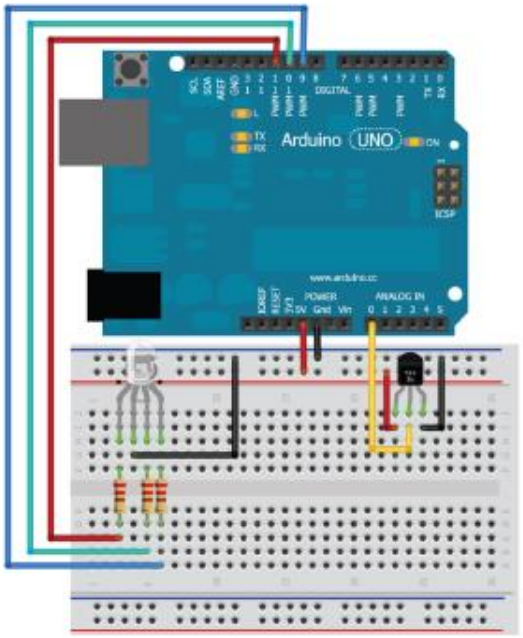
Practica 03 – Arduino, DAC y sensores

David Subires Parra

Cuestiones

Ejercicio 1

Implementa, usando un montaje como el que aparece, un sistema de alerta de temperatura, usando tres leds, uno que indique que la temperatura es muy baja, otro normal y otro muy alta.



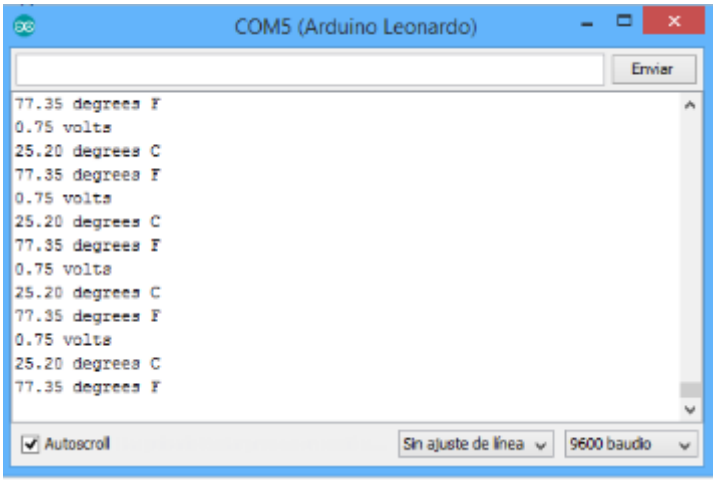
Nota: usa esta formula Temperatura(grados centigrados) x 10 = voltaje (mV) – 500

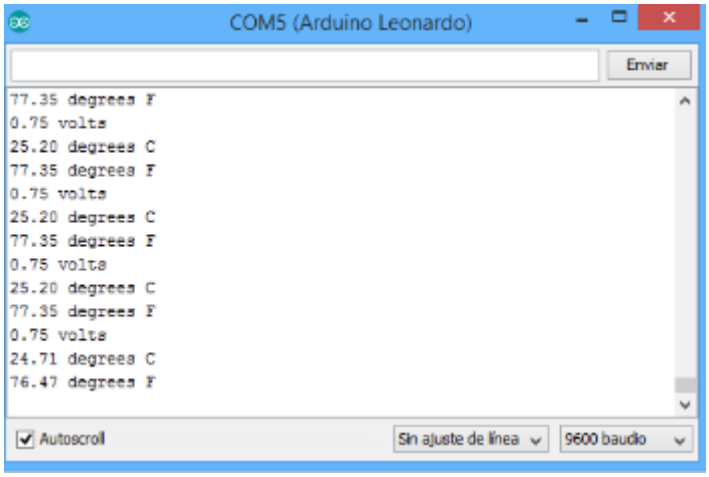
He realizado este ejercicio con el código que se muestra a continuación. Si la temperatura es menor a 5ªC, enciende el led conectado al pin 10. Si es mayor que 25ºC, se enciende el led conectado al pin 11, y en cualquier otro caso (cuando la temperatura esté entre 5-25ºC), se enciende el led conectado al pin 13.

