Electrónica e automação com Arduino

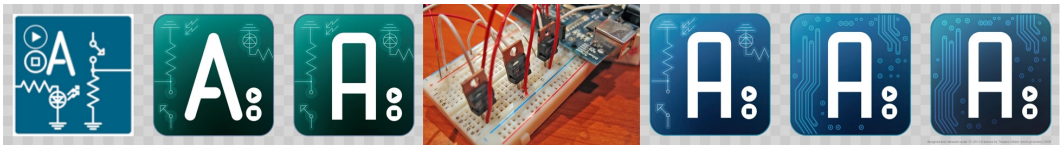
Shakib Shahidian, 2013

Versão provisória unicamente para utilização nas aulas

Electrónica e automação com Arduino

Índice

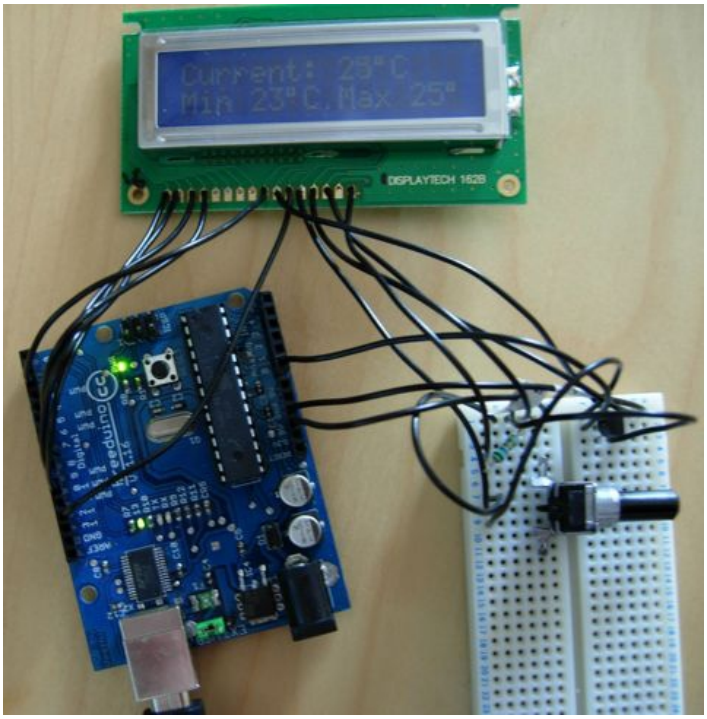
Ficha 0. Introdução à automação com Arduino
Ficha 1. O primeiro projecto: Programa Pisca
Ficha 2. Introdução aos Sensores
Ficha 2A. Sensores analógicos
Ficha 2B. Medição de voltagem e resistência
Ficha 2C. Algumas noções de electricidade
Ficha 3. Melodias
Ficha 3A. Utilização do LCD
Ficha 4. Entrada digital
Ficha 4A. Interface com um teclado
Ficha 5. Sensor de humidade ambiente
Ficha 5A. Fritzing
Ficha 6. Relógio em Tempo Real, RTC, e protocolo I^2C
Ficha 7. Automação e Comando
Ficha 7A. Automação com relés
Ficha 8. Medidor de distâncias
Ficha 9. Termómetro infravermelhos
Ficha 10. Comando de servo-motores com o Arduino
Ficha 10A. Motores de passo com o Arduino
Ficha 10B. Motores reversíveis com o Arduino
Ficha 11. Transponders
Ficha 12. LCD TFT
Ficha 13. Comando por infravermelhos
Ficha 14. GPS via porta série



Ficha 0. Introdução à automação com Arduino

O sistema Arduino é bastante económico e fácil de utilizar, permitindo uma aprendizagem interessante e rápida de electrónica e programação, estando especialmente vocacionado para a aprendizagem e produção de protótipos

O Arduino é uma plataforma de computação *open source* (livre) baseado num simples placa de entrada/saída e um ambiente de programação simples baseado no C++[1]. O Arduino foi especialmente desenvolvido para as pessoas que querem aprender electrónica e desenvolver projectos, mas que não são da área e não têm conhecimentos prévios de electrónica ou programação.

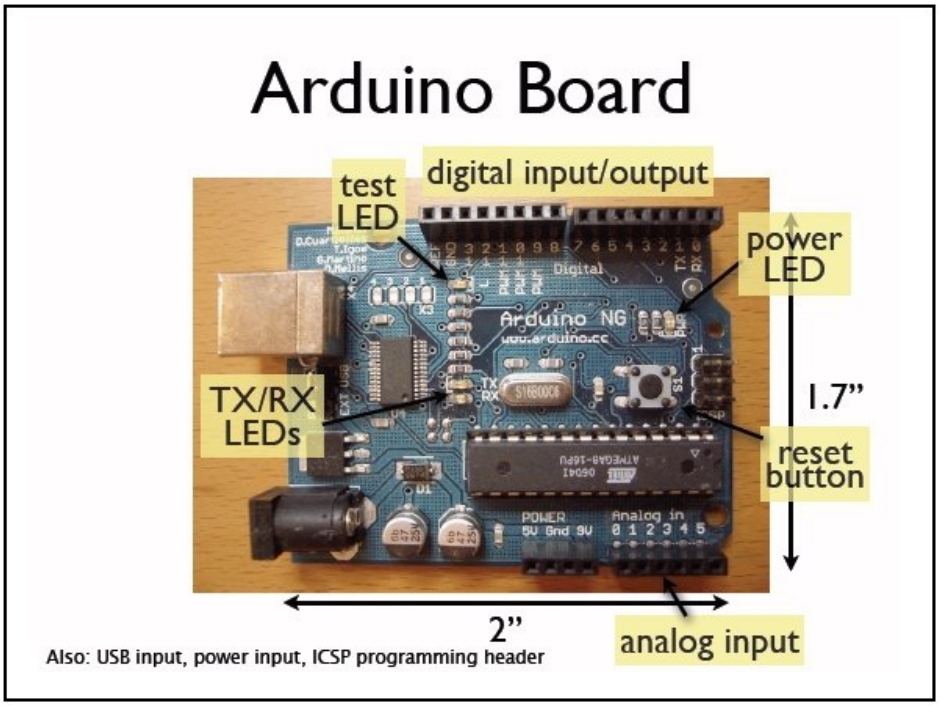


O Arduino pode ser utilizado para desenvolver objectos interactivos independents (termómetros, distribuidores de ração, commandos de porta, etc) ou pode ser ligado directamente ao seu computador e fornecer dados em tempo real para uma base de dados ou para visualização de avisos, como por exemplo a temperatura ambiente, ou as coordenadas da sua localização. Também pode ser facilmente ligado por internet ao seu computador ou ao seu *Tablet* ou telemóvel Android.

---

[1] Também utiliza bibliotecas Wiring e Processing

O ambiente de trabalho pode ser descarregado gratuitamente a partir de http://www.arduino.cc/en/Main/Software e instalado em computadores Windows, Mac e Linux.

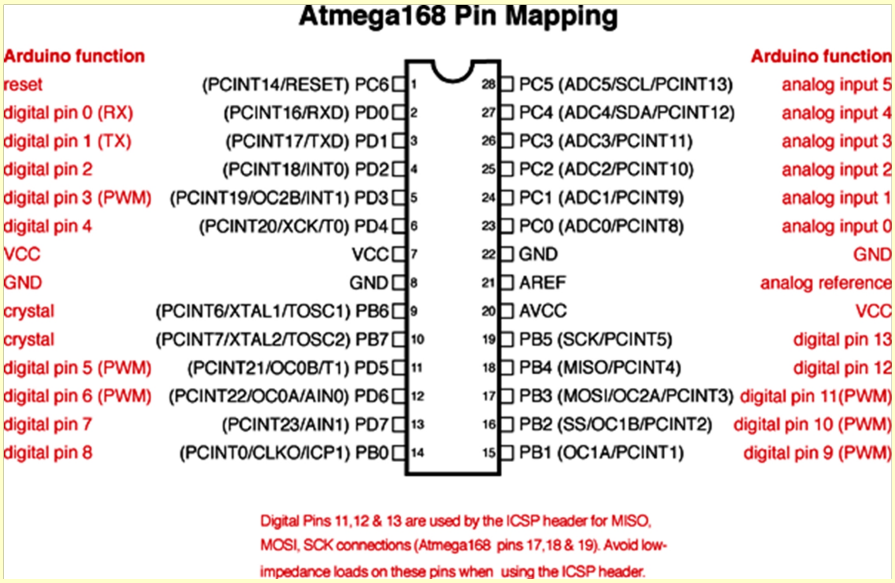


O Arduino é uma placa centrado num microcontrolador ATmega328. O ATmega tem 14 pinos de entrada/saída (das quais 6 podem ser utilizados como PWM- variação de voltagem), 6 entradas analógicas, uma porta USB, uma ficha de energia e um botão de reset[2]. A placa vem já com tudo o que é preciso para permitir o funcionamento independente do microcontolador. Na verdade, basta ligar o Arduino ao seu computador através do cabo USB e está pronto para trabalhar.

[2] Tem também um oscilador de 16MHz e uma ligação ICSP

Pontos de interesse:

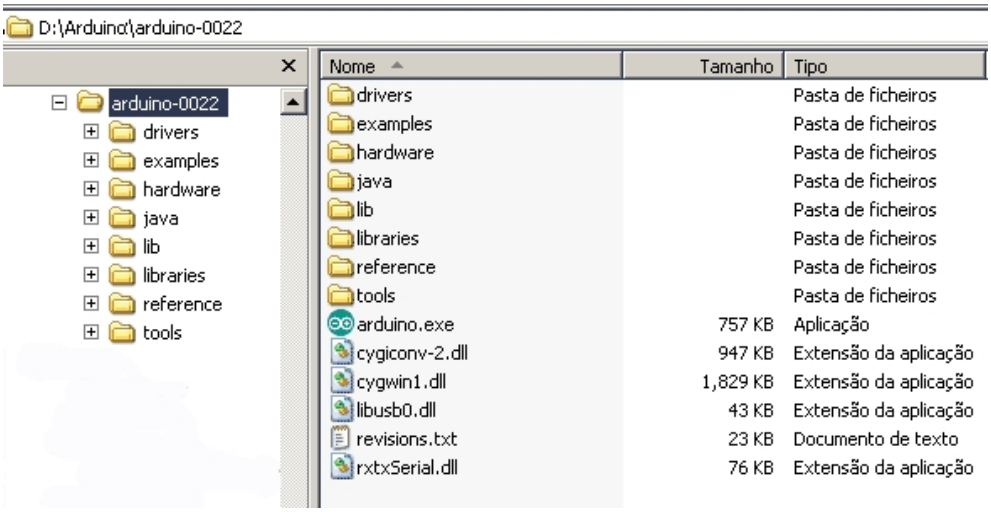
Aqui estão os pinos do Atmega328. Talvez mais a frente venha a precisar de dar uma olhada aqui para saber quais são efectivamente os pinos que precisa.



O Programa

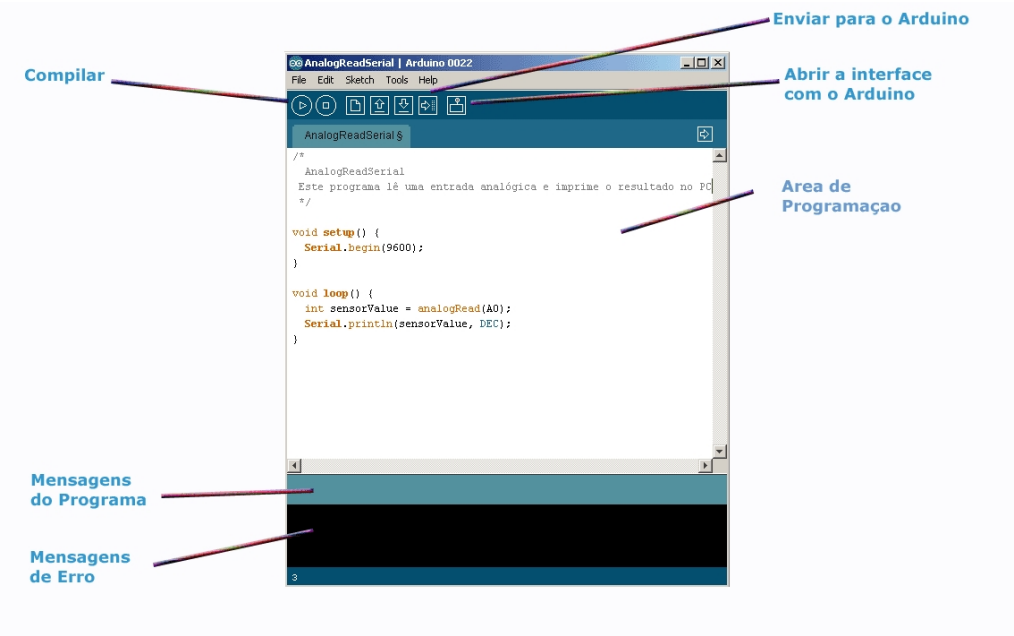
Depois de descarregar o programa (actualmente na versão 1) deve o graver no seu disco, num local que seja de fácil acesso, visto ter a necessidade de consultar frequentemente os ficheiros.

O programa instala os seguintes directories no seu computador:



O directório **drivers** inclui o driver para a porta USB.
O directório **exemplos** contém muitos exemplos práticos de como programar. Durante o presente curso iremos frequentemente buscar exemples neste directório.
O directório **reference** também é bastante útil pois traz um índice de todos os commandos do Arduino, bem como exemplos simples.
O directório **libraries** contém algumas programas importantes que já vem prontos a utilizar. Estes progamas, também chamadas de bibliotecas ou libraries vem simplificar o nosso trabalho, pois são normalmente códigos necessaries para comunicar com outros aparelhos, e assim basta incluir o seu nome no nosso programa e já não precisamos de nós preocupar mais com os pormenores desses códigos.

O Ambiente de trabalho propriamente dito é o ficheiro arduino.exe. Basta carregar dois vezes em cima dele para começar a trabalhar. O Ambiente de trabalho é constuido por 4 áreas:
Os botões de acesso rápido em cima, a área de programação em branco, uma faixa azul para mensagens e uma faixa preta para os mensagens de erro.



Os botões são, da esquerda para direita:

O botão **Verificar/Compilar** é utilizado para verificar que o código está correcto, antes de se poder enviar para o Arduino. Enquanto se está a programar é útil carregar regularmente neste botão por forma a se poder encontrar os erros e os ir corrigindo. É muito mais fácil corrigir individualmente os pequenos erros a media que se vai programando.

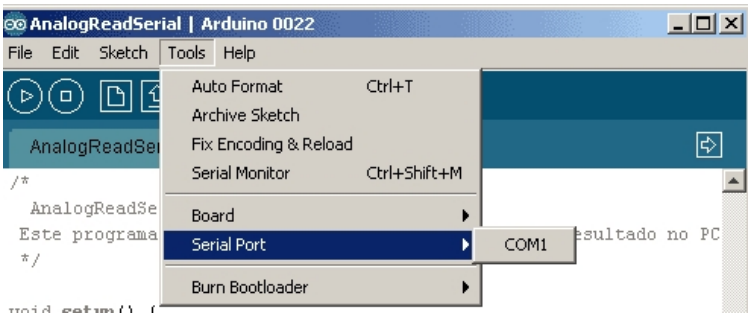
O botão **Stop** para o interface com o Arduino (Serial Monitor). Isso permite ler e analizar as mensagens recebidas com tempo.

O botão **Novo** cria um novo programa (chamado Sketch em Arduino) para poder começar a trabalhar. O que é simpático, é que não fecha o programa actual.

O botão **Abrir** permite abrir um programa existente. O arduino grava cada programa (Sketch) num directório separado, cujo nome é igual ao do programa. Caso indicação em contrário, os programas são gravados nos *Meus Documentos.*

O botão **Gravar** permite gravar o programa.

O botão **Enviar** manda o código para o Arduino. O programa primeiro faz um reset do Arduino, isto é, limpa o programa existente e o prepara para receber o novo programa. Enquanto o programa está a ser transmitido, podem-se ver duas pequenas luzes a piscar na placa Arduino. Antes de enviar, convém verificar se escolheu a placa Arduino correcto, e se a porta de comunicação também é a correcta.



Assim, que o programa estiver transmitido, o Arduino começa logo a executar o novo programa.

O botão **Interface** (Serial Monitor) abre uma interface série (RS232) com o arduino. Essa função é muito importante porque permite não só receber e enviar dados para o Arduino, como também verificar se o programa está a fucionar como desejado.

Pontos de interesse:

Um dos aspectos mais interessantes do Arduino é o facto de utilizar a porta USB. Essa porta existe hoje em quase todos os computadores e tem a grande vantagem de para além de comunicar, também alimentar directamente o Arduino sem ser necessário outra fonte de alimentação.

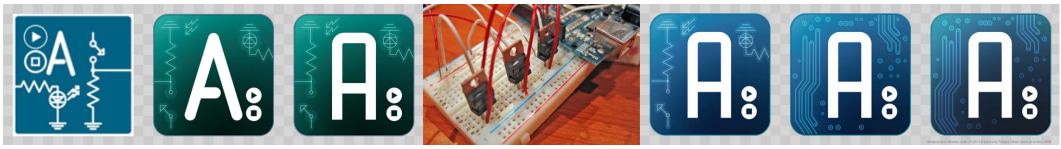
Então, podemos perguntar, porque é que no botão Interface (Serial Monitor) fala-se da comunicação série (tipo porta de rato ou RS232)?

Ora bem, o que acontece é que o USB utiliza um protocolo de comunicação relativamente complicado, enquanto que o velho RS232 tem um protocolo de comunicação bastante simples. Assim tanto o Ambiente de trabalho, como o Arduino "fingem" que estão a falar através de uma porta RS232, quando na verdade estão a utilizar uma porta USB.

É ter o melhor dos dois mundos: Facilidade de utilização e facilidade de programação.

Sites a consultar:

http://www.arduino.cc/
http://arduino.cc/en/Tutorial/HomePage



Ficha 1. O primeiro projecto: Programa Pisca

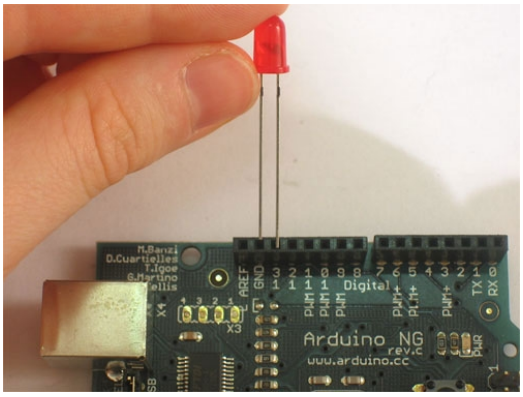
Vamos fazer o nosso primeiro projecto com o Arduino. Não se esqueça de primeiro instalar o programa no seu computador. Convém instalar num local de fácil acesso, porque será necessário aceder frequentemente ao directório do Arduino para buscar rotinas ou consultar o glossário.

Qualquer programa Arduino é constituído por pelo menos duas partes: Uma parte de declaração dos variáveis e funções, chamada "**void setup ()**", e depois uma segunda parte que corre sempre em círculo fechado, a chamada "**void loop ()**".
Qualquer uma das partes começa e acaba com chavetas: **{ }** (para escrever as chavetas deve utilizar a tecla Alt Gr e 7 ou Alt Gr e 0)

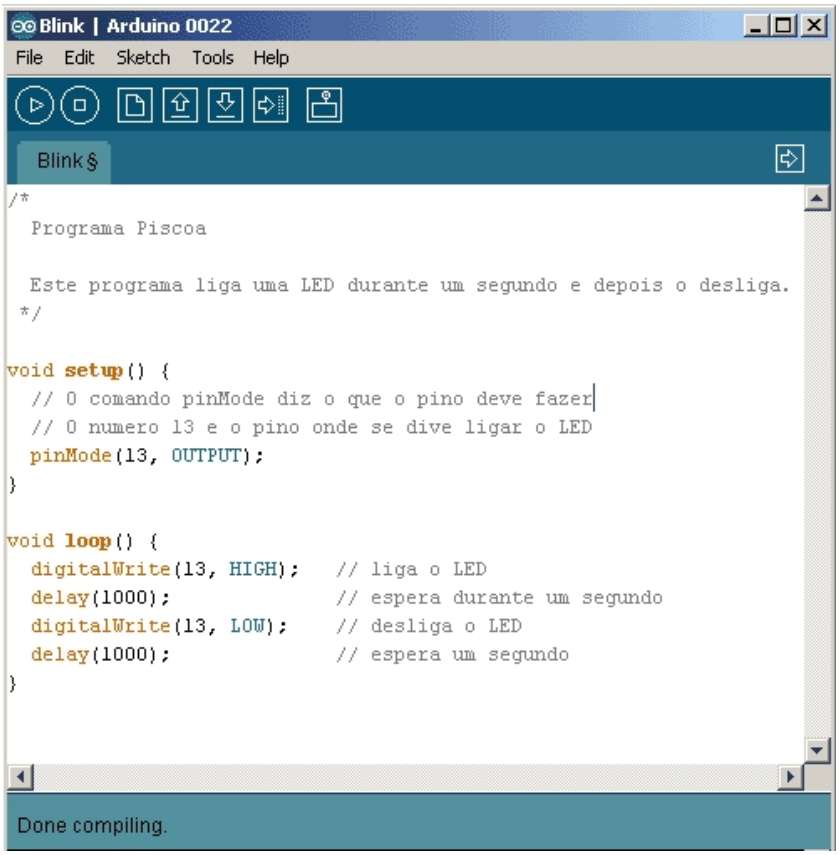
```
void setup() {

// esta parte corre uma só vez
}

void loop() {

// esta parte corre em contínuo
}
```

Projecto I

Vamos fazer um programa que liga um LED no pino 13, e um segundo depois o desliga e depois volta a ligar.
Para isso introduza um LED no pino 13 e GND (terra, negativo). O pino maior do LED deve estar no 13.



O programa será semelhante à figura seguinte:



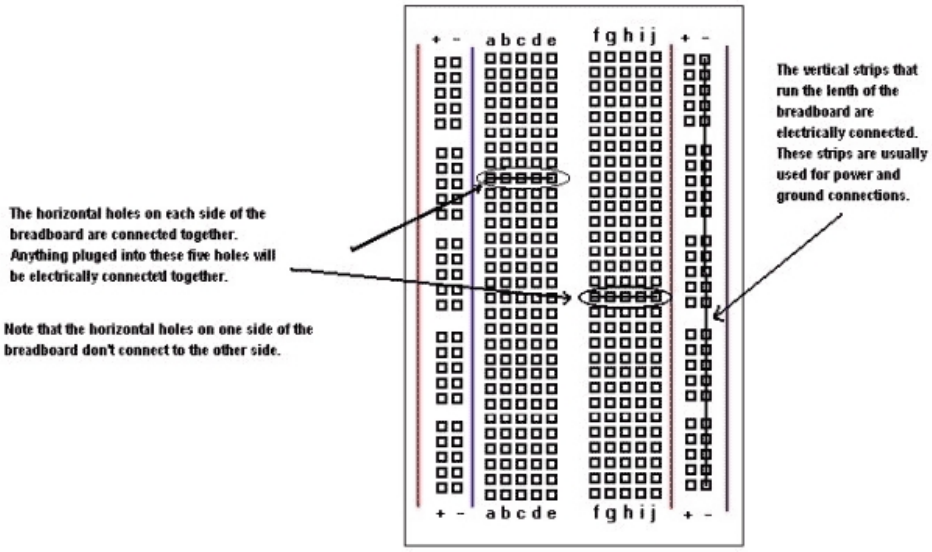
Pontos de interesse:

O comando **pinMode** diz ao microcontrolador qual deve ser a função do pino. Como se recorda, os pinos tanto podem ser de entrada como de saída. Neste exemplo, o comando pinMode(13, OUTPUT); diz que o pino 13 vai servir para saída, ou seja: vai dar um sinal de 5V, sempre que estiver ligado (HIGH) e 0V quando estiver desligado (LOW).

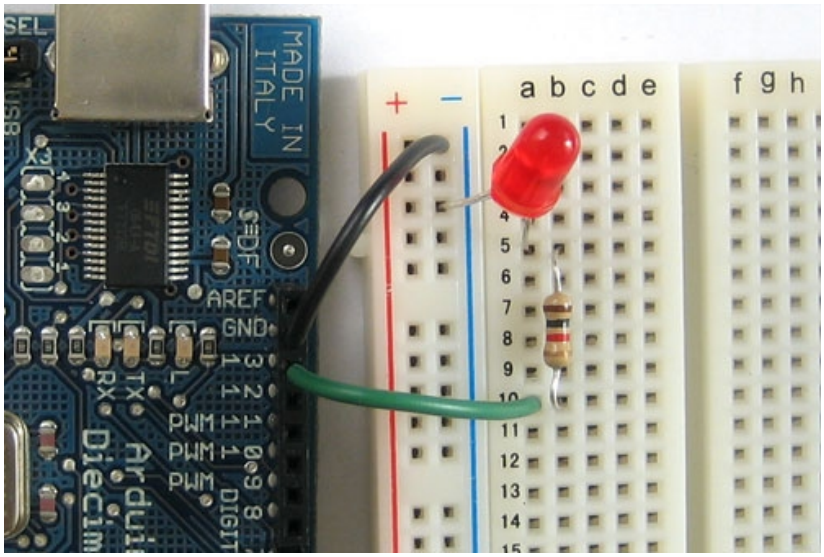
O comando **digitalWrite** escreve no pino. Neste caso o coloca no HIGH, ou seja a 5V, ou no LOW, ou seja 0V.

