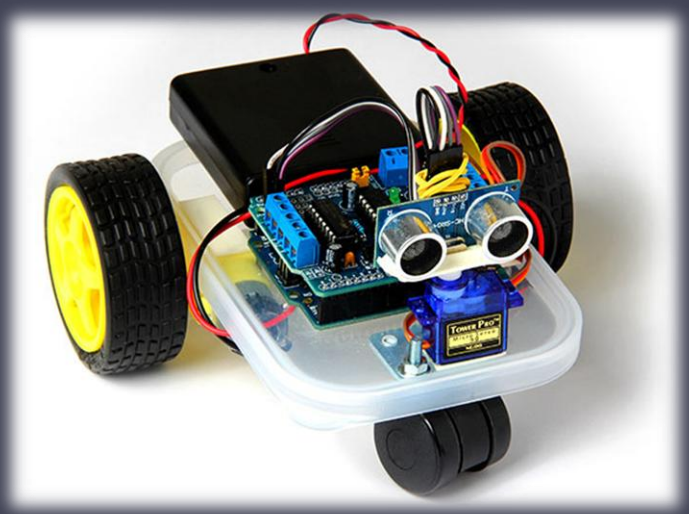


WORKSHOP ARDUINO

ZDAY e Arduino Day 2015
28 de março

Velha-a-Branca – Braga, Portugal

Fábio Oliveira
fabio_oliveira500@hotmail.com

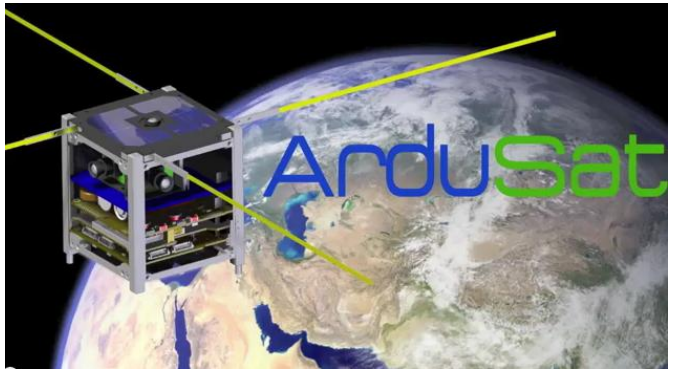
No final do workshop…

- Serás capaz de:
  - Ligar e desligar um LED;
  - Verificar estado de um botão;
  - Ler valores de sensores analógicos;
  - Controlar a luminosidade de um LED;
  - Visualizar dados no computador;
  - Controlar um servomotor;
  - Ler distâncias;
  - Controlar um Liquid Crystal Display (LCD);

O que é o Arduino?

- Arduino é uma plataforma de prototipagem eletrónica *open-source*, baseado em *hardware* e *software* flexível, de fácil uso;

- É projetado para artistas, designers, amadores e todos os interessados em criar objetos e/ou ambientes interativos.



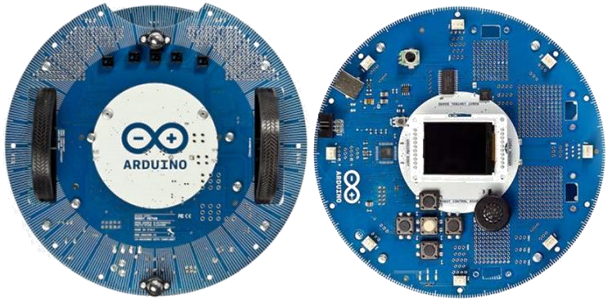
Produtos...

BOARDS

Arduino Uno

Arduino Mega 2560

LilyPad Arduino USB



Arduino Robot

SHIELDS

Arduino GSM Shield

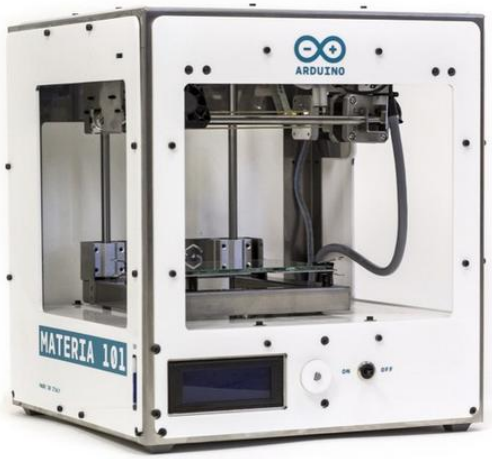
Arduino Wireless SD Shield

Arduino Ethernet Shield

KITS

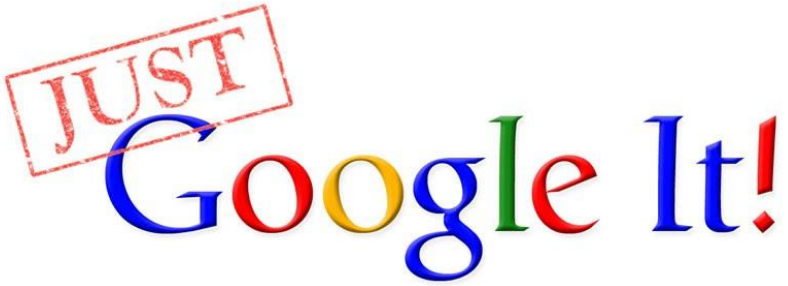
The Arduino Starter Kit

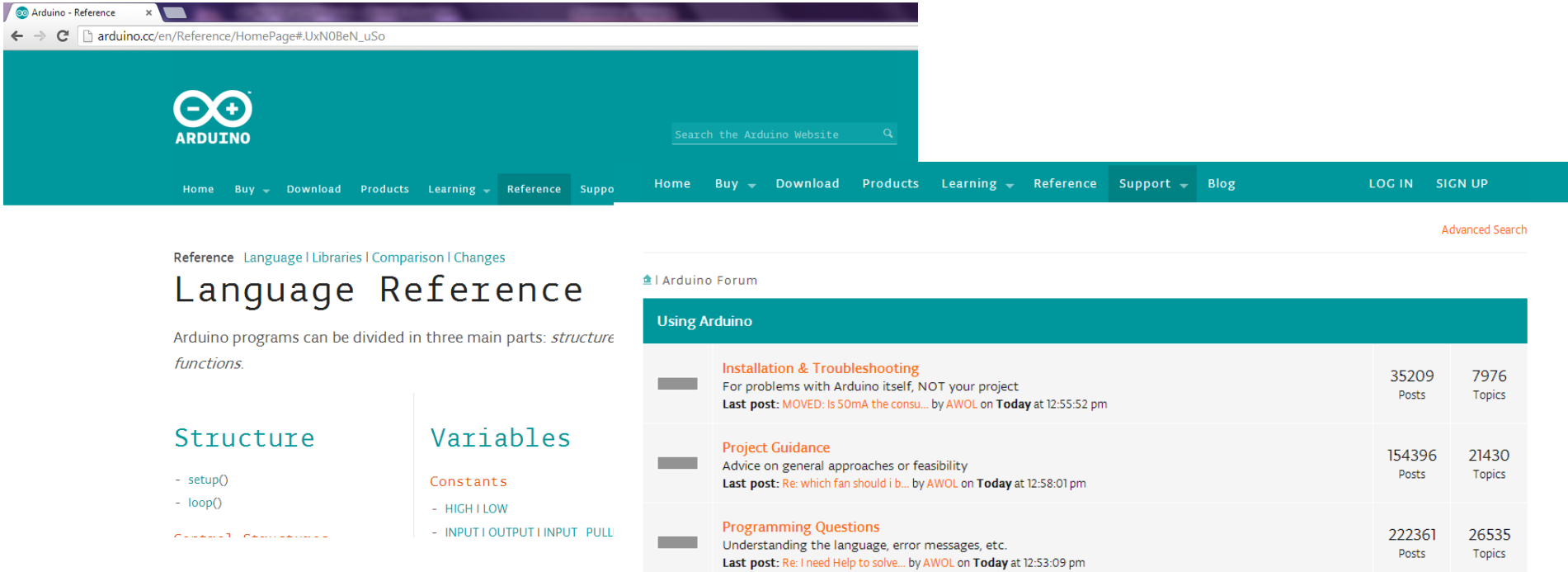
Arduino Materia 101



Comunidade

- Site global
  - http://arduino.cc/
- Fórum
  - http://forum.arduino.cc/
- Referência
  - http://arduino.cc/en/Reference/HomePage





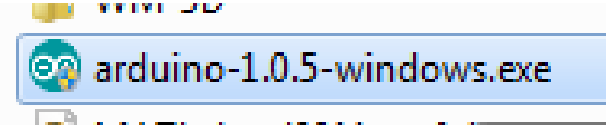
Instalar Ambiente de desenvolvimento

- Fazer o download da aplicação:
  - http://arduino.cc/en/Main/Software#.UxNpQON_uSo

