Introdução ao Arduino

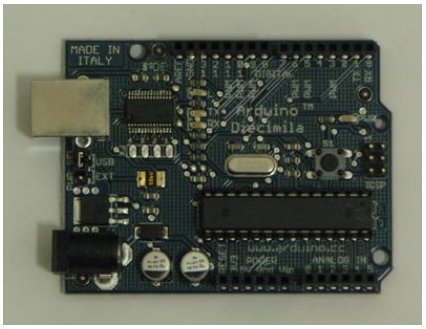
Introdução

*É objectivo deste artigo dar a conhecer o **Arduino.** Esta "ferramenta" com enormes potencialidades, que pode ser utilizada por todos, tenham ou não conhecimentos de electrónica devido à sua enorme simplicidade de utilização.*

*O **Arduino** tem ainda muita margem de desenvolvimento, começando como um pequeno projecto educacional evoluindo até aos dias de hoje. Dentro das suas vantagens pode-se encontrar o facto de ser **"Open-Source"**, correndo em ambiente Linux, Macintosh e Windows tendo ainda o aliciante de ser bastante económico comparativamente com "ferramentas" de iguais funcionalidades disponíveis no mercado.*

*Para apresentar este tema com maior simplicidade, o que não significa menor rigor, torna-se necessário dividi-lo em duas partes distintas: **Hardware** e **Software** e fazer a sua ligação.*

É indispensável referir que esta "ferramenta" possibilita abrir imensas "portas", devendo haver, por parte do leitor, um interesse em procurar, conhecer e aprender.



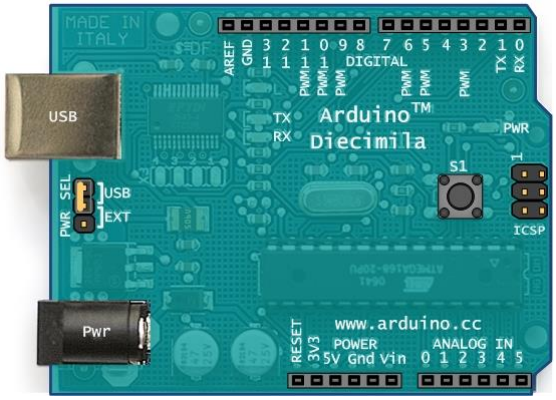
Hardware

*O Arduino pode ter várias apresentações sendo a base deste artigo o **Arduino Diecimilia** (baseado num microprocessador Atmega168 da Atmel).*

Convém dar especial atenção a este capítulo sobre hardware, pois é sobre ele que toda a programação se vai apoiar.

Não se torna necessário, como foi referido anteriormente, possuir elevados conhecimentos de electrónica, já que o único requisito realmente relevante é a vontade de aprender.

Observemos então pormenorizadamente a seguinte figura:



*Pela análise da figura atrás apresentada, que representa uma placa - **Arduino Diecimilia**, podemos constatar o seguinte "pin out":*

*- 3 Pinos de **GND** (Ground);*

*- 1 Pino de alimentação de 3.3V (**3V3**) e um de **5V**;*

*- Possui um pino denominado **Vin**, que possibilita o uso da tensão colocada à entrada (Pwr), antes de passar pelo controlador de tensão, sendo esta funcionalidade programável por software.*

*- Um pino de **Reset**, que à semelhança do botão presente no Arduino, e como o próprio nome indica faz o reinício do Arduino, ou seja, executa o programa a partir do inicio novamente, através da aplicação de um sinal de entrada. Voltando a executar o bloco de instruções da função **setup**, como referido mais à frente no artigo.*

*- 14 **Portas digitais** (0 ao 13) configuráveis como input ou output. Com a possibilidade de em seis destas (5,6,9,10,11) usar PWM – Pulse Width Modulation. Esta potencialidade é muito importante, pois através da variação da largura do impulso pode-se "simular" tensões entre 0 e 5V. Os Pinos digitais 0 e 1 possibilitam, ainda, quando configurados, o envio de informação em série. Permitindo outra interface, dependendo do fim pretendido.*

