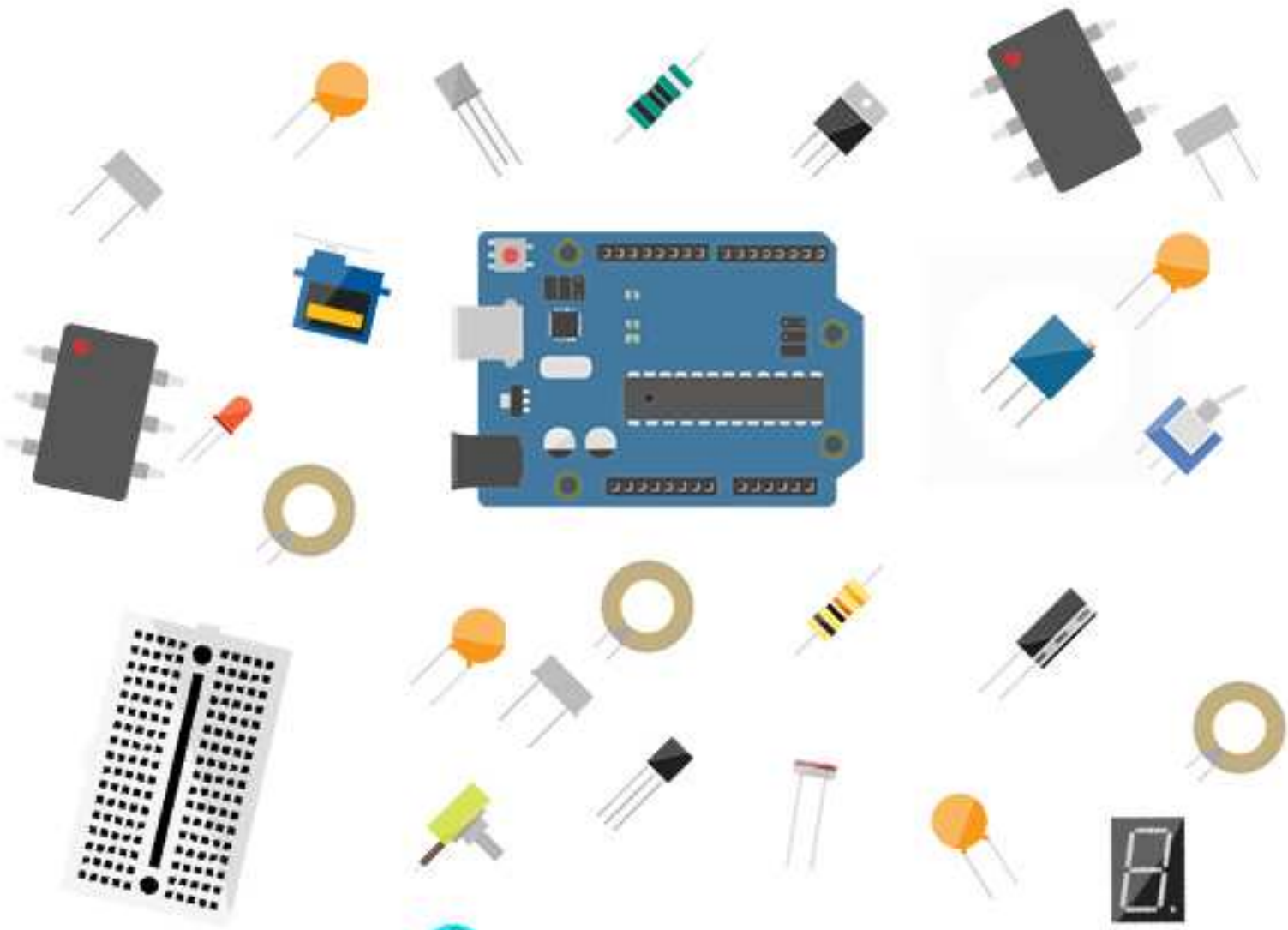
Começando com o Arduino

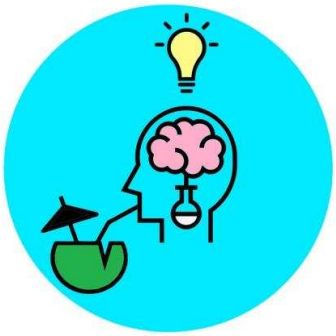
O que é necessário para aprender, fazer e dominar essa ferramenta!



ENGENHEIRO CAIÇARA

Sobre o blog

O Engenheiro Caiçara é um blog que tem, como principal intuito, desvendar a engenharia. Afinal, já existe por aí, milhares de materiais com termos técnicos, difíceis de serem compreendidos logo de cara. Independente de seu grau de conhecimento sobre a engenharia, queremos que você, caro leitor, sinta-se a vontade lendo nossos artigos e, principalmente, entenda-os sem complicação.



A Engenharia é linda, mágica e maravilhosa, depois que você entende seus conceitos e enxerga sua beleza na prática.

Ela está aí, no seu cotidiano, em tudo a sua volta e o nosso objetivo é mostrar, a você, como vê-la dessa forma.

Sobre o Autor

Formado como Técnico em Eletrônica pela ETEC Aristóteles Ferreira, atualmente, estudo Engenharia Elétrica em uma universidade localizada na cidade de Santos. Meu foco principal é seguir na área de Eletrônica, vertente na qual tenho mais facilidade e interesse.

O que me motivou a escrever esse E-book foi mostrar como o Arduino é uma ferramenta simples e fantástica. Elenquei as principais dúvidas, de quem começa com essa plataforma. E, muitas das coisas que escrevi, um dia tive dúvida e tive de pesquisar rs. Espero que esse material lhe auxilie e consiga despertar o maker, que há dentro de você. Espero que goste! Abraços!

SUMÁRIO

1. O QUE É O ARDUINO E O QUE PRECISO PARA COMEÇAR:

Não é possível falar de Arduino, sem conhecer um pouco de sua história e a idealização do projeto.

O projeto fora iniciado na Itália, na cidade de Ivrea em 2005,durante oInteraction Design Institute, tendo como objetivo a interação de projetos escolares, de forma que se obtivesse um sistema de prototipação com baixo custo e fácil utilização.

O nome Arduino é uma referência a um bar local, frequentado por membros do corpo docente e alunos do instituto. As placas eram vendidas em forma de kits, estimulando os alunos a montarem seus próprios projetos. O mundo percebeu a facilidade de uso e poder, dessa incrível plataforma para prototipagem rápida e uma forma de introduzir a programação de microcontroladores. Com isso, o projeto original foi sendo otimizado e novas versões introduzidas.

Hoje, o sucesso é tanto, que existem inúmeras versões da placa e distribuidores espalhados pelo mundo inteiro, vendendo o produto. Realmente, a plataforma Arduino mudou o conceito de prototipagem e ajudou muita gente a colocar sua ideia em prática, sem precisar ter conhecimentos avançados de eletrônica.

1.1. HARDWARE

Atualmente, há uma série de versões do Arduino, sendo a maioria baseada no microcontrolador de 8bits,Atmel AVR, com a arquitetura RISC (reduced instruction set computer). Importante ressaltar o aspecto padrão em que são ligados os pinos, permitindo assim, a ligação de módulos expansivos, conhecidos

como Shields. Tal tópico, discutiremos com mais detalhes no Capítulo 3, desse e-book .

A primeira placa foi baseada no Atmega8, o clock de 16MHz e memória flash de 8KB. Mais tarde, surgiram placas que utilizavam o Atmega168, com memória superior ao Atmega8. Hoje, as versões mais recentes, como Duemilanove e Uno, utilizam o Atmega328 com memória flash de 32KB e comutam, automaticamente, a alimentação USB e fonte externa. Caso necessite de mais entradas / saídas e memória, existe o Arduino Mega1280, com memória flash de 128KB e o Arduino Mega2560 (versão mais atual), com memória flash de 256KB.

As placas suportam diversos protocolos de comunicação como: serial, interface periférica (SPI) e I2C. Como recurso padrão em cada placa são inclusos: um conector de programação serial in-circuit (ICSP) e um botão de reset.

Existem vários modelos e cópias alternativas, porém o escopo do nosso e-book é fazer um breve resumo do hardware, com o foco no Arduino UNO, que será descrito com maiores detalhes no próximo Capítulo.

1.2. SOFTWARE

A IDE (Integrated Development Envionment ou Ambiente de Desenvolvimento Integrado) do Arduino é gratuita. Além disso, trata-se de uma aplicação multiplataforma, ou seja, funciona no

Windows, Linux e MAC OS, isto é, você escolhe a melhor opção de acordo com o seu sistema operacional.

É importante você já se familiarizar com a IDE, pois é onde escreverá seus sketchs, que são as instruções de como seu Arduino deve funcionar e o que ele deve fazer. A IDE tem uma interface muito amigável e, ao programar o Arduino, perceberá que ajuda a mascarar e muito a complexidade do hardware. Ao abrir a IDE, a interface é parecida com essa:

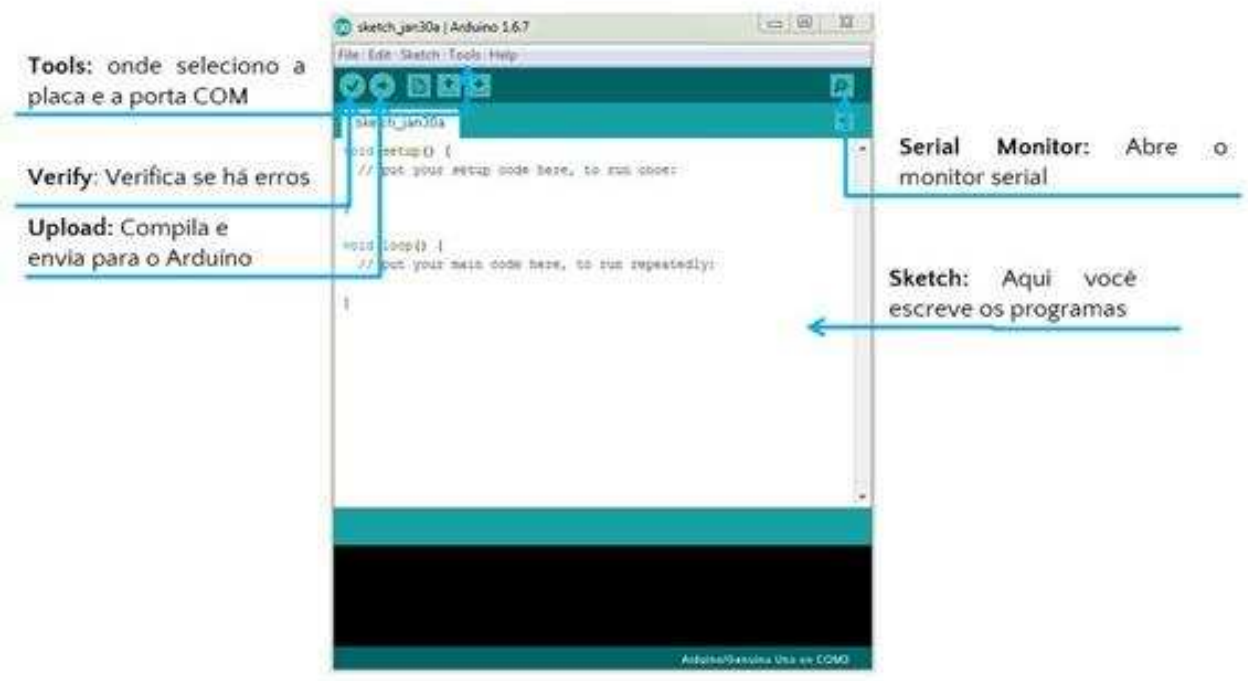


Figura 1.1 - Interface IDE Arduino

1.3. QUE SÃO AS BIBLIOTECAS

De forma resumida, uma biblioteca é um pedaço de software que fornece funcionalidade ao seu projeto, facilitando a codificação. Fazendo uma comparação, é como se você estivesse fazendo um trabalho, então pega um livro e utiliza referencias desse livro no

seu trabalho. Analogamente, você faz isso com as bibliotecas. Existem três tipos diferentes de bibliotecas: essencial(core), padrão e a de terceiros. Vamos entender um pouco sobre:

1.3.1. Biblioteca essencial (core)

Construída dentro da IDE do Arduino principal, ela é fundamental para o devido funcionamento do seu camarada, responsável por mascarar a complexidade do hardware do microcontrolador. Assim, você não precisa lidar diretamente com os registradores. A equipe de desenvolvimento do Arduino estudou muito dos projetos apresentados pelos alunos e, com isso, desenvolveram uma biblioteca para deixar os projetos simples de se executar.