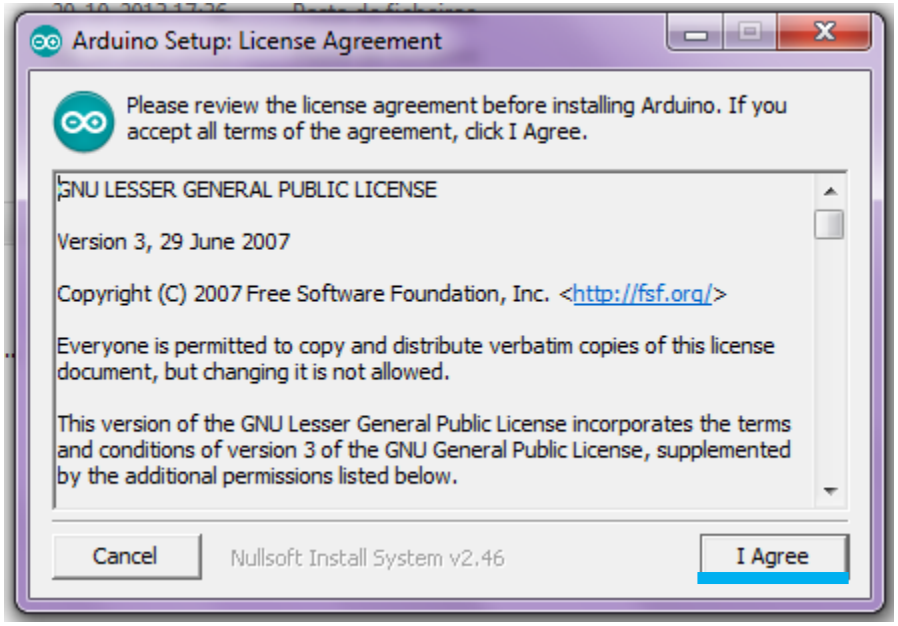
Instalar Ambiente de desenvolvimento

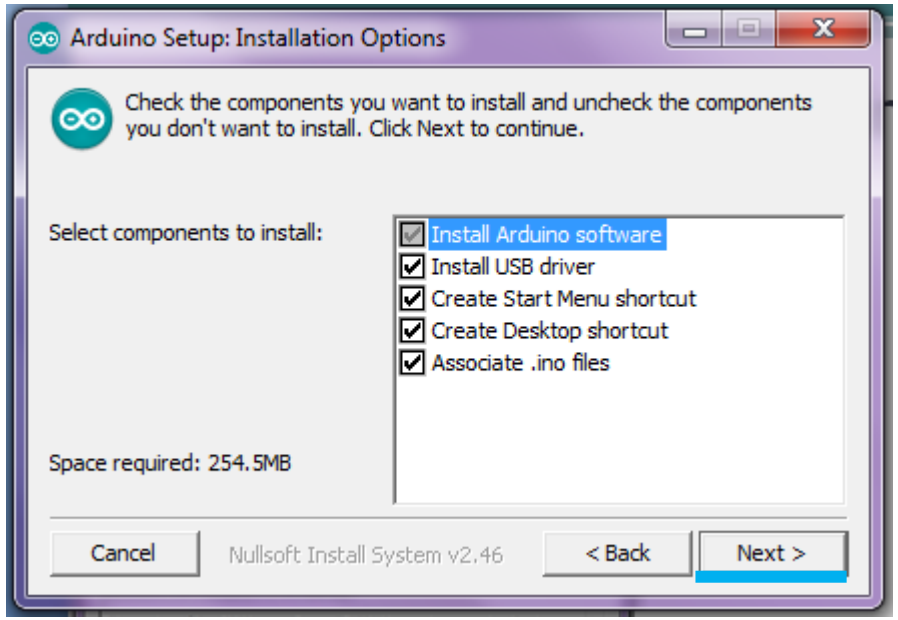
(continuação)

- Executar executável



- Correr janelas...

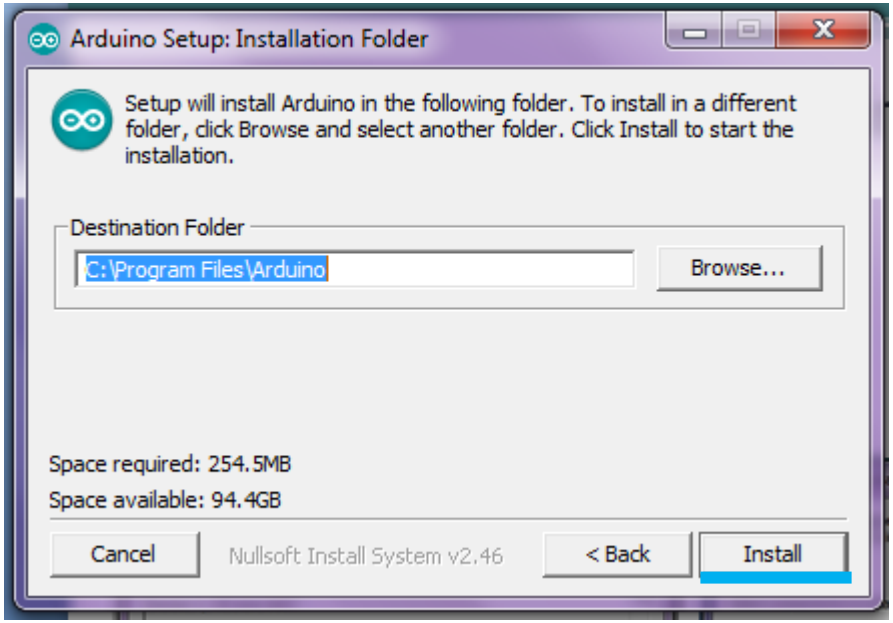


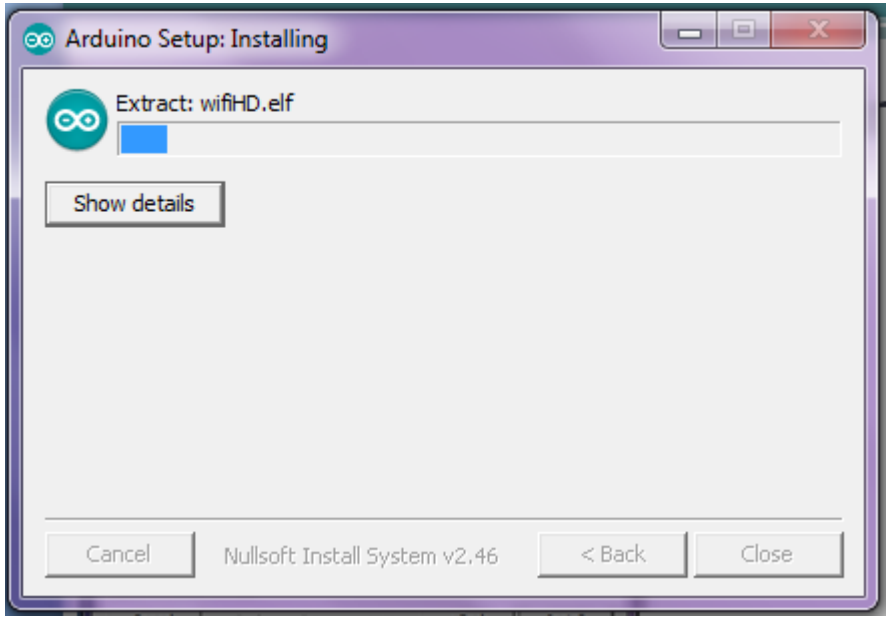


Instalar Ambiente de desenvolvimento
(continuação)

- Escolher local de instalação...

- Deixar finalizar instalação...

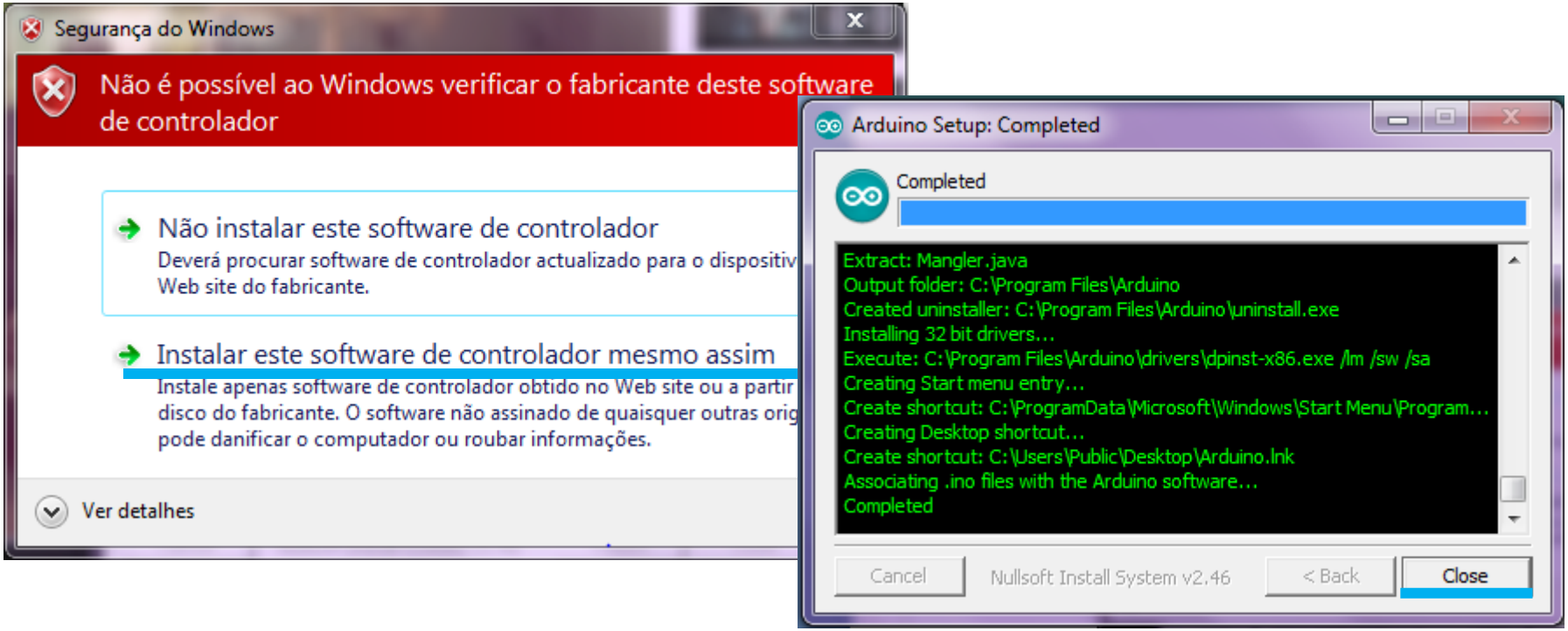




Instalar Ambiente de desenvolvimento

(continuação)

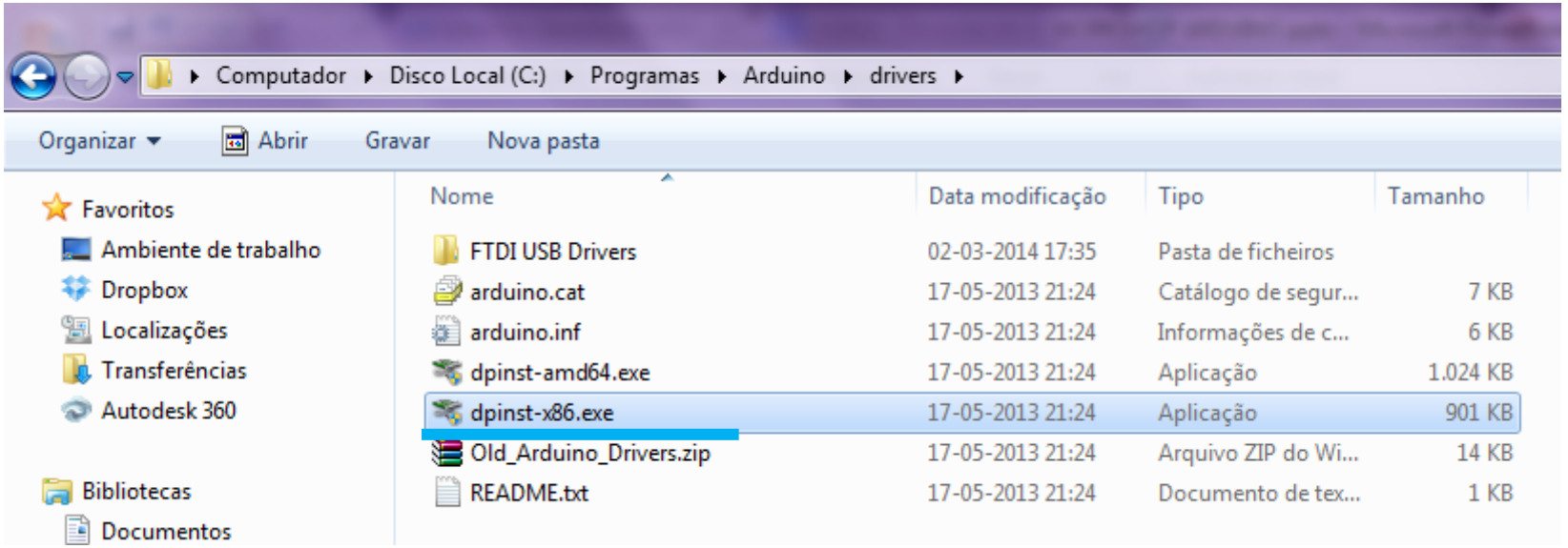
- Se aparecer aviso do *Windows*, "Instalar este *software*"…
- Concluir instalação.