Projecto 2

Agora vamos fazer piscar dois LEDs em alternativa. Ora pisca um, ora pisca outro. Como só há um GND, vamos utilizar uma placa de ensaios (breadboard) para fazer a montagem:

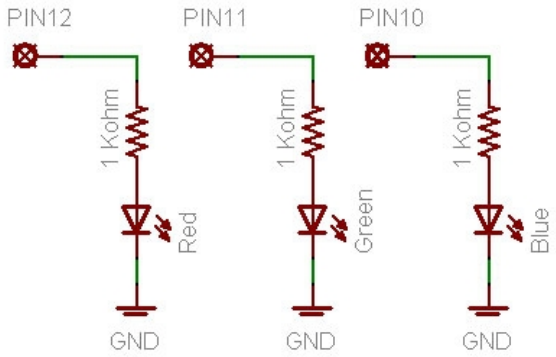


Relativamente ao circuito, é sempre necessário incluir uma pequena resistência no circuito por forma a evitar fazer curto-circuito e queimar o equipamento. Um dos pinos do LED é mais comprido. Esse deve ser ligado ao positivo, neste caso o 13 digital. O outro extremo deve ser ligado a terra "GND".



Projecto 3

Faça agora um semáforo: 3 LEDs que ligam sequencialmente. O verde durante 2 segundos, depois o laranja durante ½ segundo e depois o vermelho durante 2 segundos.



Pontos de interesse:

O comando **delay** obriga o processador a fazer uma pausa, em milisegundos.

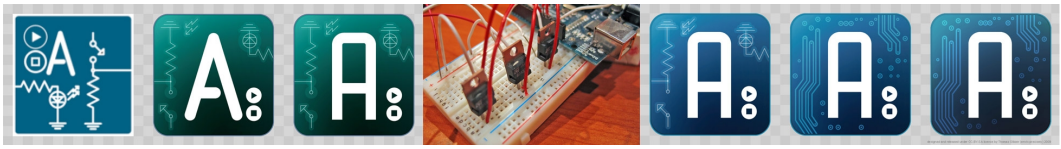
Projecto 4

Vamos fazer um caça-fantasmas. Neste projecto utiliza-se a electricidade estática dos dedos para ligar um LED. As ligações são bastante simples: ligue um LED no pino 13 e insere um cabo num dos pinos digitais, como por exemplo o 3. Deixe a outra extermidade do cabo solto. Esse cabo servirá para detectar a electricidade estática da mão e ligar o LED.

Não se esqueça de incluir uma instrução digitalRead para o pino 3.

Sites interessantes

http://www.practicalarduino.com/projects/medium/rfid-access-control
http://www.ladyada.net/learn/arduino/index.html



Ficha 2. Introdução aos Sensores

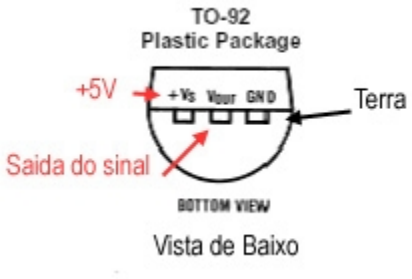
Introdução

O Arduino tem seis entradas analógicas[1], marcadas como A0 até A5. Estas entradas conseguem ler um sinal entre 0 e 5 volts (portanto um sinal analógico) e o converter em um número (ou seja algo digital). Esta capacidade de converter voltagem em um número é realizado por um componente chamado *Conversor Analógico Digital* ou *Conversor a/d*. No caso concreto do Arduino, o sinal 0-5V é convertido num número entre 0 e 1024. Assim, um sinal de 2,5V será apresentado pelo Arduino como 512.

Projecto 1 Termómetro digital

Neste exercício utiliza-se o comando AnalogRead () para ler a temperatura utilizando um circuito integrado LM35DZ. O termómetro LM35 traz já um circuito interno que converte a temperatura em voltagem. Assim, o termómetro produz 10 mV por ºC. Ou seja para uma temperatura de 15ºC, o LM35 dará uma saída de 150mV. Esse sinal é depois recebido pelo Arduino que o converte de novo em ºC. Como curiosidade se reproduz em anexo uma parte do manual do LM35

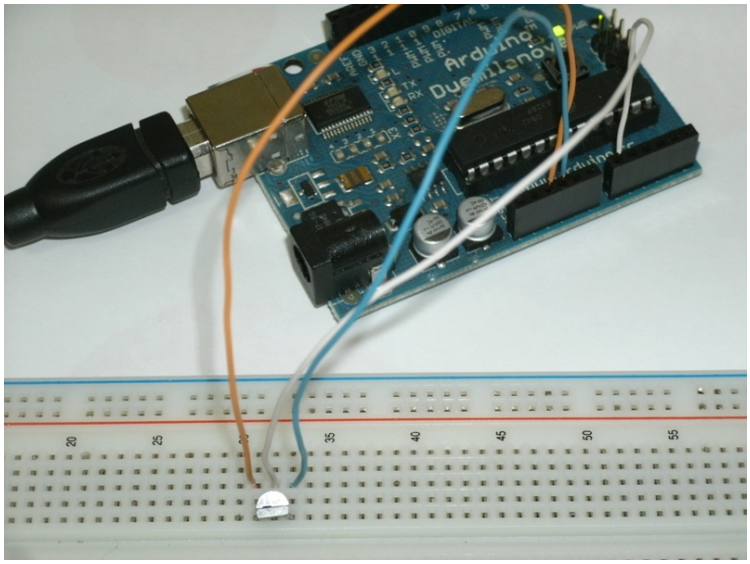
O LM35 tem três pernas, de acordo com a figura em baixo (a figura mostra o termómetro visto de baixo).



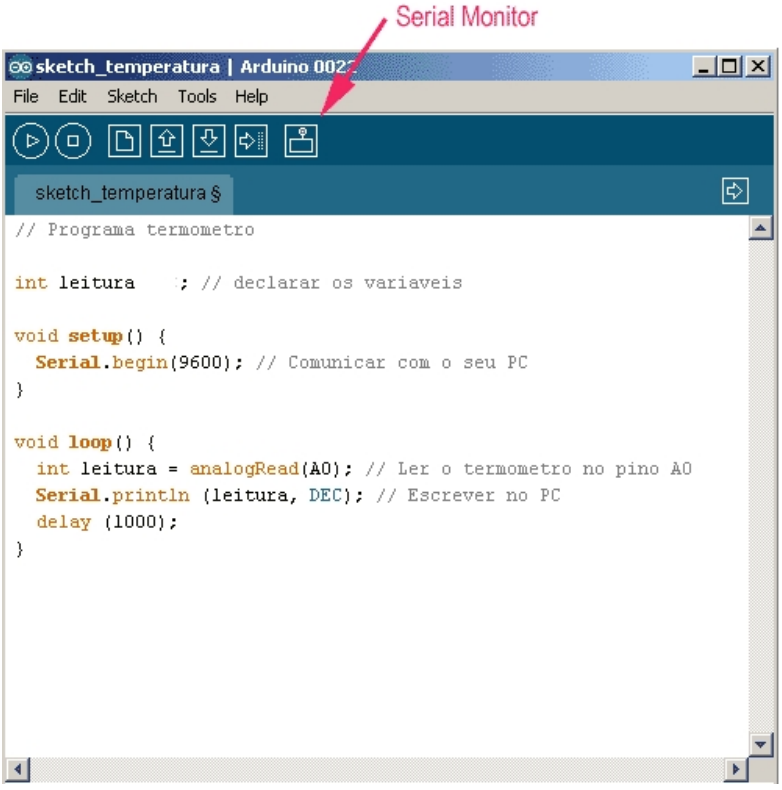
[1] Analógico em oposição à Digital. Digital quer dizer que só pode ter dois valores: 0 (desligado) e 1 (ligado). Analógico quer dizer que pode ter qualquer valor, tais como 3,14. Neste caso particular, pode medir valores entre 0V e 5V.

1ª parte

Nesta primeira parte vai-se construir um termómetro digital com o LM35. Ligar o LM35 ao Arduino, por forma a receber a alimentação 5V e estar ligado a terra. Também é necessário ligar a perna com o sinal de saída (Vout) a um pino de entrada analógica, como por exemplo o A0



Vamos também fazer um programa para ler o sensor LM35DZ e mostrar a leitura ao nosso computador.



Depois de ter tudo a funcionar, deve clicar em cima do botão *Serial Monitor* para começar a ver os resultados no seu computador.

 Pontos de interesse:

O comando **Serial.begin(9600)** diz ao Arduino que pode comunicar com o computador através da porta séria (na verdade é a porta USB). Os 9600 são a velocidade de comunicação, e corresponde a 9600 bits por segundo. Esse valor deve ser igual nos dois aparelhos que querem comunicar. Neste caso o Computador e o Arduino.

O comando **int** declara um variável. Isto quer dizer o variável, que neste caso se chama "leitura" será um numero **int**egral. Neste exemplo o variável leitura irá assumir o valor que for lido no pino analógico A0.

O comando **Serial.println**( ) escreve directamente para o computador, e os resultados podem ser observados no *Serial Monitor*.

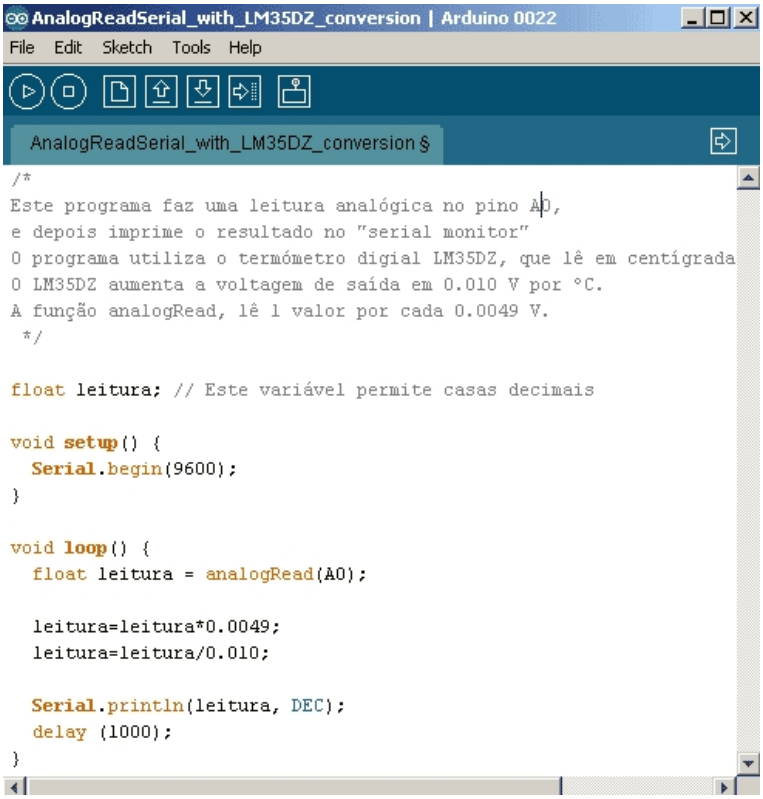
2ª Parte

Certamente já reparou que os valores obtidos não são temperaturas, mas sim leituras de voltagem expressos em números. Efectivamente ainda é preciso converter os valores lidos para ºC.

Vamos pensar um pouco:

Estamos a utilizar um termómetro digital LM35DZ, que lê em centígradas entre 0 e 100ºC. O LM35DZ aumenta a voltagem de saída em 0.010 V por ºC. Ou seja cada ºC é 0.010 V.
A função analogRead, converte 0-5V em números entre 0 e 1024, ou seja lê 1 valor por cada 0.0049 V.

Assim é necessário fazer conversões: Converter a leitura do Arduino para volts, e depois passar os volts do LM35 para Temperatura.



Agora já recebe e visualiza as leituras em ºC.

Poderá melhorar um pouco mais a apresentação dos resultados se incluir a seguinte instrução:

```
Serial.print("A temperatura da sala ");
```

Os operadores aritméticos utilizados no Arduino são:

= (operador de Atribuição)
+ (adição)
- (subtração)
* (multiplicação)
/ (divisão)
% (módulo)

Os operadores de comparação são:

= = (igual a)
!= (não igual a)
< (menor de)
> (maior de)
<= (menor ou igual a)
>= (maior ou igual a)

Pontos de interesse:

Para utilizar valores com casas decimais, é necessário utilizar um variável que permite números com casas decimais. No Arduino, isso é conseguido com variáveis do tipo "float" que podem ter 6-7 dígitos de precisão. O nome "float" quer dizer que a casa decimal é flutuante. Esses varáveis ocupam 4 bytes.

3ª parte

Uma das características mais importantes dos microcontroladores é a capacidade de decidirem. A decisão, ou a possibilidade de opção é conseguida com o comando **IF**, ou seja "se". O comando IF é seguido de uma condição. Se for verdade, o computador faz o que vem a seguir. Se não for, então passa a frente.

Utilize o comando IF (condição) para que acenda uma luz verde quando a sala está fria (por exemplo 15ºC) e uma luz vermelha quando está quentes (por exemplo mais de 20ºC).

```
if (variavel > 50)
{
Faz o que esta aqui;
}
```

4ª parte

Faça agora um programa que acende uma luz verde quando a T é inferior à 20ºC, uma luz amarela quando está entre 20 e 30ºC, e uma luz vermelha quando está acima dos 30ºC.

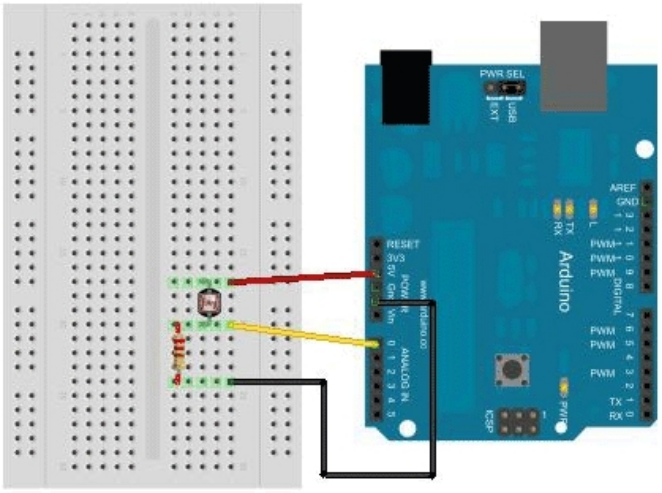
Quando acabar não se esqueça de guardar o programa porque vai-lhe dar jeito no futuro.

Projecto 2

O LDR é uma resistência cujo valor varia de acordo com a luz incidente, permitindo medir a luminosidade do local. Faça um

projecto em que quando a intensidade luminosa é baixa, acendem as lâmpadas (LEDs neste caso).

Para poder ler a resistência vai necessitar de fazer um *divisor de potência* de acordo com a figura seguinte (Ver também a ficha 2A):

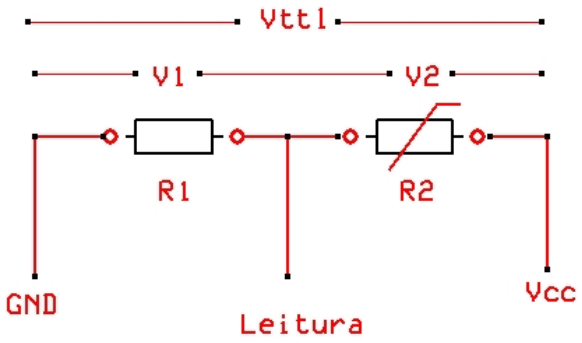


Projecto 3

Também é possível construir sensores de temperatura, utilizando simples termisteres. Os termisteres são resistências, cujo valor varia de acordo com a temperatura.

Pode utilizar a placa de ensaios (bread board) para montar o seu sensor. Deve instalar um termister e uma resistência em série entre o 5V e a terra. A leitura é feita no meio, tal como na figura, ligando um fio directamente à uma porta analógica do Arduino.

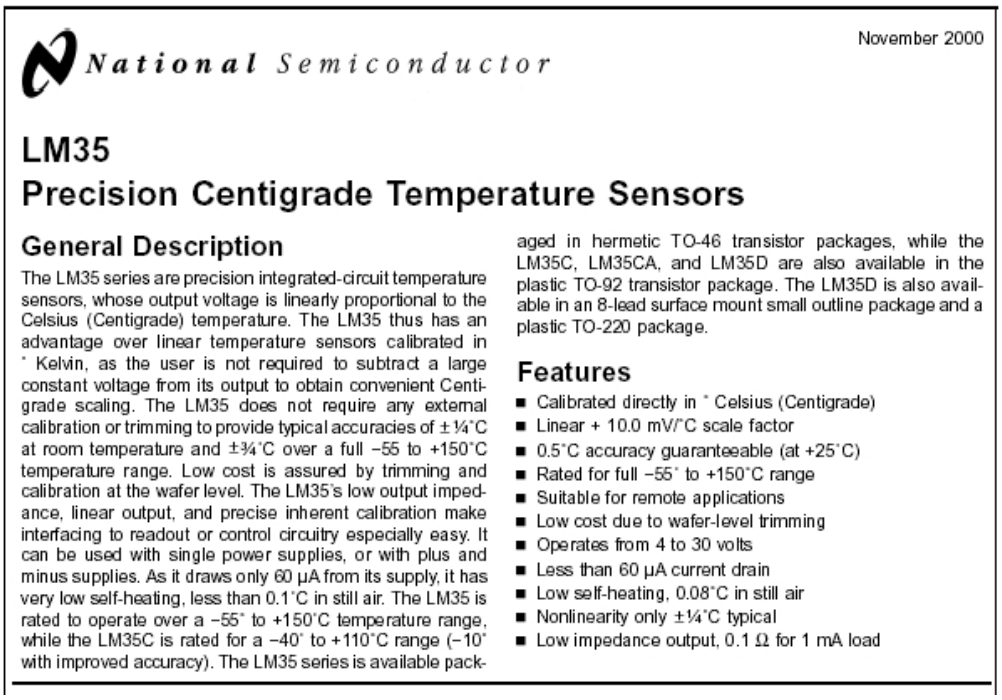
Tenta agora escrever o programa necessário para apresentar valores da temperatura ou pelo menos da resistência do termister no *Serial Monitor*. Não se esqueça do esquema de um divisor de potência e das equações em baixo (ver também a ficha 2A).



$$I = \frac{V}{R} \qquad\qquad I = \frac{V_{ttl}}{R_{ttl}} = \frac{V_1}{R_1} = \frac{V_2}{R_2}$$

$$R_{ttl} = \frac{V_{ttl} R_1}{V_1} \quad \text{e} \quad R_2 = R_{ttl} - R_1 \quad \text{que são valores conhecidos.}$$

Data Sheet



National Semiconductor

November 2000

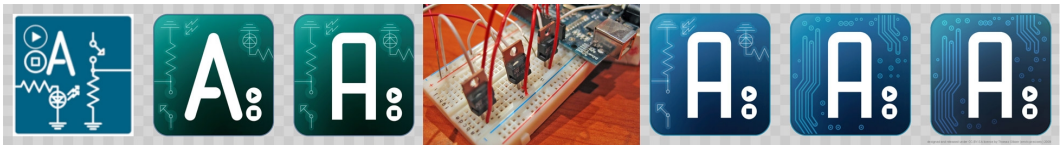
LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load



Ficha 2A. Sensores analógicos

Sensores são peças fundamentais na automação. Permitem aos sistemas electrónicos medirem os fenómenos físicos ou químicos e depois agir de acordo com os valores observados.

Qualquer sensor é constituído em primeiro lugar pelo elemento que mede o fenómeno em causa. Esse elemento deve sofrer uma alteração com variação do fenómeno a medir. Por exemplo o cabelo humano estende com a humidade e durante décadas foi utilizado para medir a humidade do ar. Adicionalmente o sensor deve possuir algum elemento que possa converter essa alteração num sinal eléctrico mensurável, para que possa ser medido pelo microcontrolador.

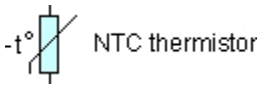
Nesta ficha vamos ver quais são os princípios fundamentais de funcionamento de sensores:

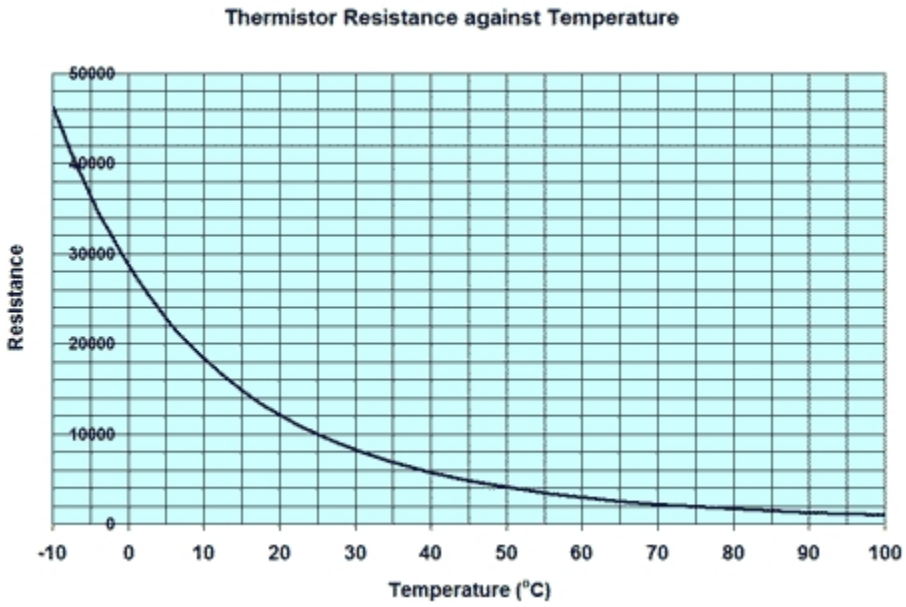
Resistência

A resistência de alguns materiais varia de acordo com alguns fenómenos físico-químicos, tais como a temperatura, humidade e luz. Aproveitando essa variação é possível construir sensores simples e económicos.

Termistors

O termistor é uma resistência, em que o seu valor muda de acordo com a temperature ambiente. Utilizando este princípio pode-se facilmente construir termómetros digitais.

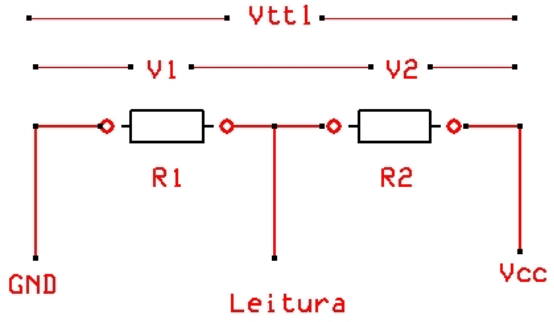




No entanto, os microcontroladores não conseguem medir a resistência, mas sim voltagem. Para isso pode-se fazer um pequeno circuito para converter a resistência em voltagem. Como se lembra:

$$V = IR$$

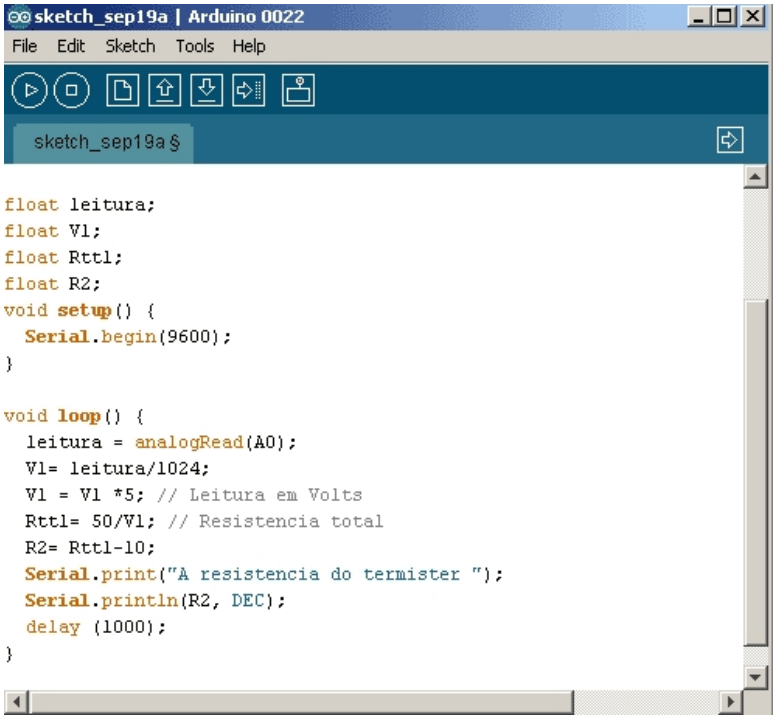
Pode-se assim pensar no seguinte circuito, que é conhecido como um divisor de potência. Aqui o R1 será uma resistência de valor conhecido, enquanto que o R2 será o termister. A leitura é realizada no meio do circuito, onde a variação da resistência irá alterar a voltagem.



