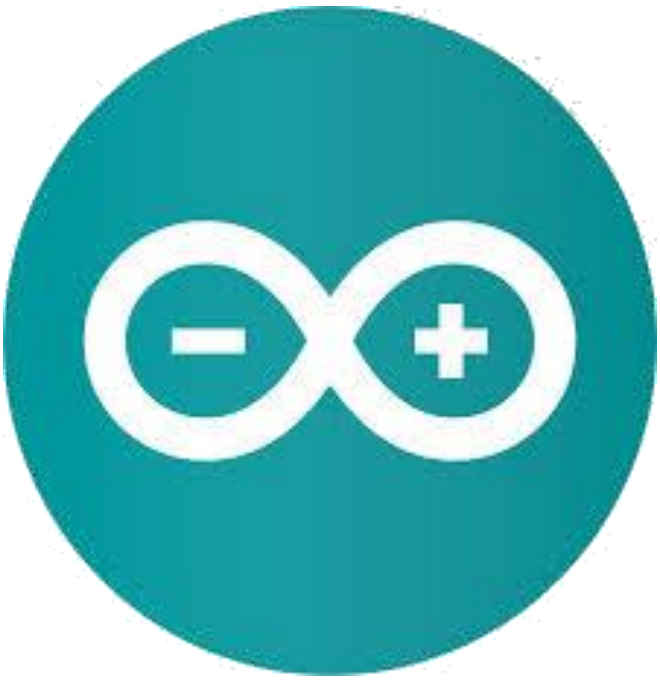
ROBÓTICA EDUCACIONAL



| INICIANTES |

INSTITUTO FEDERAL
Triângulo Mineiro
Campus Uberlândia Centro

ROBÓTICA EDUCACIONAL

2022

2019, 2020, 2022 IFTM Campus Uberlândia Centro
GPETEC - Grupo de Pesquisa em Educação, Tecnologia e Ciências
CRIAR - Clube de Robótica IFTM Arduíno e Raspberry

Projetos Participantes
Extensão: **Facilitando a vida com Arduino**
Extensão: **Usando Robótica Educacional no Ensino Básico**
Extensão: **TICs no CESEU**
Extensão: **Clube de Robótica IFTM em Arduino e Raspberry - Criar**
Extensão: **Iniciação à Robótica no CSEUB**
Extensão: **O Ensino da Robótica Associada à Computação como uma Proposta de Profissionalização de Alunos do Ensino Básico**
Pesquisa: **Pesquisando a Robótica Educacional como Alternativa Didático-Pedagógica**
Pesquisa: **Pesquisando a Robótica no Ambiente Educacional**

Autores
Walteno Martins Parreira Júnior - Professor da Licenciatura em Computação, Coordenador de Projeto de Pesquisa/Extensão e Vice-líder do Grupo de Pesquisa GPETEC
Kenedy Lopes Nogueira - Professor da Licenciatura em Computação e Lider do Grupo de Pesquisa GPETEC
Cristiano Borges dos Santos - Técnico Administrativo e Coordenador de Projeto de Extensão
Abel Antônio Alves - Discente Licenciatura em Computação, Bolsista de Extensão.
Fernando Guimarães Silva - Discente Licenciatura em Computação e Bolsista de Extensão
João Marcos de Oliveira Machado - Discente Licenciatura em Computação, Bolsista de Extensão e Membro do Grupo de Pesquisa GPETEC
Leonardo Carmo Viana - Técnico em Tecnologia da Informação
Marco Antônio Vieira Rodrigues - Discente Sistemas para Internet e Voluntário Projeto Extensão
Marco Aurélio Martins Bessa - Discente Licenciatura em Computação e Bolsista de Extensão
Samuel Oliveira Serqueira - Discente Licenciatura em Computação, Bolsista de Extensão e Membro do Grupo de Pesquisa GPETEC
Vitor Borges Tavares - Técnico Administrativo e Coordenador de Projeto de Extensão

Revisão
De responsabilidade dos autores.

Capa / Arte
Abel Antônio Alves, Leonardo C. Viana, Marcus Vinícius O. Nunes e Samuel O. Serqueira

Dados Internacionais de Catalogação na Publicação

Sumário

Apresentação

Este livro foi desenvolvido no período de 2018 a 2022 por um grupo de discentes que participaram dos projetos de pesquisa e extensão organizados pelo GPETEC (Grupo de Pesquisa em Educação, Tecnologia e Ciências) do Campus Uberlândia Centro com a Coordenação do Professor Kenedy, Professor Walteno e do Técnico Cristiano.

No início foram alguns conceitos e duas ou três experiencias, depois começou a ganhar forma de um tutorial. Várias versões do material instrucional foram desenvolvidas ao longo destes projetos, até que surgiu a ideia de transformar em livro, e que neste momento está sendo disponibilizado à comunidade interna e externa.

É necessário um agradecimento para o Professor Gustavo Prado que durante a sua gestão como Diretor do Campus e ao então Coordenador de Pesquisa, Pós-graduação e Inovação do Campus Professor Ricardo Boaventura que colaboraram e incentivaram a execução dos projetos; e também, neste momento, um agradecimento especial a Professora Lara Kuhn, atual Diretora do Campus, e a Professora Daniela Portes (Coordenadora de Ensino, Pesquisa e Extensão do Campus) que tem apoiado as iniciativas do grupo de pesquisa e incentivado a ampliação das ações extensionistas e de pesquisa.

E não mais importante, agradecer aos alunos, voluntários e bolsistas, que participaram dos projetos executados e nem todos tiveram a oportunidade de contribuir com a elaboração deste texto, mas que utilizaram e permitiram a validação das experiencias.

A importância deste trabalho é de estimular a utilização da robótica na sala de aula com discentes de todas as idades e permitir que estes recursos possam ser utilizados em qualquer escola pública ou privada. Não foram utilizados recursos tecnológicos de alto valor exatamente para que as experiencias possam ser replicadas facilmente.

Agora, o desafio é organizar o volume dois com experiencias mais complexas e considerando a utilização com conceitos de disciplinas propedêuticas.

Walteno Martins Parreira Júnior

1.0 Introdução

Algumas definições necessárias para entendimento do assunto.

1.1 O que é um Robô?

São sistemas mecatrônicos (eletromecânicos) reprogramáveis. Facilitam ou substituem vários tipos de tarefas humanas; as repetitivas ou aquelas que colocam em risco a vida humana. São máquinas projetadas para exercer alguma função que auxilia o trabalho humano.

1.2 O que é Robótica?

A robótica é a ciência que estuda a elaboração, montagem e programação de maquinas para execução de tarefas de forma automática, ou seja, os robôs. Envolve Engenharia Mecânica, Engenharia Elétrica, Engenharia Eletrônica e Engenharia da Computação.

1.3 O que é Robótica Educacional?

É uma metodologia de ensino que tem como objetivo fomentar no aluno a investigação e materialização dos conceitos aprendidos no conteúdo curricular. Vai muito além da construção de projetos e programação de robôs. Proporciona um aprendizado prático que desenvolve no aluno a capacidade de pensar e achar soluções aos desafios propostos. Incentiva o trabalho em grupo, a cooperação, planejamento, pesquisa, tomada de decisões, definição de ações, promove o diálogo e o respeito a diferentes opiniões.

1.4 O que é Computador?

O computador é uma máquina que processa informações eletronicamente, na forma de dados e pode ser programado para as mais diversas tarefas. As fases do processamento são: Entrada de Dados (Informações iniciais), Processamento (Instruções), Saída de Dados (Resultados).

1.5 Programa de Computador o que é?

Um programa de computador, ou software, é uma sequência de instruções enviadas para o computador. Cada tipo de microprocessador entende um conjunto de instruções diferente, ou seja, o seu próprio "idioma" - linguagem de máquina. As linguagens de máquina são as únicas linguagens que os computadores entendem, são muito difíceis para os seres humanos entenderem. Para facilitar a compreensão existe a linguagem de programação. No caso de sistemas como o Arduino (os chamados sistemas embarcados), o software que roda no microprocessador é também chamado de firmware.

1.6 Linguagem de Programação?

Há a necessidade de converter as ideias para a forma que os computadores possam processar, ou seja, a linguagem de máquina. Os computadores de hoje (ainda) não conseguem entender a linguagem natural usada no dia a dia, então é preciso de um outro "idioma" especial para instruir o computador a fazer as tarefas desejada. Esse "idioma" é uma linguagem de programação, e na verdade existem muitas delas. Essas linguagens de programação também são chamadas de linguagens de programação de alto nível.

1.7 Algoritmo (Programa) é?

Um algoritmo, ou simplesmente programa, é uma forma de dizer para um computador o que ele deve fazer. Os algoritmos normalmente são escritos em linguagens de programação de alto nível, isso se aplica a praticamente qualquer computador.

Um programa é composto de uma sequência de comandos, normalmente escritos em um arquivo de texto.

1.8 O que é um Arduino

O Arduino consiste em uma plataforma de prototipagem em eletrônica, elaborado por Massimo Banzi e David Cuartielles em 2005 na Itália, e tem como objetivo facilitar o desenvolvimento de projetos, desde os mais simples aos mais complexos. Com esta plataforma é possível controlar diversos sensores, motores, leds, dentre vários outros componentes eletrônicos.