*- 6 **Portas analógicas**, possuindo um conversor analógico digital de 10 bits. E fazendo as contas:*

$$2^{10} = 1024$$

Como a tensão máxima de referência, por definição, se encontra nos 5v correspondendo ao valor 1023, obtemos a seguinte resolução:

$$5 \div 1024 \cong 0{,}00488\,V \cong 5\,mV$$

O que significa que só se conseguirá "detectar" variações superiores a 5 mV. Ou seja, o

valor lido pelo Arduino só se altera a cada 5 mV de variação do sinal analógico de entrada.

Em caso de aplicação de sensores, como por exemplo de sensores de temperatura do tipo termo-pares, que podem ter variadíssimas apresentações e que funcionam na casa dos mV, correspondendo 5 mV em certos casos a subidas de temperatura da ordem dos 80 ºC. Torna-se essencial encontrar uma solução, sem recorrer a electrónica externa.

*Assim, para tal existe uma porta de entrada denominado **AREF**, que significa "**Analog Reference**". Este pino permite mudar a referência analógica do standard 5V para o valor introduzido. Ficando todas as entradas analógicas com a referência introduzida.*

*Simplificando, se se introduzir no pino **AREF** a tensão de 2V obtém-se a seguinte resolução:*

$$2 \div 1024 \cong 1.953\ mV \ \cong 2\ mV$$

É importante ter em conta que todas as portas ficam com esta referência, sendo necessária também a sua configuração por Software.

É igualmente de referir que após configurar o Arduino para o uso do pino AREF, ele deixa de ter disponíveis os pinos de $3.3V$ e $5V$. E que estando estes desligados, será então necessário recorrer a alimentação externa.

*O Arduino possui capacidade de operar alimentado, quer pela porta USB ou por uma entrada **Pwr**. Sendo recomendada a sua utilização entre os 7 e os 12V, que possibilita uma operação do tipo "**Standalone**".*

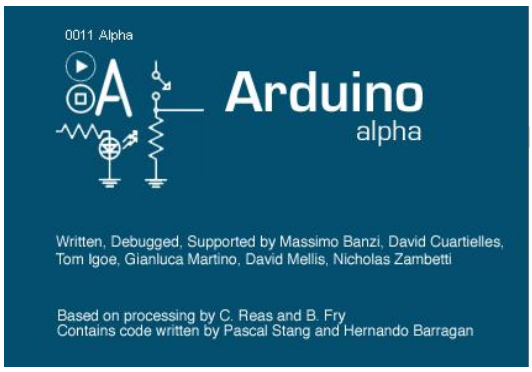
Poderíamos referir aspectos de funcionamento do próprio microprocessador, o que não se torna relevante para o aproveitamento das capacidades do Arduino. O que não significa que não seja objecto de interesse. Devendo o leitor, caso queira, aprofundar os seus conhecimentos nesta matéria fazendo uma pequena pesquisa.

*Havendo para isso muita documentação disponível online, sendo uma paragem obrigatória o site oficial do Arduino - **http://www.arduino.cc**.*