Instalar Ambiente de desenvolvimento

(continuação)

- Se instalação na localização pré-definida, abrir a pasta C:\Program Files\Arduino\drivers , e instalar ficheiro abaixo para sistema operativo de 32 bits:

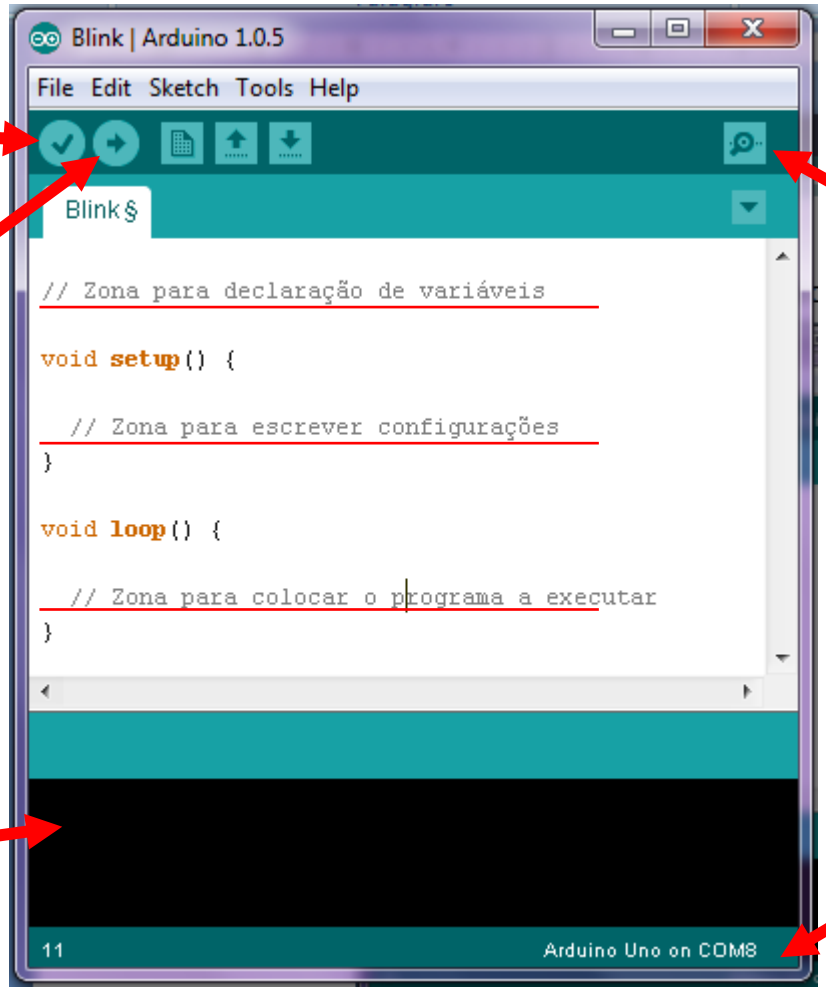


Utilizar o IDE do Arduino

Integrated Development Environment

IDE Arduino

- Executar o programa fazendo duplo click no atalho criado:



Verificar código

Programar placa

Abrir o Monitor Série

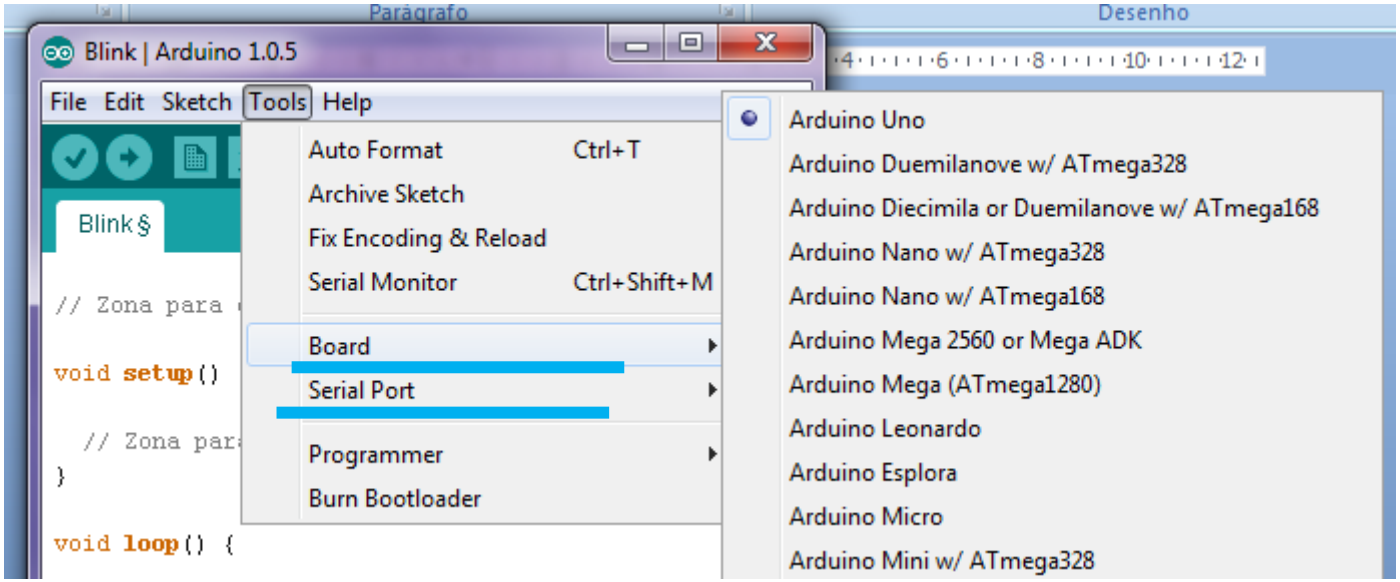
Mensagens de erro, *upload*, etc...

Placa conectada e qual a porta em uso

IDE Arduino

(continuação)

- Selecionar a porta Série a utilizar;
- Selecionar a placa em uso;



Programação

Como “falar” com um Arduino?

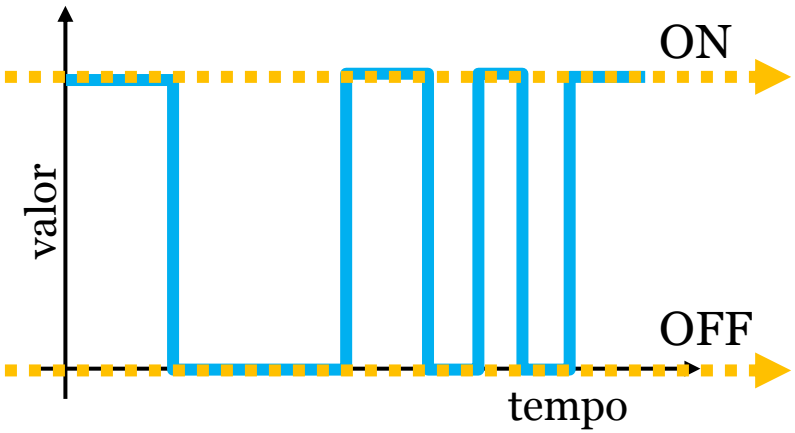
Tipo de portas

- Portas digitais

(só podem assumir 2 valores)

```
// Configura porta 13 como saída
pinMode(13, OUTPUT);
// Configura porta 12 como entrada
pinMode(13, INPUT);

// Ativa saída digital 13
digitalWrite(13, HIGH);
// Desativa saída digital 13
digitalWrite(13, LOW);

// Lê o estado de uma entrada digital
// e guarda naa variável int valor
valor = digitalRead(13);
```

DIGITAL

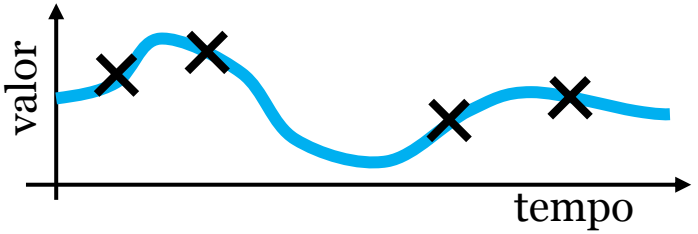


- Portas analógicas

(assumem um valor ou 'simulam' um valor)

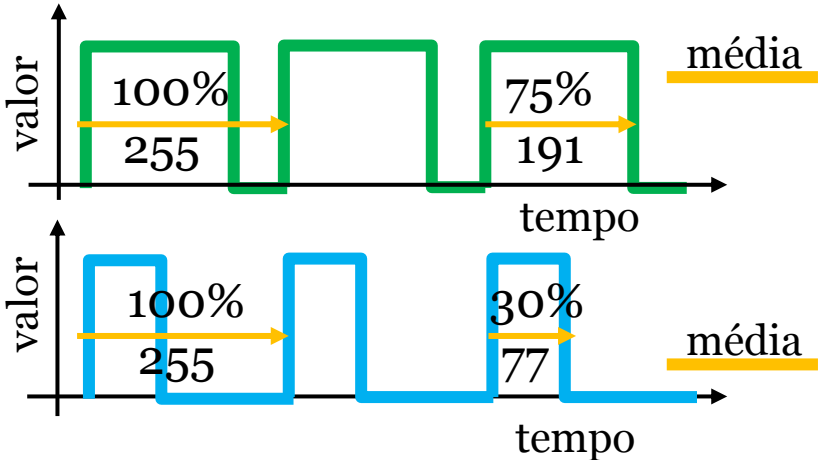
```
// Leitura do valor analógico da porta 0
// Devolve valor entre 0 (0 V) e 1023 (5 V)
valor = analogRead(0);
```