Um ponto forte sobre o Arduino, é que todo material disponibilizado pelo fabricante, como a IDE de desenvolvimento, bibliotecas e até mesmo o projeto eletrônico das placas são open-source, ou seja, é permitida a utilização e reprodução sem restrição sobre os direitos autorais dos idealizadores do projeto. Porém o nome Arduino, o logotipo, assim como o design gráfico de suas placas são registrados e protegidos por direitos autorais. Saiba mais acessando página oficial do fabricante.

O Projeto Arduino une Hardware e Software, e resulta em uma plataforma de fácil desenvolvimento utilizando um micro controlador.

2.0 Software Arduino

Figura 1 – Arduino Fonte: Acervo pessoal João Marcos (2022)

É necessário, ter em mente que se precisa de:

- ✓ Um computador (Windows, Mac ou Linux);
- ✓ Uma placa Arduino ou compatível;
- ✓ O cabo USB apropriado para a placa escolhida;
- ✓ Acesso à internet

2.1 IDE - Baixando e Instalando

O primeiro passo para começar a trabalhar com o Arduino é instalar a versão mais atual do ambiente de desenvolvimento (IDE), em que todos os códigos serão desenvolvidos. O ambiente de desenvolvimento pode ser baixado gratuitamente no site do Arduino na aba software (https://www.arduino.cc/en/software).

O procedimento de instalação do ambiente de desenvolvimento (IDE) do Arduíno é diferente para cada sistema operacional.

No MacOSX o procedimento é diferente. Após o download da IDE deve-se copia-lo para o aplicativo "Arduino.app" em seu computador (sugestão para que a pasta de aplicativos fique padronizada).

No Linux talvez a maneira fácil é instalar via terminal, no site encontre os arquivos para instalação Linux.

No Windows, após baixar a IDE especifica basta seguir o procedimento de instalação intuitivamente. Sempre utilize a IDE mais recente. Começar a programar é relativamente fácil, basta clicar no ícone “Arduino” no sistema operacional que estiver usando.

Nas imagens a baixo está o exemplo de tela para download do software Arduino, atualizado, na época da elaboração desse material.

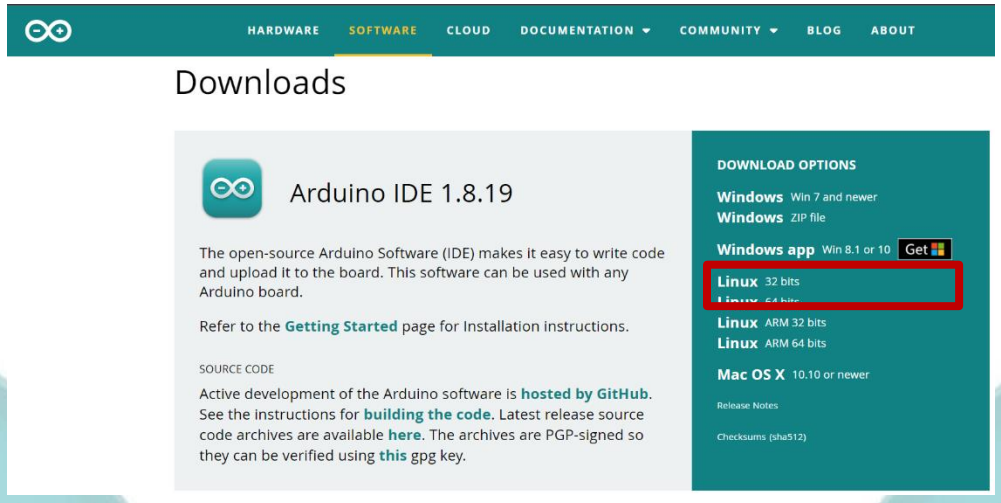


Figura 2 - Página com o programa para download - Fonte: https://www.arduino.cc/en/software (2022)

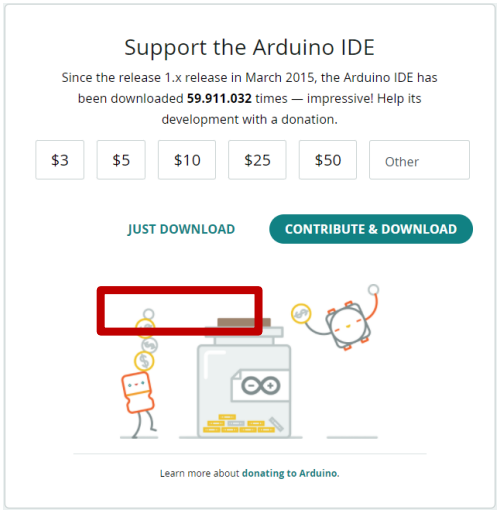


Figura 2 - Continuação da ação para o download Fonte: https://www.arduino.cc/en/software (2022)

Arduino IDE é um software de código aberto (open source), ou seja, todo o seu código fonte está disponível para utilização e pode ser adaptado para diferentes fins. Todas as vezes que se faz um download na página oficial do software, uma tela aparece solicitando uma doação para ajudar a acelerar o desenvolvimento do projeto. Não é obrigatório doar para baixar a IDE, basta clicar em **JUST DOWNLOAD** para fazer o download sem custo algum.

2.2 Configurando

Após a instalação, abra a Arduino IDE para se acostumar com o ambiente de desenvolvimento que te acompanhará ao longo dessa apostila. Deve-se visualizar uma janela similar a figura 4:

1. Verificar: compila e aprova o código. O compilador detectará erros de

sintaxe (como ponto e vírgula ou parênteses faltantes);

2. Carregar: envia o código para a placa Arduino;Novo: abre uma nova janela de código;
3. Abrir: permite abrir um esboço existente;
4. Salvar: salva o esboço atualmente ativo;
5. Monitor serial: abre uma janela que exibirá qualquer informação serial que a placa Arduino está transmitindo para o computador;
6. Nome do esboço: mostra o nome do esboço no qual está trabalhando atualmente;
7. Área do código: esta é a área onde compõe o código para o esboço;
8. Área de mensagens: é onde a IDE diz se houve algum erro no código ou se o código foi compilado e carregado corretamente;
9. Console de texto: mostra mensagens de erro completas. O console de texto é muito útil para a depuração;
10. Placa e porta serial: mostra a placa e a porta serial selecionadas.

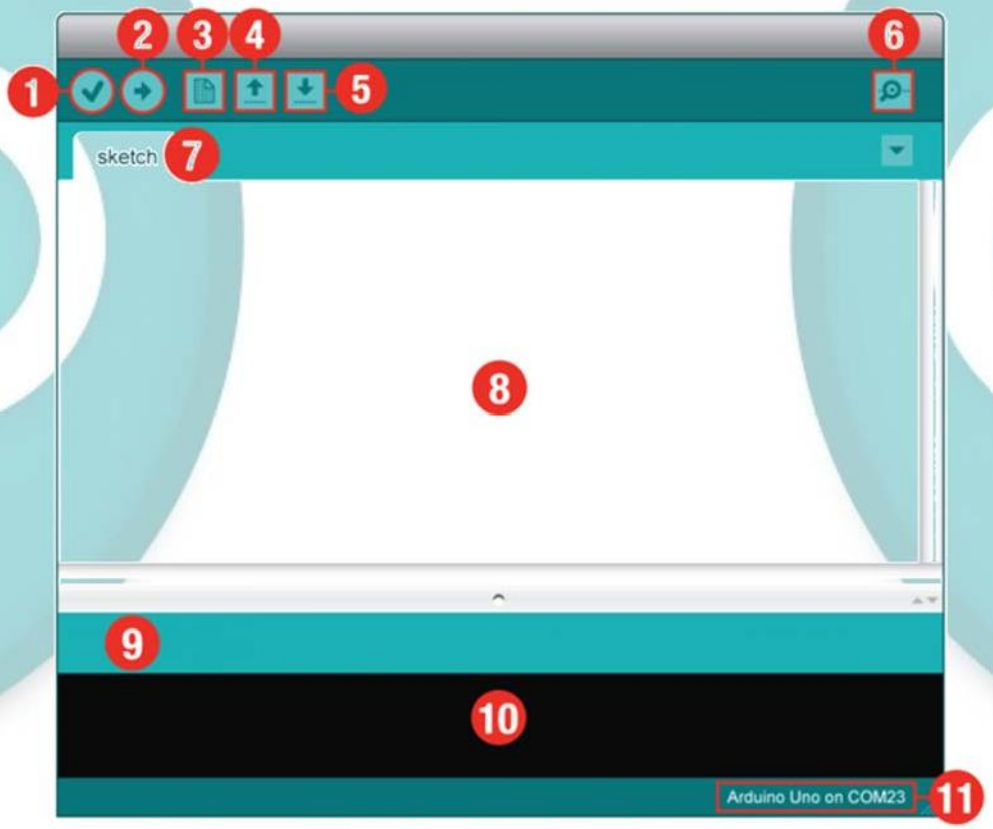


Figura 4 - IDE Arduino - Fonte: www.sparkfun.com

Após a instalação da IDE no computador, cabo USB presente no kit conectado na placa e em uma porta USB do computador.