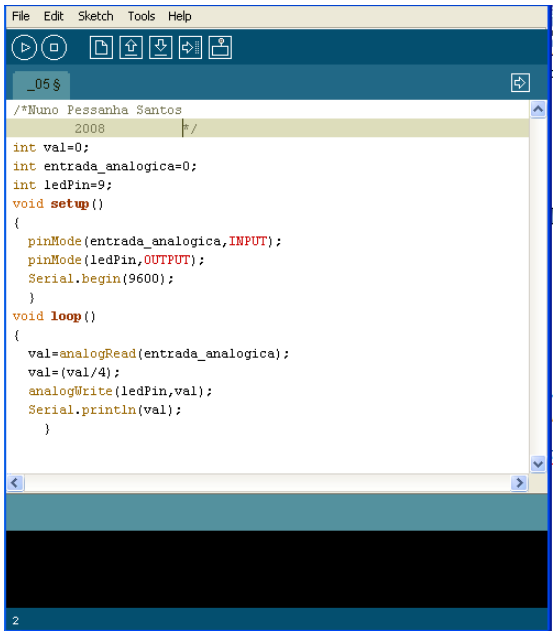
Software

*O ambiente de desenvolvimento pode ser obtido através do site oficial do Arduino, correspondendo a última versão ao **Arduino 0012**.*

*Sendo a sua distribuição completamente livre (sendo "**Open-Source**", como já foi referido anteriormente).*



A linguagem usada nesta aplicação é uma versão simplificada de C, possuindo o mesmo tipo de regras e funções básicas.



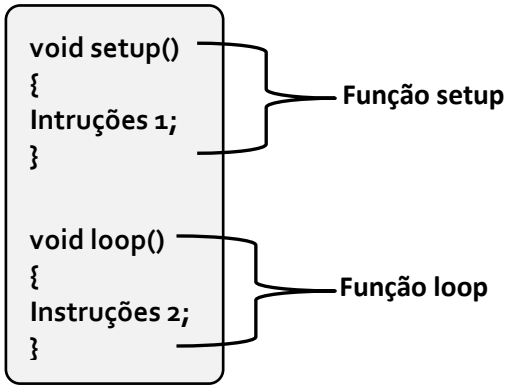
*Na figura acima apresentada, é possível visualizar o ambiente de desenvolvimento do Arduino, muito intuitivo e fácil de usar. Para carregar um programa é simplesmente necessário, elaborá-lo e com um simples click no botão **Upload to I/O Board**, o programa é compilado e enviado para o Arduino.*

*Isto é possível ser feito sem recorrer a Hardware externo, pois o Arduino possui um **Bootloader** de origem. Ferramenta esta que quando ligada, possibilita que o Arduino receba os comandos enviados pela porta USB.*

No entanto, se não chegarem dados, o último programa carregado é executado. E no caso de ser a primeira utilização do Arduino, o único programa em memória será ele mesmo.

1- Constituição do Código

O código é constituído por dois blocos de funções distintas.



A- Função Setup

*A **função Setup** é executada quando o programa começa, sendo este bloco de instruções apenas executado uma vez. É executado quando o Arduino é **ligado** ou quando se efectua o **reset**.*

É usada normalmente como a função responsável por inicializar variáveis, definir os pinos (I/O), definição de bibliotecas entre outros.

Exemplo 1:

```
int buttonPin = 5;                    1
void setup()                          2
{                                     3
  Serial.begin(9600);                 4
  pinMode(buttonPin, INPUT);          5
}                                     6
```

*Dentro do bloco de instruções está a ser configurado o modo de comportamento do pino 5, definido como **buttonPin** (linha 1). No exemplo apresentado, está definido que o pino 5 está configurado como INPUT. Contudo é bastante fácil configurá-lo como OUTPUT, ficando:*

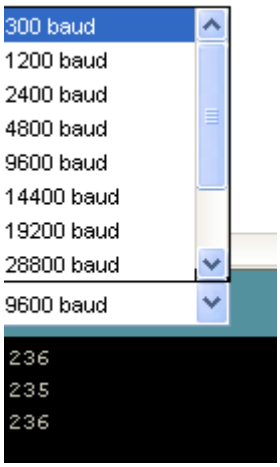
Exemplo 2:

```
  pinMode(buttonPin, OUTPUT);
```

*A função **Serial.begin(int taxa_bps)**, é usada para definir a taxa de transmissão em série. Tipicamente para comunicar com o computador, temos taxas de 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 ou 115200 bps.*

Sendo comum definirem-se valores diferentes dos apresentados, para comunicar com outros dispositivos.

