Arduino + Brino para a robótica educacional

Gabriel Rodrigues Pacheco, Giulia Fricke Galice, Mateus Berardo de Souza Terra, Rafael Mascarenhas Dal Moro, Victor Rodrigues Pacheco.

Brasília, 2016

**Índice**

1. **Introdução ......................................................................................................**
2. **Começando ....................................................................................................** 
   1. Arduino....................................................................................................
   2. Brino.........................................................................................................
   3. Fritzing....................................................................................................
   4. Pronto para começar...............................................................................
3. **Projetos com Arduino.....................................................................................**
   1. Piscar........................................................................................................
   2. Ligar luz com botão..................................................................................
4. **Introdução à Programação............................................................................**
   1. Variáveis...................................................................................................
   2. Comentários.............................................................................................
   3. Incrementadores.....................................................................................
   4. Instruções e laços de controle.................................................................
   5. Operadores lógicos..................................................................................
   6. Funções ou métodos...............................................................................
5. **Introdução a eletrônica básica.......................................................................**
   1. Unidades...............................................................................................
      1. Correntes Eletricas (I).......................................................................
         1. Corrente Continua (C.C.).........................................................
         2. PWM........................................................................................
         3. Corrente Alternada (A.C.)........................................................
      2. Resistencia.......................................................................................
      3. Tensão(U).........................................................................................
   2. Resistores..............................................................................................
      1. LDR..........................................................................................................
   3. Buzzer...................................................................................................
   4. Interruptores............................................................................................
      1. Relés................................................................................................
   5. Capacitores..............................................................................................
      1. Capacitor de cerâmica........................................................................
      2. Capacitor eletrolítico......................................................................
   6. Diodos.....................................................................................................
      1. LEDs.......................................................................................................
   7. Correntes elétricas..........................................................................................
      1. Corrente Continua (C.C.)......................................................................
      2. PWM................................................................................................
         1. Corrente Alternada (A.C.).......................................................................
6. **Motores..................................................................................................................**
   1. Motores C.C......................................................................................................
   2. Servo motores……………………………………………………….
      1. Rotação limitada…………………………………………………
      2. Rotação continua……………………………………………………
   3. Motores de passo....................................................................................
7. **Baterias...................................................................................................................**
8. **Materiais importantes..................................................................................**
   1. Protoboard....................................................................................................
   2. Jumpers.....................................................................................................
   3. Fonte de alimentação......................................................................................
9. **Tabelas importantes......................................................................................**
   1. Valor de resistores..................................................................................
   2. ASCII.....................................................................................................
   3. LEDs...........................................................................................................
10. **Associações.....................................................................................................**
    1. Resistores.................................................................................................
    2. Capacitores...............................................................................................
    3. Pilhas/Baterias...........................................................................................
11. **Introdução**

A tecnologia está presente em todos os aspectos da vida humana, desde rastreadores de sono e outras tecnologias vestíveis até ferramentas educacionais e os mais tradicionais computadores e smartphones. Dentro desse contexto, o desenvolvimento de habilidades especificas se torna cada vez mais importante, mesmo para quem não trabalha na área de TI, ao ponto de, em 2020, saber uma linguagem de programação será tão fundamental quanto conhecer o inglês.

            Uma linguagem de programação é uma série de instruções, com regras sintáticas e semânticas, que são executadas por um processador. Estas são, em sua maioria, feitas utilizando o inglês como base. Observando a dificuldade de vários jovens e adultos com a língua, nós criamos uma linguagem de programação em português para o Arduino, o Brino, que almeja facilitar o entendimento da lógica de programação, sendo caracterizada por ser intuitiva e de simples compreensão.

O Arduino é uma placa com um micro controlador que pode ser utilizada para prototipagem de sistemas e máquinas de forma fácil e rápida. Não apenas ao hardware, mas também a um conjunto de software se refere o nome. A placa é baseada em um processador Atmel AVR e é feita em hardware livre. Portanto, é possível acessar a página do Arduino e baixar os esquemáticos da placa para montar o seu próprio clone.

Tal plataforma é atualmente utilizada em diversas áreas. Não apenas no ramo da tecnologia e desenvolvimento, mas também por pessoas adeptas do movimento maker (Faça você mesmo, do inglês DIY - Do it yourself), que adotam esta placa pela curva de aprendizagem relativamente pequena. Além de hobbistas, artistas estão aproveitando as possibilidades da placa para construir obras interativas e responsivas.

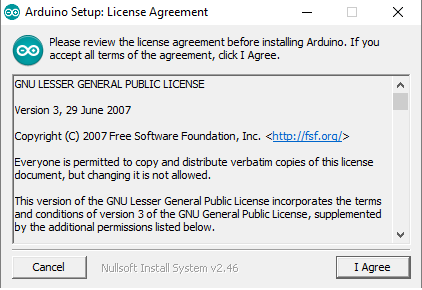
A versatilidade do Arduino fez com que este se tornasse muito popular. Nosso primeiro contato aconteceu no ano de 2014. Desde lá, desenvolvemos diversos projetos: uns voltados para IoT (Internet of things - internet das coisas), alguns jogos e até robôs autônomos. Para aprender, utilizamos alguns livros e muita internet! O próprio fórum do Arduino é um ótimo local para buscar auxílio assim como outros pela WEB. Sites como o [*instructables*](http://instructables.com) também é uma mina de ouro para quem busca um projeto interessante e não sabe por onde começar.

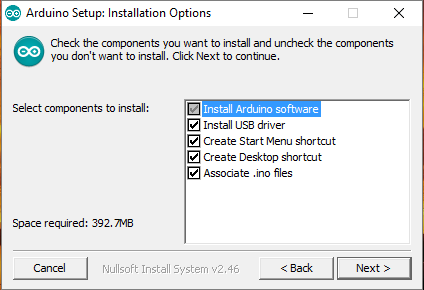
Os projetos disponíveis na internet são programados utilizando a própria linguagem do Arduino. Se você está utilizando o Brino, pode buscar nosso auxílio por e-mail, na página do Facebook ou até utilizar o código nativo do arduino em seu rascunho (o Brino oferece suporte ao código nativo do Arduino). Tudo bem, você já leu uma folha inteira de teoria sobre o que é o Arduino, linguagens de programação, etc.... Se você, como nós, tem um espírito maker, já está de saco cheio de texto e quer começar a desenvolver seu projeto e fazer seu arduino ser útil, então vamos pôr a mão na massa! Antes disso, uma pequena dica: não seja levado completamente por sua impulsividade, aproveite um tempo antes de cada projeto para elaborar um bom planejamento e avaliar as possibilidades, isso evitará que perca tempo e dinheiro.

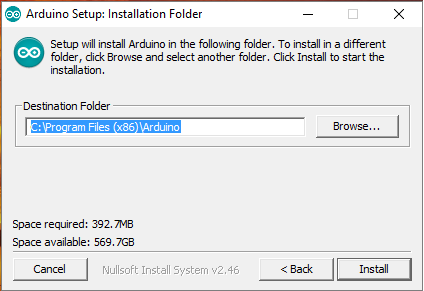
1. **Começando**
   1. **Arduino:**

Para começar você vai precisar, claro, de um Arduino físico ou de uma versão virtual funcional dele, que ensinaremos como instalar no capítulo de instalações. Seja um original ou um clone, a placa que você preferir provavelmente funcionará. Nessa apostila utilizaremos o Arduino nano como referência. Recomendamos que o resto dos componentes seja adquirido quando você começar a desenvolver o projeto do capítulo, uma vez que os comprar todos juntos pode ser custoso. Instruções de confecção de placas de circuito impresso, soldagem e outras habilidades necessárias para o desenvolvimento de alguns projetos, podem ser encontradas no final do livro.

Primeiro, vamos configurar o seu computador para que você possa programar sua placa. Entre no site do Arduino [[*arduino.cc*](http://arduino.cc)] e vá na aba downloads para baixar a IDE mais recente, que na ocasião era a versão 1.6.7. Siga as imagens para completar a instalação:

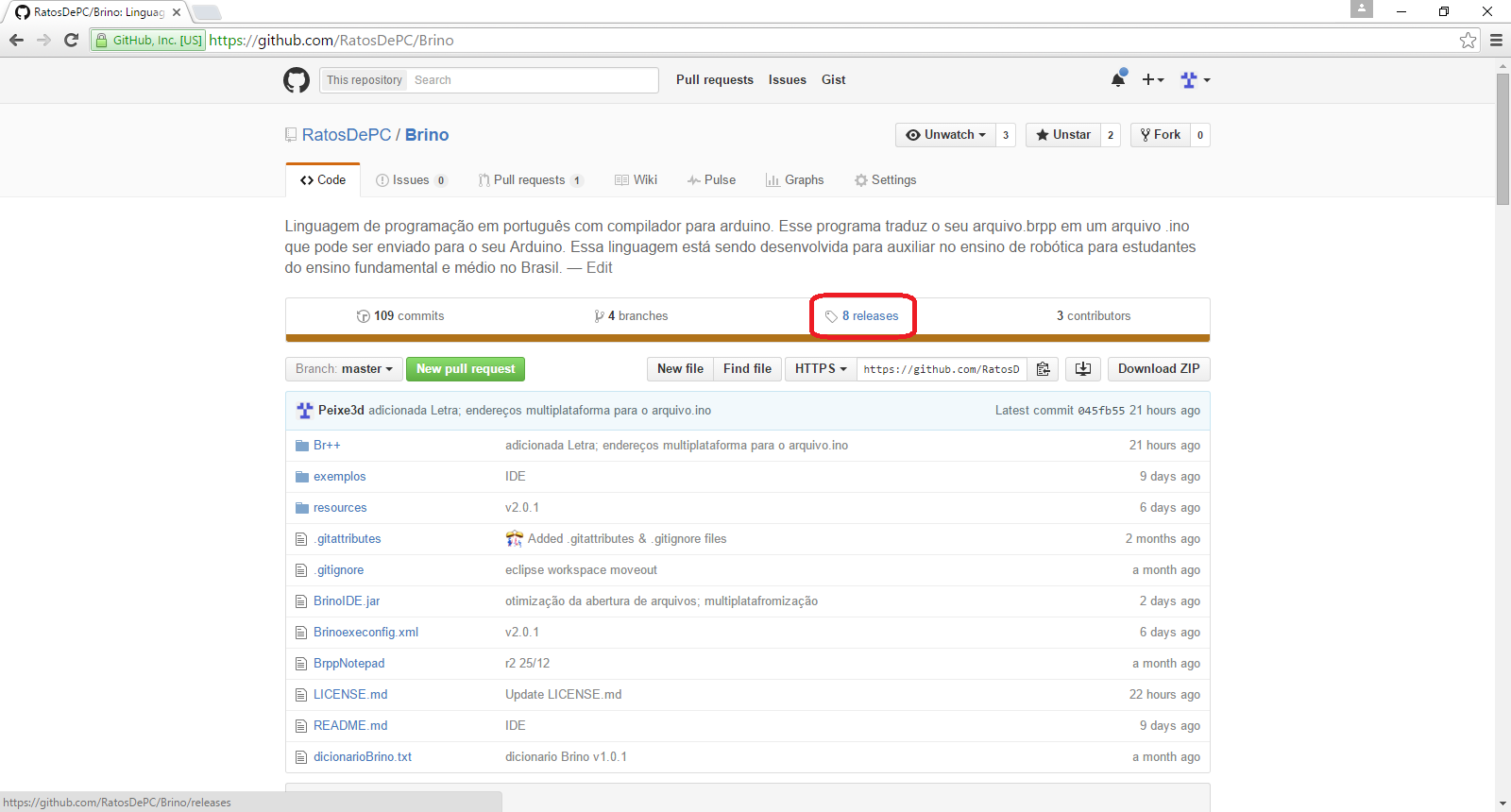
****

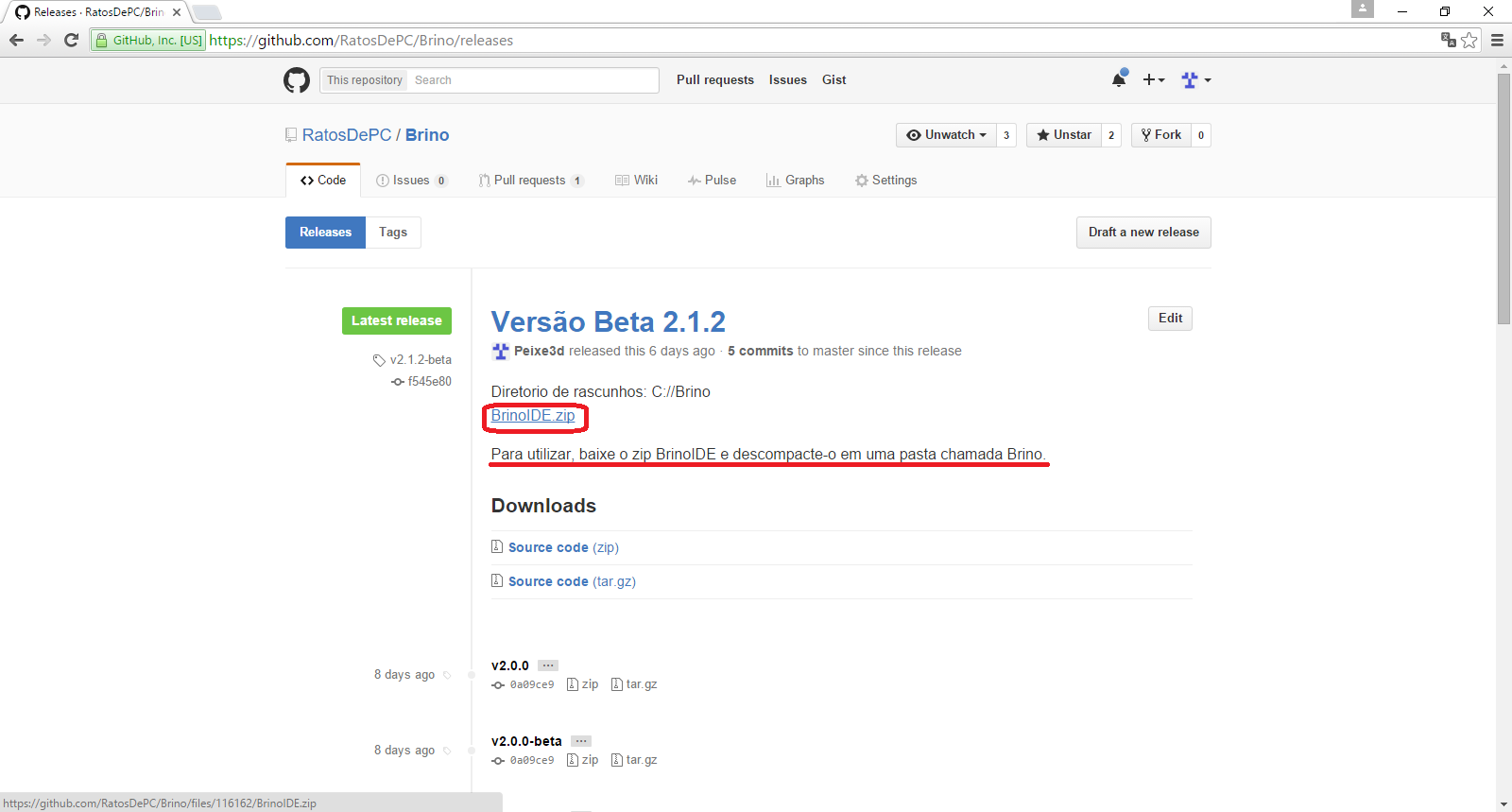
****

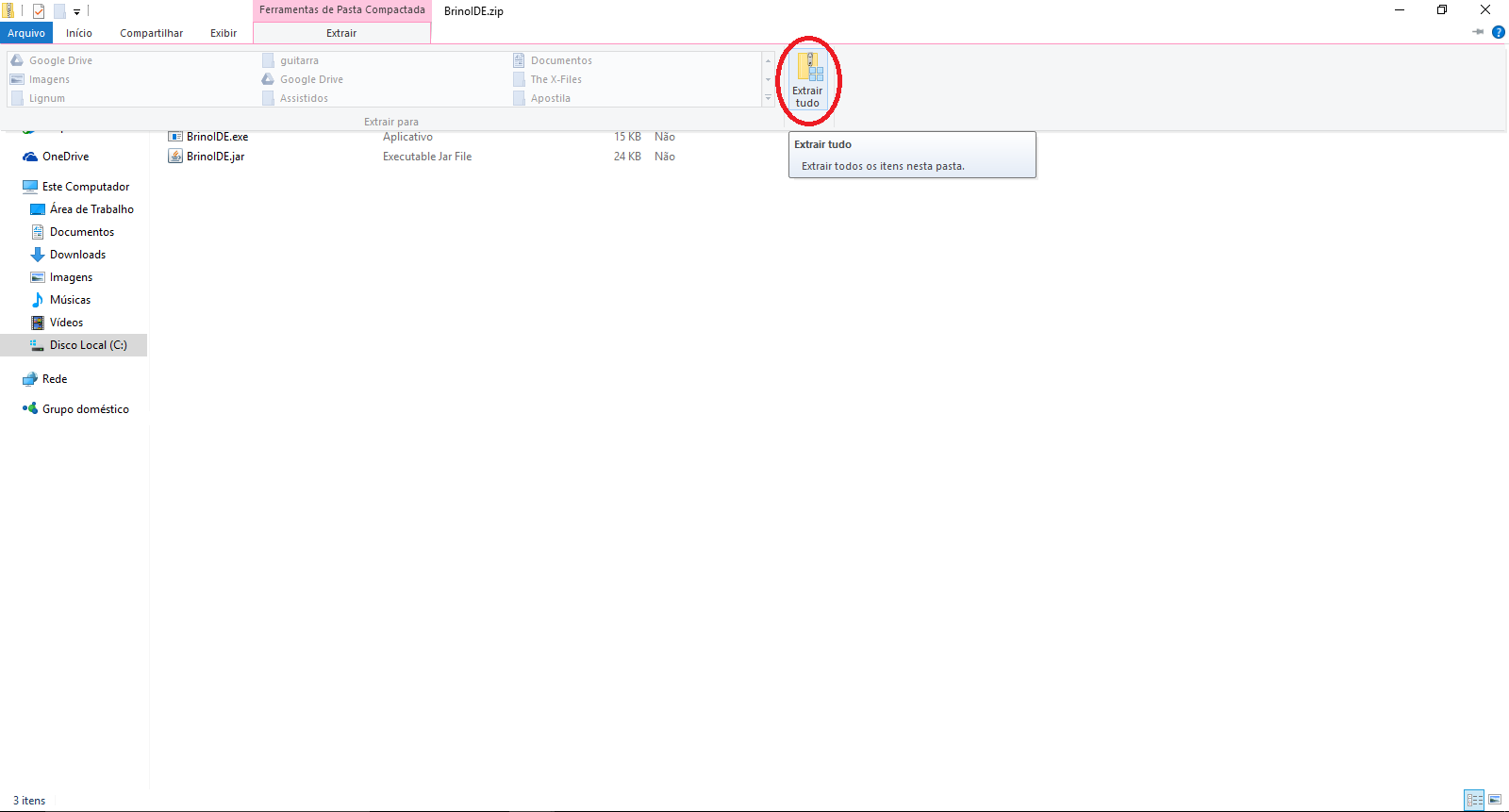
****

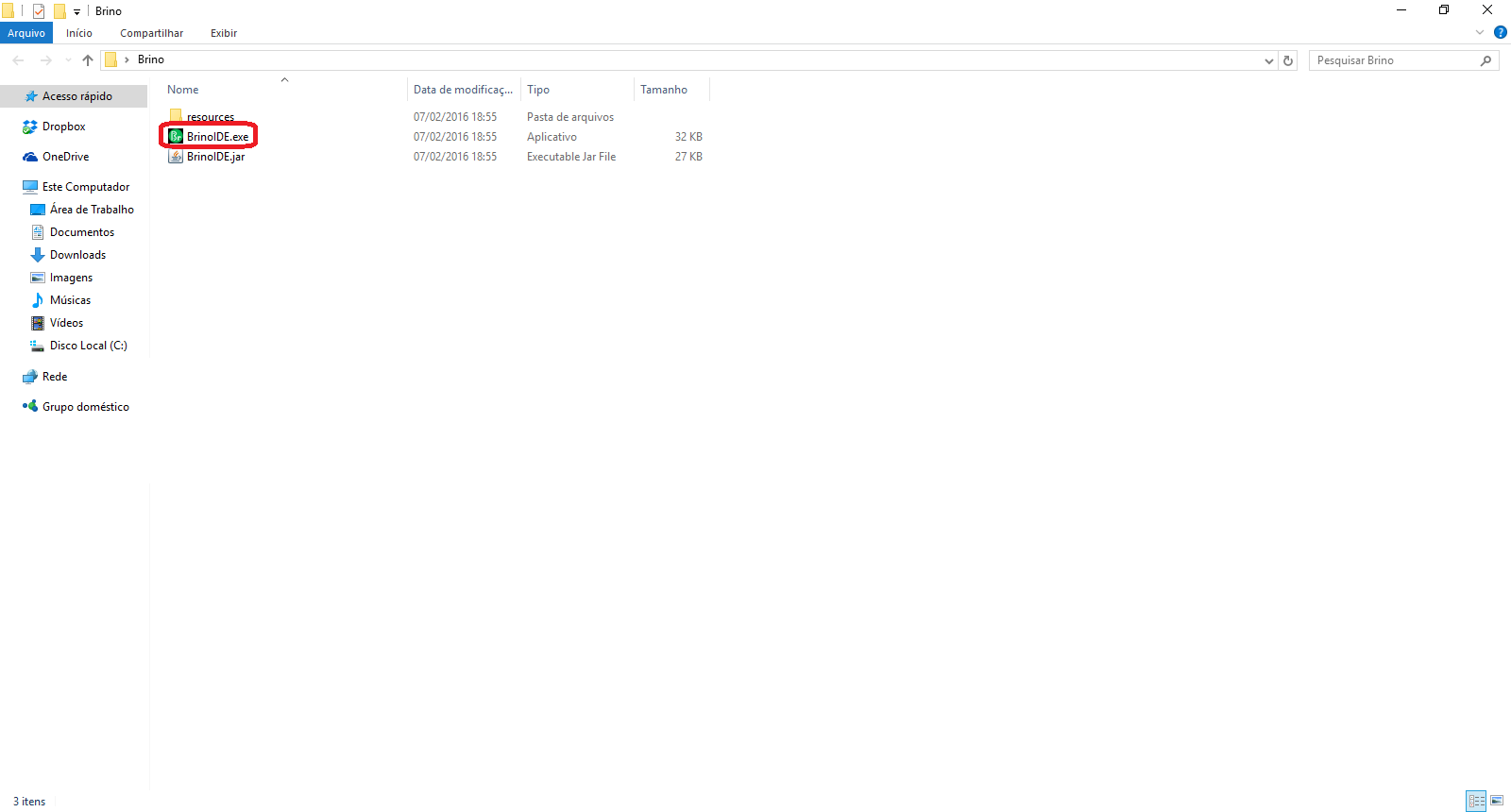
* 1. **Brino:**

Depois de instalar a IDE do Arduino, você está pronto para instalar o Brino. Para isso, basta acessar a página do GitHub: [*http://ratosdepc.github.io/Brino*](http://ratosdepc.github.io/Brino); na área de primeiros passos, você encontra o link para a página de releases e os nossos contatos no final. Já na página de releases, baixe o zip da versão mais recente e descompacte-o em algum lugar (de preferência onde você possa encontrá-lo facilmente). Depois disso, basta abrir o arquivo BrinoIDE.exe e você estará pronto para começar! Caso queira consultar algum código de exemplo, estes podem ser facilmente encontrados na pasta exemplos. Siga as imagens abaixo para completar a instalação.

**Nota:** Quando for compilar algum código, não se esqueça de selecionar a placa com a qual você está trabalhando na aba de ferramentas/placa.







* 1. [](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiYqrnZr_XKAhVDhpAKHXvlDIMQjRwIBw&url=https://learn.sparkfun.com/tutorials/make-your-own-fritzing-parts/what-is-fritzing&bvm=bv.114195076,d.Y2I&psig=AFQjCNHjECna2YIYNje5uEfgklgxRtpfUw&ust=1455474614752199)**Fritzing:**

Fritzing é um programa que busca tornar a eletrônica e a programação acessível a todos. É uma ferramenta intuitiva e rápida para quem deseja documentar, digitalmente, projetos feitos com Arduino. Ele possui diversas opções para a montagem de circuitos sobre uma protoboard virtual, disponibilizando inúmeros modelos de Arduino para o usuário. Além disso, o Fritzing permite exportar esquemáticos, pode ser usado como IDE, entre outras funções.

Para baixar acesse[*http://fritzing.org/home/*](http://fritzing.org/home/) e, na parte superior, clique em downloads e busque pela versão mais recente compatível com o seu sistema operacional. Uma vez obtida a pasta .zip, basta extrai-la para uma pasta convencional. Dentro de tal pasta encontra-se o arquivo fritzing.exe, que é o programa.

* 1. **Pronto para começar:**

Depois de instalar tudo, é hora de desenvolver seu primeiro projeto: “Piscar”. Esse programa é extremamente simples, não necessita de experiência prévia, ou de qualquer outro componente além da sua placa Arduino, pois ela já contém um LED que pode ser controlado. Entretanto, se você nunca programou antes, sugerimos que você leia o capítulo de Introdução à Programação, localizado na área de habilidades adicionais.

**Nota:** àqueles que já possuem alguma experiência com programação, recomendamos ler os tipos de variáveis da linguagem Brino e as funções e métodos obrigatórios em todo rascunho.

1. **Projetos com o Arduino**
   1. **Piscar:**

Neste capítulo trabalharemos o mais simples dos projetos de robótica: piscar um LED (como se o título não fosse autoexplicativo o suficiente...). Mesmo que rudimentar, tal projeto é de grande importância para entender como funcionam os tipos de saída do Arduino e como utilizar a IDE do Brino.

Esse experimento pode ser feito tanto com o LED interno do Arduino, como também com um LED externo. Demonstraremos, inicialmente, como fazê-lo utilizando o LED interno.

**O Código**

Abra a IDE do Arduino e digite o código a seguir:

|  |
| --- |
| // Projeto 1 – Piscar  Numero pinoLed = 13;  Configuracao(){  Pino.definirModo(pinoLed, Saida);  }  Principal(){  Pino.ligar(Digital.pinoLed);  esperar(2000);  Pino.desligar(Digital.pinoLed);  esperar(2000);  } |

Depois, conecte seu Arduino ao computador e verifique a porta serial que ele está conectado. Abra as ferramentas do IDE e selecione a porta e a placa que você está utilizando. Uma vez selecionado, clique em Verificar e Carregar. Se uma janela aparecer solicitando o nome do rascunho, insira “Piscar”, clique em “OK” e espere até o log do IDE (área embaixo do editor) mostrar que o rascunho foi compilado e carregado. Caso ocorra qualquer erro, verifique o seu código, a placa e a porta serial selecionadas. Depois de carregado, observe o LED da sua placa acender por dois segundos e depois desligar também por dois segundos.

**Analisando o código**

A primeira linha do código é:

*//Projeto 1 - Piscar*

Essa linha é apenas um comentário com o nome do projeto.

A linha seguinte é:

*Numero pinoLed = 13;*

Essa linha cria uma variável do tipo Número com o nome pinoLed para armazenar o número do pino em que conectamos o LED.

Logo depois, declaramos a função Configuraçao():

*Configuracao(){*

*Pino.definirModo(pinoLed, Saida);*

*}*

A função de configuração é um dos métodos obrigatórios a todo rascunho Brino e é responsável por preparar o necessário para a execução da função Principal(). Neste caso, ela possui apenas uma instrução em seu bloco de código:

*Pino.definirModo(pinoLed, Saida);*

Essa linha define o modo do *pinoLed*, que possui o valor 13, como saída, ou seja, o Arduino irá emitir uma corrente elétrica de 0 ou 5 V. O método *definirModo* do conjunto *Pino* tem como argumentos o número do pino e o modo, que pode ser *Entrada* ou, como neste caso, *Saida*. Depois de executar a configuração o Arduino inicia o método *Principal()* que nesse rascunho é:

*Principal(){*

*Pino.ligar(Digital.pinoLed);*

*esperar(2000);*

*Pino.desligar(Digital.pinoLed);*

*esperar(2000);*

*}*

A função principal é o segundo método obrigatório a todo rascunho Brino e será repetido enquanto o Arduino estiver ligado. Sua primeira linha é:

*Pino.ligar(Digital.pinoLed);*

O método *ligar* do conjunto *Pino* liga o pino fornecido como argumento, no caso *Digital.pinoLed.* A indicação “*Digital.*” antes de nossa variável avisa para o Arduino que estamos utilizando um pino digital. Quando ligamos uma porta enviamos 5 volts para ela e quando desligamos enviamos 0 volts. A linha seguinte é:

*esperar(2000);*

A função esperar é um método Arduino que faz uma “pausa” na execução do código durante o número de milissegundos indicados entre os parênteses, no caso 2000, que equivale a 2 segundos. Depois de esperar o tempo definido o Arduino executa a próxima linha:

*Pino.desligar(Digital.pinoLed);*

Se a função *ligar* fazia com que o pino ligasse, o método *desligar* faz o contrário, desliga o pino. Ou seja, essa linha irá enviar 0V pelo pino digital *pinoLed* fazendo o LED apagar. Depois disso o código apresenta outra espera de dois segundos e reinicia o método *Principal()*, repetindo-o enquanto o Arduino permanecer ligado.

Agora, antes de demonstrar esse projeto utilizando um LED externo, listamos os materiais necessários:

* Protoboard
* LED
* Resistor de 470 ohms (amarelo, lilás, marrom)
* Jumpers

**Montando o hardware:**

Primeiramente, reproduza o circuito abaixo, se necessário consulte também o diagrama ou esquemático das ligações:





/

**Analisando o hardware**

O hardware montado para esse projeto é bem simples. Conectamos à porta 13 do Arduino um resistor de 470 Ω e, ao resistor, um LED que tem sua outra perna conectada ao ground do circuito. O resistor é utilizado para provocar uma queda de corrente e tensão no circuito, pois o Arduino envia, em cada porta, quando ligada, 5V de tensão e 40 mA de corrente e seu LED, geralmente utiliza uma corrente de 20 mA e uma tensão abaixo dos 3V. Para calcular o melhor resistor para seu circuito, deve-se buscar o datasheet do LED e verificar a corrente e tensão nominais dele. Com esses dados em mãos, aplica-se a seguinte fórmula:

Em que R é o valor da resistência que deve ser usada; Vs é a voltagem fornecida (no caso os 5V do Arduino); Vl é a voltagem nominal do LED; e i é a corrente nominal do LED.

**Nota:** O datasheet é um documento que contém todas as informações técnicas de um determinado componente

* 1. **Ligar luz com botão**

Neste projeto vamos utilizar um botão para controlar o LED do último projeto. Durante o desenvolvimento desse projeto vamos aprender como utilizar entradas digitais no Arduino com resistores *pull-down* externos e *pull-up* internos. As entradas digitais são aquelas que possuem apenas dois valores, Ligada ou Desligada. Para desenvolver esse projeto você precisará de:

* Protoboard
* LED
* Resistor de 470 ohms
* Resistor de 1K ohm
* Botão (Interruptor tátil)

**Montando o hardware**

Com o seu Arduino desconectado, monte o circuito mostrado abaixo:

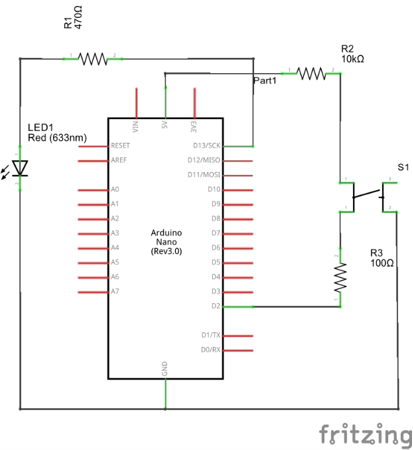
**Nota:** nunca monte seus circuitos com o seu Arduino conectado/ligado.





