Apostila de Arduino

Gabriel Vasiljević

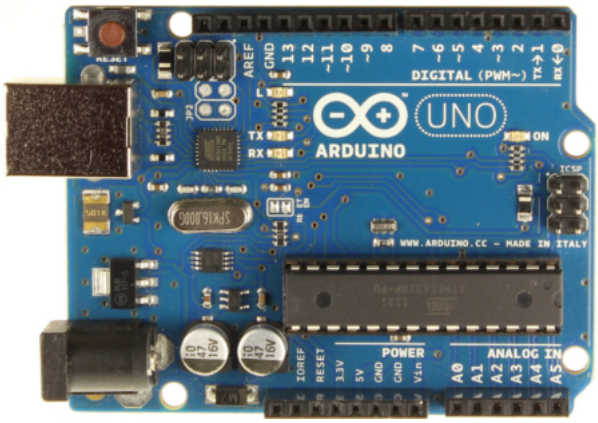
February 1, 2013

Contents

1 Introdução

1.1 O que é Arduino?

Arduino é uma plataforma eletrônica de prototipação, criada na Itália, constituído basicamente de uma placa microcontroladora, uma linguagem de programação típica com um ambiente de desenvolvimento e suporte a entrada e saída de dados e sinais. Foi criada em 2005 com o objetivo de servir como base para projetos de baixo custo, sendo simples o suficiente para ser usado por desenvolvedores amadores. É bastante flexível e não requer um domínio profundo de eletrônica, o que o fez ser bastante popular entre os artistas e iniciantes, além de desenvolvedores experientes que nao têm acesso a plataformas mais complexas.



Arduino Uno

1.2 Por que usar Arduino?

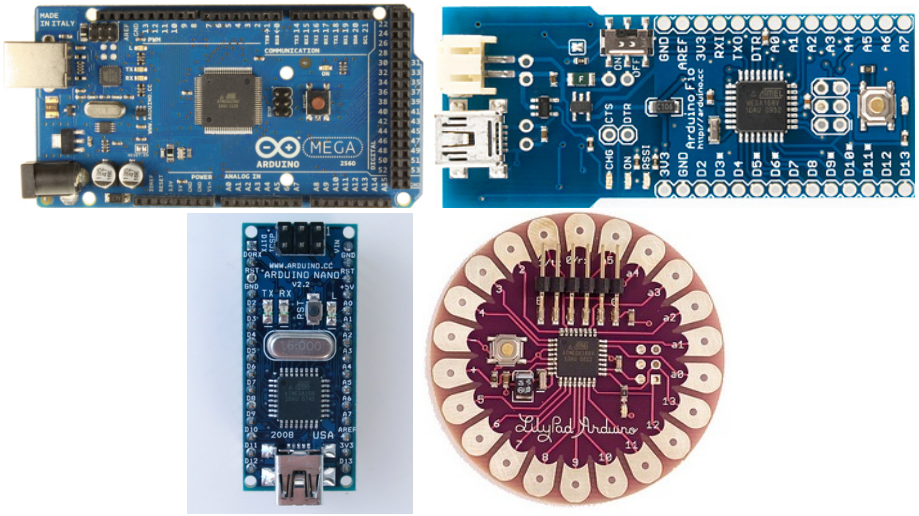
O Arduino foi criado com o propósito de ser uma plataforma extremamente fácil de usar se comparado às outras, o que o torna ideal tanto para desenvolvedores iniciantes quanto para os mais experientes, que farão seus projetos muito mais rapidamente, de forma menos trabalhosa. Outro fator que torna o Arduino atrativo é sua filosofia de *hardware livre*, ou seja, as pessoas podem usá-lo para criar diversos projetos sem custo algum de direitos pela utilização da plataforma, podendo ser distribuído gratuitamente, se elas desejarem. Isto traz diversos benefícios; além de serem criadas e distribuídas diversas novas bibliotecas e ferramentas para auxiliar o desenvolvimento de projetos todos os dias, conta-se com uma comunidade de milhares de pessoas que divulgam informações e detalhes sobre o que criam, fazendo com que nunca falte ajuda ou algum conhecimento necessário para concluir o que se desejas construir. Esses são também alguns dos motivos pelo qual a popularidade do Arduino vem crescendo atualmente entre os desenvolvedores.

1.3 O que podemos fazer com um Arduino?

Praticamente qualquer coisa! tendo os equipamentos necessários, é possível criar projetos que só são limitados pela sua imaginação (e às leis da física, é claro!). Alguns exemplos abaixo mostram um pouco do que podemos fazer:

1.4 Só existe um tipo de Arduino?

Não! existem diversos modelos de Arduino para se utilizar dependendo do que se deseja fazer, com diferentes formatos e configurações de hardware. O Arduino Uno é um dos mais utilizados, mas o Arduino Mega, por exemplo, possui muito mais portas de entrada, possibilitanto a criação de dispositivos maiores e mais complexos. O Arduino Nano, como o nome já diz, é uma versão reduzida de um Arduino comum, para a criação de objetos eletrônicos menores (ou ocupação de menos espaço, dependendo do projeto). Seguem abaixo algumas imagens com alguns dos vários tipos de Arduino existentes hoje em dia.



Respectivamente, os modelos são Arduino Mega, Arduino Fio, Arduino Nano e LilyPad Arduino. Cada um possui uma funcionalidade diferente que justifica sua criação. O LilyPad, por exemplo, foi criado para poder ser utilizado em vestimentas, podendo ser costurado diretamente sobre tecidos. Para mais informações sobre os diversos modelos do Arduino, basta consultar o site oficial: www.arduino.cc.

2 Conceitos Básicos e noções de eletricidade

Antes de começar a desenvolver qualquer dispositivo eletrônico, é necessário conhecer tanto como funcionam seus componentes quanto porque eles funcionam. A eletricidade é o principal elemento e é importantíssimo conhecer pelo menos seus mais básicos conceitos, para evitar prejuízos com peças perdidas e acidentes desnecessários. Primeiro, devemos saber o que *é* a eletricidade. Vamos começar analisando a nível atômico, que é onde tudo ocorre.