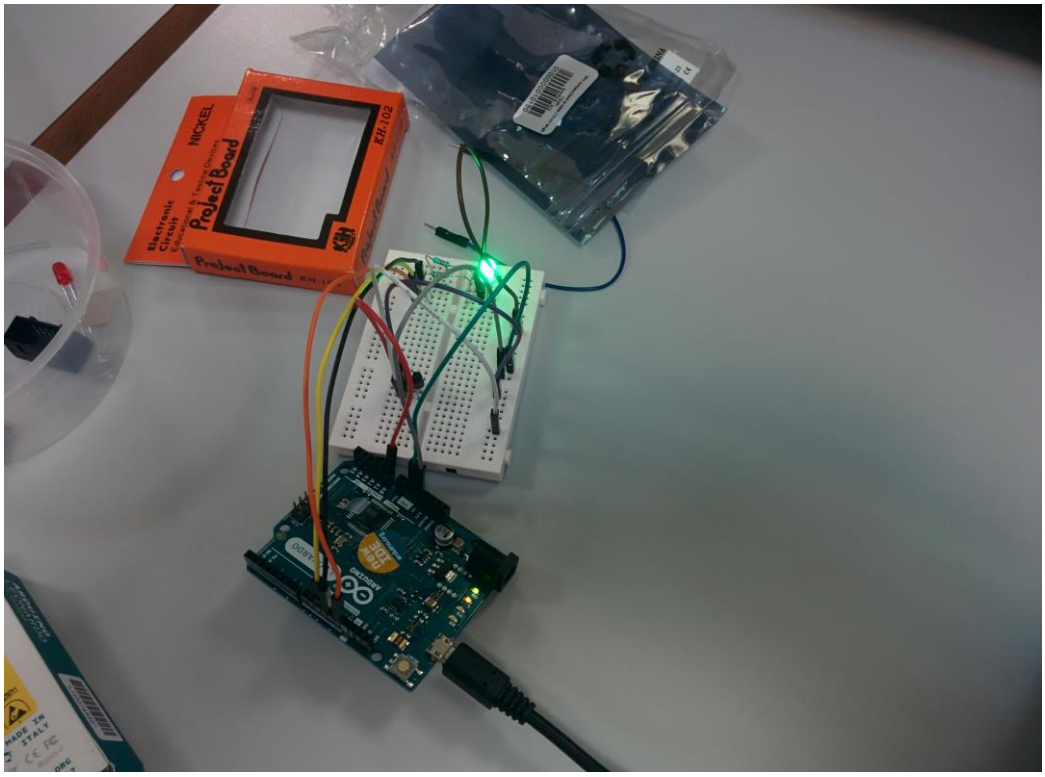
```
int sensorPin = 0;

void setup() {
  // put your setup code here, to run once:
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(13, OUTPUT);

  Serial.begin(9600);
}

void loop() {

 //getting the voltage reading from the temperature sensor
 int reading = analogRead(sensorPin);

 // converting that reading to voltage, for 3.3v arduino use 3.3
 float voltage = reading * 5.0;
 voltage /= 1024.0;

 // print out the voltage
 Serial.print(voltage); Serial.println(" volts");

 // now print out the temperature
 float temperatureC = (voltage - 0.5) * 100 ;  //converting from 10 mv per degree wit
500 mV offset
                                        //to degrees ((voltage - 500mV) times 100)
```