**Analisando o hardware**

O hardware desse projeto é muito parecido com o do primeiro, com a adição do botão na porta digital 2 do Arduino. Para montá-lo, utilizamos um resistor *pull-down* de 1k ohm. Os/ resistores *pull-down* e *pull-up* são utilizados como filtros para as portas digitais do Arduino, para garantir que não leiam valores aleatórios. A diferença entre eles está no estado que eles mantêm na porta do Arduino. Resistores *pull-down* mantém valores baixos ou desligados enquanto o botão não for pressionado, pois conectam o ground (0V) à porta digital. Já os resistores *pull-up* fariam uma inversão dos estados: manteriam a porta alta, ou ligada (5V), enquanto o botão não estivesse pressionado. Abaixo mostramos uma imagem da montagem de resistores *pull-up* no circuito do projeto 2*.*



**O Código**

Abra a IDE do Brino e digite o código abaixo ou baixe-o da página do github da apostila.

|  |
| --- |
| //Projeto 2 – Ligar luz com botão  Constante Numero pinoLed = 13;  Constante Numero botao = 2;  Configuracao(){  Pino.definirModo(pinoLed, Saida);  Pino.definirModo(botao, Entrada);  }  Principal(){  Numero estadoBotao = Pino.ler(Digital.botao);  se (estadoBotao == Ligado){  Pino.escrever(Digital.pinoLed, Ligado);  esperar(1000);  }  Pino.escrever(Digital.pinoLed, Desligado);  } |

**Analisando o código**

A partir desse capítulo destacaremos apenas métodos e palavras-chaves não descritas anteriormente.

*Constante Numero pinoLed = 13;*

Diferente do outro código, dessa vez marcamos a variável *pinoLed* como uma constante, ou seja, avisamos para o Arduino que seu valor não será alterado durante a execução do rascunho.

*Pino.definirModo(botão,* ***Entrada****);*

Nessa linha definimos que o nosso botão atuará como uma entrada e não como uma saída, dessa forma o Arduino poderá ler o valor da porta e saber se o botão está apertado ou não.

*Numero estadoBotao = Pino.ler(Digital.botao);*

A partir do método *ler* do conjunto *Pino* podemos obter o valor da porta digital *botao* que poderá ser *Ligado*, caso exista corrente fluindo, ou *Desligado*, caso não exista corrente fluindo. Depois de ler a porta, guardamos o valor na variável *estadoBotao*.

*se (estadoBotao == Ligado){*

*...*

*}*

Depois de ler a porta, criamos um bloco condicional que será executado *se* a porta estiver ligada, ou seja, se o botão estiver apertado. Caso ele esteja solto, o Arduino simplesmente ignorará todo o código dentro do bloco do *se*.

*Pino.escrever(Digital.pinoLed, Ligado);*

O método *escrever* nada mais é do que outra forma de ligar e desligar o pino. No primeiro projeto utilizamos os métodos *ligar* e *desligar*, que equivalem a escrever o estado *Ligado* ou *Desligado*, respectivamente, ao pino. Os métodos *ligar* e *desligar* são abstrações, ou seja, simplificações, do método *escrever*.

**Pull-up**

O arduino já possui internamente resistores pull-up. É possível utilizar o modo *Entrada\_PULLUP* para aproveitar estes resistores. A montagem e o código ficariam da seguinte forma:



|  |
| --- |
| //Projeto 2.1 – Ligar luz com botão pull-up  Constante Numero pinoLed = 13;  Constante Numero botao = 2;  Configuracao(){  Pino.definirModo(pinoLed, Saida);  Pino.definirModo(botao, Entrada\_PULLUP);  }  Principal(){  Numero estadoBotao = Pino.ler(Digital.botao);  se (estadoBotao == Desligado){  Pino.escrever(Digital.pinoLed, Ligado);  esperar(1000);  }  Pino.escrever(Digital.pinoLed, Desligado);  } |

Como você pode perceber a leitura do botão fica invertida, logo, quando ele é pressionado, a leitura será igual a 0V ou desligado.

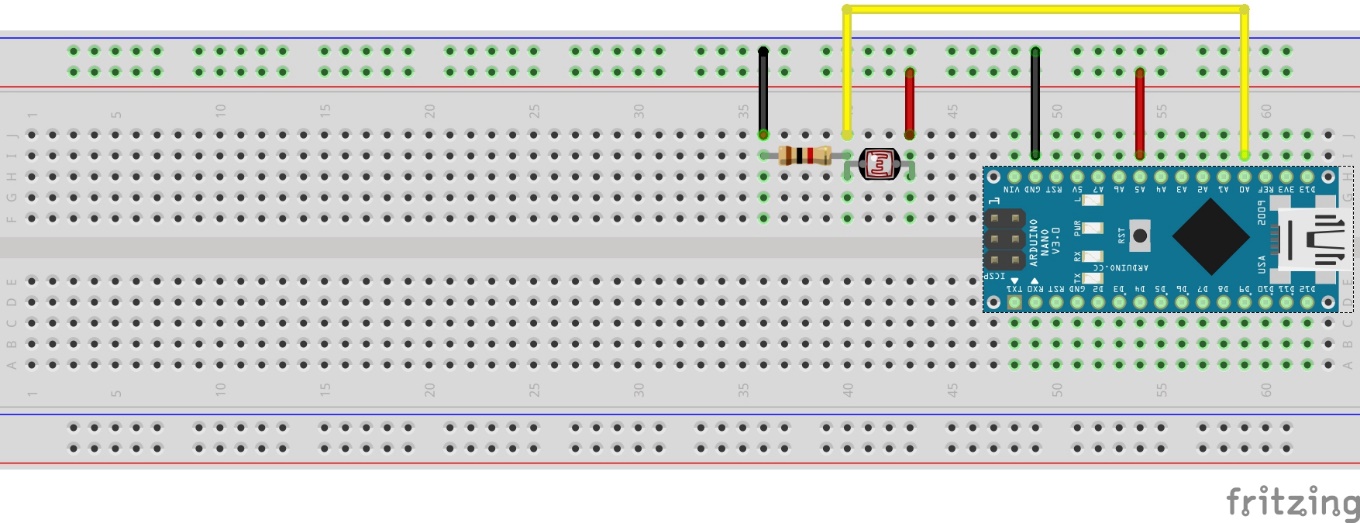
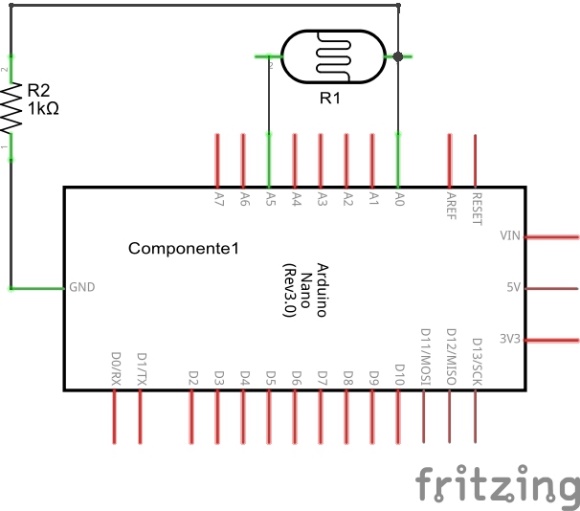
**3.3 LDR (serial)**

Nesta unidade aprenderemos a fazer leituras de um LDR, um sensor resistivo muito utilizado para medir diferenças de luminosidade. Esse será o segundo sensor abordado nessa apostila, mas o primeiro que utiliza entradas analógicas. Durante o desenvolvimento desse projeto, aprenderemos a utilizar a porta analógica do Arduino. Para desenvolvê-lo você precisará de:

* Protoboard;
* LDR (Light Diode Resister);
* Resistor de 1K ohm;

**Montando o hardware**

Com o Arduino desconectado, monte o circuito mostrado abaixo:

** **

**Analisando o hardware**

O hardware desse projeto é extremamente simples. O Arduino alimenta o circuito por meio da porta 5V, a corrente flui pelo LDR, que de acordo com a luz do ambiente varia sua resistência, alterando a quantidade de energia que passa por ele. Em seguida, essa energia volta para a placa, fechando o circuito. Há um resistor conectado ao GND do Arduino e a uma das extremidades do LDR (esse componente não possui polaridade), atuando como um filtro e estabilizando a tensão de uma das saídas. Essa mesma perna do LDR ligada ao resistor é conectada na porta A0 do Arduino, para que possamos fazer as leituras necessárias.

**O Código**

Abra a IDE do Brino e digite o código abaixo ou baixe-o da página do GitHub da apostila.

|  |
| --- |
| Numero Constante LDR = 0; // entrada do sensor LDR  Numero leitura; // variável para armazenar a leitura do sensor  Configuracao() {  USB.conectar(9600);  }  Principal() {  leitura = Pino.lerAnalogico(LDR); // Lê o sensor e armazena o valor:  USB.enviar(leitura); // Envia a leitura obtida para o monitor  esperar(500); // Intervalo entre uma medida e outra  } |

**Analisando o código**

O código começa definindo a porta de entrada do sensor e, em seguida, cria a variável que vai guardar o valor das medidas. Essa parte do código está exposta abaixo:

Numero LDR = 0;

Numero leitura = 0;

Depois, inicia-se a comunicação serial via USB para que seja possível apresentar os dados lidos pelo Arduino na tela do computador:

USB.conectar(9600);

A primeira linha do Principal() é:

leitura = Pino.lerAnalogico(LDR);

Ela faz a leitura da porta analógica onde está conectado o LDR (no caso a porta A0) e atribui o valor dessa medida a variável ‘leitura’ para que possa ser exibida tela pela próxima linha e, por fim, o código possui uma pausa de meio segundo entre uma medida e outra, para facilitar o controle dos dados obtidos.