O átomo é constituído principalmente de um núcleo, onde estão localizados os prótons e os neutrons, e uma eletrosfera que o envolve, onde estão os elétrons em órbita. Por convenção, os prótons possuem carga elétrica positiva, enquanto os elétrons possuem carga

nevativa, e o neutron possui carga elétrica nula.
Por vezes, um átomo pode perder ou ganhar um elétron a mais, fazendo com que ele tenha uma carga resultante positiva ou negativa. Elétrons soltos se movimentam de forma aleatória, mas, quando submetidos a um campo magnético em movimento ou a uma DDP, passam a se mover de forma ordenada, gerando uma **corrente elétrica**. Uma DDP (Diferença de Potencial, ou Tensão) acontece quando dois pontos possuem potencial elétrico diferentes, fazendo os elétrons irem do ponto de maior potencial para o de menor potencial. Isto acontece porque tudo na natureza tende a estar em seu estado natural e, no ponto de maior potencial, os elétrons não o estão. Podemos fazer uma analogia com o potencial elástico: quando esticamos um elástico, estamos tirando-o de seu estado natural (parado), dando a ele energia potencial elástica. Quando o soltamos, ele rapidamente se contrai novamente, voltando ao seu estado natural. O elástico saiu do ponto de maior potencial para o de menor potencial - o mesmo que acontece com os elétrons, com a diferença que eles possuem energia potencial elétrica. A unidade de tensão elétrica e o Volt (V). Um exemplo prático são as tomadas de nossa casa, que são ligadas a dois cabos de tensões diferentes, que passam pelos postes: um possui 110 ou 220 V (dependendo da região do país) chamado de Fase, enquanto o outro é o Neutro, que possui 0 V. Cada encaixe das tomadas representa um desses fios (há um terceiro encaixe, o fio-terra, que liga o circuito à terra para descarregar o excesso de cargas acumuladas). Enquanto não conectados, não há corrente entre eles - é preciso que o circuito seja **fechado** para que a diferença de potencial ocorra. Quando conectamos algum aparelho elétrico na tomada, o circuito é fechado e ocorre a diferença de potencial entre o fio Fase e o Neutro, criando uma corrente elétrica que passará pelo equipamento alimentando-o e sairá pelo fio neutro. Por motivos históricos, convencionamos que a corrente elétrica ocorre em sentido contrário ao percurso dos elétros - ou seja, se os elétrons fluem para um lado, a corrente elétrica ocorre para o outro lado. Fica como exercício achar o sentido disso (mas ele existe!)

3 Componentes eletrônicos

3.1 Fios

Muitas vezes, é inviável conectar diretamente um componente ou um dispositivo diretamente aos terminais de um gerador ou de uma fonte de energia qualquer. Os fios permitem que possamos conectar um dispositivo elétrico à uma fonte de energia à grandes distâncias e está longe de ser substituído por algo mais prático, já que ainda não descobrimos uma forma de transmitir energia de outra maneira (embora Nikola Tesla já tenha feito pesquisas em transmissão de energia sem fio por volta do início do século XX, pouco se sabe a respeito de seus resultados). Em projetos com o Arduino os fios (ou *Jumpers*, como são chamados, pelo fato de fazerem a corrente elétrica "pular" de um local ao outro) são essenciais, principalmente pelo fato de podermos economizar diversas saídas/entradas da placa com um único fio. Eles são normalmente constituídos de um metal condutor encapuzado por um isolante, de modo a não causar choques. Quanto mais grosso o isolante, mais tensão tem o fio - e mais perigoso ele é. O fio de um carregador de celular, por exemplo, é relativamente fino, enquanto o cabo de alimentação do computador é bem mais grosso.

3.2 Resistores

Os resistores, como o próprio nome diz, são componentes que resistem à corrente elétrica. Todo material condutor apresentar certa resistência à passagem da corrente, mesmo que isso não seja o ideal (não existe condutor perfeito). Normalmente, queremos controlar a passem da corrente apenas em algumas partes do circuitos, para não danificar certos componentes ou para transformação de energia elétrica em térmica. Este último é chamado de **Efeito Joule**, e acontece porque os elétrons se chocam com os átomos do material na qual estão passando. Aplicações do efeito Joule são inúmeras e bastante comuns em nosso dia-a-dia. Por exemplo, o chuveiro elétrico funciona basicamente com um resistor ajustável (chamado de *potenciômetro* ou *reostato*), em que podemos ajustar a posição do contato para controlarmos quanta corrente passará por ele. Quanto maior a distância que a corrente percorre no resistor, mais calor é gerado (e mais energia é gasta, por isso o consumo é tão alto). Portanto, ao ajustarmos a temperatura para a posição Inverno, estamos aumentando o percurso da corrente pelo resistor, aquecendo mais a água. Outro exemplo é a lâmpada incandescente, que possui um filamento que incandece ao passar a corrente elétrica, devido ao efeito Joule. Este tipo de lâmpada é bastante ineficaz, pois mais de 95% da energia elétrica é convertida em calor, sendo somente o restante convertida em energia luminosa. A unidade de medida de resistência elétrica é o Ohm, representado pela letra grega Ω (omega maiúsculo), e pode ser medido como a divisão da tensão pela corrente elétrica.

