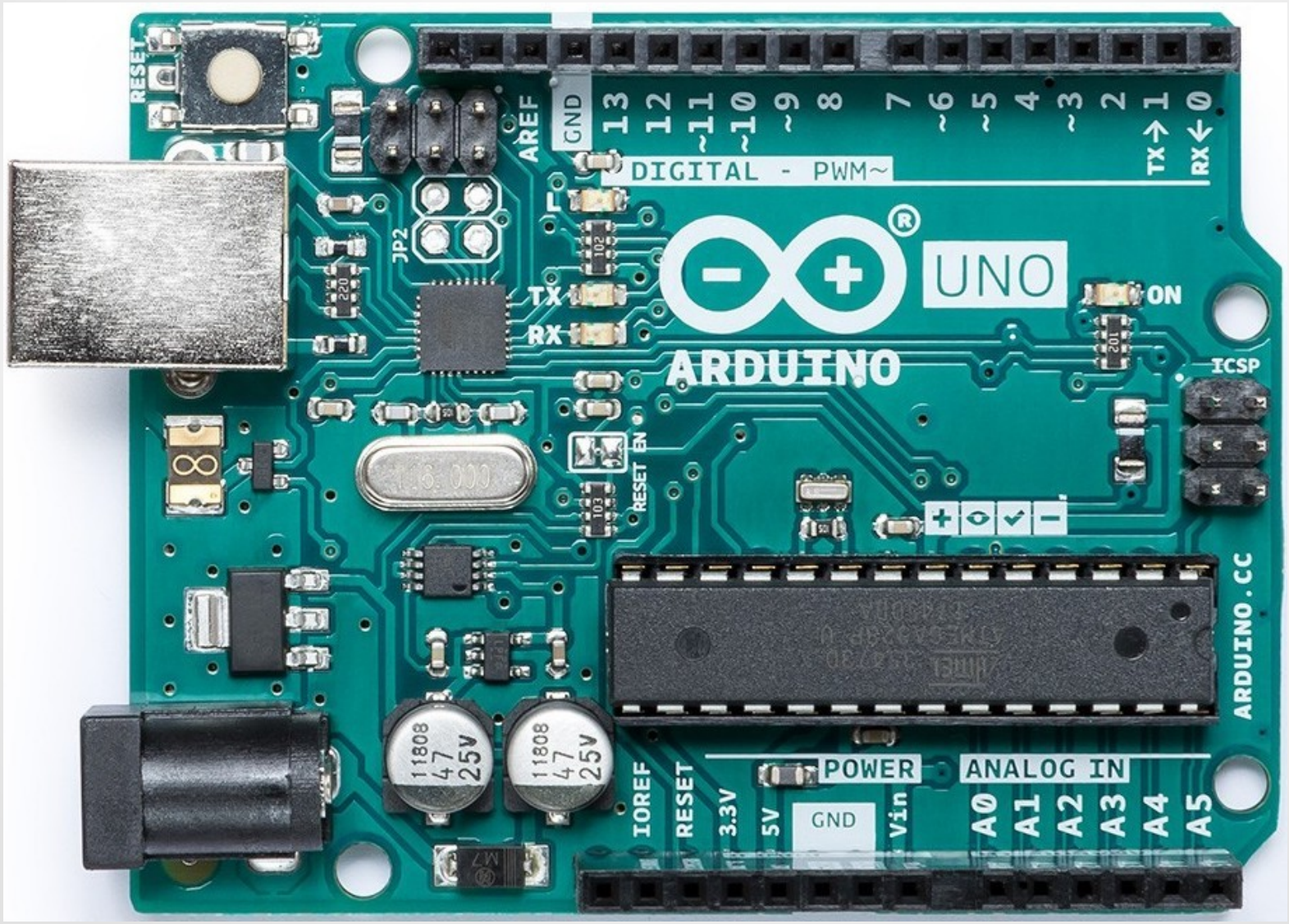
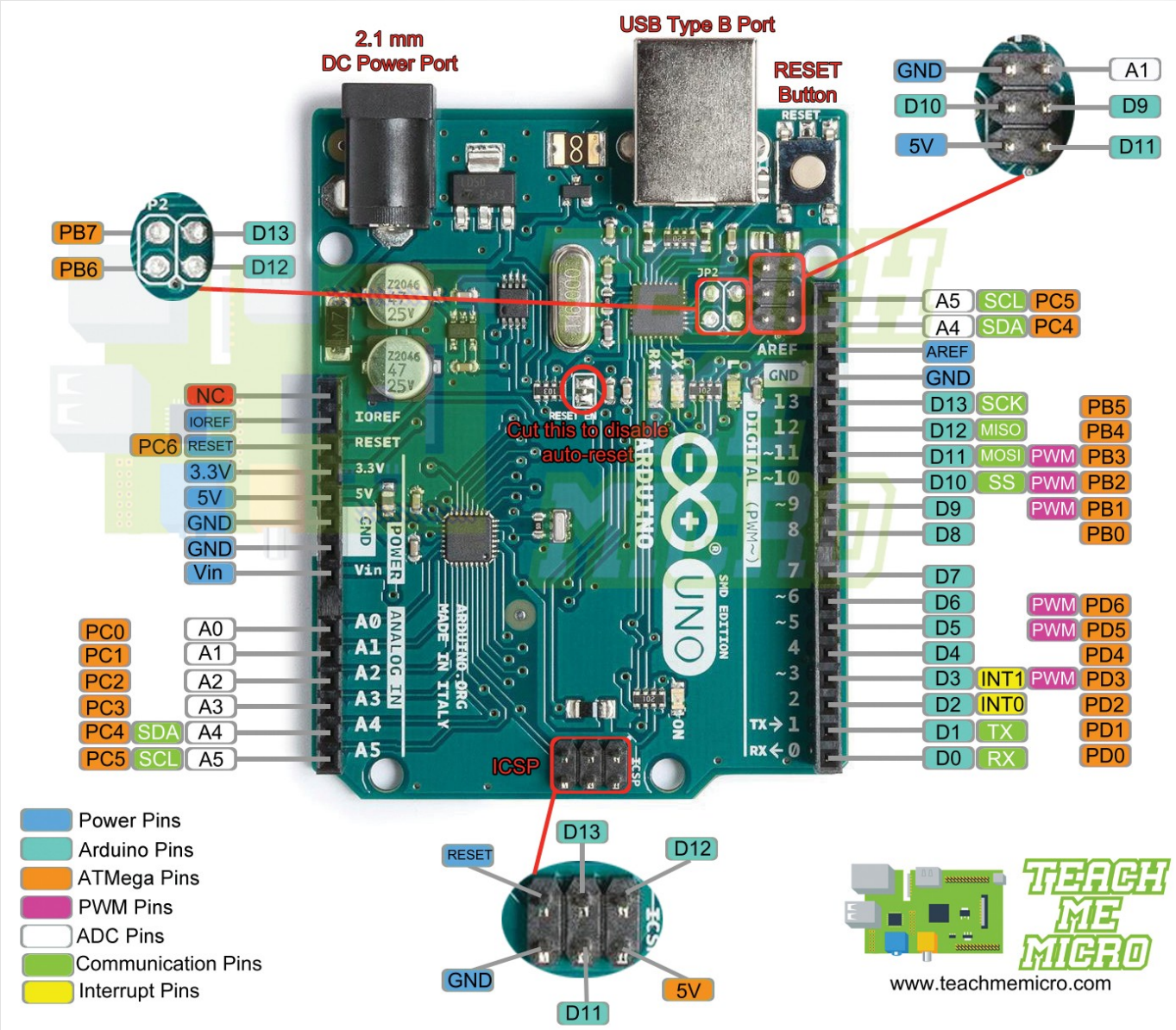
Entradas analóxicas



