Arduino
Guia Iniciante

MULTILÓGICA SHOP

Open Source Hardware

www.multilogica-shop.com

Versão 1.0

Índice

Prefácio

A Multilógica-Shop, a partir do "Guia del Arduino" criado pela Tienda de Robótica da Colombia, traz este guia que aborda a aprendizagem sobre o conceito DIY (*Do it yourself*) ou em portugês "Faça você mesmo".

A partir de um detalhado estudo elaboramos o Kit Arduino Iniciante, baseado na placa Arduino Uno R3, base para todo este material didático.

Neste guia abordaremos temas fundamentais como o hardware e software livre, revisando de maneira cuidadosa o projeto Arduino e usando como base o software Fritzing para realizar montagens claras e semelhantes à realidade.

Não é necessário que você saiba de eletrônica ou programação porque com os próximos capítulos repassaremos os conceitos fundamentais.

Após conhecer estes temas básicos teremos um capítulo dedicado a explicar componentes eletrônicos como um LED, um motor, um relê e muito mais. Também foi desenvolvido um capítulo especial sobre Arduino e Fritzing que você não pode perder.

O Kit Arduino Iniciante e este guia são ideais para todo âmbito de aprendizagem desde o colégio até a universidade, e se você se dedica por hobby não pode deixar de ter este guia, já que um capítulo completo está dedicado a mostrar todo o kit.

Finalmente chegamos a uma parte mais que especial, ao capítulo dos tutoriais, onde passo a passo serão explicados alguns exemplos, durante os quais você encontrará perguntas, dicas e exercícios.

A Multilógica-Shop

MULTILÓGICA SHOP

Open Source Hardware

A Multilógica foi fundada em 1990 com foco em tecnologia e desenvolvimento. Em 2009 iniciou uma nova etapa com a importação de produtos Open Source e marcas relacionadas de grande expressividade internacional. O Arduino sempre foi um dos principais objetivos e com ele centenas de produtos hoje fazem parte do catálogo virtual da loja.

A qualidade no serviço e a atenção ao cliente sempre estiveram entre nossos principais objetivos, por conta disso possuímos o catálogo mais completo do Brasil no setor e mantemos a meta de atender aos mais exigentes consumidores e desenvolvedores.

Hoje, depois de milhares de cliente atendidos, apresentamos o Guia Iniciante do Arduino.

Aproveitem a leitura!

Objetivo

Conhecer o funcionamento das coisas é algo que estimula nossa curiosidade desde o início dos tempos. Hoje em dia vemos uma realidade onde encontramos com facilidade a automação, a domótica (automação de casas e edifícios), a interação entre pessoas e máquinas, a eletrônica, a mecânica e a programação.

Quase qualquer processo que podemos imaginar tem uma porcentagem de dependência destas máquinas. Por exemplo, seu despertador. Ele tocou às 6 horas da manhã para que você fosse para a escola ou o trabalho. Essa máquina trabalhou durante toda a noite para, ao final, avisar que era hora de despertar.

O propósito deste guia é abordar o conceito de computação física, que é a capacidade de interação e comunicação de uma máquina com os humanos, usando sensores e atuadores.

As decisões serão executadas pelo microcontrolador que é o núcleo da placa Arduino, o coração deste material.

O Que Vou Aprender?

Muitas vezes pensamos que os temas tecnológicos pedem uma grande habilidade técnica e um grande conhecimento, mas isso não é verdade. Nosso objetivo com este guia é fazer com que esses processos tecnológicos sejam simples de entender, e demonstrar que aqueles mais complexos na verdade são a união de vários processos simples.

Com este guia você vai aprender conceitos que poderá aplicar em seus próprios projetos.

Fonte de Informação

A Wikipedia é uma enciclopédia na nuvem onde você pode encontrar grande variedade de informação em diferentes idiomas e é de uso livre.

Neste guia incorporamos conteúdo da Wikipedia (textos e imagens) com o objetivo de explicar os diversos conceitos que abordamos. O conteúdo da Wikipedia utilizado neste guia foi traduzido diretamente em alguns casos, e em outros os conceitos foram reeditados para explicar mais facilmente uma ideia.

Wikipedia é uma plataforma de conteúdo livre, de modo que todo o texto está disponível sob a licença Creative Commons (Atribuição – Compartilhamento pela mesma Licença - by-sa).

A maior parte do conteúdo também está disponível sob a Licença de Documentação Livre GNU (GFDL). Isso significa que o conteúdo Wikipedia pode ser distribuído de acordo com o estabelecido nestas licenças.

WIKIPEDIA
The Free Encyclopedia

Segurança e Cuidados

Sua área de trabalho é um espaço muito importante para desenvolver seus projetos Arduino.

Trabalhe sempre em um ambiente tranquilo e bem iluminado e mantenha sua bancada sempre limpa e seca.

Cuidado ao trabalhar com qualquer tipo de circuito impresso como o próprio Arduino. As conexões destas placas são expostas, portanto, não apoie sua placa em suportes condutores e tenha cuidado com cabos desencapados nas proximidades. Embora estas situações possam causar curtos circuitos elas não oferecem riscos físicos devido às baixas voltagens e potências envolvidas. Lembre-se, você não vai se machucar, mas destruir uma placa é sempre frustrante.

Os produtos do site, bem como os kits destinados a iniciantes, são apenas bases para a compreensão dos princípios de funcionamento dos diversos elementos envolvidos.

Sendo assim, a Multilógica-Shop não se responsabiliza por mal uso dos produtos, imperícia ou falha prática na execução dos experimentos. Não se responsabiliza pela aplicação em equipamentos de terceiros, pelo uso comercial de qualquer experimento ou informação contida, bem como insucesso ou lucros cessantes de qualquer natureza.

Não recomenda e não dá anuência em testes com animais, no corpo humano e em suporte de vida. Não se responsabiliza por, e não recomenda o uso em transporte de cargas suspensas.

Não se responsabiliza por acidentes ou ferimentos que possam advir de experimentos com quaisquer de nossos produtos. E não recomenda e desencoraja o manuseio de tensões diretas da rede de distribuição elétrica.

1 Conceitos Básicos

1.1 Computação Física

A computação física significa a construção de sistemas interativos físicos mediante o uso de software e hardware que integrados podem sentir e responder ao mundo analógico. Embora esta definição seja ampla o suficiente para englobar aspectos como os sistemas inteligentes de controle de tráfico de automóveis ou os processos de automatização em fábricas, em um sentido mais amplo a computação física é uma estrutura criativa para a compreensão da relação entre os seres humanos e o mundo digital.

Na prática, frequentemente este termo descreve desenhos de projetos DIY ou objetos que utilizam sensores e microcontroladores para traduzir entradas analógicas a sistemas baseados em software, ou controlar dispositivos eletromecânicos como motores, servos, iluminação ou outro hardware.

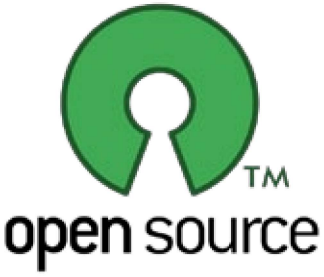
Outras implementações de computação física trabalham com o reconhecimento de voz, que captam e interpretam as ondas sonoras através de microfones ou outros dispositivos de detecção de ondas sonoras, também a visão por computador, que aplica algoritmos aos vídeos detectados por algum tipo de câmera. Interfaces táteis são também um exemplo de computação física.

O prototipado (criar montagens rápidas com ajuda de uma protoboard e componentes básicos de eletrônica) tem um papel importante na computação física. Ferramentas como o Arduino e o Fritzing são úteis para designers, artistas, estudantes e hobistas porque ajudam a elaborar protótipos rapidamente.

1.2 Open Source Hardware

Open Source Hardware consiste em dispositivos físicos de tecnologia concebidos e oferecidos pelo movimento de design aberto. Tanto o software livre como o open source hardware são criados sob o movimento de cultura open source e aplica este conceito a uma variedade de componentes. O termo normalmente significa que a informação sobre o hardware é facilmente reconhecida. O design no hardware (ou seja, desenhos mecânicos, esquemas, lista de materiais, dados de layout do PCB, código fonte e dados de layout de circuitos integrados), além do software livre que aciona o hardware, estão todos liberados com a abordagem livre e open source.

Anualmente a *Open Source Hardware Association* organiza a conferência *Open-Hardware Summit*, da qual a Multilógica-Shop é patrocinadora, que é a primeira conferência abrangente do mundo sobre hardware aberto, um espaço para discutir e chamar a atenção para este movimento em rápido crescimento.



1.3 Software Livre

Software livre é o software que é distribuído juntamente com o seu código-fonte, e é liberado sob os termos que garantem aos usuários a liberdade de estudar, adaptar/modificar e distribuir o software. O software livre é muitas vezes desenvolvido em colaboração entre programadores voluntários como parte de um projeto de desenvolvimento de software open source.

A *Free Software Foundation* considera um software como livre quando atende aos quatro tipos de liberdade para os usuários:

Liberdade 0: A liberdade para executar o programa, para qualquer propósito;

Liberdade 1: A liberdade de estudar o software;

Liberdade 2: A liberdade de redistribuir cópias do programa de modo que você possa ajudar ao seu próximo;

Liberdade 3: A liberdade de modificar o programa e distribuir estas modificações, de modo que toda a comunidade se beneficie.

Os usuários deste tipo de software são livres porque não precisam pedir permissão e não estão vinculados a licenças proprietárias restritivas.

A *Open Source Initiative* (OSI) - Iniciativa pelo Código Aberto - é uma organização dedicada a promover o software de código aberto ou software livre. Ela foi criada para incentivar uma aproximação de entidades comerciais com o software livre. Sua atuação principal é a de certificar quais licenças se enquadram como licenças de software livre, e promovem a divulgação do software livre e suas vantagens tecnológicas e econômicas.

A OSI, assim como muitos membros da comunidade, considera que o software é, em primeiro lugar, uma ferramenta, e que o mérito dessa ferramenta deve ser julgado com base em critérios técnicos. Para eles, o software livre no longo prazo é economicamente mais eficiente e de melhor qualidade e, por isso, deve ser incentivado. Além disso, a participação de empresas no ecossistema do software livre é considerada fundamental, pois são as empresas que viabilizam o aumento no desenvolvimento, implantação e uso do software livre.

1.4 Arduino

Arduino é uma plataforma de eletrônica aberta para a criação de protótipos baseada em software e hardware livres, flexíveis e fáceis de usar. Foi desenvolvida para artistas, designers, hobistas e qualquer pessoa interessada em criar objetos ou ambientes interativos.

O Arduino pode adquirir informação do ambiente através de seus pinos de entrada, para isso uma completa gama de sensores pode ser usada. Por outro lado, o Arduino pode atuar no ambiente controlando luzes, motores ou outros atuadores.

Os campos de atuação para o controle de sistemas são imensos, podendo ter aplicações na área de impressão 3D, robótica, engenharia de transportes, engenharia agronômica, musical, moda e tantas outras.

O microcontrolador da placa Arduino é programado mediante a linguagem de programação Arduino, baseada em Wiring, e o ambiente de desenvolvimento (IDE) está baseado em Processing.

Os projetos desenvolvidos com Arduino podem ser executados mesmo sem a necessidade de estar conectados a um computador, apesar de que também podem ser feitos comunicando-se com diferentes tipos de software (como Flash, Processing ou MaxMSP).

As placas podem ser feitas a mão ou compradas montadas de fábrica. O download do software pode ser feito de forma gratuita e os desenhos da placa estão disponíveis sob uma licença aberta, assim você também é livre para adaptá-lo às suas necessidades.

www.arduino.cc

1.5 Processing

Processing é uma linguagem de programação de código aberto e ambiente de desenvolvimento integrado (IDE), construído para as artes eletrônicas e comunidades de projetos visuais com o objetivo de ensinar noções básicas de programação de computador em um contexto visual.

O projeto foi iniciado em 2001 por Casey Reas e Ben Fry, ambos ex-membros do Grupo de Computação do MIT Media Lab.

Um dos objetivos do Processing é atuar como uma ferramenta para não-programadores iniciados com a programação, através da satisfação imediata com um retorno visual.

1.6 Fritzing

Fritzing é um programa de automação de design eletrônico open source destinado a ajudar designers e artistas a passar dos protótipos (utilizando, por exemplo, placas de teste) para os produtos finais.

Fritzing foi criado sob os princípios de Processing e Arduino e permite a designers, artistas, pesquisadores e amadores documentar seu protótipo baseado em Arduino e criar diagramas de circuitos impressos para mais tarde fabricar. Além disso, tem um site complementar que ajuda a compartilhar e discutir projetos, experiências e reduzir os custos de fabricação.

1.7 Creative Commons

A CC é uma organização não governamental sem fins lucrativos localizada na California, voltada a expandir a quantidade de obras criativas disponíveis, através de suas licenças que permitem a cópia e compartilhamento com menos restrições que o tradicional todos direitos reservados.

As licenças *Creative Commons* foram idealizadas para permitir a padronização de declarações de vontade no tocante ao licenciamento e distribuição de conteúdos culturais em geral (textos, músicas, imagens, filmes e outros), de modo a facilitar seu compartilhamento e recombinação, sob a égide de uma filosofia copyleft.

Creative Commons tem sido abraçada por muitos criadores de conteúdo, pois permite controle sobre a maneira como sua propriedade intelectual será compartilhada.

1.8 Licença da Obra

Renúncia - Qualquer uma das condições acima pode ser renunciada pelo titular do direito de autor ou pelo titular dos direitos conexos, se obtiver deste uma autorização para usar o trabalho sem essa condição.

Domínio Público - Quando a obra ou qualquer dos seus elementos se encontrar no domínio público, nos termos da lei aplicável, esse estatuto não é de nenhuma forma afetado pela licença.

Outros Direitos - A licença não afeta, de nenhuma forma, qualquer dos seguintes direitos:

Os seus direitos de "uso legítimo" (fair dealing ou fair use) concedidos por lei, ou outras exceções e limitações aplicáveis ao direito de autor e aos direitos conexos;

Os direitos morais do autor;

Direitos de que outras pessoas possam ser titulares, quer sobre o trabalho em si, quer sobre a forma como este é usado, tais como os direitos de publicidade ou direitos de privacidade.

Aviso — Em todas as reutilizações ou distribuições, tem de deixar claro quais são os termos da licença deste trabalho. A melhor forma de fazê-lo é colocando um link para esta página.

2 Eletrônica

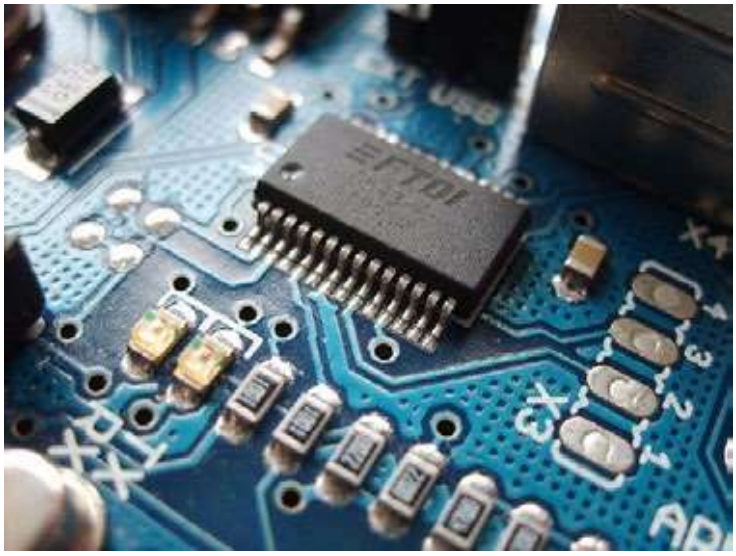
2.1 Conceito de Eletrônica

Numa definição mais abrangente, podemos dizer que a eletrônica é o ramo da ciência que estuda o uso de circuitos formados por componentes elétricos e eletrônicos, com o objetivo principal de representar, armazenar, transmitir ou processar informações além do controle de processos e servo mecanismos.

Sob esta ótica, também se pode afirmar que os circuitos internos dos computadores, os sistemas de telecomunicações, os diversos tipos de sensores e transdutores estão, todos, dentro da área de interesse da eletrônica.

Divide-se em analógica e em digital porque suas coordenadas de trabalho optam por obedecer estas duas formas de apresentação dos sinais elétricos a serem tratados.

Também é considerada um ramo da eletricidade que, por sua vez, é um ramo da Física onde se estudam os fenômenos das cargas elétricas elementares, as propriedades e comportamento do elétron, fótons, partículas elementares, ondas eletromagnéticas, etc.



2.2 Voltagem

Tensão elétrica, também conhecida como diferença de potencial (DDP) ou voltagem, é a diferença de potencial elétrico entre dois pontos ou a diferença em energia elétrica potencial por unidade de carga elétrica entre dois pontos. Sua unidade de medida é o volt, ou joules por coulomb.

A diferença de potencial é igual ao trabalho que deve ser feito por unidade de carga contra um campo elétrico para se movimentar uma carga qualquer.

Um voltímetro pode ser utilizado para se medir a diferença de potencial entre dois pontos em um sistema, sendo que usualmente um ponto referencial comum é o terra.

A tensão elétrica pode ser causada por campos elétricos estáticos, por uma corrente elétrica sob a ação de um campo magnético, por campo magnético variante ou uma combinação dos três.

2.3 Corrente Elétrica

A corrente elétrica é o fluxo ordenado de partículas portadoras de carga elétrica, ou também, é o deslocamento de cargas dentro de um condutor, quando existe uma diferença de potencial elétrico entre as extremidades.

A unidade padrão no Sistema Internacional de Unidades para medir a intensidade de corrente é o ampere. Para medir a corrente, pode-se utilizar um amperímetro.

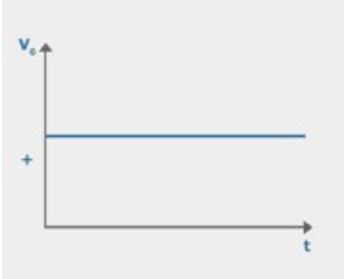
Uma corrente elétrica, já que se trata de um movimentos de cargas, produz um campo magnético, um fenômeno que pode ser usado como um eletroímã, sendo este o princípio de funcionamento de um motor.

2.4 Corrente Contínua

Corrente contínua, corrente direta, corrente galvânica ou ainda corrente constante (CC ou DC do inglês *direct current*) é o fluxo ordenado de elétrons sempre numa direção.

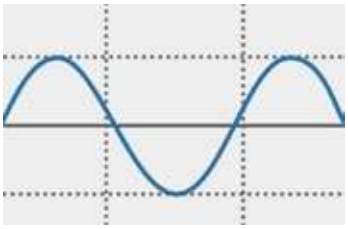
Esse tipo de corrente é gerado por baterias de automóveis ou de motos (6, 12 ou 24V), pequenas baterias (geralmente de 9V), pilhas (1,2V e 1,5V), dínamos, células solares e fontes de alimentação de várias tecnologias, que retificam a corrente alternada para produzir corrente contínua. Normalmente é utilizada para alimentar aparelhos eletrônicos (entre 1,2V e 24V) e os circuitos digitais de equipamento de informática (computadores, modems, hubs, etc.).