Código de cores

Como os resistores utilizados nas placas impressas de circuitos são muito pequenos, a impressão do valor de sua resistência em ohms numericamente é bastante complicada. Por causa disso, foi convencionado um código de cores que facilitou esse reconhecimento. O padrão e, de certa forma, bastante simples: existem quatro faixas de cores ao redor do resistor. as três primeiras faixas sao as A, B e C, e a quarta é a tolerância (quantos porcentos, para mais ou para menos, aquela resistência pode estar - não existe resistor 100% preciso). A grosso modo, a faixa A representa a primeira casa decimal, a faixa B a segunda casa decimal e a faixa C a ordem de grandeza. As cores vão do branco ao preto na segunte ordem, do menor para o maior: preto, marrom, vermelho, laranja,amarelo, verde, azul, violeta, cinza e branco. Na terceira faixa, o cinza e o branco são substituídos por prata e dourado. A quarta faixa só possui três cores: prata, dourado e marrom.

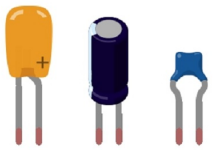
3.3 Potenciômetros

Como foi anteriormente mencionado, potenciômetros, ou reostatos, são resistores ajustáveis. Eles permitem que ajustemos a intensidade da corrente que passará por determinada parte

do circuito. Eles sao bastante usados, por exemplo, em caixas de som: se você abrir uma, verá que o ajustador de volume dela é um potenciômetro, que somente ajusta o quão forte aquele som vai sair. Ao aumentarmos o volume da caixa, estamos diminuindo a resistência e permitindo uma maior intensidade da corrente.

Normalmente, o potenciômetro possui três conectores, dois laterais e um central. um dos conectores laterais é por onde a energia entrará, e o outro é conectado ao terra. O central é por onde a corrente sairá, depois de tê-lo percorrido.

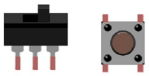
3.4 Capacitores



Capacitores são componentes capazes de armazenar energia elétrica. São como pequenas pilhas ou baterias, mas possuem capacidade mais limitada, dependendo de seu tamanho e de sua tensão. Normalmente são formados por duas placas condutoras separadas por uma camada isolante (dielétrica), de modo que não tenham contato uma com a outra. Quando acumulam cargas o suficiente, a rigidez dielétrica do isolante é rompida (ou seja, ele passa a conduzir corrente), e toda a energia é liberada quase que instantaneamente. É o que acontece com os relâmpagos: A núvem e o solo são as duas placas, e o ar é o dielétrico. Quando a rigidez dielétrica do ar é rompida, toda a energia é liberada de uma vez, e forma-se o relâmpago.

Uma aplicação bastante famosa dos capacitores é o flash das máquinas fotográficas. Durante alguns segundos, a bateria da máquina carrega o capacitor, que libera toda a energia de uma vez para a lâmpada do flash.

3.5 Chaves

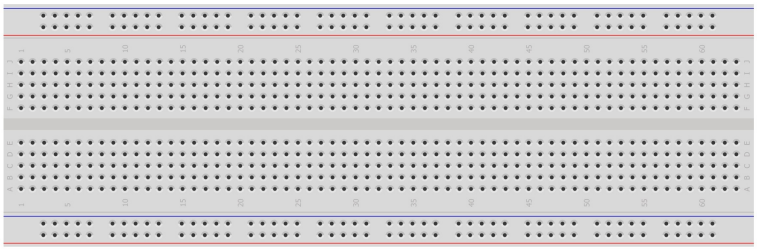


As chaves, ou interruptores, servem para abrir ou fechar um circuito, permitindo ou não que haja diferença de potencial e, conseqüentemente, corrente elétrica. Daí seu nome: eles interrompem a corrente. Podemos utilizar um único interruptor para controlar a corrente em várias partes do circuito ao mesmo tempo. Os insterruptores das lâmpadas de nossa casa são um ótimo exemplo. Podemos ligar várias lâmpadas ao mesmo tempo ou somente uma. Eles podem funcionar de diversas formas; por exemplo, ao apertar o botão, ele libera a passagem da corrente e, ao soltá-lo, ele a impede novamente, enquanto outros podem manter continuamente a corrente passando até ser pressionado novamente.

3.6 LEDs

LEDs (Light Emitting Diode) são, como o próprio nome diz, diodos emissores de luz. Um diodo é um semicondutor; isso significa que ele tem a capacidade de conduzir corrente elétrica por um de seus terminais e bloqueá-la pelo outro. Diodos são bastante usados em circuitos em geral por esta capacidade, e os LEDs têm destaque por suas inúmeras aplicações. Eles servem de base tanto para relógios digitais quanto para televisores e controles remotos (no caso do LED infravermelho). Seu fincionamento é relativamente simples: um material com cargas negativas extras, chamado de material tipo-N, é colocado separado por uma pequena distância, chamada de zona vazia, de um material com carpas positivas extras, chamado de material tipo-P. Ao conectarmos o eletrodo do material tipo-P ao terminal positivo do gerador e fizermos o mesmo com os lados negativos, as cargas se repelirão, fazendo a corrente fluir. Ao fazer isso, os elétrons liberam energia em forma de luz (fótons), que é a luz que vemos quando os ligamos. A principal vantagem dos LEDs é que gastam bem menos energia e são tão eficientes quanto as lâmpadas incandescentes, o que pode ser verificado pela quantidade de calor que eles geram pelo efeito Joule: como esquentam muito pouco, pouca energia é desperdiçada em energia térmica, ao contrário das incandescentes.

3.7 Protoboard



Para conectar todos estes componentes de forma a termos um circuito funcional, teríamos que unir suas entradas e saída de alguma forma, de modo que a corrente que passe por uma peça continue se propagando à próxima. Geralmente, fazemos isso através de um processo de **solda**; ou seja, aquecemos um metal maleável e condutor de forma que ele derreta, para unir as duas partes usando-o como uma espécie de cola. O ato de soldar é uma prática delicada e requer habilidade, tanto para obter resultados satisfatórios quanto para evitar uma queimadura que geralmente pode ser fatal (uma maquina de solda aquecida derrubada por acidente pode causar queimaduras de 3º grau). Além do óbvio perigo, o processo de solda é de certo modo "permanente", pois as peças ficarão unidas até que derreta-se o metal novamente e utilize-se um sugador para retirá-lo, o que pode ser bastante trabalhoso se houver grande quantidade de peças soldadas. Para evitar todos esses problemas, existe a **protoboard**, uma placa com furos dispostos em formato de grade, com conexões condutoras em seu interior. A protoboard faz com que a conexão entre os componentes seja extremamente

simples, sem necessidade de solda. Obviamente ela não é usada em aplicações industriais, já que o desacople dos fios é muito fácil de ocorrer (motivo pelo qual ela também é chamada de *placa de ensaio*). Uma protoboard comum possui dois tipos de conexão: verticais, presentes nos furos laterais, e horizontais, presente nos restantes. A corrente percorre todos os furos da mesma linha, alimentando todos os fios ou componentes que ali estiverem conectados.