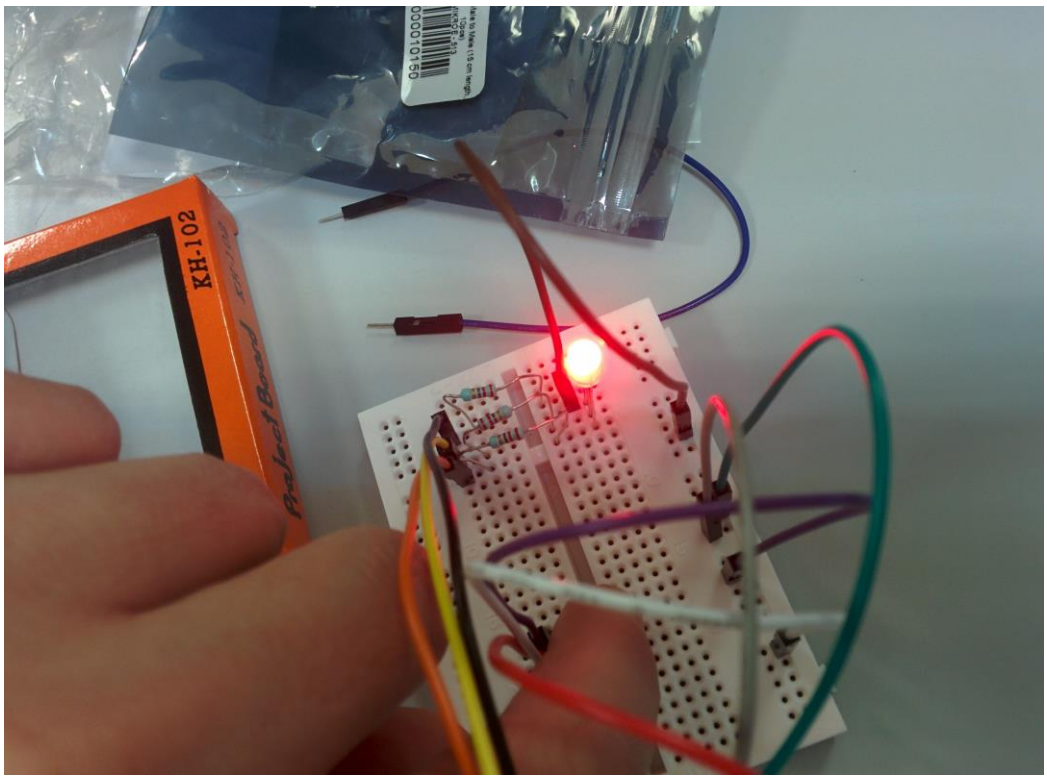
```
 Serial.print(temperatureC); Serial.println(" degrees C");

 // now convert to Fahrenheit
 float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
 Serial.print(temperatureF); Serial.println(" degrees F");


  //Dependiendo de la temperatura encendemos un led u otro
  if(temperatureC < 5){
    digitalWrite(10,HIGH);
    digitalWrite(11, LOW);
    digitalWrite(13, LOW);
  }else if(temperatureC > 25){
    digitalWrite(10, LOW);
    digitalWrite(11, HIGH);
    digitalWrite(13, LOW);
  }else{
    digitalWrite(10, LOW);
    digitalWrite(11, LOW);
    digitalWrite(13, HIGH);
  }

  //waiting a second
  delay(1000);
}
```

A continuación, se muestran capturas de ejecución y del montaje:





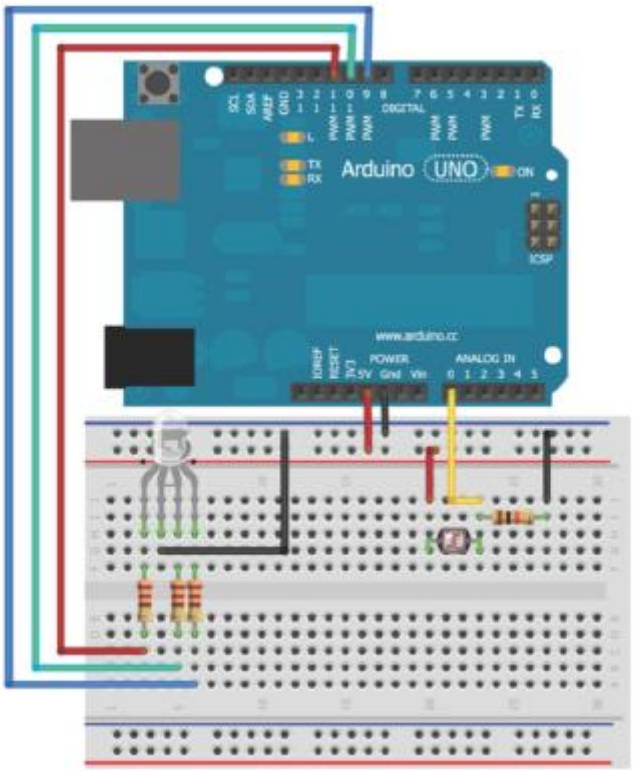




En las fotos se puede apreciar que al poner el dedo sobre el sensor de temperatura, esta asciende y se enciende el led correspondiente.