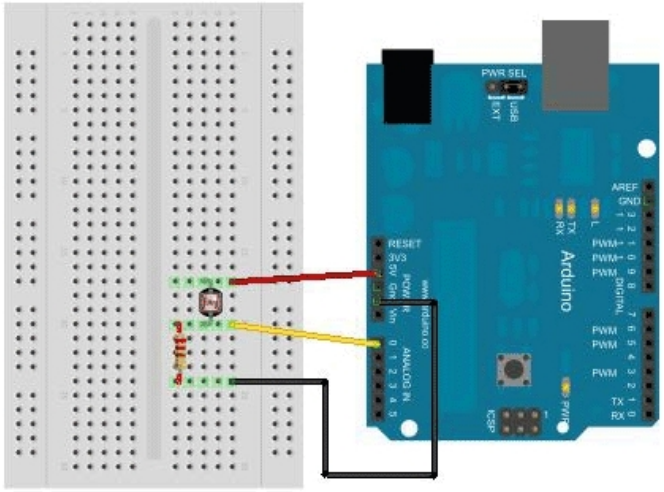
Verifica-se que a Intensidade de corrente é igual entre o Vcc e a terra (GND). Assim, podemos colocar a intensidade em evidência e temos:

$I = \frac{V}{R}$ e portanto: $I = \frac{V_{ttl}}{R_{ttl}} = \frac{V_2}{R_2} = \frac{V_1}{R_1}$ e também: $\frac{V_{ttl}}{R_{ttl}} = \frac{V_1}{R_1}$

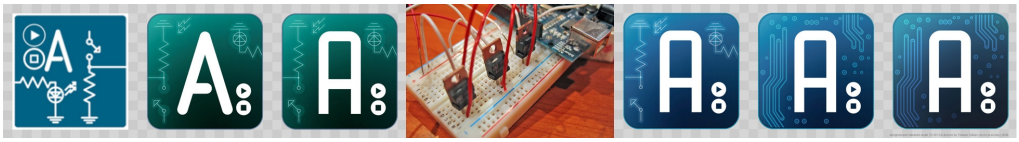
pelo que: $R_{ttl} = \frac{V_{ttl} R_1}{V_1}$ e $R_2 = R_{ttl} - R_1$ que são valores conhecidos.



A resistência de um termister NTC diminui a medida que a temperatura aumenta, de acordo com o gráfico em cima, pelo que terá de fazer uma equação de calibração.



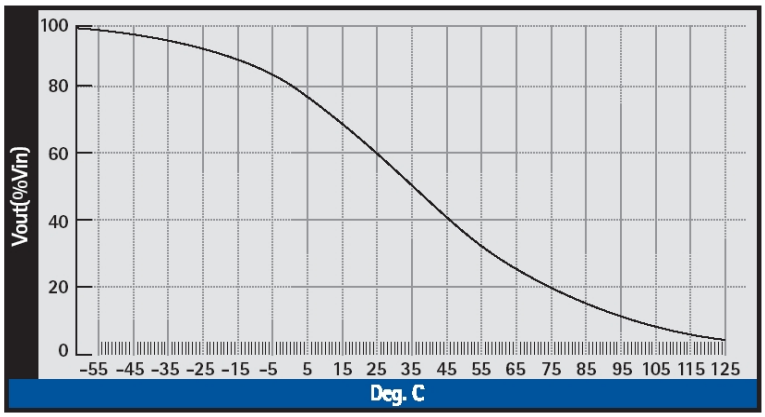
O mesmo esquema poderá ser utilizado para medir humidade, luminosidade e outros parâmetros para os quais existam sensores baseados na alteração de resistência.



Ficha 2B. Medição de voltagem e resistência

Os sensores comerciais, apesar de toda a sua sofisticação, são normalmente constituídos por um elemento muito simples cuja resistência varia em função da grandeza a medir. É o caso típico dos sensores de temperatura feitos a base de termisters, e sensores de luz construídos à volta de um LDR (Resistência Dependente da Luz).

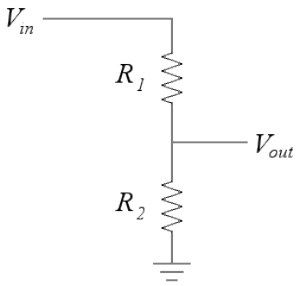
Por exemplo na imagem seguinte pode-se ver a variação da voltagem (%Volt saída, em função da voltagem de entrada) típico de um termister em função da temperatura. Como se vê, existe uma proporcionalidade directa entre a voltagem e a Temperatura. Efectivamente na gama de temperaturas que nós interessa, a relação é bastante linear.

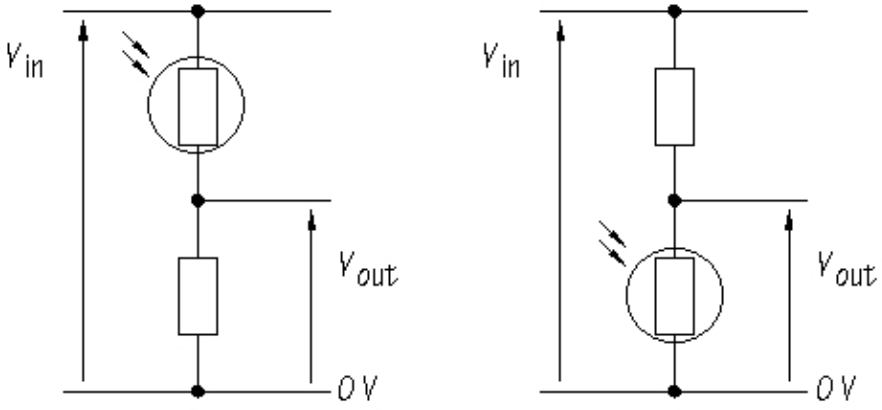


Um sensor de luz

Para se poder medir a variação de voltagem num sensor, é necessário liga-lo em série com uma resistência de valor conhecida.

Efectivamente há só duas formas de construir o sensor: ou se coloca o sensor (LDR neste caso) na parte de cima, ou o LDR na parte de baio.



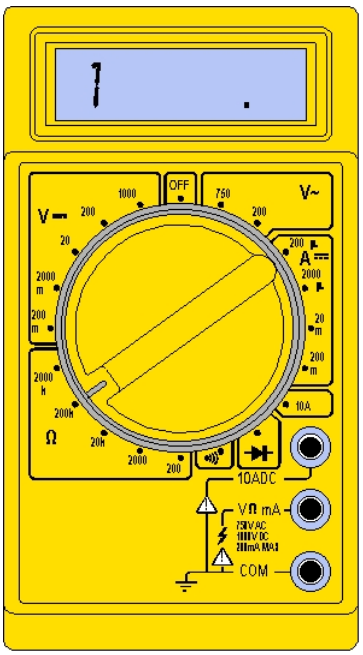


Como se lembra a fórmula para o cálculo do V_{out} é:

$$V_{out} = \frac{R_{\inf erior}}{R_{\inf erior} + R_{\sup erior}} x V_{in}$$

Se tiver uma bateria de 12 V (V_{in} = 12V), e a resistência for de 500 A, e o LDR apresentar uma resistência de 250 A, qual será o valor do V_{out}, em cada caso?

Medir a resistência utilizando um multimetro



Primeiro colocar o multimetero numa escala de resistência:

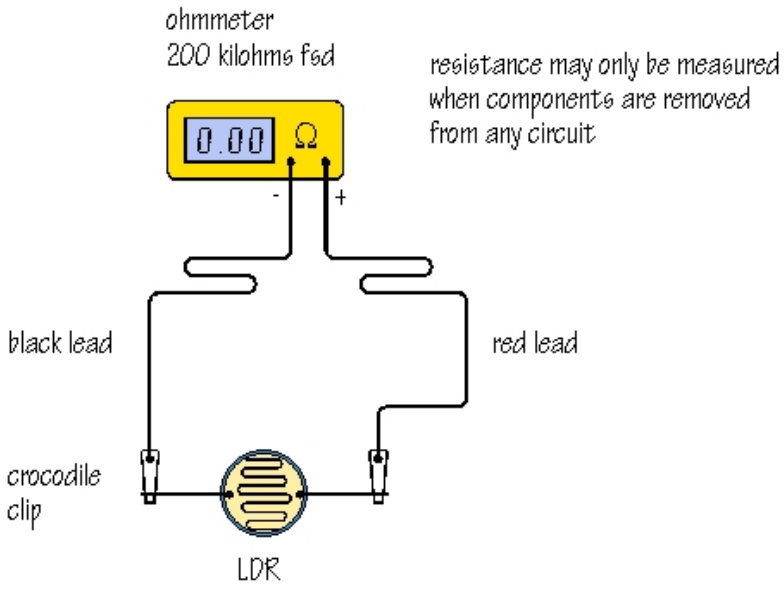
Vamos tentar primeiro com uma escala de 0 a 200 kΩ (se os valores estiverem fora deste limite, terá de alterar a escala). Vamos agora tentar medir a resistência do LDR.

Quando não está nada ligado, a écrã mostra o valor de 1.

Inserir a sonda negra do voltimetro na ficha COMum e a sonda **vermelha** na ficha VΩmA . O que acontece quando os sensores se tocam?

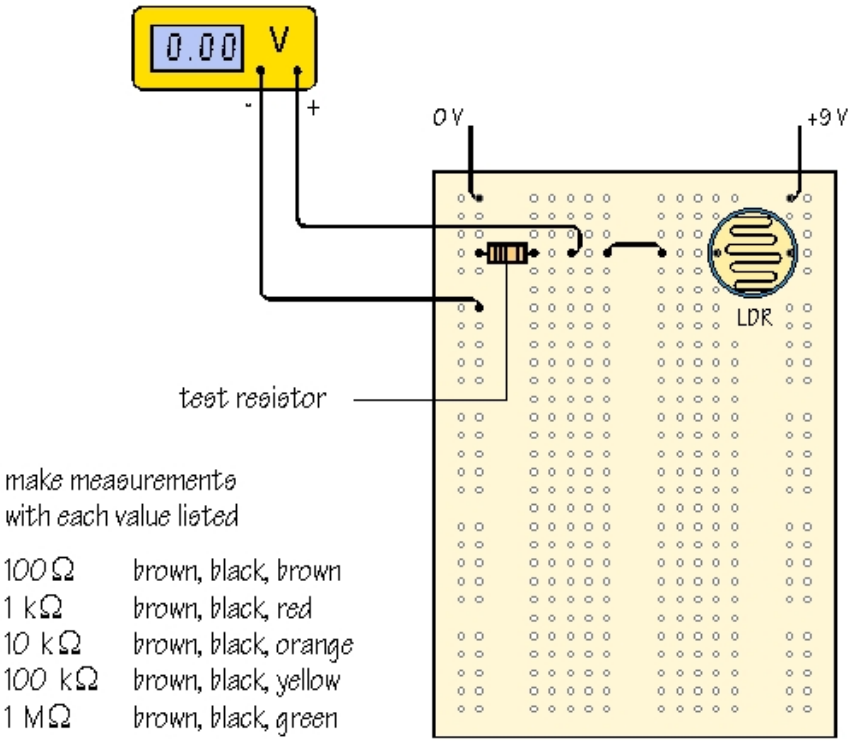
Se molhar os dedos, pode medir a resistência da sua pele) talvez tenha de mudar para a escala 2000 kΩ.

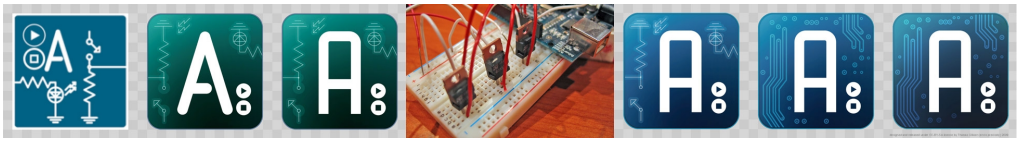
Agora pode medir a resistência do LDR, exposto a diferentes graus de luminosidade. Quando o LDR está à sombre, o que é que acontece à sua resistência?



Circuito para o sensor da luz

Construa um circuito para o sensor da luz, e experimente com diferentes resistências.





Ficha 2C. Algumas noções de electricidade

1. Intensidade de Corrente eléctrica

Um material será um bom condutor se tiver electrões livres. No seu estado natural, os electrões livres estão em movimento desordenado com actividade em todas as direcções, logo, todos os pontos da secção transversal metálica têm o mesmo potencial eléctrico.

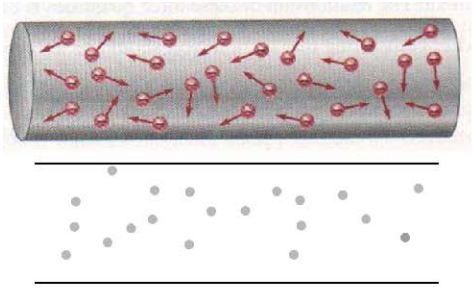


Figura 1 – Movimento desordenado dos electrões

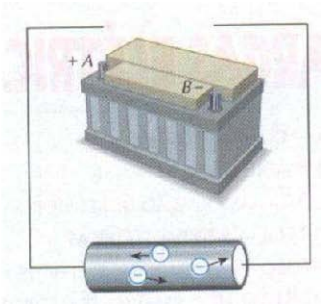


Figura 2 – Condutor desligado do gerador

Quando se liga a um gerador eléctrico, o comportamento dos electrões altera-se. Começa a haver um movimento ordenado de electrões de um extremo para o outro. Esse movimento ordenado de electrões é chamado corrente eléctrica e a sua grandeza é a **Intensidade**.

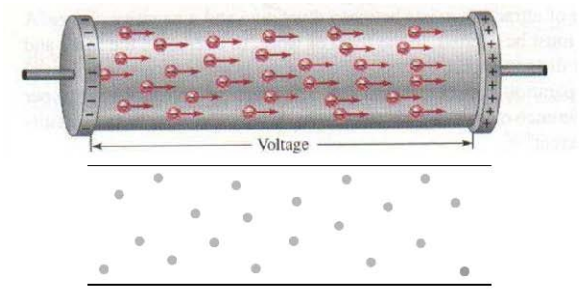


Figura 4 – Movimento ordenado dos electrões

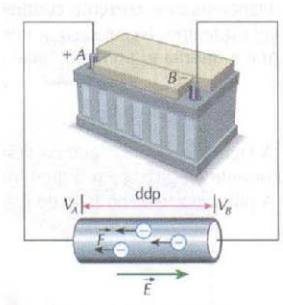


Figura 5 – Condutor ligado do gerador

Um **Ampere** representa o fluxo de uma carga de 1 Coulomb por segundo

$$A = \frac{C}{s}$$

O Coulomb é a quantidade de electrões que atravessa o circuito. É constituído por 6,25 x 10^{18} electrões. O coulomb (símbolo: C) é a unidade da carga eléctrica no Sistema Internacional de Unidades (SI)

Assim a corrente se deve ao movimento de electrões do polo negativo para o positivo[1].

A equação anterior também pode ser expressa da seguinte forma:

$$I = \frac{\Delta Q}{\Delta t}$$

em que a Intensidade, ***I***, é igual à Quantidade de carga eléctrica, ***Q***, que passa pela secção, num determinado intervalo de tempo, ***t***.

Utilizando uma outra definição, um Ampere é a corrente eléctrica constante que, se mantida entre dois fios condutores rectos e infinitos ou com secção transversal desprezível, afastados por uma distância de um metro no vácuo, produz a força por metro de fio equivalente a $2x10^{-7}$ N.
– A unidade Ampere tem como submúltiplos:
Miliampère (mA) → 1 mA = $1x10^{-3}$ A
Microampère (μA) → 1 μA = $1x10^{-6}$ A

Exercício

Quantos electrões passam por um cabo se a intensidade de corrente é de 8 A?

A Carga eléctrica é o movimento ordenado dos electrões ao longo de um condutor

A Carga pode ser calculado através da seguinte equação:

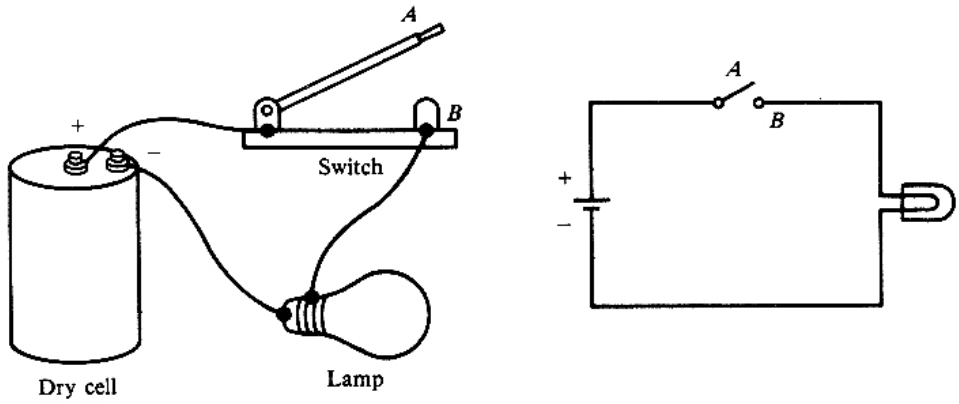
$$Q = I\ t$$

---

[1] Iniciamente pensava-se que a corrente ia do pólo positivo para o negativo, e por razões práticas, vai-se continuar a utilizar o sentido convencional: do positivo para o negativo.

2*. Esquemas eléctricos*

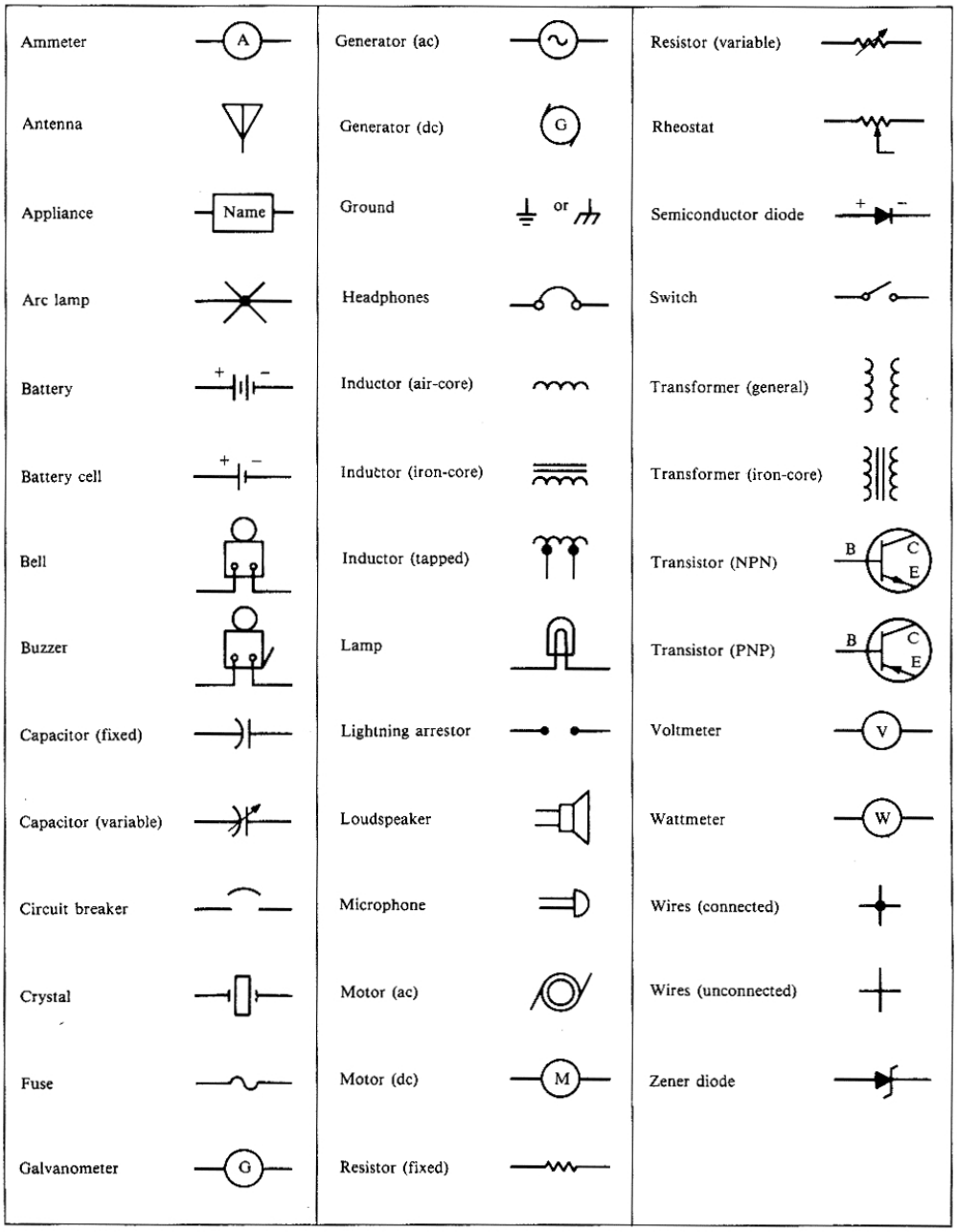
É importante que os esquemas eléctricos sejam sempre feitos de forma que sejam facilmente compreensíveis. Existem para o efeito normas internacionais que facilitam o desenho e a compreensão dos mesmos por qualquer outra pessoa. A figura seguinte mostra o circuito simples de uma lâmpada. A esquerda está o desenho e a direita o diagrama esquemático



Desenho Esquema

Normalmente procura-se fazer diagramas esquemáticos, porque facilitam a sua compreensão por outras pessoas. O mais fácil é sempre começar pela bateria ou f.e.m. e desenhar sequencialmente as diferentes peças. No fim, naturalmente o circuito tem que voltar para a f.e.m.

Os símbolos normalmente utilizados estão na figura seguinte.



Standard circuit symbols

3. Tensão (volts)

A Tensão é uma medida da capacidade da corrente para realizar trabalho (joules) por unidade de carga eléctrica (Coulomb).

A tensão é normalmente medida em volts que é o trabalho (em joules) realizado por unidade de electrões (1 Coulomb).

$$V = \frac{J}{C}$$

Para facilitar o entendimento da tensão eléctrica pode-se fazer um paralelo desta com a pressão hidráulica. Quanto maior a diferença de pressão hidráulica entre dois pontos, maior será o fluxo, caso haja comunicação entre estes dois pontos. O fluxo (que em electricidade é corrente eléctrica) será assim uma função da pressão hidráulica (tensão eléctrica) e da oposição à passagem do fluido (resistência eléctrica).

Grandeza	**Unidade**	**Símbolo**	**Equivalência**
Energia, E	Joule	J	N.m
Força, F	Newton	N	kg•m/s²
Potência,	Watt	W	J/s
Carga Eléctrica, C	Coulomb	C	6,25 x 10^{18} electrões
Tensão, V	Volt	V	J/C
Resistência, R	ohm	Ω	A/V
Frequência	Hertz	Hz	1/s
Intensidade, I	Amperes	A	C/s

Em Portugal as tomadas domésticas tem 220V, e praticamente todos os electrodomésticos vendidos trabalham a 220 V. Isto quer dizer que cada Coulomb de electrões consegue produzir 220 Joules de trabalho.

4. Resistência

Quando é estabelecida uma corrente eléctrica em um condutor metálico, um número muito elevado de electrões livres passa a se deslocar nesse condutor. Nesse movimento, os electrões colidem entre si e também contra os átomos que constituem o metal. Portanto, os electrões encontram uma certa dificuldade para se deslocar, isto é, existe uma resistência à passagem de corrente no condutor. Para medir essa resistência, foi definido uma grandeza que foi denominado Resistência Eléctrica.