3.8 Sensores

Os sensores são componentes usados para ler e interpretar variáveis do ambiente. Intensidade de luz, som, objetos, temperatura etc., podem ser medidos e traduzidos como uma determinada voltagem. Alguns dos sensores mais comuns são:

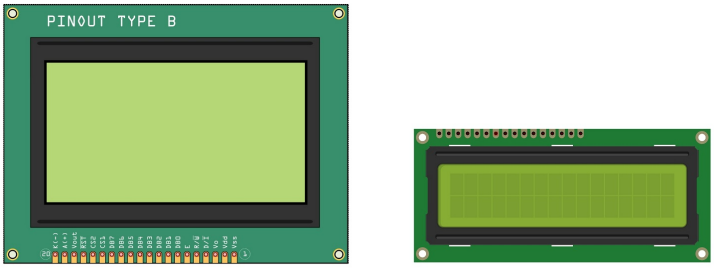
3.8.1 Sensor de luz

Como o próprio nome diz, sensores de luz detectam uma certa intensidade de luz no ambiente. Os mais comuns são os LDR (Light Dependent Resistor - resistor dependente de luz, no inglês), que são resistores que se ajustam de acordo com a intensidade de luz - ou seja, deixam passar mais ou menos corrente, e é esse sinal que é detectado e interpretado.

3.8.2 Sensor de temperatura

Sensores de temperatura funcionam como termômetros, captando a temperatura do ambiente. Mas, ao invés de exibir esta informação visualmente, ele a reproduz em forma de uma tensão de saída. O mais famoso e comum é o LM35, pois é barato e possue uma boa precisão. Neste sensor, por exemplo, para cada grau Celsius a tensão de saída aumenta em 10mV. Sua faixa de precisão varia de -55ºC a 150ºC, com precisão de ± 0,5ºC.

3.9 Displays



Os displays são interfaces gráficas que utilizamos para representar informações visualmente. Podemos, por exemplo, exibir uma mensagem, mostrar a temperatura do ambiente, etc. Existem os famosos displays de 7 segmentos (ou 11 segmentos), que são usados para representar dígitos decimais, e os displays mais complexos, capazes de exibir cadeias de caracteres e até figuras. O Arduino possui vários displays com bibliotecas prontas que basta encaixar e usar, bastate práticos. Os acima são, respectivamente, displays de 128x64 e 16x2.

4 Desenvolvimento com Arduino

Agora que temos uma boa noção de como funcionam alguns dos diversos componentes eletrônicos que podemos utilizar e sabemos como o fazer, está na hora de começarmos (finalmente) a desenvolver com o Arduino. Nesta sessão explica-se como instalar e utilizar o software necessário, bem como detalhes para a montagem de alguns exemplos de projetos iniciais, que gradativamente vão se tornando mais complexos a medida em que adicionamos mais funcionalidades ao dispositivo.

4.1 Primeiros passos

Antes de mais nada, precisamos obter os arquivos necessários para podermos enviar os comandos ao Arduino. Para isso, devemos fazer o download da plataforma no site oficial (www.arduino.cc) dependendo do sistema operacional.

4.1.1 Instalação da IDE no Windows

Após baixar os arquivos necessários do Windows pelo site, extraia-os para o local desejado e execute o programa do Arduino, localizado na raiz da pasta principal. Em seguida:

• Conecte o Arduino ao computador usando o cabo USB. A luz com nome PWR (Power) deve acender, indicando que a placa está ligada.

• Deve-se ser solicitado que um novo driver seja instalado. Para isso, recuse que o Windows procure os drivers automaticamente na internet e, na seleção manual, peça para que sejam procurados os drivers na pasta do Arduino que tu extraistes no primeiro passo.

• Caso não seja acionada a detecção automática de driver, vá ao Painel de Controle, abra o Gerenciador de Dispositivos e procure por drivers desatualizados. Selecione a atualização manual do driver e indique a pasta mencionada no passo anterior.

• Após isso, a instalação deve estar concluída.

4.1.2 Instalação da IDE no Linux

Para o Linux, abra o terminal e execute os seguintes comandos:

• sudo add-apt-repository ppa:arduino-ubuntu-team/ppa

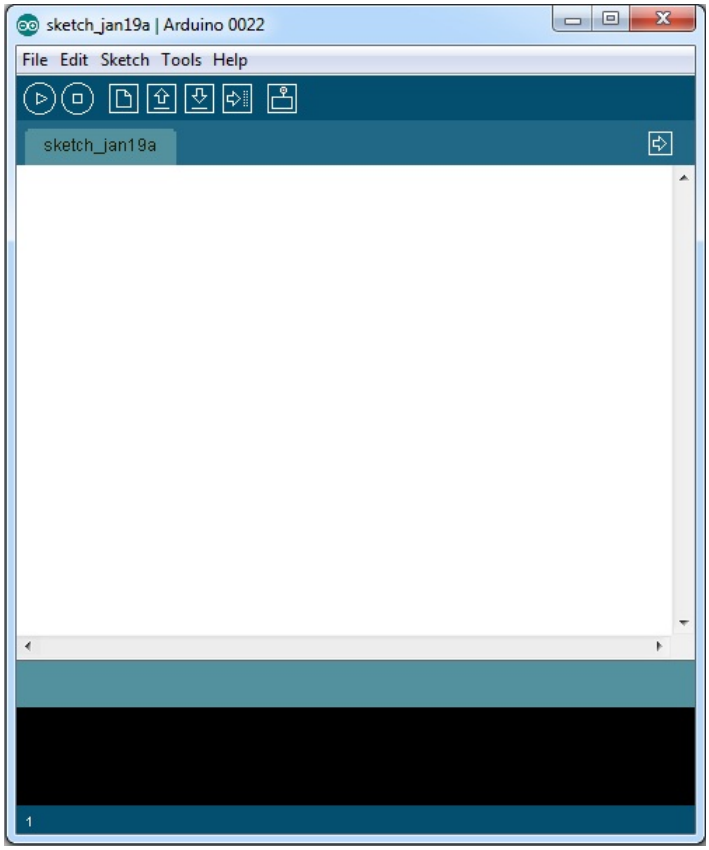
• sudo apti-tude update

• sudo aptitude install arduino

Em seguida, já pode-se acessar a IDE do Arduino através dos aplicativos de desenvolvimento.

4.2 IDE

O IDE do Arduino é bastante simples. Foi projetado para ser uma interface amigável à pessoas que nunca tiveram contato com desenvolvimento de software, e portanto é bastante intuitiva. Foi desenvolvido em Java e possui recursos simples de realce de palavras-chave e uma base com diversos códigos prontos para servir como exemplo.



IDE do Arduino

Além do espaço para a escrita do código, existem 7 botões na parte superior: Verify, Stop, New, Open, Save, Upload e Serial Monitor. Eles servem respectivamente para verificar se o programa possui erros, parar a execução do código atual, criar um novo documento, abrir um documento já existente, enviar os dados para o Arduino e o ultimo é um monitor para dados seriais, que será explicado mais a frente. Na parte inferior, também existe uma janela de console para informar ao usuário mensagens de erro ou de execução do programa.

4.3 Linguagem, comandos e funções

A linguagem de programação utilizada pela plataforma Arduino é basicamente C e C++. Praticamente todos os comandos utilizados em C e C++ podem ser utilizados para configurar o comportamento de nossos circuitos, o que facilita (e muito) nosso trabalho, mesmo para quem não tem o conhecimento de tais linguagens, por serem bastantes simples e intuitivas.

4.3.1 Principais funções

O fluxo de execução de qualquer programa do Arduino começa na função **setup()**, em que definimos quais pinos iremos usar e se eles serão de entrada ou de saída. Para isso, usamos

a função **pinMode(pino, modo)**, onde *pino* é um inteiro representando o número do pino que estamos configurando e *modo* é o tipo de pino que ele será, no caso entrada ou saída - Input ou Output, respectivamente. Por exemplo, *pinMode(13, OUTPUT)* indica que o pino 13 será usada como uma saída; ou seja, ele fornecerá sinal ao circuito.

Após a execução da função *setup()*, o programa passa a executar a função *loop()*, e continuirá a repetí-la até a execução do programa terminar. Destarte, ela é uma das funções mais importantes que iremos utilizar na construção de projetos, pois é nela que iremos descrever ttudo que o circuito irá fazer e como ele vai se comportar.

4.4 Tipos de portas

No Arduino existem dois tipos de portas: digitais, divididas entre binárias comuns e PMW, e analógicas. Cada tipo de porta é designada para objetivos específicos.

4.4.1 Portas Digitais

As portas digitais são utilizadas para trabalhar com valores binários de tensão: 0V e 5V. Destarte, componentes conectados a essas portas só poderão mandar e receber dados na forma destas duas tensões. No Arduino Uno, existem 14 portas digitais, numeradas de 0 a 13, sendo que 5 delas são PWM (que será explicado mais adiante), e as 0 e 1 são para os LEDs imbutidos RX e TX, respectivamente.

As principais funções do Arduino para manipular as portas digitais são:

- *digitalRead(pino)*
  Lê um valor da porta *pino* e retorna o valor HIGH quanto está em 5V e LOW quando está em 0V.

- *digitalWrite(pino, estado)*
  Utilizada para dizermos ao Arduino que queremos que *pino* esteja no *estado* ligado (HIGH) ou desligado (LOW).

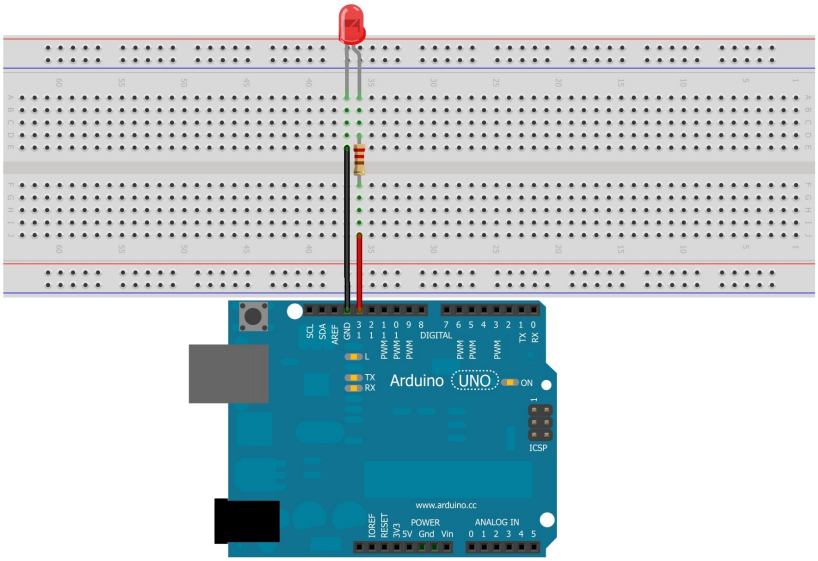
Para entender melhor estes conceitos, vamos usar um exemplo simples para demonstrar o uso destas portas.