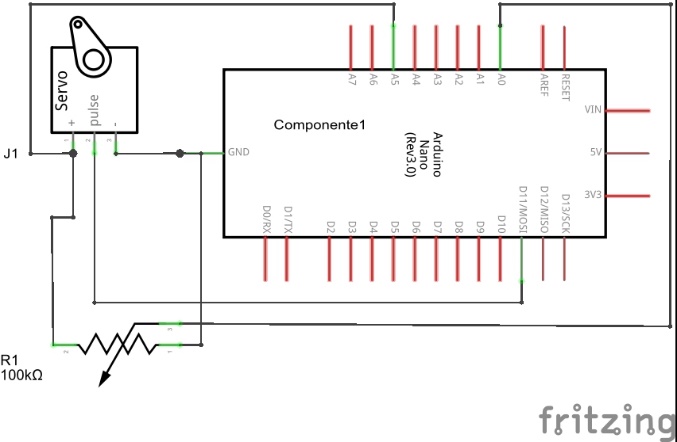
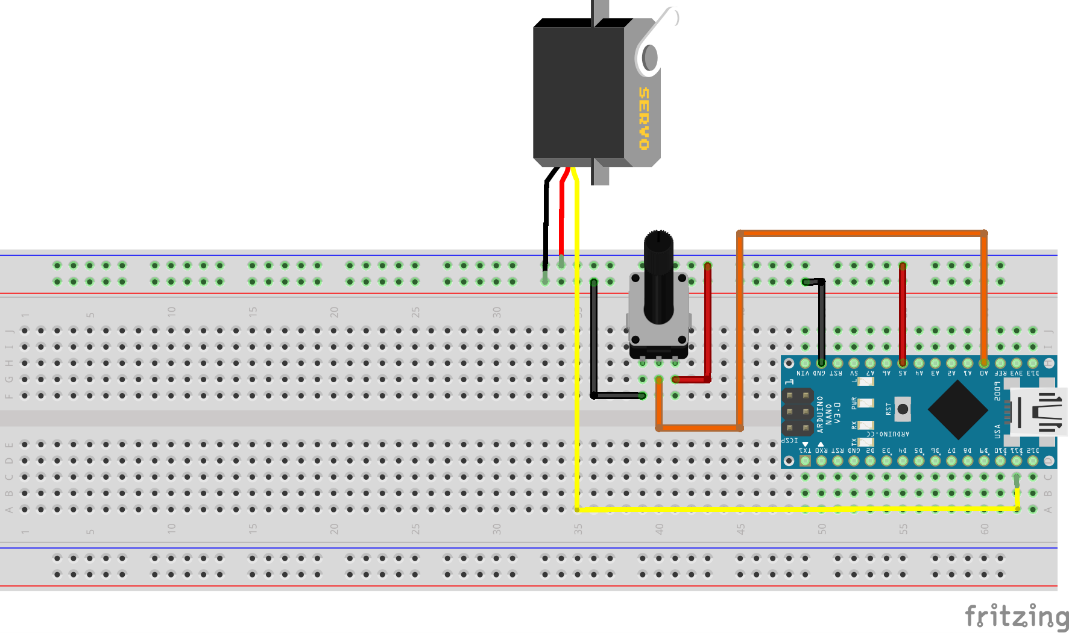
**3.4 Servo com Potenciômetro**

Neste capítulo aprenderemos como controlar um servo motor utilizando um Arduino com dados recebidos de um potenciômetro. Para isso teremos que lidar com portas analógicas para realizar a comunicação do Arduino com o potenciômetro e com o servo motor. O material necessário para a montagem desse circuito é:

* Sevo motor
* Protoboard
* Potenciômetro de 10K ohms

**Montando o Hardware**

Com o Arduino desconectado, monte o circuito abaixo:

**Analisando o Hardware**

O hardware desse projeto é dividido em duas partes, a do o servo motor e o a do potenciômetro. O servo motor é conectado na alimentação, que nesse caso vem do próprio Arduino, e o seu sinal é controlado por uma porta PWM. Já o potenciômetro deve ter seus pinos mais externos conectados ao 5V e ao GND e seu pino do meio, por onde sai o nosso sinal, deve ser lido por uma porta analógica. Para isso usaremos a porta A0 do Arduino.

**O Código**

|  |
| --- |
| usar Servo // Adiciona a biblioteca para uso de servos ao nosso codigo  Servo meuServo; // Define o nome do nosso servo  Numero Constante Apotenciometro = 0;  Numero valorPotenciometro;  Numero angulo;  Configuracao() {  meuServo.attach(5); // Define a porta onde o servo esta conectado  }  Principal(){  valorPotenciometro = Pino.ler(Apotenciometro); // Faz a leitura do potenciometro e a salva na variavel  angulo = proporcionar(valorPotenciometro, 0, 1023, 0, 180); // Faz uma regra de 3 simples  meuServo.write(angulo); // Muda o angulo do servo  esperar(15);  } |

**Analisando o código**

O código desse projeto começa com a adição da biblioteca para controle de servos e, em seguida, nos damos um nome ao servo:

usar Servo

Servo meuServo;

Depois disso as variáveis que iremos usar são declaradas e o bloco de Configuracao() possui apenas uma linha que é:

meuServo.attach(5);

Essa linha está associando o servo (meuServo) a porta 5 usando o comando ‘attach’. No loop Principal() podemos observar que ocorre a leitura do nosso potenciômetro (Apotenciometro) e a salva na nossa variável (valorPotenciometro). Em seguida ele faz uma regra de três entre a entrada, que varia de 0 a 1023, com a saída digital PWM para o servo e salva esse valor na variável angulo.

angulo = proporcionar(valorPotenciometro, 0, 1023, 0, 180);

Em seguida é feito o ajuste do servo motor com base na variável enviada usando o comando ‘write’.

meuServo.write(angulo);

**3.5 Sensor ultrassônico + memória EEPROM**

Neste capítulo, vamos montar um medidor de distâncias que registrará os valores obtidos na memória interna do Arduino, utilizando-os depois em outro programa. Este será o primeiro projeto que utilizará uma biblioteca externa do Arduino. Caso você não saiba instalá-las, existe um capítulo no final do livro explicando como fazer isso. Ela se chama Ultra e está disponível em “[*github.com/RatosDePC/Ultra*](http://github.com/RatosDePC/Ultra)”.

O material necessário para essa prática é:

* Protoboard;
* Sensor ultrassônico HC- sr04;

**Montando o Hardware**

Com o Arduino desconectado, monte o circuito abaixo:

//imagem

**Analisando o Hardware**

O hardware desse projeto é bem simples. A porta //x é conectada ao trigger do ultrassônico, que é a porta de controle e a //y ao *echo*, que transmite a resposta. Além das linhas de dados, conecta-se os pinos de alimentação – 5V ao 5V e GND ao GND.

**O Código**

Abra a IDE do Brino e digite o código a seguir:

|  |
| --- |
| usar Ultra  usar Memoria  Ultra u(5,4);  Configuracao(){  //não é necessário código de configuração  //USB para debugar  USB.conectar(9600);  para( Numero x = 0; x < 5; x++){  Numero d = u.medir();  Memoria.escrever(x, d);  USB.enviarln(d);  }  para( Numero x = 0; x < 5; x++){  USB.enviarln(Memoria.ler(x));  }  }  Principal(){  } |

**Analisando o código**

A primeira linha do código é:

*usar Ultra*

Essa linha irá importar a biblioteca *Ultra* para podermos utilizar seus métodos durante a execução, ou seja, avisamos ao Arduino que precisaremos deles para fazer o que queremos. A próxima linha também é uma importação:

*usar Memoria*

Nessa linha, avisamos o Arduino que utilizaremos a memória disponível internamente e os métodos associados à sua utilização.

Logo depois temos a linha:

*Ultra u(5,4);*

Essa linha cria um objeto *Ultra* que tem sua porta trigger (que envia o pulso ultrassônico)ligada ao pino 5, e a porta eccho ( que notifica a recepção do pulso)ao pino 4.

O método *Configuracao* começa inicializando a conexão *USB* para podermos verificar o bom funcionamento do nosso código. Depois temos um loop *para*. Esse loop cria uma variável que utilizaremos de contador e enquanto ela estiver no parâmetro definido, no caso, menor que 5, ele repetirá seu bloco realizando o incremento definido ao final, em nosso código, *x++*.

*para(* *Numero x; x < 5; x++){*

O computador repetirá o código 5 vezes (pois após esse número de repetições, x será maior ou igual a 5). O bloco de código do *para* começa com a seguinte linha:

*Numero d = u.medir();*

Essa linha cria uma variável pra guardar o valor da distância medida pelo ultrassônico com o método *medir()*. Depois temos:

*Memoria.escrever(x, d);*

Essa linha irá guardar a distância *d* no endereço *x* da memória, onde *x* é o contador do loop *para*. Além de guardar na memória iremos enviar para a porta USB para podermos verificar o valor lido pelo Arduino. Em seguida, repetiremos o *para* com um código que, ao invés de escrever, irá ler o que gravamos na memória e mostrar embaixo do que o primeiro *para* mostrou. A linha que faz isso é:

*USB.enviarln(Memoria.ler(x));*

O nosso método principal está vazio, pois não queremos que o Arduino repita todo esse processo, só queremos que ele seja executado uma vez.

1. **Introdução à Programação**

Para criar nossos próprios códigos é necessário aprender palavras-chaves e suas estruturas. Assim como um texto em língua Portuguesa ou em inglês, programas de computador utilizam linguagens com regras específicas de sintaxe para que o computador possa compreender o que queremos que ele faça.

Nesse capítulo abordaremos assuntos como:

* Variáveis;
* Comentários;
* Incrementadores;
* Instruções e laços de controle;
* Operadores lógicos;
* Funções ou Métodos.
  1. **Variáveis:**

Variáveis são muito usadas na programação, pois elas são capazes de armazenar dados. Uma das formas mais simples de pensar em uma variável é como uma caixa ou balde onde o computador pode armazenar ou ler dados. O uso de variáveis torna o código mais fácil de se entender e mais fácil de ser mantido. Por exemplo, se você possuir uma variável chamada *pinoLED* que define o pino 13 como uma saída para um LED e, mais a frente, decidir usar o pino 8 para isso, será fácil a troca, sendo necessária apenas substituir o valor 13 pelo valor 8 na declaração do valor da sua variável.

Esse exemplo poderia ser aplicado no segmento a seguir:

|  |
| --- |
| Numero pinoLED = 13; <= Aqui está a nossa variável!  Configuracao() {  Pino.definirModo(pinoLED, Saida);  }  Principal() {  Pino.ligar(Digital.pinoLED); // liga a porta digital acedendo o LED  esperar(1000); // espera por um segundo  Pino.desligar(Digital.pinoLED); // desliga a porta digital apagando o LED  esperar(1000); // espera por um segundo  } |

Variáveis são divididas em locais e globais, dependendo da parte do código em que ela for declarada. As locais são aquelas declaradas dentro de uma função (Principal(), para(), if(), etc.), enquanto as globais são declaradas fora delas. Variáveis locais só podem ser usadas dentro de suas funções, já as globais podem ser usadas em qualquer parte do programa.

**Nota:** É possível criar duas variáveis locais com o mesmo nome em funções diferentes, mas tome cuidado para não se esquecer que elas serão independentes.

Agora que entendemos qual a utilidade das variáveis em nossos programas vamos examinar os seus tipos:

* Numero: As variáveis do tipo Numero (deve ser escrito dessa forma para ser entendido pelo compilador, sem acento e com letra maiúscula. Isso se aplica a diversas palavras que serão abordadas no capítulo) são muito usadas pois são capazes de, como o próprio nome sugere, armazenar números inteiros entre

-32.768 a 32.767, ou seja, um número de 16 bits.

Ex.: Numero minhaVariavel = 3600;

* NumeroDecimal: Esse tipo de variável é capaz de armazenar números de até 32 bits com um ponto decimal.

Ex.: NumeroDecimal raio = 3,5;

* Letra: Essa variável armazena um caractere ASCII (iremos abordar isso mais a frente), ou seja, ela é capaz de armazenar qualquer caractere (a, A, 1, &, entre outros). Operações aritméticas podem ser aplicadas sobre esses dados (também estudaremos isso mais a fundo). Seu dado deve vir entre aspas simples (‘ ‘).