A placa deve ser reconhecida automaticamente pelo computador, uma vez que a IDE já foi instalada, caso o sistema operacional não reconheça a placa automaticamente, veja nas páginas a seguir como proceder em cada caso.

Quando o computador reconhece a placa Arduino, uma porta de comunicação é criada para que a IDE consiga se comunicar com a placa.

No Windows pode verificar a porta criada acessando o Gerenciador de Dispositivos do sistema. Esta porta tem o nome COMx, onde x é um número, e esta porta deverá ser selecionada no IDE do Arduino, no menu Ferramentas > Porta. No Mac esta porta terá um nome como /dev/tty.usbmodemX, onde novamente o x representa um número específico para a porta criada. Ainda no menu Ferramentas, deverá optar pela sua placa Arduino em Placas.

Selecione tanto a porta de comunicação quanto a placa. Usando a sequência para configurar o programa: Ferramentas → Placa → Arduino/Genuino Uno (Ou outra placa conforme a que estiver usando), veja na figura 5.

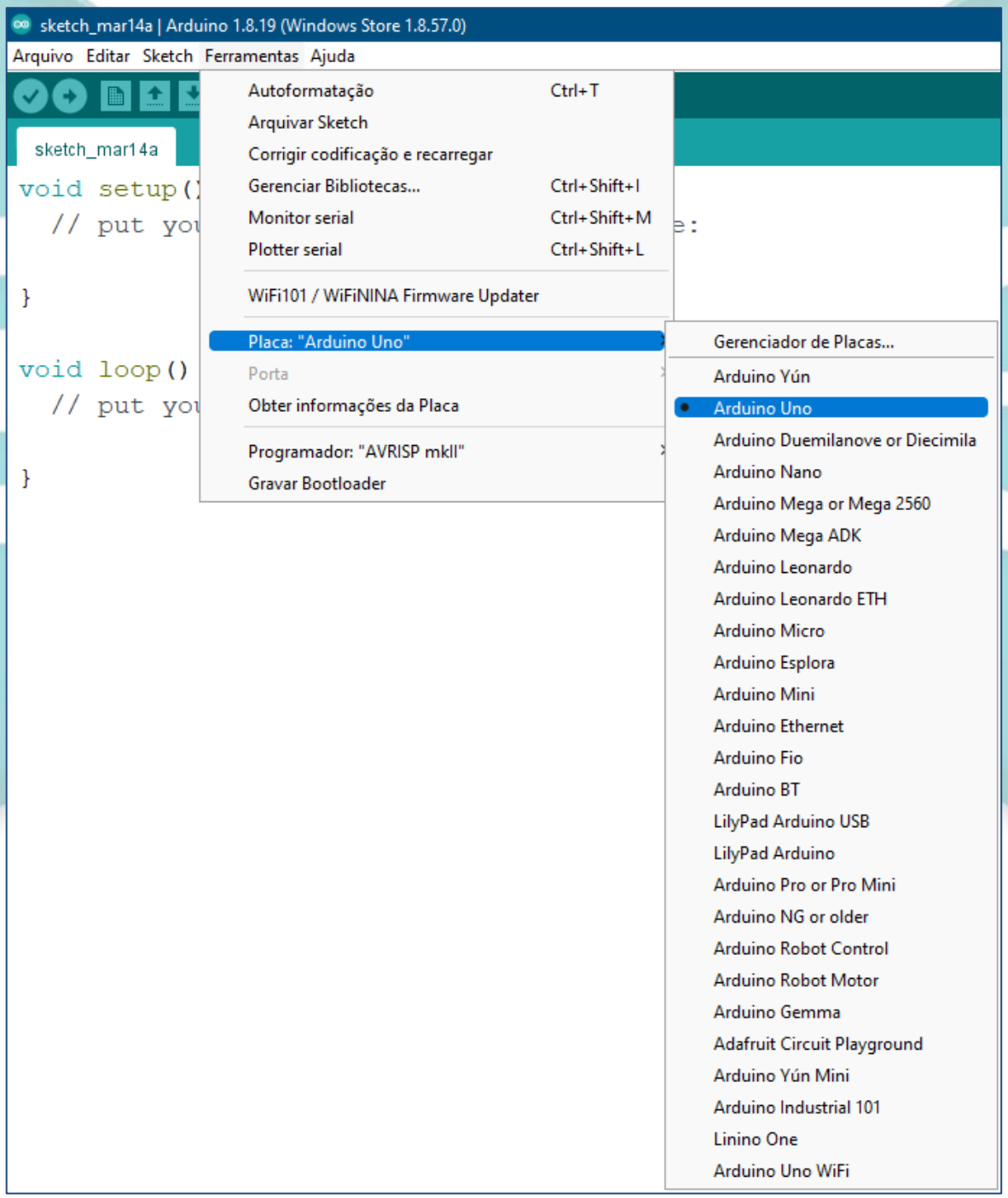


Figura 5 - Configurando a placa no IDE Fonte: acervo pessoal João Marcos (2022)

Para configurar a porta de comunicação (o Arduino deve estar conectado), siga a sequência: Ferramentas → Porta → COMX (O "X" é o número da porta que estiver usando), conforme a figura 6.

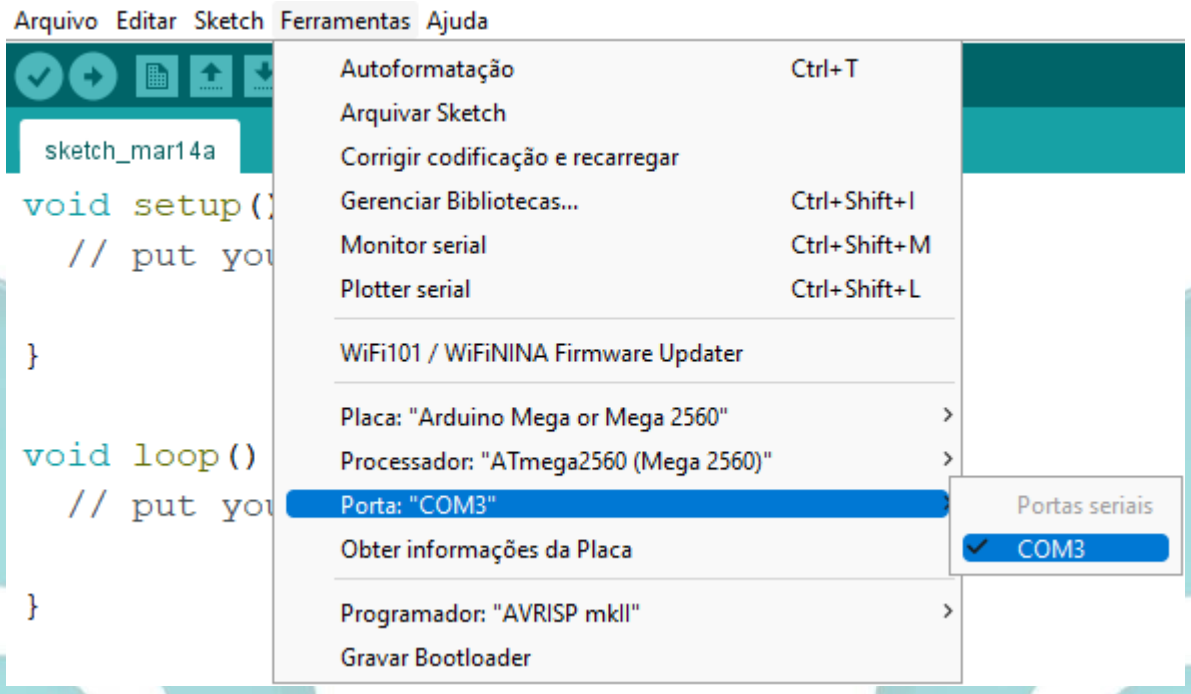
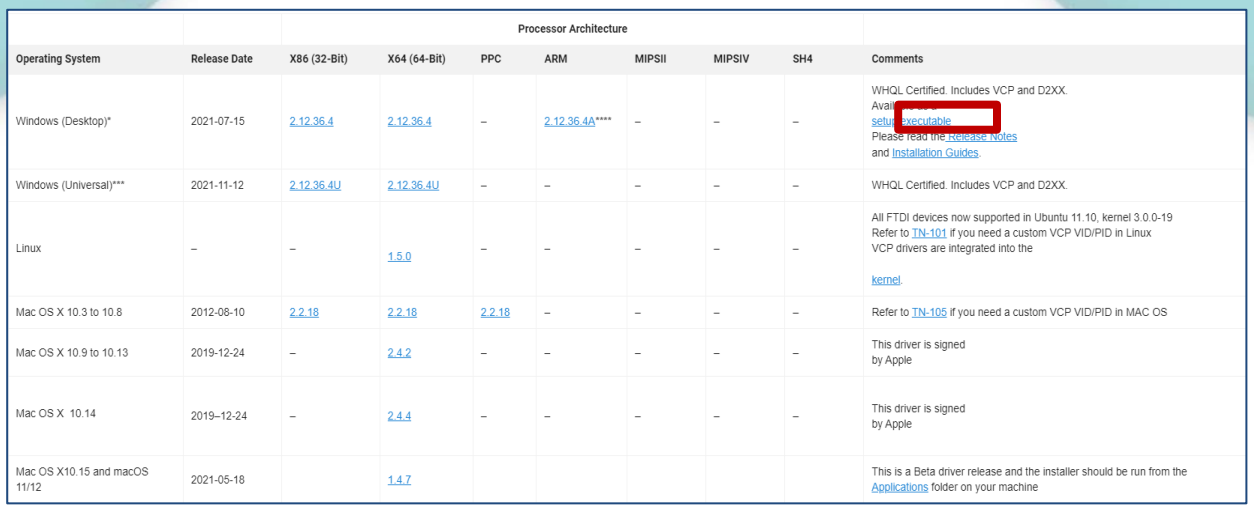


Figura 6 - Configurando a porta no software Fonte: acervo pessoal João Marcos (2022)

2.3 Instalando o driver da placa

Os sistemas operacionais Linux e MacOS vêm de fábrica com o driver instalado, portanto antes de prosseguir tente conectar a placa no seu computador e verificar se ela foi reconhecida corretamente.