Este tipo de circuito possui um polo negativo e outro positivo (é polarizado), cuja intensidade é mantida. Mais corretamente, a intensidade cresce no início até um ponto máximo, mantendo-se contínua, ou seja, sem se alterar. Quando desligada, diminui até zero e extingue-se.



2.5 Corrente Alternada

A corrente alternada (CA ou AC - do inglês *alternating current*), é uma corrente elétrica cujo sentido varia no tempo, ao contrário da corrente contínua cujo sentido permanece constante ao longo do tempo. A forma de onda usual em um circuito de potência CA é senoidal por ser a forma de transmissão de energia mais eficiente. Entretanto, em certas aplicações, diferentes formas de ondas são utilizadas, tais como triangular ou ondas quadradas. Enquanto a fonte de corrente contínua é constituída pelos pólos positivo e negativo, a de corrente alternada é composta por fases (e, muitas vezes, pelo fio neutro).



2.6 Resistência

Resistência elétrica é a capacidade de um corpo qualquer se opor à passagem de corrente elétrica mesmo quando existe uma diferença de potencial aplicada. É medida em ohms (Ω).

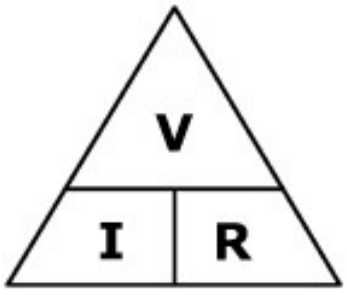
Resistores são componentes que têm por finalidade oferecer uma oposição à passagem de corrente elétrica, através de seu material. A essa oposição damos o nome de resistência elétrica. Causam uma queda de tensão em alguma parte de um circuito elétrico, porém jamais causam quedas de corrente elétrica, apesar de limitar a corrente. Isso significa que a corrente elétrica que entra em um terminal do resistor será exatamente a mesma que sai pelo outro terminal, porém há uma queda de tensão.

Utilizando-se disso, é possível usar os resistores para controlar a tensão sobre os componentes desejados.

2.7 Lei de Ohm

A Lei de Ohm afirma que a corrente (I) que circula através de um dado circuito é directamente proporcional à voltagem aplicada (V), e inversamente proporcional à resistência (R) da mesma.

A pirâmide ao lado é muito útil para conhecer esta fórmula. Por exemplo, cubra com um dedo a letra V (voltagem), então a voltagem será igual à corrente (I) vezes a resistência (R). Ou, para calcular a resistência, divida a voltagem (V) pela corrente (I).



2.8 Sistemas Eletrônicos

Um sistema eletrônico é um conjunto de circuitos que interagem entre si para obter um resultado. Uma forma de entender os sistemas eletrônicos consiste em dividi-los em entradas, saídas e processamento de sinais.

2.9 Entradas

As entradas, ou inputs, são sensores eletrônicos ou mecânicos que tomam os sinais (em forma de temperatura, pressão, umidade, contato, luz, movimento, ph, etc.) do mundo físico e converte em sinais de corrente ou voltagem.

Exemplos de entradas são sensores de gás, temperatura, pulsadores, fotocélulas, potenciômetros, sensores de movimento, e muitos mais.

2.10 Saídas

As saídas, ou outputs, são atuadores, ou outros dispositivos que convertem os sinais de corrente ou voltagem em sinais fisicamente úteis como movimento, luz, som, força ou rotação, entre outros.

Exemplos de saídas são motores, LEDs ou sistemas de luzes que acendem automaticamente quando escurece ou um buzzer que gere diversos tons.

2.11 Processamento de Sinal

O processamento de sinal é realizado mediante circuitos conhecidos como microcontroladores. São circuitos integrados construídos para manipular, interpretar e transformar os sinais de voltagem e corrente vindos dos sensores (entradas) e e ativar determinadas ações nas saídas.

2.12 Resumo dos Sistemas Eletrônicos

Como exemplo imaginamos um aparelho de TV. A entrada é um sinal recebido por uma antena ou um cabo. Os circuitos integrados do interior do aparelho extraem a informação sobre brilho, cor e som deste sinal. Os dispositivos de saída são a tela LCD, que converte os sinais eletrônicos em imagens visíveis, e as caixas de som, que emitem o som.

Outro exemplo pode ser um circuito que controle a temperatura de um ambiente. Um sensor de temperatura e um circuito integrado são os responsáveis por converter um sinal de entrada em um nível de voltagem apropriado. Se a temperatura registrada do ambiente é muito alta, este circuito enviará a informação a um motor para que este ligue um ventilador que resfriará o local.

2.13 Sinais Eletrônicos

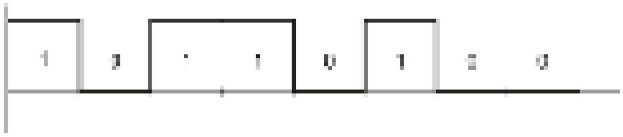
As entradas e saídas de um sistema eletrônico serão consideradas como sinais variáveis. Em eletrônica se trabalha com variáveis que são tomadas na forma de tensão ou corrente, que podem simplesmente ser chamados de sinais.

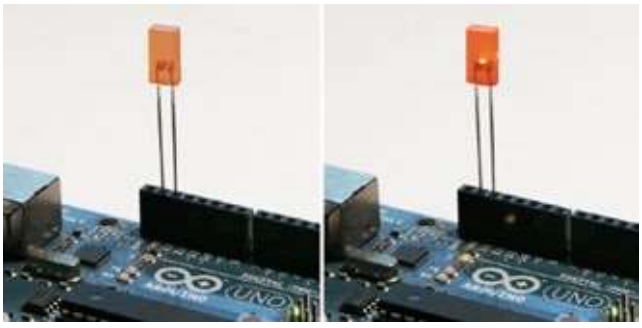
Os sinais podem ser de dois tipos: digital ou analógico.

2.14 Variável Digital

Também chamadas de variáveis discretas, se caracterizam por ter dois estados diferentes e portanto também podem ser chamadas de binárias (em lógica seria valores Verdadeiro (V) e Falso (F), ou poderiam ser 1 ou 0 respectivamente).

Um exemplo de um sinal digital é o interruptor da campainha da sua casa, porque ele tem somente dois estados, pulsado e sem pulsar.





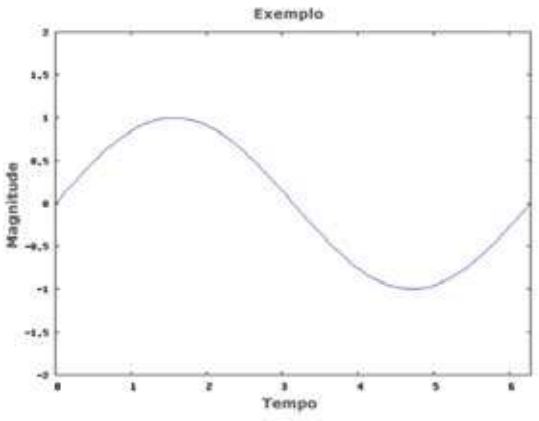
Apagado
0

Aceso
1

2.15 Variável Analógica

São aquelas que podem tomar um número infinito de valores compreendidos entre dois limites. A maioria dos fenômenos da vida real são sinais deste tipo (som, temperatura, luminosidade, etc.).

Um exemplo de sistema eletrônico analógico é de um palestrante, que se preocupa em amplificar o som da sua voz para que seja escutado por uma grande audiência. As ondas de som que são analógicas na sua origem são capturadas por um microfone e convertidas em uma pequena variação analógica de tensão, denominada sinal de áudio.



2.16 Entrada/Saída Digital

Entrada
Botão

Saída
LED

Entrada
Reed switch

2.17 Entrada/Saída Analógica

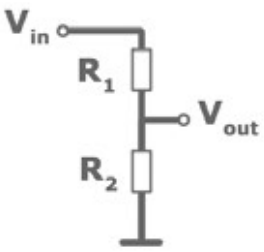
Entrada
LDR

Saída
Motor DC

Entrada
Potenciômetro

2.18 Divisor de Voltagem

Em eletrônica, a regra do divisor de tensão é uma técnica de projeto utilizada para criar uma tensão elétrica (V_{out}) que seja proporcional à outra (V_{in}). Desta forma a voltagem de uma fonte é repartida entre uma ou mais resistências conectadas em série. Em um circuito deste tipo, duas resistências são ligadas em série como no esquema a seguir:



A tensão de saída, V_{out}, é dada pela fórmula:

$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in}$$

Desta forma podemos obter qualquer fração entre 0 e 1 da tensão V_{in}.

2.19 Conversor Analógico-Digital

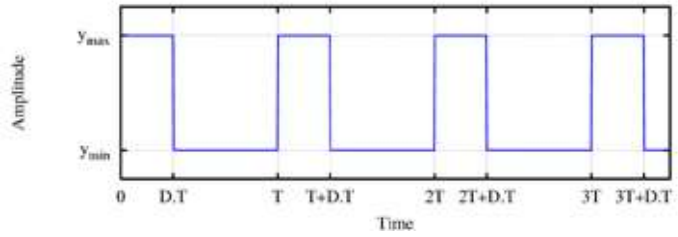
Um conversor analógico-digital (ou ADC em inglês - *Analog-to-Digital Converter*) é um dispositivo eletrônico capaz de gerar uma representação digital a partir de uma grandeza analógica, convertendo uma entrada analógica de voltagem em um valor binário. Se utiliza em equipamentos eletrônicos como computadores, gravadores de som e vídeo e equipamentos de telecomunicações.

Estes conversores são muito úteis na interface entre dispositivos digitais e dispositivos analógicos e são utilizados em aplicações como leitura de sensores, digitalização de áudio, vídeo, etc..

2.20 Modulação por Largura de Pulso PWM

A modulação por largura de pulso (MLP) - mais conhecida pela sigla em inglês PWM (*Pulse-Width Modulation*) - de um sinal ou em fontes de alimentação envolve a modulação de sua razão cíclica (*duty cycle*) para transportar qualquer informação sobre um canal de comunicação ou controlar a quantidade de energia que se envia em uma carga.

Por exemplo, se aplicamos PWM a um LED podemos variar a intensidade do brilho, e se aplicamos PWM a um motor DC conseguimos variar a velocidade do mesmo com a característica de manter sua força constante.



2.21 Comunicação Serial

É uma interface de comunicação de dados digitais em que a informação é enviada um bit de cada vez, sequencialmente. É diferente da comunicação paralela, em que todos os bits de cada símbolo são enviados juntos. A comunicação serial é usada em toda comunicação de longo alcance e na maioria das redes de computadores.

Um de seus usos é monitorar através da tela do computador o estado de um periférico conectado. Por exemplo ao pulsar a letra A do teclado se deve acender um LED conectado de maneira remota ao computador.

3 Componentes Eletrônicos

3.1 Microcontrolador

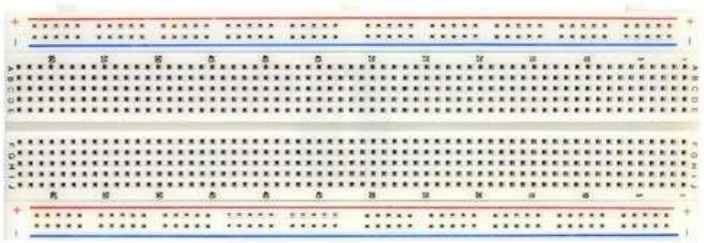
Um microcontrolador é um circuito integrado programável, capaz de executar as ordens gravadas em sua memória. Um microcontrolador possui em seu interior três unidades funcionais principais: unidade central de processamento, memória e periféricos de entrada e saída.

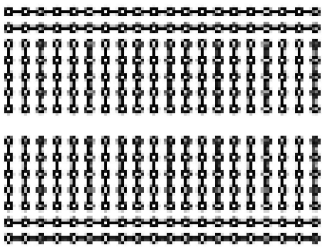
Os microcontroladores se diferenciam dos processadores pois, além dos componentes lógicos e aritméticos usuais de um microprocessador de uso geral, o microcontrolador integra elementos adicionais em sua estrutura interna, como memória de leitura e escrita para armazenamento de dados, memória somente de leitura para armazenamento de programas, EEPROM para armazenamento permanente de dados, dispositivos periféricos como conversores analógico/digitais (ADC), conversores digitais/analógicos (DAC) em alguns casos; e, interfaces de entrada e saída de dados.

São geralmente utilizados em automação e controle de produtos e periféricos, como sistemas de controle de motores automotivos, controles remotos, máquinas de escritório e residenciais, brinquedos, sistemas de supervisão, etc. Por reduzir o tamanho, custo e consumo de energia, e se comparados à forma de utilização de microprocessadores convencionais, aliados a facilidade de desenho de aplicações, juntamente com o seu baixo custo, os microcontroladores são uma alternativa eficiente para controlar muitos processos e aplicações.

3.2 Protoboard

É uma placa reutilizável usada para construir protótipos de circuitos eletrônicos sem solda. Uma protoboard é feita por blocos de plástico perfurados e várias lâminas finas de uma liga metálica de cobre, estanho e fósforo.





Conexões internas.

3.3 Resistor

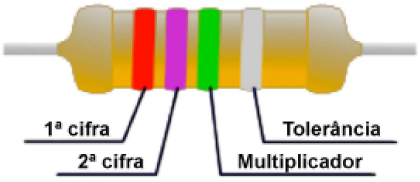
É um componente formado por carbono e outros elementos resistentes usados para limitar a corrente elétrica em um circuito.

Por seu tamanho muito reduzido, é inviável imprimir nos resistores as suas respectivas resistências. Optou-se então pelo código de cores, que consiste em faixas coloridas no corpo do resistor indicadas como a, b, c e % de tolerância. As primeiras três faixas servem para indicar o valor nominal de suas resistência e a última faixa, a porcentagem na qual a resistência pode variar seu valor nominal, conforme a seguinte equação:

$R = (10a + b) \times 10^{c} \pm \%$ da tolerância

Exemplo:

Um resistor de 2.700.000Ω (2,7MΩ), com uma tolerância de ±10% seria representado pela figura.



Símbolo

1ª cifra: vermelho (2)
2ª cifra: violeta (7)
Multiplicador: verde (10^5)
Tolerância: prata (±10%)

Valor nominal

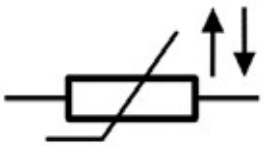
Cor	Preto	Marrom	Vermelho	Laranja	Amarelo	Verde	Azul	Violeta	Cinza	Branco
Valor	0	1	2	3	4	5	6	7	8	9

Valor da tolerância

Cor	Marrom	Dourado	Prata	Sem cor
Valor	±1%	±5%	±10%	±20%

3.4 Termistor

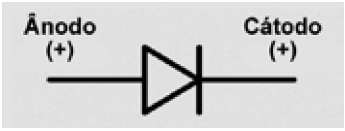
Símbolo

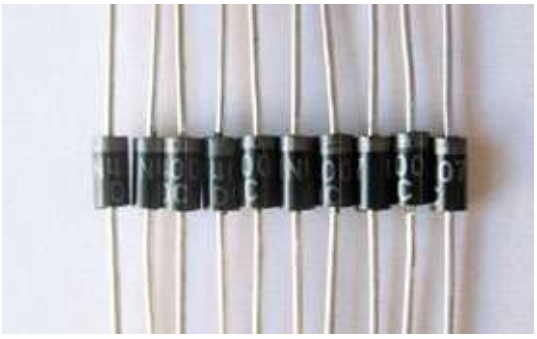


O termistor NTC (do inglês *Negative Temperature Coefficient*) é um componente eletrônico semicondutor sensível à temperatura, utilizado para controle, medição ou polarização de circuitos eletrônicos. Possui um coeficiente de variação de resistência que varia negativamente conforme a temperatura aumenta, ou seja, a sua resistência elétrica diminui com o aumento da temperatura.

3.5 Diodo

Símbolo

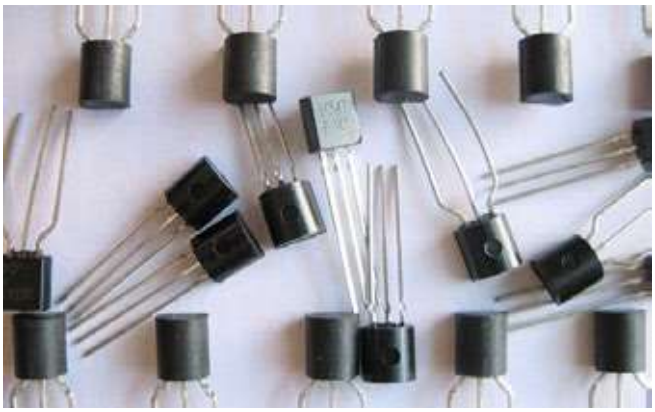




É o tipo mais simples de componente eletrônico semicondutor. É um componente que permite que a corrente atravesse somente em um sentido.

3.6 Transistor

Símbolo



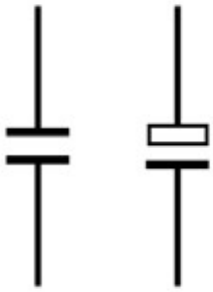
É utilizado principalmente como amplificador, interruptor de sinais elétricos e como retificador elétrico em um circuito. O termo provém do inglês *transfer resistor* (resistor/ resistência de transferência), como era conhecido pelos seus inventores.

O processo de transferência de resistência, no caso de um circuito analógico, significa que a impedância característica do componente varia para cima ou para baixo da polarização pré-estabelecida. Graças a esta função, a corrente elétrica que passa entre coletor (C), base (B) e emissor (E) do transistor varia dentro de determinados parâmetros pré-estabelecidos e processa a amplificação de sinal.

Entende-se por "amplificar" o procedimento de tornar um sinal elétrico mais fraco num mais forte. Um sinal elétrico de baixa intensidade, como o sinal gerado por um microfone, é injetado num circuito eletrônico (transistorizado por exemplo), cuja função principal é transformar este sinal fraco gerado pelo microfone em sinais elétricos com as mesmas características. A este processo todo dá-se o nome de ganho de sinal.

Atualmente os transistores se encontram em todos os aparelhos de uso doméstico e cotidiano: rádios, televisões, gravadores, aparelhos de som, microondas, lavadoras, carros, calculadores, impressoras, celulares, etc.

3.7 Capacitor

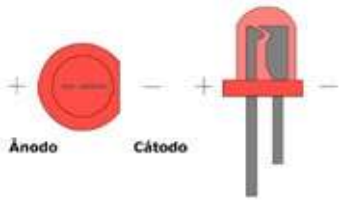


Símbolo

O capacitor é um dispositivo elétrico capaz de armazenar cargas elétricas. Em circuitos eletrônicos alguns componentes necessitam que haja alimentação em corrente contínua, enquanto a fonte está ligada em corrente alternada. A resolução deste problema é um dos exemplos da utilidade de um capacitor.