A0




```
// Cria uma onda quadrada com o duty_cycle pretendido
// Valores entre 0 (0%) e 255 (100%)
analogWrite(ledPin, duty_cycle);
```

(PWM~)



Por exemplo, para o Arduino UNO

Portas digitais

Ligação ao PC

Alimentação externa

Alimentação de circuitos

Portas analógicas

Variáveis

- Permitem guardar dados;
  - Consoante os tipos de dados, podemos ter diferentes tipos de variáveis
    - Valores inteiros (-32’768 até 32’767)

```
// Declaração da variável
int numero = 2;
// Alteração da variável
numero = 10;
```

    - Valores inteiros (-2’147’483’648 até -2’147’483’648)

```
// Declaração da variável
long numero = 100;
// Alteração da variável
numero = 100000;
```

    - Valores com casas decimais

```
// Declaração da variável
float numero = 10.1;
// Alteração da variável
numero = 3.14159265359;
```

    - Caracteres (apenas 1)

```
// Declaração da variável
char letra = 'a';
// Alteração da variável
letra = '!';
```

    - Texto (neste caso 19 caracteres)

```
// Declaração da variável
char texto[20] = "ZDAY 2014";
// Alteração da variável
strcpy(texto,"Ola mundo!");
```

Operadores aritméticos

- Permitem efetuar contas

= (assignment operator)

+ (addition)

- (subtraction)

* (multiplication)

/ (division)

```
// Variável passa a ser igual 10
variavel = 10;
```

```
// Variável passa a ser igual 11
variavel = 10 + 1;
// Variável passa a ser igual a
// 1 mais o valor contido em número
variavel = 1 + numero
```

```
// Variável passa a ser igual 9
variavel = 10 - 1;
// Variável passa a ser igual a
// 10 menos o valor contido em número
variavel = 10 - numero
```

```
// Variável passa a ser igual 20
variavel = 10 * 2;
// Variável passa a ser igual a
// 10 vezes o valor contido em número
variavel = 10 * numero
```

```
// Variável passa a ser igual 5
variavel = 10 / 2;
// Variável passa a ser igual a
// à divisao de 10 pelo valor contido em número
variavel = 10 / numero
```

Operadores de comparação

- Permitem comparar dados

== (equal to)

!= (not equal to)

< (less than)

> (greater than)

<= (less than or equal to)

>= (greater than or equal to)

```
temperatura = 20;
referencia = 10;

// Condição falsa
( temperatura == referencia )

// Condição verdadeira
( temperatura != referencia )

// Condição falsa
( temperatura < referencia )

// Condição verdadeira
( temperatura > referencia )

// Condição verdadeira
( referencia <= temperatura )

// Condição falsa
( temperatura <= referencia )
```

Operadores boleanos

- Permitem ligar diferentes condições

&& (and)

|| (or)

```
numero_1 = 10;
numero_2 = 20;

// Todas as condições têm de ser verdadeiras,
// para que o resultado seja verdadeiro

// Como a 1.ª e 2.ª condição são verdadeiras
// o resultado será verdadeiro
( numero_1 <= 11 && numero_2 > 18 )

// Como a 2.ª condição é falsa
// o resultado será falso
( numero_1 >= 10 && numero_2 != 20 )
```

```
numero_1 = 10;
numero_2 = 20;

// Basta 1 condição ser verdadeira
// para que o resultado seja verdadeiro

// Como a 1.ª condição é verdadeira
// o resultado será verdadeiro
( numero_1 == 10 || numero_2 == 18)

// Como a 1.ª e a 2.ª condição são falsas
// o resultado será falso
( numero_1 >= 11 || numero_2 <= 19 )
```

Serão utilizados em ciclos lógicos!!!

Ciclos lógicos

- Ciclo “if” – Se verdadeiro, executa 1 vez

```
if( /* Escrever condição */){
  // Escrever código a executar caso condição verdadeira
}
else{
  // Escrever código a executar caso condição falsa
  // A escrita da parte 'else' não é obrigatória
}
```

```
// letra toma o valor de b
letra = 'b';

// Se letra for igual a a
if (letra == 'a'){
  // Ativa saída 13
  digitalWrite(13, HIGH);
}
// Caso não seja igual a a
else{
  // Desativa saída 13
  digitalWrite(13, LOW);
}
```

- Ciclo “switch” – Executa uma das condições 1 vez

```
// Consoante o valor de variavel
switch (variavel) {

  case valor1:
    // Colocar código a executar caso 'variavel' seja igual a valor1
  break;

  case valor2:
    // Colocar código a executar caso 'variavel' seja igual a valor2
  break;

  default:
    // Colocar código a executar caso nenhuma das condições anteriores seja verdadeira
  break;
}
```

```
// Valor toma o valor de 2
valor = 2;

// Consoante o número contido em valor
switch (valor) {
  case 1:
    // Se for 1, desativa saída 13
    digitalWrite(13, LOW);
  break;
  case 2:
    // Se for 2, ativa saída 13
     digitalWrite(13, HIGH);
  break;
  default:
    // Se tiver outro valor, não faz nada
  break;
}
```

Ciclos lógicos
(continuação)

- Ciclo “while” – Executa enquanto condição for verdadeira

```
while( /* Escrever condição */){
  // Escrever código a executar caso condição verdadeira
}
```

```
// Numero toma o valor de 0
numero = 0;
// Enquanto numero for menor do que 10
while( numero < 10 ){
  // Desativa saída 13
  digitalWrite(13, LOW);
  // Espera 1000 ms
  delay(1000);
  //    ativa saída 13
  digitalWrite(13, HIGH);
  // Espera 1000 ms
  delay(1000);

  // Soma 1 a numero
  numero = numero + 1;
}
```

- Ciclo “do, while” – Executa sempre 1 vez, e depois enquanto uma condição for verdadeira

```
do{
  // Código que é sempre executado pelo menos 1 vez
}while( /* Escrever condição */ );
```

```
// numero toma o valor de 0
numero = 0;
do{
   // Soma 1 a numero
   numero = numero + 1;
// Enquanto numero for menor do que 21
}while( numero <= 20 );
```

Ciclos lógicos
(continuação)

- Ciclo “for” – Executa n vezes até determinada condição ser verdadeira, e incrementa uma variável

```
for( /*valor inicial*/ ; /*condição para parar*/ ; /*valor a incrementar*/ ){
  // Código a executar
}
```

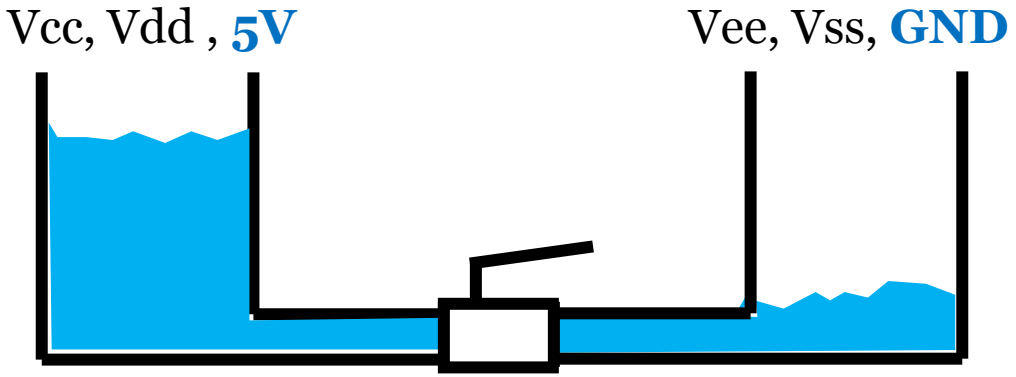
```
// Numero começa como 0, variará de 3 em 3, e o ciclo
// deixa de ser executado quando numero for maior do que 9
for( numero=0 ; numero < 10 ; numero = numero + 3 ){
  // Aguarda 1000 ms
  delay(1000);
  // Envia o valor de numero pela porta série
  Serial.println(numero);
}
```

Eletricidade?

A energia que faz a eletrónica funcionar…

Se pensarmos na eletricidade como água, então…

- Tensão
  - Diferença entre alturas
- Corrente
  - Caudal da água
- Resistência
  - Grau de impedimento do passador



Corrente = Tensão / Resistência

E isto é importante porque…

- Cada porta do Arduino tem um limite máximo de corrente que pode fornecer;
  - Resistências pequenas (passador muito aberto) irão conduzir a correntes maiores (mais água a passar);
  - Resistências grandes (passador muito fechado) irão conduzir a correntes menores (pouca água a passar);

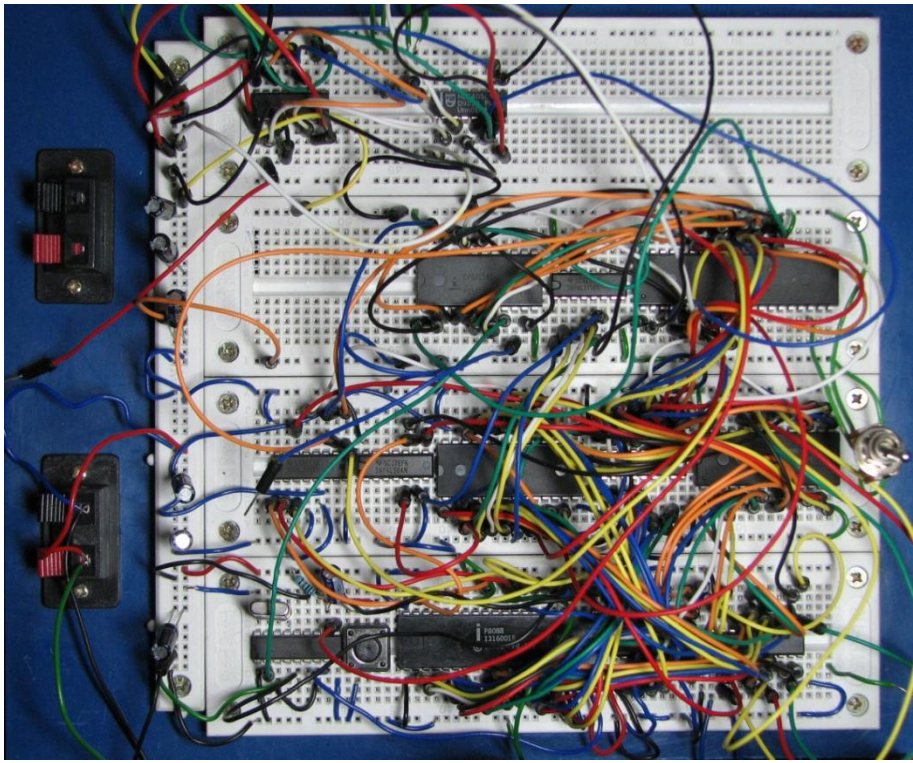
Não nos preocupemos agora com estes valores!
Para circuitos mais complexos terá de se ter estes fatores em consideração.