A instalação é bem simples, basta acessar o seguinte link: https://ftdichip.com/drivers/vcp-drivers/ e escolher a versão de seu sistema operacional. Para usuários de Windows, recomendamos baixar a versão executável conforme imagem abaixo. É recomendado executar o programa baixado como administrador (botão direito do mouse > executar como administrador).



		Processor Architecture							
Operating System	Release Date	X86 (32-Bit)	X64 (64-Bit)	PPC	ARM	MIPSII	MIPSIV	SH4	Comments
Windows (Desktop)*	2021-07-15	2.12.36.4	2.12.36.4	–	2.12.36.4A****	–	–	–	WHQL Certified. Includes VCP and D2XX. Avail… setup executable Please read the Release Notes and Installation Guides.
Windows (Universal)***	2021-11-12	2.12.36.4U	2.12.36.4U	–	–	–	–	–	WHQL Certified. Includes VCP and D2XX.
Linux	–	–	1.5.0	–	–	–	–	–	All FTDI devices now supported in Ubuntu 11.10, kernel 3.0.0-19 Refer to TN-101 if you need a custom VCP VID/PID in Linux VCP drivers are integrated into the kernel.
Mac OS X 10.3 to 10.8	2012-08-10	2.2.18	2.2.18	2.2.18	–	–	–	–	Refer to TN-105 if you need a custom VCP VID/PID in MAC OS
Mac OS X 10.9 to 10.13	2019-12-24	–	2.4.2	–	–	–	–	–	This driver is signed by Apple
Mac OS X 10.14	2019–12-24	–	2.4.4	–	–	–	–	–	This driver is signed by Apple
Mac OS X10.15 and macOS 11/12	2021-05-18		1.4.7						This is a Beta driver release and the installer should be run from the Applications folder on your machine

Figura 7 - Pagina para download de Driver - Fonte: https://ftdichip.com/drivers/vcp-drivers/ (2022)

Após a instalação, basta conectar a placa na porta USB e ir em Gerenciador de Dispositivos e verificar qual a porta COM criada. Veja na imagem abaixo como sua placa aparecerá caso o driver tenha sido instalado corretamente:

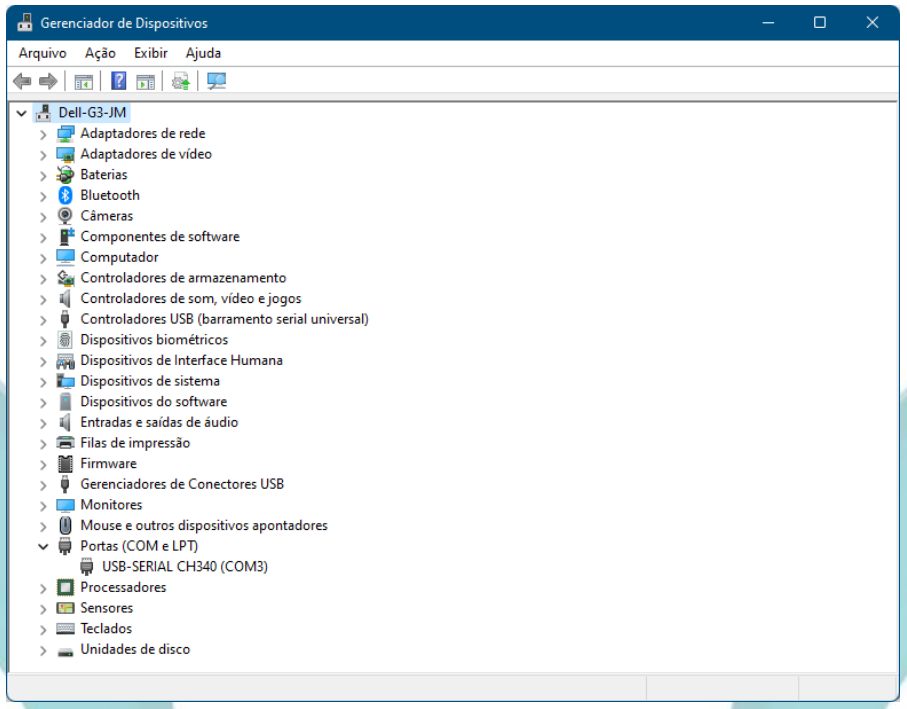


Figura 8 - Gerenciador de Dispositivo – Acervo pessoal João Marcos (2022)

Repare que na imagem foi criada a porta COM3, mas a cada porta que for sendo criada este número irá aumentar. Agora é só abrir o IDE selecionar a placa Arduino UNO, a porta COM e sua placa está pronta para ser utilizada.

2.4 Componentes básicos para utilizar na robótica

2.4.1 Arduino

O Arduino é um computador como qualquer outro, possuindo:

- Microprocessador (responsável pelos cálculos e tomada de decisão)
- Memória RAM (utilizada para guardar dados e instruções, volátil)Memória flash (utilizada para guardar o software, não volátil)
- Temporizadores (timers)
- Contadores
- Clock, e etc.

Ou seja, é um computador, porém em menor escala. Possui menos memória e menor poder de processamento. O Arduino Uno, por exemplo, possui as seguintes especificações:

Micro controlador: ATmega328

- Portas Digitais: 14
- Portas Analógicas: 6

- Memória Flash: 32 KB (0,5 KB usado no bootloader[2])
- SRAM: 2 KB
- EEPROM: 1 KB
- Velocidade do Clock: 16MHz

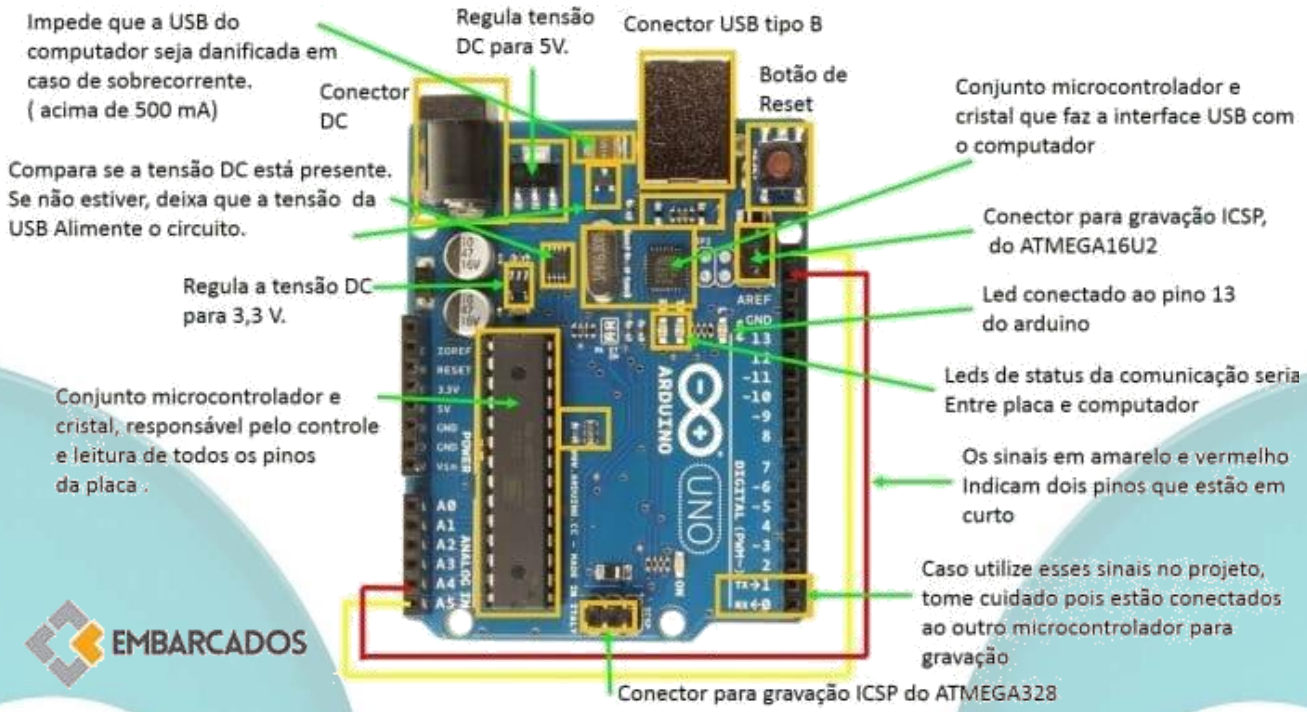


Figura 9 - Arduino Uno - Fonte: www.embarcados.com.br/arduino-uno/

Figura 10 - Blocos identificados de uma placa Arduíno Uno Fonte: https://www.embarcados.com.b r/arduino-uno/