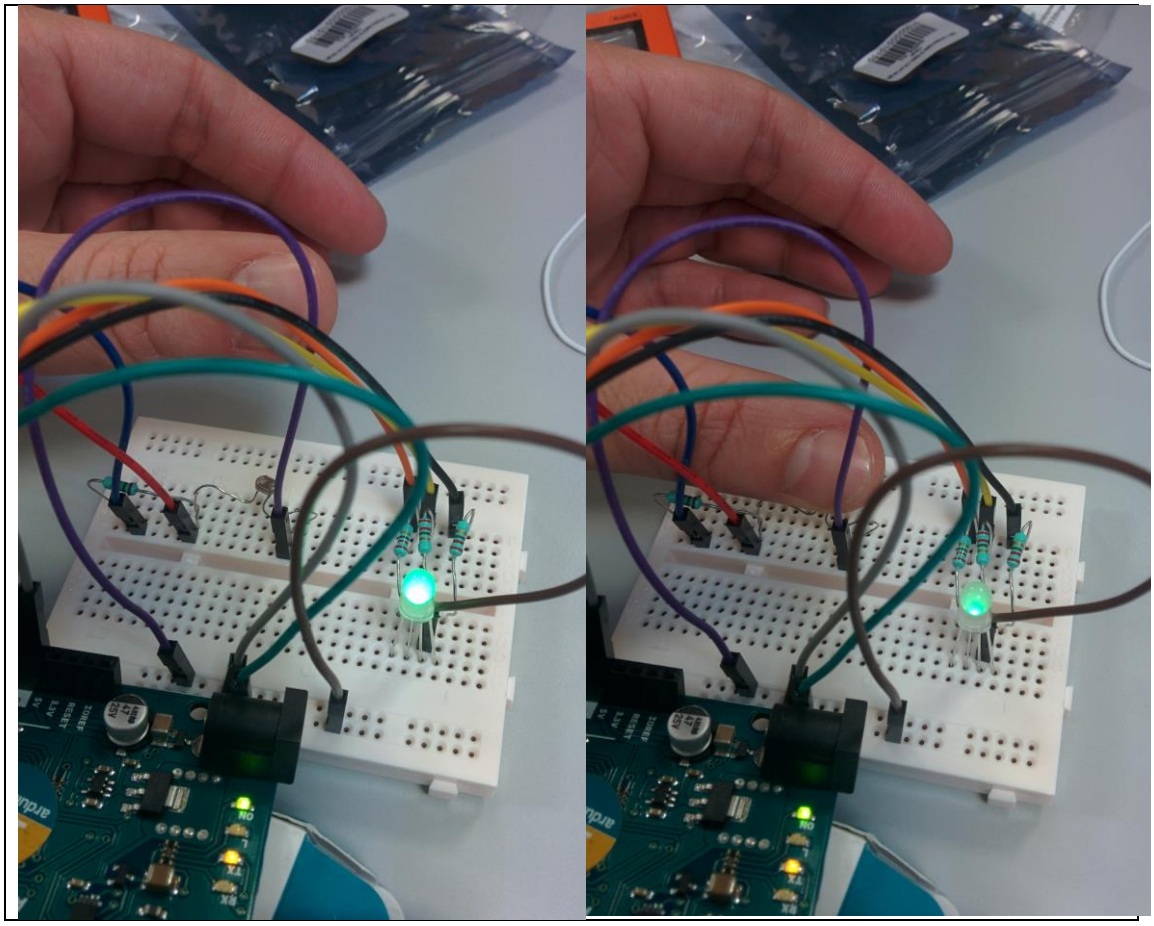
Ejercicio 2

Foto resistor, a menor luz ambiente mayor luz debe emitir el LED. Debes montar un sistema que haga que el LED brille con más intensidad cuando se esté a oscuras y menor intensidad cuando haya luz ambiental.