Este elemento é capaz de armazenar energia potencial elétrica durante um intervalo de tempo, e é construído utilizando um campo elétrico uniforme. Um capacitor é composto por duas peças condutoras, chamadas armaduras e um material isolante com propriedades específicas chamado dielétrico.

3.8 LED



Simbolo

O LED (*Light Emitting Diode*) é um diodo que emite luz quando energizado. Os LED´s apresentam muitas vantagens sobre as fontes de luz incandescentes como um consumo menor de energia, maior tempo de vida, menor tamanho, grande durabilidade e confiabilidade. O LED tem uma polaridade, uma ordem de conexão. Ao conectá-lo invertido não funcionará corretamente. Revise os desenhos para verificar a correspondência do negativo e do positivo.

São especialmente utilizados em produtos de microeletrônica como sinalizador de avisos. Também é muito utilizado em painéis, cortinas e pistas de led. Podem ser encontrados em tamanho maior, como em alguns modelos de semáforos ou displays.

3.9 LED RGB

Um LED RGB é um LED que incorpora em um mesmo encapsulamento três LED´s, um vermelho (*Red*), um verde (*Green*) e outro azul (*Blue*). Desta forma é possível formar milhares de cores ajustando de maneira individual cada cor. Os três LED´s estão unidos por um negativo ou cátodo.

3.10 Display de LCD

Um display de cristal líquido, ou LCD (*liquid crystal display*), é um painel fino usado para exibir informações por via eletrônica, como texto, imagens e vídeos.

Um LCD consiste de um líquido polarizador da luz, eletricamente controlado, que se encontra comprimido dentro de celas entre duas lâminas transparentes polarizadoras. Suas principais características são leveza e portabilidade. Seu baixo consumo de energia elétrica lhe permite ser utilizado em equipamentos portáteis, alimentados por bateria eletrônica.

Um display de LCD pode variar o número de linhas e caracteres por linha, a cor dos caracteres e a cor do fundo, assim como ter ou não *backlight*. Os modelos com *backlight* possuem melhor visualização.

3.11 Botão

Um botão, ou pulsador, é utilizado para ativar alguma função. Os botões são em geral ativados ao serem pulsados. Um botão em um dispositivo eletrônico funciona geralmente como um interruptor elétrico. No seu interior há dois contatos, e se é um dispositivo normalmente fechado ou normalmente aberto, ao pulsar o botão, se ativará a função inversa à que se está realizando no momento.

Símbolo

3.12 Reed Switch

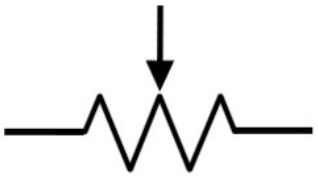
É um interruptor elétrico ativado por um campo magnético, por exemplo com um ímã. Quando os contatos estão abertos se fecham na presença de um campo magnético. Quando estão fechados se abrem.

É comumente usado em sensores de portas e janelas de alarmes anti-roubo. O ímã vai preso à porta e o reed switch ao batente.

Símbolo

3.13 Potenciômetro

Um potenciômetro é uma resistência cujo valor é variável. Desta maneira, indiretamente, pode-se controlar a intensidade de corrente que flui por um circuito se está conectado em paralelo, ou controlar a voltagem ao conectá-lo em série. São adequados para uso como elemento de controle em aparelhos eletrônicos. O usuário o aciona para variar os parâmetros normais de funcionamento. Um exemplo é o botão de volume de um rádio.



Símbolo

3.14 Fotocélula

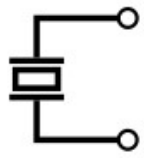
O LDR (*Light Dependant Resistor*) é uma resistência cujo valor em ohms varia de acordo com a luz incidente. Uma fotocélula apresenta um baixo valor de resistência na presença de luz e um alto valor na sua ausência.

Pode ser encontrada em vários artigos de consumo, como por exemplo em câmaras, medidores de luz, relógios com rádio, alarmes de segurança ou sistemas de iluminação pública.

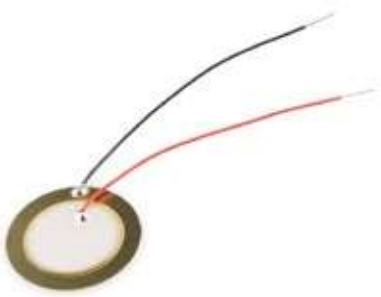
Símbolo

3.15 Transdutor Piezoelétrico

Um transdutor piezoelétrico é muito prático para detectar vibrações ou golpes. Pode ser usado como sensor através da leitura da voltagem de saída. Este transdutor eletroacústico também pode ser usado como um pequeno buzzer para produzir um som ou zumbido contínuo ou intermitente.



Símbolo

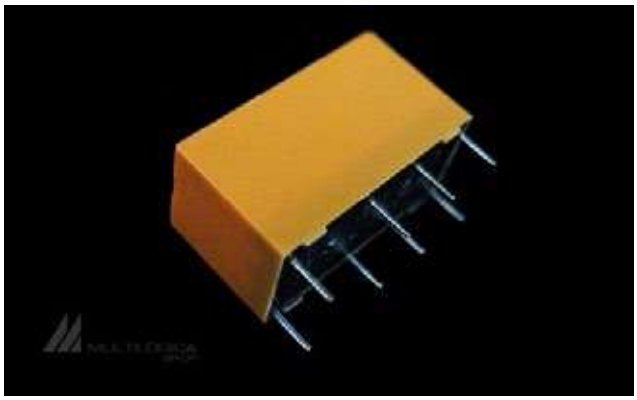


3.16 Motor CC

O motor de corrente contínua (CC) é uma máquina que converte a energia elétrica em mecânica provocando um movimento rotatório. Esta máquina de corrente contínua é uma das mais versáteis. Seu fácil controle de posição, pausa e velocidade a convertem em uma das melhores opções em aplicações de controle e automação de processos. Por exemplo, pode-se encontrar na tração de carros de brinquedo a pilhas ou nas rodas de um robô.

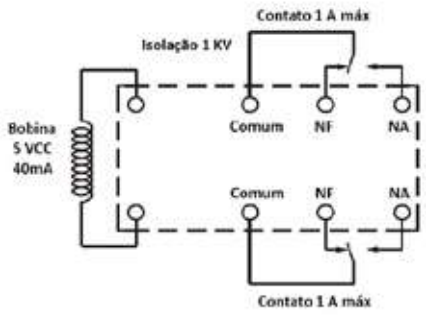
Símbolo

3.17 Relê



É um interruptor eletromecânico usado para ligar ou desligar dispositivos. Quando uma corrente circula pela bobina interna, esta cria um campo magnético que atrai um ou uma série de contatos fechando ou abrindo circuitos. Ao cessar a corrente da bobina o campo magnético também cessa, fazendo com que os contatos voltem para a posição original.

Símbolo



4 Arduino

4.1 O Projeto Arduino

O projeto Arduino começou no ano de 2005 com o objetivo de criar um dispositivo para estudantes que oferecesse controle integrado de projetos de design e interação, e que fosse mais econômico que os sistemas de criação de protótipos disponíveis até o momento.

O que chamamos hoje de Arduino (o microcontrolador) nasceu na cidade italiana de Ivrea. Nesta mesma cidade nos séculos X e XI houve um outro Arduino (um nobre) que se auto proclamou rei de toda a Itália, obviamente a coisa não funcionou e, como era comum na época, ele foi morto pelos rivais. O fato é que em sua cidade natal ele ainda é muito lembrado, a avenida principal da cidade se chama "Via Arduino" bem como muitos comércios locais.

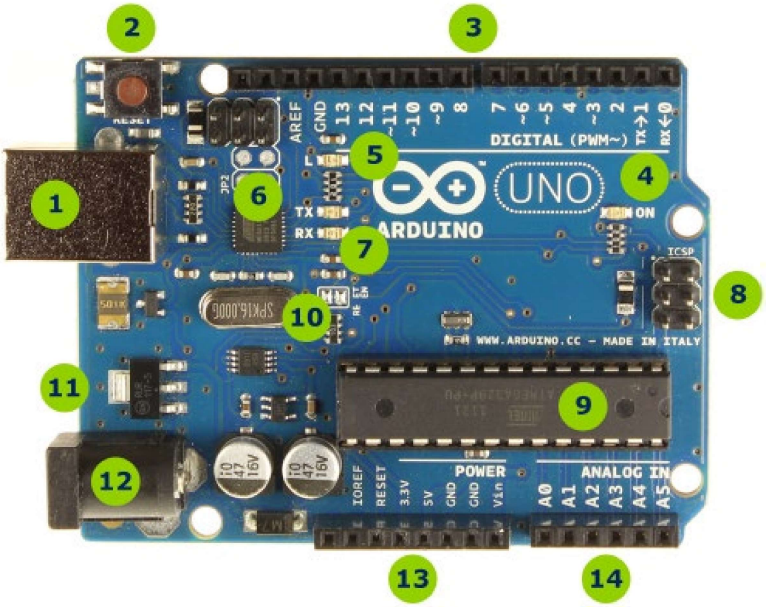
Enquanto viviam lá, os integrantes do time que criou o Arduino (o microcontrolador), depois do expediente iam tomar uma cerveja. Onde? No Bar Arduino. Assim o nome do Arduino (o microcontrolador) é uma homenagem ao Arduino (o bar) que por sua vez era uma homenagem ao outro Arduino (o nobre).

O projeto Arduino foi desenvolvido por Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. Está baseado em uma plataforma de código aberta chamada Wiring, criada pelo artista colombiano Hernando Barragán durante uma tese de um master no *Interaction Design Institute Ivrea*. Por outro lado, Wiring está baseado em Processing e seu entorno de desenvolvimento integrado foi criado por Casey Reas e Ben Fry.

"*Não acredito que o Arduino existiria sem Wiring, e não acredito que Wiring existiria sem Processing. E que Processing sem dúvida não existiria sem Design by Numbers e John Maeda*".

Entrevista a Casey Reas e Ben Fry, Shiffman Daniel (Set/2009)

4.2 Arduino Uno R3



1 - Conector USB para o cabo tipo AB
2 - Botão de reset
3 - Pinos de entrada e saída digital e PWM
4 - LED verde de placa ligada
5 - LED laranja conectado ao pin13
6 - ATmega encarregado da comunicação com o computador
7 - LED TX (transmissor) e RX (receptor) da comunicação serial
8 - Porta ICSP para programação serial
9 - Microcontrolador ATmega 328, cérebro do Arduino
10 - Cristal de quartzo 16Mhz
11 - Regulador de voltagem
12 - Conector fêmea 2,1mm com centro positivo
13 - Pinos de voltagem e terra
14 - Entradas analógicas

4.3 Família Arduino

Com o passar dos anos a linha Arduino vem crescendo mais e mais e trazendo soluções para os mais diversos projetos. Conheça um pouco mais desta família:

Arduino Leonardo

Arduino Mega2560 R3

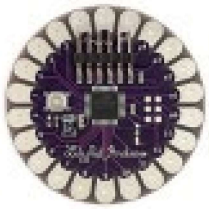
Arduino Esplora

Arduino Mega ADK

Arduino Pro

Arduino FIO V3

Arduino LilyPad



Arduino Mini 05

Arduino Pro Mini

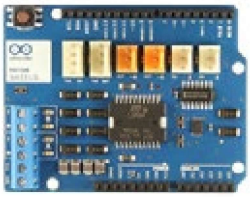
Arduino Micro

4.4 Shields para Arduino

Um shield é uma placa que permite expandir as funcionalidades originais do Arduino. Alguns exemplos:

Arduino Ethernet Shield R3

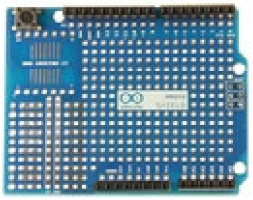
Kit Motor Shield R3



Arduino WiFi Shield

Arduino XBee Shield

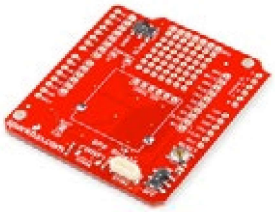
Arduino ProtoShield R3



Kit Joystick Shield

Shield celular com SM5100B

Shield GPS

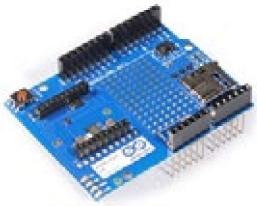


Shield LCD Colorido

Shield MP3 Player

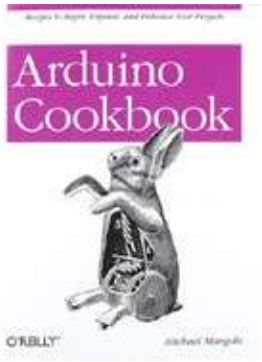
Shield WiFly

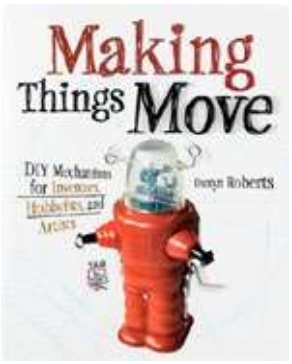
Wireless SD Shield



4.5 Livros

Várias edições em português e em inglês estão disponíveis, tanto para você começar seus estudos com a plataforma Arduino, como para ampliar seus conhecimentos.





5 Instalação de Software

5.1 Arduino em Windows

1 - Placa Arduino e um cabo USB AB

Este tutorial serve para instalação das placas Arduino Uno, Arduino Duemilanove, Nano, Arduino Mega 2560 ou Diecimila. Para outras placas da linha Arduino deve-se buscar o tutorial correspondente. Você também vai precisar de um cabo USB AB.

2 - Download do software do Arduino

Faça download da última versão do software do Arduino. Ao terminar, descompacte o arquivo e mantenha a estrutura de pastas e sub-pastas. Se quiser guarde esta pasta no drive C: do seu computador. Dentro desta pasta existe um arquivo chamado arduino.exe que é o ponto de entrada do programa do Arduino, a IDE (*Integrated Development Environment*).

3 - Conectando o Arduino

O Arduino Uno isolado usa a energia do computador através da conexão USB, não sendo necessária energia externa. Conecte a placa Arduino ao computador usando o cabo USB AB. O LED verde de energia (PWR) deve acender.

4 - Instalando os drivers

Drivers para Arduino Uno ou Arduino Mega 2560 com Windows 7, Vista ou XP:

. Conecte a placa ao computador e aguarde o Windows iniciar o processo de instalação do driver. Depois de alguns momentos o processo vai falhar. Clique em concluir e dispense a ajuda do assistente.

. Clique no Menu Principal e abra o Painel de Controle.

. Dentro do Painel de Controle, navegue até Sistema e Segurança. Na sequência clique em Sistema, selecione Hardware e depois clique em Gerenciador de Dispositivos.

. Procure por Portas (COM & LPT), onde você deve ver uma opção Arduino UNO (COMxx).

. Clique com o botão da direita em Arduino UNO (COMxx) e escolha a opção Atualizar Driver.

. Depois escolha a opção Instalar de uma lista ou local específico (Avançado), e clique em avançar.

. Finalmente navegue e escolha o driver arduino.inf localizado na pasta Drivers do software do Arduino que você baixou.

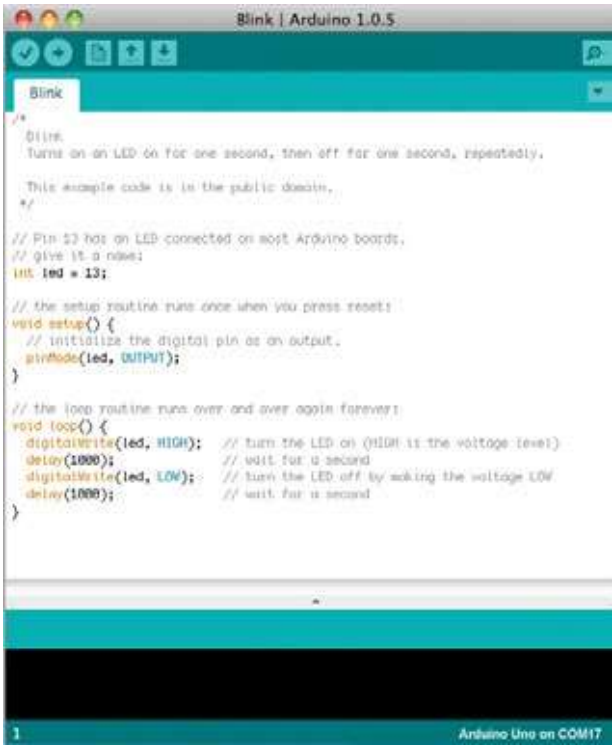
. O Windows vai finalizar a instalação do driver a partir deste ponto.

5 - Abrindo o programa Arduino

Clique duas vezes na aplicação do Arduino, o arquivo arduino.exe. Caso o programa carregue com o idioma que não é da sua preferência você pode alterar na sessão de preferências do programa.

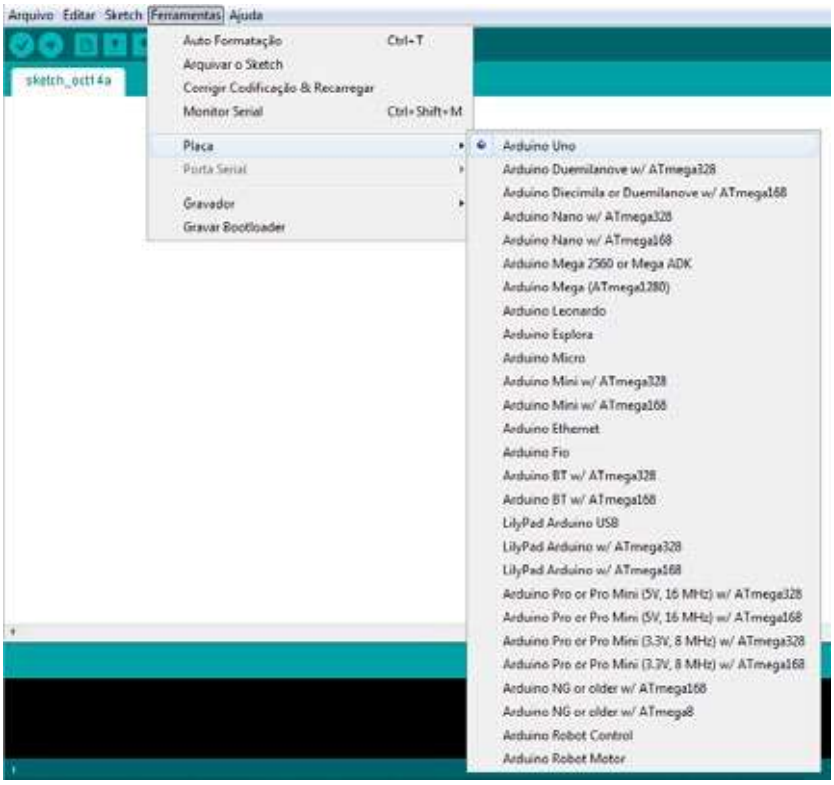
6 - Exemplo Piscar

Abra o exemplo Piscar (blink): Arquivo > Exemplos > 01.Basics > Blink



7 - Selecione sua placa

Você deve selecionar qual a sua placa Arduino: Ferramentas > Placa > Arduino Uno.