Não existe nenhum condutor perfeito de electricidade. Todos os corpos apresentam alguma resistência à passagem da corrente e é preciso saber quantificar essa resistência

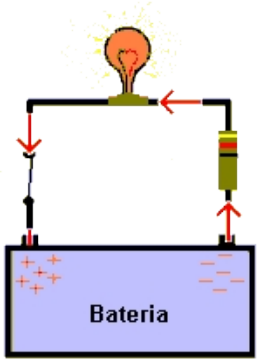
Lei de Ohm

Para uma dada resistência, a uma dada temperatura, a Intensidade de corrente, ***I***, é directamente proporcional à voltagem aplicada, ***V***:

$$V = I\ R$$

A unidade de resistência no Sistema Internacional é o Ohm (***Ω***).
Assim, quanto maior a Resistência ***R***, menor será a Intensidade de corrente, ***A***, para uma dada voltagem:

$$1\Omega = \frac{1V}{1A}$$



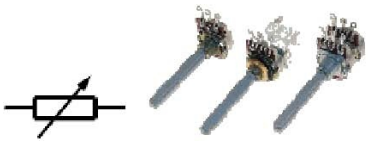
Factores que influenciam no valor de uma resistência:

1) A resistência de um condutor é tanto maior quanto maior for seu comprimento.
2) A resistência de um condutor é tanto maior quanto menor for a área de sua secção, isto é, quanto mais fino for o condutor.
3) A resistência de um condutor depende do material de que ele é feito.

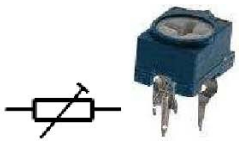
Símbolo da resistência eléctrica

Resistência carbónica

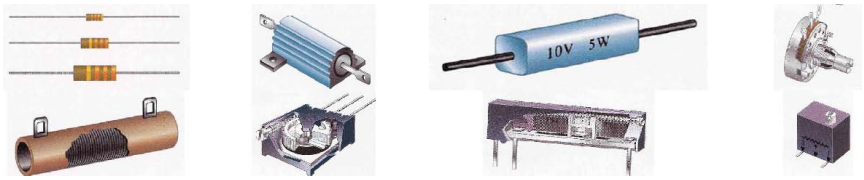
Resistência metálica



Potenciómetro



Resistência ajustável (Trimmer)



Tipos mais comuns de resistências

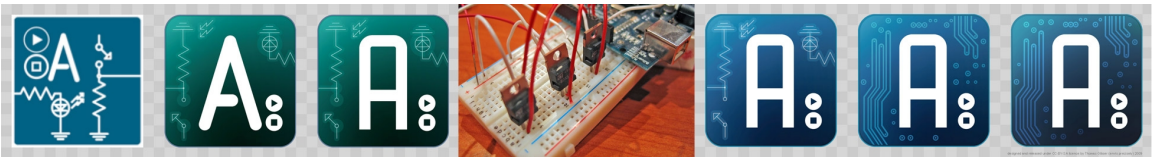
A Lei de ohm pode também ser escrita da seguinte forma:
A resistência de um condutor a uma dada Tº é directamente proporcional ao seu comprimento e inversamente à área da sua secção e depende do tipo de material:

$$R = \rho \frac{l}{A}$$

em que ρ é a resistividade em Ωm e depende do tipo de material, A é a área da secção, em m^2 e l é o comprimento em m.

As resistividades de diversos materiais se encontram na tabela seguinte:

Material	Resistividade, Ω.m
Alumínio	$2,8 \times 10^{-8}$
Cobre	$1,72 \times 10^{-8}$
Ferro	$9,5 \times 10^{-8}$
Tungsténio	$5,5 \times 10^{-8}$



Ficha 3 Melodias

Objectivos

Electrónica	Programação
Modulação da onda: É possível ter uma saída "analógica" no Arduino, ligando e desligando um pino rapidamente. Isto é chamado PMW: Power Modulation W	Variáveis indexadas: É uma variável cujos valores são dados numa tabela, e podem ser chamadas através do número de índice. É por exemplo o caso do variável IDuracao, que pode assumir diversos valores, conforme o índice. int IDuracao[ ] = {4, 8, 8, 4, 4, 4, 4, 4}; O IDuracao[1] será 4, enquanto que o IDuracao[2] será 8.
Utilização de um alti-falante	
Cálculo de Intensidade de corrente: W= V A	

As saídas do Arduino podem ser utilizadas para sintetizar notas musicais. O Arduino tem o comando tone() que gere uma onda de determinada frequência no pino designado, e tem o seguinte síntaxe:

```
tone(pino, frequencia)
tone(pino, frequencia, duração)
```

O pino será onde se vai ligar o fio positivo do altifalante; a frequência é a frequência da nota musical pretendida, e a duração, que é optativa, será o tempo, em milisegundos.

No exemplo, as frequências das notas musicais são dados no ficheiro **pitch.h** . Pode buscar os ficheiros nos exemplos que são fornecidos com o Arduino (ficheiro Tonemelody, ou escrever a versão mais simples apresentada em baixo (deve o copiar do seu directório Arduino\examples\digital\tonemelody e o gravar no directório do programa.)

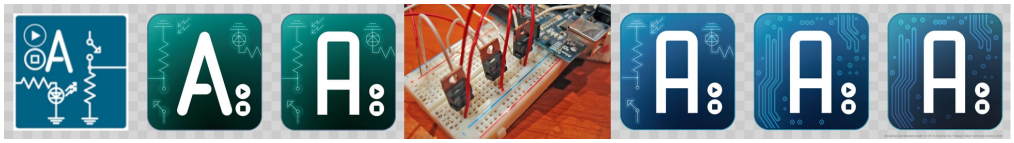
```
//Colocar o alto-falante no pino 8.
#include "pitches.h"

// as notas na música:
int melody[ ] = {
  NOTE_C4, NOTE_G3,NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3, NOTE_C4};

// duração cada nota: 4 = quartas, 8 = oitavas, etc.:
int Duracaol[ ] = {4, 8, 8, 4, 4, 4, 4, 4};

void setup() {
}

void loop() {
 // corre as notas um por um
  for (int Nota = 0; Nota < 8; Nota++) {

    // duração da nota
    int duracao = 1000/Duracaol[Nota];
    tone(8, melody[Nota],duracao);

    int pausa = duracao * 1.30;
    delay(pausa);

    // parar
    noTone(8);
  }
delay(5000);
}
```

Cálculo da intensidade de corrente do alto-falante

Cada pino do Arduino pode fornecer 40mA, ou seja 0,04A. Para se pode utilizar o alti-falante torna-se necessário verificar a sua intensidade de corrente. Por exemplo vamos assumir que tem um alti-falante de 0,2W. Sabemos que

$$W = V \times A$$

Como a corrente fornecida pelo Arduino é 5V, fica então A= 0,2/5 , o que dá 0,04A, e portanto pode ser utilizado com segurança.

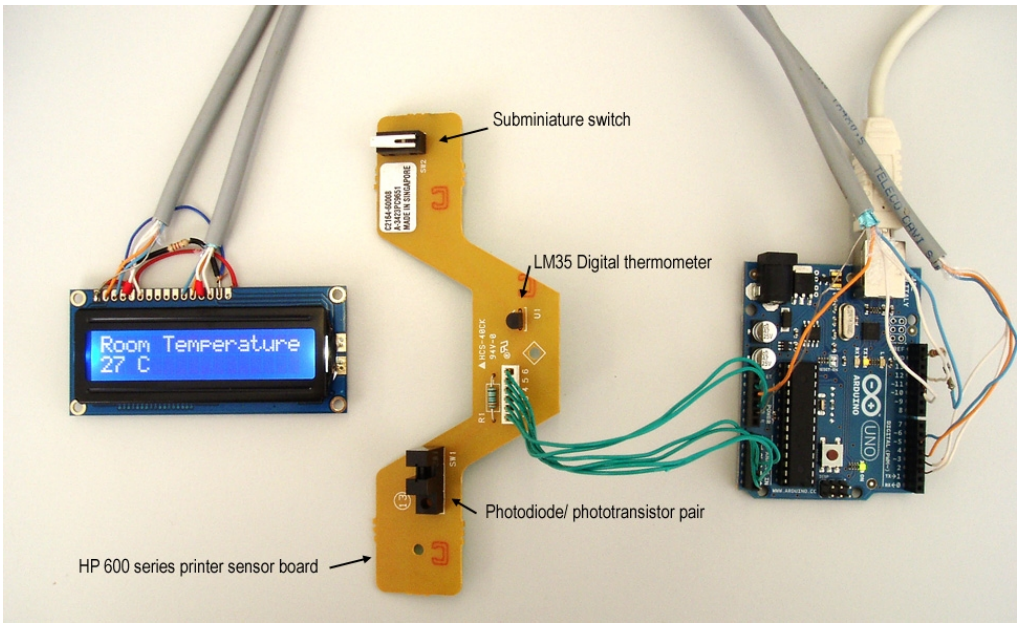


Ficha 3A. Utilização do LCD

Objectivo: Conseguir fazer a interface entre o Arduino e um LCD. Comunicar com o utilizador com mensagens informativos.

A comunicação entre o utilizador e os equipamentos é muito importante. As ecrãs LCD constituem um meio eficaz e económico de comunicar com o utilizador do sistema de automação. Na figura em baixo está um protótipo simples de um termómetro digital.

Os LCDs comunicam com os controladores através de números, tal como qualquer outro aparelho digital. Para isso existe uma tabela, chamada ASCII, que converte cada carácter em um número. Assim, consegue-se imprimir qualquer carácter, enviando o seu código para o LCD. No fim desta ficha existe uma tabela dos caracteres e dos respectivos códigos numéricos.

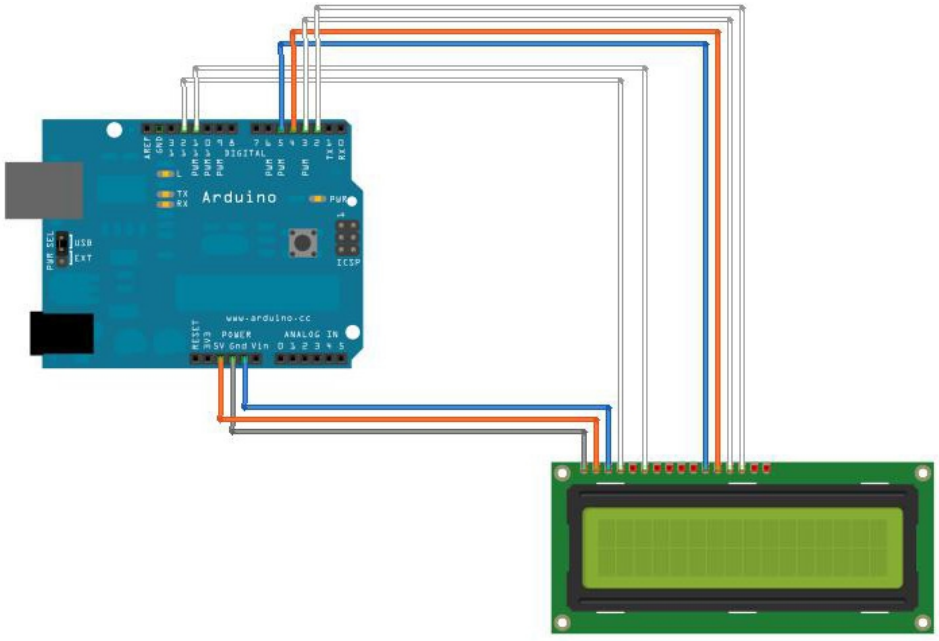


Existem diversas interfaces para o LCD. Um dos primeiros controladores de LCD e actualmente dos mais comuns é o Hitachi HD44780, que permite uma ligação em paralelo entre o Arduino e o LCD. O inconveniente desta interface é que ocupa um número razoável de pinos do Arduino, o que poderá ser um factor limitante à sua utilização.

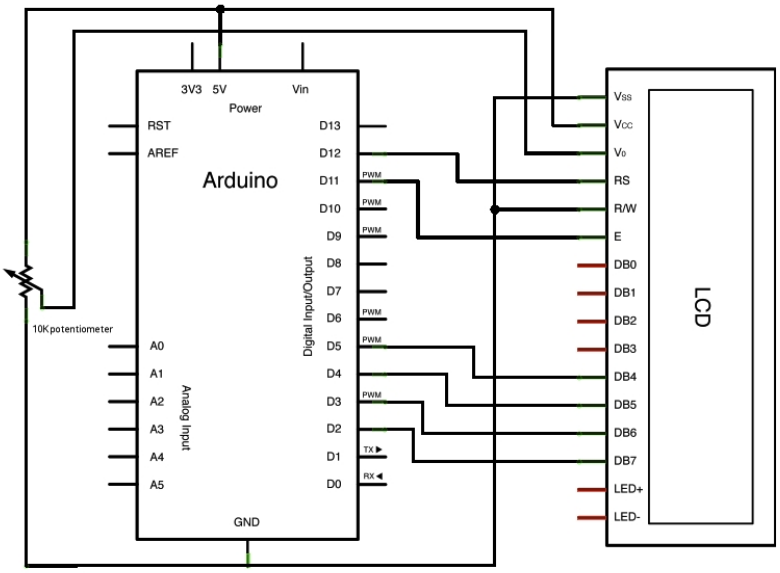
Em alternativa é possível utilizar o interface I^2C, que é um "Bus" permitindo a comunicação entre diferentes aparelhos utilizando apenas 2 pinos (ver caixa em baixo). Estes LCDs em série são bastante mais caros e difíceis de encontrar.

Projecto 1. O LCD

O LCD que temos utiliza o chip Hitachi HD44780, que é muito comum neste tipo de aplicações. Precisa de ligar o LCD de acordo com a figura em baixo. Apenas de referir que o terceiro pino do LCD serve para ajustar o contraste, o que pode ser feito com um potenciómetro (como está na figura) ou através de ligação ao pino GND (terra) do Arduino, utilizando uma resistência de cerca de 5 kΩ.



ou se preferir:



Um programa simples para pôr a funcionar o LCD está em baixo:

```
#include <LiquidCrystal.h>       // initializa a biblioteca e estabelece os pinos
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {                        // estabelece o número de colunas e linhas
  lcd.begin(16, 2);                    // Escreve no LCD.
  lcd.print("Ja funciona");
}

void loop() {                          // muda o cursor para primeira coluna da linha 1
   lcd.setCursor(0, 1);           // escreve o número de segundos desde que o Arduino foi ligado
  lcd.print(millis()/1000);
}
```

Vai reparar que logo na primeira linha, o programa tem o comando incluir:

```
#include <LiquidCrystal.h>
```

Este comando vai chamar um outro programa, neste caso o programa LiquidCrystal.h. Qual é a utilidade disso? É que quando se trata de partes de programa que se utilize muitas vezes, é mais fácil guardar esse código num ficheiro a parte, e o chamar cada vez que é necessário. Assim evita-se de ter de escrever sempre o mesmo código em todos os programas. O comando #include <LiquidCrystal.h> vai chamar o código sempre que for necessário para o programa.

Estes códigos são chamados de bibliotecas, ou Libraries em Inglês. Veja no seu computador, no directório Arduino se encontra uma pasta com as bibliotecas?

Pontos de interesse:

Os pinos do LCD são:
1. Ground
2. VCC (+3.3 to +5V)
3. Contrast adjustment (VO)- *O ideal é ter um potenciómetro para ajustar o contraste, no entanto também se pode utilizar um simples resistência*

4. Register Select (RS). RS=0: Command, RS=1: Data
5. Read/Write (R/W). R/W=0: Write, R/W=1: Read
6. Clock (Enable). Falling edge triggered

Como os caracteres mais habituais cabem em 4 bits, e para poupar pinos, muitas vezes se pode omitir os quatro bits seguintes:

7. Bit 0 *(Não utilizado no modo 4-bits)*
8. Bit 1 *(Não utilizado no modo 4-bits)*
9. Bit 2 *(Não utilizado no modo 4-bits)*
10. Bit 3 *(Não utilizado no modo 4-bits)*

11. Bit 4
12. Bit 5
13. Bit 6
14. Bit 7

15. Backlight Anode (+) *Estes dois pinos alimentam a retro-iluminação. Aqui também se pode utilizar um potenciómetro para ajustar a retro-iluminação.*
16. Backlight Cathode (-)

Projecto 2

Agora vamos alterar o projecto para escrever no LCD a temperatura do Ar. Para isso vamos juntar o código que utilizamos no exercício anterior. Tente fazer por si. Só se tiver dúvidas consulte a figura em baixo.

Projecto 3

Inclua alguns alarmes por forma a avisar quando a temperatura está muito quente ou fria. Por exemplo acender uma luz vermelha quando a T ultrapassa os 28 ºC.

Projecto 4

O seguinte código escreve no LCD o que se introduzir através do teclado do computador:

```
// declarar os pinos do LCD

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup(){    // define o número de caracteres e linhas
 lcd.begin(16, 2);   // inicializa a comunicação:
 Serial.begin(9600);
}

void loop()
{
  // quando é detectado um caracter na porta série
  if (Serial.available()) {     // esperar até receber toda a mensagem
    delay(100);       // limpar a écrã
    lcd.clear();      // ler todos os caractéres
    while (Serial.available() > 0) {       // escrever todos os caracteres no écrã
      lcd.write(Serial.read());
    }
  }
}
```

VER também:

http://quarkstream.wordpress.com/2010/03/24/arduino-11-a-character-lcd/
http://www.nuelectronics.com/estore/index.php?main_page=product_info&cPath=1&products_id=12

Aqui está a tabela dos caracteres e dos seus valores correspondentes

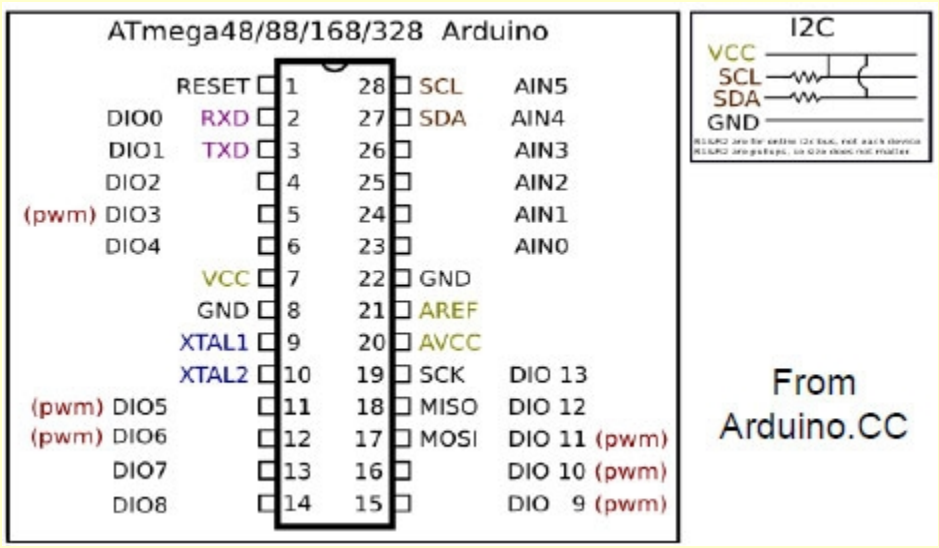
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Pontos de interesse:

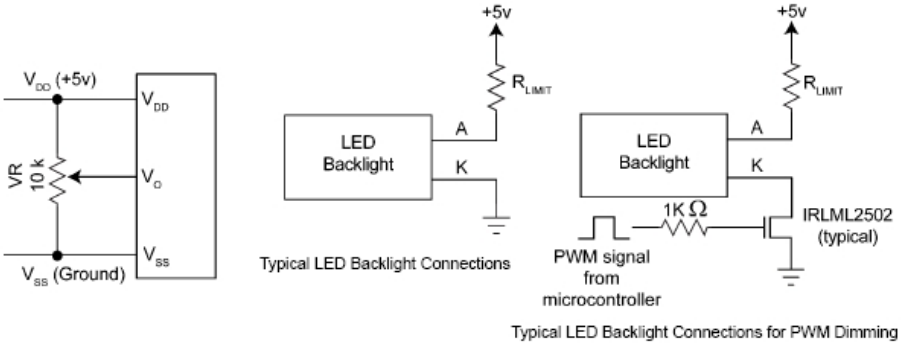
I²C (Inter-Intergrated Circuit) é um barramento multi-mestre desenvolvido pela Philips que é usado para conectar periféricos de baixa velocidade (escravos) a uma placa mestre.

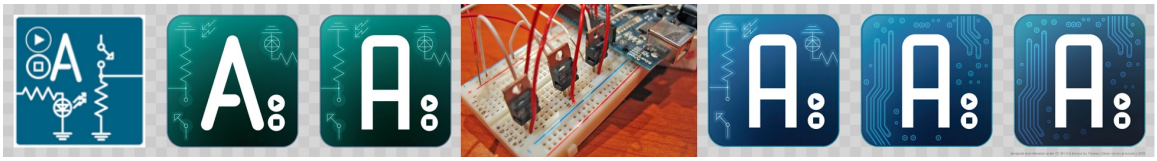
O nome significa Circuito Inter-integrado e é pronunciado I-ao quadrado-C e também, incorretamente, I-dois-C. A utilização e implementação deste sistema é livre, contudo, algumas taxas ainda são exigidas para obtenção de endereços escravos I²C. SMBus é um subconjunto do I²C que define convenções eléctricas rígidas e de protocolo.

O I²C utiliza apenas duas linhas bidirecionais de dreno aberto, Dados Seriais (Serial Data - SDA) e Clock Serial (Serial Clock - SCL), computadores. Este protocolo especifica dois sinais de comunicação, um com o sinal de clock (gerado pelo mestre), e outro de dados, bidirecional. No Arduino, o I²C é implementado nos pinos Analog 4 (SDA) e Analog 5 (SCK)



Alguns circuitos adicionais para o LCD: $R_{LIMIT} = (V_{BACKLIGHT} - 4.2V) / 0.12A$





Ficha 4. Entrada digital

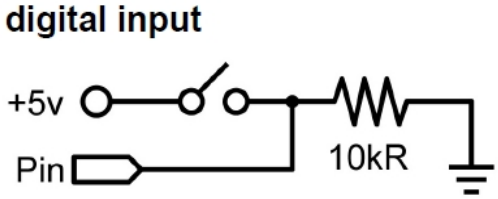
Objectivos

Electrónica	Programação

Utilização dos pinos do Arduino para entrada de sinal.	Utilização do condicional -if- no Arduino:
	
	`if (x > 120) digitalWrite(2, HIGH);`
	
	`if (x > 120)`
	`digitalWrite(2, HIGH);`
	
	`if (x > 120){ digitalWrite(2, HIGH); }`
	
	`if (x > 120){`
	`  digitalWrite(2, HIGH);`
	`  digitalWrite(3, HIGH);`
	`}  // estao todos certos`

Vimos que o arduino tem 14 pinos que podem ser configurados tanto como entrada como saída. São os pinos digitais 1-14. Para utilizar estes pinos, deve primeiro os declarar com o comando pinMode (), e podem ser tanto INPUT como OUTPUT.

Vamos agora ligar um destes pinos a um interruptor. Para o efeito deve utilizar a terra GND e os 5V do próprio Arduino. No entanto deve inserir uma resistência, por exemplo de 10kΩ no circuito para evitar problemas:



Vamos agora fazer um pequeno programa que pisca cada vez que se carrega no interruptor:

Projecto 2

Faça um dispositivo que conta o número de vezes que se carrega no botão e os apresenta no Ecrã.

Projecto 3

Que tal uma campainha musical? Quando se carrega no botão, toca uma melodia. E pisca uma luz?

Pontos interessantes

Neste trabalho vimos a utilização do comando IF ou seja uma condição que caso se verifica, o Arduino executará os comandos seguintes.

No Arduino, este comando pode tomar as seguintes formas, que são todas válidas:

if (x > 120) digitalWrite(LEDpin, HIGH);

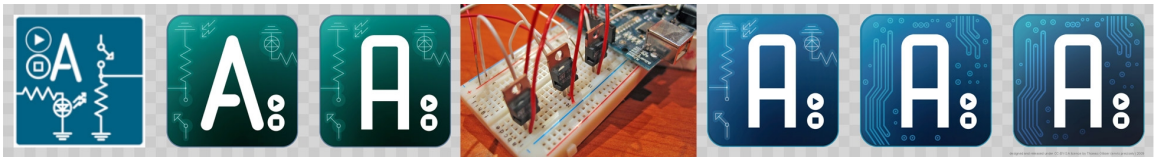
if (x > 120)

```
digitalWrite(LEDpin, HIGH);

if (x > 120){ digitalWrite(LEDpin, HIGH); }

if (x > 120){
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}                              // estao todos certos
```

As afirmações que estão a ser avaliados dentro dos parentes precisam de um ou mais dos seguintes operadores:

Operadores de comparação:

```
 x == y (x is equal to y)
 x != y (x is not equal to y)
 x <  y (x is less than y)
 x >  y (x is greater than y)
 x <= y (x is less than or equal to y)
 x >= y (x is greater than or equal to y)
```



Ficha 5. Sensor de humidade ambiente

A humidade relativa do ar é um dos parâmetros mais importantes para a saúde e conforto dos animais em cativeiro. Se a humidade estiver dentro dos limites tabulados, os animais terão um crescimento saudável.
Existem hoje circuitos integrados muito avançados que permitem medir a humidade ambiente com bastante precisão. A gama HIH 4000 da Honeywell inclui sensores muito evoluídos com uma precisão bastante elevada ao ponto de podem ser utilizados em instrumentos científicos.