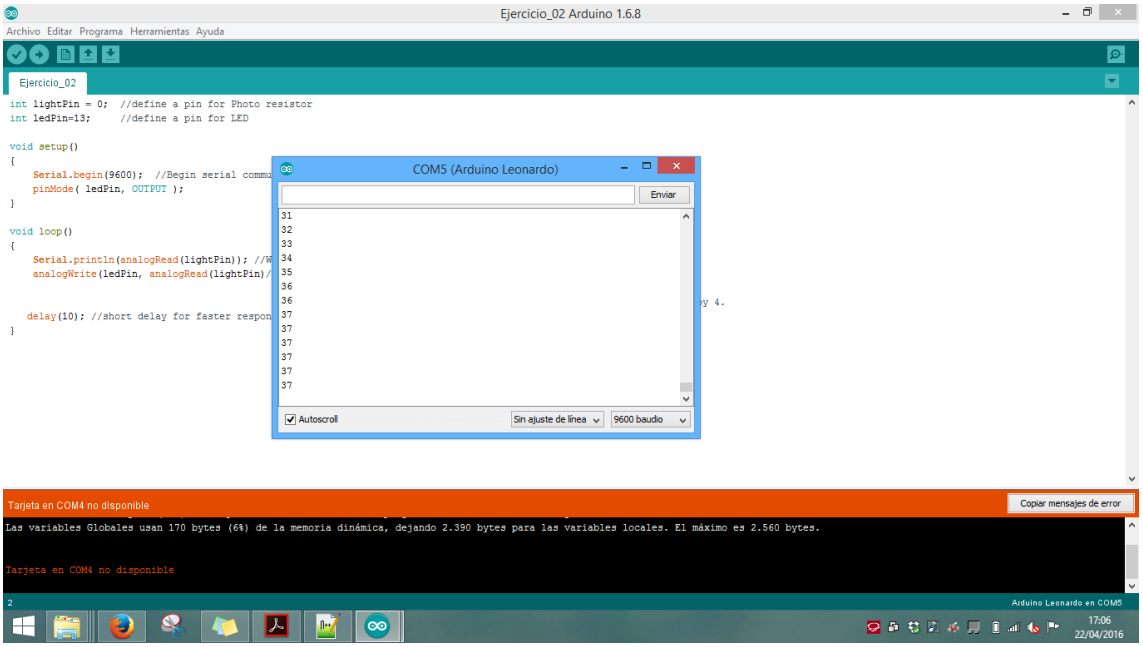


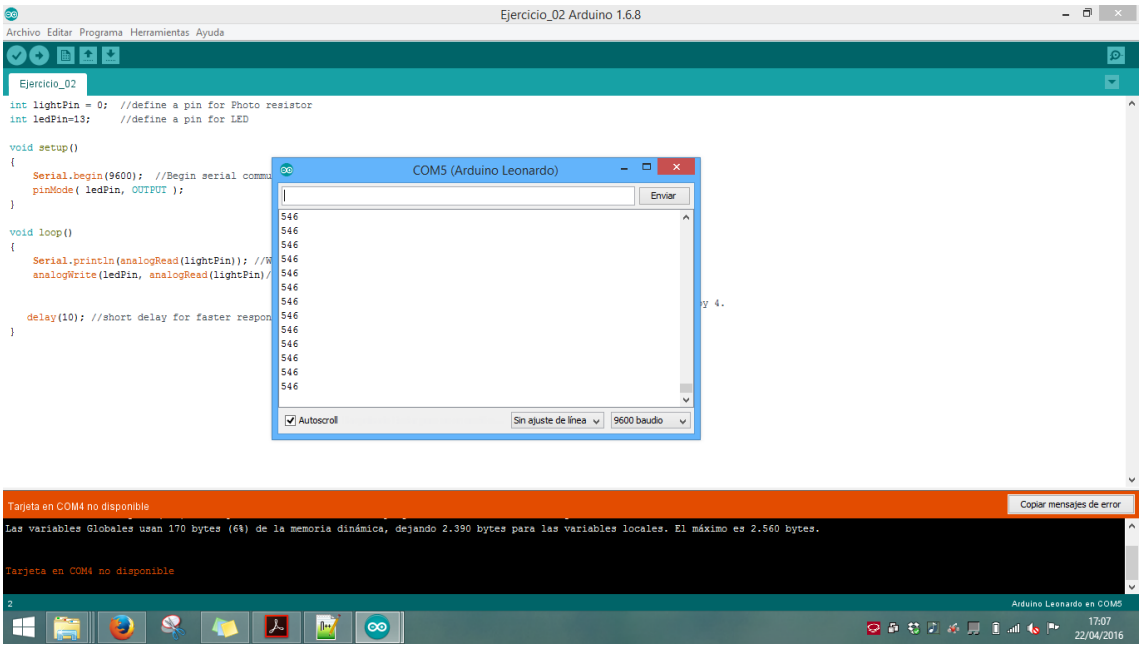

```
int lightPin = 0;  //define a pin for Photo resistor
int ledPin=13;     //define a pin for LED

void setup()
{
   Serial.begin(9600);  //Begin serial communcation
   pinMode( ledPin, OUTPUT );
}

void loop()
{
   Serial.println(analogRead(lightPin)); //Write the value of the photoresistor to the
serial monitor.
   analogWrite(ledPin, analogRead(lightPin)/4);  //send the value to the ledPin.
Depending on value of resistor
                              //you have  to divide the value. for example,
                              //with a 10k resistor divide the value by 2, for 100k resistor
divide by 4.
  delay(10); //short delay for faster response to light.
}
```

Mediante el sencillo código anterior, podemos hacer uso de la foto resistencia y del led en arduino. El led se iluminará en función de la cantidad de luz que detecte la foto resistencia.
En las dos fotos que se muestran a continuación, podemos ver el montaje realizado y como tras tapar la foto resistencia con el dedo la intensidad del led disminuye notablemente.



Además, por consola podíamos comprobar el valor que devolvía el foto resistor como podemos ver a continuación