As taxas de transmissão atrás referidas são as suportadas pelo Software padrão do Arduino.



Ao utilizar esta funcionalidade (comunicação pela porta série), que permite ler facilmente valores, p.ex. de sensores, é também possível, através de qualquer Software que faça leituras da porta série, tratar e guardar esses dados, bem como apresentá-los sobre a forma de gráficos ou tabelas, conforme o pretendido.

*Um exemplo de um programa muito comum, que possibilita a leitura pela porta série é o **Matlab.** Este permite o armazenamento e tratamento de dados, como referido anteriormente.*

Exemplo 3:

```
s = serial('COM7', 'BaudRate',19200); 1
fopen(s)                              2
fprintf(s, 'hello arduino!');         3
end                                   4
fclose(s)                             5
```

*Definido o número da porta série, neste caso **COM7**, e o **BaudRate** como 19200 bps, na situação apresentada (linha 1). É de seguida enviada a seguinte mensagem **'hello arduino!'** (linha 3). Terminando com um fecho da ligação **fclose(s)** (linha 5).*

*Sendo o **Matlab** apenas um exemplo de muitos softwares que existem disponíveis e que possibilitam Rx/Tx através da porta série (p.ex. HyperTerminal).*

É ainda possível obter mais informações sobre este tema na página oficial Arduino.

B- Função Loop

*Após a execução da função **Setup,** a função **Loop** é iniciada. E como o próprio nome indica, faz um **loop** sucessivo e permite que o programa faça acções pré-estabelecidas.*

*Um exemplo deste tipo de acções pré-estabelecidas é a leitura de botões ou sensores, apenas possível através desta característica da função **loop**.*

Exemplo 4:

```
void loop()                                  
{                                            
  if (digitalRead(buttonPin) == HIGH)        1
    Serial.println('H');                     2
  else                                       3
    Serial.println('L');                     4
  delay(1000);                               5
}                                            6
```

*O exemplo acima está a fazer a leitura de um **buttonPin** (linha 1), quando este apresenta um valor lógico 1 (HIGH) é enviado pela porta série o valor 'H' (linha 2). Caso essa condição não se verifique é enviado pela porta série o valor 'L'. (linha 4)*

Comportando-se o ciclo if apresentado no exemplo 4, como um ciclo if de programação usado p.ex: na linguagem C.

*Para executar as instruções **Serial.println(data)** na função **Loop**, é necessário efectuar a declaração na função **Setup** da função **Serial.begin(int taxa_bps)**. Definindo o bit rate (bps) desejado para a comunicação.*

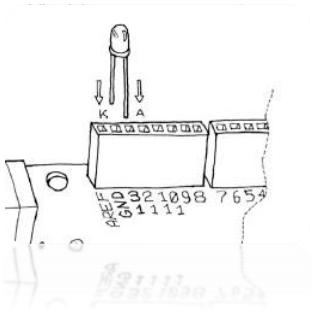
2- Hardware vs Software

Torna-se necessário, à semelhança do 1º programa que se faz em C, o famoso "Hello World", efectuar também um programa inicial. Programa esse que não vai apresentar no ecrã a string "Hello World", fazendo, no seu lugar, um simples led piscar.

O programa para o efectuar é o seguinte:

Exemplo 5:

```
int ledpin = 13;                             1
void setup()                                 2
{                                            3
       pinMode(ledPin, OUTPUT);              4
}                                            5
Void loop()                                  6
{                                            7
       digitalWrite(ledPin, HIGH);           8
       delay(1000);                          9
       digitaWrite(ledPin, LOW);             10
       delay(1000);                          11
}
```



Analisando o exemplo 5:

*Na linha (1), foi efectuado a declaração da variável **ledpin**, sendo esta declarada como sendo do tipo integer.*

As variáveis podem ser do tipo:

- *char*
- *byte*
- *int*
- *unsigned int*
- *long*
- *unsigned long*
- *float*
- *double*

Char

Ocupa um byte de memória a guardar um caracter.