Apenas deve-se ter cuidado para não tocar na abertura do sensor, o que pode danifica-lo. Em baixo encontra a descrição do sensor e alguns elementos importantes para a sua utilização.

HIH-4000 Series

Humidity Sensors

The HIH-4000 Series Humidity Sensors are designed specifically for high volume OEM (Original Equipment Manufacturer) users. Direct input to a controller or other device is made possible by this sensor's linear voltage output. With a typical current draw of only 200 µA, the HIH-4000 Series is often ideally suited for low drain, battery operated systems. Tight sensor interchangeability reduces or eliminates OEM production calibration costs. Individual sensor calibration data is available.

The HIH-4000 Series delivers instrumentation-quality RH (Relative Humidity) sensing performance in a competitively priced, solderable SIP (Single In-line Package). Available in two lead spacing configurations, the RH sensor is a laser trimmed, thermoset polymer capacitive sensing element with on-chip integrated signal conditioning. The sensing element's multilayer construction provides excellent resistance to most application hazards such as wetting, dust, dirt, oils and common environmental chemicals.

FIGURE 3. MOUNTING DIMENSIONS
for reference only mm/[in]

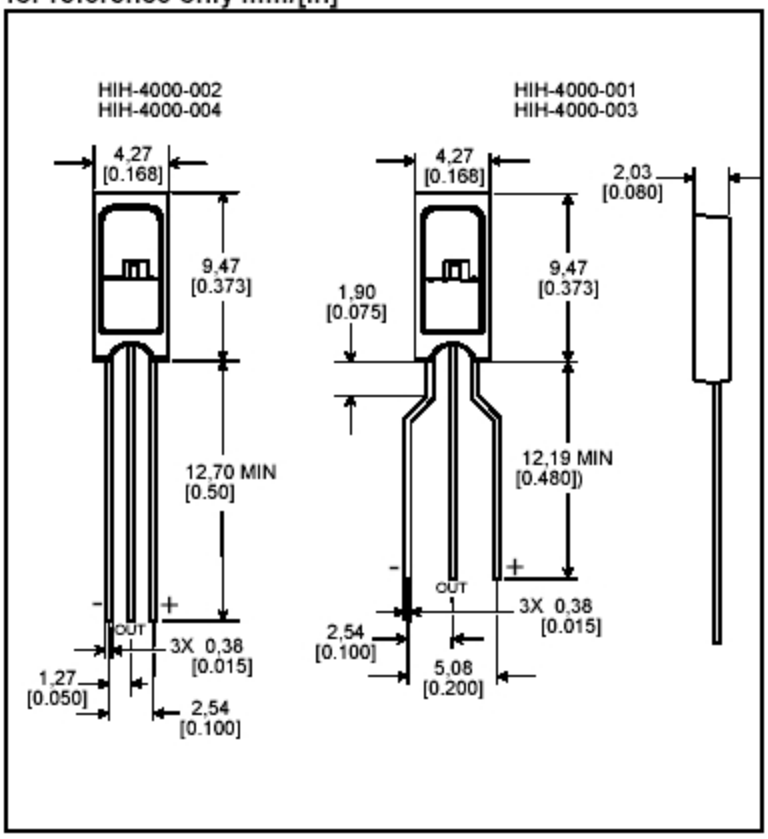


FIGURE 4. TYPICAL BEST FIT STRAIGHT LINE

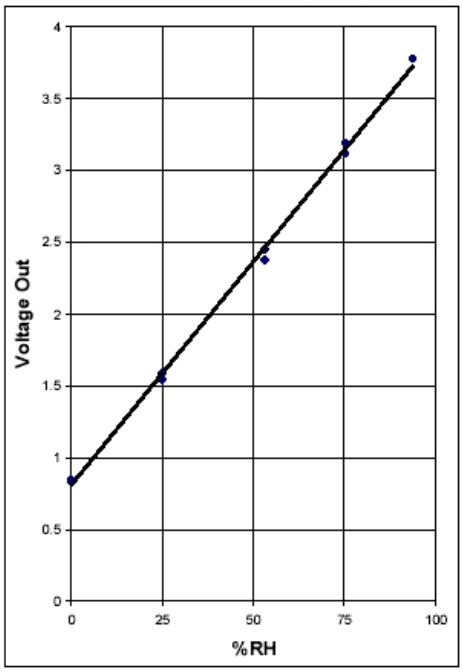
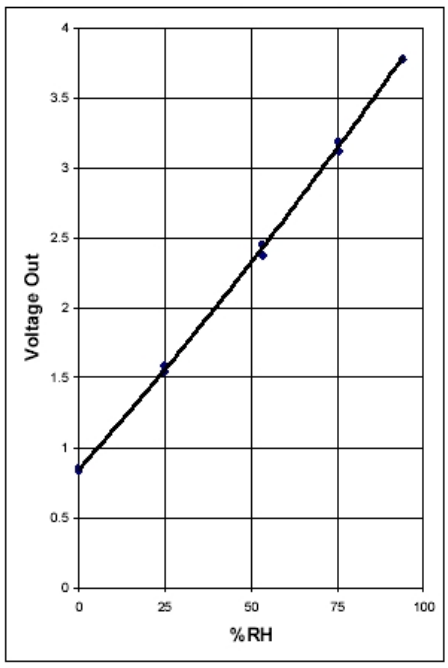


FIGURE 5. TYPICAL 2nd ORDER CURVE FIT



Utilizar o programa que já vimos para ler uma entrada analógica:

```
int leitura = 0;    // variavel que vai ler o pino 0

void setup() {
  // abrir a porta série em 9600 bits/segundo
  Serial.begin(9600);
}

void loop() {
  // fazer uma leitura no pino A0
  leitura = analogRead(0);

   Serial.println(leitura, DEC);  // escrever o resultado


  delay(1000);
}
```

e incluir as linhas necessárias para passar os valores para voltagem e depois para Humidade relativa. O fabricante refere a seguinte equação de calibração genérica para uma corrente de alimentação de 5V:

```
Sensor RH= (Vout-0.805)/0.031
```

(Atenção: o sistema lê de 0 V a 5 V em 1024 passos. Assim, cada passo corresponde a 5/1024, ou seja 0.0049 V.

Assim, o seu programa será algo como:

```
int leitura = 0;    // variavel que vai ler o pino 0

void setup() {
  // abrir a porta série em 9600 bits/segundo
  Serial.begin(9600);
}

void loop() {
  // fazer uma leitura no pino A0
  leitura = analogRead(0);
  float voltagem= 0;
  voltagem = leitura *0.00488;
  float HR= voltagem-0.805;
  HR= voltagem/0.031;
Serial.println("Humidade relativa do ar");
   Serial.println(HR, DEC);  // escrever o resultado


  delay(1000);
}
```

Se também medir a T, então pode compensar em função da T, em ºC:

Temperature compensation : True RH= Sensor RH/(1.0305+0.000044T-0.0000011T^2), T in ºC

Ou

Temperature compensation Vout=(0.0305+0.000044T-0.0000011T^2)(Sensor RH)+(0.9237-0.0041T+0.000040T^2) , T=Temperature in ºC

Pontos de interesse:

Uma função muito importante da programação é a possibilidade de utilizar variáveis. Ou seja atribuir um nome a um número e depois poder fazer contas com esse número e até alterar o seu valor. Efectivamente o programa memoriza o nome do variável e vai actualizando o seu valor. Por exemplo no exemplo anterior foi definido o variável leitura:

```
int leitura = 0
```

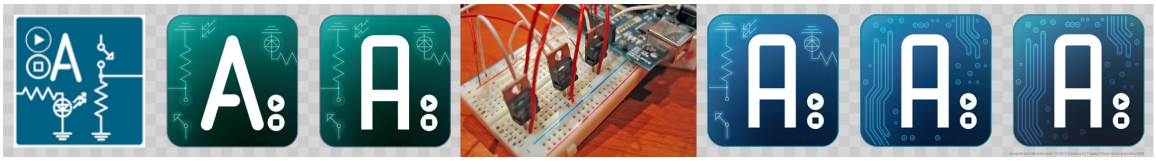
Mais tarde dissemos ao programa que o valor da leitura seria igual ao valor lido no pino analógico A0:

```
leitura = analogRead(0)
```

e depois voltamos a fazer contas com esse variável e até passamos o seu valor para outro variável:

```
voltagem = leitura *0.00488
```

Existem vários tipos de variáveis. Os mais importantes são o “int” que é um número inteiro e o “float” que é um número com casa decimal e com maior precisão.

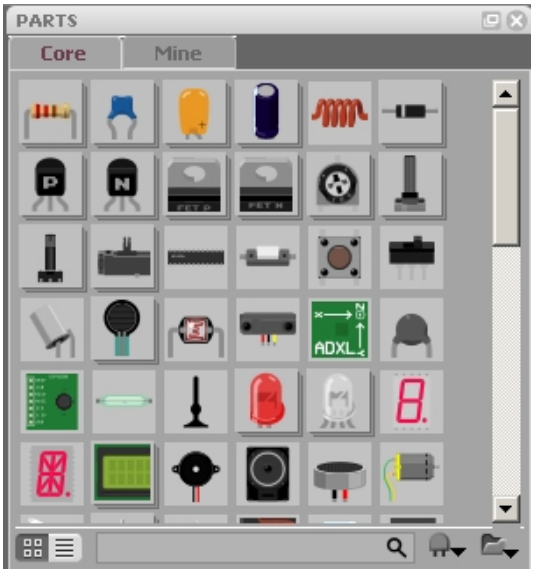


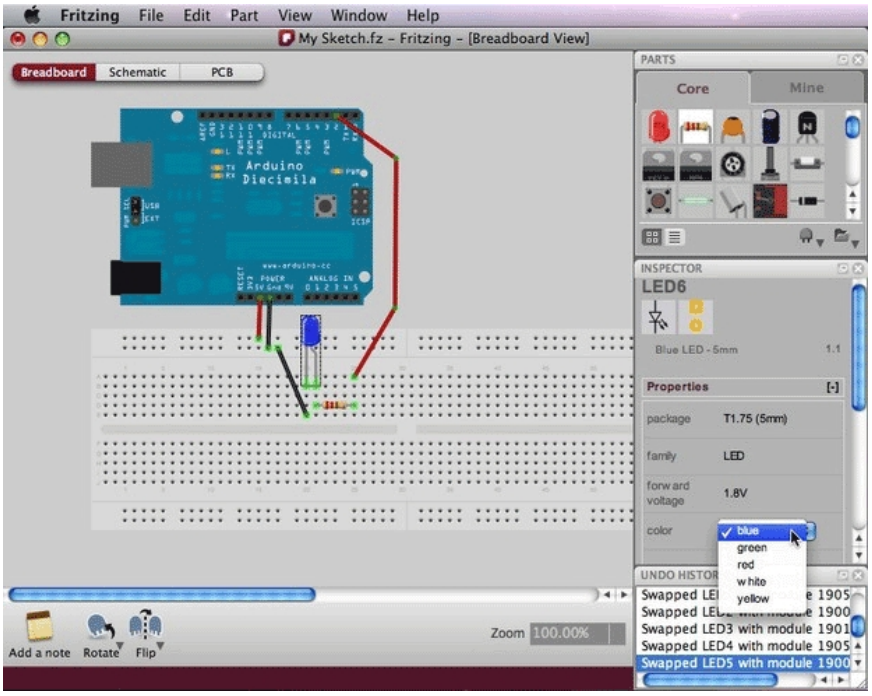
Ficha 5A. Fritzing

Em qualquer trabalho de engenharia é muito importante poder realizar um esquema do trabalho, para que o trabalho possa ser replicado no futuro. No caso de Arduino, existe felizmente um programa excelente que permite realizar os esquemas eléctricos de forma muito fácil. É o Fritzing:

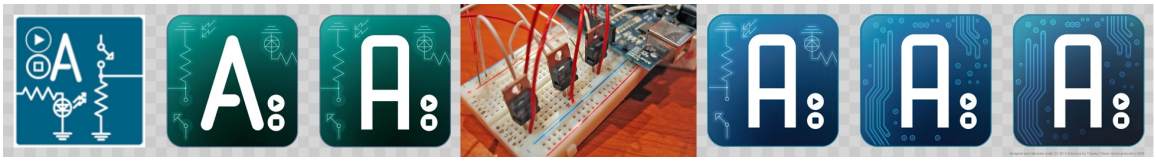
http://fritzing.org/

O programa é muito simples: Do lado direito tem uma caixa com muitas peças. É só escolher a peça e a arrastar para a placa de ensaios. Também existem diversos Arduinos para os seus projectos.





A partir de agora, e à medida que vai criar projectos mais ambiciosos, será boa ideia os registar no Frtizing e guardar para o futuro.

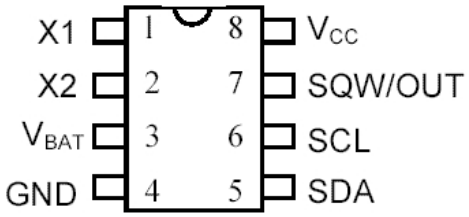


Ficha 6. Relógio em Tempo Real, RTC, e protocolo I²C

Quando começamos a desenvolver aplicações mais a sério, deparamos com a necessidade de ter acesso a um relógio em tempo real. Isto é um relógio que forneça as horas, minutos, segundos, dia, mês e ano, e que não se engane mesmo quando há uma falha de energia.

Para isso, uma das melhores opções é utilizar o chip DS1307+ da Dallas. É um chip que não só serve como calendário e relógio, mas como veremos mais a frente serve par ter acesso à memória não volátil.

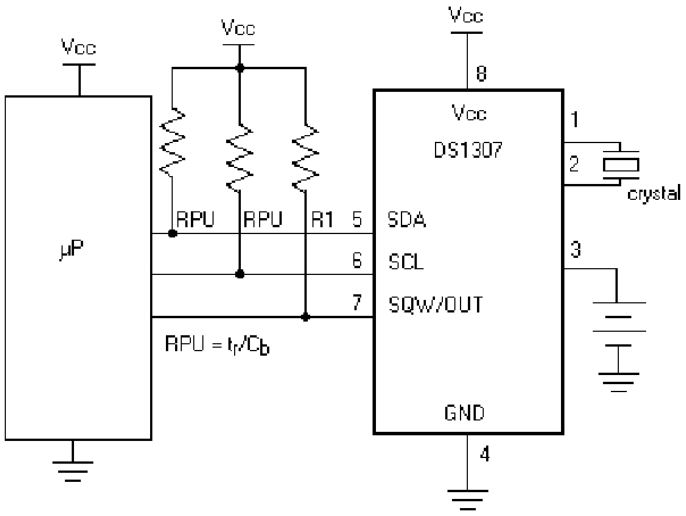
O chip é um DIP de 8 pinos, com a seguinte configuração:

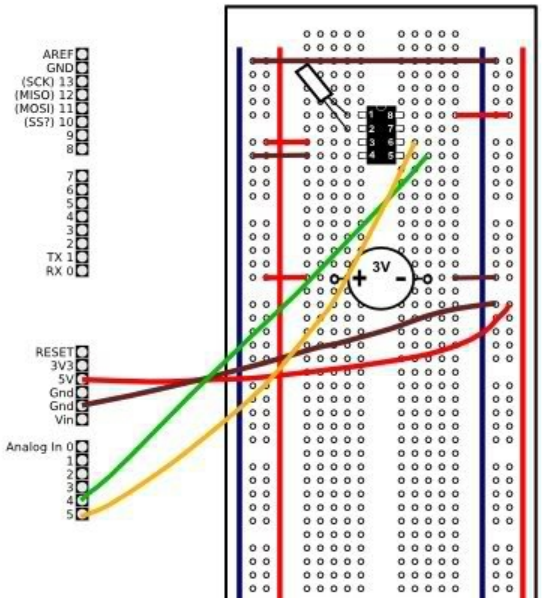


DS1307 8-Pin DIP (300-mil)

Para fazer este projecto necessita, para além do DS1307, de um oscilador (cristal) de 32.768 kHz, e de uma bateria de apoio, o CR2032. O DS1307 alimenta-se da corrente 5V, e só na ausência desse, passa a utilizar a bateria, para não perder os dados. Se tiver um computador velho, pode lhe tirar o oscilador, a bateria e o suporte da bateria. A única peça adicional que vai precisar é do DS1307.

A ligação é (as resistências do SDA e SCL até ao VCC não são necessários, pois são realizados internamente no processador):



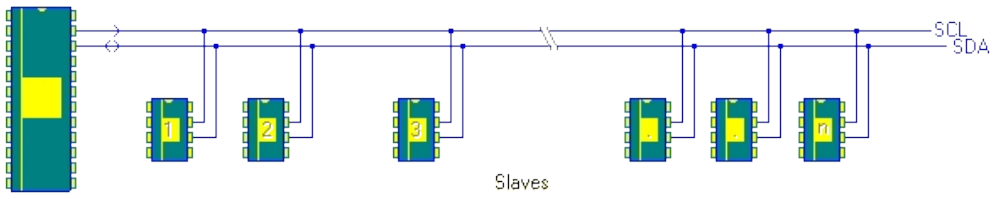


O programa é:

```
#include <Wire.h>

#include "Time.h"
#include "DS1307RTC.h"

void setup()
{
    Serial.begin(9600);
```

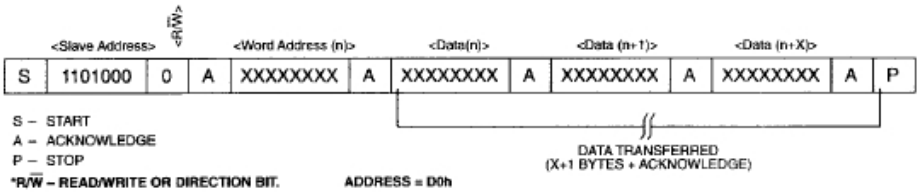
```
  // Pode retirar as proximas duas linhas se as horas ja estao certas
   setTime(17,05,0,1,6,11); // Establece a hora como 17:05:00  1 Jun 2011 (ver em baixo)
  RTC.set(now());  // acerta o RTC com a hora establecido na linha anterior

  // formato do tempo - setTime(hr,min,sec,day,month,yr);

  setSyncProvider(RTC.get);   // comando para tirar a hora do RTC

}

void loop()
{
  Serial.print( "Neste momento sao ");

  Serial.print( year() );
  Serial.write('/');

  Serial.print( month() );
  Serial.write('/');

  Serial.print( day() );
  Serial.write(' ');

  Serial.print( hour() );
  Serial.write(':');

  Serial.print( minute() );
  Serial.write(':');

  Serial.println( second() );
delay(1000);

}
```

Um dos aspectos mais interessantes do DS1307 é funcionar sobre o Bus I²C. Esse bus permite a um grande número de aparelhos funcionarem em conjunto. Basicamente, haverá um mestre, e os restantes serão escravos. A comunicação é feita por dois fios (SCL e SDA) para além do natural 5V e terra.



No caso do Arduino existe uma óptima biblioteca para comunicar através do protocolo I2C, que é a biblioteca Wire.

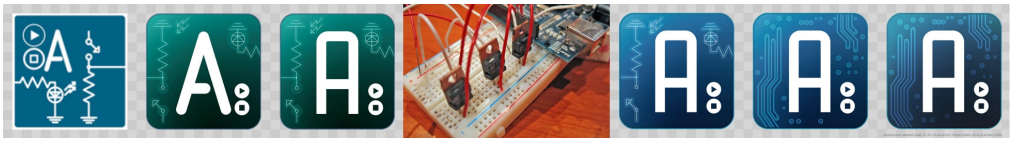
Para comunicar no I^2C, cada aparelho escravo precisa de ter um número que o identifique. No caso do DS1307, o seu identificador é: 1101000,ou seja 0x68. Este número deve ser seguido do endereço dos dados e depois dos dados propriamente ditos.



Outro aspecto bastante do DS1307 é o facto de possuir alguma memória não volátil livre, que podem ser utilizados para guardar variáveis. Esse espaço é particularmente interessante para guarda

Projecto 1

Crie um depertador que dispara a hora que marcarmos.



Ficha 7. Automação e Comando

Introdução

Uma das funções principais do Arduino é ligar e desligar (comutar) aparelhos. Para isso o Arduino pode enviar +5V para os pinos 0-13. No entanto os pinos do Arduino estão directamente ligados ao microprocessador, pelo que só podem fornecer correntes muito pequenas, com um máximo recomendado de 40mA, o que é muito pouco para as aplicações reais. A corrente fornecida por estes pinos, serve efectivamente para ligar LEDs ou outro equipamento muito pequeno.

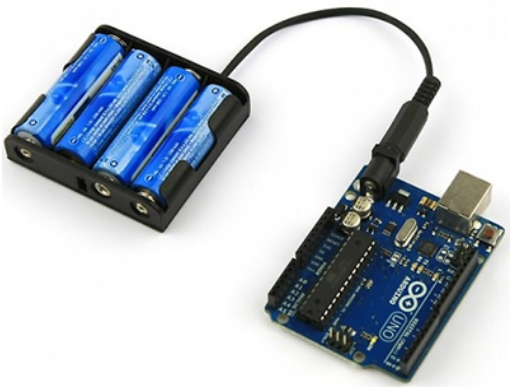
De acordo com a carga que é necessário comutar, seja ela uma pequena ventoinha, uma cortina eléctrica ou uma bomba, existem diversos equipamentos que permitem "ampliar" o sinal fornecido pelo Arduino. Nesta ficha vai-se trabalhar diversos equipamentos normalmente utilizados para esse fim, começando pelos de menor potência.

Projecto 0 Arduino a funcionar com baterias

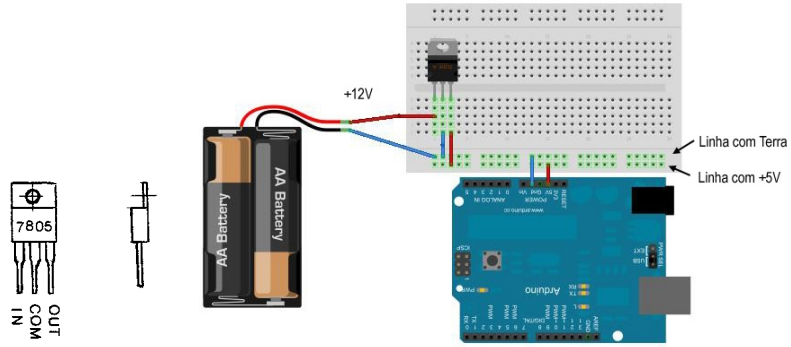
A porta USB do seu computador consegue dar 500 mA de corrente. Assim, se vai utilizar algo que gasta mais do que isso, deve procurar alimentar o seu Arduino de outra forma.

Também num projecto a "sério" não vai querer ter o Arduino sempre ligado ao seu computador. Assim, muitas vezes vai precisar de alimentar o sistema a partir de um outro fonte de alimentação. Para isso tem várias opções:

- O mais simples é utilizar uma ficha "banana" para ligar o Arduino à bateria (6-12Volts). A ficha deve ser fêmea de 2.1mm, em que o positivo está ligado ao centro. Assim, a corrente será regulada por um regulador de voltagem que existe ao lado da ficha e transformado para 5V antes de ser utilizado pelo Arduino



Uma segunda opção é utilizar um chip LM7805 que produz uma corrente estabilizada de 5V[1] e pode disponibilizar até 1A. Esse chip funciona quase como um transformador e é extremamente útil e fácil de utilizar. Recomenda-se que prepare o seguinte circuito para alimentar o Arduino:

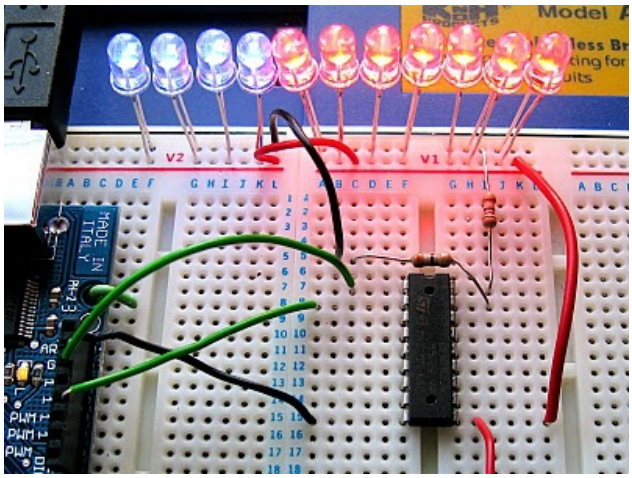


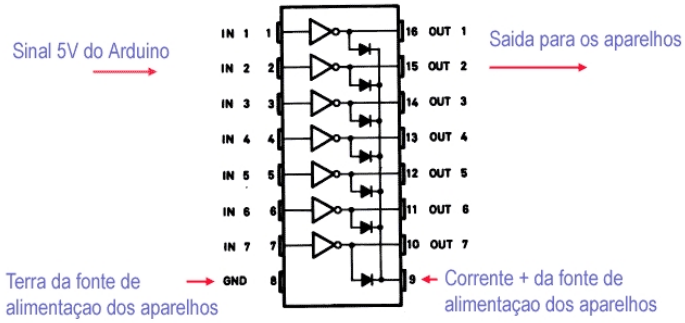
Ainda antes de montar o Arduino, ligue-o ao PC e faça um pequeno programa que ligue o Pino 2 durante 5 segundos e depois o desligue, e espere outros 5 segundos. Quando o Arduino estiver programado pode o desligar do computador, e o instalar no circuito alimentado a baterias. Para ter a certeza que está a funcionar bem, coloque uma LED no pino 2 e verifique se liga e desliga correctamente.