8 - Selecione a porta

Selecione agora a porta serial que conectará o Arduino: Ferramentas > Porta Serial.
Você deve selecionar a mesma porta que utilizou para confirgurar o sistema no passo 4.

9 - Carregue o programa

Agora simplesmente clique no botão Carregar da janela do programa. Espere alguns segundos. Você deve ver os LEDs RX e TX da placa piscarem. Se o processo foi executado normalmente você verá uma mensagem de "Transferência concluída".

Depois de alguns segundos você verá o LED do pin 13 piscar em laranja. Neste caso, parabéns! Seu Arduino está pronto e instalado.

Se você tiver problemas na instalação pode acessar a página oficial do Arduino com algumas soluções.

5.2 Arduino em Mac OS X

1 - Placa Arduino e um cabo USB AB

Este tutorial serve para instalação das placas Arduino Uno, Arduino Duemilanove, Nano, Arduino Mega 2560 ou Diecimila. Para outras placas da linha Arduino deve-se buscar o tutorial correspondente. Você também vai precisar de um cabo USB AB.

2 - Download do software do Arduino

Faça download da última versão do software do Arduino. Ao terminar o download clique duas vezes no arquivo .zip para abrir a aplicação Arduino.

3 - Instale o Software

Se você está usando uma placa Arduino Uno ou Mega 2560 não é necessário instalar nenhum software. Caso você esteja usando outra placa Arduino ou um modelo mais antigo talvez seja necessário instalar mais algum driver.

4 - Conectando o Arduino

O Arduino Uno isolado usa a energia do computador através da conexão USB, não sendo necessária energia externa. Conecte a placa Arduino ao computador usando o cabo USB AB. O LED verde de energia (PWR) deve acender.

Se você está usando uma placa Arduino Uno ou Mega 2560 uma janela deve aparecer informando que uma nova interface foi detectada. Clique em "Preferências de Sistema" e clique em "aplicar". O Uno e o Mega 2560 vão aparecer como "não configurados", mas estarão funcionando corretamente. Feche as preferências do sistema.

5 - Abrindo o programa Arduino

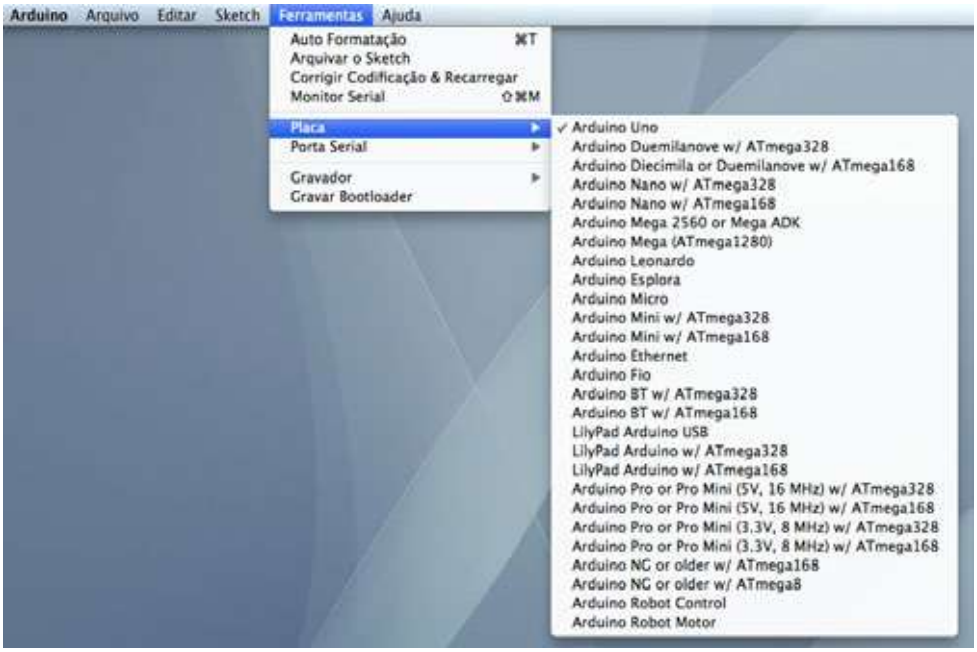
Clique duas vezes na aplicação do Arduino. Caso o programa carregue com o idioma que não é da sua preferência você pode alterar na sessão de preferências do programa.

6 - Exemplo Piscar

Abra o exemplo Piscar (blink): Arquivo > Exemplos > 01.Basics > Blink

7 - Selecione sua placa

Você deve selecionar qual a sua placa Arduino: Ferramentas > Placa > Arduino Uno.



8 - Selecione a porta

Selecione agora a porta serial que conectará o Arduino: Ferramentas > Porta Serial.

Em um Mac, esta porta deve ser algo como **/dev/tty.usbmodem** (para Uno ou Mega 2560) ou **/dev/tty.usbserial** (para placas mais antigas).

Selecionando um Uno, Mega 2560 ou uma placa mais nova:

9 - Carregue o programa

Agora simplesmente clique no botão Carregar da janela do programa. Espere alguns segundos. Você deve ver os LEDs RX e TX da placa piscarem. Se o processo foi executado normalmente você verá uma mensagem de "Transferência concluída".

Depois de alguns segundos você verá o LED do pin 13 piscar em laranja. Neste caso, parabéns! Seu Arduino está pronto e instalado.

Se você tiver problemas na instalação pode acessar a página oficial do Arduino com algumas soluções.

5.3 Arduino em Linux

Será necessário instalar alguns programas para usar Arduino em Linux. A forma do procedimento depende da distribuição.

Primeiro faça download da última versão do Arduino para Linux na página oficial.

Para mais detalhes selecione sua distribuição:

- ArchLinux
- Debian
- Fedora
- Gentoo
- MEPIS
- Mint
- openSUSE
- Puppy
- Pussy
- Slackware
- Ubuntu
- Xandros (Debian derivative) on Asus Eee PC
- CentOS 6

6 Programação

A programação é um grande recurso que nos permite criar diversas sequências de passos lógicos com o objetivo de cumprir nossas necessidades e de nossos sistemas.

Programar é uma arte que requer uma grande habilidade lógica e concentração por parte do programador.

6.1 Conceito de Programação

É o processo de projetar, escrever, provar, depurar e manter o código fonte de programas de computador. O código fonte é escrito em uma linguagem de programação. O propósito da programação é criar programas que executem um comportamento desejado.

O processo de escrever um código requer frequentemente conhecimentos em várias áreas distintas, além do domínio da linguagem a utilizar, algoritmos especializados e lógica formal. Programar engloba áreas como a análise e o projeto da aplicação.

Para criar um programa que o computador interprete e execute as instruções escritas, deve-se usar uma linguagem de programação. No início, os computadores interpretavam somente instruções em uma linguagem específica, uma linguagem de programação de baixo nível conhecida como código máquina, excessivamente complicada para programar. Consiste somente em cadeias de números 1 e 0 (sistema binário).

Para facilitar o trabalho de programação os primeiros cientistas que trabalhavam na área decidiram substituir as instruções, sequências de um e zero, por palavras ou letras do inglês, codificando e criando assim uma linguagem de maior nível conhecida como Assembly. Por exemplo, para somar se usa a letra A, do inglês *add*. Realmente escrever em linguagem assembly é basicamente o mesmo que com a linguagem máquina, mas as letras e as palavras são mais fáceis de lembrar e entender que sequências de números binários.

À medida que a complexidade das tarefas que realizavam os computadores aumentava, foi necessário desenvolver um método mais simples de programação. Então foram criadas as linguagens de alto nível. Enquanto que uma tarefa tão simples como multiplicar dois números necessita um conjunto de instruções em linguagem assembly, em uma linguagem de alto nível basta com uma.

6.2 Linguagem de Programação

Uma linguagem de programação é um idioma artificial desenvolvido para expressar operações que podem ser executadas por máquinas como os computadores. Podem ser usadas para criar programas que controlam o comportamento físico e lógico de uma máquina, para expressar algoritmos com precisão, ou como modo de comunicação entre as pessoas.

Está formada por um conjunto de símbolos e regras sintáticas e semânticas que definem sua estrutura e o significado de seus elementos e expressões.

O processo pela qual se escreve, prova, depura, compila e se mantém o código fonte de um programa informático se chama programação.

6.3 Linguagem de Máquina

Sistema de códigos diretamente interpretável por um circuito microprogramável, como o microprocessador de um computador ou um microcontrolador. Um programa em código de máquina consiste em uma sequência de números que significam uma sequência de instruções a serem executadas.

A linguagem máquina trabalha com dois níveis de voltagem. Tais níveis, por abstração, se simbolizam com o zero (0) e o um (1), por isso a linguagem de máquina só utiliza estes signos.

Os programas de computador raramente são criados em linguagem de máquina, mas devem ser traduzidos (por compiladores) para serem executados diretamente pelo computador. Existe a opção, em voga atualmente, de não executá-los diretamente, mas sim por meio de um interpretador, esse sim rodando diretamente em código de máquina e previamente compilado.

6.4 Linguagem Assembly

É uma linguagem de programação de baixo nível para computadores, microcontroladores e outros circuitos integrados programáveis. A linguagem de máquina, que é um mero padrão de bits, torna-se legível pela substituição dos valores em bruto por símbolos chamados mnemónicos. Estes símbolos são geralmente definidos pelo fabricante do hardware e está baseada em códigos que simbolizam os passos do processamento (as instruções).

Uma linguagem assembly é portanto específica de cada arquitetura de computador, podendo ser usada somente por um microprocessador específico. Isso contrasta com a maioria das linguagens de programação de alto nível que idealmente são portáteis, o que significa que um programa pode ser executado em uma variedade de computadores.

6.5 Linguagem de Alto Nível

Linguagem de programação de alto nível é como se chama, na Ciência da Computação de linguagens de programação, uma linguagem com um nível de abstração relativamente elevado, longe do código de máquina e mais próximo à linguagem humana. Desse modo, as linguagens de alto nível não estão diretamente relacionadas à arquitetura do computador. O programador de uma linguagem de alto nível não precisa conhecer características do processador, como instruções e registradores. Essas características são abstraídas na linguagem de alto nível.

Para estas linguagens é necessário certo conhecimento de programação para realizar sequências de instruções lógicas. As linguagens de alto nível foram criadas para que o usuário comum pudesse solucionar um problema de processamento de dados de uma maneira mais fácil e rápida.

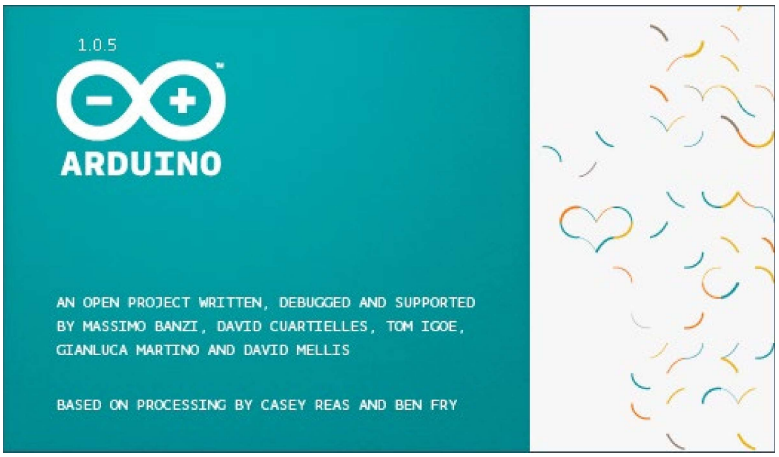
6.6 Algoritmo

Um algoritmo é uma sequência finita de instruções bem definidas e não ambíguas, cada uma das quais pode ser executada mecanicamente num período de tempo finito e com uma quantidade de esforço finita.

O conceito de algoritmo é frequentemente ilustrado pelo exemplo de uma receita culinária, embora muitos algoritmos sejam mais complexos. Eles podem repetir passos (fazer iterações) ou necessitar de decisões (tais como comparações ou lógica) até que a tarefa seja completada. Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa. Sua implementação pode ser feita por um computador, por outro tipo de autômato ou mesmo por um ser humano.

7 Programação Arduino

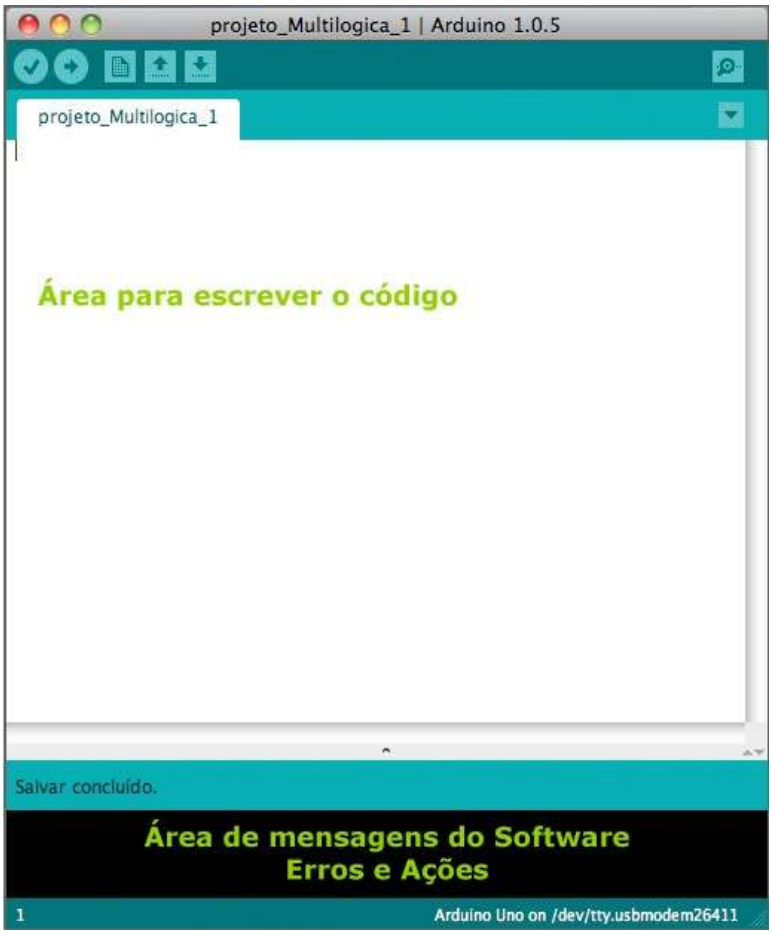
7.1 Software Arduino



Para executar o programa entramos na pasta do Arduino guardada no computador e procuramos o ícone. Clique duas vezes para abrir o programa.

O programa do Arduino também é conhecido como IDE Arduino (*Integrated Development Environment*) pois além do entorno de programação consiste também em um editor de código, um compilador e um depurador.

Espaço de trabalho:



Sketches

Softwares escritos usando Arduino são chamados de Sketches. Estes Sketches são escritos no editor de texto da IDE do Arduino e são salvos com a extensão de arquivo .ino. Este editor tem características de cortar/colar e para buscar/substituir texto. A área de mensagem dá feedback ao salvar e exportar arquivos e também exibe informações de erros ao compilar Sketches. O canto direito inferior da janela exibe a placa atual e a porta serial. Os botões da barra de ferramentas permitem que você verifique, carregue, crie, abra e salve Sketches ou abra o monitor serial.

Nota: Nas versões do IDE antes de 1.0 os Sketches são salvos com a extensão .pde. É possível abrir esses arquivos com a versão 1.0, mas você será solicitado a salvar o Sketch com a extensão .ino.

Verificar
Verifica se seu código tem erros.

Carregar
Compila seu código e carrega para a placa Arduino.

Novo
Cria um novo Sketch.

Abrir
Apresenta um menu de todos os sketches já existentes.

Salvar
Salva seu Sketch.

Monitor Serial
Abre o monitor serial.

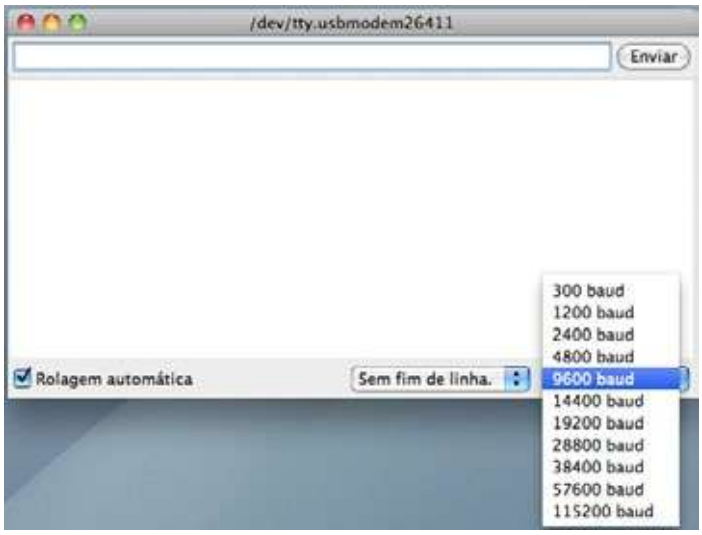
Monitor Serial

Exibe dados seriais sendo enviados da placa Arduino para o computador. Para enviar dados para a placa, digite o texto e clique no botão "enviar" ou pressione enter.

A comunicação entre a placa Arduino e seu computador pode acontecer em várias velocidades padrão pré-definidas. Para que isso ocorra é importante que seja definida a mesma velocidade tanto na Sketch quanto no Monitor Serial.

Na Sketch esta escolha é feita através da função Serial.begin. E no Monitor Serial através do menu drop down do canto inferior direito.

Note que no Mac ou Linux a placa Arduino irá resetar (executar novamente o seu Sketch desde o início), quando você abrir o monitor serial.



A comunicação serial com a placa Arduino também pode ser feita através de outras linguagens de programação como Processing, Flash, Python, MaxMSP e muitas outras.

Biblioteca Arduino

O ambiente Arduino pode ser estendido através da utilização de bibliotecas, assim como a maioria das plataformas de programação. Bibliotecas fornecem funcionalidades extras para uso em sketches. Por exemplo, para trabalhar com hardware ou manipulação de dados.
Algumas bibliotecas já vêm instaladas com a IDE Arduino, mas você também pode fazer download ou criar a sua própria.

Para usar uma biblioteca em um sketch, selecione em sua IDE Arduino:
Sketch> Importar Biblioteca.