Como utilizar uma *breadboard*?

Ligação dos diferentes componentes do sistema...

O que é uma *breadboard*?

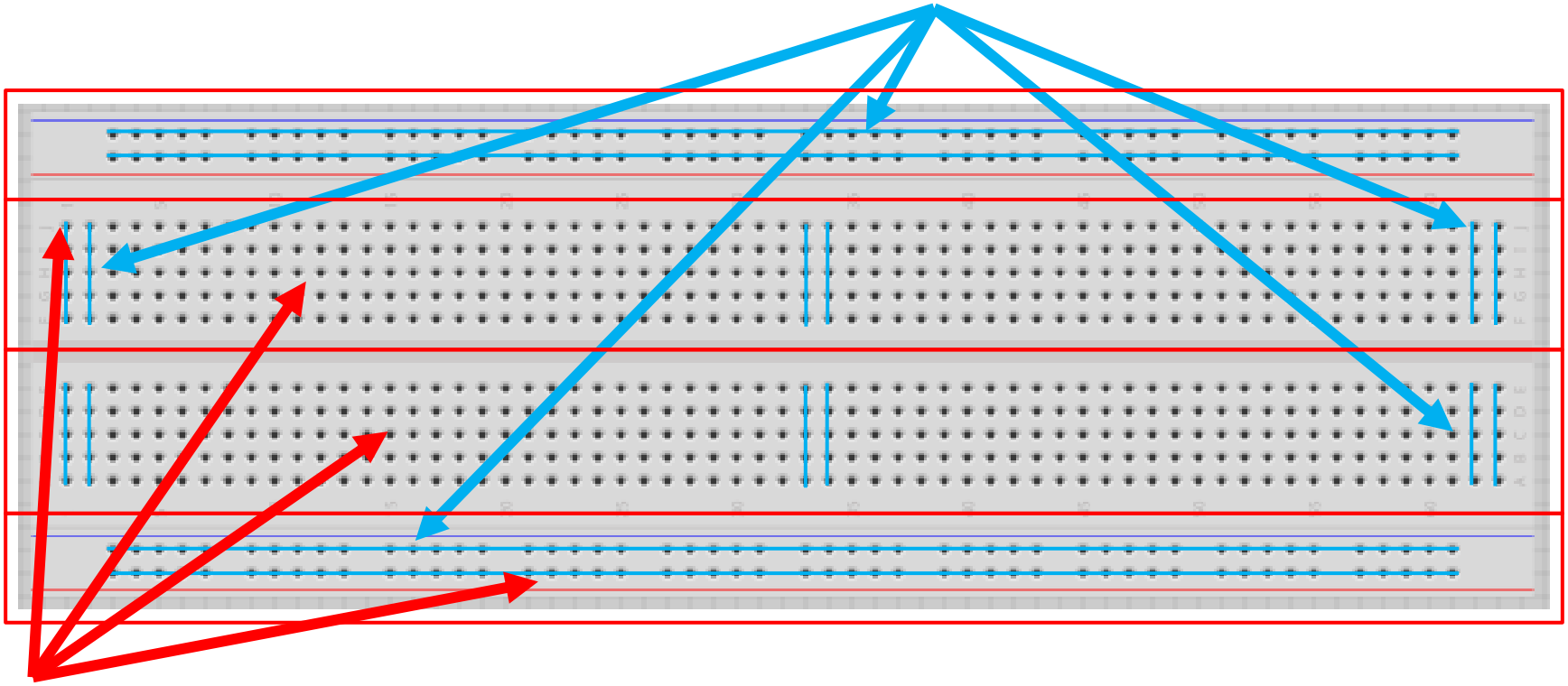
- Placa de ensaio ou matriz de contacto;
- Contém furos e conexões condutoras;
- Facilidade de inserção de componentes;
- Facilidade de ligação entre componentes;
- Não é preciso soldadura.



WIKIPEDIA
The Free Encyclopedia

Como funciona?

Há ligação interna entre as diferentes linhas!!!
Mesmo tanque!!!

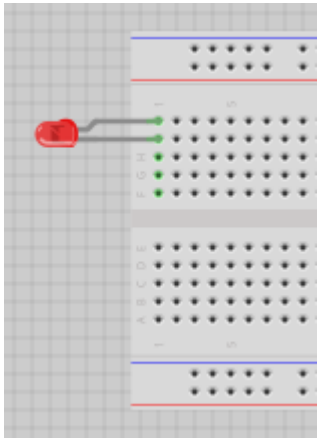


Não há ligação interna entre os diferentes blocos!!!
Tanques diferentes!!!

Normalmente, nas linhas horizontais ligam-se os 5V e o GROUND de forma a estarem acessíveis ao longo de toda a placa.

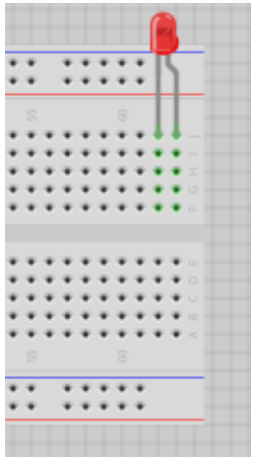
Por exemplo:

Errado...

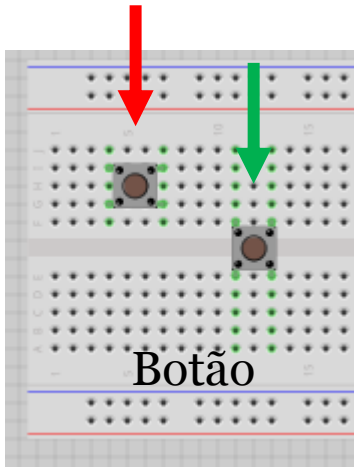


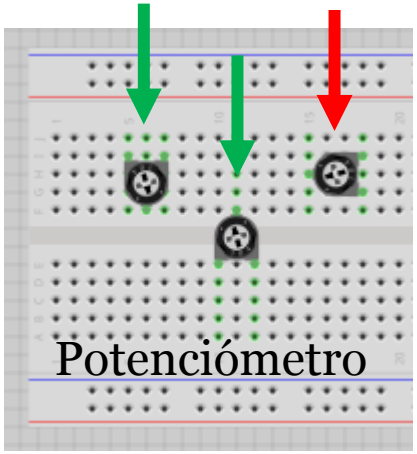
As duas pernas do LED estão na mesma linha com contacto interno. É como se estivessem a tocar uma na outra!

Certo!



As duas pernas do LED estão em linhas separadas pelo que não há contacto entre elas!



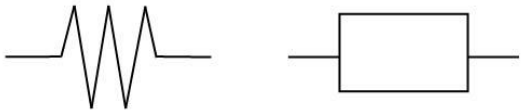


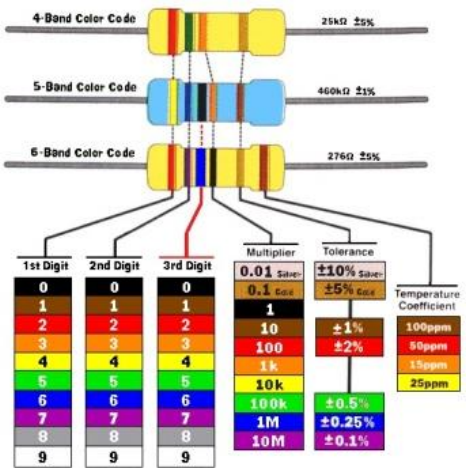
Projetos práticos

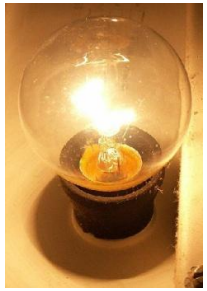
Está na altura de colocar as coisas a mexer!!!

Resistências e LEDs

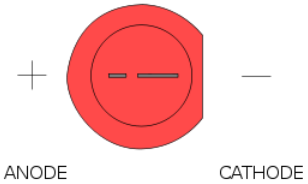
- Componente elétrico que se opõe à passagem de corrente elétrica;
- Produz calor;
- Diferentes tipos e tamanhos;
- Possui código de cores próprio.

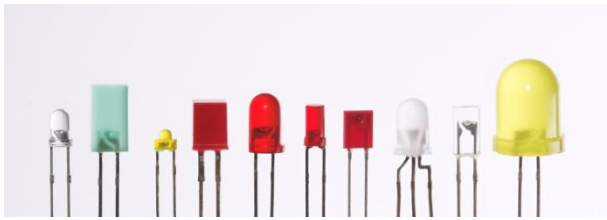


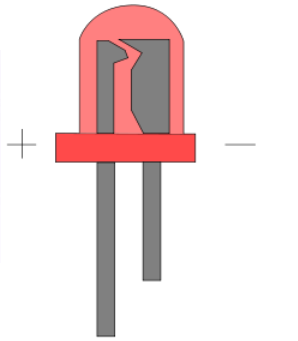




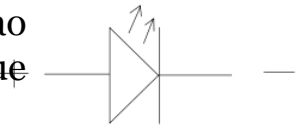
- Componente elétrico que emite luz;
- Deixa passar a corrente apenas num sentido;
- Baixo consumo;
- Diferentes tipos e tamanhos.



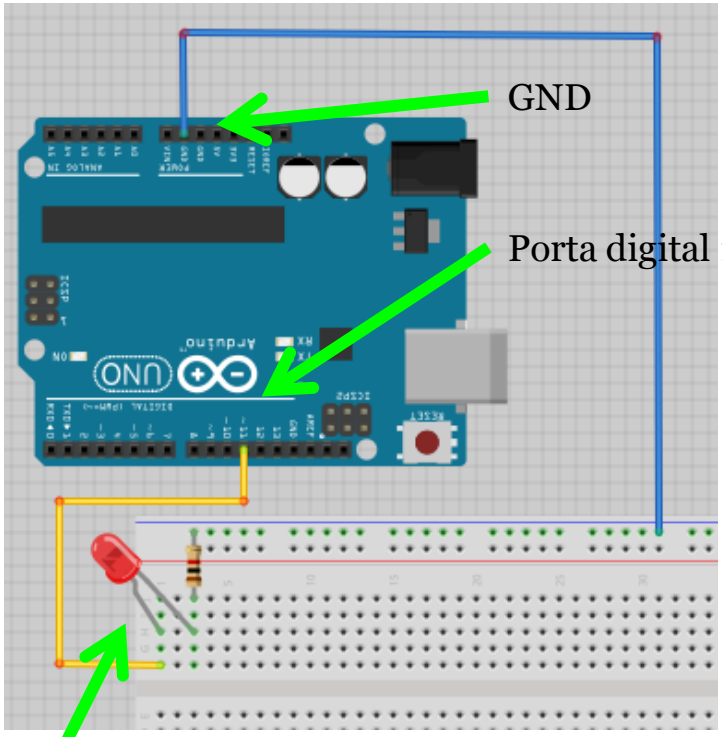




Coloca-se uma resistência ligada ao LED para regular-se a corrente que por ele passa!