Víctor . Álvarez 201911

Introdución – pinout Arduino



E/S

Dixitais:

- Pins 0 a 13
  0 a 5V, 20 mA
  Low: 0 a 2V
  High: 3 a 5V

Entradas Analóxicas

- Pins A0 – A5
  0 a 5V, 20 mA prec 1024
  0 a 3V, 50 mA prec 1024

Saídas PWM

- ~ Pins 3, 5, 6, 9, 10, 11
  0 a 5 V, 20 mA prec 256

Comunic. Serie TX/RX

- Pins 0 e 1

ICSP

- 6 pins para comunicarse directamente co proc. Atmega328
- 6 pins para programar o USB

Entradas analóxicas – Lectura de LDR

- As entradas analóxicas son realmente entradas PWM, con 1024 niveis (a diferencia das saídas que eran de 256 niveis), é dicir un valor de tensión de entre 0 e 5 V na entrada terá que ser mapeado a un valor enteiro entre 0 e 1023.
- As entradas analóxicas non é preciso declaralas no setup(), aínda que para lelas precisaremos facer uso de:

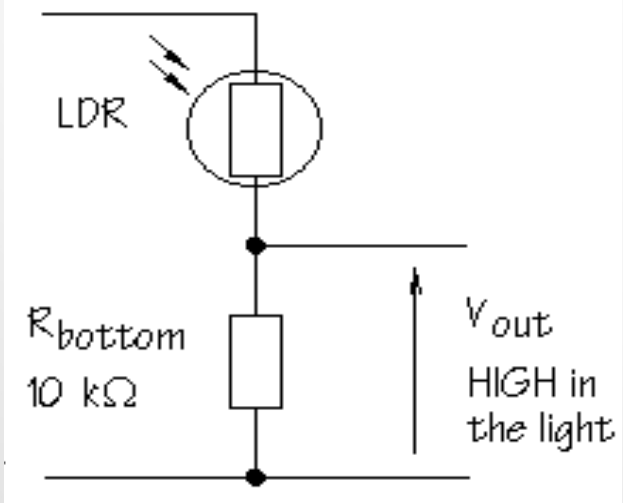
```
analogRead(pinLDR);
```

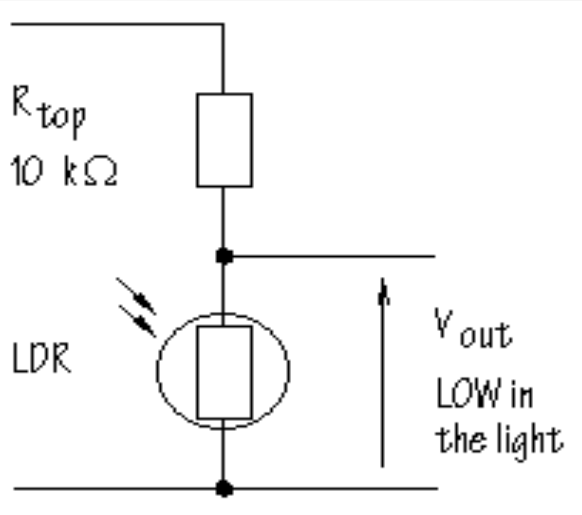
- Para recibir entradas analóxicas precisaremos dun sensor analóxico con valores no rango de 0 a 5 V, como LDRs, NTCs, potenciómetros, sensores de son e ultrasón, humidade, etc.
- Imos empezar usando resistencias LDR como entradas analóxicas e regulando o brillo dun LED en función da luz ambiental.

Víctor . Álvarez 201911

Entradas analóxicas – Lectura de LDR

- Unha LDR (Light-Dependent Resistor) é unha resistencia variable segundo a intensidade de luz: canto maior é a luz incidente, menor é a resistencia. Valores típicos son 1 MΩ en oscuridade total e entre 50 e 100 Ω baixo luz brillante.
- O xeito de conectar a LDR é empregando un divisor de tensión como o da figura inferior. Coa resistencia de pull down (esquerda) conseguimos que conforme aumente a luz incidente, a tensión de saída se aproxime aos 5 V de entrada e a 0 V coa oscuridade. Coa resistencia de pull up (dereita) é exactamente ao revés: conforme aumenta a luz incidente, diminúe a tensión de entrada ao Arduino.