Como na maioria dos casos, lê-se dados de entrada e saída, escreve-se dados de saída. Com a biblioteca essencial, essas tarefas se tornam extremamente simples.

1.3.2. Bibliotecas padrão

Ao baixar e instalar a IDE do Arduino, algumas bibliotecas padrão estão inclusas. Essas bibliotecas são vistas como necessárias pelo time de desenvolvimento do Arduino e que, muitas pessoas utilizarão em seus projetos. Elas não são inseridas por padrão em seus projetos, como a biblioteca essencial, pois o Arduino possui recursos limitados e, incluir automaticamente n bibliotecas, faria você desperdiçar recursos, deixando pouco espaço para seus futuros sketches.

A utilização dessas bibliotecas é bastante simples. Você deve incluí-las de forma explícita em seus sketches, utilizando a instrução

include no topo de seus sketch. Veja um exemplo, usando a biblioteca Softwareserial:

```
#include <Softwareserial.h>
```

Obs.: O nome da biblioteca é delimitado por sinais de < e >. Não se utiliza ponto e vírgula (;) no fim da linha.

Seguem mais alguns exemplos de biblioteca padrão (Arduino libraries):

- Biblioteca LiquidCrystal;
- Biblioteca EEPROM;
- Biblioteca Servo;
- Biblioteca SPI;
- Biblioteca Firmata;

1.3.3. BIBLIOTECA DE TERCEIROS

São disponibilizadas pela comunidade do Arduino, não sendo distribuídas como padrão em conjunto com a IDE, havendo a necessidade de realizar a instalação delas.

Geralmente, utiliza-se bibliotecas de terceiros, quando você trabalha com um determinado sensor. Então, esse sensor possui uma biblioteca, que auxilia a mascarar sua complexidade, algumas servem para adicionar funcionalidades à biblioteca padrão.

Bom, mas como você adiciona uma biblioteca de terceiros em seu projeto? Isso é bastante simples, e pode ser feito de duas formas, basicamente:

A primeira maneira é na própria IDE do Arduino, na opção (*Sketch > Incluir Biblioteca > Gerenciar Bibliotecas ...*):

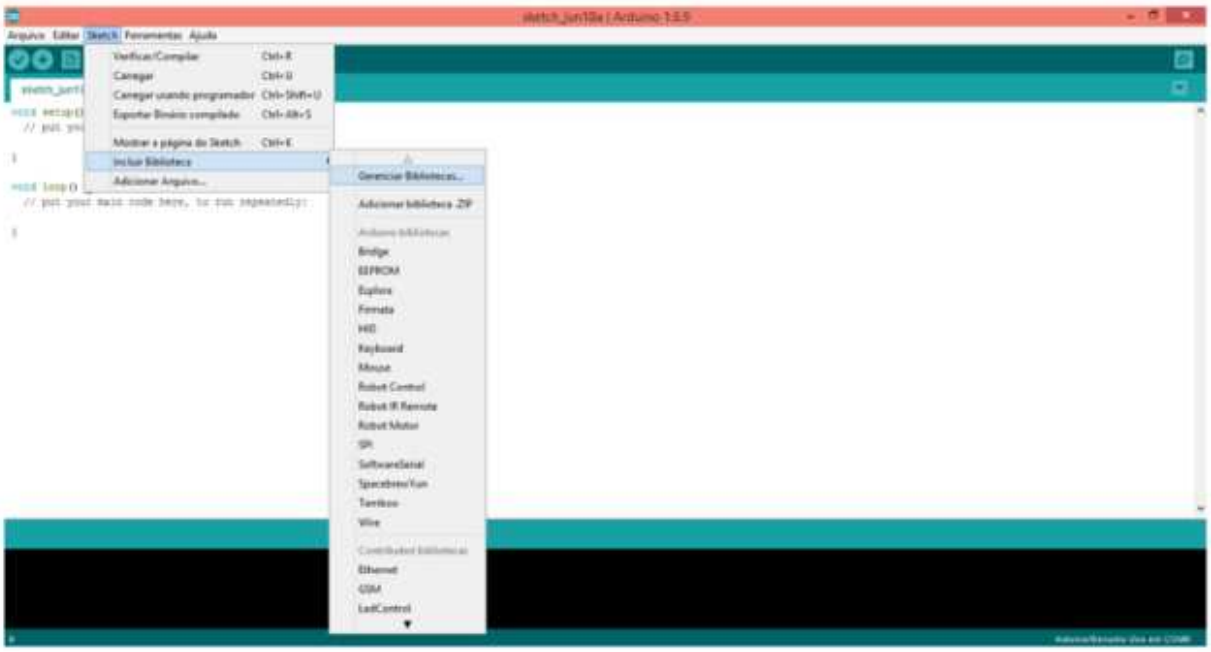


Figura 1.2 - Adicionar biblioteca de terceiros

Feito isso, aparecerá uma janela com diversas bibliotecas, em que você poderá realizar a pesquisa da biblioteca desejada. Como exemplo, utilizaremos a biblioteca DHT(utilizada para trabalhar com sensores de temperatura e umidade DHT11 e DHT22):

Figura 1.3 - Instalação biblioteca auxiliar

É só clicar em Instalar e pronto, a biblioteca estará disponível para uso.

Agora, vamos à segunda maneira. Pessoalmente, eu prefiro fazer dessa forma e creio que grande parte da comunidade também.

Você fará o download de um arquivo .zip. Feito isso, na IDE você seguirá os seguintes passos (*Sketch > Incluir Biblioteca > Adicionar biblioteca .zip*). Então, abrirá uma janela para selecionar o arquivo, ao escolhe-lo, clique em abrir. Com isso, aparecerá uma mensagem na IDE, confirmando a inclusão da biblioteca.

Obs.: *Ao instalar a biblioteca, ela estará disponível para uso em projetos futuros, assim como as bibliotecas padrão.*

Assim,concluímos nossa descrição sobre as bibliotecas, que serão exploradas nos próximos capítulos desse e-book.

1.4. BLINK LED

Diz a lenda que, todos devem começar a programar com o famoso Hello World e, só assim, terá sucesso. No caso do mundo de hardware, deve-se sempre começar pelo clássico exemplo blink led.

A essa altura, você já baixou a IDE e já tem instalada em seu computador, com isso sem delongas vamos começar. O primeiro passo é abrir a IDE do Arduino.Vá na opção (Arquivos > Exemplos > 01.Basics >Blink) e, ao clicar, você verá o seguinte exemplo:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second,
repeatedly.

  Most Arduinos have an on-board LED you can control. On the Uno
and
  Leonardo, it is attached to digital pin 13. If you're unsure what
pin the on-board LED is connected to on your Arduino model, check
the documentation at http://www.arduino.cc

  This example code is in the public domain.