Dentro da programação você inclui as funcionalidades de uma biblioteca já existente a partir do comando:

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

7.2 Programando o Arduino

Arduino se programa em uma linguagem de alto nível semelhante a C/C++ e geralmente tem os seguintes componentes para elaborar o algoritmo:

- Estruturas
- Variáveis
- Operadores booleanos, de comparação e aritméticos
- Estrutura de controle
- Funções digitais e analógicas

Para mais detalhes visite a Referência da linguagem de programação Arduino, em português. Veja a referência extendida para características mais avançadas da linguagem Arduino e a página das bibliotecas para interação com tipos específicos de hardware, no site oficial do Arduino.

CC www.arduino.cc

Estruturas

São duas funções principais que deve ter todo programa em Arduino.

A função setup() é chamada quando um programa começa a rodar. Use esta função para inicializar as sua variáveis, os modos dos pinos, declarar o uso de livrarias, etc. Esta função será executada apenas uma vez após a placa Arduino ser ligada ou ressetada.

```
setup(){
}
```

Após criar uma função setup() que declara os valores iniciais, a função loop() faz exatamente o que seu nome sugere, entra em looping (executa sempre o mesmo bloco de código), permitindo ao seu programa fazer mudanças e responder. Use esta função para controlar ativamente a placa Arduino.

```
loop(){
}
```

Variáveis

Variáveis são expressões que você pode usar em programas para armazenar valores como a leitura de um sensor em um pino analógico. Aqui destacamos algumas:

- Variáveis Booleanas

Variáveis boolenas, assim chamadas em homenagem a George Boole, podem ter apenas dois valores: verdadeiro (*true*) e falso (*false*).

```
boolean running = false;
```

- Int

Inteiro é o principal tipo de dado para armazenamento numérico capaz de guardar números de 2 bytes. Isto abrange a faixa de -32.768 a 32.767 (valor mínimo de -2^15 e valor máximo de (2^15) -1).

```
int ledPin = 13;
```

- Char

Um tipo de dado que ocupa 1 byte de memória e armazena o valor de um caractere ASCII. Caracteres literais são escritos entre aspas.

```
char myChar = 'A';
```

Operadores booleanos

Estes operadores podem ser usados dentro da condição em uma sentença **if**.

- && ("e" lógico)

Verdadeiro apenas se os dois operandos forem verdadeiros, ou seja, a primeira condição e a segunda forem verdadeiras. Exemplo:

```
if (digitalRead(2) == 1  && digitalRead(3) == 1) { // ler dois interruptores
  // ...
}
```

é verdadeiro apenas se os dois interruptores estiverem fechados.

- || ("ou" lógico)

Verdadeiro se algum dos operandos for verdadeiro, ou seja, se a primeira ou a segunda condição for verdadeira. Exemplo:

```
if (x > 0 || y > 0) {
  // ...
}
```

é verdadeiro apenas se x ou y forem maiores que 0.

- ! (negação)

Verdadeiro apenas se o operando for falso. Exemplo:

```
if (!x) {
  // ...
}
```

é verdadeiro apenas se x for falso (ou seja, se x for igual a 0).

Operadores de comparação

if, que é usado juntamente com um operador de comparação, verifica quando uma condição é satisfeita, como por exemplo um *input* acima de um determinado valor. O formato para uma verificação **if** é:

```
if (algumaVariavel > 50)
{
  // faça alguma coisa
}
```

O programa checa se algumaVariavel (colocar acentos em nomes de variáveis não é uma boa idéia) é maior que 50. Se for, o programa realiza uma ação específica. Colocado de outra maneira, se a sentença que está dentro dos parêntesis é verdadeira o código que está dentro das chaves roda; caso contrário o programa salta este bloco de código.

As chaves podem ser omitidas após uma sentença **if** se só houver uma única linha de código (definida pelo ponto e vírgula) que será executado de modo condicional:

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

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

if (x > 120) {digitalWrite(LEDpin, HIGH);}   // todos são corretos
```

A sentença que está sendo verificada necessita o uso de pelo menos um dos operadores de comparação:

x == y (x é igual a y)
x != y (x é não igual a y)
x < y (x é menor que y)
x > y (x é maior que y)
x <= y (x é menor ou igual a y)
x >= y (x é maior ou igual a y)

Operadores aritméticos

Se aplicam no uso de variáveis.

= (igualdade)
+ (adição)
- (subtração)
* (multiplicação)
/ (divisão)
% (resto da divisão)

Estruturas de controle

São instruções que permitem decidir e realizar diversas repetições de acordo com alguns parâmetros. Entre os mais importantes podemos destacar:

- Switch/case

Do mesmo modo que as sentenças **if**, as **switch/case** controlam o fluxo dos programas. **Switch/case** permite ao programador construir uma lista de "casos" dentro de um bloco delimitado por chaves. O programa checa cada caso com a variável de teste e executa o código se encontrar um valor idêntico.

```
switch (var) {
    case 1:
      //faça alguma coisa quando var == 1
    case 2:
      //faça alguma coisa quando var == 2
    default:
      // se nenhum valor for idêntico, faça o default
      // default é opcional
  }
```

- While

While fará com que o bloco de código entre chaves se repita contínua e indefinidamente até que a expressão entre parentesis () se torne falsa. Algo tem que provocar uma mudança no valor da variável que está sendo verificada ou o código vai sempre ficar dando voltas dentro do **while**. Isto poderia ser o incremento de uma variável ou uma condição externa, como o teste de um sensor.

```
var = 0;
while(var < 200){
  // algum código que se repete 200 vezes
  var++;
}
```

- For

A sentença **for** é utilizada para repetir um bloco de código delimitado por chaves. Um contador com incremento normalmente é usado para controlar e finalizar o **loop**. A sentença **for** é útil para qualquer operação repetitiva, e é frequentemente usada com *arrays* para operar em conjuntos de dados ou de pinos.

```
// Aumentar o brilho de um LED usando um pino PWM
int PWMpin = 13; // um LED  no pino 13

void setup()
{
// nenhum setup é necessário
}
void loop()
{
  for (int i=0; i <= 255; i++){
    analogWrite(PWMpin, i);
    delay(10);
  }
}
```

Funções digitais

Orientadas a revisar o estado e a configuração das entradas e saídas digitais.

- pinMode()

Configura o pino especificado para que se comporte ou como uma entrada (*input*) ou uma saída (*output*).

Sintaxe:

```
pinMode(pin, mode)
```

```
pinMode(9, OUTPUT);           // determina o pino digital 9 como uma saída.
```

- digitalRead()

Lê o valor de um pino digital especificado, *HIGH* ou *LOW*.

Sintaxe:

```
digitalRead(pin)
```

```
buttonState = digitalRead(9);   // Leitura do estado de um botão no pino 9.
```

- digitalWrite()

Escreve um valor *HIGH* ou um *LOW* em um pino digital.

Sintaxe:

```
digitalWrite(pin, valor)
```

```
digitalWrite(9, HIGH);        // Coloca o pino 9 em estado HIGH.
```

Funções analógicas

Ideais para a leitura ou escrita de valores analógicos.

- analogRead()

Lê o valor de um pino analógico especificado. A placa Arduino contém um conversor analógico-digital de 10 bits com 6 canais. Com isto ele pode mapear voltagens de entrada entre 0 e 5 volts para valores inteiros entre 0 e 1023. Isto permite uma resolução entre leituras de 5 volts / 1024 unidades ou 0,0049 volts (4.9 mV) por unidade.

Sintaxe:

```
analogRead(pin)
```

```
int a = analogRead (A0);      // Lê o valor do pino analógico A0 e armazena
                              //este valor na variável "a".
```

- analogWrite()

Escreve um valor analógico (onda PWM) em um pino. Pode ser usado para acender um LED variando o brilho ou girar um motor a velocidade variável.

Sintaxe:

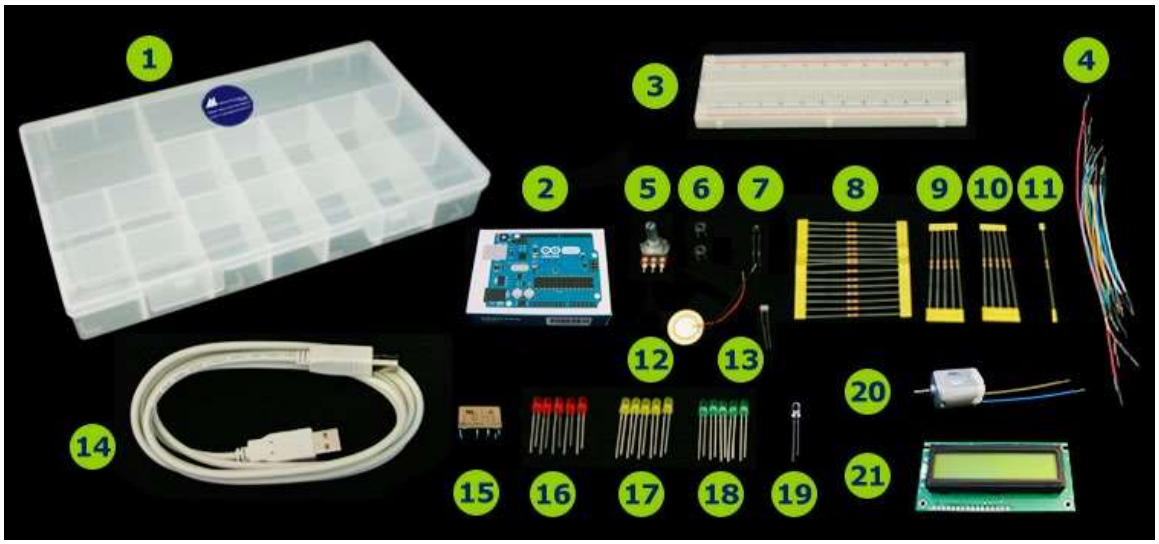
```
analogWrite(pin, valor)
```

```
analogWrite (9,134);          // Envia o valor analógico 134 para o pino 9.
```

8 Kit Arduino Uno R3 - Iniciante

O Kit Arduino Uno R3 - Iniciante desenvolvido pela Multilógica-Shop é o mais completo do Brasil.

Possui tudo o que você precisa para realizar todos os tutoriais deste guia e para começar a desenvolver seus próprios projetos com a plataforma Arduino, sem necessidade de realizar soldas.



1 - 1 Caixa organizadora Multilógica-shop
2 - 1 Arduino Uno R3
3 - 1 Protoboard
4 - 15 jumpers de tamanhos variados
5 - 1 potenciômetro 10kΩ
6 - 2 chaves momentâneas (botão)
7 - 1 Sensor de temperatura (termistor ntc 1k)
8 - 15 resistores 330Ω
9 - 5 resistores 1kΩ
10 - 5 resistores 10kΩ
11 - 1 resistor de 15Ω
12 - 1 sensor/atuador piezoelétrico
13 - 1 Sensor de luminosidade (LDR 5mm)
14 - 1 Cabo USB - Para conectar o Arduino ao seu computador.
15 - 1 Relê de uso geral, bobina de 5V, 40 mA / Dois contatos reversíveis de 1A
16 - 5 LEDs vermelhos (1,2 Vdc 20mA)
17 - 5 LEDs amarelos (1,2 Vdc 20mA)
18 - 5 LEDs verdes (1,2 Vdc 20mA)
19 - 1 LED de alto brilho branco
20 - 1 Motor CC 6V com jumpers soldados
21 - 1 Display LCD 2x16 com conector soldado (com Backlight)

9 Tutoriais

Tutoriais desenvolvidos para utilizar todos os componentes de seu Kit Arduino Uno R3 - Iniciante Multilógica-Shop.

Em cada tutorial você identifica os materiais necessários para sua execução, os conhecimentos prévios necessários e o que você vai aprender, o diagrama de montagem, o código de programação, dicas e exercícios extras.

9.1 Hello World - Piscar

Este exemplo mostra a experiência mais simples que você pode fazer com um Arduino para verificar uma saída física: piscar um LED.

Quando você está aprendendo a programar, na maioria das linguagens de programação, o primeiro código que você escreve diz "Hello World" na tela do computador. Como a placa Arduino não tem uma tela substituiremos esta função fazendo piscar um LED.

O Que Vou Aprender?

- Ativar uma saída digital
- Acender um LED em ON/OFF
- Temporizar um sinal de saída
- Sintaxe de um programa Arduino

Conhecimentos Prévios

- Sinal digital
- Função digitalWrite()
- Polaridade de um LED (página 39)
- Conexão da placa Arduino com o computador

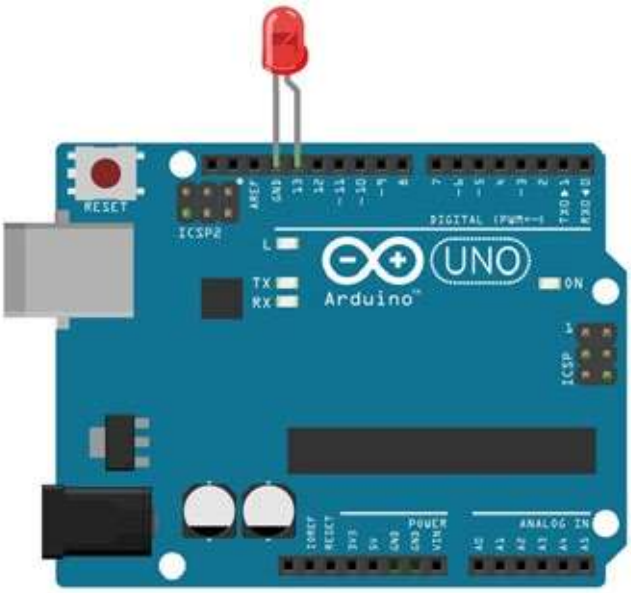
Materiais Necessários

1 Arduino Uno

1 LED

1 Cabo USB AB

Diagrama



Este código já vem junto com a IDE do Arduino. Você pode acessar em:
Arquivo > Exemplos > 01.Basics > Blink
Nós apenas reproduzimos aqui com explicações e os comentários em português.

No programa a seguir, o primeiro comando é o de inicializar o pino 13 como saída através da linha

```
pinMode(13, OUTPUT);
```

No loop principal do código, você liga o LED com esta linha de comando:

```
digitalWrite(13, HIGH);
```

Este comando direciona 5 volts ao pino 13 e o acende. Você desliga o LED com o seguinte comando:

```
digitalWrite(13, LOW);
```

Este comando retira os 5 volts do pino 13, voltando para 0 e desligando o LED. Entre desligar e ligar você precisa de tempo suficiente para que uma pessoa veja a diferença, então o comando delay() informa o Arduino não fazer nada durante 1000 milissegundos, ou um segundo. Quando você usa o comando delay(), nada mais acontece neste período de tempo. Uma vez entendido os exemplos básicos, verifique também o exemplo Piscar sem delay para aprender como criar um delay enquanto faz outras funções.

Código Fonte

```
/*
  Piscar
Acende um LED por um segundo, e depois apaga pelo mesmo tempo, repetidamente.
 */

// Estabeleca um nome para o pino 13:
int led = 13;

// Se executa cada vez que o Arduino inicia:
void setup() {
  // Inicializa o pino digital como saída.
  pinMode(led, OUTPUT);
}

// A funcao loop() continua executando enquanto o Arduino estiver alimentado,
// ou ate que o botao reset seja acionado.

void loop() {
  digitalWrite(led, HIGH);   // Acende o LED
  delay(1000);               // Aguarda um segundo (1s = 1000ms)
  digitalWrite(led, LOW);    // Apaga o LED
  delay(1000);               // Aguarda um segundo (1s = 1000ms)
}
```

Dicas

1 - Na linguagem Arduino // se utiliza para acrescentar comentários na linha de código, sendo muito útil para explicar uma sintaxe ou deixar um lembrete. Um exemplo de seu uso:

```
digitalWrite(13,LOW);       //Apaga o LED
```

2 - Os sinais digitais (Aceso e Apagado) estão presentes em muitos sensores. Conheça alguns deles:

Sensor de movimento infra vermelho

Sensor de distância Sharp GP2D120XJ00F - 4 a 30cm

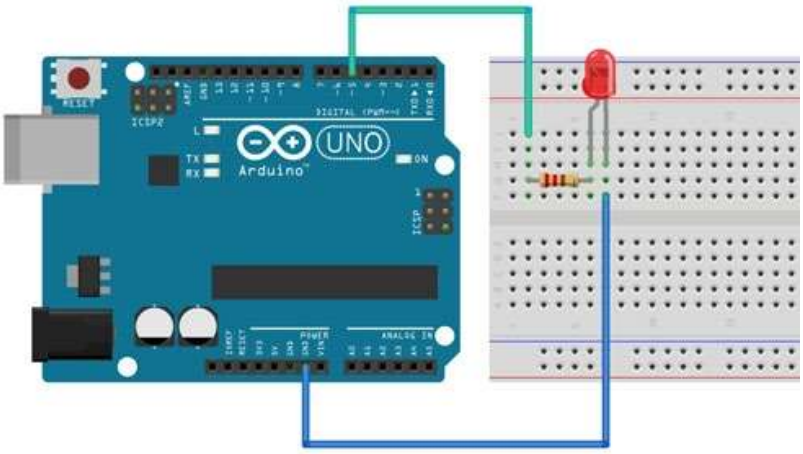
Exercício 1

A partir do código fonte apresentado neste tutorial, faça as modificações necessárias para que o LED fique:

- 3 segundos aceso e 3 segundos apagado
- 200 milissegundos aceso e 500 milissegundos apagado

Exercício 2

A partir do mesmo código fonte faça uma nova montagem deste tutorial e faça as modificações necessárias no código fonte para que o LED seja colocado no Pino 5, e fique 2 segundos aceso e 1 segundo apagado.



Note que para qualquer pino que não seja o 13 é necessário colocar um resistor em série com o LED. Neste caso um resistor de 330Ω é suficiente.

9.2 Botão

O botão é um componente que conecta dois pontos do circuito quando está pressionado. Neste exemplo quando o botão está pressionado o LED se acende.

O Que Vou Aprender?

- Cabear um circuito
- Condicional if/else
- Estado de um botão
- Ler uma entrada digital e escrever uma saída digital

Conhecimentos Prévios

- Sinal digital
- Função digitalWrite() e digitalRead()
- Divisor de voltagem
- Condicional, operadores booleanos e de comparação

Materiais Necessários

1 Arduino Uno

1 Botão

1 LED

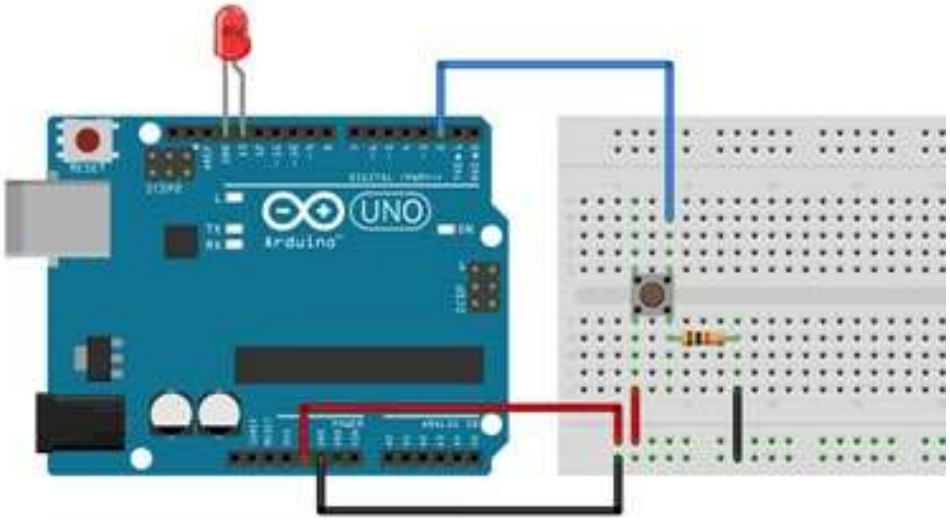
1 Resistor 10kΩ

1 Cabo USB AB

Jumpers

1 Protoboard

Diagrama



Código Fonte

```
/*
  Botao

 Liga e desliga um LED conectado ao pino digital 13 quando pressionado um
botao conectado ao pino 2.