4.4.2 Primeiro exemplo

Para o primeiro exemplo de um circuito simples usando Arduino, vamos montar um LED que pisca a cada segundo. Para isso, utilizaremos as funções e comandos vistos anteriormente. Primeiro, no IDE do Arduino, criamos a função setup. Nela, usamos o procedimento pinMode para dizer que o pino 13 será utilizado como uma saída. Isto, por enquanto, é tudo que precisamos, já que só vamos precisar de uma saída para fornecer energia para o LED. Em seguida, devemos criar a função loop, que é onde ficarão todos os comandos que vão ser executados repetidamente e indefinidamente pelo Arduino, como já foi dito. Usamos o procedimento *DigitalWrite(pino, estado)* para dizer ao Arduino que se pino 13 estará ligado ou desligado, e o procedimento *delay(time)* para dizê-lo para esperar um certo período de tempo, em milisegundos. Ao final, o código deverá estar parecido com isso:

```
void setup() {
  pinMode(13, OUTPUT);     //Diz que o pino 13 será uma saída
}

void loop() {
  digitalWrite(13, HIGH);   // Diz que o pino 13 está ligado (led acende)
  delay(1000);              // Espera 1000ms (1 segundo)
  digitalWrite(13, LOW);    // Diz que o pino 13 está desligado (led desliga)
  delay(1000);              // Espera mais um segundo
}
```

Em seguida, devemos montar o circuito. Vamos precisar do LED, da protoboard, de dois fios e de um resistor de 120Ω. O resistor restringirá a corrente de forma a não danificar o LED, portanto é importante não se esquecer dele! Devemos ligar o ânodo do LED (a "perna" maior) ao cátodo do gerador (no caso, ao pino 13) e o cátodo ao terra (GRD - ground), de forma a ter uma DDP e, assim, permitir a passagem da corrente elétrica. O resistor deve ser ligado entre o ânodo e o gerador. Por convenção, utilizaremos os fios pretos para conexões ao terra, vermelho para as portas normais e azuis para as portas de voltagem (3.3V e 5V). O esquema do circuito montado e mostrado a seguir:



Agora, basta fazer o upload do código ao Arduino e vê-lo em ação!

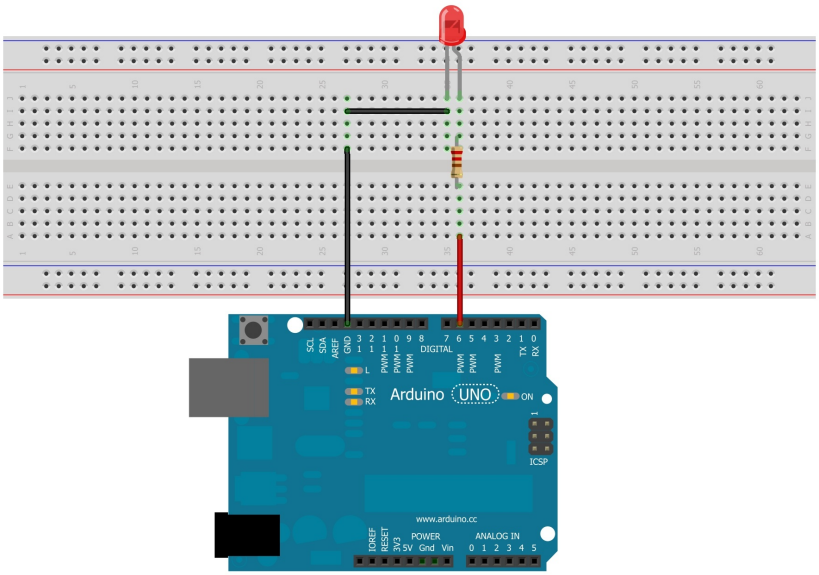
4.4.3 PWM

Podemos perceber, pelo exemplo anterior, que utilizando as portas digitais binárias só podemos fornecer dois estados ao nosso LED: totalmente ligado ou totalmente apagado. Isso acontece porque as portas mandam um sinal constante de 5V, o que faz o LED acender inteiramente. Podemos, porém, controlar o percentual de tempo que o LED fica ligado, utilizando as portas digitais PWM (Pulse Width Modulation - Modulação por Largura de Pulso, em inglês). O que essas portas têm de especial é que elas são capazes de controlar a potência de um sinal, fazendo-o oscilar entre 0V e 5V, em uma determinada freqüência: na prática, isso significa que a porta vai repetidamente ser ligada e desligada.

Mas qual a utilizade disso? Bom, primeiro, podemos economizar energia limitando a potência pelo qual o circuito está sendo alimentado. Podemos não querer, por exemplo, que o LED gaste 5V todo o tempo em que ele estiver ligado, ou uma iluminação mais suave. Para isso, basta diminuir este percentual de tempo em que a porta deixa o sinal no máximo. Para controlar esse percentual, utilizamos a função analogWrite(pino, valor), onde pino deve ser uma porta PWM e valor e um inteiro entre 0 (0% do tempo) e 255 (100% do tempo). Note que estamos utilizando uma função para manipulação de portas analógicas, mesmo as PWM sendo portas digitais. A razão por trás disso é que as portas PWN simulam o comportamento das portas analógicas, que serão explicadas em seguida.

Para exemplificar, vamos fazer um LED acender e apagar suavemente. Para isso, basta apenas trocar a porta de saída do nosso LED por uma porta PWD. No Arduino Uno, essas portas estão marcadas por um acento til (∼). No nosso exemplo, utilizaremos a porta 6.