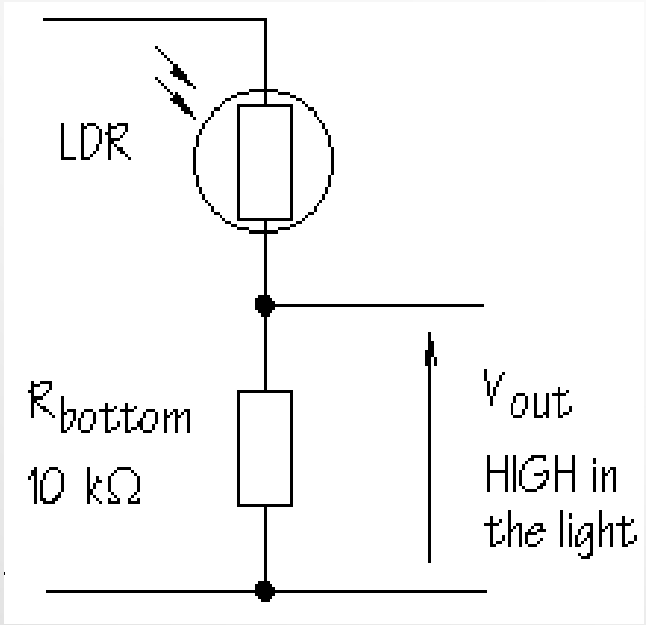


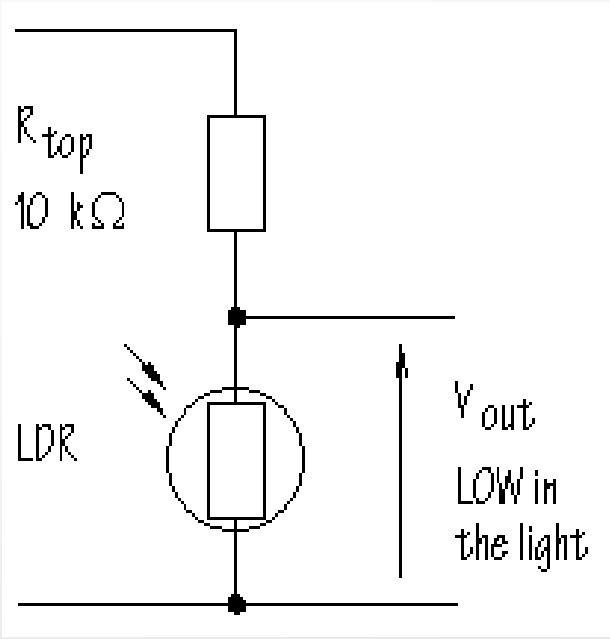


Víctor . Álvarez 201911

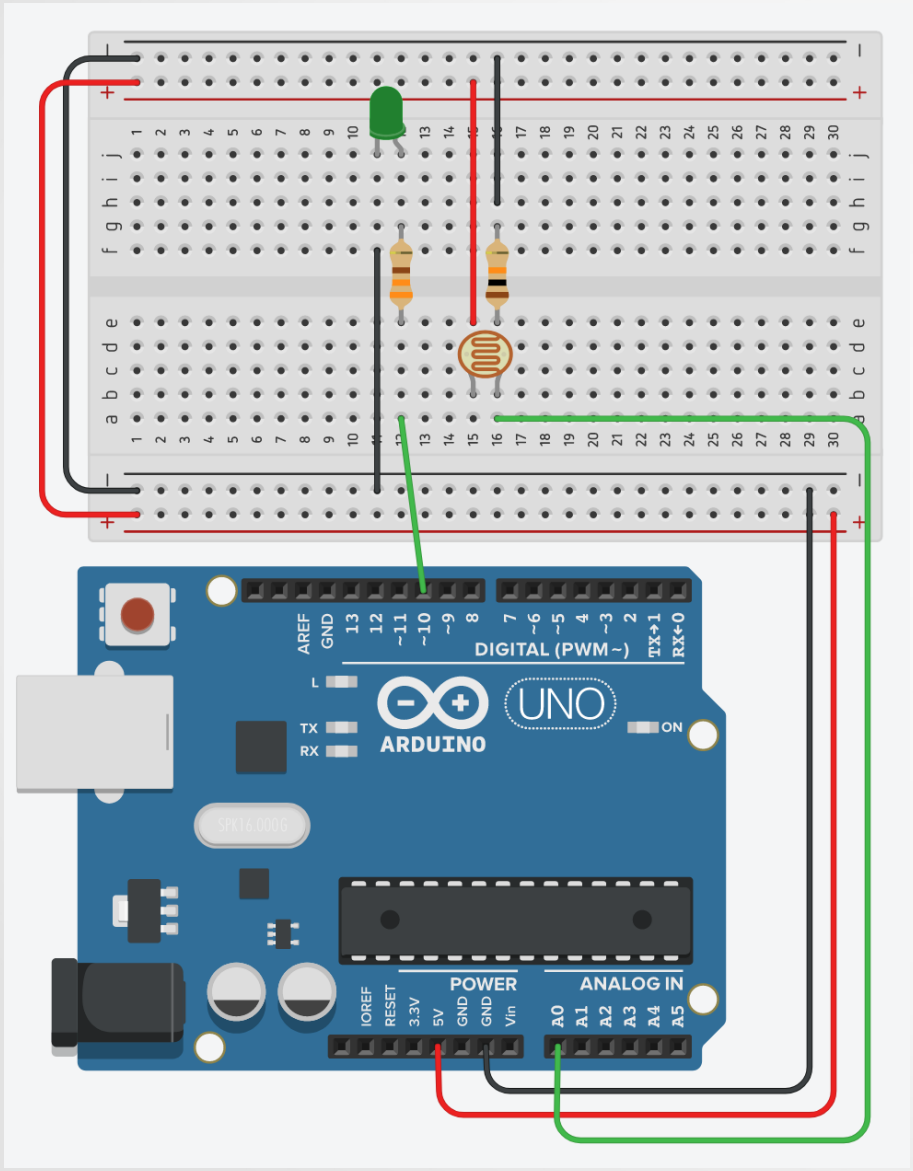
Entradas analóxicas – Lectura de LDR

- Realiza unha montaxe na que unha LDR controle o brillo dun LED, de xeito que a medida que diminúa a luz ambiental, aumente o brillo do LED.
- Emprega unha resistencia de pull down en primeiro lugar e proba despois cunha de pull up.





Entradas analóxicas – Lectura de LDR



LDR.controla.LED | Arduino 1.8.10

LDR.controla.LED

```
/*
 * LED modifica o brillo
 * conforme varía a luz
 * ambiente.
 *
 */


#define VERDE 10
#define LDR A0

int tempo = 100;

void setup() {

}

void loop() {
  int valor = analogRead(LDR);
  valor /= 4;
  analogWrite(VERDE, valor);
  delay(tempo);
}
```

Compilação Terminada

```
O rascunho usa 1090 bytes (3%) do espaço de armazenamento do
Variáveis globais usam 9 bytes (0%) de memória dinâmica, res
```

2 Arduino/Genuino Uno em /dev/cu.wchusbserial1420

Víctor . Álvarez 201911

Entradas analóxicas – Lectura de LDR

- No sketch anterior, por que razón se divide entre 4 o valor da entrada analóxica.
- Que problemas encontras ao facer esta montaxe? Son debidas á programación?
- É posible que o LED estea totalmente apagado? É posible que mostre o brillo máximo?
- Como se pode conseguir que con independencia dos valores de tensión, o LED percorra todo o posible rango de valores?

Entradas analóxicas – Lectura de LDR

- A razón de que se divida entre 4 é que as entradas analóxicas teñen 1024 posibles niveis de precisión, mentres que as saídas analóxicas só teñen 256.
- Ao dividir entre 4 o que facemos é ‘mapear’ ou interpolar os valores de entrada aos de saída.
- Un problema que nos encontramos é que por causa do divisor de tensión, nunca teremos 0 V na entrada, nin tampouco 5 V. Isto fai que se pasamos directamente os valores lidos, o LED nunca se apague de todo, nin tome o valor de maior brillo.
- Para solucionar isto, deberíamos medir na placa de probas, cal é a tensión cando a LDR non recibe luz e cal a tensión cando a luz ambiental sexa máxima.
- Con estes valores, e considerando p.ex. a montaxe con resistancia pull down, cando a luz ambiental é máixima, temos que mapear ao valor 1023 e cando é mínima ao valor 0 (máximo e mínimos valores de tensión).
- Por outra banda, necesitamos que o LED dea o maior brillo coa oscuridade e se apague cando haxa luz ambiental suficiente.

?????????? Que facemos ?????????

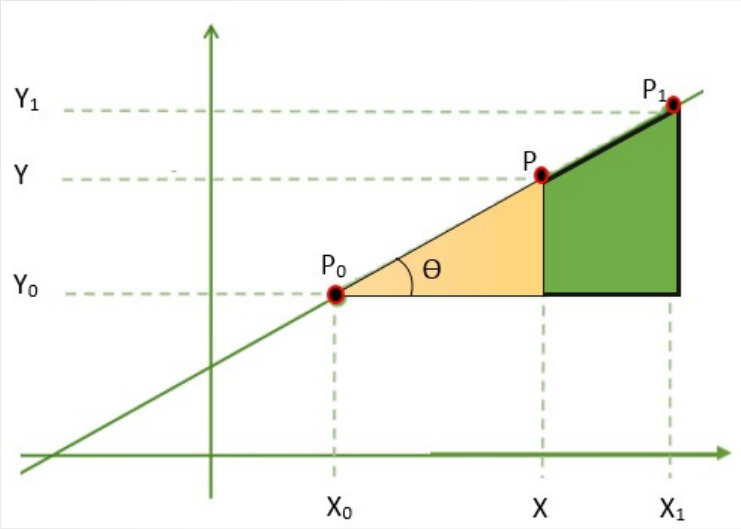
Víctor . Álvarez 201911

Entradas analóxicas – Lectura de LDR

- Temos que pensar que o anterior son dous problemas enlazados:
  - por un lado a interpolación de valores, e
  - por outro o axuste de valores ás nosas necesidades
- Nestes casos compensa pensar de forma xeralizada e tentar acadar solución que se poidan empregar noutros problemas parecidos.
- Imos programar unha función que, dadas dúas escalas (eixos X e Y), nos permita interpolar dunha á outra.