2.4.2 Compatibilidade entre placas Arduino

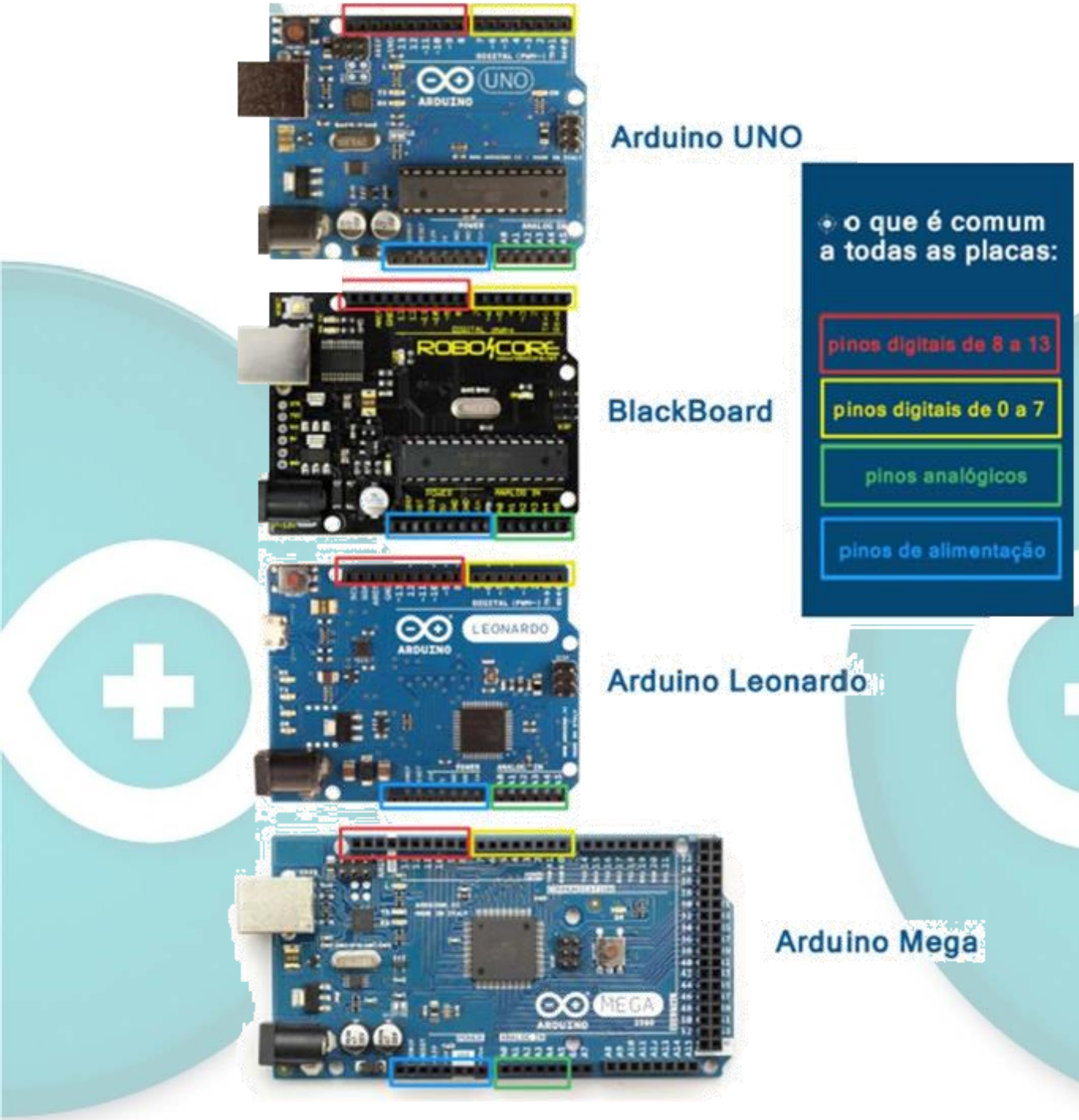


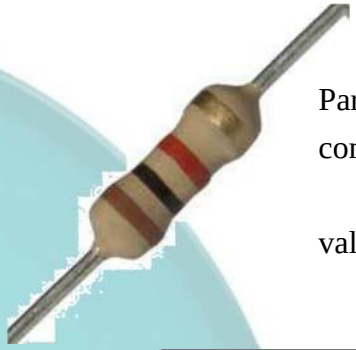
Figura 11 - Compatibilidade entre placas Arduino - Fonte: Apostila RoboCore V7, p. 10

Assim, mesmo os esquemas sendo ilustrados com a placa UNO, nada impede que use outro Arduino, por exemplo MEGA, ou qualquer outra placa!

2.5 Uma breve descrição dos componentes

Para identificar cada um dos componentes e deixar a compreensão do circuito um pouco mais fácil, aqui detalharemos um pouco cada um deles.

2.5.1 Resistor



O que isto faz: Limita a corrente elétrica que passa pelo circuito. Para limitar para mais ou para menos a corrente, o valor deste componente pode variar.

Número de pinos: 2 pinos de mesmo comprimento. Para saber o valor de cada resistor, basta seguir o esquema.

Figura 12 - Resistor

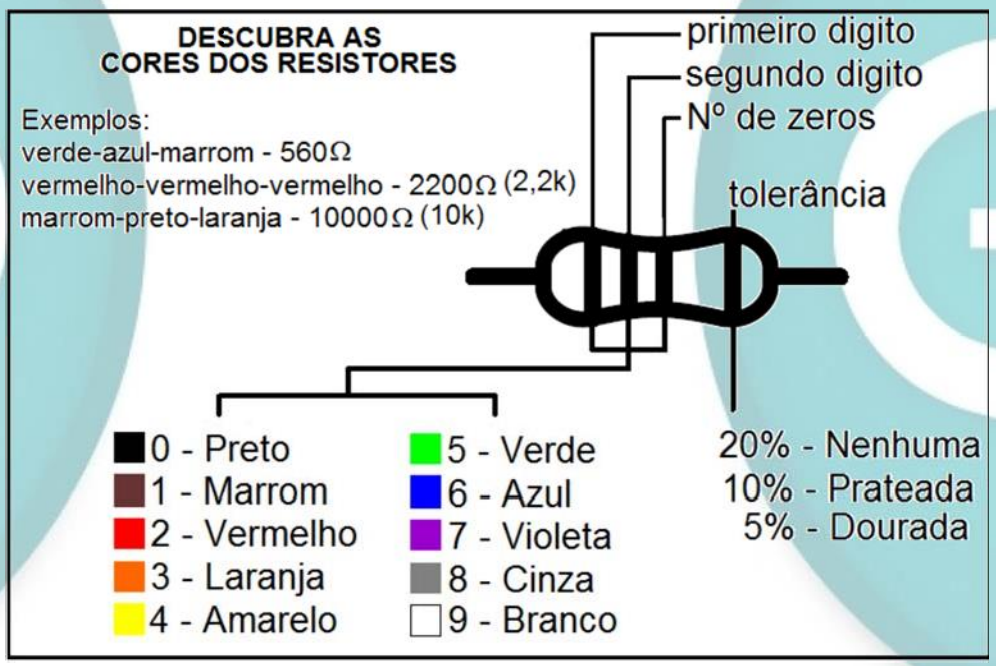
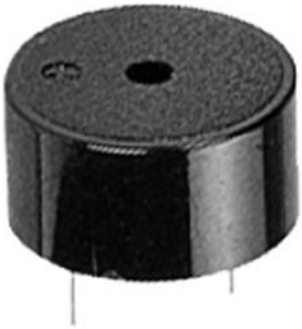


Figura 13 - Tabela de Cores – Fonte: Apostila RoboCore V7, p. 11

As práticas aqui apresentadas usam três valores de resistores, o de 300Ω (trezentos ohms), o de 10kΩ (10 mil ohms) e o de 1MΩ (1 milhão ohms). Com símbolo nos esquemas elétricos. Para se familiarizar, os símbolos são estes ao lado, veja que eles são muito parecidos, porém deve fazer a leitura de cores pelo começando pelo lado oposto ao dourado.

Figura 14 – Resistor - Fonte: Acervo pessoal Samuel Serqueira

2.5.2 Buzzer



O que isto faz: Quando uma corrente elétrica passa por ele, ele emite um som. Número de pinos: 2 pinos (este componente tem polaridade, portanto fique atento na hora de ligá-lo).

Figura 15 – Buzzer - Fonte: Acervo pessoal Samuel Serqueira

2.5.3 Chave Momentânea

O que isto faz: Quando o botão é apertado, os contatos entre os terminais de cada lado são ligados entre si. Número de pinos: 4 pinos (os 2 pinos de cada lado já estão em contato normalmente. Quando o botão é apertado os 4 entram em contato).