A montagem do circuito é a mesma do primeiro exemplo, mudando somente a porta de saída:



No código, utilizamos um *for* para irmos do brilho mínimo (i = 0) para o brilho máximo (i = 255), aumentando o brilho de 5 em 5 (i += 5). Entao, usamos a função analogWrite para mandar a tensão ao pino 6, e esperamos 30 milisegundos (sem o delay, o brilho aumenta rápido demais e não conseguimos ver!).

```
void setup(){
  pinMode(6, OUTPUT);
}

void loop(){

  for(int i = 0; i <= 255; i += 5) {
    analogWrite(6, i);
    delay(30);
  }

}
```

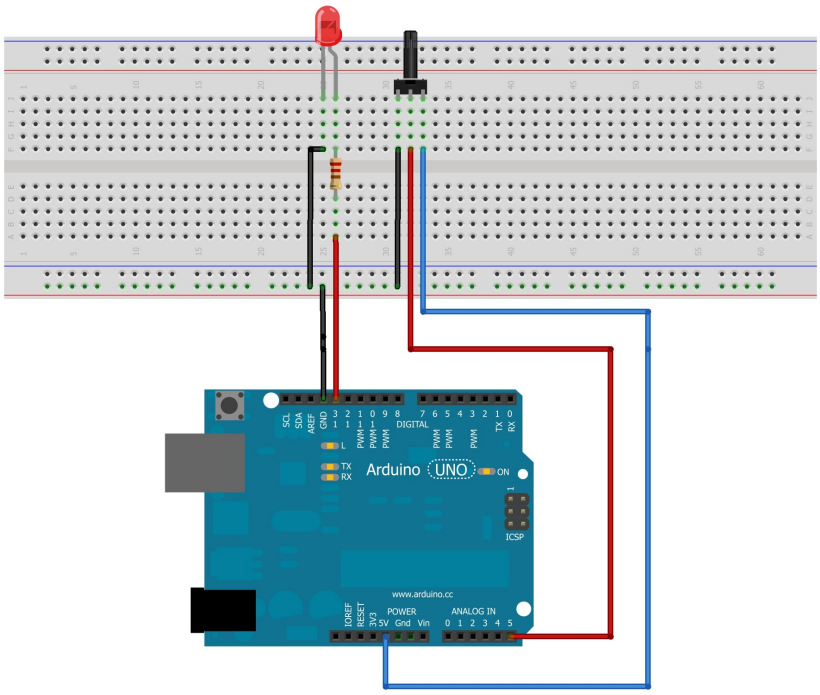
4.4.4 Portas Analógicas

Diferentemente das portas digitais, que só podem ler e enviar valores binários, as portas analógicas podem ler qualquer valor, com determinada precisão, entre 0V e 5V. No entanto, elas podem somente ler, e não enviar. Existem seis portas analógicas no Arduino Uno, numeradas de 0 a 5, e possuem precisão de 10 bits. Isso significa que elas podem ler em uma faixa de valores entre que vai de 0 a 1023 (2^{10} valores), e é com esses valores que iremos trabalhar. Para ilustrar uma situação prática, vamos criar um circuito onde um LED pisca de acordo com a leitura de um potenciômetro - ou seja, quanto mais corrente ele deixar passar, mais rápido o LED piscará.

Para isso, conecte os três pinos do potenciômetro da seguinte maneira:

- O conector central ligado à uma das entradas analógicas. No nosso caso, utilizaremos a A5 (lembrando que as entradas analógicas são nomeadas de A0 a A5);
- Um dos conectores laterias (não importa qual) conectado ao terra;
- O outro conector ligado à saída 5V do Arduino.

Em seguida, conecte o LED à porta 13, como anteriormente. Lembrando que o próprio Arduino possui um LED imbutido conectado ao pino 13, então conectar outro e opcional. O esquema deve ficar parecido com este:



Agora, criemos o código. Nele, precisamos agora declarar duas portas: a do LED e a do sensor. Como as portas analógicas só servem como entrada, não precisamos inicializá-las na função setup. Lemos o valor do potenciômetro com a função analogRead (que retorna

um inteiro entre 0 e 1023) e o usamos este valor como o atraso para acender o LED, como mostra o código:

```
const int sensor = A5;    // Entrada para o potenciômetro
const int LED = 13;       // Saída para o LED
int leitura = 0;          // Guarda o valor da leitura feita pelo potenciômetro

void setup() {
  pinMode(LED, OUTPUT);   // Declara o pino do LED como saída
}

void loop() {
  leitura = analogRead(sensor);    // Lê o valor do sensor
  digitalWrite(LED, HIGH);         // Acende o LED
  delay(leitura);                  // Dá uma pausa de acordo com a leitura

  digitalWrite(LED, LOW);          // Apaga o LED
  delay(leitura);                  // Dá outra pausa de acordo com a leitura
}
```

4.5 Comunicação Serial

Até agora, vimos como configurar o Arduino para algumas funções usando sensores e atuadores diversos. Mas, e se quisermos, por exemplo, controlar algum componente do Arduino atraves do computador? É para isso que servem as portas de comunicação serial. Utilizando a comunicação USB (Universal **Serial** Bus) que usamos normalmente para fazer o upload de dados do Arduino, podemos enviar sinais e comandos para alterar o seu estado.

Antes de mais nada, precisamos iniciar a comunicação informando a velocidade da transferência de dados. Depois, poderemos enviar e receber dados do Arduino e visualizá-los atraves do monitor serial: [imagem]

As principais funções que iremos utilizar são:

• **Serial.begin(velocidade)**
Esta função diz ao Arduino que iremos iniciar a interface serial, utilizando o parâmetro *velocidade* como taxa de transferência. Por padrão, normalmente utiliza-se 9600 como taxa.

• **Serial.print("Mensagem")**
Exibe uma mensagem no monitor serial. Podemos informar qual a leitura que um sensor está recebendo do ambiente, por exemplo, ou apenas informar alguma mensagem qualquer.

• **Serial.available()**
Retorna o número de bytes sendo lidos pela porta serial. No caso de não haver nenhuma valor, retorna zero. Utilizamos esta função para o Arduino saber quando enviamos ou recebemos algum dado - um botão que pressionamos, por exemplo.