Byte

Bytes guardam um número de 8 bits, vão do 0 ao 255. Não guardam números negativos.

Int

Guardam um valor de 2 byte, inteiro.

Unsigned int

Continua a ter um tamanho de 2 bytes, sendo um int, não contendo parte negativa.

Long

Permite guardar um número inteiro de 4 bytes. (32 bits)

Unsigned long

Semelhante ao long, não possuindo valores negativos. Aumentando assim o intervalo de valores positivo.

Float

Usada para designar números com componente decimal, são guardados em 4 bytes. (32 bits)

Analisando o exemplo 5: (Continuação)

*De seguida é inicializada a função **Setup** (linha 2), que permite definir o pino 13 como OUTPUT (linha 3).*

*Após a declaração das portas como INPUT/OUTPUT, tendo em conta a montagem desejada, inicia-se o ciclo **Loop**, que vai conter a base do nosso programa.*

*No caso apresentado, recorrendo à função **digitalWrite (pino,valor)**, é possível atribuir o valor lógico 1 (HIGH) ou 0 (LOW) a uma saída definida.*

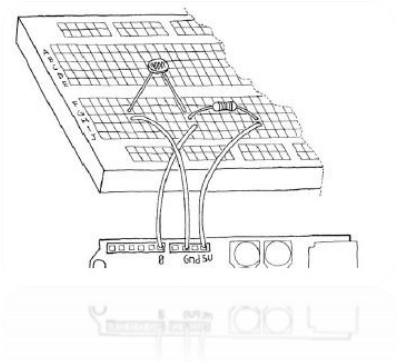
*O programa apresentado contém ainda a função **delay (ms)**, que faz uma pausa no programa pelo tempo definido. Continuando depois com as instruções procedentes à instrução **delay (ms)**.*

O exemplo apresentado permite fazer com que um led pisque de 2 em 2 segundos. Led esse que está ligado ao pino 13 como configurado. O pino 13 possui uma resistência ligada em paralelo, não havendo qualquer implicação da ligação do led directamente a esta porta. O que não acontece nos restantes pinos de saída, sendo este exemplo apenas indicado para o uso da porta 13.

De seguida, vamos ver outro exemplo, que possibilita a leitura de valores analógicos, p. ex. de um LDR, um acelerómetro analógico, entre outros.

Exemplo 6:

```
int val=0;                                  1
void setup()                                2
{                                           3
        pinMode(13, OUTPUT);                4
}                                           5
Void loop(){                                6
        Val = analogRead(0);                7
        digitalwrite(13, HIGH);             8
        delay(val);                         9
        digitaWrite(13,LOW);                10
        delay(val);                         11
}                                           12
```



Analisando o exemplo 6:

*Neste exemplo, vai-se proceder apenas à explicação do bloco de instruções da função **Loop**, pois a função **Setup** é em tudo semelhante à do exemplo 5.*

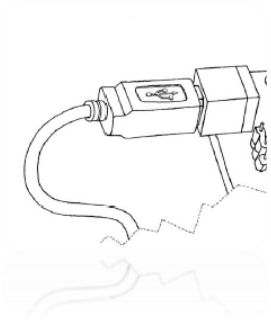
*A função **analogRead(pino)**, lê o valor analógico e guarda-o, neste caso, na variável **val**. Este valor está compreendido entre 0 e 1023, sendo esta a resolução do conversor analógico digital (10 bits). A operação de conversão leva cerca de 0.0001s, pelo que a taxa máxima de leitura é de 10000 amostras por segundo.*

*Através da variação da entrada analógica, é possível fazer variar, no caso apresentado, o **delay (ms)** da montagem.*