Ligar/desligar LED automaticamente



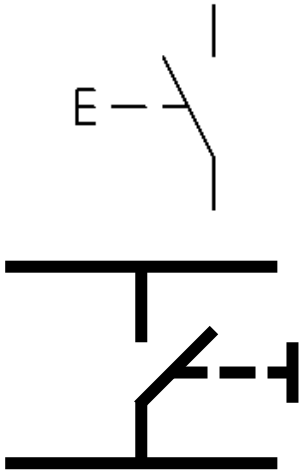

```
/*
  ZDAY 2014 - PORTO

  Piscar um LED

  Este código liga e desliga um LED
  associado à saída digital 11 a cada segundo
 */

// Variável que guarda a porta digital a utilizar
int led = 11;

// Parte do código para configurações
void setup() {
  // Definir a porta digital associada ao LED como saída
  pinMode(led, OUTPUT);
}

// Parte do código que executa infinitamente
void loop() {
  digitalWrite(led, HIGH);   // Liga o LED (ativa a porta digital)
  delay(1000);               // Aguarda 1 segundo (1000 ms)
  digitalWrite(led, LOW);    // Desliga o LED (desativa a porta digital)
  delay(1000);               // Aguarda 1 segundo (1000 ms)
}
```

Perna maior!

A resistência de 1 000 ohms determina a luminosidade do LED!

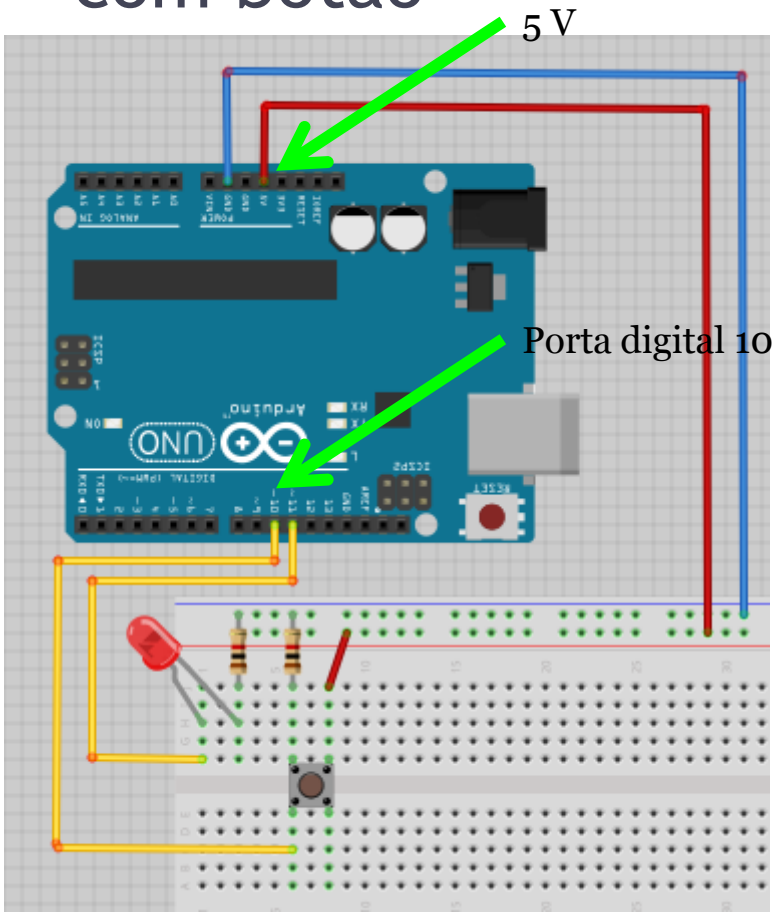
Botão

- Componente elétrico que é um interruptor: permite ou não a passagem de corrente;
- Quando pressionado, um contacto fecha (botão normalmente aberto);
- Diferentes tipos e tamanhos.



Coloca-se uma resistência ligada ao botão para evitar que se faça um curto-circuito ao carregar-se no dito!

Ligar/desligar LED com botão

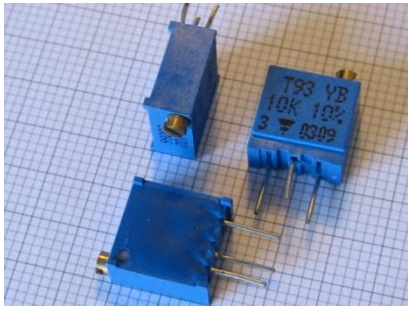


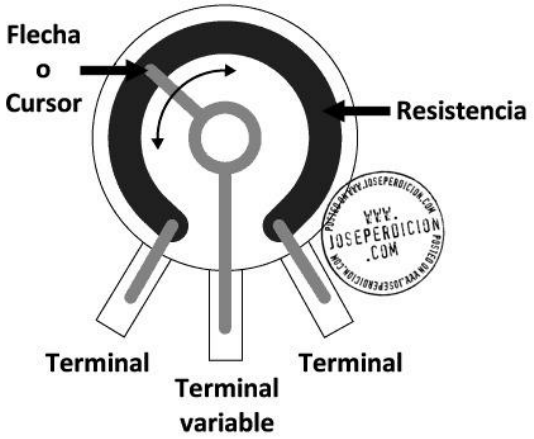
Quando carregamos no botão, fechamos um contacto, o que 'ativa' o botão, sendo esta ativação detetada pelo Arduino!

```
/*
  ZDAY 2014 - PORTO

  LED controlado com botão

  Este código permite que quando um botão é
  pressionado, um LED fique aceso
 */

int led =  11; // Porta digital associada ao LED
int botao = 10; // Porta digital associada ao botão

int estado_botao = 0; // Variável que guardará o estado do botão

// Parte do código para configurações
void setup() {
    // Definir a porta digital associada ao LED como saída
  pinMode(led, OUTPUT);
    // Definir a porta digital associada ao botão como entrada
  pinMode(botao, INPUT);
}

// Parte do código que executa infinitamente
void loop(){
  // Guarda o estado do botão na variável estado_botao
  estado_botao = digitalRead(botao);

  // Se o botão estiver premido
  if (estado_botao == HIGH) {
    // Liga o LED
    digitalWrite(led, HIGH);
  }
  // Caso o botão não esteja premido
  else {
    // Desliga o LED
    digitalWrite(led, LOW);
  }
}
```

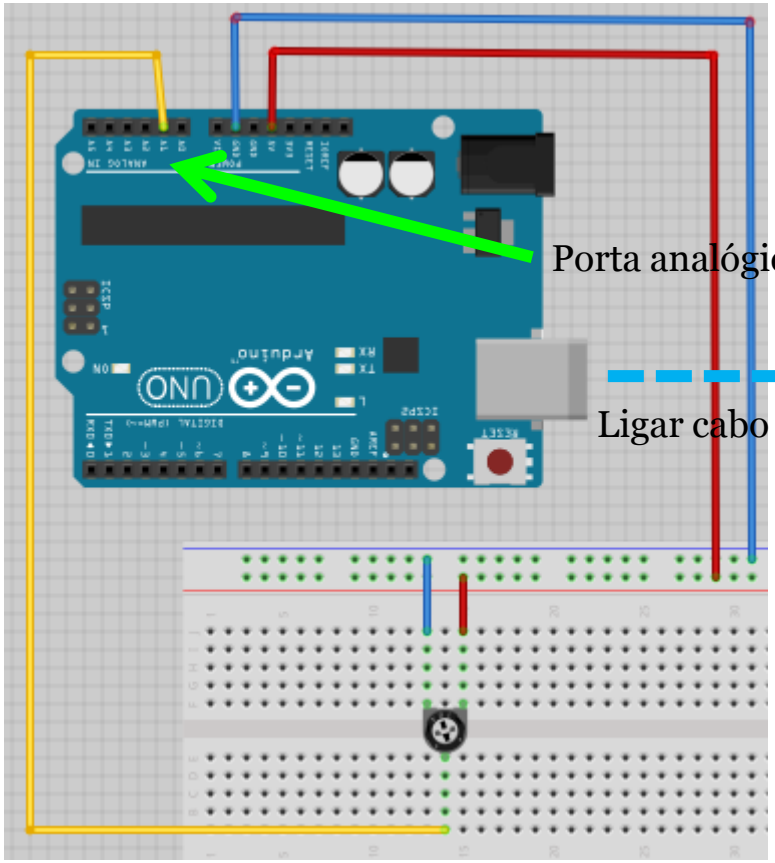
Potenciómetro

- Componente elétrico de resistência regulável, isto é, permite que o valor da sua resistência seja alterada;
- Diferentes tipos e tamanhos.