• **Serial.read()**
Esta função lê os dados que digitamos pelo teclado e os envia ao Arduino. É nosso

principal meio de comunicação!

Vamos para um exemplo prático. Neste exemplo, vamos apagar e acender um LED usando um comando no teclado. Para isso, verificamos se há alguma informação a ser lida - ou seja, se algum comando foi enviado do computador ao Arduino. Se sim, guardamos esta informação e a processamos de acordo com o que queremos. Em nosso caso, vamos acender o led no pino 13 caso enviarmos o caractere 'L' ao Arduino.

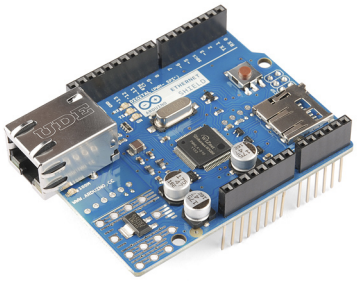
```
const int LED = 13;          // Uma boa prática é sempre definir cada pino!

void setup(){
  pinMode(LED, OUTPUT);           // seta o pino 13 como saída
  Serial.begin(9600);             // inicia a interface serial com 9600 bps
}

void loop(){
  if( Serial.available() ){         //Se há alguma informação para ser lida...
    int liga = Serial.read();       // Lê a informação como um inteiro

    if(liga == 'L')                 // Se o que foi lido foi um 'L'
      digitalWrite(LED, HIGH);     // Acende o LED

    else
      digitalWrite(LED, LOW);     // Senão, desliga-o
  }
}
```

O esquema do circuito é o mesmo do primeiro exemplo.

4.6 Shields

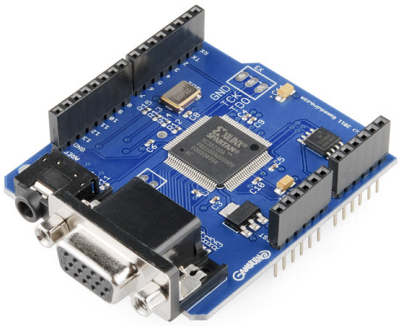
Por limitações de hardware, algumas funcionalidades importantes para vários projetos, como conexão em rede, não estão presentes no Arduino (a não ser que seja um modelo designado especificamente para tal). Para contornar esse problema, foram criados os Shields. Shields são acessorios que acoplamos ao Arduino, dando a ele uma funcionalidade extra e, normalmente, sem perder em número de portas. Existem diversos shields disponíveis no mercado, ja que qualquer um pode desenvolver seu próprio shields e comercializá-lo. Abaixo alguns exemplos de shields mais usados e famosos:

4.6.1 Ethernet Shield



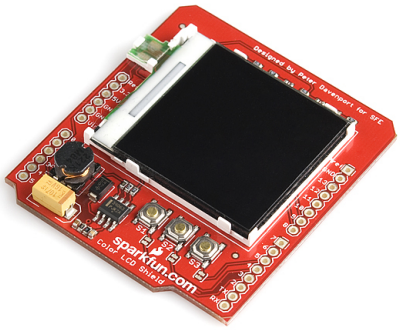
Este shield permite que seu Arduino possa se conectar à internet atraves de um cabo ethernet. Vem com um suporte para um cartão micro-SD, para armazenar informações e arquivos para ou da internet.

4.6.2 Gameduino



Permite a criação de jogos 16-bits, com vários exemplos ja inclusos. Possui saída VGA e para fone de ouvido e caixa de som.

4.6.3 LCD Shield

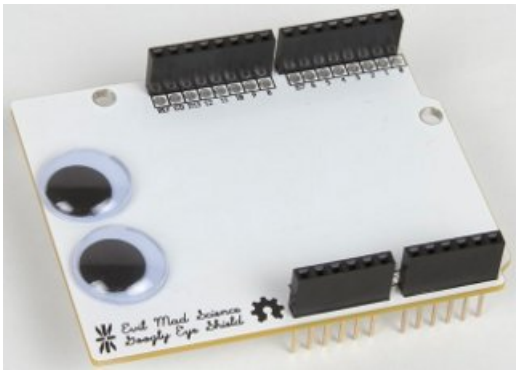


Adiciona uma tela LCD integrada, com uma resolução e capacidade superiores aos displays comuns. Existem também outras versoes com teclados imbutidos.

4.6.4 Joystick Shield

Controle com quatro botões de pressão e um analógico. Também possui funções facilitadas para a utilização dos botões.

4.6.5 Evil Mad Science Googly Eyes Shield



E por ultimo, o mais interessante! Este shield adiciona ao seu projeto a mais importante de todas as funcionalidades: olhos! (sim, isto é tudo que ele faz)

4.7 Exercícios práticos

1. Faça um circuito onde três LEDs acendam em seqüência, com um atraso de um segundo entre eles, e depois apaguem igualmente em seqüência.

2. Usando um LED vermelho, um amarelo e um verde, crie um semáforo de trânsito - ou seja, o LED verde deve ficar ligado por um determinado intervalo de tempo, seguido pelo amarelo, depois o vermelho, voltando para o verde.

3. Crie um LED inteligente: ou seja, utilizando um sensor de luminosidade, faça-o ligar se o ambiente estiver escuro demais.

4. Projete o protótipo de uma fechadura com senha - ou seja, a porta só abre se a senha digitada for correta. Para facilitar, utilize um LED vermelho para representar a porta fechada e um LED verde para representar que a senha digitada foi aceita.

5. Melhore seu protótipo adicionando um atuador sonoro, que avisa caso a senha esteja errada, caso ela esteja certa e denuncie caso alguém erre a senha mais de três vezes. Utilize sons diferentes para cada caso.

6. Utilizando um potenciômetro, crie um ajustador de volume para um buzzer. Em outras palavras, aumenta ou diminua a intensidade do som do buzzer através do potenciômetro.