Figura 16 – Chave Momentânea Fonte: Acervo pessoal Samuel Serqueira

2.5.4 Potenciômetro

O que isto faz: Varia a resistência dos terminais conforme a haste superior é girada

Número de pinos: 3 pinos (a resistência varia entre um dos pinos mais da extremidade para com o do centro).

Figura 17 – Potenciômetro - Acervo pessoal Samuel Serqueira

2.5.5 Leds

O que isto faz: Emite uma luz quando uma pequena corrente o excita (apenas em uma direção, do pino mais longo para o pino mais curto)

Número de pinos: 2 pinos (um mais longo e outro mais curto)

Figura 18 – Leds - Fonte: Acervo pessoal Samuel Serqueira

2.5.6 Leds RGB

O que isto faz: Pense em três leds de alto brilho: um vermelho, um verde e um azul. Agora, junte todos eles em um só. Pronto, isso é um LED RGB.

Número de pinos: 4 pinos, onde o maior deles é comum a todas as cores.

Figura 19 – Leds RGB - Fonte: Acervo pessoal Samuel Serqueira

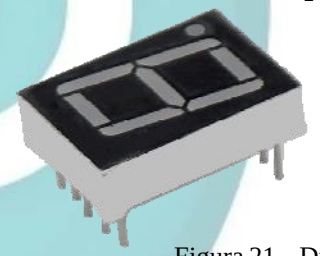
2.5.7 Sensor de Luminosidade LDR

O que isto faz: É uma resistência que varia conforme a luminosidade se altera sobre ele.

Número de pinos: 2 pinos de mesmo comprimento

Figura 20 – Sensor de Luminosidade LDR Fonte: Samuel Serqueira

2.5.8 Display de 7 segmentos catodo comum



O que isto faz: Possui 7 leds em formato de número "8", com os quais é possível acender os números de 0 a 9. Ainda possui um LED indicador de ponto. Número de pinos: 10 pinos.

Figura 21 – Display de 7 segmentos catodo comum - Fonte: Acervo pessoal Samuel Serqueira

2.5.9 Sensor de presença

O Sensor de presença, também chamado de sensor de movimento, é um módulo que usa um sensor PIR (piroelétrico), capaz de detectar a variação de luz infravermelha emitida pela radiação dos corpos. Dessa forma, detecta movimento em um ambiente quando um corpo emissor de radiação se movimenta.

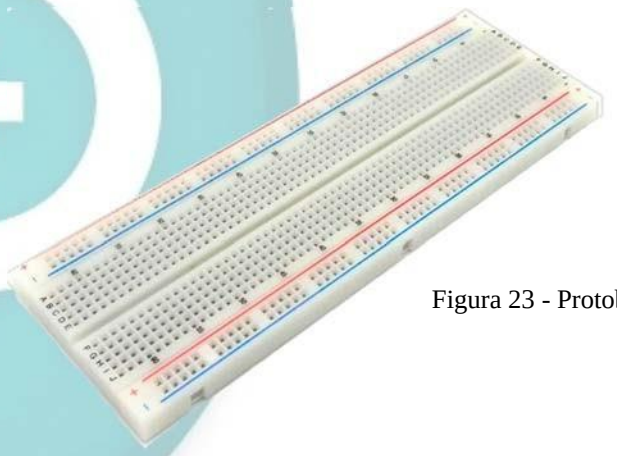
Figura 22 – Sensor de presença - Fonte: Acervo pessoal Samuel Serqueira

2.5.10 Jumpers

Os Jumpers são utilizados para conectar componentes sem a necessidade de soldas. Geralmente são utilizados em protótipos, nas protoboards, e são construídos de material condutor envolto de um material isolante. Na figura 22 estão respectivamente de cima para baixo os jumpers tipos fêmea-fêmea, macho-macho e macho-fêmea

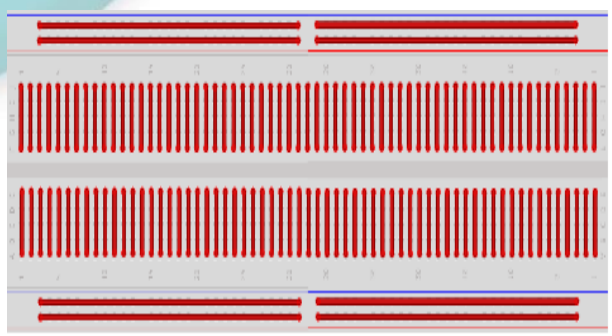
Figura 22 – Jumpers - Fonte: Acervo pessoal Samuel Serqueira

2.5.10 Protoboard



O que isto faz: trata-se de uma placa de plástico, cheia de peque-nos furos com ligações internas, onde as ligações elétricas são feitas.

Figura 23 - Protoboard - Fonte: Acervo pessoal Samuel Serqueira



Os furos nas extremidades superior e inferior são ligados entre si na horizontal, e as barras do meio são ligadas na vertical. Para ilustrar isto, veja abaixo como são as ligações internas da protoboard.

Figura 24 - Ligação interna da protoboard - Fonte: Acervo pessoal Samuel Serqueira

O led vermelho tem a extremidade direita ligada a um resistor. Este resistor está ligado a outro resistor por meio de uma das ligações internas superiores da protoboard.

Este último resistor, por sua vez, está ligado à extremidade esquerda do led, utilizando uma das ligações internas inferiores da protoboard.

Protoboard é uma placa utilizada para a prototipação eletrônica, ou seja, no ensaio de montagem de circuitos eletrônicos experimentais.

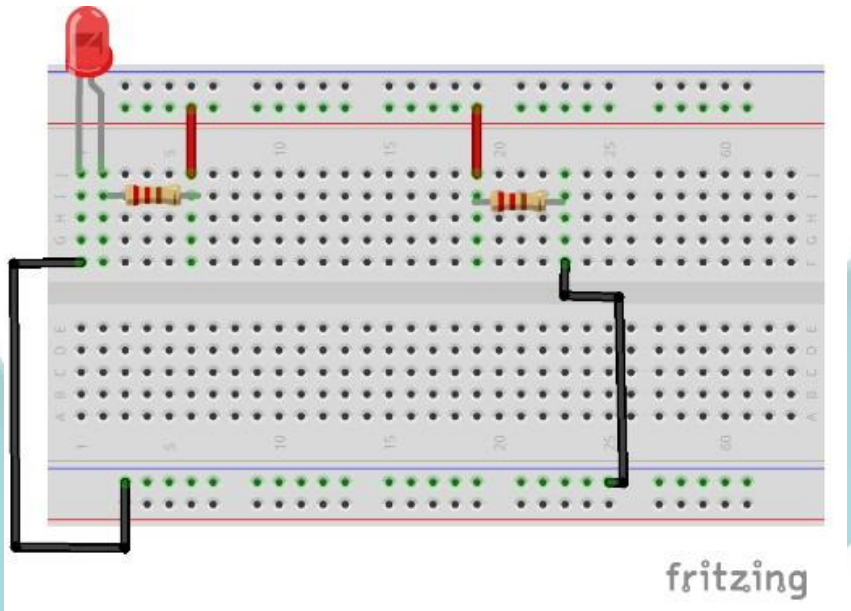


Figura 35 - Usando Protoboard Fonte: Fritzing

A vantagem é facilidade de inserção e remoção de componentes, uma vez que não é necessário soldá-los. As placas geralmente variam de 170 a 1800 furos (ou pontos). Geralmente a corrente suportada na protoboard é de 500 mA.

3.0 5 coisas com o Arduino;

"Para aprender a fazer algo, deve-se começar a tentar, pois é aos poucos que se aprende, no próprio processo de fazer."
Bernard Charlot

Existem 5 coisas que é necessário saber fazer com o Arduino;

- ✓ Fazer o LED embutido no pino 13 acender;
- ✓ Fazer o LED embutido no pino digital piscar;
- ✓ Fazer uma função para que o LED do pino digital pisque;
- ✓ Enviar dados pela porta serial;
- ✓ Receber dados.

3.1 Fazer o LED embutido acender

Componentes:
1 Arduino;

O código a seguir deve ser inserido na integra na IDE Arduino;

```
/**************************Sketch de domínio público**********************
 ********************************Liga Led*********************************
*************************************************************************\
/*O setup é executado uma vez quando é pressionado o botão reset ou a alimentação, é criado sem retorno (void) ou parâmetros ().*\

void setup() {
        pinMode(13, OUTPUT); //Seta o modo do pino 13 como saída.
}
// O método loop é executada enquanto houver alimentação, é criado sem retorno (void) ou parâmetros ().
void loop() {
        digitalWrite (13, HIGH); // escreva de forma digital no 13, Alto - Acende o LED - se trocar HIGH por LOW ele apagará
}
```

Compile o sketch (nome dado ao código no Arduino) clicando no botão Verificar do ambiente de desenvolvimento, para ver se não existe nenhum erro de código.