[1] Existem outros chips para 6V, 9V e 12V, que se chamam, respectivamente 7806, 7809 e 7812.

A. Transístores e pares Darlington[2]

Para cargas até 500mA, podemos utilizar os transístores normais, do tipo Darlington. Este chip é na verdade um conjunto de sete pares de transístores que permitem ao Arduino comandar sete aparelhos distintos com cargas até 500mA. Pode-se utilizar por exemplo o UNL2003A, que é comandado pelo sinal de 5V do Arduino. Este chip é muito simpático porque na verdade são sete chips, e portanto pode-se comutar sete aparelhos diferentes com um único chip.



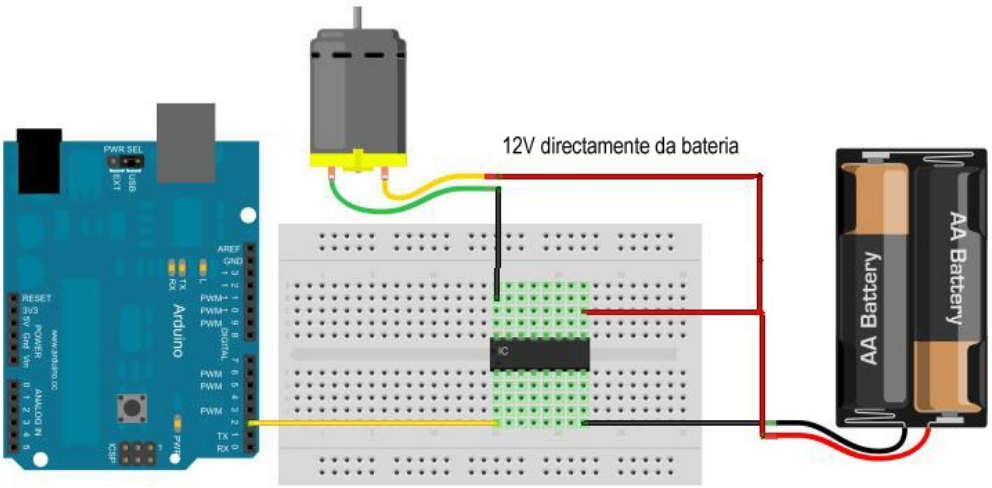


Projecto 1: Utilização de transistores

As ventoinhas pequenas existentes nos computadores funcionam a 12 V e consomem 100 a 150 mA, pelo que são ideias para o primeiro projecto. Neste projecto vai-se

[2] Darlington, em homenagem ao Sr. Darlington que inventou este chip. Basicamente consiste num transístor que depois liga a um segundo transístor. Assim, consegue-se comutar potencias muito maiores.

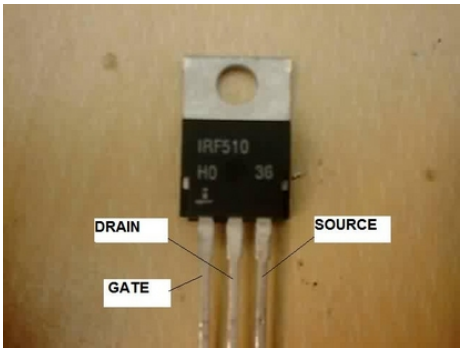
fazer uma ventoinha que ligar durante 5 segundos, e depois fica desligado durante outros 5 segundos.



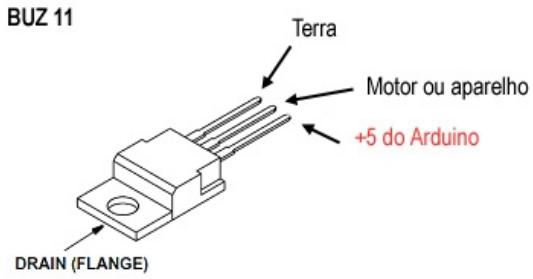
B. Cargas maiores com transístores tipo MOSFET

Pode-se utilizar transístores de grande voltagem para ligar cargas maiores. Para o efeito podem-se utilizar transístores metálicos, MOSFET, com maior capacidade (por exemplo o BUZ11 ou o P40NE) para ligar cargas até 50V e 30A[3].

[3] Há milhares de modelos diferentes. Pode obter muitos MOSFETs gratuitamente abrindo uma fonte de alimentação de um velho computador. Cada tipo terá as suas características, como por exemplo o IRF510 com capacidade para 100V e 5.6A.



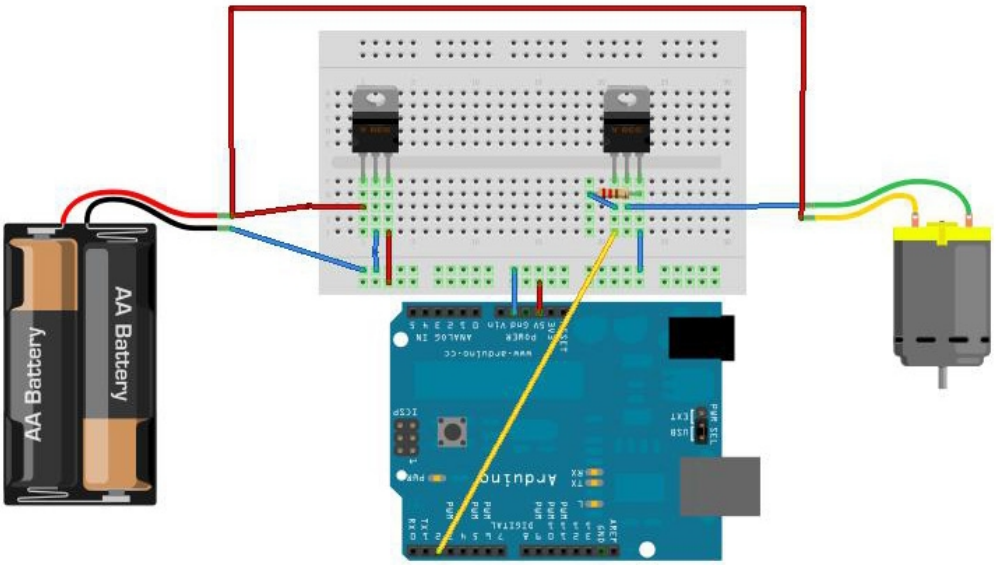
O chip tem três pinos: O Gate que recebe o sinal do Arduino (5V), e portanto servirá para comandar a comutação. O Source vai ligar à terra da fonte de alimentação, enquanto que o Drain vai ao pólo positivo do aparelho.



Projecto 2. Ligar e desligar uma lâmpada de 1 A.

Neste projecto vai-se ver como é possível comutar uma lâmpada relativamente forte utilizando o Arduino. Para o efeito vai utilizar o chip BUZ11 para "ampliar" o sinal do Arduino e comutar a lâmpada.

O circuito é bastante simples. A bateria recebe os 12V directamente da bateria e o chip BUZ11 serve para comutar a terra.



É de referir que é necessário ainda ligar uma resistência de 10KΩ entre o sinal que vem do Arduino e a terra do MOSFET. Essa resistência é importante para descarregar o sinal, e permitir que o aparelho seja desligado.

Deve ter muita atenção ao aquecimento do transístor. Recomenda-se vigiar muito de perto a temperatura do transístor depois de o ligar. Se sentir que está a aquecer muito, deve reduzir a carga ou utilizar um dissipador de calor.

Projecto 3. Variador de intensidade da luz

Com certeza já terá utilizado candeeiros em que se pode alterar a intensidade da luz. Roda-se um parafuso e a intensidade aumenta ou diminui. Com o Arduino vamos facilmente construir um variador de intensidade de luz. Mas primeiro vai desligar o seu Arduino da bateria e o ligar ao computador.

O programa para variar suavemente a intensidade da luz recorre à uma função do Arduino que permite variar a voltagem à saída dos pinos. Se reparar no seu Arduino, o Pino 3 e alguns outros pinos tem escrito ao lado PWM. Isto quer dizer que estes pinos podem alterar ou modular a voltagem. O programa que vai escrever será algo assim:

```
//Programa para alterar a intensidade da luz no pino 3

int intensidade = 0;
int passo = 5;

void setup()  {
  pinMode(3, OUTPUT);
}
```

```
void loop()  {
  analogWrite(3, intensidade);
  intensidade = intensidade + passo;
  if (intensidade == 0 || intensidade == 255) {
    passo = -passo ;
  }
  delay(200);
}
```

Projecto 4. Ventilação automática

Voltando ao projecto 1, faça as alterações necessárias para a ventoinha ligar quando a T atinge um determinado valor e depois desligar quando a temperatura baixa.

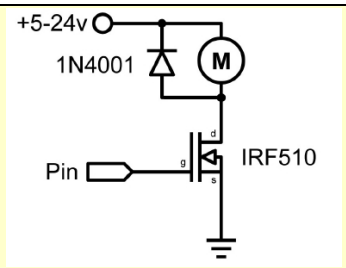
Pontos de interesse:

Os transístores são pequenos interruptores, que permitem comutar cargas maiores do que os 5V e 40mA permitido nos pinos do Arduino.

Os transístores também podem servir como amplificadores, variando a intensidade de corrente de saída, de acordo com a corrente que recebem. Efectivamente ao variar a corrente no pino gate, é possível variar a intensidade de corrente que passa no transístor. Esta possibilidade é utilizada frequentemente para alterar a intensidade de luz.

Pontos de interesse:

Para o bom funcionamento dos motores, é muitas vezes recomendada incluir um diodo entre os seus pólos. O diodo evita a passagem de corrente no sentido inverso, que poderia fazer estragos ao Arduino. Essas correntes indesejadas são chamadas correntes indutivas.



BUZ11

30A, 50V, 0.040 Ohm, N-Channel Power MOSFET

This is an N-Channel enhancement mode silicon gate power field effect transistor designed for applications such as switching regulators, switching converters, motor drivers, relay drivers and drivers for high power bipolar switching transistors requiring high speed and low gate drive power. This type can be operated directly from integrated circuits.

Formerly developmental type TA9771.

Ordering Information

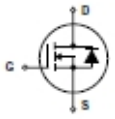
PART NUMBER	PACKAGE	BRAND
BUZ11	TO-220AB	BUZ11

NOTE: When ordering, use the entire part number.

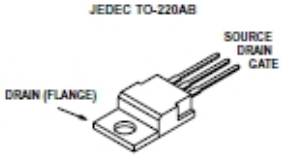
Features

- 30A, 50V
- $r_{DS(ON)} = 0.040\Omega$
- SOA is Power Dissipation Limited
- Nanosecond Switching Speeds
- Linear Transfer Characteristics
- High Input Impedance
- Majority Carrier Device
- Related Literature
  - TB334 "Guidelines for Soldering Surface Mount Components to PC Boards"

Symbol



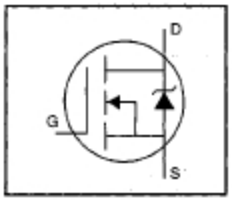
Packaging



IRF510

HEXFET® Power MOSFET

- Dynamic dv/dt Rating
- Repetitive Avalanche Rated
- 175°C Operating Temperature
- Fast Switching
- Ease of Paralleling
- Simple Drive Requirements



V_{DSS} = 100V

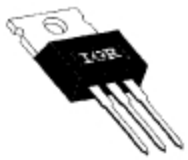
$R_{DS(on)}$ = 0.54Ω

I_D = 5.6A

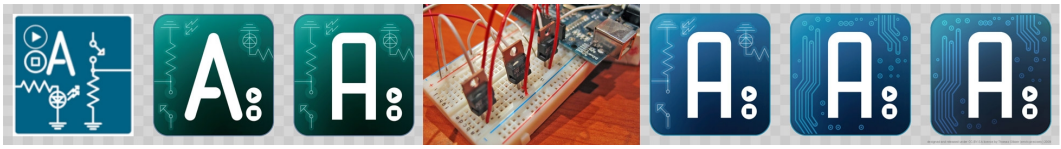
Description

Third Generation HEXFETs from International Rectifier provide the designer with the best combination of fast switching, ruggedized device design, low on-resistance and cost-effectiveness.

The TO-220 package is universally preferred for all commercial-industrial applications at power dissipation levels to approximately 50 watts. The low thermal resistance and low package cost of the TO-220 contribute to its wide acceptance throughout the industry.



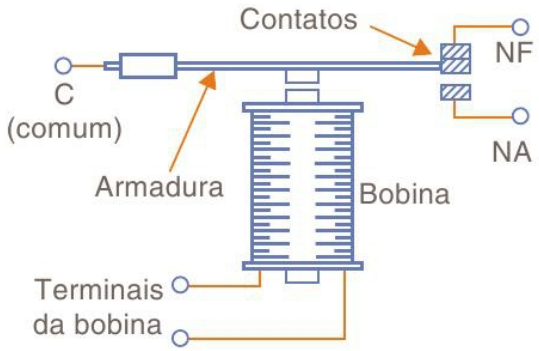
TO-220AB



Ficha 7A. Automação com relés

Introdução

O relé é um dispositivo electromecânico que serve principalmente para ligar ou desligar aparelhos eléctricos de 220V ou de maior tensão. O relé é constituído por dois circuitos que não se tocam: Existe o circuito de comando e o circuito de comutação. Quando uma corrente passa pela bobine do circuito do comando, é gerado um campo electromagnético que acciona o circuito de comutação e activa o relé. Sendo assim, uma das aplicabilidades do relé é a utilização de baixas correntes para o comando, protegendo o operador das possíveis altas correntes que irão circular no segundo circuito (contactos).



Os contactos NA (normalmente aberto) são os que estão abertos enquanto a bobina não está energizada e que fecham, quando a bobina recebe corrente. Os NF (normalmente fechado) abrem-se quando a bobina recebe corrente, ao contrário dos NA. O contacto central ou C é o comum, ou seja, quando o contacto NA fecha é com o C que se estabelece a condução e o contrário com o NF.

Os relés são comuns em electrodomésticos, geralmente quando existe um controle electrónico que liga algo como um motor ou uma lâmpada. Eles também são muito comuns em carros, onde a fonte de energia de 12V significa que quase tudo no carro precisa de uma grande quantidade de corrente. Nos modelos mais novos, os fabricantes combinam os painéis de relés na caixa de fusíveis para facilitar a manutenção. As seis caixas cinzas nesta foto da caixa de fusíveis do Ford Fiesta são relés

Os relés podem-se instalar em bases, o que facilita a sua utilização.

Projecto 1

Faça um programa que liga uma lâmpada de 220V quando a intensidade luminosa é baixa. Utilize o relé para activar e desactivar a lâmpada.

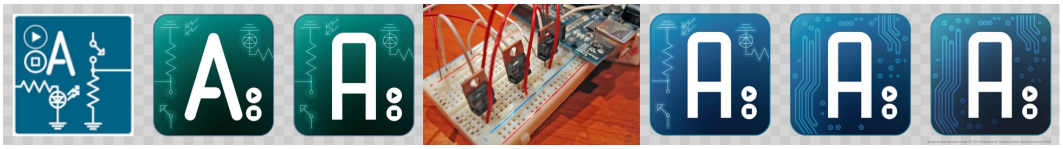
Atenção: Só ligar à tomada na presença do docente.

Pontos de interesse:

O relé é um elemento essencial em qualquer trabalho de automação. No entanto, ao contrário dos transístores só possui duas posições: ligada e desligada.

Consultar também:

http://quarkstream.wordpress.com/2009/12/11/arduino-8-relays/



Ficha 8. Medidor de distâncias

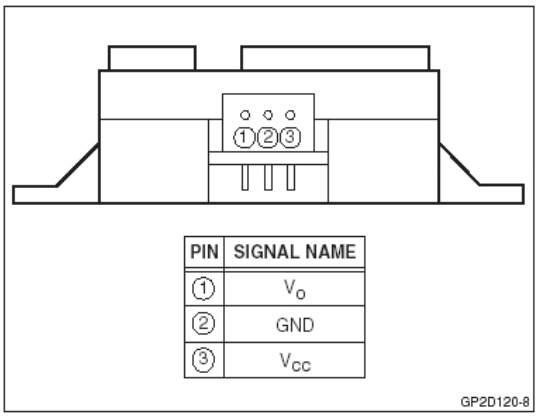
A miniaturização da electrónica e o custo reduzido dos circuitos integrados permite dotar os sensores com toda a electrónica necessária para a sua fácil integração. O medidor de distâncias da Sharp é um exemplo desta miniaturização, na medida em que todo o circuito do GP2D120 está integrado no próprio sensor. O sensor mede de 4 a 30cm, e dá um sinal analógico que tem uma relação curvilinear com a distância.

FEATURES

- Analog output
- Effective range: 4 to 30 cm
- Typical response time: 39 ms
- Typical start up delay: 44 ms
- Average Current Consumption: 33 mA

DESCRIPTION

The GP2D120 is a distance measuring sensor with integrated signal processing and analog voltage output.



Absolute Maximum Ratings

Ta = 25°C, V_{CC} = 5 VDC

PARAMETER	SYMBOL	RATING	UNIT
Supply Voltage	V_{CC}	-0.3 to +7	V
Output Terminal Voltage	V_O	-0.3 to (V_{CC} +0.3)	V
Operating Temperature	Topr	-10 to +60	°C
Storage Temperature	Tstg	-40 to +70	°C

Operating Supply Voltage

PARAMETER	SYMBOL	RATING	UNIT
Operating Supply Voltage	V_{CC}	4.5 to 5.5	V

Electro-optical Characteristics

Ta = 25°C, V_{CC} = 5 VDC

PARAMETER	SYMBOL	CONDITIONS	MIN.	TYP.	MAX.	UNIT	NOTES
Measuring Distance Range	ΔL		4	—	30	cm	1, 2
Output Terminal Voltage	V_O	L = 30 cm	0.25	0.4	0.55	V	1, 2
Output Voltage Difference	ΔV_O	Output change at ΔL (30 cm – 4 cm)	1.95	2.25	2.55	V	1, 2
Average Supply Current	I_{CC}	L = 30 cm	—	33	50	mA	1, 2

NOTES:
1. Measurements made with Kodak R-27 Gray Card, using the white side, (90% reflectivity).
2. L = Distance to reflective object.

O sensor dá um sinal em volts, que está relacionado com a distância através de uma curva.

Escreve um programa para fazer a leitura.

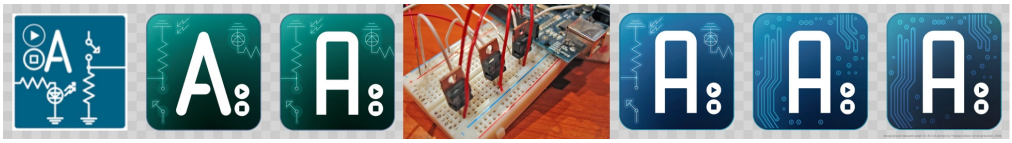
```
float voltagem; //voltagem calculada
float leitura = 0; //leitura feita com o sensor
int cm;

void setup() {

 Serial.begin(9600);
   pinMode(0, INPUT);

}

void loop() {
 leitura = analogRead(A0);
          voltagem = leitura * 0.0049;
   cm = 60.495 * pow(voltagem,-1.1904);
   Serial.print ("distancia até ao objecto: ");
   Serial.print(cm, DEC);
   Serial.println(" cm");
   delay (1000);

}
```

Projecto 1

Agora leia uns 5-6 pontos e utilize uma folha de cálculo para determinar a equação correcta.

Projecto 2

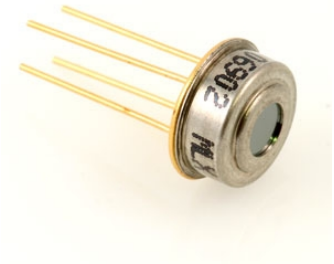
Faça um aparelho que mede a velocidade dos objectos e acende uma lâmpada vermelha se a velocidade for superior a um determinado valor.



Ficha 9. Termómetro infravermelhos

Neste projecto vamos utilizar um termómetro de infra-vermelhos para detectar a temperatura sem contacto. Os termómetros de infra-vermelhos medem a radiação emitida pelos corpos e têm duas vantagens sobre os termómetros normais: primeiro: não é necessário tocar no objecto, pelo que a medição pode ser feita à distância e segundo: medem a temperatura da radiação emitida pelo objecto, pelo que a partida não são influenciados pela temperatura ambiente.

O termómetro em causa é o MLX90614 da Melexis. Este sensor tem a particularidade de já incluir todo o circuito necessário para a leitura e processamento do sinal. Adicionalmente utiliza o Bus I^2C para a comunicação digital das leituras, o que aumenta bastante a sua precisão. Já utilizamos o Bus I2C e sabemos que trabalha com os pinos analógicos A4 e A5, para além dos 5V e terra.



Vai precisar de:
1 sensor de infravermelhos MLX90614 Axx, de 5 V (ou 3.3V)
1 condensador de 0.1 uF

Deve ligar os cabos de acordo com o seguinte esquema:

```
- Pin 1 SCL ao Arduino pin analog 5
- Pin 2 SDA ao Arduino pin analog 4
- Pin 3 Vdd ao Arduino pin +5V
- Pin 4 Vss ao Arduino pin GND


ligar um condensador de  100nf entre Vdd e Vss
```

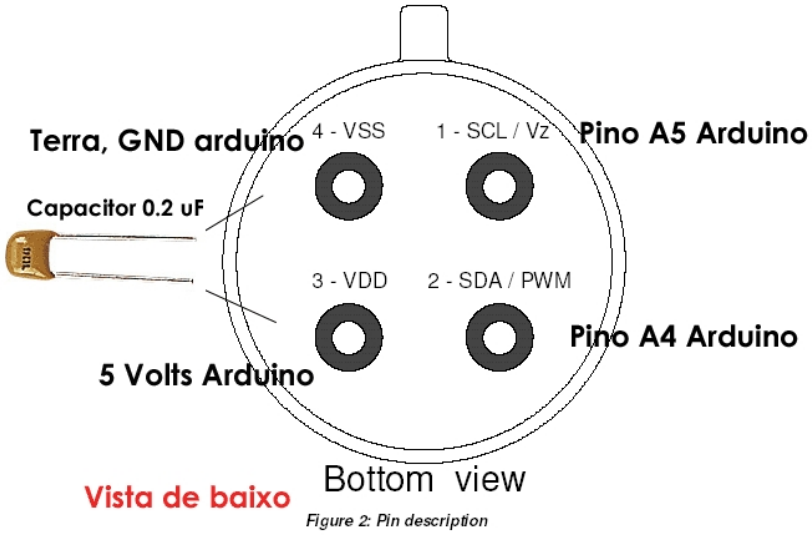


Figure 2: Pin description

Antes de abrir o interface do Arduino, deve copiar os ficheiros I2cmaster.h e twistmaster.cpp para o directório I2cmaster no seu directório libraries do directório Arduino.