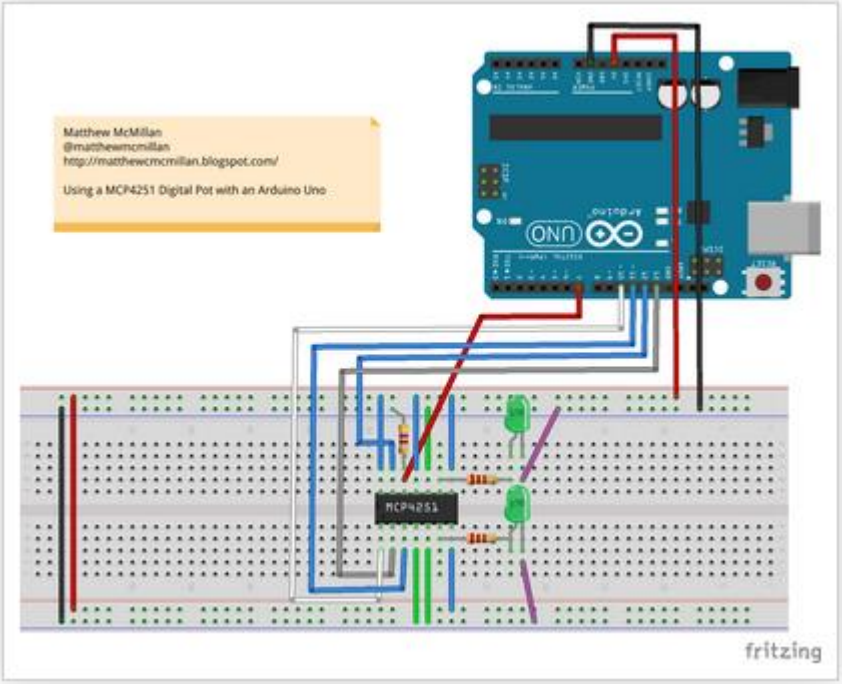

Ejercicio_02 Arduino 1.6.8
Archivo Editar Programa Herramientas Ayuda
Ejercicio_02
int lightPin = 0; //define a pin for Photo resistor
int ledPin=13;   //define a pin for LED
void setup()
{
Serial.begin(9600); //Begin serial commu
pinMode(ledPin, OUTPUT);
}
void loop()
{
Serial.println(analogRead(lightPin)); //W
analogWrite(ledPin, analogRead(lightPin)/
delay(10); //short delay for faster respon
}
COM5 (Arduino Leonardo)
Enviar
31
32
33
34
35
36
36
37
37
37
37
37
Autoscroll
Sin ajuste de línea
9600 baudio
Tarjeta en COM4 no disponible
Copiar mensajes de error
Las variables Globales usan 170 bytes (6%) de la memoria dinámica, dejando 2.390 bytes para las variables locales. El máximo es 2.560 bytes.
Tarjeta en COM4 no disponible
Arduino Leonardo en COM5
17:06
22/04/2016


Ejercicio_02 Arduino 1.6.8
Archivo Editar Programa Herramientas Ayuda
Ejercicio_02
int lightPin = 0; //define a pin for Photo resistor
int ledPin=13;   //define a pin for LED
void setup()
{
Serial.begin(9600); //Begin serial commu
pinMode(ledPin, OUTPUT);
}
void loop()
{
Serial.println(analogRead(lightPin)); //W
analogWrite(ledPin, analogRead(lightPin)/
delay(10); //short delay for faster respon
}
COM5 (Arduino Leonardo)
Enviar
546
546
546
546
546
546
546
546
546
546
546
546
Autoscroll
Sin ajuste de línea
9600 baudio
Tarjeta en COM4 no disponible
Copiar mensajes de error
Las variables Globales usan 170 bytes (6%) de la memoria dinámica, dejando 2.390 bytes para las variables locales. El máximo es 2.560 bytes.
Tarjeta en COM4 no disponible
Arduino Leonardo en COM5
17:07
22/04/2016

Ejercicio Extra

Usando la función analogRead(): Haz una primera prueba usando el potenciómetro (un potenciómetro permite variar el voltaje en función del giro, aunque en este caso al ser un potenciómetro digital, variará en función del código que se ejecute). Coge el potenciómetro, haz el montaje y lee por el puerto serie:

Para realizar este ejercicio, he utilizado en montaje que se nos proporcionó en el foro (http://matthewcmcmillan.blogspot.com.es/2014/03/arduino-using-digital-potentiometers.html)



La única diferencia entre este montaje y el que finalmente he realizado, es la conexión de los pines 11, 12 y 13, que en mi caso, por ser otra versión de arduino (arduino leonardo), estos pines los he conectado a las patillas ISCP:

La conexión de los pines 11, 12 y 13 es la siguiente:

* SDI - to digital pin 11 (MOSI pin)
* SDO - to digital pin 12 (MISO pin)
* CLK - to digital pin 13 (SCK pin)

Se puede ver el montaje en funcionamiento a través de la siguiente url:
https://www.dropbox.com/s/p6zp6qegu0kvzba/VID_20160425_022752.mp4?dl=0