Víctor . Álvarez 201911

Entradas analóxicas – Lectura de LDR



- Temos dúas variables X e Y, que no noso caso son: (i) o nivel PWM de entrada lido do divisor de tensión (1024 valores) e (ii) o nivel PWM de saída (256 valores) para o LED (X e Y, p.ex).
- Por semellanza de triángulos:

$$\frac{y-y_0}{x-x_0}=\frac{y_1-y_0}{x_1-x_0}\Rightarrow y=\frac{y_1-y_0}{x_1-x_0}\cdot\left(x-x_0\right)+y_0$$

- Por seguir co exemplo numérico, podemos supoñer que o valor de entrada 0 PWM, corresponde co 0 PWM de saída e que o 1023 PWM de entrada corresponde co 255 PWM de saída
- Deste xeito a expresión anterior transfórmase en:

$$saidaLED=\frac{255-0}{1023-0}\cdot\left(x-0\right)+0\Rightarrow$$

$$\Rightarrow saidaLED=0.25\cdot x$$

Víctor . Álvarez 201911

Entradas analóxicas – Lectura de LDR

- Chegamos á mesma conclusión, aínda que de xeito máis xeral.
- No noso caso, os límites non son tan nítidos debido ao divisor de tensión na LDR.
- Como non imos estar facendo estas contas de cada vez, programaremos unha función que as faga por nos, de maneira que:
  - debe ter como entradas os límites de ambos eixos (catro variables) e unha máis que é a lectura que nós facemos no divisor,
  - debe dar como saída un valor numérico, que é o resultado da interpolación.
- Teremos que pensar cales son os tipos das variables que temos que declarar, tanto nas de entrada como na de saída.

Víctor . Álvarez 201911

Entradas analóxicas – Lectura de LDR

- Polo que levamos dito, a función debe traballar con números enteiros, polo que usaremos ‘int’ para as variables de entrada e float para a de saída. Se a natureza das variables fose decimal (punto flotante), empregaríamos ‘float’ ou incluso ‘double’.
- Con isto debemos ter clara a definición da función:

  float interpola(int valor, int x0, int x1, int y0, int y1)

- Unha vez clara a definición, podemos traballar sobre o algoritmo que implementa, neste caso xa o fixemos na diapositiva anterior. Desta maneira:

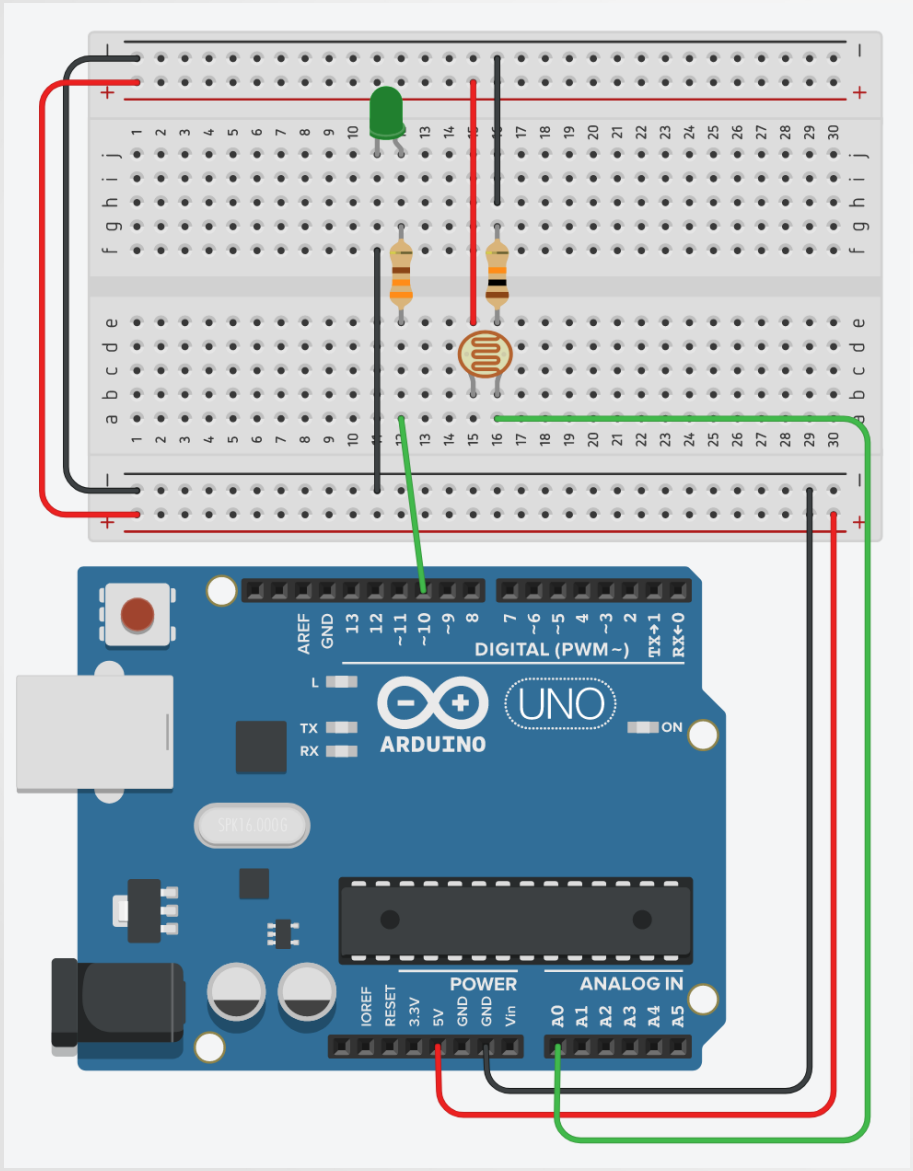
```
float interpola(int valor, int x0, int x1, int y0, int y1) {
  float interpolado = y0 + 1. * (y1 - y0) * (valor - x0) / (x1 - x0);
  return interpolado;
}
```

Entradas analóxicas – Lectura de LDR

- Co anterior temos unha función que asigna directamente un nivel de entrada (cos límites que sexa) a un valor de saída entre 0 e 255.
- Agora temos que ter en conta que o LED debe aumentar o brillo conforme diminúa a luz ambiente e viceversa.
- Para ter en conta isto debemos restar a 255 o valor interpolado que acabamos de obter.
- Un posible sketch é o da diapositiva seguinte.

Víctor . Álvarez 201911

Entradas analóxicas – Lectura de LDR



LDR.controla.LED.interpolacion | Arduino 1.8.10

LDR.controla.LED.interpolacion

```
/*
 * LED modifica o brillo conforme varía a luz ambiente.
 */

#define VERDE 10
#define LDR A0

int tempo = 1000;

void setup() {

}

void loop() {
  int valorLDR = analogRead(LDR);
  //Valores de entrada min/max: 210/997
  int valorLED =  (int) interpola(valorLDR, 210, 997, 0, 255);
  analogWrite(VERDE, 255 - valorLED);
  delay(tempo);
}

float interpola(int valor, int x0, int x1, int y0, int y1) {
  float interpolado = y0 + 1. * (y1 - y0) * (valor - x0) / (x1 - x0);
  return interpolado;
}
```

Guardado com Sucesso.