O botão é o seguinte:

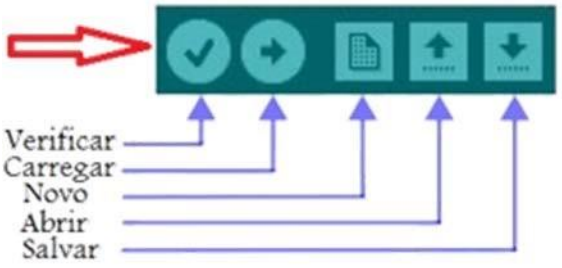


Figura 26 - Botões do Arduino Fonte: acervo pessoal Samuel Serqueira

Se na barra inferior não aparecer nenhuma mensagem de erro, o programa está pronto para ser enviado ao Arduino.

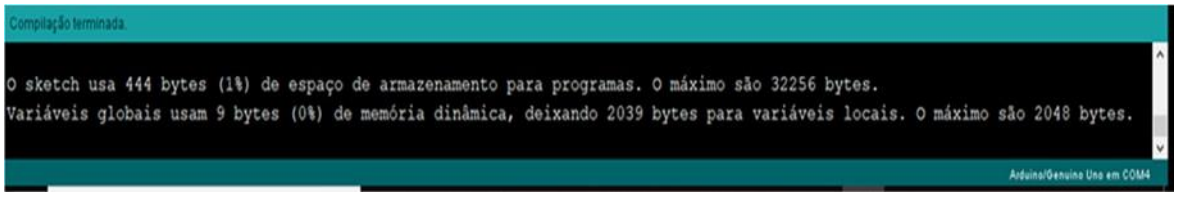


Figura 27 - Barra inferior - Fonte: acervo pessoal Samuel Serqueira

Para tanto, basta clicar no botão Carregar que é o seguinte:

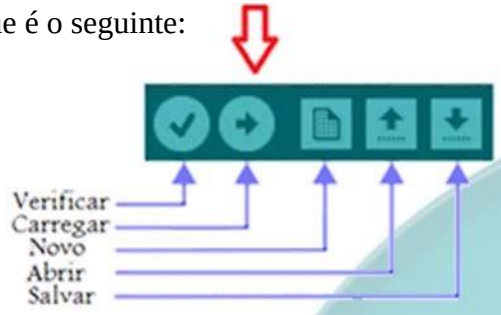


Figura 28 - Botões do Arduino Fonte: Samuel Serqueira

Dica: Experimentar no lugar de HIGH/LOW:

- ✓ 0, 1 e 2 ou true e false.

3.2 Fazer um LED piscar

O segundo procedimento mais simples de todos! Piscar um LED.

Basta alternar o pino digital entre ligado (HIGH) e desligado (LOW), aguardando um pequeno intervalo de tempo.

Para este sketch é usado uma montagem com uma protoboard, led e resistor com outra porta digital, a 8. Colocar os componentes como está sendo mostrado na imagem abaixo, bem como suas ligações.

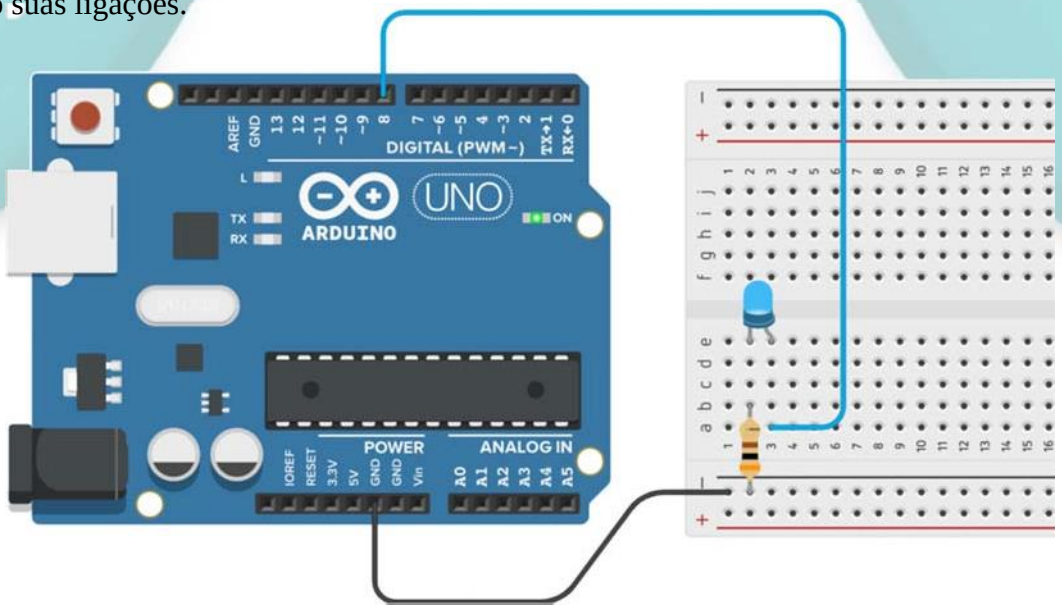


Figura 29 - Botões do Arduino Fonte: acervo pessoal Samuel Serqueira

Dica 1: Não importam as cores dos fios na hora de ligação, e sim fazer a ligação correta. Prestar atenção ao fazer as ligações pois apenas 1 fio no lugar errado pode comprometer todo o experimento.

Dica 2: Não fazer a montagem ou alterações com o Arduino conectado a fonte ou ao PC.

Componentes:

- ✓ 1 Arduino;
- ✓ 1 Led Vermelho;
- ✓ 1 Resistor de 300 ou 330 ohms(Ω);
- ✓ 1 Placa Protoboard (opcional);
- ✓ 2 Jumpers;

```
/*******************Sketch de domínio público ***************************
************************* Pisca Led************************************
*************************************************************\
//constantes (constantes não mudam de valor durante a execução do programa)
const int led = 8; //Cria a constante led, do tipo inteiro e referenciado no pino 8

void setup() {
        pinMode(led, OUTPUT); //Seta o modo do pino 8, com o nome de “led” como
saída.
}
void loop() {
digitalWrite(led, HIGH); // Acende o LED (HIGH é o nível de tensão - Alto)
delay (1000); // aguarda 1 segundo (1.000 milissegundos)
digitalWrite (led, LOW); // apaga o LED (LOW é o outro nível de tensão - Baixo)
delay (1000); // Aguarda 1 segundo (1.000 milissegundos)
}
```

Há várias maneiras de piscar um LED, porém essas três mostra um diferencial no paradigma da programação:

- ✓ A tradicional, que foi vista acima;
- ✓ A mais curta que apresenta uma lógica de inversão da leitura do estado da porta digital;
- ✓ A sem atraso (sem delay) que usa um método para contagem de tempo, e no momento que completa o intervalo chama a o comando para a ação.

A maneira mais curta

```
/*********************Sketch de domínio público*****************************
************************Pisca Led curto*************************************
*****************************************************************************\
void setup() {
  pinMode (13, OUTPUT);
}
void loop () {
  digitalWrite (13,!digitalRead (8));
  delay (1000);
}
```

Sem Atraso

```
/*********************Pisca Led sem delay*********************************
*********************Samuel Serqueira 2014*******************************
*****************************************************************************\
const int pinoLed = 8;
const int intervalo = 500;
unsigned long passouTempo = 0;
unsigned long tempoAtual = 0;
int estadoLed = LOW;
void setup () {
        pinMode (pinoLed, OUTPUT);
        digitalWrite(pinoLed, estadoLed);
}
void loop () {
        tempoAtual = millis ();
        if (tempoAtual - passouTempo >= intervalo) {
                passouTempo = tempoAtual;
                if (estadoLed == LOW){
                        estadoLed = HIGH;
                }
else {
                        estadoLed = LOW;
                }
}
        digitalWrite(pinoLed, estadoLed);
}
```

3.3 Fazer uma função para que o LED pisque

Funções são essenciais para reaproveitamento do código, ao invés de ficar digitando várias vezes a mesma coisa apenas invoque a função que neste caso altera o estado do pino de HIGH para LOW e vice-versa.

Neste sketch é usado uma lógica de controle de fluxo, o **se** e **senão** (**if** e **else**).

```
/*********************Função Pisca Led*********************************
*********************Samuel Serqueira 2014*****************************
*************************************************************************\
//constantes (constantes não mudam de valor durante a execução do programa)
const int led = 8;
const int atraso = 1000; // valor para o delay
//variáveis (variáveis mudam de valor durante a execução do programa
int valor = HIGH;//para acender ou apagar o led, iniciando com o valor alto.

void setup() {
pinMode(led, OUTPUT);
}
void loop() {
funPisca(); // chama a função
delay (atraso); // aguarda atraso de tempo
}
void funPisca () {//cria a função sem retorno
        if (valor == HIGH) //se valor for igual a HIGH
valor = LOW; //Valor recebe LOW
else //Senão
valor = HIGH;//Valor recebe HIGH
digitalWrite (led, valor) ;//escreva digitalmente o valor no pino led
}
```

3.4 Enviar dados pela porta serial

Com o Arduíno é possível enviar dados dele para o computador. Neste código o Arduíno faz uma contagem a partir da variável “armDados” e imprime no Monitor Serial.

```
/*********************Função Pisca Led*********************************
*********************Samuel Serqueira 2014*****************************
*************************************************************************\
int armDados = 0; // variável que será usada para armazenar a informação e enviar para o
PC via porta serial

void setup () {
        Serial.begin(9600); // inicia o método Serial (. begin) para comunicação com o PC
ajusta a taxa de transferência de dados para 9600 baud ou bps (bits por segundo)
}
```

```
 void loop() {
        Serial.println (armDados); // enviar pela porta serial o valor guardado na variável
armDados
        delay (1000);
armDados++; // incrementa em 1
}
```