modified 8 May 2014
by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the
board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage
level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the
voltage LOW
  delay(1000);              // wait for a second
}
```

Lista 1.1 - Exemplo Blink Led

1.5. ENTENDENDO O CÓDIGO

void setup: defino o pino 13 como uma saída digital;

void loop: deixa o pino 13 em nível lógico 1 por 1s e deixa o pino 13 em nível lógico 0 por 1s.

Assim, o led fica piscando infinitamente na rotina de loop, enquanto a placa estiver energizada. Em resumo, o fluxograma é apresentado da seguinte maneira:

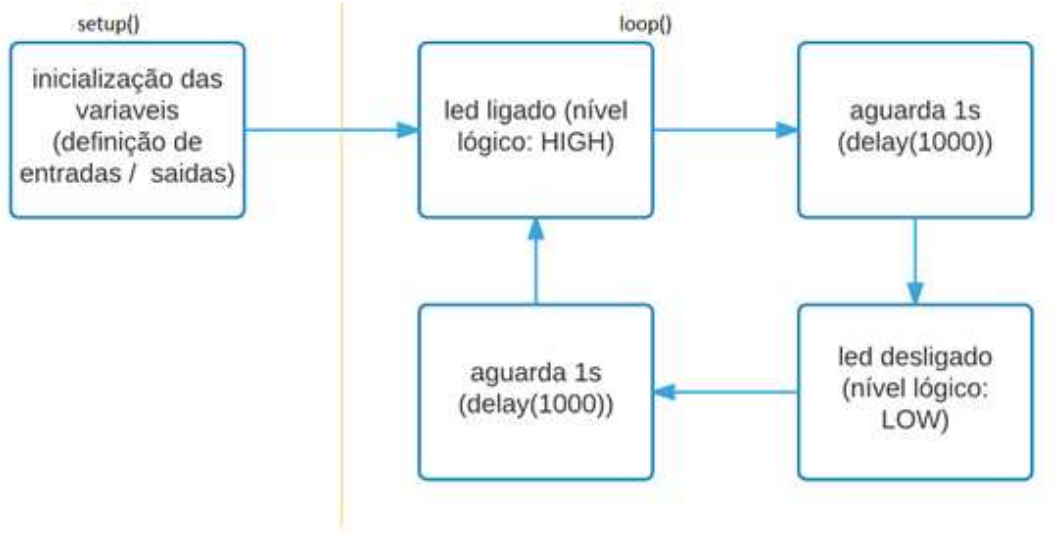


Figura 1.4 - Fluxograma funcionamento blinkled

Por padrão, o pino 13 do Arduino Uno tem um led. Dessa forma, dispensa-se a montagem na Protoboard. Claro que, se você quiser montar, será uma experiência muito interessante e recomendo fortemente.

Para isso, você precisará dos seguintes materiais:

- Protoboard;
- Arduino Uno;
- LED Vermelho ou outra cor de 5mm;
- Resistor de 220Ω;

- Fios jumper.

A montagem ficará conforme a imagem a seguir:

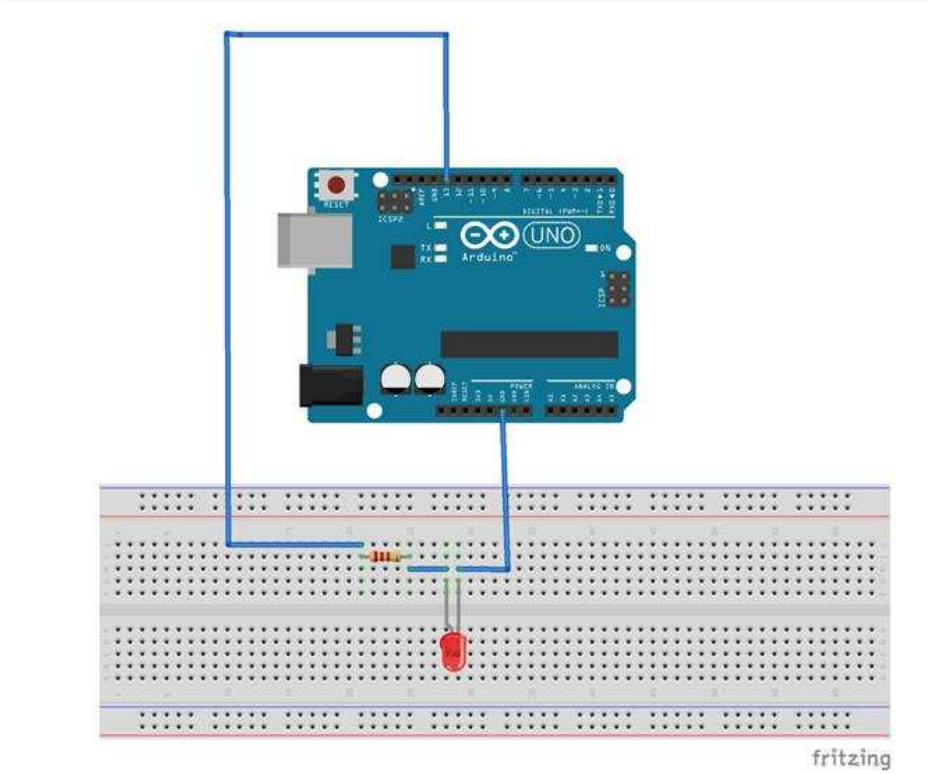


Figura 1.5 - Montagem na protoboard

Nessa montagem, foi utilizado o pino 13 para acionamento do led, o resistor com a função de limitar a corrente no led, para evitar que o mesmo queime e, por fim, do led é ligado um fio jumper para o GND. Após finalizar montagem, plugue o USB no Arduino e no computador, clique em Verificar para checar se há algum erro. Após essa verificação, clique em Carregar para passar o código fonte para a placa, com isso o código é carregado e você já vê a mágica acontecendo :D'

1.6. RESUMO

Parabéns por chegar até aqui! Esse é o Capítulo de introdução, com o objetivo de passar a você a história, o que é o Arduino e mostrar um pouco da IDE .

Vimos como piscar um led, com o clássico Blink Led. Como disse, sem fazer isso, não da pra prosseguir rs. Brincadeiras a parte, conseguimos entender a estrutura de um sketch e um pouco do hardware do Arduino.

O próximo capítulo, explicaremos o hardware do Arduino Uno e a utilização de pinos de entrada e saída.

2. ENTRADAS E SAÍDAS DO ARDUINO UNO

Escolhemos o Arduino Uno para exemplificar suas entradas e saídas, pois é o mais utilizado, considerado a porta de entrada para muitos no universo maker.

A placa já está em sua terceira revisão, onde seu diagrama esquemático, pode ser baixado diretamente no site do Arduino. Além disso, ainda há a disponibilidade de todos os arquivos para edição, já que é um projeto Open Hardware.

O Uno é relativamente pequeno e cabe na sua mão, contém 4 furos para ser fixado em alguma superfície. Veja a seguir, as dimensões do mesmo:

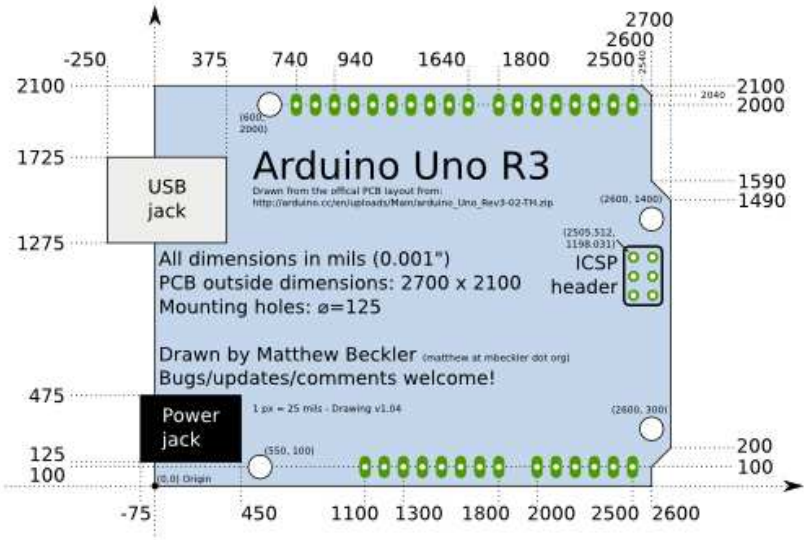


Figura 2.1 - Dimensões PCI - Arduino Uno[1]

Baseado no ATmega328, possui 14 pinos de entrada/saída digital (dos quais, 6 podem ser utilizados como PWM), 6 entradas analógicas, cristal oscilador de 16MHz, conexão USB, entrada para

[1] Fonte <https://blog.arduino.cc/2011/01/05/nice-drawings-of-the-arduino-uno-and-mega-2560/>

fonte de alimentação, cabeçalho ICSP e botão de reset, contendo tudo que é necessário para um microcontrolador funcionar.

Além de suportar os protocolos de comunicação I2C(TWI) e SPI, na IDE do Arduino há uma biblioteca padrão **Wire**, que simplifica a comunicação I2C. Pode ficar tranquilo, pois iremos abordar essa comunicação em um capítulo posterior. Em resumo, suas especificações técnicas são, conforme a tabela:

Microcontrolador	ATmega328
Tensão de Operação	5V
Tensão de Entrada (recomendado)	7-12V
Tensão de Entrada (limite)	5-20V
Entradas/Saídas Digitais	14 (6 podem ser utilizadas como PWM)
Entradas Analógicas	6
Corrente por I/O	20mA
Corrente por I/O para 3.3V	50mA
Memória Flash	32KB (0.5KB utilizado para bootloader)
SRAM	2KB
EEPROM	1KB
Clock	16MHz
Led embutido	Pino 13

Tabela 2.1 - Especificações Técnicas - Arduino UNO

Alguns pinos do UNO, contém funções especiais que é importante citar:

Comunicação Serial: Pino 0 (RX) e pino 1(TX);

Interrupção Externa: Pinos 2 e 3;

PWM: Pinos 3, 5, 6, 9, 10 e 11;

SPI: Pinos 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Permite comunicação SPI, utilizando a biblioteca padrão SPI.

I2C: Pinos A4 (SDA) e A5 (SCL). Permite comunicação I2C, utilizando a biblioteca padrão Wire.

AREF: Tensão de referência para entrada analógica;

O site oficial do Arduino traz o detalhamento de todo o hardware do UNO, além dos outros modelos e todos os arquivos para download. Entretanto, está em inglês (nada como um google translator, que não possa resolver hehe). Abaixo, segue layout da placa do Arduino UNO.

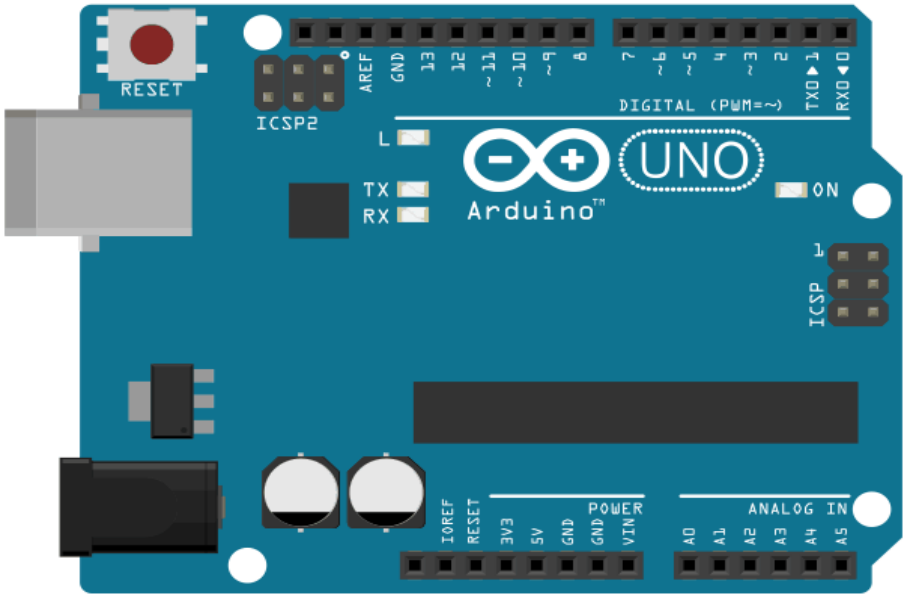


Figura 2.2 - Placa Arduino Uno

2.1. Entradas e Saídas Digitais

Você foi apresentado, brevemente, ao hardware existente do Arduino UNO. É claro que foi apenas uma introdução, já que nosso escopo é lhe apresentar o poder dessa ferramenta e, começar lotando sua cabeça de informação técnica e teórica, seria uma experiência traumatizante.

Creio que, a forma mais interessante de entender algo é explorando suas possibilidades e agregar conhecimento com projetos. Agora, vamos explorar as entradas e saídas do uno com um projeto simples, ok?

Começando com o seguinte projeto: Ao apertar um botão (entrada), o led acende (saída) e, ao pressionar de novo, o led apaga (saída). Com isso, será explorada a leitura de um pino digital e, mediante essa leitura, é acionada a saída. Vamos ao sketch:

```
/*
  Exemplo de Utilização Push-button e LED
  Utilizado: Arduino Uno, Push-button, LED
  Autor: Yhan Christian Souza Silva - Data: 17/11/2016
*/


// --- Hardware ---

#define buttonPin 2
#define ledPin 8

// --- Variáveis ---

int ledState = HIGH,buttonState, lastButtonState = LOW;
unsigned long lastDebounceTime = 0, debounceDelay = 50;

// --- Setup ---

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState);
}

// --- Loop ---

void loop() {
  int reading = digitalRead(buttonPin);
  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) >debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;
      if (buttonState == HIGH) {
        ledState = !ledState;
      }
```

```
    }
  }
  digitalWrite(ledPin, ledState);
  lastButtonState = reading;
}
```

Lista 2.1 - Exemplo Push Button e LED

2.2. ENTENDENDO O CÓDIGO:

pinos: define pino 2 (buttonPin) / pino 8 (ledPin);

variáveis: ledState inicia em nível lógico alto, buttonState para leitura do estado do botão e lastbuttonState variável, para salvar a última leitura do botão, as variáveis do tipo long são para verificação do debounce.

void setup: define buttonPin como entrada digital, define ledPin como saída digital, atribui a ledPin o valor de ledState, iniciando assim em nível lógico alto.

void loop: cria-se a variável reading para atribuir a leitura do botão, verifica se a leitura foi realizada e combate o efeito de bounce do botão, usando a variável lastDebounceTime.

A função millis() retorna o número de milissegundos, desde o início do programa.Cada vez que o botão for pressionado, você atribuirá o valor de millis() à variável lastDebounceTime. Então, compara-se o valor millis() com lastDebounceTime, se for maior que o valor da variável debounceDelay, que é de 50 milissegundos, significa que o botão foi realmente pressionado de novo. Assim, você atualiza o estado do led, invertendo o mesmo, se for HIGH fica LOW e vice-versa.

Obs.: Debounce é uma técnica para evitar oscilações e acionamentos acidentais, prejudicando o funcionamento do programa, pode-se ser aplicado por hardware ou software.

2.3. MONTAGEM DO HARDWARE

Para montar o projeto proposto, são necessários os seguintes materiais:

- Protoboard;
- Arduino Uno;
- LED Vermelho ou outra cor de 5mm;
- Push-button;
- Resistor de 220Ω;
- Resistor de 10kΩ;
- Fios jumper.

A montagem ficará conforme a imagem a seguir:

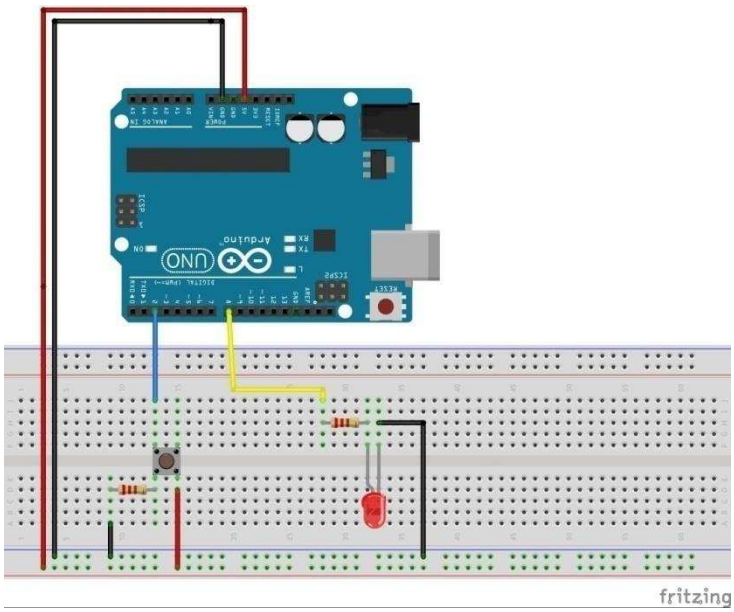


Figura 2.3 - Montagem na protoboard

Assim como no exemplo blink led, o resistor em série com o led tem a função de limitar a corrente do mesmo. Para evitar danos ao componente, com o push button é ligado um resistor de pull-down, para garantir que não haja ruídos, que interfiram no devido funcionamento do circuito.
Após finalizar montagem, plugue o USB no Arduino e no computador, clique em Verificar para checar se há algum erro. Após essa verificação, clique em Carregar para passar o código fonte para a placa. Com isso, o código é carregado e você poderá testar, pressionando o botão: Uma vez, o led mudará de ligado para desligado e, ao pressionar novamente, o led ligará, ou seja, toda vez que pressionar o botão o led mudará o estado lógico.

2.4. CONHEÇA O MUNDO ANALÓGICO

Na seção anterior, apresentamos as entradas e saídas digitais do Arduino, utilizando um botão e um led, que assumem dois estados lógicos, LIGADO ou DESLIGADO. O que fazer, caso queira medir um sinal analógico ou alterar a luminosidade de um led com um potenciômetro?

A leitura de um sinal analógico é realizada, graças as 6 entradas analógicas, que mencionamos na descrição do UNO. Mas a alteração de luminosidade de um LED, por exemplo, não é realizada variando a tensão aplicada a ele e sim, usando modulação por largura de pulso (PWM), que explicarei mais adiante.

Antes de tudo, vamos entender a diferença entre o mundo analógico e digital.

Basicamente, no mundo digital tudo possui apenas dois estados, 0 ou 1 (LIGADO ou DESLIGADO). Sendo assim, o Arduino só faz a

leitura baseada na tensão aplicada no pino, sendo 0V (LOW) e 5V(HIGH). Já no mundo analógico, as coisas possuem um intervalo de valores, como, por exemplo, a temperatura variar em diversos valores, onde estipularemos um máximo e um mínimo. Agora que foi explicada a diferença, vamos explorar um pouco desse novo mundo.

2.5. A LEITURA DE UM POTENCIÔMETRO

Esse, sinceramente, é um projeto mais simples para ilustrar o funcionamento de uma entrada analógica. Além disso, utiliza-se o potenciômetro, que é um componente de baixo custo e muito fácil de encontrar em diferentes lojas de eletroeletrônica de sua região. Confira o sketch abaixo:

```
/*
  Exemplo de Leitura Analógica
  Utilizado: Arduino Uno, Potenciômetro
Autor: Yhan Christian Souza Silva - Data: 17/11/2016
*/

// --- Hardware ---

#define potPin A0

// --- Setup ---

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

void loop() {
  int reading = analogRead(potPin);
  Serial.print("Leitura potenciometro: ");
  Serial.println(reading, DEC);
  delay(1000);
}
```

Lista 2.2 - Leitura analógica de um potenciômetro

Entendendo o código:

pinos: pino A0 é para o potenciômetro;

void setup: habilita comunicação serial;

void loop: cria-se a variável reading, para receber o valor lido pelo pino do potenciômetro, que varia de 0 à 1023 (por conta da resolução de 10 bits de uma entrada analógica no UNO).Exibe o valor em decimal no monitor serial e se aguarda 1s para nova leitura.

O código desse exemplo é, relativamente, simples e puramente didático.

2.6. MONTAGEM DO HARDWARE

Para montar o projeto proposto, são necessários os seguintes materiais:

- Protoboard;
- Arduino Uno;
- Potenciômetro 10kΩ;
- Fios Jumper.

A montagem ficará conforme a imagem a seguir:

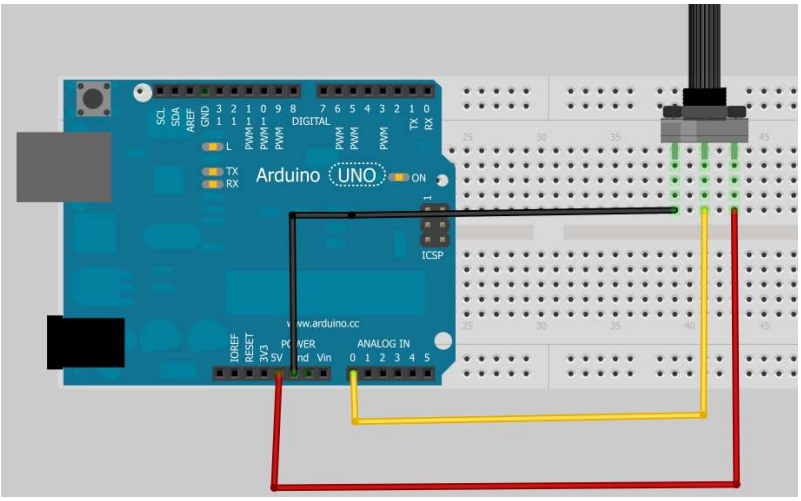


Figura 2.4 - Montagem na protoboard

Tão simples como o código, com o hardware devidamente montado, plugue o USB no Arduino e no computador, clique em Verificar para checar se há algum erro. Após essa verificação, clique em Carregar para passar o código fonte para a placa. Com isso, o código é carregado e então, abra o Monitor Serial e mexa no potenciômetro.Verificará que, em seu mínimo (gire em sentido horário até o fim) o valor será 0 e, ao girar no sentido anti-horário até o fim, o valor será de 1023. Para exemplificar:

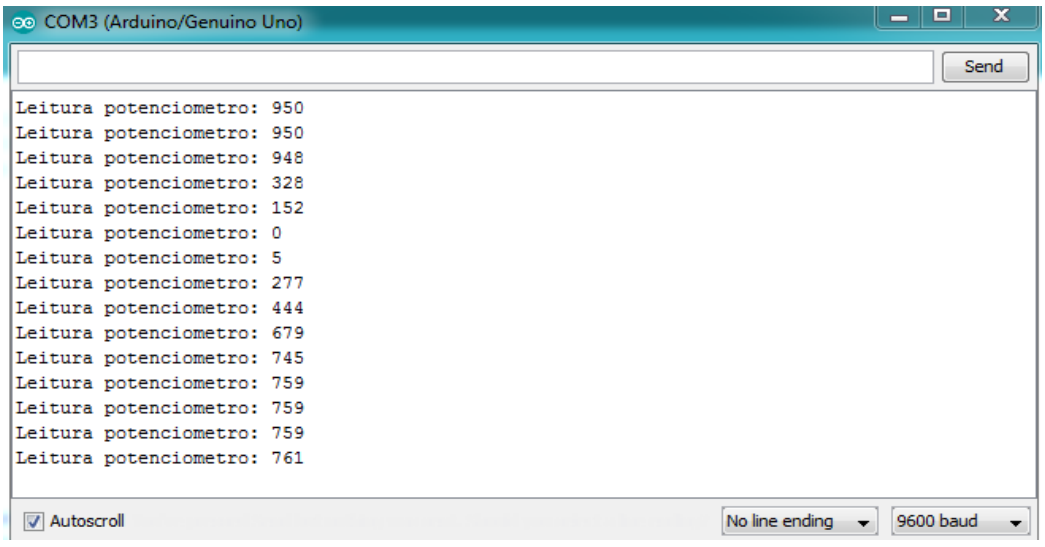


Figura 2.5 - Leitura realizada pelo potenciômetro

2.7. Resumo

Parabéns por concluir mais um Capítulo deste ebook! Após uma introdução sobre o Arduino, conhecemos um pouco do hardware do Arduino UNO, além de suas entradas e saídas.
Vimos como criar sketches, utilizando entradas e saídas digitais com um push-button e led, além de compreender um pouco do mundo analógico e analisar, de forma bem didática, como isso funciona. Desafio você a ir além: Crie sketches e explore esse mundo!
No próximo capítulo,entrarão os sensores, para você criar soluções ainda mais bacanas e aplicar os conhecimentos adquiridos nos dois Capítulos iniciais. Vamos nessa!?

3. SENSORES PARA UTILIZAR EM SEUS PROJETOS.

O que é um sensor e qual sua finalidade? No que ele é aplicado e como é importante aprender sobre o mesmo, para implementar em seus projetos?

A definição mais clássica para sensores é que, são dispositivos que respondem a um estímulo externo físico/químico de maneira específica e mensurável, como por exemplo: pressão, temperatura, umidade, corrente elétrica etc.

Os sensores são muito utilizados em sistemas de controle, para análise de diversos parâmetros de máquinas industriais e, também, nas mais diversas aplicações. Para extrair o máximo deles, é essencial compreender seu real funcionamento e a leitura de seu Datasheet. Ainda, podemos contar com sensores que podem ser lidos de forma digital, (pois possuem um circuito eletrônico para interpretação do sinal e conversão), e outros que devem ser lidos de maneira analógica (em que, através do microcontrolador, realizamos a leitura e interpretação dos valores).

3.1. OS DIFERENTES TIPOS DE SENSORES EXISTENTES

Como mencionado no tópico anterior, nota-se a importância de compreender a aplicação e uso dos sensores. Creio que, em mais 80% dos projetos que você irá desenvolver, seja profissionalmente ou como hobby, utilizará pelo menos 1 sensor, sendo sua leitura analógica ou digital. Hoje, com a plataforma Arduino, a leitura de um sinal vindo de um sensor é extremamente simples, pois geralmente o fabricante concede a biblioteca auxiliar para lidar com o mesmo, assim facilitando a codificação.

Os sensores mais comuns e fáceis de encontrar são: o LM35 (temperatura), DHT11/DHT22 (temperatura e umidade), LDR (luminosidade) e existem muitos outros.

3.1.1. Entendendo o LDR

O LDR (light dependent resistor), ou fotoresistência, é um componente eletrônico passivo, como uma resistência variável, isto é, ela muda conforme a incidência luminosa sobre ele.

O mesmo é muito utilizado em projetos, onde se faz necessário medir a luminosidade.Isto porque, seu custo é baixo e sua facilidade de uso. Pode ser encontrado em: medidores de luz, detentores de incêndio, controladores de iluminação etc.

Figura 3.1 - sensor LDR

3.1.2. Leia o valor do LDR para acionar uma carga

Agora que, você compreendeu um pouco sobre os sensores e sua importância, além de ser apresentado ao LDR, vamos a um exemplo de utilização. A partir dos valores lidos pelo LDR, será

acionada a carga LED de alto brilho, quando o valor estiver dentro das condições que estipularemos no código fonte. Para isso, digite cuidadosamente o sketch abaixo:

```
/*
  Exemplo de LDR, LED de auto brilho
  Utilizado: Arduino Uno, LDR, LED de auto brilho
  Autor: Yhan Christian Souza Silva - Data: 17/11/2016
*/


// -- Hardware --
#define pinLdr 0
#define pinLed 8

// -- Variaveis --
int leituraValor = 0;

// -- Setup --
void setup() {
  pinMode(pinLed, OUTPUT);
  digitalWrite(pinLed, LOW);
}