O rascunho usa 3922 bytes (12%) do espaço de armazenamento do programa.
Variáveis globais usam 200 bytes (9%) de memória dinâmica, restando 184

18 Arduino/Genuino Uno em /dev/cu.wchusbserial1420

Víctor . Álvarez 201911

Entradas analóxicas – Lectura de LDR

- A función interpola() que acabamo de deseñar é equivalente á función map(), que pertence á linguaxe de Arduino.
- Podes ver a referencia á función map() en:

  https://www.arduino.cc/reference/en/language/functions/math/map/
- Esta función ten unha pequena pega, averigua na documentación cal é
- Outra función interesante para usar combinada con map() é a función constrain():

  res = constrain(valor, min, max);
- Esta función asegura que a variable ‘valor’ pertenza ao intervalo [min, max]. No caso de que se sobrepase algún dos dous límites, forza ‘res’ a tomar o valor do límite superado.
- Por exemplo:

  constrain(12, 35, 90);
- Dá como resultado o valor 35, que é o límite inferior sobrepasado por 12.
- A referencia á función constrain() podes encontrala en:

  https://www.arduino.cc/reference/en/language/functions/math/constrain/

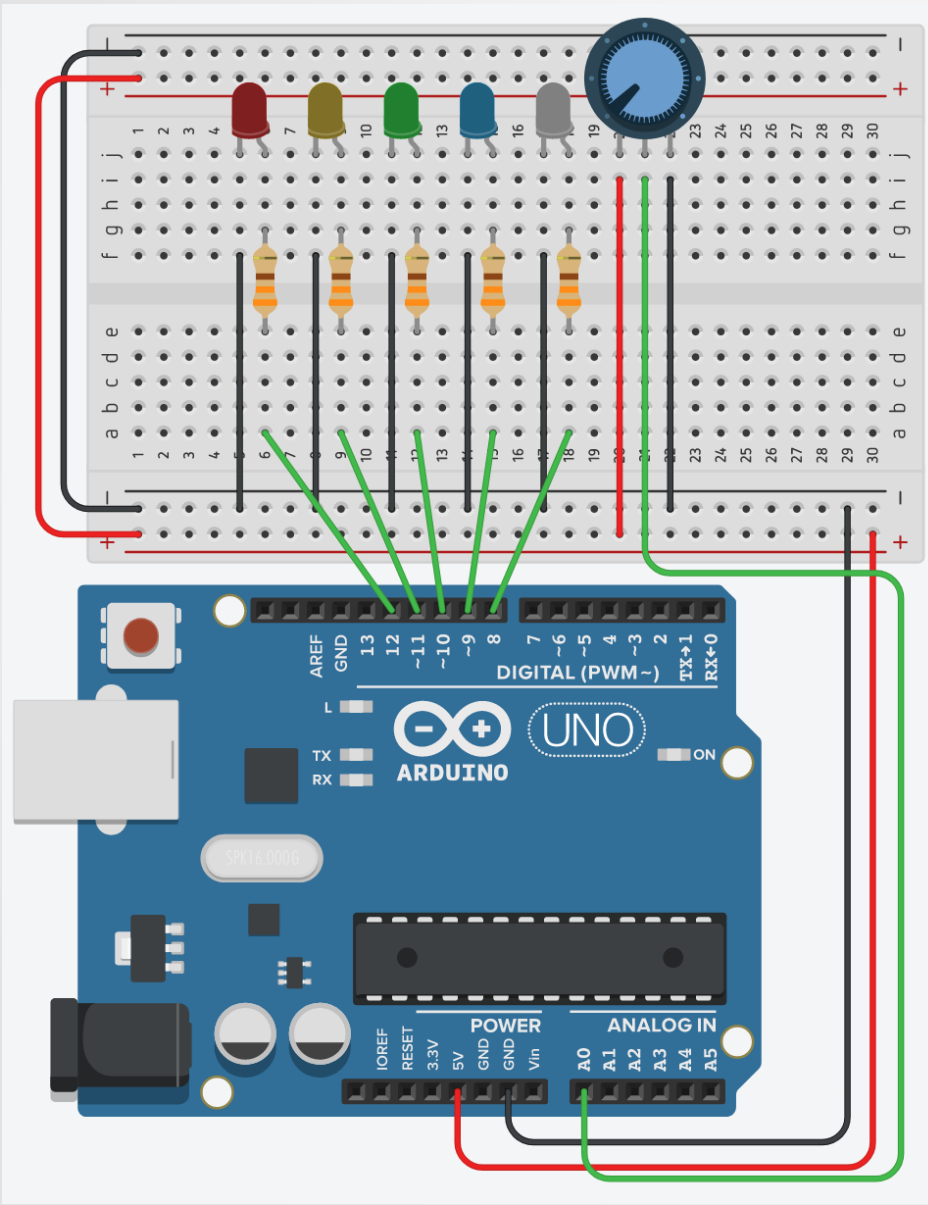
Víctor . Álvarez 201911

Entradas analóxicas – Potenciómetro

- Para o seguinte script imos ler o valor de tensión á saída dun potenciómetro, para encender sucesivamente 5 LEDs.
- Un potenciómetro é unha resistencia variable, que consiste nunha resistencia fixa sobre a que desliza un cursor. Por tanto é un elemento con tres contactos. Normalmente os dos extremos son a resistencia máxima e o central corresponde co cursor, que vai deixando a un lado ou outro unha parte da resitencia total.
- Conecta un extremo do potenciómetro a 5 V e o outro GND, o cursor será o valor da entrada analóxica. Os 5 LEDs poden ir conectados a saídas dixitais, xa que só os imos por en ON/OFF.
- A partir a entrada analóxica e mapeando aos valores 0 a 1024, reparte o rango en cinco partes e ilumina os LEDs en función do valor recibido.
- Podes empregar tanto map() como interpola().

Víctor . Álvarez 201911

Entradas analóxicas – Potenciómetro



potenciometro.t.leds.if.else | Arduino 1.8.10

potenciometro.t.leds.if.else

```
/*
 * Encéndense 5 LEDs en función do valor
 * analóxico lido no potenciómetro.
 */

#define BRANCO 8
#define AZUL 9
#define VERDE 10
#define AMARELO 11
#define VERMELLO 12
#define POTENCIOMETRO A0

int tempo = 200;
int valor = 0;

void setup() {
  pinMode(VERDE, OUTPUT);      pinMode(AMARELO, OUTPUT);
  pinMode(VERMELLO, OUTPUT);   pinMode(AZUL, OUTPUT);
  pinMode(BRANCO, OUTPUT);
}

void loop() {
  valor = analogRead(POTENCIOMETRO); //Entre 0 e 1023
  valor = constrain(valor, 0, 1023);
  digitalWrite(BRANCO, LOW);     digitalWrite(AZUL, LOW);
  digitalWrite(VERDE, LOW);      digitalWrite(AMARELO, LOW);
  digitalWrite(VERMELLO, LOW);
  if(valor > 0 && valor < 204) digitalWrite(BRANCO, HIGH);
  else if(valor >= 204 && valor < 408) digitalWrite(AZUL, HIGH);
  else if(valor >= 408 && valor < 612) digitalWrite(VERDE, HIGH);
  else if(valor >= 612 && valor < 816) digitalWrite(AMARELO, HIGH);
  else if(valor >= 816 && valor <= 1023) digitalWrite(VERMELLO, HIGH);
  delay(tempo);
}
```