 O Circuito:
 * LED conectado ao pino 13 e ao terra
 * botao conectado ao pino 2 desde 5V
 * resistor de 10K conectado ao pino 2 desde o terra
*/

// constantes nao sao alteradas.
// Sao usadas aqui para definir os numeros dos pinos:
const int buttonPin = 2;     // o numero do pino do botão
const int ledPin =  13;      // o numero do pino do LED

// variaveis que devem mudar:
int buttonState = 0;         // variavel para ler o estado do botao

void setup() {
  // inicializa o pino do LED como saida:
  pinMode(ledPin, OUTPUT);
  // inicializa o pino do botao como entrada:
  pinMode(buttonPin, INPUT);
}

void loop(){
  // faz a leitura do valor do botao:
  buttonState = digitalRead(buttonPin);
```

```
  // verifica se o botao esta pressionado.
  // em caso positivo, buttonState e HIGH:
  if (buttonState == HIGH) {
    // liga o LED:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // desliga o LED:
    digitalWrite(ledPin, LOW);
  }
}
```

Dicas

1 - Quando você está programando com o software do Arduino, muitas das palavras que você escreve são reservadas para a linguagem. Estas palavras se colocam com uma cor diferente, e é uma dica para verificar se estão escritas corretamente. Como no exemplo:

```
void loop()   {
digitalWrite(13,HIGH);
delay(1000);
digitalWrite(13,LOW);
delay(1000);
}
```

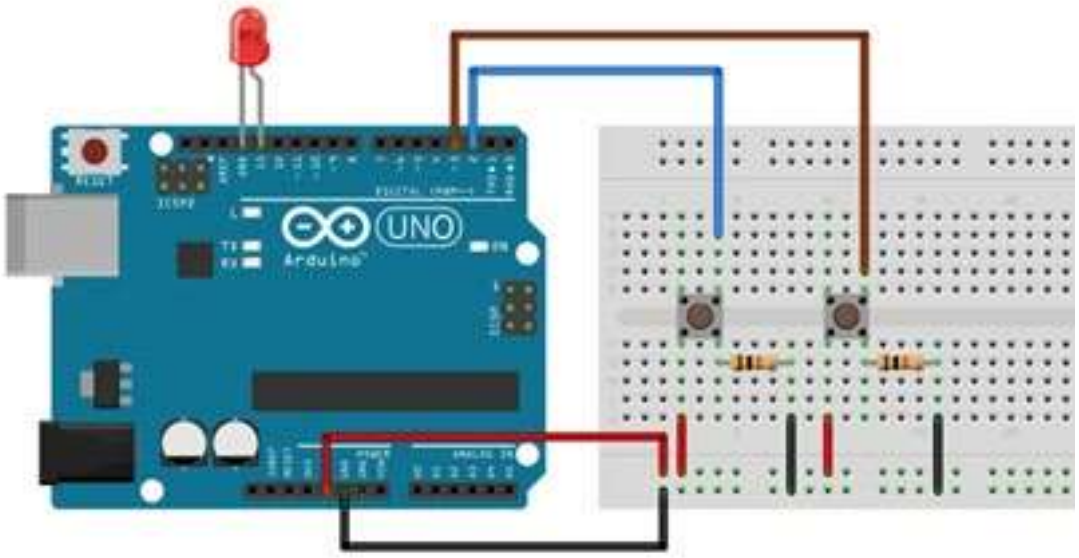
2 - Em um projeto com uso de vários botões com funcionalidades diferentes, pode ser útil trabalhar com peças como estas:

Conjunto de botões tácteis coloridos

Exercício 1

Para evitar acidentes no ambiente de trabalho, uma regra de segurança em vários equipamentos industriais é obrigar que um usuário aperte dois botões, um com cada mão, para acionar uma máquina. É o caso da máquina de corte usada em fábricas de papel.

Com a seguinte montagem podemos simular esta situação. O LED somente acende se os dois botões do circuito estiverem pressionados:



Exercício 2

Faça mais uma modificação no código fonte do exercício 1 para que você possa acender o LED do pino 13 pressionando ou o botão 1 ou o botão 2. Ao deixar de pressionar, o LED se apaga.

9.3 Leitura Serial de uma Entrada Digital

Este exemplo mostra como monitorar o estado de um interruptor estabelecendo a comunicação serial entre seu Arduino e o computador através da USB.

O Que Vou Aprender?

- Controlar uma entrada digital
- Ver dados pelo computador
- Monitor Serial
- Ler uma entrada digital

Conhecimentos Prévios

- Sinal digital
- Função digitalRead() e Serial.print
- Função digitalWrite() e Operadores de comparação

Materiais Necessários

1 Arduino Uno

1 Botão

1 LED

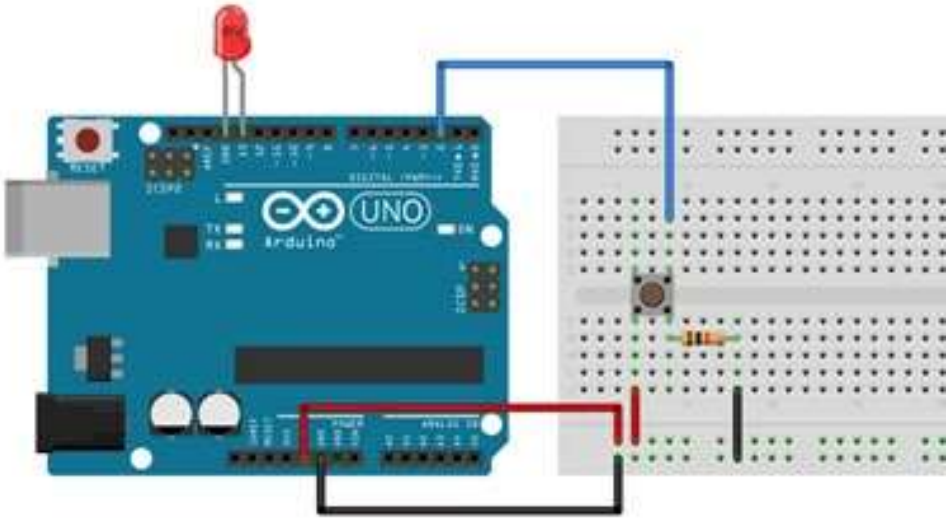
1 Resistor 10kΩ

1 Cabo USB AB

Jumpers

1 Protoboard

Diagrama



Código Fonte

Neste programa a primeira coisa que você vai fazer na função de configuração é começar a comunicação serial a 9600 bits de dados por segundo entre o Arduino e seu computador:

```
Serial.begin (9600);
```

Lembre-se de colocar o valor 9600 também no Monitor Serial como explicado na página 73.

Em seguida, inicializar o pino digital 2, o pino que vai fazer a leitura do botão como uma entrada digital:

```
int pushButton = 2;
```

Quando o botão for pressionado, 5 volts fluirão livremente através de seu circuito, e quando não for pressionado o pino de entrada será ligado ao terra. Esta é uma entrada digital, o que significa que a chave só pode ter um estado (visto pelo seu Arduino como "1", ou HIGH) ou um estado off (visto pelo seu Arduino como um "0", ou LOW), sem nada no meio.

Agora quando você abrir o seu Monitor Serial no ambiente Arduino você verá um fluxo de "0" se a sua chave estiver aberta, ou "1" se a sua chave estiver fechada.

```
/*
  DigitalReadSerial
  Le a entrada digital no pino 2 e imprime o resultado no monitor serial.
  Este exemplo e de dominio publico.
*/

int pushButton = 2;        // o pino 2 tem um botao ligado nele.
int ledPin = 13;           // entrada do LED no pino 13.

void setup() {
  // Inicializa a comunicacao serial a 9600 bits por segundo:
  Serial.begin(9600);

  pinMode(pushButton, INPUT);       // define o botao como uma entrada.
  pinMode(ledPin, OUTPUT);          //define o LED como uma saída.
}

void loop() {
  // faz a leitura do pino de entrada:
  int buttonState = digitalRead(pushButton);
  if (buttonState == 1) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
  // imprime o estado do botao:
  Serial.println(buttonState);
  delay(1);        // delay entre leituras (em milissegundos)
}
```

Dicas

1 - O sistema binário é um sistema de numeração posicional em que todas as quantidades se representam com base em dois números, ou seja, zero e um (0 e 1). Os computadores trabalham internamente com dois níveis de tensão, pelo que o seu sistema de numeração natural é o sistema binário (aceso, apagado).

O sistema binário é base para a álgebra booleana que permite fazer operações lógicas e aritméticas usando-se apenas dois dígitos ou dois estados (sim e não, falso e verdadeiro, tudo ou nada, 1 ou 0, ligado e desligado). Toda a eletrônica digital e computação está baseada nesse sistema binário e na lógica de Boole, que permite representar por circuitos eletrônicos digitais (portas lógicas) os números, caracteres, realizar operações lógicas e aritméticas. Os programas de computadores são codificados sob forma binária e armazenados nas mídias (memórias, discos, etc) sob esse formato.

2 - Para lembrar:

- Para ler um sinal digital use: digitalRead(numeroPin).
- Para escrever um sinal digital use: digitalWrite(numeroPin, valor).
- Uma saída ou entrada digital sempre é HIGH ou LOW.

Exercício 1

Aqui faremos mais um exercício usando a mesma montagem deste Tutorial.

Uma vez que você tenha o botão funcionando, muitas vezes você quer fazer alguma ação com base no número de vezes que o botão for pressionado. Para isso, você precisa saber quando o botão muda de estado de desligado para ligado, e contar quantas vezes essa mudança de estado acontece. Isso é chamado de detecção de mudança de estado. Cada 4 pulsações o LED será ligado.

```
/*
  Contador de pulsos (edge detection)
  criado em 27/09/2005, modificado em 30/08/2011 por Tom Igoe
  Este exemplo e de dominio publico.
  http://arduino.cc/en/Tutorial/ButtonStateChange
*/

// constantes nao sao alteradas:
const int buttonPin = 2;     // o numero do pino do botao
const int ledPin = 13;       // o numero do pino do LED

// variaveis que devem mudar:
int buttonPushCounter = 0;   // contador para o numero de impressoes do botao
int buttonState = 0;         // atual estado do botao
int lastButtonState = 0;     // anterior estado do botao

void setup() {
  pinMode(buttonPin, INPUT);     // inicializa o pino do botao como entrada
  pinMode(ledPin, OUTPUT);       // inicializa o pino digital como saida
  Serial.begin(9600);            // inicializa a comunicacao serial
}

void loop() {
  // faz a leitura do valor do botao:
  buttonState = digitalRead(buttonPin);
```

```
  // compara o estado atual do botao com seu estado anterior
  if (buttonState != lastButtonState) {
    // se o estado do botao foi alterado, incrementar o contador
    if (buttonState == HIGH) {
      buttonPushCounter++;
      Serial.print("numero de pulsos:  ");
      Serial.println(buttonPushCounter);
    }
 }
      // salva o estado atual do botao como ultimo estado para iniciar o
      // proximo loop
  lastButtonState = buttonState;

  // Liga o LED cada 4 pulsacoes checando o modulo de contador de botao
     if (buttonPushCounter % 4 == 0) {
     digitalWrite(ledPin, HIGH);
     }else {
     digitalWrite(ledPin, LOW);
  }
}
```

9.4 Leitura Serial de uma Entrada Analógica

Este exemplo mostra como ler um pino de uma entrada analógica, mapear o resultado para um intervalo de 0 a 255, e usar esse resultado para definir a modulação PWM de um pino de saída para acender e apagar um LED como um dímer.

O Que Vou Aprender?

- Controlar uma entrada analógica
- Ver dados pelo computador
- Múltiplos estados de um potenciômetro
- Ler uma entrada analógica

Conhecimentos Prévios

- Sinal analógica
- Função analogRead() e Serial.print

Materiais Necessários

1 Arduino Uno

1 LED

1 Resistor 330Ω

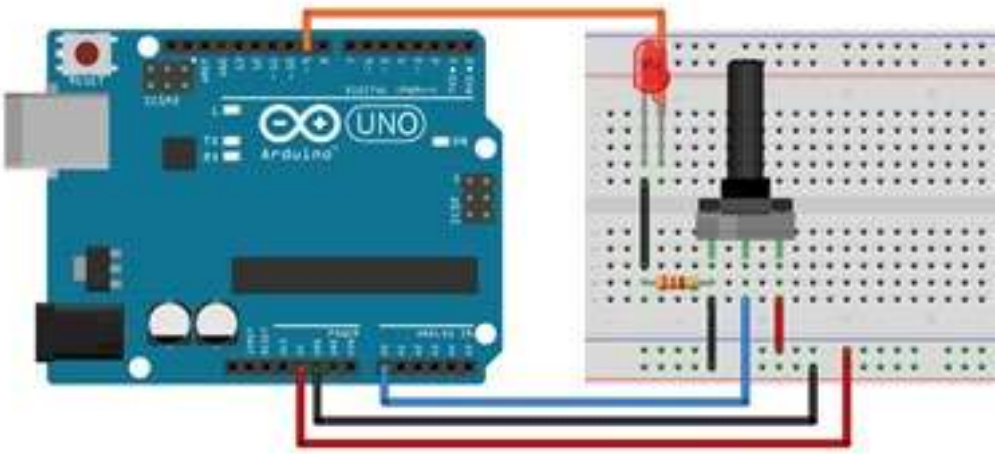
1 Potenciômetro

1 Cabo USB AB

Jumpers

1 Protoboard

Diagrama



Código Fonte

```
/*
  Entrada Analogica, Saida Analogica, Saida serial

Le o pino de entrada analogica, mapeia o resultado para um intervalo entre 0
e 255 e usa o resultado para estabelecer o pulso PWM do pino de saida.
Tambem e possivel acompanhar o resultado atraves do Monitor Serial.

 O circuito:
 - O pino central do Potenciometro conectado ao pino analogico 0. Os pinos
lateriais do potenciometro conectados no terra e 5V.
 - LED conectado no pino digital 9 e no terra.

 Criado em 29/12/2008, Modificado em 09/04/2012 por Tom Igoe
 Este exemplo e de dominio publico.
 */

// constantes nao sao alteradas:
const int analogInPin = A0;  // Entrada analogica do potenciometro
const int analogOutPin = 9;  // Saida analogica onde o LED esta conectado

int sensorValue = 0;        // leitura do potenciometro
int outputValue = 0;        // leitura da saida PWM (analogica)

void setup() {
  // inicializa a comunicacao serial:
  Serial.begin(9600);
}

void loop() {
  // faz a leitura da entrada analogica:
  sensorValue = analogRead(analogInPin);
```

```
  // mapeia o resultado da entrada analogica dentro do intervalo de 0 a 255:
  outputValue = map(sensorValue, 0, 1023, 0, 255);

  // muda o valor da saida analogica:
  analogWrite(analogOutPin, outputValue);

  // imprime o resultado no monitor serial:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // Aguarda 2 milissegundos antes do proximo loop:
  delay(2);
}
```

Dicas

1 - Note que as entradas analógicas do Arduino têm uma resolução de 10 bits (valores de 0 a 1023), mas as saídas analógicas por PWM têm uma resolução de 8 bits (valores de 0 a 255). É por isso que é necessária a função 'map', para "mapear" os valores de modo que se mantenham proporcionais.

2 - Outros elementos que também são potenciômetros:

Potenciômetro de Membrana rotativo Softpot

Potenciômetro de Membrana SoftPot – 50mm

9.5 Comando com Comunicação Serial

Através deste tutorial você vai controlar o acionamento de um relê e de um LED desde o Monitor Serial de seu computador.

O Que Vou Aprender?

- Executar um comando através da Comunicação Serial
- Controlar o acionamento de um relê e de um LED através do computador
- Variável char

Conhecimentos Prévios

- Variáveis booleanas
- Serial.print

Materiais Necessários

1 Arduino Uno

2 LEDs

1 Resistor 330Ω

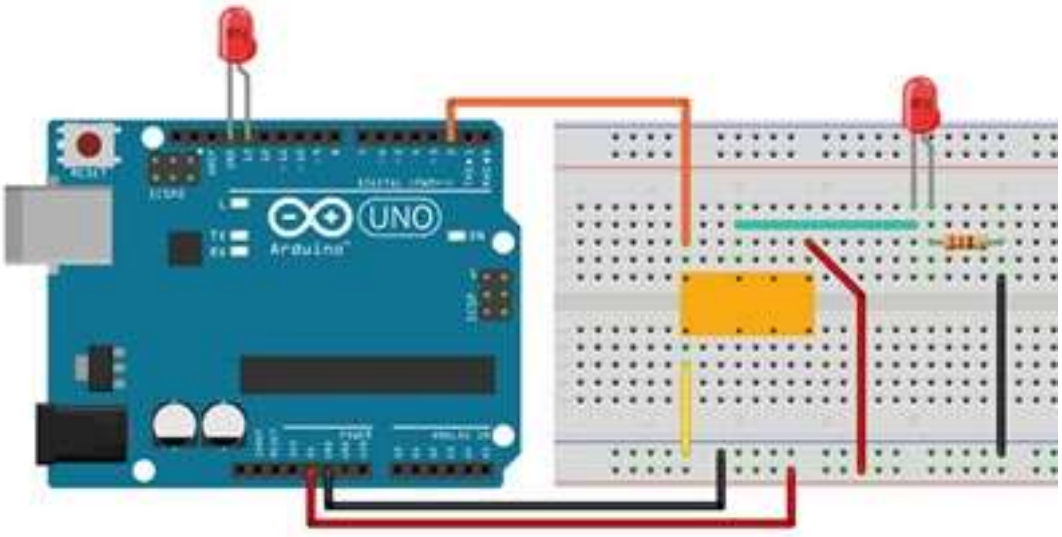
1 Relê

1 Cabo USB AB

Jumpers

1 Protoboard

Diagrama



Código Fonte