El código empleado en este ejercicio es el siguiente:

```
/*
  Matthew McMillan
  @matthewmcmillan
  http://matthewcmcmillan.blogspot.com
  Created 12 Mar 2014
  Digital Pot Control (MCP4251)

  This example controls a Microchip MCP4251 digital potentiometer.
  The MCP4251 has 2 potentiometer channels. Each channel's pins are labeled:
    A - connect this to voltage
    W - this is the pot's wiper, which changes when you set it
    B - connect this to ground.
  The MCP4251 also has Terminal Control Registers (TCON) which allow you to
  individually connect and disconnect the A,W,B terminals which can be useful
  for reducing power usage or motor controls.

  A value of 255 is no resistance
  A value of 128 is approximately 5k ohms
  A value of 0 is 10k ohm resistance

  The MCP4251 is SPI-compatible. To command it you send two bytes,
```

one with the memory address and one with the value to set at that address.

The MCP4251 has few different memory addresses for wipers and tcon (Terminal Control)

*Wiper 0 write

*Wiper 0 read

*Wiper 1 write

*Wiper 1 read

*TCON write

*TCON read

The circuit:

* All A pins of MCP4251 connected to +5V

* All B pins of MCP4251 connected to ground

* An LED and a 220-ohm resistor in series connected from each W pin to ground

* VSS - to GND

* VDD - to +5v

* SHDN - to digital pin 7 and a 4.7k pull down resistor

* CS - to digital pin 10 (SS pin)

* SDI - to digital pin 11 (MOSI pin)

* SDO - to digital pin 12 (MISO pin)

* CLK - to digital pin 13 (SCK pin)

created 12 Mar 2014

Thanks to Heather Dewey-Hagborg and Tom Igoe for their original tutorials

*/

// inslude the SPI library:

```
#include <SPI.h>

// set pin 10 as the slave select for the digital pot:
const int slaveSelectPin = 10;
const int shutdownPin = 7;
const int wiper0writeAddr = B00000000;
const int wiper1writeAddr = B00010000;
const int tconwriteAddr = B01000000;
const int tcon_0off_1on = B11110000;
const int tcon_0on_1off = B00001111;
const int tcon_0off_1off = B00000000;
const int tcon_0on_1on = B11111111;

void setup() {
  // set the slaveSelectPin as an output:
  pinMode (slaveSelectPin, OUTPUT);
  // set the shutdownPin as an output:
  pinMode (shutdownPin, OUTPUT);
  // start with all the pots shutdown
  digitalWrite(shutdownPin,LOW);
  // initialize SPI:
  SPI.begin();
}

// This loop adjusts the brightness of two LEDs and
// uses the TCON registers to indvidually disconnect
// the wipers which turns off the LEDs.
void loop() {
  digitalWrite(shutdownPin,HIGH); //Turn off shutdown
```

```
  delay(1000);
  digitalPotWrite(wiper0writeAddr, 200); // Set wiper 0 to 200
  digitalPotWrite(wiper1writeAddr, 200); // Set wiper 1 to 200
  delay(1000);
  digitalPotWrite(tconwriteAddr, tcon_0off_1on); // Disconnect wiper 0 with TCON register
  delay(1000);
  digitalPotWrite(tconwriteAddr, tcon_0off_1off); // Disconnect both wipers with TCON register
  digitalPotWrite(wiper0writeAddr, 100); //Set wiper 0 to 255
  delay(1000);
  digitalPotWrite(tconwriteAddr, tcon_0on_1on); // Connect both wipers with TCON register
}

// This function takes care of sending SPI data to the pot.
void digitalPotWrite(int address, int value) {
  // take the SS pin low to select the chip:
  digitalWrite(slaveSelectPin,LOW);
  //  send in the address and value via SPI:
  SPI.transfer(address);
  SPI.transfer(value);
  // take the SS pin high to de-select the chip:
  digitalWrite(slaveSelectPin,HIGH);
}
```