*Sendo os exemplos 5 e 6 bastantes acessíveis à compreensão, é imperativo focar um ponto muito importante que possibilita responder à questão: **Como enviar dados através da porta série para o meu Arduino?***

Exemplo 7:

```
int val=0;                                  1
int ledPin=9;                               2
void setup()                                3
{                                           4
  pinMode(ledPin,OUTPUT);                   5
  Serial.begin(9600);                       6
}                                           7
void loop()                                 8
{                                           9
  if (Serial.available() > 0)               10
  {                                         11
    val = Serial.read();                    12
    Serial.print("Eu recebi:");             13
    Serial.println(val,BYTE);               14
    analogWrite(ledPin,val);                15
  }                                         16
```



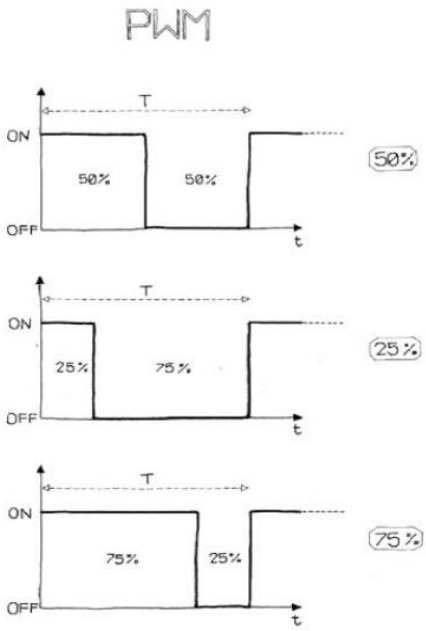
Analisando o exemplo 7:

O ciclo if apresentado permite utilizando a função ***Serial.available()*** *(linha 10), detectar se existem dados a serem enviados para o Arduino acima de um determinado valor. Em caso afirmativo executa o bloco de instruções (linha 11 à 16).*

O valor lido através da função ***Serial.read()*** *(linha 12) é guardado, neste caso, na variável* ***val*** *(linha 12). Sendo depois enviada através da função* ***Serial.println(data)*** *(linha 14) pela porta série e apresentada e/ou guardada por algum programa que possibilite leituras da porta série. Como é o caso do Software* ***Arduino*** *que possibilita essas mesmas leituras, mas não o seu armazenamento.*

Através da função ***AnalogWrite (pino,valor)*** *é possível utilizar a funcionalidade das saídas digitais* ***PWM*** *(Pulse Width Modulation). O parâmetro de entrada,* ***valor*** *da função* ***analogWrite,*** *varia entre 0 e 255, pelo que ao 0 corresponderá 0V e ao valor 255, respectivamente, 5V.*

Por exemplo se o ***valor*** *for 128, o valor de saída será 5V durante metade do tempo e 0V durante a outra metade. Originando um valor eficaz de aproximadamente 2.5V.*



Conclusões

Este artigo tem como objectivo dar a conhecer uma "ferramenta" que como o leitor já deve ter reparado, possui enormes potencialidades.

O trabalho aqui exposto constitui uma ínfima parte do que existe para conhecer e aprender sobre o tema, sendo este documento uma tentativa para despertar o interesse e busca de conhecimento, que tantas vezes faltam nos dias de hoje.

Sendo a melhor solução adquirir um exemplar, caso tenha o leitor manifesto interesse em descobrir mais sobre esta "ferramenta".

Quanto a pormenores das funções utilizadas pelo software oficial ***Arduino****, deve o leitor recorrer ao site oficial Arduino (****http://www.arduino.cc****) ou a tutoriais disponíveis online.*

Pois o saber não ocupa espaço de memória.

Nuno Pessanha Santos
nuno.pessanha.santos@marinha.pt
nuno.pessanha.santos@gmail.com
Escola Naval
Departamento da Classe de Engenheiros Navais
Ramo de Armas e Electrónica