```
//************************************************************************
//* Codigo para teste de Arduino acionando rele do kit Multilogica,
//* ligado na saida digital 2 e GND, monitorado pelo Led 13
//* este codigo tem dominio publico
//************************************************************************

//inicializa uma variavel do tipo char que utiliza 1 byte para armazenar
//1 caracter
char input= 0;
int rele=2;
int led=13;
boolean y=true;  // inicializa uma variavel do tipo booleano

void setup() {
  pinMode(rele,OUTPUT);
  pinMode(led,OUTPUT);

  Serial.begin(9600);
  Serial.println();
  Serial.print("**Codigo para acionar rele conectado ao pino 2 do Arduino ");
  Serial.println("atraves do monitor serial**");
  Serial.println("");
  Serial.println("Pressione 1 e depois ENTER para inverter o estado do rele
novamente");
  Serial.println("Aguardando comando :");
}
void loop() {
  if (Serial.available() > 0) {
    input= Serial.read();
```

```
    if (input =='1'){
      Serial.print("O rele agora esta ");

      if(y){
        digitalWrite(rele, HIGH);
        digitalWrite(led, HIGH);
        Serial.println("ligado");
      }
      else {
        digitalWrite(rele, LOW);
        digitalWrite(led, LOW);
        Serial.println("desligado");
      }
      y=!y; // altera o valor de y, se le y e igual a nao y
    }
    else {
      Serial.println("Comando invalido");
    }
  }
}
```

9.6 Fade

Este exemplo demonstra o uso da função analogWrite() para apagar um LED em fade (variação gradual). AnalogWrite usa um pulso PWM, alternando o pino digital on e off rapidamente, criando o efeito de fade.

O Que Vou Aprender?

- Acender e apagar um LED em fade
- Intensificar o conceito de PWM

Conhecimentos Prévios

- PWM
- Função AnalogWrite()
- Polaridade de um LED
- Incrementar e manipular variáveis

Materiais Necessários

1 Arduino Uno

1 LED

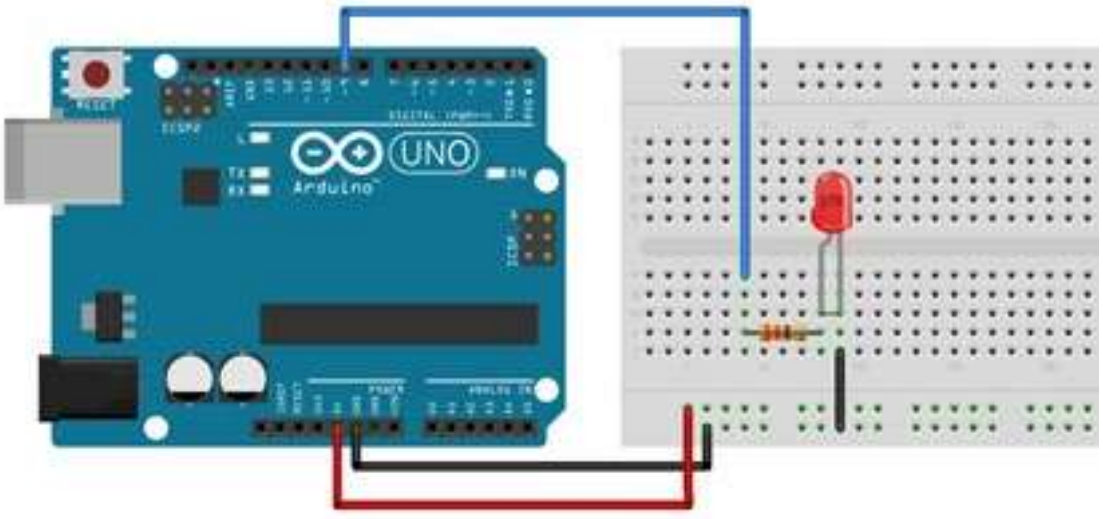
1 Resistor 330Ω

1 Cabo USB AB

Jumpers

1 Protoboard

Diagrama



Código Fonte

Conecte a perna mais longa do LED no pino digital 9 de seu Arduino através de um resistor de 330Ω. Conecte a perna mais curta do LED diretamente ao terra.

Após definir que o pino 9 será seu ledPin, nada mais deverá ser feito na função setup() do código.

A função analogWrite() que você vai usar no loop principal do código requer dois argumentos: um deles informando à função qual pino deve acionar e outra indicando qual valor PWM utilizar.

Para executar o fade no LED, gradualmente aumente o valor PWM de 0 (totalmente desligado) a 255 (totalmente ligado) e depois diminua novamente a 0 para completar o ciclo. No código abaixo, o valor PWM é definido usando uma variável chamada brightness. Cada vez que o loop roda ele aumenta o valor da variável de acordo com o fadeAmount.

Se brightness é definida entre os valores extremos (0 ou 255), então fadeAmount muda para seu negativo. Por exemplo, se fadeAmount é 5, em seguida ele é definido como -5. Se é -5, então seria definido 5. A próxima vez que rodar o loop, esta mudança causa que o incremento de brightness mude também de direção.

analogWrite() pode mudar o valor PWM muito rapidamente, então o delay no final do código controla a velocidade do fade. Tente modificar o valor do delay e veja como isso muda o programa.

```
/*
 Fade

 Este exemplo mostra como executar um fade em um LED no pino 9 usando a
funcao analogWrite().
 Este exemplo e de dominio publico
 */

int led = 9;           // pino do LED
int brightness = 0;    // intensidade do brilho do LED
int fadeAmount = 5;    // em quantos pontos aplicar o fade no LED

void setup()  {
  // define o pino 9 como saida:
  pinMode(led, OUTPUT);
}

// o loop roda em sequencia continuamente:
void loop()  {
  // define o brilho do pino 9:
  analogWrite(led, brightness);

  // muda o brilho para o proximo loop:
  brightness = brightness + fadeAmount;

  // inverte a direcao do fade ao final do mesmo:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // aguarda 30 milissegundos para ver o efeito dimer:
  delay(30);
}
```

Dicas

Outros elementos também utilizam sinais PWM e têm diferentes funções:

Micromotor metálico

Variar a velocidade do giro do motor.

Servomotor

Variar a posição do servo motor em graus.

9.7 Loop

Muitas vezes você deseja repetir uma ação sobre uma série de pinos e fazer alguma coisa diferente para cada um. Neste caso o exemplo faz piscar 6 LEDs usando a função for() loop para fazer circular ida e volta entre os pinos 2 e 7. Os LEDs acendem e apagam em sequência, usando ambos as funções digitalWrite() e delay().

Podemos chamar este exemplo de "Super Máquina" lembrando a série de televisão dos anos 80 em que o famoso ator David Hasselhoff dirigia seu Pontiac com inteligência artificial. O carro foi turbinado com vários LEDs de vários tamanhos possíveis para reproduzir efeitos brilhantes.

Consideramos que seria interessante usar esta metáfora da "Super Máquina" com o objetivo de aprender mais sobre programação sequencial e boas técnicas de programação para as informações de E/S da placa.

O Que Vou Aprender?

- função for() loop
- digitalWrite()
- delay()

Materiais Necessários

1 Arduino Uno

6 LEDs

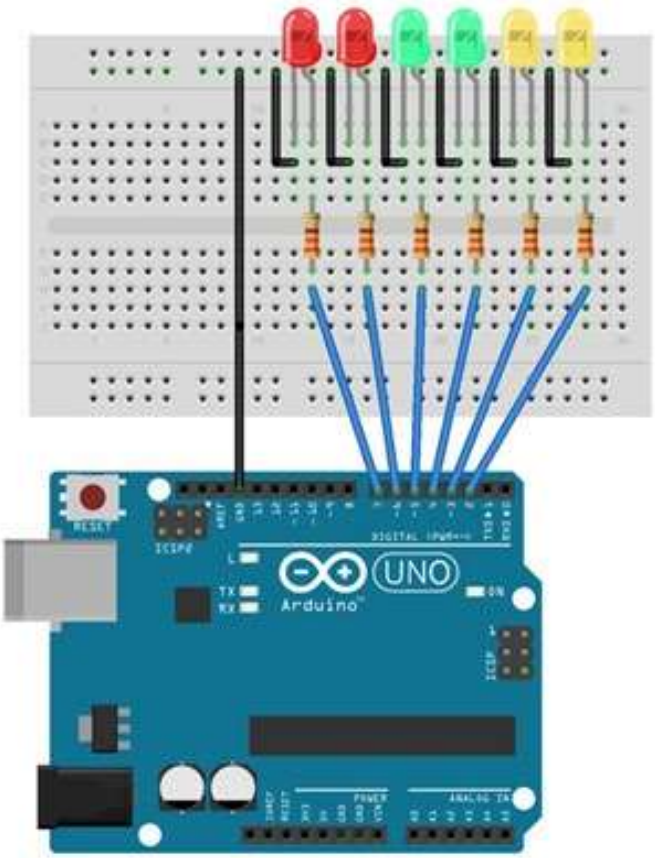
1 Protoboard

6 Resistores 330Ω

1 Cabo USB AB

Jumpers

Diagrama



Código Fonte

O código abaixo começa utilizando a função for() loop para designar os pinos digitais 2 a 7 como saídas dos 6 LEDs usados. No loop principal do código, dois for() loops são usados para incrementar o laço, percorrendo os LEDs, um por um, a partir de pino 2 ao pino 7. Uma vez que o pino 7 está aceso, o processo inverte, percorrendo de volta através de cada LED.

Para mais informações da função for() veja página 82.

```
/*
  Loop

 Demonstra o uso da funcao for() loop.
 Acende varios LEDs em sequencia, e logo ao reves.

 O circuito:
 * LEDs entre os pinos 2 ao 7 e ao terra

 Criado em 2006 por David A. Mellis
 Modificado em 30 de Agosto de 2011 por Tom Igoe

 Este codigo e de dominio publico.
  http://www.arduino.cc/en/Tutorial/ForLoop
 */

int timer = 100;     // Quanto maior o valor, mais lenta a sequencia de Leds.

void setup() {
  // Use for loop para inicializar cada pino como saida:
  for (int thisPin = 2; thisPin < 8; thisPin++)  {
    pinMode(thisPin, OUTPUT);
  }
}
```

```
void loop() {
  // loop desde o pino mais baixo ate o mais alto:
  for (int thisPin = 2; thisPin < 8; thisPin++) {
    // liga este pino:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // desliga este pino:
    digitalWrite(thisPin, LOW);
  }

  // loop desde o pino mais alto ate o mais baixo:
  for (int thisPin = 7; thisPin >= 2; thisPin--) {
    // liga este pino:
    digitalWrite(thisPin, HIGH);
    delay(timer);
    // desliga este pino:
    digitalWrite(thisPin, LOW);
  }
}
```

9.8 Sensor LDR

Neste tutorial vamos usar um LDR (*Light Dependent Resistor*) para simular uma compensação de luz de 5 níveis, ou seja, dependendo se há mais ou menos luz incidindo no sensor o sistema liga ou desliga uma série de LEDs.

Este programa poderia ser usado em um sistema de iluminação com cinco linhas de luz que vão acendendo conforme o sol se põe, compensando progressivamente a deficiência de luz.

Além disso, um potenciômetro ajusta o nível crítico mínimo de luz, a partir do qual se ativará o circuito.

O Que Vou Aprender?

- Leitura serial de um sensor analógico
- Utilização de uma leitura analógica
- pino AREF do Arduino

Conhecimentos Prévios

- Função digitalWrite()
- Condicional if/else

Materiais Necessários

1 Arduino Uno

1 LDR

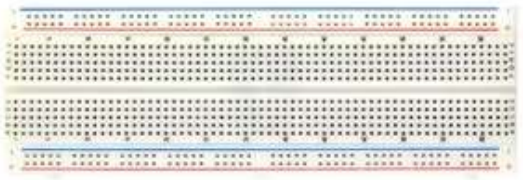
5 LEDs

1 Cabo USB AB

1 Potenciômetro

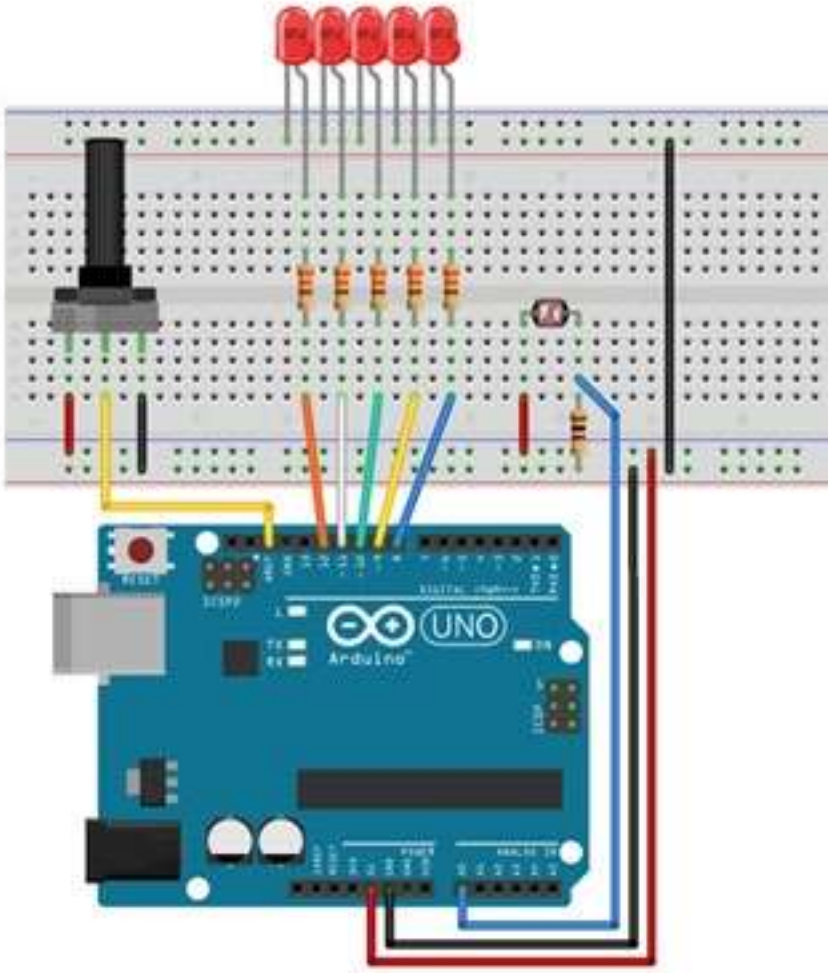
5 Resistores 330Ω

Jumpers



1 Protoboard

Diagrama



Código Fonte

```
/*
 Sensor LDR
 Conectar um LDR a uma entrada analogica para controlar cinco saidas em
funcao da luz ambiente.
 Este codigo e de dominio publico.
 Criado em 27/11/2011 por Arduteka.
 Modificado em 13/01/2014 por Multilogica-Shop.
*/

//Armazenar os dados recolhidos pelo sensor LDR:
int valorLDR = 0;

//Definir os pinos de entrada dos LEDs:
int pinLed1 = 12;
int pinLed2 = 11;
int pinLed3 = 10;
int pinLed4 = 9;
int pinLed5 = 8;

//Definir pino de entrada do sensor LDR
int pinLDR = 0;

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

//Definir os pinos de saida dos LEDs:
  pinMode(pinLed1, OUTPUT);
  pinMode(pinLed2, OUTPUT);
  pinMode(pinLed3, OUTPUT);
```

```
  pinMode(pinLed4, OUTPUT);
  pinMode(pinLed5, OUTPUT);

  //Definimos o uso de uma referencia externa:
  pinMode(EXTERNAL);

}
void loop()
{
  //Guardar o valor da leitura de uma variavel:
  valorLDR = analogRead(pinLDR);
  Serial.println(valorLDR);

  //Definicao do padrao de controle dos LEDs:
  if(valorLDR >= 1023)
  {
    digitalWrite(pinLed1, LOW);
    digitalWrite(pinLed2, LOW);
    digitalWrite(pinLed3, LOW);
    digitalWrite(pinLed4, LOW);
    digitalWrite(pinLed5, LOW);
  }
  else if((valorLDR >= 823) & (valorLDR < 1023))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, LOW);
    digitalWrite(pinLed3, LOW);
    digitalWrite(pinLed4, LOW);
    digitalWrite(pinLed5, LOW);
  }

  else if((valorLDR >= 623) & (valorLDR < 823))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, LOW);
```

```
    digitalWrite(pinLed4, LOW);
    digitalWrite(pinLed5, LOW);
  }
  else if((valorLDR >= 423) & (valorLDR < 623))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, HIGH);
    digitalWrite(pinLed4, LOW);
    digitalWrite(pinLed5, LOW);
  }
  else if((valorLDR >= 223) & (valorLDR < 423))
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, HIGH);
    digitalWrite(pinLed4, HIGH);
    digitalWrite(pinLed5, LOW);
  }
  else
  {
    digitalWrite(pinLed1, HIGH);
    digitalWrite(pinLed2, HIGH);
    digitalWrite(pinLed3, HIGH);
    digitalWrite(pinLed4, HIGH);
    digitalWrite(pinLed5, HIGH);
  }
}
```

Dica

Quando o Arduino recebe um sinal analógico ele o converte para digital em 1024 partes. Esta operação é padrão já que o Arduino pensa que o sinal que vai receber varia entre 0v e 5v o que nos dá um valor para cada parte de aproximadamente 4,88 mV. Mas podemos dizer que não, que realmente o sistema vai funcionar entre 0v e 3v, obtendo assim 1024 partes distribuídas entre 0v e 3v, o que nos dá um valor para cada parte de 2,9 mV, ou seja, uma resolução muito maior. A distribuição destes valores vamos dividir igualmente em nosso programa para fazer uma ativação progressiva das linhas de iluminação.

Se colocarmos a referência muito baixa, os LEDs começam a funcionar com menos luz ambiente que se colocarmos um sinal mais alto, lembre-se:

Mais luz = menor resistência = Vout maior
Menos luz = maior resistência = Vout menor

Este controle será feito via potenciômetro, onde poderemos calibrar o sistema através da luz ambiente.