void loop() {
  leituraValor = analogRead(pinLdr);
  if(leituraValor < 600) {
    digitalWrite(pinLed, HIGH);
    delay(125);
  }
  else  digitalWrite(pinLed,LOW);
  delay(125);
}
```

Lista 3.1 - Leitura de LDR e acionamento de LED

3.1.3. ENTENDENDO O CÓDIGO:

pinos: pinLdr (A0) e pinLed (8);

void setup: define-se o pinLed como saída digital e escreve em sua saída nível lógico baixo (LOW), para garantir que comece desligado.

void loop: A variável leituraValor recebe a leitura analógica do pinLdr, onde cria-se uma condição para acionamento: se o valor for menor que 600 (valor que eu estipulei, pode ser outro), o LEDé

acionado; se não, continua desligado. O delay de 125ms é o intervalo para uma nova leitura do sensor LDR.

3.1.4. MONTAGEM DO HARDWARE

Para montar o projeto proposto, são necessários os seguintes materiais:

- Protoboard;
- Arduino Uno;
- LED de auto brilho 5mm;
- Sensor LDR;
- Resistor de 330Ω;
- Resistor de 10kΩ;
- Fios jumper.

A montagem ficará conforme a imagem a seguir:

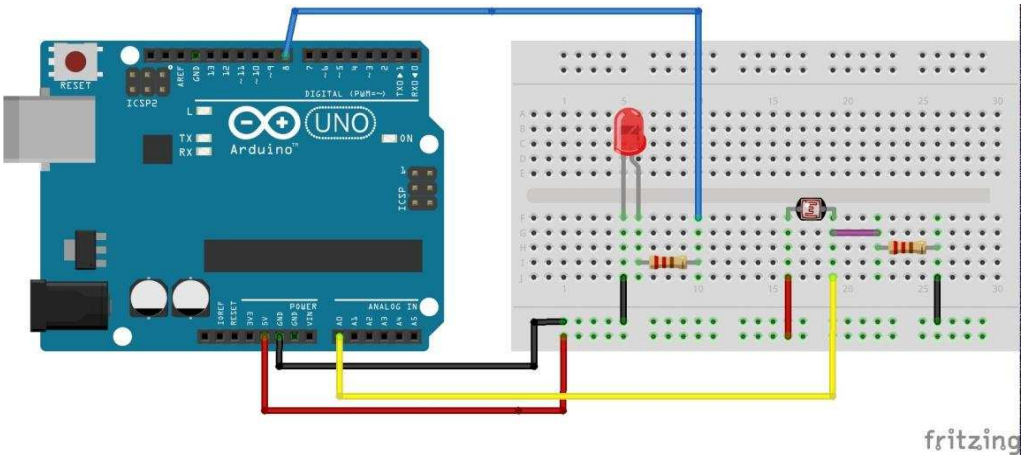


Figura 3.2 - Montagem do circuito na protoboard

O resistor no LED tem a função de limitar a corrente, como já explicado nos exemplos anteriores. O outro resistor ligado em

conjunto com o LDR, funcionará como um divisor de tensão. Após finalizar montagem, plugue o USB no Arduino e no computador, clique em Verificar, para checar se há algum erro. Após essa verificação, clique em Carregar, para passar o código fonte para a placa.Com isso, o código é carregado e você poderá testar, colocando sua mão próxima ao LDR para variar a luminosidade, ou apagar a luz de seu quarto, laboratório etc. Assim o LED acenderá e quando houver maior luminosidade o LED apagará.
Tenho uma observação sobre o valor para acionamento do LED: recomendo que antes de efetuar o teste com o LDR, exibindo seus valores no Monitor Serial e, caso houver necessidade, altere 600 para o valor que se adeque a sua necessidade, fica o desafio hein :).

3.2. RESUMO

Parabéns por concluir mais um Capítulo deste ebook! Este, foi o Capítulo mais breve, pois a intenção foi apresentar o que são sensores e uma aplicação simples com os mesmos. Claro, existem várias aplicações, com uso de bibliotecas e outras ferramentas que vale a pena explorar e desafio vo cê a isso.
No próximo capítulo, que será um tópico mais detalhado, falareidos módulos e shields. Se você não conhece, pode relaxar, que esse tópico será bem explorado logo a seguir. Vamos nessa!?

4. MÓDULOS E SHIELDS PARA INCREMENTAR SEUS PROJETOS.

Os módulos e shields existentes, com certeza, são uma bela mão na roda para seus projetos. Eles ajudam, e muito, a eliminar a complexidade do hardware e você ganha tempo para testar sua ideia e elaborar seu protótipo.Antes de tudo, vamos entender a diferença entre eles.

O módulo é, basicamente, uma plaquinha que contém componentes eletrônicos e, até, alguns circuitos microcontrolados. Para utilizar com o Arduino, por exemplo, existe o módulo relé, que já possui o circuito de proteção contra corrente reversa, o módulo wifi ESP8266, que já contém o circuito para o funcionamento, módulo cartão SD, entre outros. Muitos dos módulos podem ser substituídos pela montagem de uma PCB com os componentes. Em certos casos, a montagem da placa fica até mais barata, mas a sua principal vantagem é a facilidade de uso.

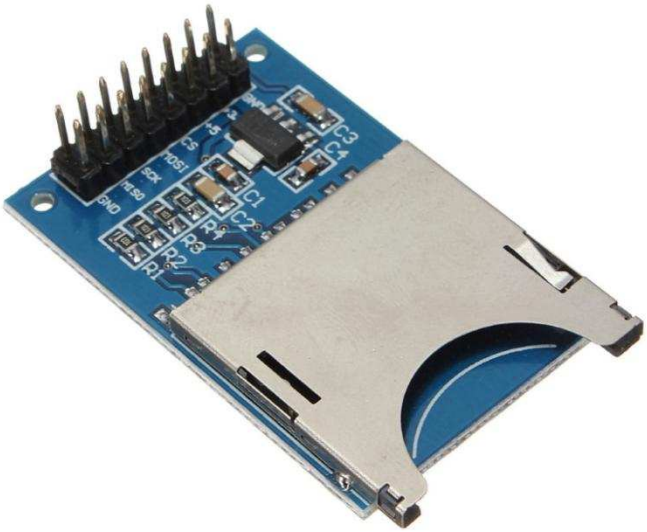


Figura 4.1 - Módulo SD Card

O shield é uma placa de expansão de hardware, que se encaixa no Arduino, fazendo um (sanduíche). A intenção de seu uso é acoplar à placa principal funções, que mesma não tem. Inúmeros shields possuem seus próprios conectores, possibilitando empilhar um sobre o outro. Geralmente, para aproveitar todas suas funcionalidades, é necessária a instalação de bibliotecas adicionais, disponíveis, na maioria das vezes, no website do fabricante. Caso você tenha dúvida sobre como instalar uma biblioteca de terceiros, veja o primeiro Capítulo do Ebook, onde tratamos o assunto.

O shield mais conhecido de todos é o Ethernet Shield, muito utilizado para projetos com Arduino.

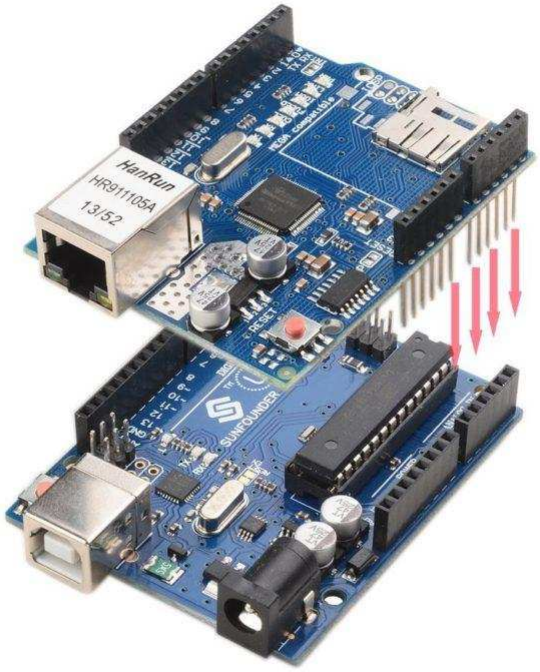


Figura 4.2 - Ethernet shield

Só de visualizar a imagem é possível distinguir o Shield e o Módulo. Um você encaixa na placa principal. Já o outro, você utilizará pinos

do Arduino, para realização de acionamento ou leitura de algum dado.

4.1. Cuidados que devem ser tomados

Ao utilizar um módulo e, principalmente, um shield, alguns cuidados devem ser tomadas, para evitar danos aos seus equipamentos utilizados. Esse cuidado deve ser redobrado, caso você utilize mais de um shield (empilhar um em cima de outro). Isso porque, deve haver compatibilidade de pino e alimentação. Veja um site, que contém uma lista com inúmeros shields e suas características: www.shieldlist.org.

Veja, também, se o shield é compatível com seu Arduino. Alguns, por exemplo, não são compatíveis com o Arduino Mega, outros necessitam de algumas modificações de hardware, para trabalhar com esse modelo.

4.2. Explorando o módulo relé de 2 canais

Esse módulo é um dos mais comuns e fáceis de encontrar nas lojas de venda de componentes no Brasil, além do baixo custo, claro. Esse módulo permite o acionamento de até 2 cargas AC, ou cargas até 30VDC, sendo a corrente máxima suportada de 10A. Possui o circuito de proteção, para evitar problemas com corrente reversa, além de leds indicadores de cada canal. Com esse módulo, você controla lâmpadas, ventiladores, equipamentos eletrônicos, etc.

Figura 4.3 - Módulo relé 2 canais

Veja a pinagem desse módulo:

- JD-VCC: Alimentação relé;
- VCC GND: Permite alimentação de uma fonte externa de 5V;
- GND: Ligar ao Terra do Arduino;
- IN1: Aciona o relé 1, liga-se a um pino de saída do Arduino;
- IN2: Aciona o relé 2, liga-se a um pino de saída do Arduino;
- VCC: Ligar ao VCC do Arduino.

Já do lado dos bornes KRE, a pinagem para cada Relé:

- NC: Normal Fechado;
- K: Comum;
- NO: Normal Aberto.

Para acionar os relés é muito simples: ele é uma saída digital, logo, você define seu estado HIGH ou LOW. Um detalhe importante desse módulo em questão, é que para acionar a carga, deve-se mandar nível lógico baixo, ou seja, LOW.

4.2.1. ACIONANDO CARGAS COM O MÓDULO RELÉ

Como já explicamos, o módulo relé acionará a carga, ao receber o nível lógico baixo, por conta de suas características de construção.No exemplo abaixo, acionaremos duas cargas AC, pode ser duas lâmpadas ou outra carga que desejar. O exemplo é bastante simples: haverão 2 push-buttons, cada um responsável por acionar cada carga. Com isso, vamos utilizar duas entradas e duas saídas em nosso pequeno exemplo, vamos ao sketch:

```
/*
  Exemplo de Utilização Módulo Relé e botões
  Utilizado: Arduino Uno, Push-button, módulo relé 2 canais
  Autor: Yhan Christian Souza Silva - Data: 25/11/2016
*/


// --- Hardware ---

#define btnRelay01 2
#define btnRelay02 3
#define relay01 8
#define relay02 9

// --- Variáveis ---

int buttonState01, buttonState02, relay01State = HIGH, relay02State
= HIGH, lastBtn01State = LOW, lastBtn02State = LOW;
unsigned long lastDebounceTime = 0, debounceDelay = 50;

// --- Setup ---

void setup() {
  pinMode(btnRelay01, INPUT);
  pinMode(btnRelay02, INPUT);
  pinMode(relay01, OUTPUT);
  pinMode(relay02, OUTPUT);
  digitalWrite(relay01, relay01State);
  digitalWrite(relay02, relay02State);
}

// --- Loop ---

void loop() {
  int readRelay01 = digitalRead(btnRelay01);
  int readRelay02 = digitalRead(btnRelay02);
  if (readRelay01 != lastBtn01State) {
    lastDebounceTime = millis();
  }
  if (readRelay02 != lastBtn02State) {
    lastDebounceTime = millis();
  }
```