Ex.: Letra nota = ‘A’;

* Palavra: Esse tipo especial de variável pode ser comparado a uma serie de caracteres. Ela é usada para armazenar palavras e frases. Seu dado deve vir entre aspas simples ou duplas (‘’ ou “”).

Ex.: Palavra saudacao = ‘oi’;

* Condicao: A menor variável que vamos estudar é usada para guardar apenas dois valores, Verdadeiro ou Falso, e será muito usada em operações lógicas e como controle.

Ex.: Condicao chovendo = Falso;

Outro ponto importante quanto as variáveis são os nomes que elas podem receber. Salvo as palavras chave predefinidas pela linguagem de programação vigente, variáveis podem ter qualquer nome. Mesmo com toda essa liberdade, recomendamos fortemente que não usem acentos ou caracteres especiais, como ç, @, etc. Além de preferir, sempre que possível, nomes sugestivos (não, variável\_123 não é nada sugestivo) para que o código possa ser mais facilmente entendido pelos outros e por você mesmo. Tais nomes devem começar comecem com uma letra ou uma underline. São exemplos de bons nomes: valorSensor, motorDireito, porta\_LED e leituraDistancia.

Caso se deseje trabalhar com uma constante, a palavra-chave Constante pode ser adicionada ao início da variável, tornando-a invariável.

Ex.: Constante Numero minhaVariavel = 3600;

**Nota:** Com os tipos de números inteiros, você pode obter uma situação chamada *roll over*, em que um valor é somado ou subtraído extrapolando os limites da variável fazendo com que o fim de um intervalo role para outra extremidade. Por exemplo, temos uma variável do tipo Numero com o valor armazenado de 32.767 e somamos 1 a ela. O valor resultante dessa operação será de -32.768 e não 32.768.

* 1. **Comentários:**

Comentários são um recurso muito utilizado na programação, uma vez que, por meio de “notas do autor”, contribuem para um melhor entendimento e organização do código, pelo autor e por outros. Na hora da compilação, essas linhas são ignoradas pela máquina, fato que não desmerece a relevância dessa ferramenta. Enquanto trabalham em algoritmos extensos e complexos, é de suma importância que os programadores deixem notas explicando o que está sendo feito e qual o objetivo de cada bloco. Tal processo dinamiza o desenvolvimento. Imagine procurar um bug, ou consertar um erro em um código com 500 linhas (marco não muito difícil de ser alcançado) sem ter a menor ideia de onde procurá-lo...



No Brino, os comentários podem ser introduzidos por meio de uma barra dupla na frente da linha (//) fazendo com que essa linha em específico seja um comentário. Outra forma de usar essa ferramenta é utilizar uma barra acompanhada por um asterisco (/\*) abrindo um bloco de comentário que só terminará em um asterisco seguido por uma barra (\*/).

Obs.: evite colocar acentos em um comentário e em todo o código no geral. Ao abrir o código em diferentes editores de texto, eles podem não entender um caractere acentuado da mesma forma. De forma simples, nem todos os editores de texto “escrevem” na mesma língua, por isso entendem os acentos de forma diferente.

Ex.: // Isso e um comentario.

Ex. 2: /\*

\* Isso e um comentario de bloco

\*/

* 1. **Incrementadores:**

No Brino existem alguns operadores que podem incrementar ou decrementar o valor de uma variável. Além daqueles a que estamos acostumados (+, -, \*, /), há também incrementadores especiais. São eles:

|  |  |  |
| --- | --- | --- |
| Operador | Exemplo | Resultado |
| ++ | Variável++ | O valor da variável vai ser incrementado em uma unidade |
| -- | Variável-- | O valor da variável vai ser decrementado em uma unidade |
| += | Variável += n | O valor da variável será incrementado em n unidades (no caso de Palavras, o trecho n será adicionado ao final) |
| -= | Variável -= n | O valor da variável será decrementado em n unidades |
| \*= | Variável \*= n | O valor da variável será igual ao produto do seu antigo valor e n |
| /= | Variável /= n | O valor da variável será igual ao quociente do seu antigo valor e n |

**Nota:** Quando estamos falando de programação os símbolos da divisão e da multiplicação costumam ser substituídos por uma barra simples (/) e por um asterisco (\*) respectivamente.

**Nota:** Quando atribuímos o valor por meio de uma igualdade (=), o valor da direita é atribuído ao lado esquerdo.

Ex.: X = 2 \* 5

* 1. **Instruções e laços de controle:**

Assim como nós, as máquinas são capazes de tomar decisões, porém de maneira muito mais simples. Os códigos fazem decisões por meio das chamadas operações booleanas. Essas operações aceitam apenas dois resultados, o verdadeiro e o falso. Uma forma de pensar quanto a isso é relacionar esse recurso a situações do cotidiano. Por exemplo, se estiver chovendo, devo pegar o guarda-chuva, caso contrário, não preciso pegá-lo. Aqui, a expressão booleana seria o fato de estar ou não chovendo e a instrução de controle seria pegar ou não o guarda-chuva.

Uma forma muito usada para representar essas situações é por meio de diagramas de blocos como o representado a seguir:

Agora que temos uma noção do que se trata, podemos estudar os Operadores Relacionais. Eles nada mais são do que comparadores que usaremos para analisar informações tendo uma saída de Verdadeiro ou Falso. Os operadores estão representados na tabela a seguir:

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Descrição | Exemplo | Resultado |
| > | Maior que | 2 > 1 | Verdadeiro |
| >= | Maior ou igual a | 2 >= 2 | Verdadeiro |
| < | Menor que | 2 < 1 | Falso |
| <= | Menor ou igual a | 2 <= 2 | Verdadeiro |
| == | Igualdade | A == A | Verdadeiro |
| != | Desigualdade | A != A | Falso |

Assim como podemos observar os operadores não estão limitados a comparar apenas números, mas também podem comparar Letras e outras variáveis.

**Nota:** Tome cuidado!!! Não se esqueça que o símbolo = faz uma atribuição sendo diferente do == que faz uma comparação.

Tendo uma ideia de como obter resultados Verdadeiros e Falsos podemos desenvolver um pouco mais a ideia usando os operadores condicionais. Esses são os operadores que vamos usar para tomar decisões quando associados aos comparadores. Veremos a seguir como usar os operadores se, senao e o senao se.

Eles seguem as estruturas exemplificada abaixo:

|  |
| --- |
| se(expressão booleana){  // Se a expressão for Verdadeira esse bloco será executado.  }  // Se ela for Falsa o bloco será ignorado. |

|  |
| --- |
| se(expressão booleana){  // Se a expressão for Verdadeira esse bloco será executado.  }  senao{  // Se ela for Falsa esse bloco será executado.  } |

|  |
| --- |
| se(expressão booleana){  // Se a expressão for Verdadeira esse bloco será executado.  }  senao se(outra expressão booleana){  // Se a segunda expressão for Verdadeira  // esse bloco será executado.  }  senao{  // Se nenhuma dos blocos anteriores forem executados,  // esse será.  } |

Além do *se* e do *senao,* existe a instrução enquanto(). O laço enquanto(), como o próprio nome sugere, executa um bloco de código enquanto uma condição, entre os seus parênteses, for verdadeira. Ele é usado para realizar um processo enquanto for necessário, enquanto a expressão for verdadeira, e para quando não for mais necessário, a expressão se tornar falsa.

|  |
| --- |
| enquanto(Condicao){  // Esse bloco é repetido enquanto a condição for verdadeira.  } |

O laço para() é usado para repetir um determinado bloco de código um número determinado de vezes usando, para isso, uma variável como contador. Ao contrário dos outros anteriormente citados, o para() aceita mais parâmetros. O primeiro é a declaração das variáveis locais, sendo seguido pela expressõ booleana e por uma expressão de incremento ou decremento do valor da variável.

|  |
| --- |
| para(Tipo <nome> = <valor>; <nome> <operador> <valorReferencia>; <incremento>){  // Bloco que será repetido.  } |

Exemplo:

|  |
| --- |
| para(Numero x = 0; x <= 10; x++){  //Bloco a ser repetido;  } |

* 1. **Operadores lógicos:**

Os operadores lógicos são usados quando uma expressão booleana não é o suficiente para a tomada de decisões, então, por meio deles, nós podemos ter mais do que uma expressão booleana com apenas uma saída. Um jeito interessante de pensar nesses problemas é voltando ao exemplo do guarda-chuva. Eu estou saindo de casa, se estiver ensolarado eu não irei pegar o guarda-chuva, mas caso esteja chovendo ou pareça que vai chover, devo pega-lo. A seguir estão representados os operadores:

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Descrição | Exemplo | Resultado |
| E | Se ambas forem verdadeiras | 2 > 1 e 2>=2 | Verdadeiro |
| Ou | Se uma das duas for verdadeira | 2 >= 2 ou 1>3 | Verdadeiro |

Repare nos exemplos, para saber a saída, devemos fazer essa operação por etapas. No caso 2 > 1 e 2>=2, temos duas expressões booleanas. Como já sabemos resolvê-las podemos dizer que o problema fica Verdadeiro e Verdadeiro. Podemos observar que ambas as expressões têm uma saída verdadeira e estão ligadas pelo operador *e*, logo, o resultado será Verdadeiro.

* 1. **Funções ou Métodos:**

Funções ou métodos consistem basicamente em um determinado bloco de código escrito pelo desenvolvedor para evitar repetir um conjunto de instruções. O Brino requer o uso de pelo menos duas funções. Essa ferramenta é extremamente importante para reduzir o tamanho dos códigos e otimizar a utilização da memória do computador. Para se declarar uma função, é necessário dizer que tipo de dado ela nos retornará, ou seja, qual a resposta que ela nos fornece após efetuar todas as suas instruções, que pode ser qualquer tipo de variável ou *SemRetorno* se ela não responde nada. Além do retorno, precisamos declarar os dados, ou argumentos, que ela receberá para efetuar suas operações. A declaração de uma função segue o modelo:

|  |
| --- |
| TipoDeRetorno <nome>(TipoDeVariável <nome\_argumento\_1>, ..., argumento n){  // Bloco de instruções da função ou método.  } |

O Brino possui duas funções obrigatórias que iremos debater:

* Configuraçao():

Executada uma vez quando o Arduino é inicializado. É responsável por preparar o hardware para a execução do loop principal. O método de Configuracao() é executado uma única vez quando o Arduino é ligado e é ignorado até que seja reiniciado. É nele que definimos a conexão com a porta USB, os modos de operação dos pinos, entre outros parâmetros.

|  |
| --- |
| Configuracao(){  // Esse bloco é repetido apenas uma vez na inicialização.  } |

* Principal():

O método Principal() é um dos mais usados para se colocar a parte principal do programa. Ele é executado a partir do momento que o Arduino é ligado, após a execução da configuração, até o momento em que ele é desligado podendo ser repetido incontáveis vezes. Nele colocamos todas as operações necessárias para o funcionamento contínuo do projeto.

|  |
| --- |
| Principal(){  // Esse bloco é repetido continuamente.  } |

1. **Introdução a eletrônica básica**

Uma parte muito importante da robótica é a eletrônica, ou o hardware, pois ela que é capaz de interagir com o meio externo ao robô. É por meio de atuadores (motores e servomotores) e sensores que ele será capaz de analisar e se locomover interagindo com o ambiente a sua volta. Esse é um assunto muito amplo e que pode requerer muitos cálculos, mas como esse material possui caráter apenas introdutório, só abordaremos as partes realmente necessárias como:

* Grandezas;
  + Corrente Elétrica (I);
  + Resistência (R);
  + Tensão (U);
* Componentes:
  + Resistores;
    - LDR;
  + Buzzer;
  + Interruptores;
  + Capacitores;
  + Diodos;
    - LEDs;
* Associação de componentes;
  1. **Grandezas:**

Eletrônica é o ramo da física estuda as propriedades e aplicações de mecanismos cujo funcionamento baseia-se do movimento de elétrons. Nesta unidade veremos as principais grandezas pertencentes à essa ciência.