```
pinMode(EXTERNAL);
```

Com esta instrução estamos dizendo a nosso Arduino que não use a tensão de referência (+5V), mas sim a que vamos aplicar através do pino AREF.

9.9 Termistor

Neste tutorial vamos usar um Termistor (*Temperature Dependent Resistor*) para fazer uma leitura da temperatura.

O resultado, em graus Celsius, veremos através do Monitor Serial da IDE do Arduino.

O Que Vou Aprender?

- Leitura serial de um sensor analógico (Termistor)
- Utilização de uma leitura analógica
- Variável float

Conhecimentos Prévios

- Função analogRead
- Serial.print

Materiais Necessários

1 Arduino Uno

1 Termistor

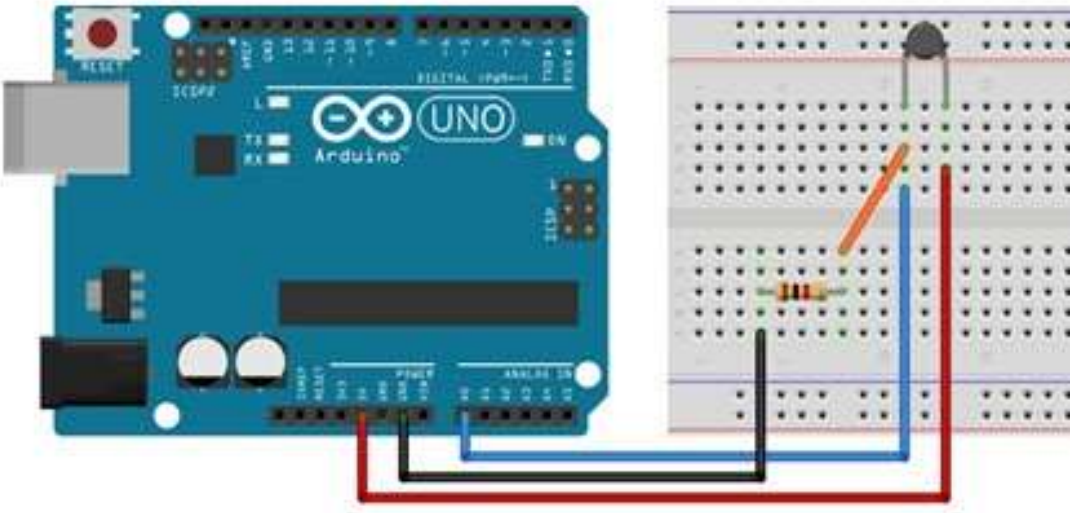
1 Resistor 1KΩ

1 Cabo USB AB

Jumpers

1 Protoboard

Diagrama



Código Fonte

```
/* Codigo para leitura aproximada de temperatura
 utilizando termistor de 1K do kit Multilogica
 Note que nao e um termometro preciso, apenas um exemplo
 aproximado baseado em dados empiricos.
 Ligar resistor 1k de A0 para terra e termistor de +5V para A0 */

#define pino_termistor A0
void setup(void) {
  Serial.begin(9600);
}
void loop(void) {
  float leitura;
  float leitura1;
  leitura = analogRead(pino_termistor);
  Serial.print("Leitura pino A0 = ");
  Serial.println(leitura);
  leitura1 = (leitura*0.2027)-82;
  Serial.print("Temperatura aprox. Celsius = ");
  Serial.println(leitura1);
  Serial.println("");
  delay(2500);
}
```

Dica

Existem basicamente dois tipos de termistores:

- NTC (*Negative Temperature Coefficient*) - termistores cujo coeficiente de variação de resistência com a temperatura é negativo: a resistência diminui com o aumento da temperatura.

- PTC (*Positive Temperature Coefficient*) - termistores cujo coeficiente de variação de resistência com a temperatura é positivo: a resistência aumenta com o aumento da temperatura conforme a curva/tabela característica do termistor, o seu valor de resistência pode diminuir ou aumentar em maior ou menor grau em uma determinada faixa de temperatura.

Assim alguns podem servir de proteção contra sobreaquecimento, limitando a corrente elétrica quando determinada temperatura é ultrapassada. Outra aplicação é a medição de temperatura (em motores por exemplo), pois podemos com o termistor obter uma variação de resistência elétrica em função da variação de temperatura.

O termistor incluído no kit Multilógica é do tipo NTC e obedece esta tabela, cujos dados podem ser utilizados para cálculos e aplicações.

T [°C]	R_Nom [Ω]	R_Min [Ω]	R_Max [Ω]	ΔR/R25 [±%]	ΔT [±°C]
-55	59.147,00	48.631,00	69.664,00	17,80	2,70
-50	42.651,00	35.521,00	49.781,00	16,70	2,60
-45	31.088,00	26.207,00	35.969,00	15,70	2,50
-40	22.903,00	19.530,00	26.276,00	14,70	2,50
-35	17.052,00	14.700,00	19.405,00	13,80	2,40
-30	12.827,00	11.172,00	14.482,00	12,90	2,30
-25	9.746,00	8.572,00	10.920,00	12,00	2,20
-20	7.477,00	6.638,00	8.316,00	11,20	2,20
-15	5.790,00	5.186,00	6.394,00	10,40	2,10
-10	4.523,00	4.086,00	4.961,00	9,70	2,00
-5	3.564,00	3.246,00	3.883,00	8,90	1,90
0	2.832,00	2.599,00	3.065,00	8,20	1,80
5	2.267,00	2.096,00	2.438,00	7,50	1,70
10	1.829,00	1.703,00	1.955,00	6,90	1,60
15	1.486,00	1.393,00	1.578,00	6,30	1,50
20	1.215,00	1.146,00	1.283,00	5,60	1,40
25	**1.000,00**	**950,00**	**1.050,00**	**5,00**	**1,30**
30	828,20	781,60	874,70	5,60	1,50
35	689,90	647,20	732,50	6,20	1,70
40	577,80	539,10	616,60	6,70	1,90
45	486,60	451,40	521,80	7,20	2,10
50	411,80	380,00	443,70	7,70	2,40
55	350,20	321,40	379,00	8,20	2,60
60	299,20	273,20	325,20	8,70	2,80
65	256,70	233,20	280,20	9,20	3,00
70	221,20	200,00	242,40	9,60	3,30
75	191,40	172,20	210,60	10,00	3,50
80	166,20	148,80	183,60	10,50	3,80
85	144,80	129,10	160,60	10,90	4,00
90	126,70	112,40	140,90	11,30	4,30
95	111,20	98,22	124,10	11,70	4,50
100	97,87	86,10	109,60	12,00	4,80
105	86,43	75,72	97,14	12,40	5,00
110	76,55	66,79	86,31	12,80	5,30
115	67,99	59,08	76,90	13,10	5,60
120	60,56	52,42	68,69	13,40	5,90
125	54,07	46,63	61,52	13,80	6,20

9.10 Motor CC

Neste tutorial vamos controlar um motor de corrente contínua através do Arduino.
O acionamento do botão vai ligar nosso motor.

O Que Vou Aprender?

- Leitura digital de um botão
- Controlar um motor de corrente contínua com Arduino

Conhecimentos Prévios

- Função digitalWrite()
- Função digitalRead()
- Condicional if/else

Materiais Necessários

1 Arduino Uno

1 Motor CC

1 Botão

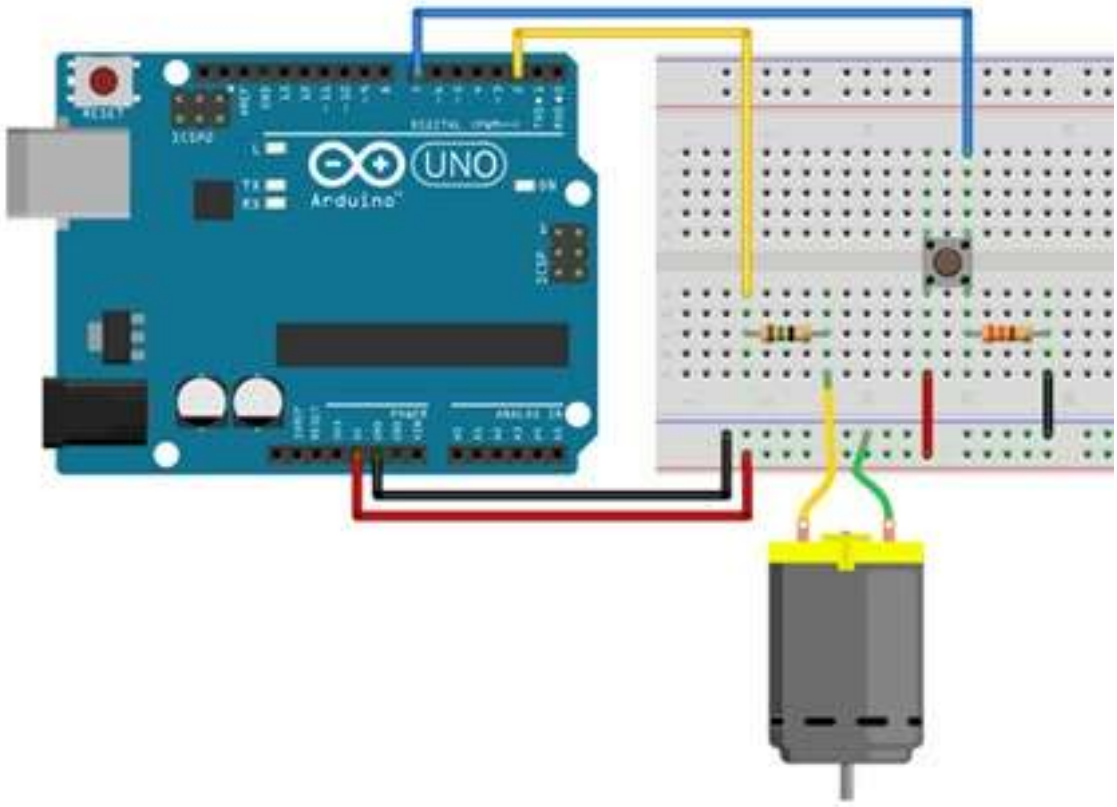
1 Resistor 330Ω
1 Resistor 15Ω

1 Cabo USB AB

Jumpers

1 Protoboard

Diagrama



Código Fonte

```
// Ligar motor no pino 2 em serie com um resistor de 15 ohms
// para limitar a corrente em 40mA para nao sobrecarregar o Arduino

//Este codigo e de dominio publico.
//Criado em 2014 por Multilogica-Shop.

const int motorPin = 2;
const int buttonPin = 7;
int buttonState = 0;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(motorPin, OUTPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    digitalWrite(motorPin, HIGH);
  }
  else {
    digitalWrite(motorPin, LOW);
  }
}
```

Dica

1 - Sentido do Giro

Podemos modificar o sentido do giro de um motor de corrente contínua simplesmente invertendo o sentido da corrente. Com a mesma montagem deste tutorial, teste inverter as conexões do motor e verifique que o mesmo passará a girar no sentido contrário.

9.11 Display LCD

O display de LCD é uma peça importante em projetos em que você precisa visualizar a leitura de um sensor ou mesmo para transmitir uma informação para o usuário.

Neste exercício você aprenderá a conectar o Display LCD 2x16 do seu Kit, que já vem com os pinos soldados.

O Que Vou Aprender?

- Conectar seu display LCD ao Arduino Uno
- Programar frases para aparecer no visor do LCD
- Ajustar o brilho do display com um potenciômetro
- Conhecer as funções da biblioteca LiquidCrystal.h
- Usar as funções:
    - lcd.print
    - lcd.setCursor
    - scrollDisplayLeft()
    - scrollDisplayRight()

Materiais Necessários

1 Arduino Uno

1 Display LCD

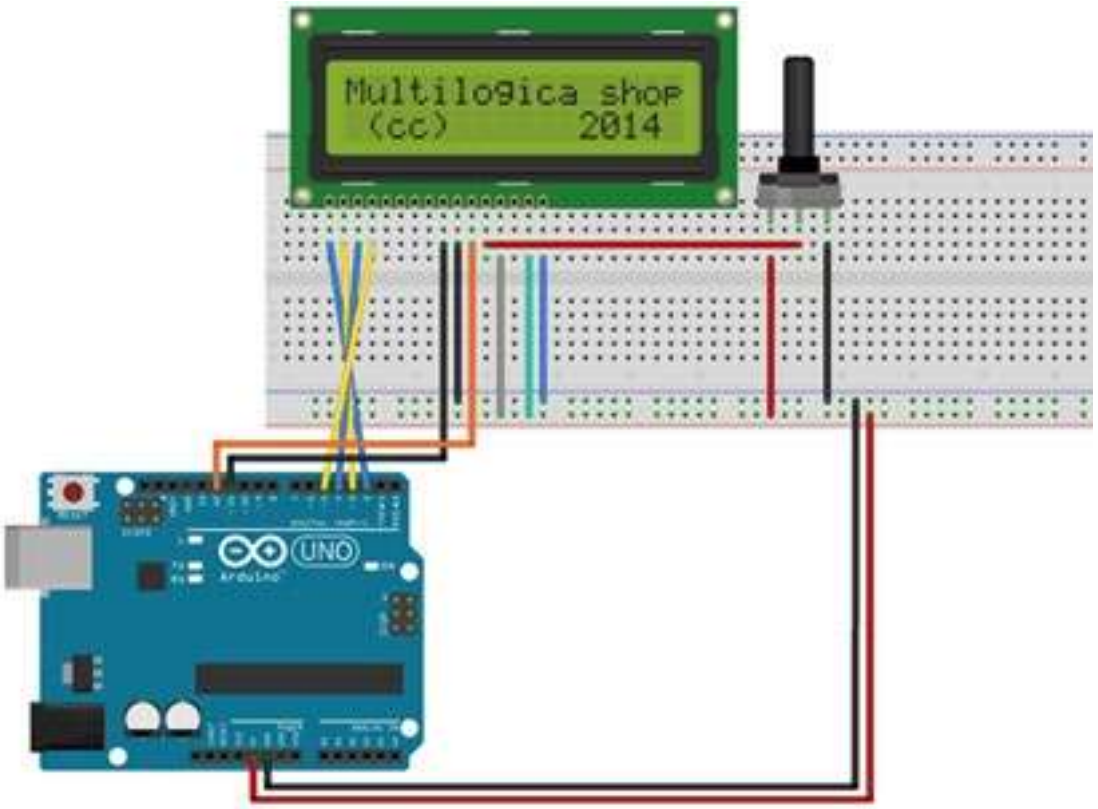
1 Potenciômetro

1 Cabo USB AB

Jumpers

1 Protoboard

Diagrama



Código Fonte

```
/*
 Biblioteca LiquidCrystal codigo Multilogica
 Demonstra o uso do display de 16x2 caracteres
 Esta biblioteca funciona com todos displays compativeis com o
 driver Hitachi HD44780.
 Este codigo escreve :

 Multilogica shop
   (cc)     2014

  Circuito :
 * LCD pino RS no pino digital 12
 * LCD pino Enable no pino digital 11
 * LCD pino D4 pin no pino digital 5
 * LCD pino D5 pin no pino digital 4
 * LCD pino D6 pin no pino digital 3
 * LCD pino D7 pin no pino digital 2
 * LCD pino R/W no terra
 * Trimpot de 10K :
 * +5V no +5V
 * Terra no terra
 * wiper to LCD VO pin (pin 3)

 Codigo de dominio publico baseado no tutorial original :
  http://www.arduino.cc/en/Tutorial/LiquidCrystal
 */

// Inclui o codigo da biblioteca:
#include <LiquidCrystal.h>
```

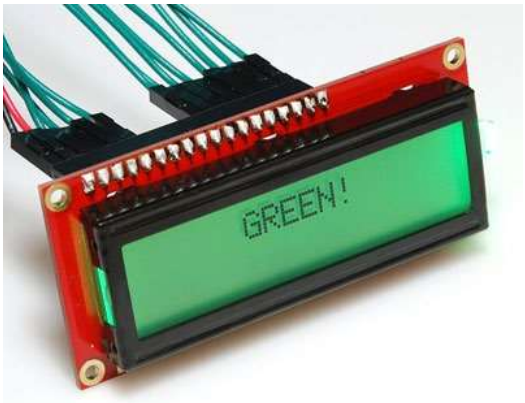
```
// Inicializa a biblioteca e define os pinos utilizados
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // define o numero de colunas e linhas do Display :
  lcd.begin(16, 2);
  // Envia a mensagem para o display.
  lcd.print("Multilogica shop");
  lcd.setCursor(0, 1); //Posiciona o cursor na primeira coluna(0) e na
segunda linha(1) do Display
  lcd.print(" (cc)      2014 ");
}

void loop() {
 }
```

Dicas

Se o seu projeto necessita mais espaço para visualizar informações ou um display de LCD diferenciado, conheça estas outras opções:

Display LCD 2x40 - branco sobre azul

Display LCD 2x16 – fundo RGB



Exercício 1

O site do Arduino oferece vários outros projetos com a Biblioteca LiquidCrystal.h. Aqui faremos mais um exercício usando a mesma montagem deste Tutorial.

Neste exercício você poderá também modificar o texto original e controlar o tempo que seu texto fica fixo e a duração do scroll para a direita ou para a esquerda.

```
/*
  LiquidCrystal Library - scrollDisplayLeft() and scrollDisplayRight()

 Biblioteca LiquidCrystal codigo Multilogica
 Demonstra o uso do display de 16x2 caracteres
 Esta biblioteca funciona com todos displays compativeis com o
 driver Hitachi HD44780

 Este codigo escreve "Multilogica Shop" no LCD e usa
 scrollDisplayLeft() e scrollDisplayRight() para passar o texto.

  Circuito :
* LCD pino RS no pino digital 12
* LCD pino Enable no pino digital 11
* LCD pino D4 pin no pino digital 5
* LCD pino D5 pin no pino digital 4
* LCD pino D6 pin no pino digital 3
* LCD pino D7 pin no pino digital 2
* LCD pino R/W no terra
* Trimpot de 10K :
* +5V no +5V
* Terra no terra
* wiper to LCD VO pin (pin 3)
```

```
 Library originally added 18 Apr 2008 by David A. Mellis
 library modified 5 Jul 2009 by Limor Fried (http://www.ladyada.net)
 example added 9 Jul 2009 by Tom Igoe
 modified 22 Nov 2010 by Tom Igoe

 Codigo de dominio publico baseado no tutorial original:
  http://arduino.cc/en/Tutorial/LiquidCrystalScroll
 */

// Inclui o codigo da biblioteca:
#include <LiquidCrystal.h>

// Inicializa a biblioteca e define os pinos utilizados
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // define o numero de colunas e linhas:
  lcd.begin(16, 2);
  // Envia a mensagem para o display.
  lcd.print("Multilogica Shop");
  delay(2000);
}

void loop() {
  // caminha 16 posicoes para o texto sair do display a esquerda:
  for (int positionCounter = 0; positionCounter < 16; positionCounter++) {
    // caminha uma posicao para a esquerda:
    lcd.scrollDisplayLeft();
    // Aguarda um instante:
    delay(250);
  }

  // caminha 32 posicoes para o texto sair do display a direita:
  for (int positionCounter = 0; positionCounter < 32; positionCounter++) {
    // caminha uma posicao para a direita:
    lcd.scrollDisplayRight();
```

```
    // Aguarda um instante:
    delay(250);
  }

    // caminha 16 posicoes para a esquerda para mover de novo ao centro:
  for (int positionCounter = 0; positionCounter < 16; positionCounter++) {
    // caminha uma posicao para a esquerda:
    lcd.scrollDisplayLeft();
    // Aguarda um instante:
    delay(250);
  }

  // delay no final do full loop:
  delay(2000);

}
```

Open Source Hardware

Versão 1.0

Guia desenvolvido pela Equipe da Multilógica-Shop.

Baseado no "Guia del Arduino" criado pela Tienda de Robótica da Colombia.

www.multilogica-shop.com