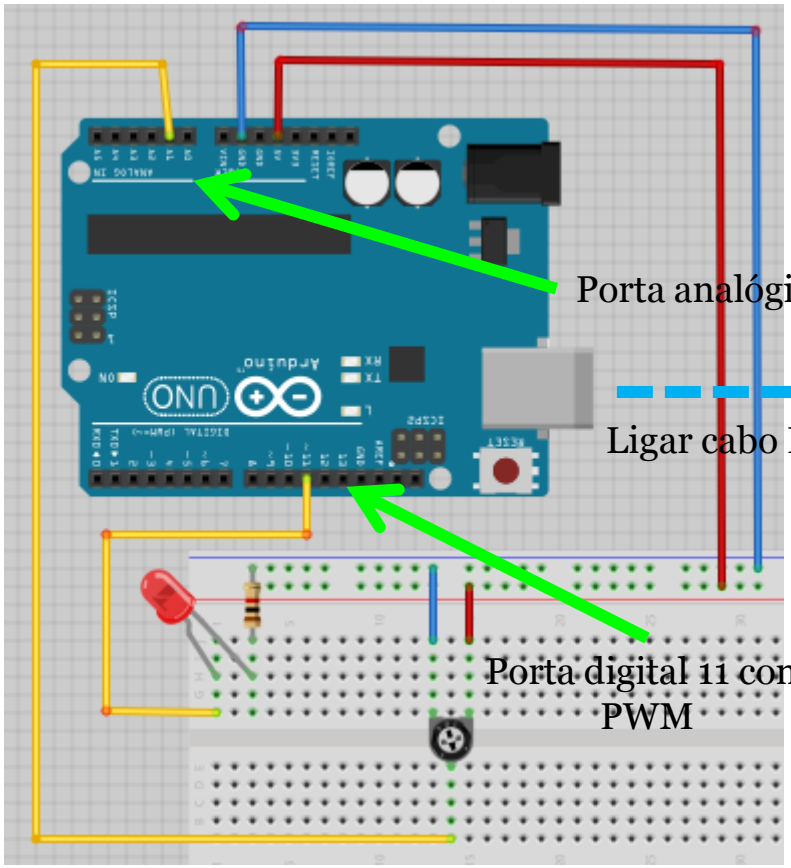
Ler um valor analógico e enviar para o PC




```
/*
  ZDAY 2014 - PORTO

  Leitura de um valor analógico
  e envio para o computador

  O valor analógico é dado pela variação
  de um potenciómetro
 */

int valor = 0; // Variável para guardar o valor lido
int potenciometro = 1; // Porta analógica associada ao potenciómetro

// Parte do código para configurações
void setup() {
  // Configuração da velocidade comunicação da porta série
  Serial.begin(9600);
}

// Parte do código que executa infinitamente
void loop() {

  // Leitura do valor da porta analógica 1
  // isto é, o estado do potenciómetro
  valor = analogRead(potenciometro);
  // Envio o valor lido para o computador
  Serial.println(valor);
  delay(500); // Aguarda meio segundo (500 ms)
}
```

Ao rodarmos o potenciómetro, alteramos a tensão de saída, sendo esta detetada pelo Arduino!

Controlar a luminosidade de um LED e enviar PC

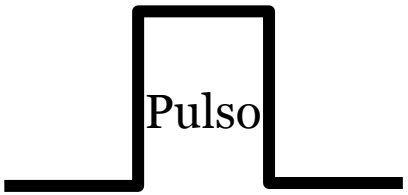



```
/*
  ZDAY 2014 - PORTO

  Controlo da lumiosidade de um LED

  O valor da lumiosidade é dado pela variação
  de um potenciómetro
 */

int valor = 0; // Variável para guardar o valor lido
int potenciometro = 1; // Porta analógica associada ao potenciómetro
int led = 11;          // Porta digital associada ao LED

// Parte do código para configurações
void setup() {
  // Definir a porta digital associada ao LED como saída
  pinMode(led, OUTPUT);
  // Configuração da velocidade comunicação da porta série
  Serial.begin(9600);
}

// Parte do código que executa infinitamente
void loop() {

  // Leitura do valor da porta analógica 1
  // isto é, o estado do potenciómetro
  valor = analogRead(potenciometro);
  // Envio o valor lido para o computador
  Serial.println(valor);
  // Alteração da luminosidade do LED
  // O valor devolvido pela leitura analógica é no máximo 1023
  // O valor máximo aceite pela seguite função é 255
  analogWrite(led, valor/4);
  delay(50); // Aguarda 50 ms)
}
```

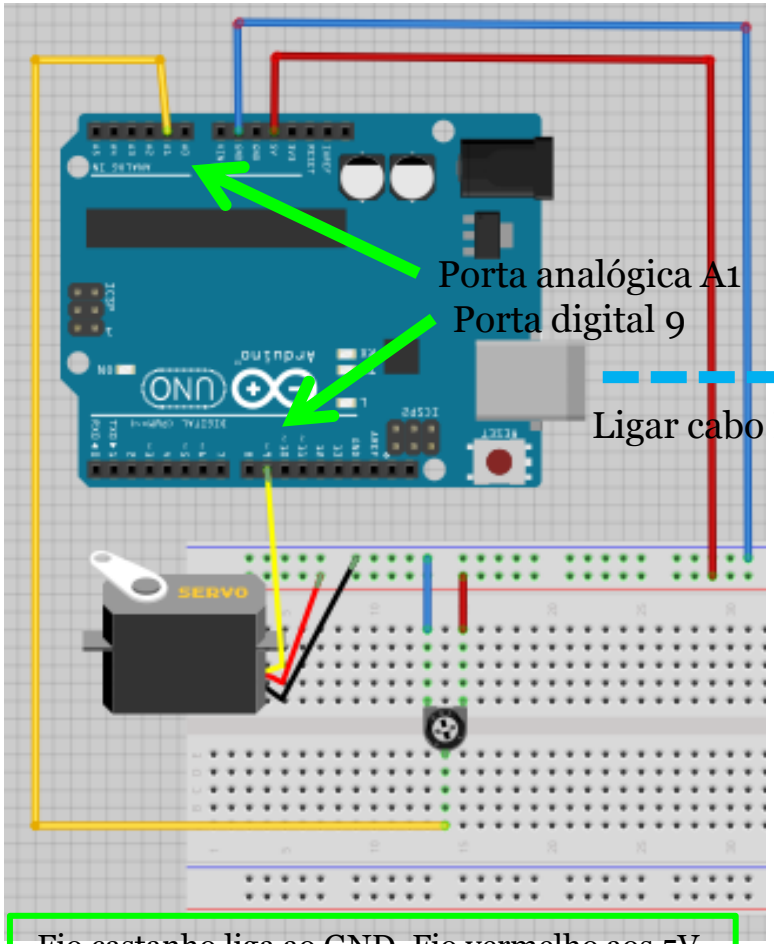
Ao rodarmos o potenciómetro, alteramos o valor lido pelo Arduino e consequentemente a luminosidade!

Servomotores

- Motor que tem rotação limitada;
- A posição é controlada pelo envio de um pulso variável consoante a posição pretendida;
- Mantém a última posição mesmo forçando-o;
- Possuem um torque muito elevado relativamente ao seu tamanho;
- Três fios de ligação: positivo, negativo e sinal;
- Diferentes tipos e tamanhos.



Controlar um servomotor



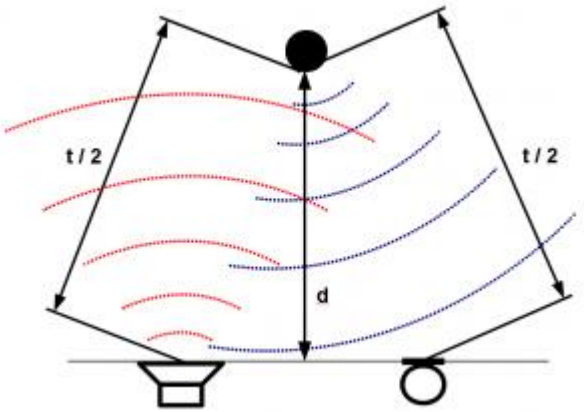
Fio castanho liga ao GND, Fio vermelho aos 5V, Fio laranja à porta digital 9

Ao rodarmos o potenciómetro, alteramos o valor lido pelo Arduino e consequentemente o ângulo do servomotor!

```
/*
  ZDAY 2014 - PORTO

  Controlo de um servomotor

  O ângulo do servomotor será dado pelo potenciómetro
 */

// Bilioteca que contém todas as funções a utilizar
#include <Servo.h>

// Variável do tipo Servo que está
// associada ao controlo do servomotor
Servo servomotor;

int pin_servomotor = 9; // Porta digital associada ao servomotor
int potenciometro = 1; // Porta analógica associada ao potenciómetro
int valor = 0; // Variável para guardar o valor lido do potenciómetro

// Parte do código para configurações
void setup() {
  // Associa a porta digital 9 ao servomotor
  servomotor.attach(pin_servomotor);
  // Configuração da velocidade comunicação da porta série
  Serial.begin(9600);
}

// Parte do código que executa infinitamente
void loop() {
  // Leitura do valor da porta analógica 1
  // isto é, o estado do potenciómetro
  valor = analogRead(potenciometro);
  // O valor devolvido pela leitura analógica é no máximo 1023
  // A seguinte função 'dimensiona' o valor para o intervalo pretendido
  valor = map(valor, 0, 1023, 0, 180);
  // Alteração da posição do servomotor
  servomotor.write(valor);
  Serial.println(valor);  // Envia o ângulo para o computador
  delay(20); // Dá tempo para o servomotor chegar ao ângulo
}
```

Sensor ultrassónico para medir distâncias

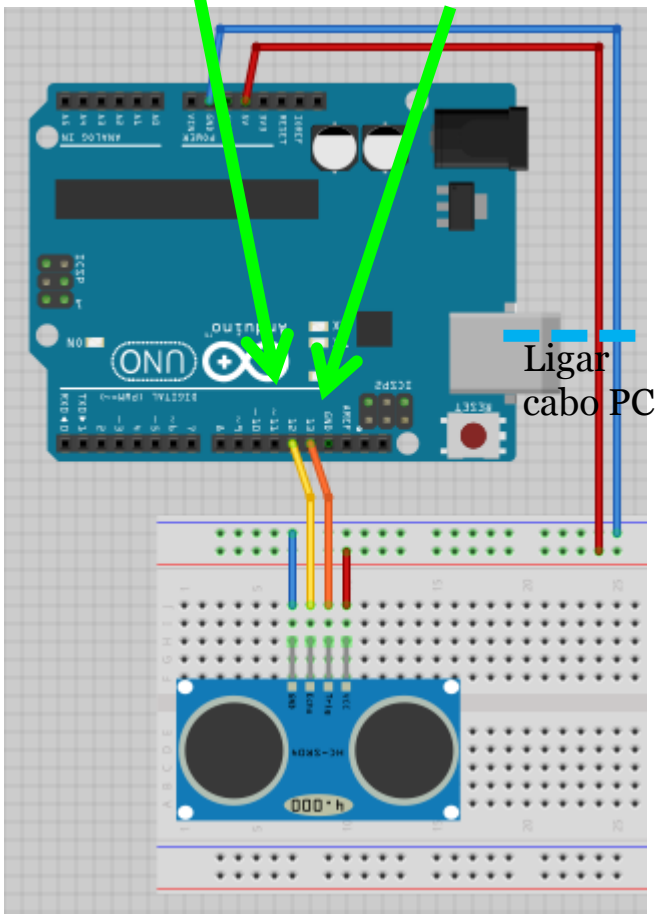
- Emite ondas ultrassónicas (inaudíveis) e aguarda pelo eco dessas ondas;
- Um pino recebe um pulso para iniciar leitura (*triger*);
- Outro pino devolve um pulso proporcional à distância lida (*echo*);
- Utilizando-se o valor de referência da velocidade do som, calcula-se a distância.