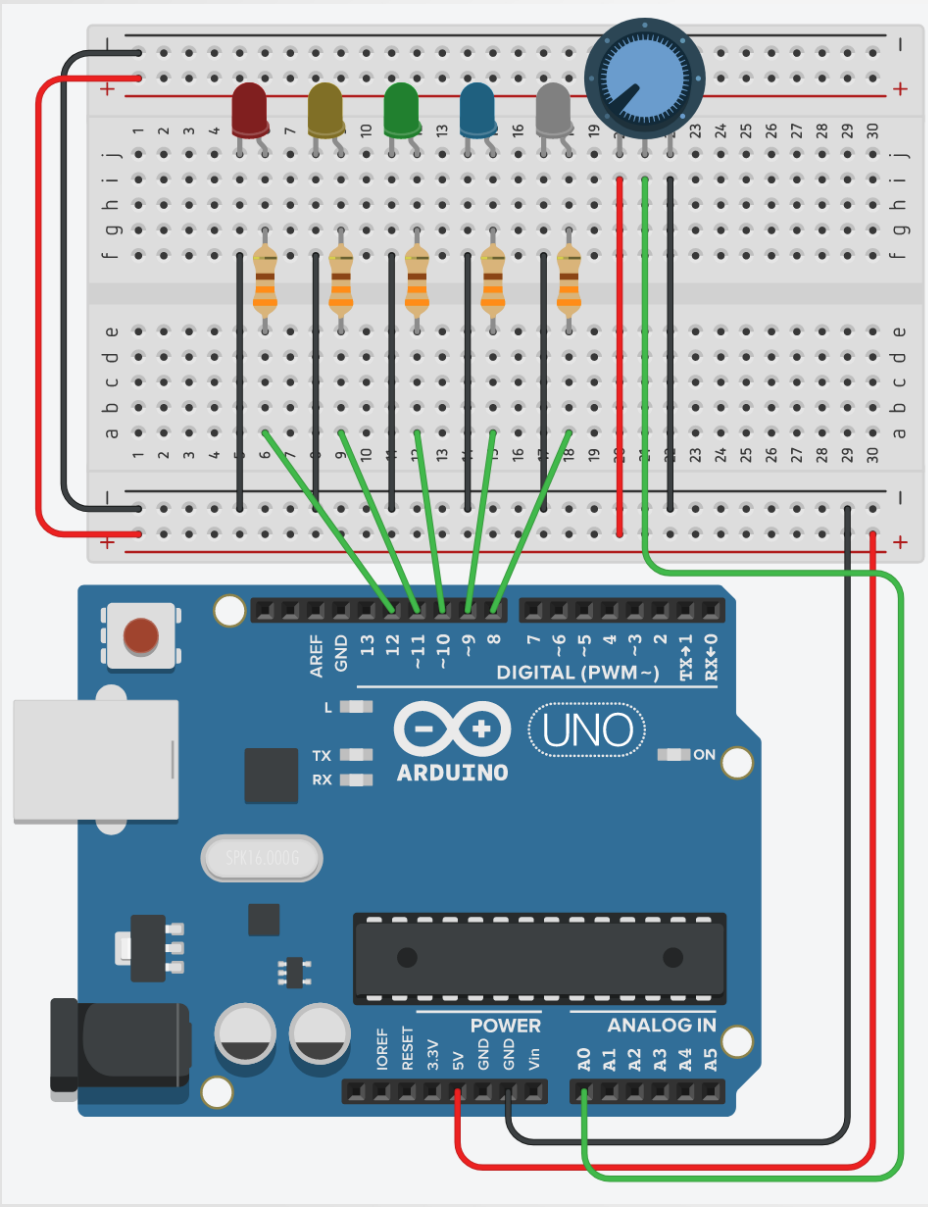
Compilação Terminada

O rascunho usa 1114 bytes (3%) do espaço de armazenamento do programa
Variáveis globais usam 11 bytes (0%) de memória dinâmica, restando 20

34 Arduino/Genuino Uno em /dev/cu.wchusbserial1420

Víctor . Álvarez 201911

Entradas analóxicas – Potenciómetro



potenciometro.5.leds.switch.case | Arduino 1.8.10

potenciometro.5.leds.switch.case

```
/*
 * Encéndense 5 LEDs en función do valor
 * analóxico lido no potenciómetro.
 */

#define BRANCO 8
#define AZUL 9
#define VERDE 10
#define AMARELO 11
#define VERMELLO 12
#define POTENCIOMETRO A0

int tempo = 200;
int valor = 0;

void setup() {
  pinMode(VERDE, OUTPUT);      pinMode(AMARELO, OUTPUT);
  pinMode(VERMELLO, OUTPUT);   pinMode(AZUL, OUTPUT);
  pinMode(BRANCO, OUTPUT);
}

void loop() {
  valor = analogRead(POTENCIOMETRO); //Entre 0 e 1023
  valor = constrain(valor, 0, 1023);
  valor = map(valor, 0, 1023, 0, 4); //Mapea a 0, 1, 2, 3, 4
  digitalWrite(BRANCO, LOW);     digitalWrite(AZUL, LOW);
  digitalWrite(VERDE, LOW);      digitalWrite(AMARELO, LOW);
  digitalWrite(VERMELLO, LOW);
  switch(valor) {
    case 0:
      digitalWrite(BRANCO, HIGH); break;
    case 1:
      digitalWrite(AZUL, HIGH); break;
    case 2:
      digitalWrite(VERDE, HIGH); break;
    case 3:
      digitalWrite(AMARELO, HIGH); break;
    case 4:
      digitalWrite(VERMELLO, HIGH); break;
  }
  delay(tempo);
}
```

Guardado com Sucesso.

O rascunho usa 1280 bytes (3%) do espaço de armazenamento do programa
Variáveis globais usam 11 bytes (0%) de memória dinâmica, restando 20

39 Arduino/Genuino Uno em /dev/cu.wchusbserial1420

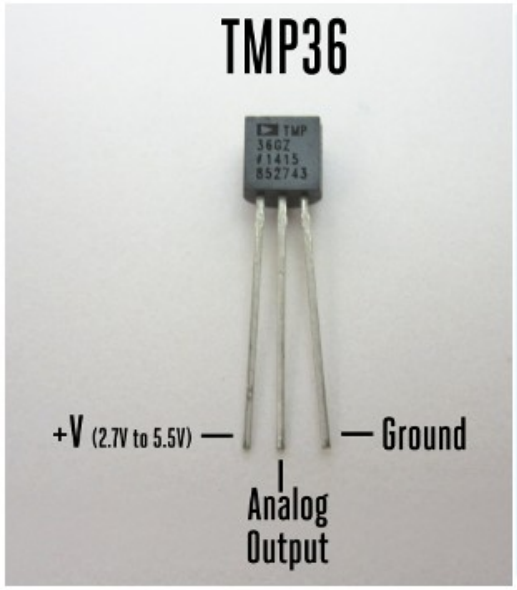
Víctor . Álvarez 201911

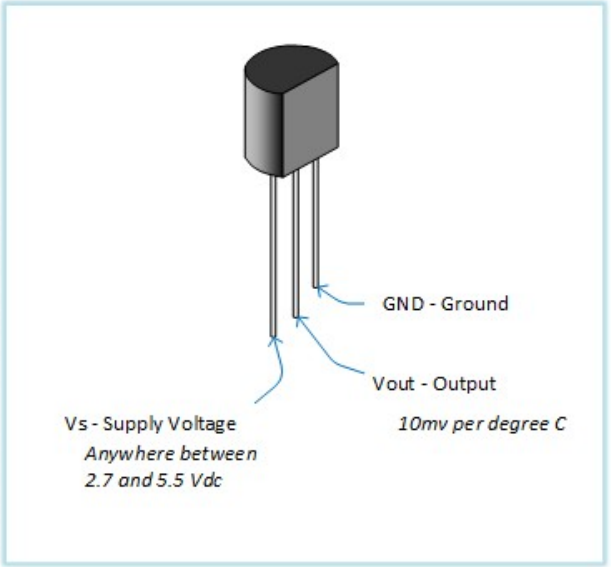
Entradas analóxicas – TMP36 -> 5 LEDs

- Para o seguinte script imos empregar un sensor de temperatura TMP36. Este sensor proporciona un valor analóxico a partir do que se pode calcular a temperatura medida.
- Comercialízase encapsulado de forma parecida a moitos transistores. Coa parte plana cara nós, o pin central é a saía analóxica, e os laterais alimentación (GND o da dereita).
- É un sensor que proporciona 10 mV/ºC e ten un rango de medición lineal entre -40 ºC e +125 ºC, aínda que pode operar até +150 ºC. A súa precisión está ±1 ºc ao redor de 25 ºC e nos ±2 ºC nos extremos do rango. Aliméntase desde 2.7 a 5.5 V. Nós alimentaremos a 5 V.
- Nas especificacións técnicas di que a 25 ºC proporciona 750 mV e aumenta ou diminúe 10 mV/ºC, polo que a tensión na saída en función da temperatura ambiente será:

$$V_{out} = 10\,mV/ºC \cdot t + 500\,mV$$

Datasheet TMP36

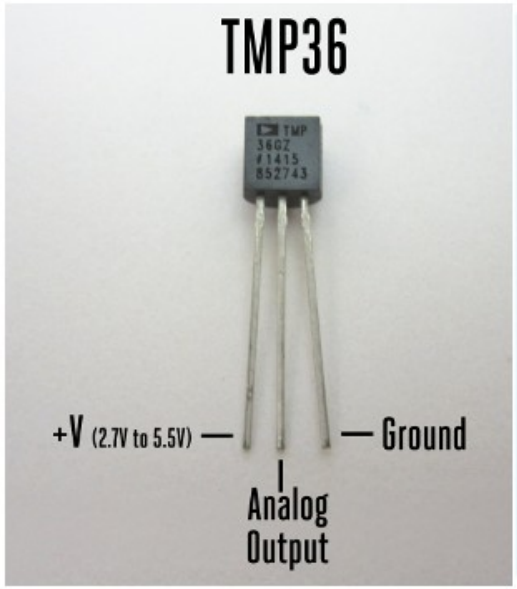


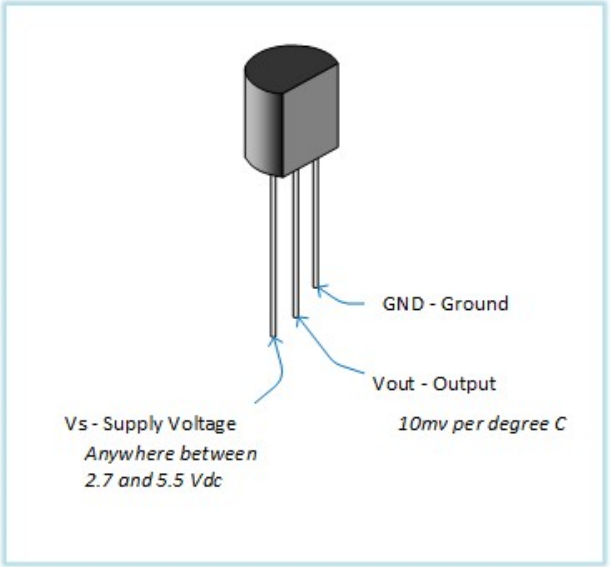


Víctor . Álvarez 201911

Entradas analóxicas – TMP36 -> 5 LEDs

- Aproveita a montaxe dos 5 LEDs anteriores e establece que se encendan sucesivamente en función da temperatura no rango [14 ºC, 30 ºC]. Os LEDs deben permanecer acesos conforme a temperatura vai aumentando e apagárense ao ir diminuíndo a mesma.
- Necesitarás empregar as funcións constrain() e map(), así como algunha estructura condicional que pode ser switch-case ou if-else.
- Asegúrate que por debaixo de 14 ºC non hai ningún LED aceso e por cima dos 30 ºC permanecen todos acesos.