O código é:

```
#include <i2cmaster.h>

char st1[30];

void setup()
{
  Serial.begin(9600);
  Serial.println("Termometro de infravermelhos Melexis
MLX90614");

  PORTC = (1 << PORTC4) | (1 << PORTC5);  //liga a resistênica
interna

}

void loop()
{

  long int tpl;

  tpl=readMLXtemperature(0); // ler temperature objecto
  tpl = tpl *10;
  tpl = tpl / 5;
  tpl=tpl-27315;
  sprintf(st1,"object temp: %03li.%li",tpl/100, abs(tpl %100) );
  Serial.print(st1);
  Serial.print("   ");

  tpl=readMLXtemperature(1); // ler temperature ambiente
```

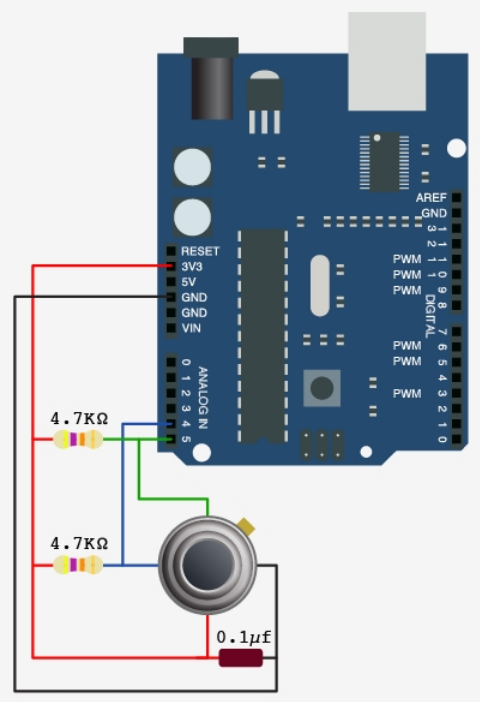
```
  tpl = tpl *10;
  tpl = tpl / 5;
  tpl=tpl-27315;
  sprintf(st1,"ambient temp: %03li.%li",tpl/100, tpl %100 );
  Serial.print(st1);
  Serial.print("   ");
  Serial.println("");
  delay(2000);

}
//*************************************************************
***
// Rutina para ler as temperatures do MLX90614 no bus i2c
long int readMLXtemperature(int TaTo) {
    long int lii;
    int dlsb,dmsb,pec;
    int dev = 0x5A<<1;

  i2c_init();
  i2c_start_wait(dev+I2C_WRITE);  // establece o endreço e o
modo (escrever)
  if (TaTo) i2c_write(0x06); else i2c_write(0x07);  // ler temp
objecto ou temp ambient

  i2c_rep_start(dev+I2C_READ);    // establece o endereco e
modo (leitura)
  dlsb = i2c_readAck();       // ler data lsb
  dmsb = i2c_readAck();      // ler data msb
  pec = i2c_readNak();
  i2c_stop();

  lii=dmsb*0x100+dlsb;
  return(lii);
}
```

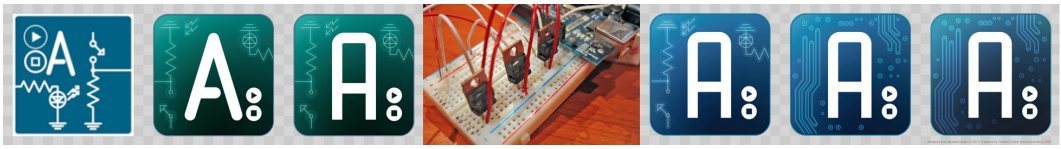
Pontos de interesse:

O condensador é uma pequena bateria recarregável. Muitas vezes é utilizado à entrada de sensores para estabilizar o corrente: Se temos um sensor ou um aparelho que exige picos rápidos de corrente, isso pode destabilizar a fonte de alimentação. Ao introduzirmos paralelamente um pequeno condensador, este consegue absorver a corrente quando não é preciso, e depois ceder energia quando há uma grande necessidade.

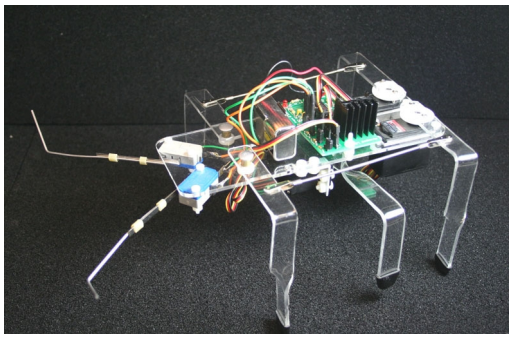
Os condensadores maiores tem polaridade, indicado pelo comprimento das pernas (perna maior é positivo).

Com o Melexis de 3 Volts, pode-se fazer:





Ficha 10. Comando de servo-motores com o Arduino



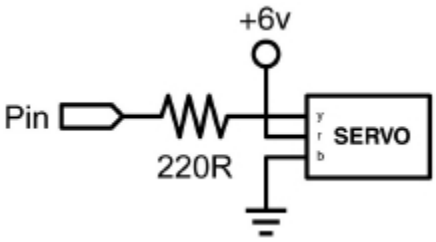
Servomotor é um motor que apresenta movimento proporcional a um comando, em vez de girar ou se mover livremente sem um controle mais efectivo de posição como a maioria dos motores; servomotores são dispositivos de malha fechada, ou seja: recebem um sinal de controle; verificam a posição atual; atuam no sistema indo para a posição desejada

Em contraste com os motores contínuos que giram indefinidamente, o eixo dos servo motores possui a liberdade de apenas cerca de 180º graus mas são precisos quanto a posição.

São muito utilizados na robótica e na modelação para obter um movimento preciso e um posicionamento fácil.

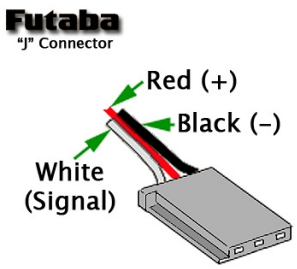
Neste projecto vamos utilizar servo-motores Futaba S3003, que tem um movimento de 180º, de acordo com a frequência de entrada. Devido ao grande consumo de energia devem ter uma alimentação separada.

O esquema eléctrico é:



Todos os servo-motores têm três cabos:

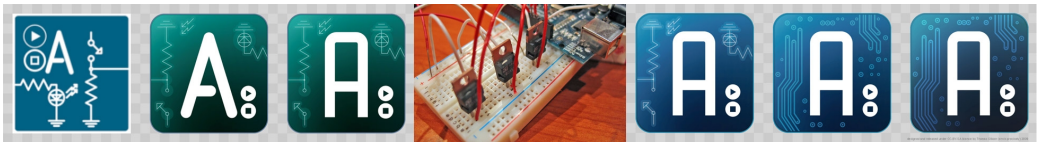
Preto é a terra, o vermelho é alimentação (~4.8-6V) e o amarelo ou branco é o sinal (3-5V).



Enquanto que os cabos vermelho e preto fornecem energia ao motor, o cabo sinal comanda a operação do servomotor. O princípio de funcionamento é o envio de uma onda lógica quadrada com um determinado comprimento de onda para o servomotor que especifica o seu ângulo. O comprimento de onda dita exactamente o ângulo.

O programa é o que consta em baixo, que utiliza o pino 2 do Arduino

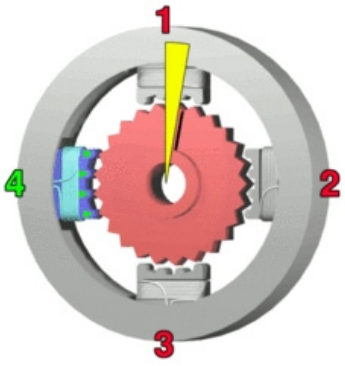
```
int angulo; // ângulo de 0-180
int pulseWidth; // função do servoPulse
int pino = 2;

void setup()
{
pinMode(2, OUTPUT);
}

void servoPulse(int pino, int angulo)
{
pulseWidth = (angulo * 10) + 600; // establece o delay
digitalWrite(2, HIGH);
delayMicroseconds(pulseWidth); // pausa em microsegundos
digitalWrite(2, LOW);
}

void loop()
{
// começa a rotação entre os 10º e os 170º.
for (angulo=10; angulo<=170; angulo++)
{
servoPulse(2, angulo); // O pino e o angulo
delay(20);
}
// o servo volta dos 170º para os 10º
for (angulo=170; angulo>=10; angulo--)
{
servoPulse(2, angulo); // O pino e o angulo
delay(20);
}
}
```

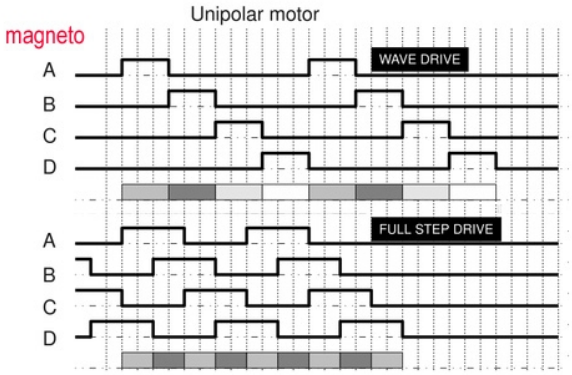


Ficha 10A. Motores de passo com o Arduino

Introdução

O motor de *passos* é um motor de precisão que se utiliza, por exemplo, nas impressoras para posicionar o papel no local pretendido. Uma volta completa é dividida num grande número de *passos* que são controlados individualmente. O movimento do motor pode ser controlado com grande precisão, pois em cada passo o motor só roda um determinado ângulo[1].
Num motor deste tipo podem existir quatro magnetos que são ligados sequencialmente, o que faz com que o motor rode em pequenos passos, sendo atraído pela sequencia dos magnetos.



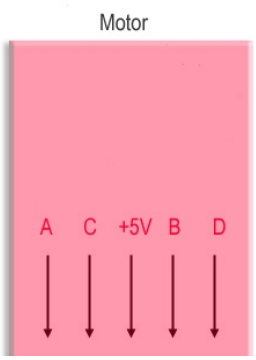


A figura a direita pretende apresentar duas formas de escalonamento dos passos. Na opção de cima se ligam os quatro magnetos por ordem. Na opção de baixo, também se ligam por ordem, mas dois de cada vez. Esta é a melhor opção e permite o dobro de potência.

[1] no caso particular do motor que iremos utilizar, o PM35L, cada passo é 7,5º, o que significa que para fazer os 360º, são necessários 48 passos.

Projecto 1

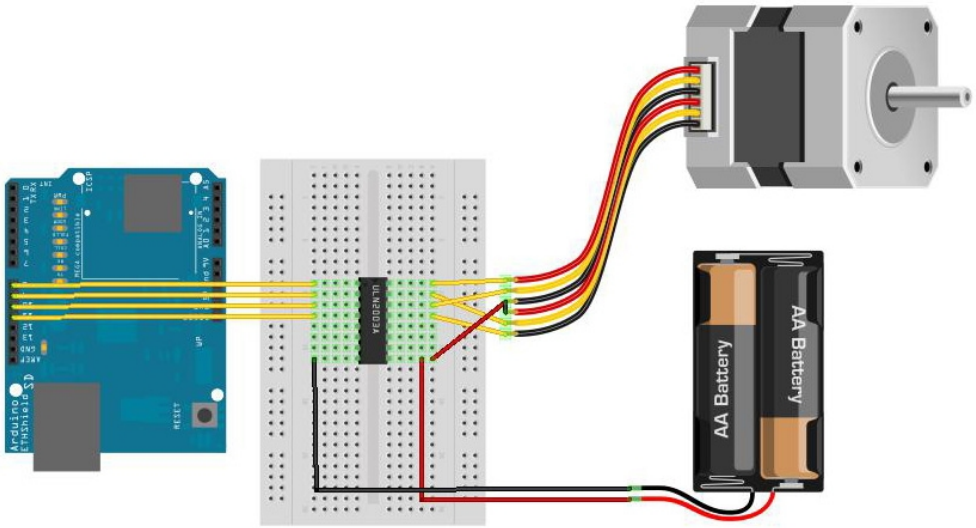
Neste projecto vai-se utilizar o motor de passos PM35L que se pode tirar duma impressora HP ou Epson. O motor tem 5 cabos que são, por ordem A, B, +5V, C e D. A ordem é importante para depois os poder ligar ao Arduino. A ordem dos cabos a saída do motor está indicada na figura em baixo.
A direita está um quadro com a ordem pela qual os magnetos devem ser ligados pelo Arduino para se obter uma rotação.[2]



	A	B	C	D	+5V
	A1 Black	A3 Brown	B1 Orange	B3 Yellow	A2/B2 Red/Red
1	–			–	+
2	–		–		+
3		–	–		+
4		–		–	+

CCW Rotation
CW Rotation

Como o motor utiliza 500mA, não poderá ser ligado directamente ao Arduino. Para isso vamos precisar de um transístor. Neste trabalho vamos utilizar o ULN2003A[3] que são sete transístores montados numa só peça, o que é muito conveniente.



O programa simplesmente tem de ligar os magnetos pela ordem apresentada no quadro em cima, para que o motor rode.

[2] CW significa Clock wise, ou seja segundo ponteiros de relógio. CCW é counter clock wise, e significa sentido contrário aos ponteiros de relógio
[3] O UNL2003A está especialmente concebido para trabalhar com os 5V provenientes do Arduino, também chamado de TTL.

Importante: Recomenda-se programar o Arduino e depois o desligar do computador. Só depois deve ligar o Arduino à fonte de alimentação externa.

```
// Programa para motor de passo unipolar PML35
//ligar nos pinos 8,9,10 e 11 do Arduino

int A = 8;
int B = 9;
int C = 10;
int D = 11;
int pausa = 100;

void setup() {
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
}

void loop() {
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  delay(pausa);

  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  delay(pausa);

  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, HIGH);
  digitalWrite(D, LOW);
  delay(pausa);

  digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  delay(pausa);
}
```

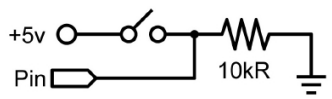
Projecto 2

Altere o programa para que o motor acelere e abrande sozinho.

Projecto 3

Altera o programa para o motor acelerar e abrandar quando se carregue em botões.

digital input



Em termos de programação, primeiro deve declarar (no void setup) o pino onde vai ligar o botão:

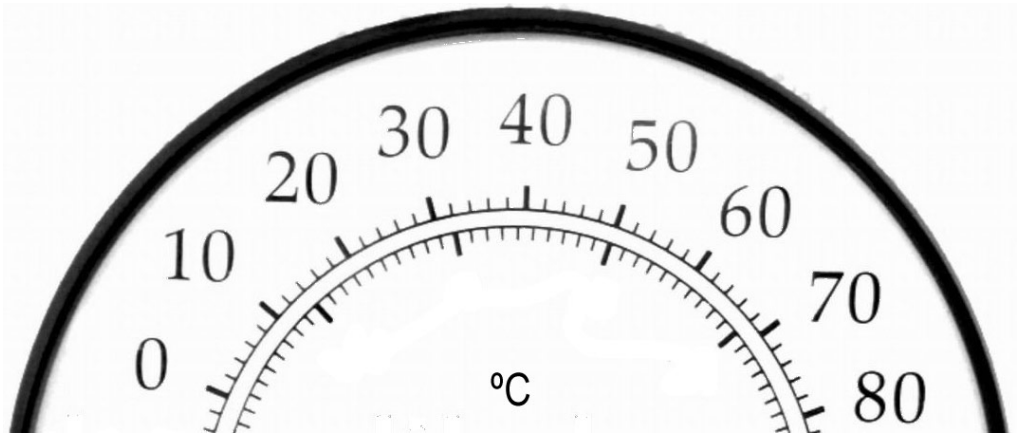
```
pinMode(3, INPUT);
```

e depois no void loop, incluir um comando para ler:

```
if (digitalRead(3)==HIGH) {

}
```

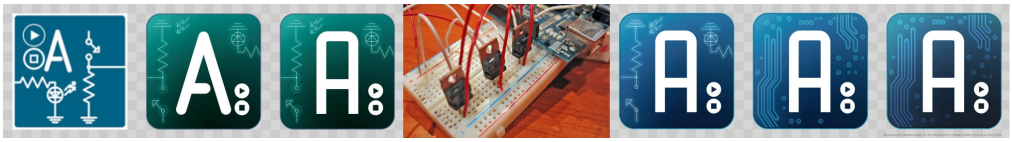
Projecto 3

Faça um termómetro analógico com um ponteiro que indica a temperatura na folha em baixo.



Pontos de interesse:

Alterando a pausa entre os comandos, pode-se alterar a velocidade da rotação do motor.

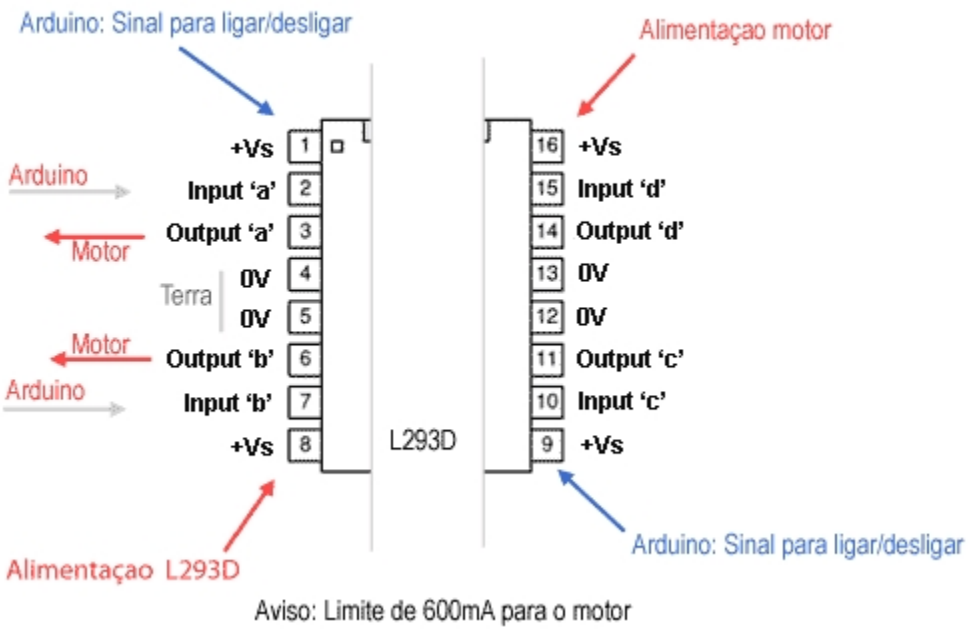


Ficha 10B. Motores reversíveis com o Arduino

Introdução

Muitas vezes se pretende um motor que possa funcionar em ambos os sentidos. Por exemplo um portão deve poder abrir e fechar. Um elevador também deve poder subir e descer. Isso é possível utilizando um chip especializado que é chamado uma ponte H, ou H bridge. Resumidamente o chip permite inverter o sentido da corrente que vai para o motor de forma muito simples.

O chip basicamente tem dois pares de entradas que ligam ao Arduino e dois pares de saídas, que ligam a dois motores. Portanto cada par de pinos do Arduino podem comandar um motor. Para rodar o motor num sentido basta ligar um dos pinos. Para rodar no sentido inverso, liga-se o outro pino e desliga-se o primeiro. Qualquer outra combinação faz o motor parar.
No desenho em baixo está representado o chip L293D. No desenho o chip está dividido em duas partes, pois cada metade é totalmente separada da outra.



Aviso: Limite de 600mA para o motor

Projecto 1

O programa para rodar o motor num sentido e outro é bastante simples e consiste basicamente em ligar o pino A e desligar o B, e depois fazer o inverso para o motor rodar no sentido inverso.

Vamos utilizar uma drive de CDs. Vamos abrir a porta e depois fechar a porta. Os drives de CDs tem um motor Rf-300e que utiliza 3-7 Volts, a uma intensidade de 0.02 a 0.39 A.

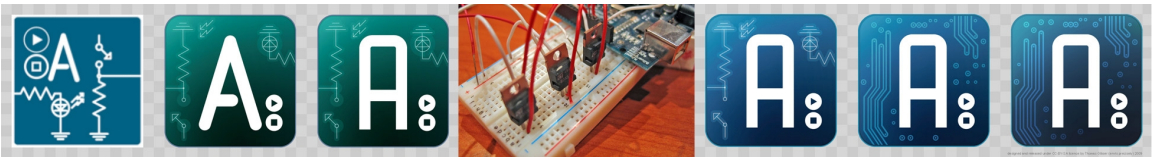
```
int A = 2;
int B = 3;
int Ligar = 4;
int state = 0;

void setup() {
pinMode(A, OUTPUT);
pinMode(B, OUTPUT);
pinMode(Ligar, INPUT);
}

void loop() {
state = digitalRead(Ligar);

switch(state) {
case 0:
digitalWrite(A, HIGH);
digitalWrite(B, LOW);
break;
case 1:
digitalWrite(A, LOW);
digitalWrite(B, HIGH);
break;
}
}
```

Projecto 2

Se utilizar um pino PWM do arduino, poderá variar a velocidade do motor. Experimente.

 Pontos de interesse:

Este método permite abrir e fechar uma cancela de forma bastante simples.



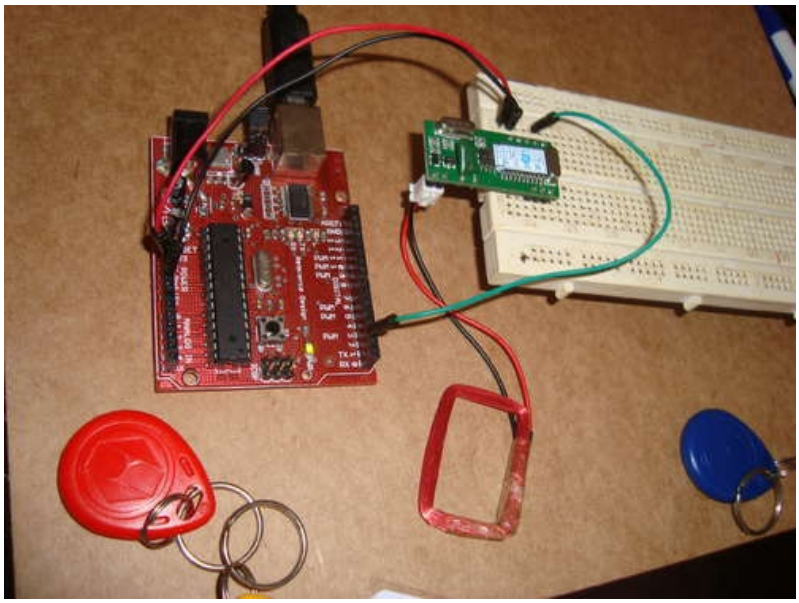
Ficha 11. Transponders

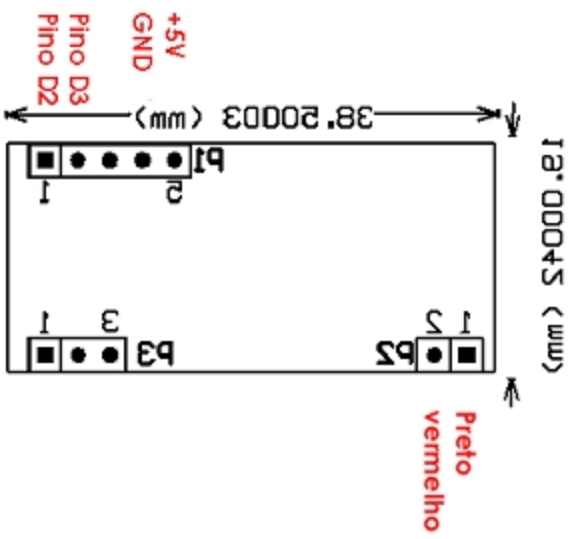
INTRODUÇÃO AO SISTEMA RFID ou transponders

O sistema RFID – Radio Frequency Indentification, ou em português Identificação por Rádiofrequencia, é a mais recente tecnologia em desenvolvimento. Os sistemas *RFID* são sistemas automáticos de identificação que, usando sinais de radio-freqüência, propiciam a identificação e localização automática de aparelhos portadores de dados – etiquetas – que vêm apensos a itens. As etiquetas podem ser detectadas por radio-freqüências a uma distância remota por um leitor, sem a necessidade de contacto físico ou de este estar na linha de "visão" do leitor, pois as radio-freqüências atravessam praticamente qualquer material sólido. Um sistema de gestão computadorizado pode usar a informação recolhida pelos leitores e processá-la de acordo com a aplicação pretendida, como por exemplo, a gestão logística. Os sistemas *RFID*, permitem o armazenamento de mais informação e não necessitam da intervenção de operadores para a realização da leitura da etiqueta. Adicionalmente, e entre outras vantagens, as etiquetas *RFID* são mais robustas em condições adversas.

Para este trabalho vamos utilizar o módulo RDM 630 da Seeeduino. Este módulo funciona a 125 kHz e comunica com o Arduino a 9600, 8, N, 1.