* + 1. **Corrente Elétrica (I):**

A corrente elétrica é o fluxo ordenado de cargas por um condutor, ou seja, para os nossos estudos ela pode ser definida o fluxo de elétrons por materiais como fios e componentes. Ela ocorre entre dois pontos com potenciais diferentes e pode ser medida em Ampere (A). Uma forma de pensar nela é comparando-a com a agua. A corrente seria a quantidade de água que passa pelo cano, e o cano o seu condutor. Essa corrente pode ser dividia nos seguintes tipos:

* + - 1. **Contínua (C.C.):**

A corrente continua ocorre quando o fluxo das cargas ocorre sempre na mesma direção, ou seja, quando os polos se mantem constantes. Ela é constituída pelos polos positivo e negativo e é usada em baterias e pilhas. Abaixo, vê-se em um gráfico do valor corrente em relação com o tempo:

[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjNnLDi6vDKAhXHipAKHb2MAOUQjRwIBw&url=http://www.sofisica.com.br/conteudos/Eletromagnetismo/Eletrodinamica/caecc.php&psig=AFQjCNEMtb0Vg_PVwb1uV0usXbDBrE1ddQ&ust=1455318696410416)

* + - 1. **PWM:**

PWM (Pulse-Width Modulation) significa modulação por largura de pulso e pode ser usado para fazer comunicações ou para controlar o valor da alimentação. Ele consiste em ligar e desligar várias vezes consecutiva uma voltagem em um curto período de tempo, fazendo com que aparentasse de que se usou um valor intermediário. Ele é usado no controle de alguns motores.



* + - 1. **Corrente Alternada (C.A.):**

A corrente alternada, diferentemente da contínua, altera o seu sentido com o decorrer com o tempo, fazendo com que as cargas fiquem indo e vindo. Ela é composta por fases e, muitas vezes, pelo fio neutro. Ela possui menor perda quando comparada a corrente contínua e, por isso, é usada principalmente em linhas de transmissões e tomadas. A sua forma de onda mais comum é a senoidal (representada abaixo), por ser a mais eficiente. Mas, em certas aplicações, outras formas podem ser utilizadas.



* + 1. **Resistência (R):**

A resistência elétrica é a capacidade de um corpo qualquer de se opor a corrente elétrica mesmo quando uma diferença de potencial é aplicada. Na análise com a água a resistência está diretamente relacionada a espessura do cano, permitindo que passe mais ou menos água. Ela é medida em ohm (Ω) e está diretamente relacionada a lei de Ohm, que será explicada mais a frente.

* + 1. **Tensão (U):**

A tensão, também conhecida como DDP (Diferença De Potencial) ou voltagem, assim como o nome já sugere, representa a diferença de potencial entre dois pontos, ou seja, é a quantidade de energia gerada para movimentar uma [carga elétrica](http://www.infoescola.com/fisica/carga-eletrica/). A tensão é medida em Volt (V) em homenagem ao físico italiano [Alessandro Volta](https://pt.wikipedia.org/wiki/Alessandro_Volta" \o "Alessandro Volta) Um jeito de pensar nela é como a força que “empurra” as cargas. Na comparação com a hidráulica, ela seria a diferença de altura de duas caixas d’água que causa uma energia potencial nas moléculas.

* 1. **Componentes:**
     1. **Resistores:**

Um resistor é um componente que apresenta uma certa dificuldade ou resistência à passagem de corrente. Os resistores são amplamente utilizados em circuitos, seja para regular a tensão ou a corrente. A resistência de um resistor é medida em ohms. Para identificar os diferentes resistores de acordo com sua resistência, utilizam-se listras coloridas que seguem um código convencionado.

**Nota**: A tabela está localizada no capítulo 8 para consulta.

Os resistores podem ser associados para obter valores não disponíveis comercialmente, ou específicos para o projeto. Eles podem ser associados de três formas:

* Em série: os resistores são conectados um na frente do outro de forma que a corrente percorre todos. A resistência equivalente é igual à soma das n resistências em série.



* Em paralelo: os resistores são conectados paralelamente, dessa forma, a corrente se divide entre eles. A resistência equivalente segue a fórmula a seguir:



* Mista: os resistores são conectados tanto em paralelo quanto em série. A resistência equivalente será calculada utilizando as fórmulas de associações em série e em paralelo.



Além dos resistores comuns, existem resistores que variam a sua resistência de acordo com fenômenos físicos como a luminosidade, no caso do LDR, ou a temperatura, no caso do Termistor. Devido a suas propriedades, tais resistores são utilizados como sensores na robótica. Depois de alguns capítulos, você perceberá que grande parte dos sensores são resistivos. Agora falaremos um pouco sobre o LDR e sobre o potenciômetro, os resistores variáveis mais utilizados:

* + - 1. **LDR:**

Um LDR (Light Dependent Resistor) é um Resistor Dependente de Luz ou Fotoresistência. Ele tem a capacidade de variar a sua resistência em função da luz que incide sobre ele. Tipicamente, à medida que a intensidade da luz aumenta, a sua resistência diminui, permitindo que mais corrente flua por ele. Ele pode ser usado para confecção de sensores de luminosidade ou de reflexão. Abaixo podemos ver uma imagem de um LDR e seu símbolo esquemático.

[](https://pt.wikipedia.org/wiki/Ficheiro:Photoresistor.svg)[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=&url=http://blog.vidadesilicio.com.br/arduino/basico/sensor-de-luz-ldr/&psig=AFQjCNG3hJ_-1HS9xW9H0U2PbD8WcO9Upg&ust=1455035345763683)

* + 1. **Buzzer:**

Um buzzer é um componente capaz de produzir sons na frequência recebida atuando de maneira semelhante a uma caixa de som, porem com um consumo menor. Ele possui polaridade definida (a perna mais longa é o positivo) e é composto 2 camadas de metal e uma camada interna de cristal piezoeléctrico.

[](https://en.wikipedia.org/wiki/File:BuzzerSymbol.png)[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiQvPf5yejKAhVIipAKHfsIB0IQjRwIBw&url=http://www.baudaeletronica.com.br/buzzer-5v.html&psig=AFQjCNHbJSbGUzCKCN2cZ93_hrACkv-Cqg&ust=1455035004277653)

* + 1. **Interruptores e botões:**

Componente responsável por manter o circuito aberto ou fechado, quando acionado, fecha o circuito e permite a passagem de corrente.

[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjDzJXNs-jKAhWGS5AKHYAVBUIQjRwIBw&url=http://blog.vidadesilicio.com.br/arduino/basico/botoes/&bvm=bv.113370389,d.Y2I&psig=AFQjCNGkTiArKsJ-VRITqgbWub6wlnGCMA&ust=1455029002935621)

* + - 1. **Relés:**

Um relé é um interruptor eletromecânico que é acionado por uma corrente elétrica mudando a posição da chave, assim permitindo ou não a passagem de uma corrente elétrica. Ele possui diversas aplicações, seu destaque é o uso de baixas correntes para o controle de correntes maiores como, por exemplo, na automação residencial e no controle de portas de elevadores.

[](http://reprap.mercadoshops.com.br/modulo-rele-canal-para-microcontrolador-pic-arm-arduino-83xJM)

* + 1. **Capacitores:**

Capacitores são dispositivos capazes de armazenar energia elétrica na forma de campo eletroestático. Tal habilidade, chamada capacitância, é qualificada de acordo com a quantidade de carga armazenada. A unidade de medida é o Farad. A seguir discutiremos dois dos tipos de capacitores, que são os mais utilizados em pequenos projetos:

[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjJyK7ghfXKAhXIC5AKHW54CMIQjRwIBw&url=http://www.sabereletrica.com.br/entenda-o-funcionamento-dos-capacitores&psig=AFQjCNGWxVsUn0lecRqlsz7dmhRgp8672w&ust=1455462591624557) [](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=&url=http://www.pcs.usp.br/~labdig/manuais.html&psig=AFQjCNGWxVsUn0lecRqlsz7dmhRgp8672w&ust=1455462591624557)

* + - 1. **Capacitor de cerâmica:** Também conhecidos como capacitores cerâmicos, destacam-se como os capacitores mais utilizados atualmente. São usados em circuitos de corrente continua (C.C.) aos de frequências muito altas. Não possui polaridade, ou seja, funciona nos dois sentidos.
      2. **Capacitor eletrolítico:** Não recomendados para projetos que envolvam sinais de frequências elevadas, sendo para esses mais recomendados outros capacitores. Possuem destaque para conexões entre circuitos e filtragem de sinais de baixa frequência.
    1. **Diodos:**

O diodo é um componente de dois terminais que conduz a corrente em apenas um sentido, bloqueando a sua passagem no sentido oposto. Ele é um semicondutor muito usado em retificadores e em circuitos de proteção. O uso dele geralmente implica em uma pequena queda de tensão e existem diversos modelos com diversas funções.

 [](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwibyM6mu-jKAhUFQ5AKHeU4CEIQjRwIBw&url=http://www.protostack.com/diodes&bvm=bv.113370389,d.Y2I&psig=AFQjCNEDH4UurCh-PO-1i2o-RlqUg3gu5Q&ust=1455031072154917)

[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwibyM6mu-jKAhUFQ5AKHeU4CEIQjRwIBw&url=http://www.protostack.com/diodes&bvm=bv.113370389,d.Y2I&psig=AFQjCNEDH4UurCh-PO-1i2o-RlqUg3gu5Q&ust=1455031072154917)

[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwibyM6mu-jKAhUFQ5AKHeU4CEIQjRwIBw&url=http://www.protostack.com/diodes&bvm=bv.113370389,d.Y2I&psig=AFQjCNEDH4UurCh-PO-1i2o-RlqUg3gu5Q&ust=1455031072154917)

* + - 1. **LEDs:**

Um LED (**Light Emitting Diode) é um diodo emissor de luz. Isso significa que ele tem as propriedades de um diodo (descrito acima) e é capaz de emitir luz própria como uma pequena lâmpada. Ele é um semicondutor e a sua simbologia está representada a seguir junto a uma imagem identificando os seus polos:**

/

**Nota:** O valor de tensão e corrente de cada LED pode ser consultado no capitulo 9, na parte de LEDs.

**Nota:** // escrever como testar um LED com o multimetro

1. **Motores**

Nesse capitulo iremos abordar os motores elétricos, mais precisamente os de corrente continua (C.C.). Esse tipo de motor é capaz de converter a energia elétrica em energia mecânica, e vice-versa. Eles podem ser classificados como atuadores e são um componente que deve ser escolhido com calma. A seguir veremos os principais tipos de motores C.C..