Medidor de distâncias ligado ao PC

Echo liga à porta 12

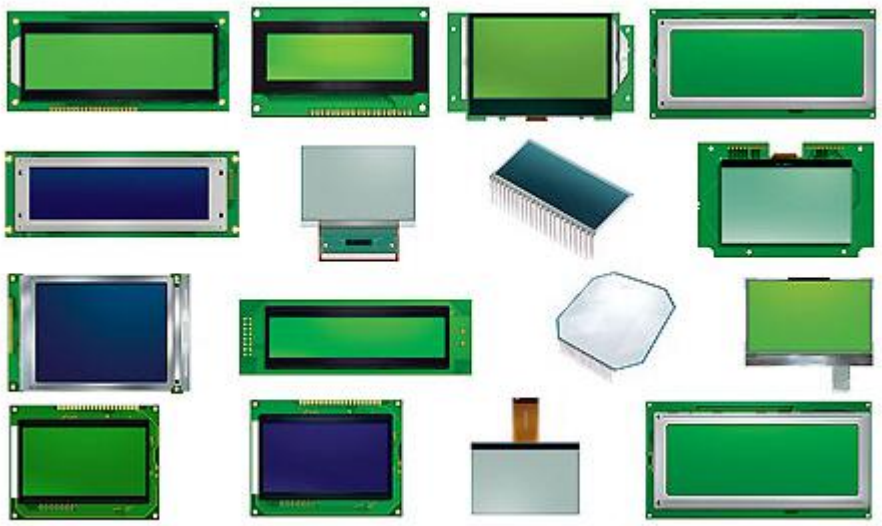
Triger liga à porta 13



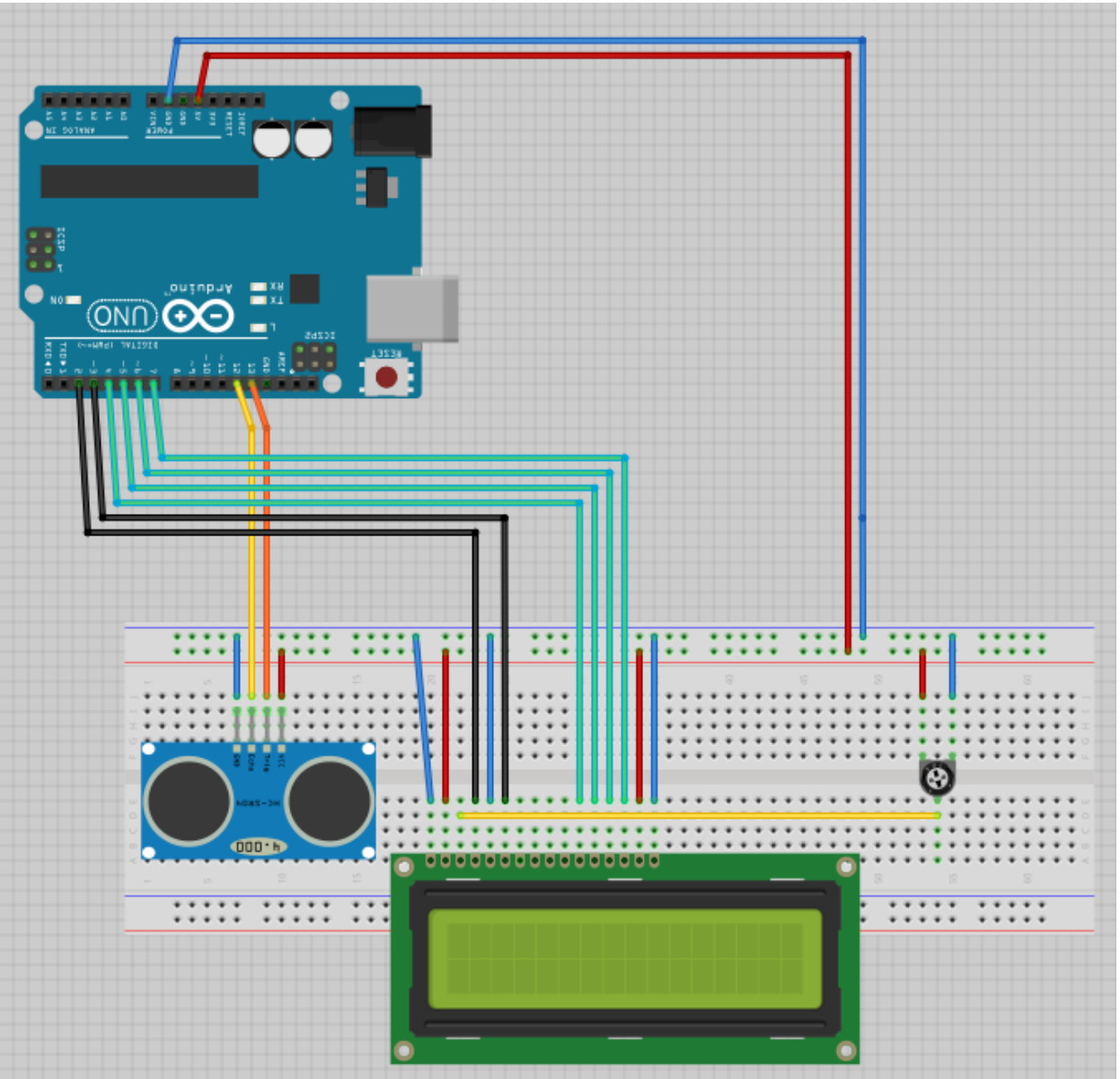

```
/*
  ZDAY 2014 - PORTO
  Medidor de distâncias
 */

int pino_ativa_sensor = 13; // Porta digital associada ao pino de ativação do sensor
int pino_resultado_sensor = 12; // Porta digital associada ao pino do resultado do sensor
long tempo = 0; // Variável para guardar o tempo de voo
int distancia = 0; // Variável para guardar a conversão para centímetros

void setup() {
  // Definir a porta digital associada à ativação do sensor como saída
  pinMode(pino_ativa_sensor, OUTPUT);
  // Definir a porta digital associada ao resultado do sensor como entrada
  pinMode(pino_resultado_sensor, INPUT);
  // Configuração da velocidade comunicação da porta série
  Serial.begin(9600);
}

// Parte do código que executa infinitamente
void loop() {
  // Para ativar o sensor, é necessário emitir um pulso, pelo que
  // colocamos a porta no estado LOW, colocamos depois como HIGH
  // e por fim novamente LOW
  digitalWrite(pino_ativa_sensor, LOW);  // Desativa saída digital
  delayMicroseconds(2);                  // Aguarda 2 microssegundos
  digitalWrite(pino_ativa_sensor, HIGH); // Ativa saída digital
  delayMicroseconds(5);                  // Aguarda 5 microssegundos
  digitalWrite(pino_ativa_sensor, LOW);  // Desativa saída digital

  // O sensor devolve um pulso proporcional ao tempo que passou entre
  // a emissão e a receção do sinal. A função seguinte conta quantos
  // microssegundos este pulso dura e guarda o valor na variável tempo
  tempo = pulseIn(pino_resultado_sensor, HIGH);
  // O som percorre 340 m/s a 15°C, o que corresponde aproximadamente a
  // 29 microssegundos por centimetro. Como o tempo contado é o de ida e
  // o de volta, tem de se dividir por dois
  distancia = tempo / 29 / 2;
  Serial.print(distancia); // Envia a distância obtida para o PC
  Serial.println(" cm");   // Envia o texto " cm" e nova linha
  delay(200); // Aguarda 200 ms
}
```

LCD (*liquid crystal display*)

- Permite a visualização de dados;
- Pino próprio para regulação do contraste (através de um potenciómetro);
- Permite comunicação utilizando-se 4 ou 8 fios dedicados para dados (D0 a D7) e 3 para controlo (E, R/W, RS);
- Maior parte possui luz de fundo (*backlight*) para utilização noturna;
- Diferentes tamanhos.



Medidor de distâncias portátil



Pino LCD	Destino
VSS	GND
VDD	5V
Vo	Pino médio potenciómetro
RS	Arduino porta digital 2
R/W	GND
E	Arduino porta digital 3
Do	Não ligado
D1	Não ligado
D2	Não ligado
D3	Não ligado
D4	Arduino porta digital 4
D5	Arduino porta digital 5
D6	Arduino porta digital 6
D7	Arduino porta digital 7
A	5V
K	GND

Medidor de distâncias portátil (continuação)

```
/*
  ZDAY 2014 - PORTO
  Medidor de distâncias portátil
 */
// Inclusão da biblioteca com as funções para controlar o LCD
#include <LiquidCrystal.h>

int pino_ativa_sensor = 13; // Porta digital associada ao pino de ativação do sensor
int pino_resultado_sensor = 12; // Porta digital associada ao pino do resultado do sensor
long tempo = 0; // Variável para guardar o tempo de voo
int distancia = 0; // Variável para guardar a conversão para centímetros

// Criação de uma variável do tipo LiquidCrystal e configuração dos pinos da
// seguinte forma (rs, en, d4, d5, d6, d7)
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {
  // Definir a porta digital associada à ativação do sensor como saída
  pinMode(pino_ativa_sensor, OUTPUT);
  // Definir a porta digital associada ao resultado do sensor como entrada
  pinMode(pino_resultado_sensor, INPUT);
  // Configuração do número de colunas e linhas do display
  lcd.begin(16, 2);
  // Coloca o cursor na posição inicial (coluna, linha) (começa 0)
  lcd.setCursor(0, 0);
  // Escreve o texto apresentado no display
  lcd.print("   DISTANCIA:   ");
}

// Parte do código que executa infinitamente
void loop() {
  digitalWrite(pino_ativa_sensor, LOW);  // Desativa saída digital
  delayMicroseconds(2);                  // Aguarda 2 microssegundos
  digitalWrite(pino_ativa_sensor, HIGH); // Ativa saída digital
  delayMicroseconds(5);                  // Aguarda 5 microssegundos
  digitalWrite(pino_ativa_sensor, LOW);  // Desativa saída digital
  tempo = pulseIn(pino_resultado_sensor, HIGH); // Leitura do tempo de voo
  distancia = tempo / 29 / 2; // Calculo da distancia em cm

  lcd.setCursor(0, 1);  // Coloca o cursor na coluna 0, linha 1
  lcd.print(distancia); // Envia a distancia lida para o display
  lcd.print(" cm  ");   // Escreve o simbolo de cm
  delay(200);           // Aguarda 200 ms
}
```