O Módulo comunica com o Arduino através de um protocolo igual à porta série. Mas se for instalado com os pinos normais da comunicação (0 e 1) depois não se pode comunicar com o computador. Assim, instala-se nos pinos 2 (Rx do Arudino, e portanto Pin1 do módulo) e pino 3 (Tx do arduino e portanto Pin2 do módulo), utilizando a biblioteca #include <SoftwareSerial.h>




+5V
GND
Pino D3
Pino D2
38.50003 (mm)
P1
P2
P3
Preto
vermelho
19.00042 (mm)

2. Pin definition（TTL interface RS232 data format）

P1：

PIN1	TX
PIN2	RX
PIN3	
PIN4	GND
PIN5	+5V(DC)

P2:

PIN1	ANT1
PIN2	ANT2

P3:

PIN1	LED
PIN2	+5V(DC)
PIN3	GND

O Programa:

```
/**
 * Este projecto utiliza um leitor RDM630 RFID da Seeed Studio para encontrar
 * 125KHz RFID tags.
 * O programa lê o código do cartão e se fizer parte da lista diz o nome da pessoa.
 * Para manter os pintos TX e RX do Arduino livres, o programa utiliza a biblioteca
 * SoftwareSerial para comunicar com o módulo **/
#include <SoftwareSerial.h>

// Utilize os pinos pins D2 / D3 do Arduino:
#define rxPin 2
#define txPin 3

// Create a software serial object for the connection to the RFID module
SoftwareSerial rfid = SoftwareSerial( rxPin, txPin );

#define ledPin 13


// Specify how long the strike plate should be held open.
#define unlockSeconds 2

// A lista dos tansponders reconhecidos
char* allowedTags[] = {
```

```
  "3000AF1134",        // Tag 1
  "3C00CE46EA",        // Tag 2
  "0413BBBF23",        // Tag 3
};

// Lista dos nomes associados com os cartões
char* tagName[] = {
  "Maria",     // Tag 1
  "Manuel",     // Tag 2
  "Joana",      // Tag 3
};

// Check the number of tags defined
int numberOfTags = sizeof(allowedTags)/sizeof(allowedTags[0]);

int incomingByte = 0;    // To store incoming serial data

void setup() {
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);

  Serial.begin(9600);   // Serial port for connection to host
  rfid.begin(9600);     // Serial port for connection to RFID module

  Serial.println("RFID reader starting up");
}

void loop() {
  byte i        = 0;
  byte val      = 0;
  byte checksum  = 0;
  byte bytesRead = 0;
  byte tempByte  = 0;
  byte tagBytes[6];    // "Unique" tags are only 5 bytes but we need an extra byte for the checksum
  char tagValue[10];

  // Read from the RFID module. Because this connection uses SoftwareSerial
  // there is no equivalent to the Serial.available() function, so at this
  // point the program blocks while waiting for a value from the module
  if((val = rfid.read()) == 2) {        // Check for header
    bytesRead = 0;
    while (bytesRead < 12) {           // Read 10 digit code + 2 digit checksum
      val = rfid.read();

      // Append the first 10 bytes (0 to 9) to the raw tag value
      if (bytesRead < 10)
```

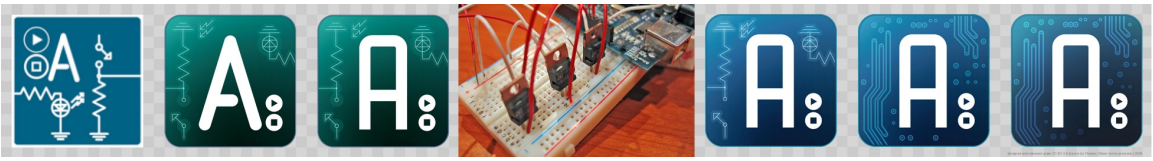
```
  {
    tagValue[bytesRead] = val;
  }

  // Check if this is a header or stop byte before the 10 digit reading is complete
  if((val == 0x0D)||(val == 0x0A)||(val == 0x03)||(val == 0x02)) {
    break;                        // Stop reading
  }

  // Ascii/Hex conversion:
  if ((val >= '0') && (val <= '9')) {
    val = val - '0';
  }
  else if ((val >= 'A') && (val <= 'F')) {
    val = 10 + val - 'A';
  }

  // Every two hex-digits, add a byte to the code:
  if (bytesRead & 1 == 1) {
    // Make space for this hex-digit by shifting the previous digit 4 bits to the left
    tagBytes[bytesRead >> 1] = (val | (tempByte << 4));

    if (bytesRead >> 1 != 5) {                // If we're at the checksum byte,
      checksum ^= tagBytes[bytesRead >> 1];   // Calculate the checksum... (XOR)
    };
  } else {
    tempByte = val;                           // Store the first hex digit first
  };

  bytesRead++;                                // Ready to read next digit
}

// Send the result to the host connected via USB
if (bytesRead == 12) {                        // 12 digit read is complete
  tagValue[10] = '\0';                        // Null-terminate the string

  Serial.print("Tag read: ");
  for (i=0; i<5; i++) {
    // Add a leading 0 to pad out values below 16
    if (tagBytes[i] < 16) {
      Serial.print("0");
    }
    Serial.print(tagBytes[i], HEX);
  }
  Serial.println();
```

```
    Serial.print("Checksum: ");
    Serial.print(tagBytes[5], HEX);
    Serial.println(tagBytes[5] == checksum ? " -- passed." : " -- error.");

    // Show the raw tag value
    //Serial.print("VALUE: ");
    //Serial.println(tagValue);

    // Search the tag database for this particular tag
    int tagId = findTag( tagValue );

    // Only fire the strike plate if this tag was found in the database
    if( tagId > 0 )
    {
      Serial.print("Authorized tag ID ");
      Serial.print(tagId);
      Serial.print(": unlocking for ");
      Serial.println(tagName[tagId - 1]);   // Get the name for this tag from the database
      unlock();                             // Fire the strike plate to open the lock
    } else {
      Serial.println("Tag not authorized");
    }
    Serial.println();     // Blank separator line in output
   }

   bytesRead = 0;
  }
}

/**
 * Fire the relay to activate the strike plate for the configured
 * number of seconds.
 */
void unlock() {
  digitalWrite(ledPin, HIGH);
    delay(unlockSeconds * 1000);
   digitalWrite(ledPin, LOW);
}

/**
 * Search for a specific tag in the database
 */
int findTag( char tagValue[10] ) {
  for (int thisCard = 0; thisCard < numberOfTags; thisCard++) {
    // Check if the tag value matches this row in the tag database
    if(strcmp(tagValue, allowedTags[thisCard]) == 0)
```

```
  {
    // The row in the database starts at 0, so add 1 to the result so
    // that the card ID starts from 1 instead (0 represents "no match")
    return(thisCard + 1);
  }
 }
 // If we don't find the tag return a tag ID of 0 to show there was no match
 return(0);
}
```

Mais informações

http://www.practicalarduino.com/projects/rfid-access-control-system
http://www.instructables.com/id/Arduino-and-RFID-from-seeedstudio/



Ficha 12. LCD TFT

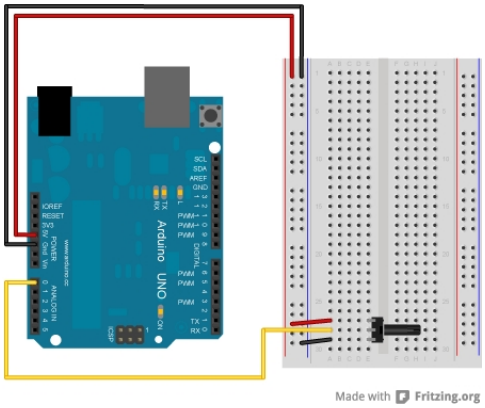
Objectivos

Electrónica	Programação
	Variáveis indexadas: É uma variável cujos valores são dados numa tabela, e podem ser chamadas através do número de índice. É por exemplo o caso do variável IDuracao, que pode assumir diversos valores, conforme o índice. int IDuracao[ ] = {4, 8, 8, 4, 4, 4, 4, 4}; O IDuracao[1] será 4, enquanto que o IDuracao[2] será 8.
	Utilização de cores RGB

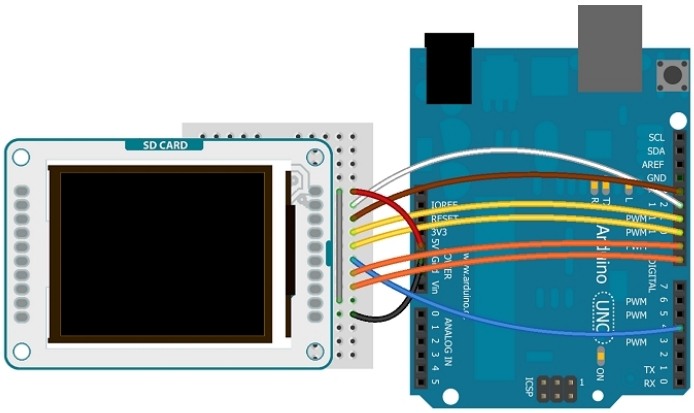
Neste projecto vamos utilizar um LCD para mostrar dados. O LCD é a cores, pelo que cada pixel é constituído por 3 pequenos pixels, com as três cores: RGB- vermelho, verde e azul. A cor desejada é obtida pela mistura das três cores.
Os pinos do LCD são:

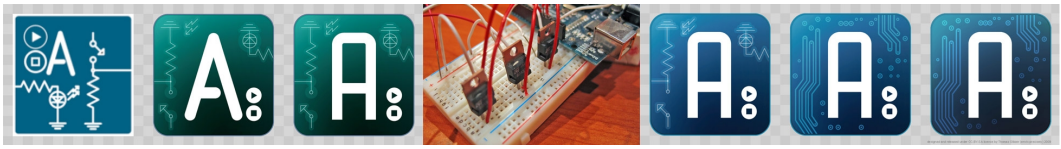
+5V: +5V
MISO: pin 12
SCK: pin 13
MOSI: pin 11
LCD CS: pin 10
SD CS: pin 4
D/C: pin 9
RESET: pin 8
BL: +5V
GND: GND

Neste primeiro exemplo vamos utilizar o LCD para mostrar os valores obtidos de um sensor ligado ao A0.

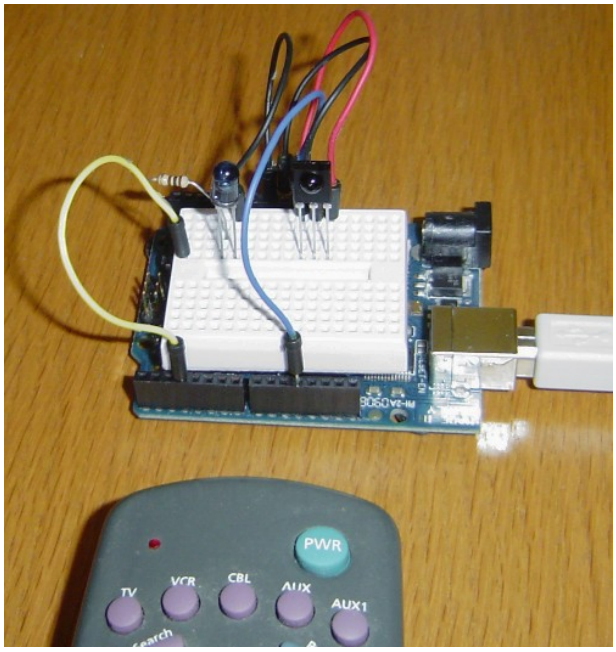



```
/*
  Ligar sensor pino A0
   http://arduino.cc/en/Tutorial/TFTDisplayText
  */
#include <TFT.h>
#include <SPI.h>
TFT TFTscreen = TFT(10, 9, 8);

// variavel texto a ser escrito com 4 caracters
char leitura[4];

void setup() {
  TFTscreen.begin();
  // pintar o texto com a cor 250,0,0
  TFTscreen.background(250, 0, 0);
  // cor da letra: RGB- branco
  TFTscreen.stroke(255,255,255);
  // tamanho da letra
  TFTscreen.setTextSize(2);
  // escrever o texto no canto superior esquerdo: 0,0
  TFTscreen.text("Leitura:\n ",0,0);
  // tamanho da leitura para o resto
  TFTscreen.setTextSize(5);
}

void loop() {
   String sensorVal = String(analogRead(A0));
  // converter a leitura a um variael de caracters
  sensorVal.toCharArray(leitura, 4);
  // cor da letra
  TFTscreen.stroke(255,255,255);
  // escreer no ponto 0,20
  TFTscreen.text(leitura, 0, 20);
  delay(1000);
  // Limpar com a cor 250,0,0, escrevendo por cima do texto
  TFTscreen.stroke(250,0,0);
  TFTscreen.text(leitura, 0, 20);
}
```


SD CARD
SCL
SDA
AREF
GND
IOREF
RESET
3V3
5V
GND
GND
Vin
ANALOG IN
0 1 2 3 4 5
DIGITAL
PWM
PWM
PWM
PWM
PWM
TX
RX
ON
ICSP
Arduino
UNO
www.arduino.cc
Made with Fritzing.org

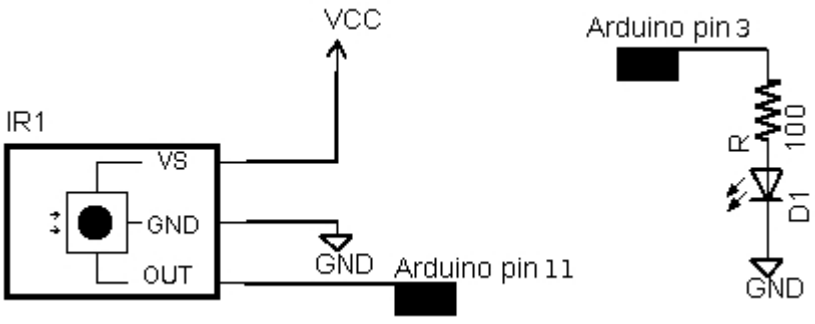


Ficha 13. Comando por infravermelhos



O controlo remoto ocupa hoje um lugar central em aplicações electróncias, pois permite uma comunicação rápida e fácil entre o utilizador e o equipamento. Por outro lado é um protocolo económico e elegante de implementar.

Basicamente existem duas partes: O emissor, que é um LED e um receptor



Esses elementos trabalham a 5V, pelo que é muito fácil os integrar utilizando o Arduino.

Poderá comprar o equipamento, ou aproveitar os de um electrodoméstico velho. Poderá também simplesmente adquirir as duas peças necessárias numa loja. O transmissor pode ser o NTE 3027 LED e o receptor o Panasonic PNA4602.

A biblioteca IRremote trata de ler os códigos recebidos e os decifrar. Infelizmente cada marca utiliza os seus próprios códigos. Essa biblioteca traz os códigos para Sony, NEC e Philips e tem uma aplicação (IRrecvDump) que permite ler os códigos recebidos para poderem ser descodificados por nós.

Para enviar, um programa simples é:

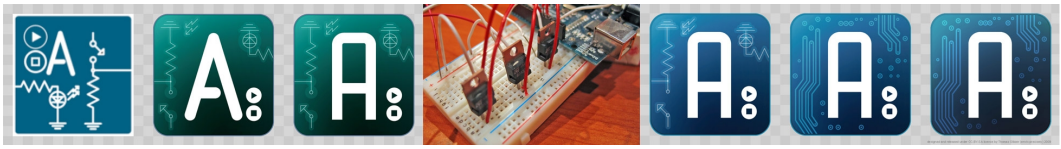
```
#include <IRremote.h>
IRsend irsend;

void setup()
{
  Serial.begin(9600);
}

void loop() {
  if (Serial.read() != -1) {
    for (int i = 0; i < 3; i++) {
      irsend.sendSony(0xa90, 12); // Sony TV power code
      delay(100);
    }
  }
}
```

e para receber:

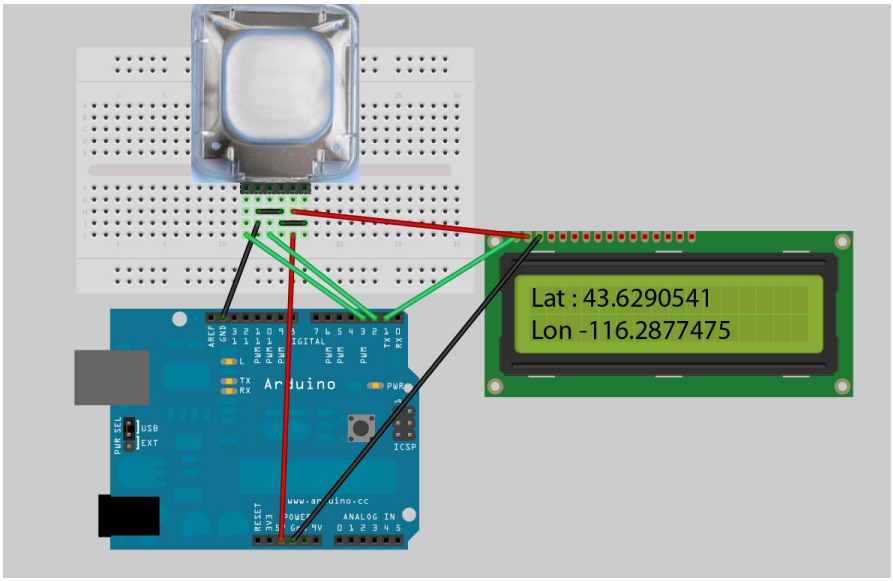
```
#include <IRremote.h>

int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600);
```

```
  irrecv.enableIRIn(); // Start the receiver
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
}
```

Pontos de interesse:



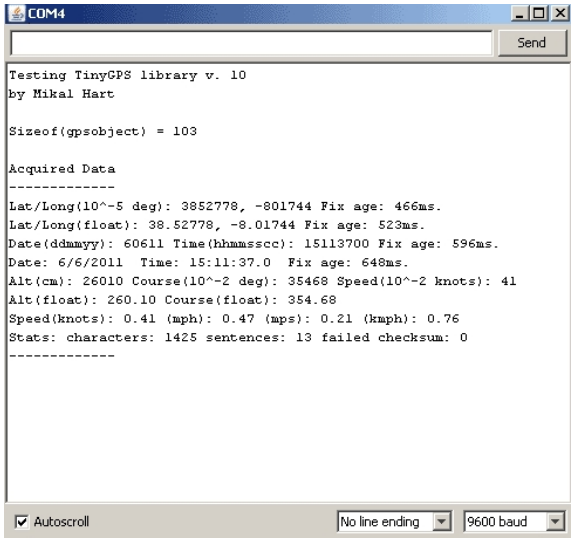
Ficha 14. GPS via porta série



Pontos de interesse:

O sistema de posicionamento geográfico, GPS é uma ferramenta muito útil para localização de objectos, mapeamento, condução e até levantamentos topográficos. O sistema consiste numa rede de satélites que constituem pontos de referência no espaço. O receptor comunica com um conjunto de satélites (quanto maior o número, melhor) e com base no tempo de resposta a cada um, sabe a distância até cada um deles. A partir dessa informação e do conhecimento de posição de cada um, o GPS consegue saber a sua posição.

No Arduino é possível construir com alguma facilidade um GPS, que transmite a informação para o computador via porta série.

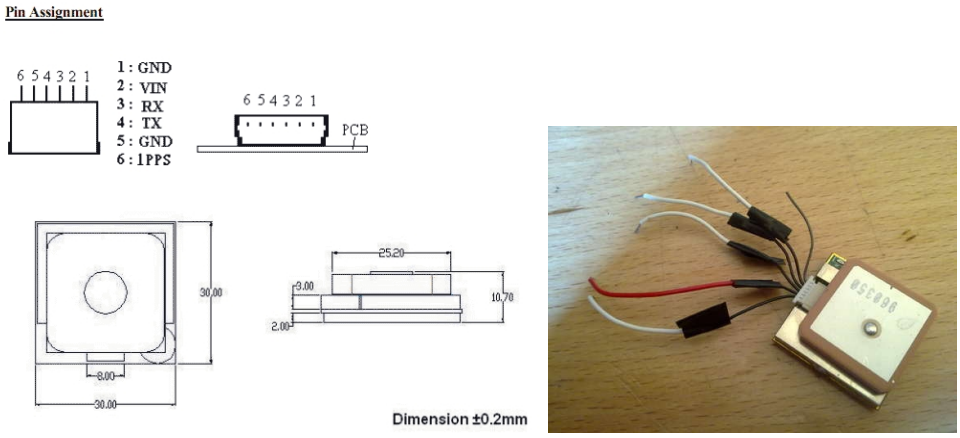


Vamos utilizar o GPS EM-406A da GlobalSat. Este receptor utiliza o chip SiRF Star III, e vai ser ligado através de uma porta série virtual, utilizando os pinos D2 e D3.

Vai precisar de copiar para a sua biblioteca, o biblioteca TinyGPS.h, que serve para simplificar e traduzir toda a informação enviada pelo GPS.

As ligações são:

1 - GND
2 – 5V
3 – Pino 3 (Rx do GPS) – cabo branco
4 – Pino 2 (Tx do GPS) – cabo laranja



O código utiliza a biblioteca NewSoftSerial para criar uma porta Serial virtual. O biblioteca TinyGPS simplifica e filtra a informação recebida do GPS

```
#include <NewSoftSerial.h>
#include <TinyGPS.h>

/* This sample code demonstrates the normal use of a TinyGPS object.
   It requires the use of NewSoftSerial, and assumes that you have a
   4800-baud serial GPS device hooked up on pins 2(rx) and 3(tx).
*/

TinyGPS gps;
NewSoftSerial nss(2, 3);

void gpsdump(TinyGPS &gps);
bool feedgps();
void printFloat(double f, int digits = 2);

void setup()
{
  Serial.begin(9600);
  nss.begin(4800);

  Serial.print("TinyGPS library v. ");
Serial.println(TinyGPS::library_version());
  Serial.println("by Mikal Hart");
  Serial.println();
  Serial.print("Sizeof(gpsobject) = "); Serial.println(sizeof(TinyGPS));
  Serial.println();
}

void loop()
{
  bool newdata = false;
  unsigned long start = millis();

  // Every 5 seconds we print an update
  while (millis() - start < 5000)
  {
    if (feedgps())
          newdata = true;
  }

  if (newdata)
  {
    Serial.println("Dados adquiridos");
    Serial.println("-------------");
    gpsdump(gps);
    Serial.println("-------------");
    Serial.println();
  }
}
```

```
void printFloat(double number, int digits)
{
  // Handle negative numbers
  if (number < 0.0)
  {
     Serial.print('-');
     number = -number;
  }

  // Round correctly so that print(1.999, 2) prints as "2.00"
  double rounding = 0.5;
  for (uint8_t i=0; i<digits; ++i)
    rounding /= 10.0;

  number += rounding;

  // Extract the integer part of the number and print it
  unsigned long int_part = (unsigned long)number;
  double remainder = number - (double)int_part;
  Serial.print(int_part);

  // Print the decimal point, but only if there are digits beyond
  if (digits > 0)
    Serial.print(".");

  // Extract digits from the remainder one at a time
  while (digits-- > 0)
  {
    remainder *= 10.0;
    int toPrint = int(remainder);
    Serial.print(toPrint);
    remainder -= toPrint;
  }
}

void gpsdump(TinyGPS &gps)
{
  long lat, lon;
  float flat, flon;
  unsigned long age, date, time, chars;
  int year;
  byte month, day, hour, minute, second, hundredths;
  unsigned short sentences, failed;

  gps.get_position(&lat, &lon, &age);
  Serial.print("Lat/Long(10^-5 deg): "); Serial.print(lat); Serial.print(", ");
Serial.print(lon);
  Serial.print(" Fix age: "); Serial.print(age); Serial.println("ms.");
```

```
  feedgps(); // If we don't feed the gps during this long routine, we may drop
characters and get checksum errors

  gps.f_get_position(&flat, &flon, &age);
  Serial.print("Lat/Long(float): "); printFloat(flat, 5); Serial.print(", ");
printFloat(flon, 5);
  Serial.print(" Fix age: "); Serial.print(age); Serial.println("ms.");

  feedgps();

  gps.get_datetime(&date, &time, &age);
  Serial.print("Date(ddmmyy): "); Serial.print(date); Serial.print("
Time(hhmmsscc): "); Serial.print(time);
  Serial.print(" Fix age: "); Serial.print(age); Serial.println("ms.");

  feedgps();

  gps.crack_datetime(&year, &month, &day, &hour, &minute, &second,
&hundredths, &age);
  Serial.print("Date: "); Serial.print(static_cast<int>(month)); Serial.print("/");
Serial.print(static_cast<int>(day)); Serial.print("/"); Serial.print(year);
  Serial.print("  Time: "); Serial.print(static_cast<int>(hour)); Serial.print(":");
Serial.print(static_cast<int>(minute)); Serial.print(":");
Serial.print(static_cast<int>(second)); Serial.print(".");
Serial.print(static_cast<int>(hundredths));
  Serial.print("  Fix age: ");  Serial.print(age); Serial.println("ms.");

  feedgps();

  Serial.print("Alt(cm): "); Serial.print(gps.altitude()); Serial.print("
Course(10^-2 deg): "); Serial.print(gps.course()); Serial.print(" Speed(10^-2
knots): "); Serial.println(gps.speed());
  Serial.print("Alt(float): "); printFloat(gps.f_altitude()); Serial.print("
Course(float): "); printFloat(gps.f_course()); Serial.println();
  Serial.print("Speed(knots): "); printFloat(gps.f_speed_knots()); Serial.print("
(mph): ");  printFloat(gps.f_speed_mph());
  Serial.print(" (mps): "); printFloat(gps.f_speed_mps()); Serial.print(" (kmph):
"); printFloat(gps.f_speed_kmph()); Serial.println();

  feedgps();

  gps.stats(&chars, &sentences, &failed);
  Serial.print("Stats: characters: "); Serial.print(chars); Serial.print(" sentences:
"); Serial.print(sentences); Serial.print(" failed checksum: ");
Serial.println(failed);
}

bool feedgps()
```

```
{
  while (nss.available())
  {
    if (gps.encode(nss.read()))
          return true;
  }
  return false;
}
```