* Motores C.C.;
* Servo motores;
* Motores de passo.

* 1. **Motores C.C.:**

Os motores “normais” de corrente continua possuem baixa potência e alta velocidade, tornando necessário, o uso de uma caixa de redução. Uma caixa de redução nada mais é do que um conjunto de engrenagens que mudam a velocidade e a potência a mesma proporção. O uso desse tipo motor com o arduino normalmente requer uma ponte H.

[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjcz76dr-7KAhUFEpAKHTYBCJMQjRwIBw&url=https://multilogica-shop.com/motor-cc-6v&psig=AFQjCNFyPBsuyBVENojGo2qHTnEteW262w&ust=1455233994296438)[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjv-qqxr-7KAhVEvJAKHY3oDncQjRwIBw&url=http://www.filipeflop.com/pd-11d0db-motor-dc-3-6v-com-caixa-de-reducao-e-eixo-duplo.html&bvm=bv.113943665,d.Y2I&psig=AFQjCNH5yfA3_9l3-UQOLkvIWgaxudOrFQ&ust=1455234016209349)

* 1. **Servo motores:**

Servo motores são motores de corrente continua com uma caixa de redução acoplada e um circuito controlador. Existem dois tipos de servo motores;

Nota: Fique atento ao consumo deles, eles podem precisar de uma alimentação externa.

* + 1. **Rotação limitada:**

Esse é o tipo mais comum de servo motor. Ele saber o ângulo em que ele se encontra e, por meio de um sinal pode alterar a sua posição para o ângulo desejado. Esse tipo de motor usa uma porta PWM e geralmente tem sua movimentação em ângulos de 180° ou de 360°.

[](https://www.google.com.br/imgres?imgurl=http://blog.filipeflop.com/wp-content/uploads/2013/08/Micro_Servo_9g_SG90_TowerPro1.jpg&imgrefurl=http://blog.filipeflop.com/motores-e-servos/&h=781&w=850&tbnid=YAzgbK7xtGqb-M:&docid=d5_twbGubslEOM&ei=Ocu7Vv7GLMKCwQTTyYGYBQ&tbm=isch&ved=0ahUKEwi-x-aNsO7KAhVCQZAKHdNkAFM4rAIQMwgpKCYwJg)

* + 1. **Rotação continua:**

Esse tipo de motor não possui nenhum limite de angulação, porém não se é possível controlar o ângulo deles, mas sim o seu sentido de rotação. Seu controle também deve ser feito por uma porta PWM.

[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjW5abKr-7KAhVCjpAKHUPeC9UQjRwIBw&url=http://www.solucoesindustriais.com.br/empresa/instalacoes_e_equipamento_industrial/confiman/produtos/maquinas-ferramenta/motores-servo&bvm=bv.113943665,d.Y2I&psig=AFQjCNHvUJPLnDQElQfazzkk6ViguIb-TA&ust=1455234071084179)

* 1. **Motores de passo:**

Os motores de passo são usados para realizações de movimentos muito precisos, sendo usado para isso “passos”, que seriam pequenos movimentos do motor. Ele usa uma quantidade variada de fios que podem acionar as suas bobinas internas podendo controlar o seu posicionamento por meio de padrões. Ele é usado, por exemplo, em impressoras.

**Nota:** Existem placas especiais para facilitar o seu controle.

[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjbqrKVkIzLAhVJE5AKHbMLDfAQjRwIBw&url=http://www.techmount.com.br/motor-de-passo-5v-4-polos-5-fios&psig=AFQjCNExI6jaQgSvXBpPG8r-XvFJ6rrmjg&ust=1456256420159832)

1. **Baterias**
2. **Materiais importantes**
   1. **Protoboard:**

Também conhecida como breadboard, é uma maneira simples e eficaz de se realizar simulações de circuitos sem o uso de soldas, recomendada para prototipagem. Caracteriza-se como uma placa de plástico com diversos orifícios, abaixo desses orifícios encontra-se uma malha metálica que realiza as ligações como é indicado no diagrama abaixo:

[](http://produto.mercadolivre.com.br/MLB-706683459-protoboard-400-pontos-_JM)[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjFq6CIsOjKAhWElZAKHaUODkMQjRwIBw&url=http://artefactos.leame.com/protoboard/&psig=AFQjCNE67Ixn_IkQ150IxXF1s_k6aGrUDw&ust=1455028037719740)

* 1. **Jumpers:**

Jumpers são fios usados para fazer ligações, como as realizadas na protoboard. Caracterizados como fios convencionais com segmentos de fios rígidos em suas extremidades para facilitar o seu uso para prototipagem, como numa protoboard . Eles podem ser facilmente conectados aos furos da protoboard ou de alguns Arduinos. Segmentos de fio soldados a uma placa de circuito impresso com o objetivo de ligar dois pontos da mesma também podem ser chamados de jumpers.

**Nota:** Eles podem ser as linhas que fazem as ligações em esquemáticos!!!

[](http://produto.mercadolivre.com.br/MLB-719927626-10-cabos-fio-jumper-machomacho-20cm-protoboard-arduino-pic-_JM)

* 1. **Fonte de alimentação:**

[](https://www.google.com.br/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjn9OKZsujKAhWHvJAKHRTGAT8QjRwIBw&url=http://www.eletrodex.com.br/fonte-de-alimentac-o-regulavel-dupla-hk-3003d.html&psig=AFQjCNFsOX3Kgs9PccOs-6ae2MlS-1cc8w&ust=1455028606826695)É um aparelho capaz de gerar corrente continua (CC) a partir da corrente alternada (AC) de uma tomada. É possível regular a voltagem e a corrente a partir de potenciômetros normalmente localizados na parte dianteira do aparelho.

**// criar capitulo de ferramentas (soldador, multímetro...)**

1. **Tabelas importantes**
   1. **Valor de resistores:**

A unidade de medida de resistência é o ohm, representado pelo símbolo grego Ômega. Para saber a resistência de um resistor basta ler as duas primeiras faixas e multiplicar esse valor pela terceira. A quarta mostra a tolerância do resistor que é o valor de sua variação em relação ao valor nominal. Alguns resistores possuem 5 cores, sendo a quarta o multiplicador e a quinta a tolerância.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cor | Valor | Multiplicador | Cor quarta faixa | Tolerância |
| Dourado | - | 0.1 | Prata | 10% |
| Preto | 0 | 1 | Ouro | 5% |
| Marrom | 1 | 10 | Amarelo | 4% |
| Vermelho | 2 | 100 | Laranja | 3% |
| Laranja | 3 | 1.000 | Vermelho | 2% |
| Amarelo | 4 | 10.000 | Marrom | 1% |
| Verde | 5 | 100.000 | - | - |
| Azul | 6 | 1000.000 | - | - |
| Violeta | 7 | Não existe | - | - |
| Cinza | 8 | Não existe | - | - |
| Branco | 9 | Não existe | - | - |

* 1. **ASCII:**

A memória do computador não é capaz de armazenar diretamente caracteres, tendo que os armazenar na forma de números. Cada caractere possui por conseguinte o seu equivalente em código numérico: é o **código ASCII** (*American Standard Code for Information Interchange* - Código Americano Padrão para a Troca de Informações). Existem versões estendidas desse código, mas aqui trataremos da sua versão básica que possui 7 bits, ou seja, possui 128 caracteres.

Nessa tabela o código 0 a 31 não são realmente caracteres, sendo chamados de *caracteres de controle*. Os códigos 65 a 90 representam as letras maiúsculas e os códigos 97 a 122 representam as letras minúsculas. Abaixo representamos a tabela a partir do código 32.

**Nota:** Bastar somar ou subtrair 32 ao código ASCII para trocar entre as letras maiúsculas e minúsculas. Isso representa a troca do 6° bit da representação binaria.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Código** | **Caractere** | **Código** | **Caractere** | **Código** | **Caractere** |
| 32 | ESPAÇO | 64 | @ | 96 | ` |
| 33 | ! | 65 | A | 97 | a |
| 34 | “ | 66 | B | 98 | b |
| 35 | # | 67 | C | 99 | c |
| 36 | $ | 68 | D | 100 | d |
| 37 | % | 69 | E | 101 | e |
| 38 | & | 70 | F | 102 | f |
| 39 | ‘ | 71 | G | 103 | g |
| 40 | ( | 72 | H | 104 | h |
| 41 | ) | 73 | I | 105 | i |
| 42 | \* | 74 | J | 106 | j |
| 43 | + | 75 | K | 107 | k |
| 44 | , | 76 | L | 108 | l |
| 45 | - | 77 | M | 109 | m |
| 46 | . | 78 | N | 110 | n |
| 47 | / | 79 | O | 111 | o |
| 48 | 0 | 80 | P | 112 | p |
| 49 | 1 | 81 | Q | 113 | q |
| 50 | 2 | 82 | R | 114 | r |
| 51 | 3 | 83 | S | 115 | s |
| 52 | 4 | 84 | T | 116 | t |
| 53 | 5 | 85 | U | 117 | u |
| 54 | 6 | 86 | V | 118 | v |
| 55 | 7 | 87 | W | 119 | w |
| 56 | 8 | 88 | X | 120 | x |
| 57 | 9 | 89 | Y | 121 | y |
| 58 | : | 90 | Z | 122 | z |
| 59 | ; | 91 | [ | 123 | { |
| 60 | < | 92 | \ | 124 | | |
| 61 | = | 93 | ] | 125 | } |
| 62 | > | 94 | ^ | 126 | ~ |
| 63 | ? | 95 | **­**underline | 127 | DEL |

* 1. **Leds:**

Para que cada LED possa ser utilizado, ele precisa de uma tensão e de uma corrente especifica. Para saber como escolher as medidas certas basta saber o tipo de LED com que se está trabalhando e seguir a tabela abaixo:

|  |  |  |
| --- | --- | --- |
| LED | Tensão (V) | Corrente (mA) |
| Infravermelho 940 nm | 1,5 | 50 |
| Vermelho normal | 1,7 | 10 |
| Infravermelho 880 nm | 1,7 | 50 |
| Vermelho super brilhante | 1.85 | 20 |
| Amarelo normal | 2,1 | 10 |
| Laranja normal | 2,1 | 10 |
| Verde normal | 2,2 | 20 |
| Azul brilhante | 3,6 | 20 |
| Azul ultra brilhante | 4,5 | 20 |

//conferir tabela

1. **Associações**
   1. **Resistores:**
   2. **Capacitores:**
   3. **Pilhas/ Baterias:**