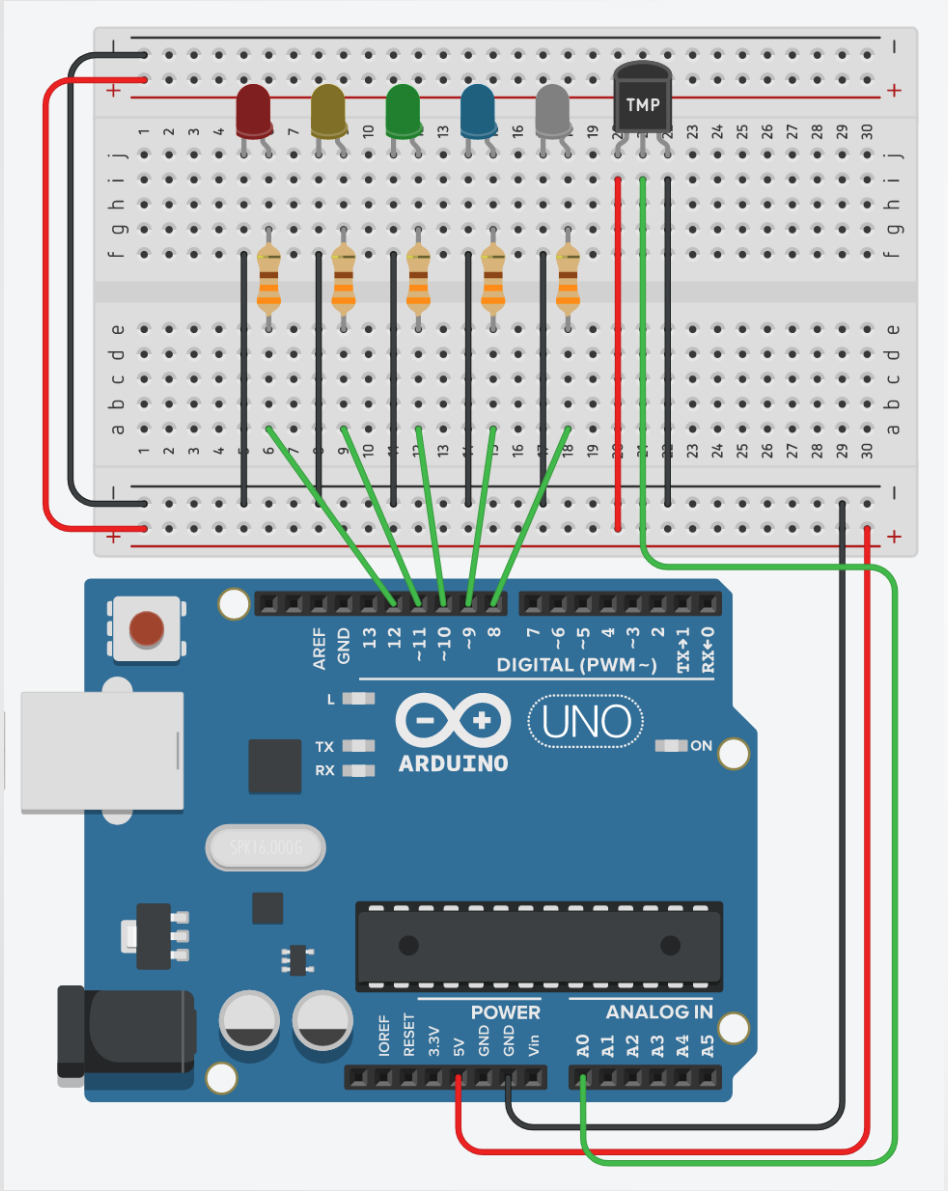




Datasheet TMP36

Víctor . Álvarez 201911

Entradas analóxicas – TMP36 -> 5 LEDs



tmp36 | Arduino 1.8.10

tmp36

```
#define AZUL 9
#define VERDE 10
#define AMARELO 11
#define VERMELLO 12
#define TMP36 A0

int tempo = 200;
int vs = 0;          //Vs na saída do TMP36 en mV
float tempC = -50;  //Temp. ambiente en °C
int idLED = -1;      //Identificador do LED

void setup() {
  pinMode(VERDE, OUTPUT);      pinMode(AMARELO, OUTPUT);
  pinMode(VERMELLO, OUTPUT);   pinMode(AZUL, OUTPUT);
  pinMode(BRANCO, OUTPUT);
}

void loop() {
  vs = map(analogRead(TMP36), 0, 1023, 0, 5000); //mapea a mV
  tempC = 1.* (vs - 500) / 10;  //Calcula temp en °C
  idLED = constrain((int) tempC, 14, 30); //Restrinxe a [14, 30] °C
  idLED = map(idLED, 14, 30, 0, 4);
  digitalWrite(BRANCO, LOW);     digitalWrite(AZUL, LOW);
  digitalWrite(VERDE, LOW);      digitalWrite(AMARELO, LOW);
  digitalWrite(VERMELLO, LOW);
  if(tempC >= 14) {
    switch(idLED) {
      case 4:
        digitalWrite(VERMELLO, HIGH);
      case 3:
        digitalWrite(AMARELO, HIGH);
      case 2:
        digitalWrite(VERDE, HIGH);
      case 1:
        digitalWrite(AZUL, HIGH);
      case 0:
        digitalWrite(BRANCO, HIGH);
    }
  }
  delay(tempo);
}
```

Compilação Terminada

O rascunho usa 2126 bytes (6%) do espaço de armazenamento do programa
Variáveis globais usam 15 bytes (0%) de memória dinâmica, restando 20

48 Arduino/Genuino Uno em /dev/cu.wchusbserial1420

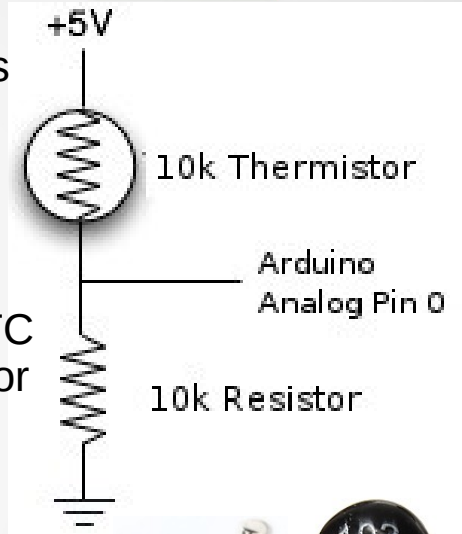
Víctor . Álvarez 201911

Entradas analóxicas – Outros sensores de T

- Existe unha gran variedade de sensores de temperatura analóxicos. Algúns dos máis empregados son os termistores ou resistencias variables (NTC ou PTC). Adoitan ser moi baratos e sensibles. Normalmente para conectalos en divisor de tensión. Un dos problemas que teñen é que hai que calibralos, xa que a súa saída non e lineal.
- Para conectar p.ex. unha NTC, dispoñemos un divisor de tensión cunha resistencia de pull down dun valor similar á resistencia que presenta a NTC á temperatura ambiente, para que o punto central do divisor estea ao redor de 2.5 V á temperatura ambiente (ou á temperatura que queramos ter de referencia).
- Para calibrar NTC necesitamos os parámetros que veñen no seu datasheet e usar unha calculadora online como a de:

  https://www.thinksrs.com/downloads/programs/therm%20calc/ntccalibrator/ntccalculator.html

- Outra forma, menos precisa pero máis sinxela, é calibar a NTC considerando que o rango no que imos traballar é lineal. Para isto identificamos os valores extremos do rango de temperaturas e a que tensións corresponden no punto central do divisor. Empregando constrain() para asegurar os valores dentro do rango e map() para asignar ás temperaturas correspondentes. Unha vez feito isto, podemos desenvolver o script final.

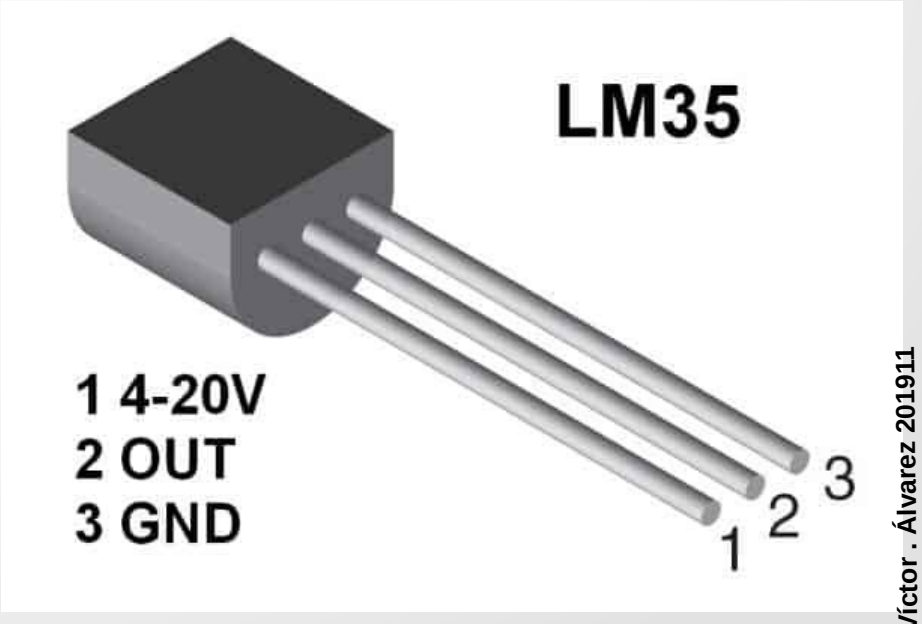


Víctor . Álvarez 201911

Entradas analóxicas – Outros sensores de T

- Outos sensores analóxicos moi empregados son os da familia do LM35, que se asemellan moito ao TMP36, aínda que con mellores resultados en canto a precisión en rango de medida. A súa resposta é lineal e está nos 10 mV/ºC e non é necesario restarlles ningún offset (en 0 ºC, a lectura é 0 V). A alimentación dos mesmos pode estar entre 4 e 20 V, aínda que a saída refírese a 5 V (ou 3.3 V se modificamos AREF).
- Segundo isto, a saída do sensor será: $V_{out} = 10\,mV/ºC \cdot t$
- Reutiliza a montaxe do exercicio anterior e sustitúe o sensor TMP36 polo LM35.
- Revisa o script do TMP36 e adáptao ao novo sensor LM35.
- Comproba que ambos sensores son moi fáciles de sustituir un polo outro.
- Encontras discrepancias nos valores medidos por ambos sensores?

Datasheet LM35



Entradas analóxicas

- Nesta unidade aprendemos a:
  - usar unha entrada analóxica (non é preciso declarala), con analogRead()
  - conectar sensores analóxicos como divisores de tensión (LDR, NTC, potenciómetro, etc)
  - declarar, elaborar e usar unha función definida polo usuario
  - realizar ‘casting’ de variables, p.ex. float a int
  - mapear entradas (precisión 10 bits) a valores de variables ou a saídas PWM (precisión 8 bits)
  - usar as funcións constrain() e map()
  - usar a estructura condicional switch-case como alternativa á if-else
  - combinar switch-case con map() para facilitar a lexibilidade do script
  - comparar varios sensores analóxicos de temperatura (TMP36, LM35, NTC)

Víctor . Álvarez 201911