```
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (readRelay01 != buttonState01) {
      buttonState01 = readRelay01;
      if (buttonState01 == HIGH) {
        relay01State = !relay01State;
      }
    }
    if (readRelay02 != buttonState02) {
      buttonState02 = readRelay02;
      if (buttonState02 == HIGH) {
        relay02State = !relay02State;
      }
    }
  }
  digitalWrite(relay01, relay01State);
  digitalWrite(relay02, relay02State);
  lastBtn01State = readRelay01;
  lastBtn02State = readRelay02;
}
```

Lista 4.1 - Utilização módulo relé 2 canais

4.2.2. ENTENDENDO O CÓDIGO:

pinos: 2 (btnRelay01), 3 (btnRelay02), 8 (relay01) e 9 (relay02);

void setup(): define como entrada o btnRelay01 e btnRelay02, como saída o relay01 atribuindo o estado da variável relay01State, que foi inicializada como HIGH, define como saída o relay02 atribuindo o estado da variável relay02State, que foi inicializado como HIGH. Atribui-se esse valor para garantir que, o estado do relé inicial seja não acionado.Lembrando que, para acionar o mesmo, deve-se levar a nível lógico baixo LOW.

void loop(): cria-se as variáveis readRelay01 e readRelay02 para leitura do estado dos botões, verifica-se se os botões foram acionados, utiliza-se a função millis() para debounce (tópico explicado no Capítulo 2 deste E-book), caso for acionado, altera o estado da variável relay01State caso o btnRelay01 for pressionado, ou altera o estado da variável relay02State caso o btnRelay02 for

pressionado, em ambos os casos inverte-se o estado se for HIGH ficará LOW e vice-versa.

Por fim, o programa será atualizado; atribui-se à relay01 o estado da variável relay01State e ao relay02 o estado da variável relay02State, além de atualizar os valores das variáveis lastBtn01State e lastBtn02State. Se você comparar com o sketch apresentado no Capítulo 2, que lê um push-button e altera o estado de um led, a estrutura é a mesma, apenas adicionando mais uma entrada e saída. Isto é, um relé acionado é uma saída, que pode ser ativada por uma entrada, assim como um led. Bastante simples não é mesmo rs! :)'

4.2.3. MONTAGEM DO HARDWARE

Para montar o projeto proposto, são necessários os seguintes materiais:

- Protoboard;
- Arduino Uno;
- Módulo Relay 2 Canais;
- 2 Lâmpadas 220 ou 110V (poderá usar outra carga AC);
- Cabo Flexível de 2,5 mm²;
- Fios jumper.

A montagem ficará conforme a imagem a seguir:

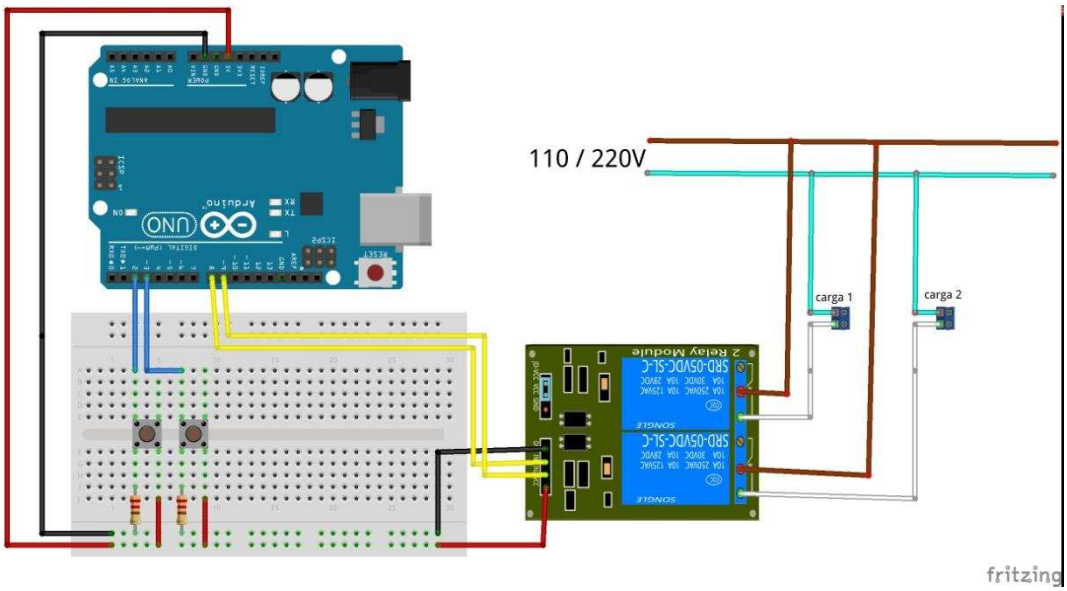


Figura 4.4 - Montagem na protoboard

Há dois push-buttons com os resistores de pull-down, para garantir que fique em nível lógico baixo enquanto não for acionado, nas portas 8 e 9, liga-se os pinos IN1 e IN2 do módulo relé, o pino GND, liga-se ao GND do circuito e o VCC ao VCC do circuito, respectivamente.Perceba que se manteve o jumper em JD-VCC, sem a necessidade de uma fonte externa apenas para o relé.

Agora, na ligação das cargas alternadas, uma fase vai direto ao comum do relé K e outra fase na carga; um fio vai da carga para o contato NO (normalmente aberto) do relé, completando o circuito.

Recomendo, fortemente, que faça todas as ligações com o circuito desenergizado e, após a montagem, evite testar com seu Arduino plugado a uma porta USB em seu computador pessoal.Utilize uma fonte externa, para alimentar o Arduino, creio que em sua casa deve ter um carregador de 12V de algum equipamento. Geralmente, seu modem utiliza essa fonte e conecte a entrada P4 de sua placa.

Antes de ligar para testar, verifique todas as conexões e tenha cuidado. Você estará lidando com tensões de 110 ou 220V, não manuseie jamais a placa com o circuito energizado, pois poderá tomar um belo de um choque. Vale ressaltar também, que faça a montagem em um local reservado e longe de curiosos em sua casa, a fim de evitar pequenos acidentes.

Após de se certificar que tudo está de acordo, ligue o circuito e teste o mesmo.Verificará que, ao pressionar o botão, o relé será acionado (faz um barulhinho bem legal), ligando assim a lâmpada, ou outra carga que decidiu testar.

Obs.: Com 2 metros de cabo flexível de 2,5mm², você consegue realizar a montagem e ainda sobra.Faça cada ligação com um fio de 20cm.

4.3. MÓDULOS E SHIELDS INTERESSANTES DE SE EXPLORAR

Agora que você conheceu com maiores detalhes o módulo relé, que tal pegar seu kit com módulos e shields e começar a explorar o mundo Arduino? Um dos módulos mais interessantes de se trabalhar é o SD Card. Tente implementar um datalogger e outros desafios, um shield que é conhecido e muito utilizado é o Ethernet Shield, bastante usado para projetos de automação e comunicação web com seu Arduino.

Explore os módulos e seja feliz! Apenas tenho uma recomendação:Antes de prosseguir com qualquer codificação, veja se o módulo tem uma biblioteca disponível e observe suas especificações técnicas antes de sair ligando. Alguns, por exemplo, trabalham com níveis de tensão de 3.3V.

4.4. RESUMO

Parabéns por concluir mais um Capítulo deste ebook! Neste Capítulo, descobrimos o que é um shield e um módulo, suas diferenças e algumas especificações para utilizarmos os mesmos. Além disso, foi mostrado como utilizar o módulo mais comum de todos, o relé, com uma aplicação usando os conceitos de entrada e saída do Arduino. Vale lembrar que, devemos nos atentar quando lidamos com o circuito AC.

O último capítulo desse Ebook, tratará sobre as diferentes comunicações existentes como Arduino e explorará alguns recursos interessantes. Para isso, antes, fique craque em compreender os conceitos apresentados nos Capítulos anteriores. Vamos nessa ;) !?

5. COMUNICANDO SEU ARDUINO COM O MUNDO.

O Arduino pode se comunicar com outros dispositivos microcontrolados e alguns sensores. Além disso, também pode se comunicar com o mundo externo, graças a utilização de um módulo ou shield bluetooth, ethernet e wi-fi. Muitos de seus projetos envolverão comunicação, principalmente via Ethernet, Wi-Fi, entre outros. Vamos conhecer um pouco de cada comunicação e alguns dos protocolos, aos quais essa plataforma pode utilizar.

5.1. COMUNICAÇÃO SERIAL

A comunicação mais simples de se implementar com o Arduino, a comunicação serial (UART - Universal asysnchronous receiver / transmitter), possui por padrão no Arduino UNO, os pinos 0 (Rx) e 1 (Tx). Além disso, você pode habilitar outras portas, com a biblioteca Softwareserial, geralmente utilizado quando se precisa enviar as informações serialmente, como um módulo GPS e também o módulo bluetooh. As principais funções da comunicação serial são:

Função	Descrição
available()	Retorna número de bytes disponíveis para leitura
begin()	Configura a taxa de transmissão, baud rate
print()	Escreve na serial texto em formato ASCII
println()	Similar a função print(), com quebra de linha
read()	Lê o byte mais recente
write()	Escreve dados na porta serial

Tabela 5.1 - Funções Serial

Essa comunicação é uma verdadeira mão na roda e ajuda em inúmeros exemplos com sensores. Antes de você implantar um display para exibição dos valores, quer ver se o valor retornado por

um sensor de corrente AC está correto? Mande exibir via serial e abra o serial monitor que ajuda bastante. Além disso, um módulo bluetooth bem famoso, o HC-05, envia as informações e recebe as informações por meio dessa comunicação.

5.2. COMUNICAÇÃO SPI

A comunicação SPI (Serial Peripheral Interface) é um protocolo de dados síncrono, que pode se comunicar com diversos dispositivos, inclusive microcontroladores, rapidamente em pequenas distâncias. Sendo um dispositivo o mestre (geralmente o microcontrolador), controlando outros periféricos, que serão os escravos. Os canais de comunicação, geralmente, são:

- MISO: Entrada mestre, saída escravo;
- MOSI: Entrada escravo, saída mestre;
- SCK: Clock serial.

A função de cada item mencionado acima é bem simples: MISO envia os dados do escravo para o mestre, MOSI envia os dados do mestre para o escravo e o SCK é um pulso de clock, que sincroniza a transmissão de dados. Além disso, existe um quarto pino seletor escravo (SS) em cada periférico, que você poderá usar para selecionar um dispositivo que irá se comunicar. Usando o pino SS, você pode ter múltiplos dispositivos, alternar entre os dispositivos, definindo HIGH para ignorar o mestre e LOW para se comunicar com o mestre. No Arduino Uno, os pinos são:

Pino	Função
10	SS (slave)
11	MOSI
12	MISO
13	SCK

Tabela 5.2 - Arduino Uno: Comunicação SPI

Para facilitar a codificação, a IDE do Arduino tem uma biblioteca padrão denominada SPI, com dois exemplos de aplicação, que valem a pena conferir. As funções da biblioteca SPI são:

Função	Descrição
begin()	Inicializa barramento SPI, definindo pinos SCK,MOSI LOW e SS High
end()	Desabilita comunicação SPI
setBitOrder(order)	Define a ordem na qual os bits são deslocados no barramento SPI
setClockDivider(amt)	Define o divisor de clock de SPI em relação ao clock do mcu, por padrão adota-se um quarto do valor
byte transfer(val)	Transfere um byte através do barramento SPI em ambas as direções
setDataMode(mode)	Define o modo de clock

Tabela 5.3 - Funções biblioteca SPI

Muitos dispositivos utilizam esse protocolo como meio de comunicação, como por exemplo, matrizes de LED, módulo SD e o próprio Ethernet Shield, que utiliza SPI para se comunicar com o Arduino.

5.3. COMUNICAÇÃO I2C

O protocolo serial síncrono I2C, também muito conhecido como TWI (Two Wire Interface), foi criado pela Phillips Semiconductors, no início dos anos 90. Hoje, chama-se NXP Semiconductors. Este protocolo é muito utilizado para transmissões entre dispositivos em baixa velocidade, sendo o barramento composto por dois fios, SDA e SCL, além da alimentação claro. Assim como no SPI, o I2C possui um dispositivo Mestre e outro Escravo. Onde o mestre coordena todos os periféricos escravos.

A função da linha SCL é de ser o clock do barramento e a SDA pela transmissão dos dados. O Arduino UNO tem suporte a essa comunicação, sendo os pinos:

Pino	Função
A4	SDA
A5	SCL

Tabela 5.4 - Arduino Uno: Comunicação I2C

Para facilitar a codificação, a IDE do Arduino tem uma biblioteca padrão denominada Wire, com alguns exemplos de aplicação que valem a pena conferir. As funções da biblioteca Wire são:

Função	Descrição
available()	Retorna o número de bytes disponíveis
begin()	Inicializa barramento I2C
beginTransmission(adress)	Inicia transmissão de dados para o endereço fornecido
endTransmission()	Finaliza transmissão de dados
onReceiver(handler)	Registra uma função a ser chamada, quando um dispositivo escravo recebe uma transmissão de um mestre.
onRequest(handler)	Registrar uma função a ser chamada, quando um mestre solicita dados a partir deste dispositivo escravo.

read()	Lê os bytes transmitidos do escravo para o mestre
requestFrom()	Mestre requisita bytes de um dispositivo escravo
write(value)	Escreve dados do escravo em resposta a requisição do mestre

Tabela 5.3 - Funções biblioteca Wire

Parece bem confusa as descrições das funções da biblioteca Wire...Entretanto na prática, as coisas ficam mais fáceis de compreender. Como mencionei acima, veja os exemplos e tente compreender cada etapa do processo.

Muitos dispositivos utilizam esse protocolo como meio de comunicação, como por exemplo, o módulo RTC DS1307, o módulo I2C utilizado com o display LCD.Outro ponto importante de citar, é que se pode utilizar mais de um dispositivo I2C ao mesmo tempo.Basta, apenas, definir o endereço de cada dispositivo.

5.4. Comunicação One Wire

O protocolo One Wire foi projetado pela Dallas Semiconductor Corp, onde provê dados de baixa velocidade, com o conceito similiar ao I2C, mas com taxas mais baixas de dados e um alcance maior. O diferencial desse protocolo de comunicação é o fato de utilizar, apenas, 1 fio para dados.Para isso, o dispositivo dispõe de um capacitor de 800pF para armazenar carga e alimentar o dispositivo, durante a transmissão de dados.

Inúmeros dispositivos utilizam esse meio para transmissão de seus dados. Um exemplo disso é o iButton, tecnologia utilizada para identificação de usuários, com a vantagem do baixo custo e resistência do material, que possibilita trabalhar em ambientes

hostis. Além disso, sensores utilizam esse meio para transmitir seus dados, um exemplo é o sensor de temperatura DS18B20.

Para facilitar a codificação, a comunidade desenvolveu uma biblioteca denominada One Wire, que vale a pena baixar e instalar. Se você tem dúvida de como baixar e instalar uma biblioteca de terceiro, veja o Capítulo 1 deste E-book, onde expliquei esse processo.

5.5. Protocolo Firmata

O protocolo Firmata permite que seu Arduino se comunique com um host, via software. Isso permite que você programe seu Arduino em outras linguagens, como C#, Javascript, Python entre outras. Uma das vantagens deste protocolo é que um programador, que já trabalha com uma dessas linguagens, poderá programar sem grandes dificuldades. A maior desvantagem é que você tem um host no caminho. Sendo assim, se sua aplicação necessita de resposta rápida, não é recomendado trabalhar com esse protocolo.

A IDE do Arduino, por padrão, traz a biblioteca Firmata, para facilitar ainda mais a comunicação com o software. Para permitir a comunicação, você deve carregar o sketch disponível na biblioteca Firmata, seguindo os seguintes passos: Abra a IDE do Arduino, vá em *Files > Examples > Firmata > StandartFirmata*, clique em Upload, aguarde o sketch ser compilado e carregado para placa.Com isso, seu hardware já estará preparado.

Como o escopo do E-book não é detalhar esse protocolo, bem como suas aplicações, vou deixar aqui alguns links para você pesquisar e, claro, botar em prática para estudo, com algumas linguagens de programação e plataformas disponíveis:

- Python: Pyduino, PyFirmata;
- Javascript: CylonJS, JohnnyFive;
- .NET: Arduino;
- Golang: Gobot;
- Ruby: Artoo;

Além das linguagens citadas acima, com algumas plataformas que listei, existem outras que podem trabalhar com Firmata. Você pode pesquisar e trabalhar em uma linguagem, a qual tem mais familiaridade. Em minha opinião, esse protocolo serve mais como aprendizado, em aplicações mais robustas, creio que seja dispensável.

5.6. Comunicação Bluetooth

A comunicação com dispositivos via bluetooth é uma realidade. E, estamos cercados de todo o tipo de aparato que suporta essa conexão, seja smartphones, fones de ouvido, computadores, entre outros. Essa tecnologia é muito popular e bastante aplicada, hoje em dia. Com os módulos ou shields bluetooth, podemos adicionar essa facilidade ao Arduino e deixar nossos projetos mais interessantes.

Antes de tudo, conheça um pouco dessa tecnologia. Ela foi desenvolvida pela Ericsson, tendo como objetivo, realizar uma comunicação sem fio próxima. Isto é, seu alcance e velocidades são bem limitados. A tecnologia vem sendo aprimorada e, com certeza, hoje é muito aplicada.

Enviar dados de seu Arduino via Bluetooth é muito simples, pois ele se comporta como uma portal serial virtual, ou seja, você utiliza os mesmos comandos de uma comunicação serial normal.

Vamos explorar um pouco dessa tecnologia, em especial o módulo Bluetooth HC-06.A seguir, abordaremos suas características e demonstraremos uma aplicação prática com o mesmo.

5.6.1. Conheça o módulo bluetooth HC-06

O módulo HC-06 é muito utilizado para comunicação via bluetooth com o Arduino, para deixar os projetos mais interessantes e dinâmicos. O alcance do mesmo é de, aproximadamente, 10 metros. Além disso, esse módulo só funciona em modo slave (escravo), ou seja, ele permite que outros dispositivos se conectem a ele, mas não que ele se conecte à outros dispositivos. Sua pinagem é a seguinte:

Pino	Função
VCC	Alimentação VCC 3,6 à 6V
GND	Ground
RX	Comunicação c/ Arduino via Serial
TX	Comunicação c/ Arduino via Serial

Tabela 5.4 - Pinagem módulo bluetooth HC-06

Esse módulo trabalha com 3,3V no pino de sinal (não confunda com a alimentação). E, por conta disso, nas aplicações é utilizado um divisor de tensão, para evitar a queima o módulo.

5.6.2. CONTROLE UM LED RGB VIA BLUETOOTH

Essa aplicação, bastante simples, visa demonstrar como é possível acionar um pino através da comunicação bluetooth. Será utilizado um led RGB que, ao receber os comandos do smartphone pareado ao módulo bluetooth, será acionado. Com isso, perceberá que a comunicação é a mesma que a serial, onde basicamente se lê o que recebe via serial e, a partir desses valores, aciona-se o led com a cor desejada ou desliga o mesmo. Veja com mais detalhes, o código abaixo:

```
/*
  Controle de LED RGB via bluetooth
  Utilizado: Arduino Uno, Led RGB, HC-06
  Autor: Yhan Christian Souza Silva - Data: 01/12/2016
*/

#define redPin 8
#define greenPin 9
#define bluePin 10

char val;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  while (Serial.available() > 0) {
    val = Serial.read();

    switch (val) {
      case 'R':
        digitalWrite(redPin, HIGH);
        Serial.println("LED Vermelho - ligado");
        break;
      case 'r':
        digitalWrite(redPin, LOW);
        Serial.println("LED Vermelho - desligado");
        break;
      case 'G':
        digitalWrite(greenPin, HIGH);
        Serial.println("LED Verde - ligado");
        break;
      case 'g':
        digitalWrite(greenPin, LOW);
        Serial.println("LED Verde - desligado");
        break;
```

```
      case 'B':
        digitalWrite(bluePin, HIGH);
        Serial.println("LED Azul - ligado");
        break;
      case 'b':
        digitalWrite(bluePin, LOW);
        Serial.println("LED Azul - desligado");
break;
    }
  }
}
```

Lista 5.1 - Controle de Led RGB via bluetooth

5.6.3. ENTENDENDO O CÓDIGO:

Pinos e variáveis: 8 (redPin), 9 (greenPin), 10 (bluePin), variável val para receber os dados enviados via serial. O pino TX do módulo bluetooth é ligado ao pino 0 (RX) do Arduino e o pino RX do módulo bluetooth é ligado ao pino 1 (TX) do Arduino

void setup(): define como saída os pinos redPin, greenPin e bluePin, que são os pinos do Led RGB.Inicia-se a comunicação serial com baud rate de 9600.

void loop(): verifica se há dados disponíveis na serial através, atribui-se o valor lido a variável val, onde será verificada as seguintes condições:

Val	Saída
R	Led Vermelho ligado
r	Led Vermelho desligado
G	Led Verde ligado
g	Led Verde desligado
B	Led Azul ligado
b	Led Azul desligado

Tabela 5.5 - Condições para acionamento dos pinos do led RGB

Perceba que o código é relativamente simples. Basicamente, utiliza-se apenas a comunicação serial e pronto :)'

5.6.4. MONTAGEM DO HARDWARE

Para montar o projeto proposto, são necessários os seguintes materiais:

- Protoboard;
- Arduino Uno;
- Módulo Bluetooh HC-06;
- Led RGB;
- Fios jumper.

A montagem ficará conforme a imagem a seguir:

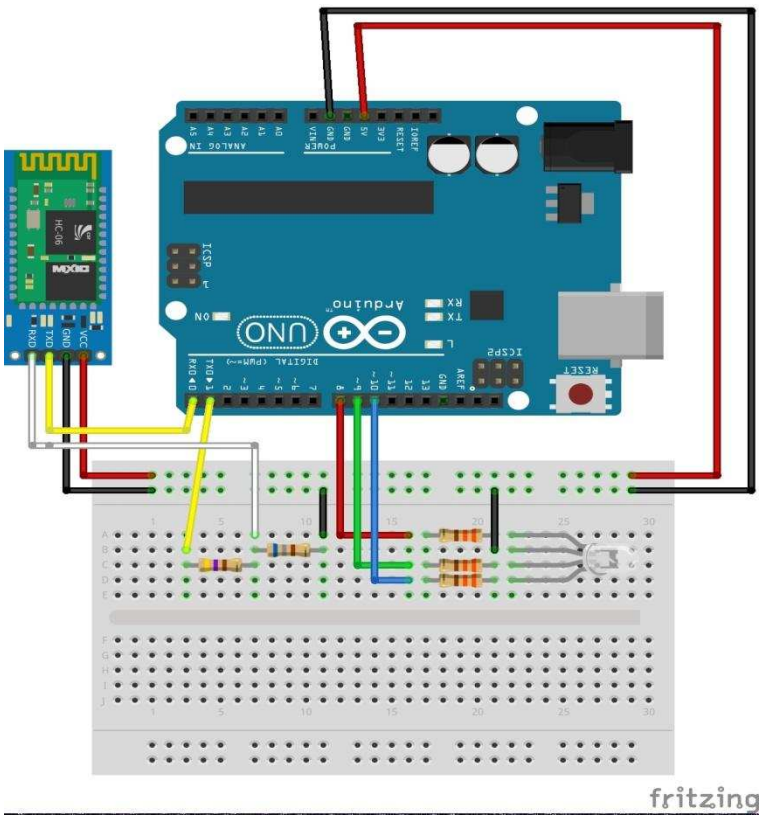


Figura 5.1 - Montagem do Hardware

O resistor nos pinos do LED RGB tem a função de limitar a corrente, como já explicado nos exemplos anteriores. Os resistores, ligados ao módulo bluetooth, tem a função de criar um divisor de tensão e garantir o nível de 3,3V nos pinos de sinal (como explicado acima). Após finalizar montagem, plugue o USB no Arduino e no computador, clique em Verificar, para checar se há algum erro. Após essa verificação, clique em Carregar (antes disso, desconecte o pino RX e TX do módulo bluetooth no Arduino), para passar o código fonte para a placa. Com isso, o código é carregado, então reconecte os pinos RX e TX do módulo bluetooth no Arduino.Agora, precisaremos testar se realmente isso funciona ;)'.

5.6.5. FAÇA O TESTE COM UM APLICATIVO

Faça o teste de funcionamento, utilizando um aplicativo Android. No caso, eu utilizei o Arduino Bluetooh, disponível na PlayStore: https://play.google.com/store/apps/details?id=com.giumig.apps.bluetoothserialmonitor&hl=pt_BR.

Eu gostei desse aplicativo, pois além de fornecer o modo terminal, ele tem outro modos de controle, que lhe auxiliaram em futuros projetos. Além disso, sua interface é bem agradável e simples de mexer, mesmo não sendo o mais leve (já que tem aplicativos só com terminal bluetooh), que não chegam a ter 1MB de tamanho, esse aplicativo é interessante.

Ao instalar o aplicativo, acesse as configurações do seu smartphone e habilite a comunicação Bluetooth. Após isso, veja a lista de dispositivos e faça o pareamento com o módulo HC-06.Geralmente, a senha para pareamento é 1234 ou 0000.

Figura 5.2 - Pareando dispositivo

Feito isso, abra o aplicativo.O mesmo tem a seguinte aparência:

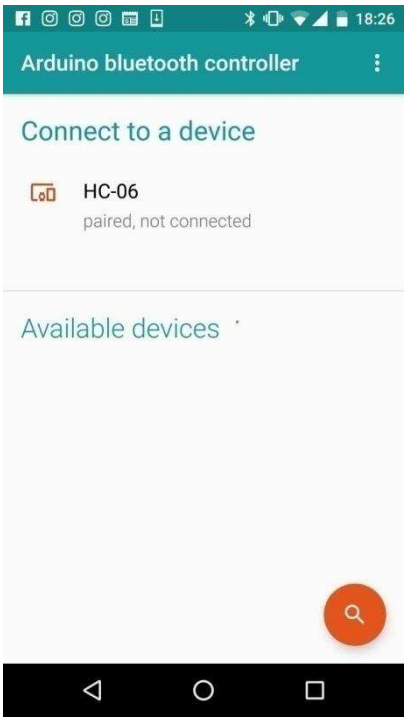


Figura 5.3 - Tela inicial aplicativo

Com o aplicativo aberto, clique sobre o HC-06 e escolha o **modo Terminal**. Ao abrir a tela, clique logo abaixo em **type in command** e digite uma das letras para acionamento dos leds RGB. Perceberá que o LED ligará e, ao mandar o comando para desligar, o mesmo desligará (veja a tabela, logo acima, na explicação do código).

Com os acionamentos sendo realizados, a tela do seu smartphone ficará mais ou menos assim:

Figura 5.4 - Acionamentos via bluetooth

Caso não tenha funcionado, verifique as ligações realizadas, se não há algum mau contato e se certifique que o sketch foi carregado para seu Arduino. Ao funcionar, divirta-se! Acione o vermelho e o azul, fazendo seu Led ficar roxo... Faça combinações de acionamentos e otimize seu código! Coloque mais opções e outros devices. Tente, futuramente, acionar um módulo relé via bluetooth, desafie-se ;)'.

5.7. Outras comunicações

É possível comunicar seu Arduino com a internet, através do Ethernet Shield e, também, módulo Ethernet. Além disso, há o Wi-fi, caso queira trabalhar com internet sem fio, que é mais proveitoso, com o Wi-fi Shield. Hoje, um módulo bem famoso, barato e muito utilizado é o ESP-8266 que, além de ser utilizado como módulo Wi-fi, vem sendo explorado para projetos IoT.

O nosso foco é lhe introduzir as comunicações existentes. E, abordei uma maneira interessante e simples, que é a comunicação bluetooth. Mas como mencionei acima, é possível fazer seu Arduino conversar com a internet, o que deixa ainda mais seus projetos divertidos e de brilhar os olhos de quem vê.

5.8. Resumo

Parabéns, você finalizou o E-book - Começando om Arduino! Espero que o conteúdo tenha lhe auxiliado. E, nessa jornada ao longo de 5 Capítulos, podemos ver bastante coisa como: o que é o Arduino, o que são entradas e saídas, o que é um Sensor, o que é um módulo e Shield e agora, para finalizar, as diferentes comunicações existentes.

O último Capítulo foi o mais longo e com uma abordagem um pouco mais teórica. Desculpe-me, mas creio que seja importante saber o poder que seu Arduino tem. Não detalhei cada tipo de comunicação, pois não era nosso objetivo. Entretanto, consegui passar o que é e, também, o que é possível fazer.

Espero que tenha curtido o E-book...E, agora, mão na massa! Faça projetos, deixe-os mais complexos, pois com o básico, constrói-se coisas magníficas.

Pow, tem alguma dúvida sobre código ? Não sabe o que diabos é um if, else, swich , etc. Separei, para você, 2 apêndices bem legais logo abaixo, indicando bons locais para estudar e aprender mais. E, além disso, um pouco sobre a programação do Arduino.

A. Apêndice - Guia para codificar seu Arduino.

Aqui vai um guia básico para a linguagem do Arduino, como criar seus sketches com sucesso.Para isso, abordaremos os seguintes tópicos:

- Variáveis
- Instruções de Controle
- Loops
- Funções

A.1. Variáveis

Podemos comparar as variáveis a pequenas caixas, cada caixa contém algo. Seu código procurará nas caixas individuais, o que há e poderá alterar seu conteúdo. As vantagens de utilizar variáveis são: elas deixam o código mais fácil de entender e, consequentemente, mais fácil de manter.

Você já viu o quão úteis são as variáveis, deixando o código mais simples e legível.Agora, vamos entender os tipos existentes.

A.1.1. Tipos de variáveis

Existe uma série de tipos de variáveis.Antes de usá-las, é necessário declarar seu nome e tipo, como no exemplo abaixo:

```
char val;
```

Veja, na tabela abaixo, os tipos de variáveis, sua descrição e seu intervalo.

Tipo de variavel	Descrição
byte	Um número de 8 bits com um intervalo de 0 a 255
boolean	Detém dois valores: verdadeiro ou falso
char	Um único caractere ASCII, armazenado como um número de 8 bits
float	Um número que tem ponto decimal, armazenado como um valor de 32 bits, com o intervalo de 3,4028325e+38 à -3,4028325e+38
int	Um número inteiro de 16 bits, com o intervalo de 32.767 à -32.768
long	Um número inteiro longo armazenado como um valor de 32 bits, com o intervalo de 2.147.483.647 à -2.147.483.648
unsigned int	Um número inteiro, sem valores negativos de 16 bits, com o intervalo de 0 à 65.545
unsigned long	Um número inteiro longo, sem valores negativos de 32bits, com o intervalo de 0 à 4.294.967.296
word	Um número sem sinal, armazenado como um valor de 16 bits, com o intervalo de 0 à 65.545

Tabela A.1 - Tipos de Variáveis

A.1.2. ARRAYS

Formam um conjunto de variáveis, que são indexadas como um número. Deve ser declarado com o tipo, antes de ser utilizado, como por exemplo:

```
int vet[4] = {1,2,3,4};
```

O índice de um array começa a partir do 0, ou seja, no exemplo acima no vet[0] o valor do é igual a 1.

Normalmente, são usados para manipulação de valores, dentro de um loop *for*, o qual explicaremos adiante.

A.1.3. STRINGS

São textos que podem ser usados de duas maneiras: ou como um array do tipo char; ou você utiliza a classe String. A vantagem de utilizar a classe String é que contém uma gama de funções, facilitando a manipulação de texto. Isso é muito útil, quando trabalhamos com displays LCD.

A.1.4. CONSTANTES

Como o nome já diz, são variáveis que não alteram seu valor. Ou seja, quando inicializa essa variável e se atribui um valor, o mesmo não pode ser modificado. Exemplificando:

```
const float pi = 3.14;
```

A.1.5. ESCOPO DE VARIÁVEIS

As variáveis podem ser declaradas no topo do sketch, como exemplificado no e-book. Assim, são denominadas como variáveis globais, ou seja, podem ser acessadas de qualquer lugar do sketch. As variáveis declaradas dentro de uma função são denominadas de variáveis locais, só estando disponível à função em que estão declaradas.Por exemplo:

```
const int myLed = 13; //Variável global

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

void loop() {
  const int myLedLocal = 14; //Variável Local
  pinMode(myLedLocal, OUTPUT);
}
```

Creio que demonstrando, ficou mais fácil de entender hehe ;)'.

A.2. INSTRUÇÕES DE CONTROLE

Em inúmeros momentos em seus códigos, você deverá tomar decisões: seja para acionamento de uma carga, um led, entre outros. O seu código de sketch toma decisões, a partir dessas condições, que podem ter apenas dois resultados: verdadeiro ou falso. Em resumo, sempre serão realizados testes, a fim de comprovar se uma condição: é verdadeira e, com isso, tomar um conjunto de medidas; ou falsa e, a partir daí, tomar outro conjunto de medidas.Para isso, vamos entender os operadores relacionais e operadores lógicos, descritos na tabela abaixo:

Operador	Descrição
>	Maior que
>=	Maior ou igual a
==	Igualdade
<	Menor que
<=	Menor ou igual a
!=	Desigualdade
&&	Operador lógico E (AND)
||	Operador lógico OU (OR)
!	Operador lógico NÃO (NOT)

Tabela A.2 - Operadores relacionais e lógicos

A.2.1. IF, ELSE, ELSE IF

A instrução if (se) testa uma condição e verifica, se a mesma for verdadeira, executa determinada ação, conforme programamos. Caso não seja, else (se não), retorna um valor ou executa determinada ação caso o valor seja falso. No último caso, o *else if* é utilizado, quando testamos mais de uma condição. Exemplificando:

```
if( a > b) {
  //então a é maior que b
}

else if (b > a) {
   //então b é maior que a
}

else {
  //nenhuma das condições anteriores é verdadeira logo a = b
}
```

A.2.2. SWITCH CASE

Utilizadas em casos em que, é necessários simplificar as instruções if, else if, sendo simples de entender. Nada mais do que, uma série de comparações caso a caso e, quando a condição é verdadeira, pode-se utilizar a instrução break, para quebrar o laço e não fazer as demais comparações, veja um exemplo:

```
switch (val) {
      case 'R':
        digitalWrite(redPin, HIGH);
        Serial.println("LED Vermelho - ligado");
        break;
      case 'r':
        digitalWrite(redPin, LOW);
        Serial.println("LED Vermelho - desligado");
        break;
      case 'G':
        digitalWrite(greenPin, HIGH);
        Serial.println("LED Verde - ligado");
        break;
      case 'g':
        digitalWrite(greenPin, LOW);
        Serial.println("LED Verde - desligado");
        break;
      case 'B':
        digitalWrite(bluePin, HIGH);
        Serial.println("LED Azul - ligado");
        break;
      case 'b':
        digitalWrite(bluePin, LOW);
        Serial.println("LED Azul - desligado");
        break;
    }
```

A.3. LOOP

Você já conhece o void loop(), rotina o qual o Arduino fica executando, infinitamente, até ser desligado. Mas, existem outras maneiras de criar loops, caso necessite repetir determinada ação, até que uma determinada condição seja verdadeira, ou seja, se essa condição for falsa o loop continuará. Os três loops são: for, while e do while.

A.3.1. FOR

Utilizados para rodar, através de um bloco de código, um determinado número de vezes. São muito utilizados como contador, seja para incrementar um valor, ou decrementar o mesmo. Sua estrutura é a seguinte:

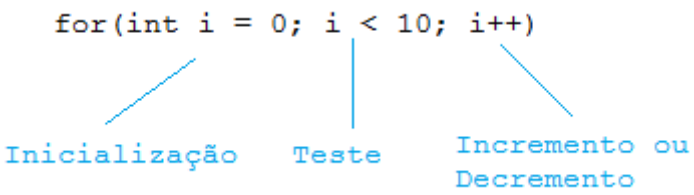


Figura A.1 - Estrutura de um laço for

Inicializa-se a variável i; o teste é realizado a cada vez que o loop é realizado, o i é menor que 10, incrementa-se em 1. Essa rotina continuará até o i ser 10, pois a condição de teste será falso.

A.3.2. WHILE

O while testa uma expressão e vai continuar executando a mesma, enquanto ela for verdadeira. Geralmente é usado, quando você não tem certeza, de quanto tempo o código ficará preso em um

loop. Além disso, é muito usado para testes de entradas de sensores ou botões, veja um exemplo:

```
int sensorValue = 0;
while(sensorValue < 1300) {
  sensorValue = analogRead(analogPin);
}
```

Este código ficará em loop, enquanto o valor de sensorValue for menor que 1300.

A.3.3. DO WHILE

O loop *do while* não é tão utilizado, como os dois citados anteriormente. A diferença entre do while e while é que, ele testa a condição na extremidade do bloco do código, ou seja, ao menos 1 vezes esse bloco será executado.

```
do {
  int sensorValue = 0;
  sensorValue = analogRead(analogPin);
} while(sensorValue < 1300);
```

Assim como nos demais loops, esse bloco ficará rodando, até que a condição seja falsa.

A.4. FUNÇÕES

As funções necessárias para funcionamento do sketch são setup() e loop(). O Arduino simplifica e muito as tarefas, usando funções simples de acessar, para controlar entradas e saídas de dados digitais ou analógicos, assim como funções de tempo, etc.

Mas, você consegue criar suas próprias funções. Elas, geralmente, são usadas quando temos tarefas repetitivas ou cálculos

necessários, em seu código. Para criar uma função, você precisa declarar o tipo da função, dar um nome a ela e, entre parênteses, por os parâmetros dessa função. Ok, é um pouco difícil de entender de primeira rs. Mas,vamos ao exemplo de conversão de temperatura de Fahrenheit para Celsius e retorno desse valor.

```
float calculaTemperatura(float fahrenheit) {
  float celsius;
  celsius = (fahrenheit - 32) / 1.8;
  return celsius;
}
```

Como pode ver, utilizar funções auxiliará bastante na simplificação do seu código.

Obs.: Para realizar comentários em seu código, você pode utilizar "//" para comentar em uma linha, ou iniciar com "/" e finalizar com "*/" no caso de comentar mais de uma linha.*

A.5. RESUMO

Espero que tenha gostado deste apêndice!Vimos algumas coisas da linguagem do Arduino, como as variáveis nos ajudam, o que são operadores relacionais e lógicos, o que são as condições e, por fim, os loops. Espero que lhe ajude esse material, mesmo sem conhecimentos avançados em programação, pode ver como é simples botar sua plaquinha pra funcionar ;)'.

B. Apêndice - Onde aprender mais.

Bom, espero que isso lhe ajude...Separei, para vocês, alguns links que achei úteis, relacionados a Arduino.

B.1. Blog Engenheiro Caiçara

Primeiros passos com Arduino:
http://engenheirocaicara.com/category/primeiros-passos-arduino/

Projeto Leitura de dois sensores via bluetooth:
http://engenheirocaicara.com/series/projeto-do-mes-completo/

Projeto Controle PID de temperatura com Arduino e Scilab:
http://engenheirocaicara.com/series/projeto-do-mes-2/

B.2. Outros links e materiais úteis

Filipe Flop:
http://blog.filipeflop.com/

Renato Aloi (YouTube):
https://www.youtube.com/watch?v=inYEsklZXNE

WR Kits:
https://www.youtube.com/playlist?list=PLZ8dBTV2_5HSyOXhJ77d-iyt5Z_v_1DPM

GBK Robotics:
http://gbkrobotics.com.br/index.php/category/tutoriais/

Embarcados:
http://www.embarcados.com.br/

Considerações Finais...

Com isso, concluímos nosso E-book - Começando com Arduino. Nós, do Engenheiro Caiçara, queremos deixar BEM claro que, nossa intenção, com a criação do E-book é ajudar aquele leitor que, precisa saber o que é Arduino o que é possível fazer com ele, entretanto, não sabe para onde correr. Não somos um fornecedor de cursos on-line, apenas queremos dar um auxílio a você, caro leitor, com nossos conhecimentos, adquiridos ao longo de nossas carreiras acadêmicas e profissionais. Esse trabalho é baseado em alguns artigos do blog, além da explicação de algumas informações, que são confusas para os iniciantes dessa plataforma. Agradecemos, e muito, por você que acompanha nossas postagens no blog e baixou nosso E-book. Ele foi confeccionado com toda dedicação que possuímos, para você ter uma ferramenta de fácil entendimento, sobre o assunto abordado. Então, não deixe de nos acompanhar nas redes sociais e conferir o nosso conteúdo no blog! Afinal, ele é pensado e escrito para você, querido leitor! E sua participação é mais do que importante, para a evolução do nosso conteúdo.

Até a próxima!

Equipe Engenheiro Caiçara.

Não se esqueçam de nos seguir em nossas Redes Sociais...