3.5 Receber dados pela porta serial

Também é possível enviar dados do computador para o Arduino. Abrir o Monitor Serial, digitar a letra “a” (minúscula) e clicar em Enviar. O LED acenderá, e imprime “Aceso”, se enviar a letra “a” novamente vai apagar e imprimir “Apagado”.

```
/*****************Receber dados pela porta serial**********************
*******************Samuel Serqueira 2014*****************************
*************************************************************************/
//Declaração das constantes
const int ledPin = 8;
//variáveis
char tecla; //variável do tipo char
int estado = HIGH; // variável que guarda o estado para o pino digital.
void setup () {
        Serial.begin(9600); // inicia a comunicação serial onde escreve os dados
        pinMode (ledPin, OUTPUT);
        digitalWrite (ledPin, estado) ;// coloca o valor de estado no pino ledPin
}
void loop () {
        if (Serial.available() > 0) {//o método “available’ verifica se há dados a serem
transmitidos pela serial retornando “true” ou “1” se sim ou “false” ou “0” se não.
          tecla = Serial.read(); // armazena em "tecla" o retorno do método read(), esse
método lê um valor que é escrito na porta serial
          if(tecla == 'a') { //se for igual ao caractere ‘a’
             if(estado == HIGH) { //se o LED estiver aceso
                estado = LOW;// apague
           Serial.println("Apagou"); //escreve mensagem no Monitor Serial
         }
         else {//senão
                        estado = HIGH;//estado recebe alto
                        Serial.println("Acendeu"); //escreve mensagem no Monitor Serial
                }

          }
         }
         digitalWrite(ledPin, estado);
}
```

4.0 Projetos com Arduino

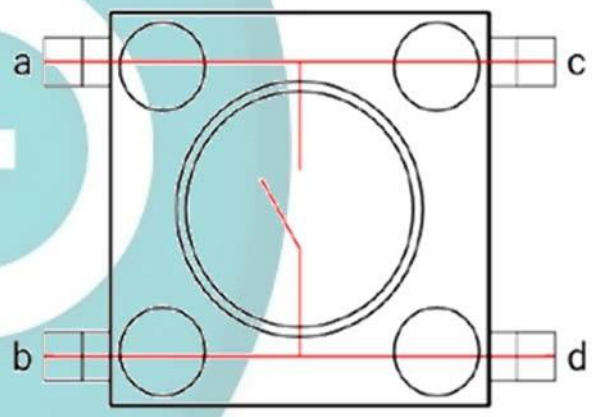
4.1 Projeto controlando uma porta digital com um botão

O botão é um dos dispositivos mais populares quando se fala de interface humano-máquina. Seja no smartphone, nos elevadores ou até mesmo o próprio interruptor da luz da sala, os botões são responsáveis por disparar ações em diversos sistemas eletrônicos presentes no dia-a-dia. Com um simples apertar pode ligar uma lâmpada, travar a tela do celular ou interagir com um jogo eletrônico.

Então entender sobre esse recurso simples e barato, que é extremamente versátil e será utilizado em diversos projetos.

Os pinos do Arduino funcionam como saída digital, onde a energia flui por eles até a carga. Entretanto, eles também podem ser configurados para fazer o contrário, deixando com que a energia flua de fora para dentro, podendo então ser mensurada. Assim fica fácil detectar quando a tensão no pino é 5 V e quando é 0 V.

Para usar um botão com a placa, basta conectá-lo corretamente a um pino que esteja configurado como entrada e pode detectar se o botão está pressionado ou solto.



Este projeto usa um botão para controlar o estado de um LED. O botão para este projeto é o que possui quatro terminais que são interligados aos pares (terminal a com c e b com d). Ao pressionar o botão, os quatro terminais são conectados entre si, gerando continuidade no circuito. Podemos então utilizar esse funcionamento para alterar o estado lógico de uma entrada digital do Arduino.

Figura 30 – Botão. Fonte: https://www.robocore.net/tutoriais/kit- iniciante-v8-lendo-um-botao.html

Se simplesmente conectar um dos pares de terminais ao VCC e o outro ao pino do Arduino, quando pressionar o botão o pino terá 5v. Mas qual sinal chegará ao pino do Arduino quando o botão não estiver pressionado? Nada? Na verdade, a melhor resposta seria "qualquer coisa", pois como não tem nenhum sinal válido (5 V ou 0 V) conectado a ele, o caminho fica livre para os ruídos existentes no circuito. O pino do Arduino então poderá assumir um valor lógico verdadeiro ou falso a qualquer momento.

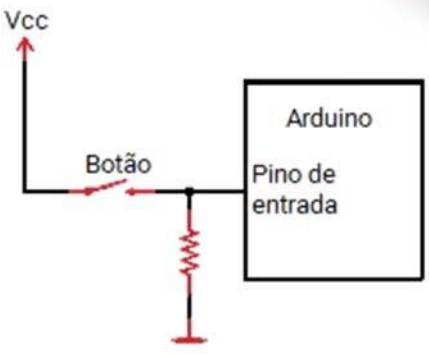


Figura 41 - Botão Fonte: Samuel Serqueira

Para retirar essas flutuações que prejudicam a interpretação dos sinais, é utilizado normalmente um resistor de pull-down. Ele atua garantindo que o sinal de GND chegue até o pino enquanto o botão não for pressionado, e protege o circuito contra um curto circuito quando você aperta o botão. Veja um exemplo de ligação na figura 30.

Componentes:

- ✓ 1 Placa Protoboard;
- ✓ 1 Arduíno uno R3 ou compatível com cabo USB;
- ✓ 1 Chave momentânea;
- ✓ 1 Resistores 10kΩ;
- ✓ 1 Resistor 300 ou 330 ohms(Ω);
- ✓ 1 Led (qualquer cor);
- ✓ 5 Jumpers;

Trata-se de fazer um botão acender um led quando pressionado e, quando solto, o led deverá apagar. Colocar os componentes como está sendo mostrado na imagem abaixo, bem como suas ligações, conectar os componentes do projeto na protoboard. Para isso, desligue o cabo USB do Arduino e monte o circuito.

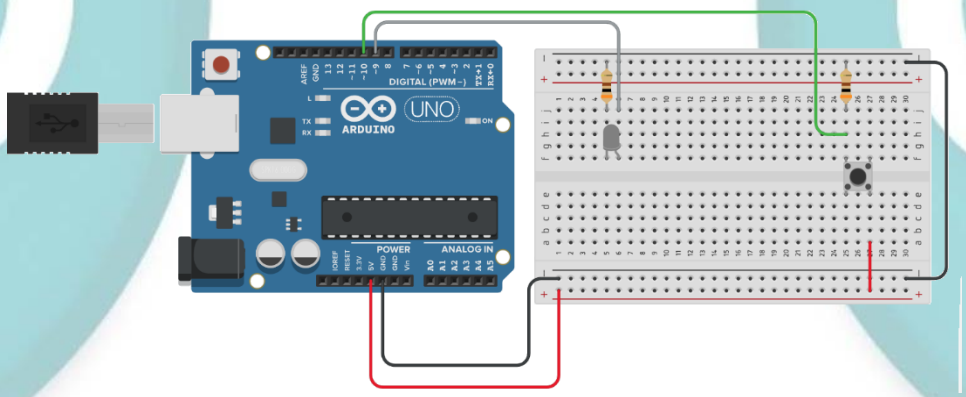


Figura 32 - Botões do Arduino Fonte: João Marcos (2022)

Realizar a configuração dos pinos declarando as variáveis, definir os pinos como saída e entrada no SETUP.

Criar uma variável que não utiliza portas do Arduino, e serve para guarda o estado do botão como verdadeiro ou falso, 0 ou 1. Esta variável determina o sinal enviado para o led.

No LOOP uma condição faz a leitura do botão, o botão envia um sinal para o Arduino (botão pressionado), o Arduino envia um sinal alto para o led (liga o led). Se não, ou seja, se o botão não estiver sendo pressionado, envia sinal baixo para o led, deixando o componente desligado.

Desta maneira o led se mantem aceso somente durante o período que o botão é pressionado.

```
    /**********************Liga Led 1************************************
**********************Samuel Serqueira 2014*****************************
*****************************************************************************/
// Declaração das constantes
const int LED = 9;
const int botao = 10; //constante botão refere-se ao pino digital 9.
int estadoBotao = 0;
void setup() {
  pinMode(LED, OUTPUT); //Definindo o pino digital como pino de saída.
```

```
  pinMode(botao, INPUT); //Definindo o pino digital como pino de entrada.
}
void loop() {
  /*Lendo o estado do pino 9, constante botao, e atribuindo o resultado a variável estadoBotao,
poderá ser ALTO (HIGH) se o botão estiver pressionado, ou BAIXO (LOW), se o botão
estiver solto */
  estadoBotao = digitalRead(botao);
  //Verificando o estado do botão para definir se acenderá ou apagará o LED. if
   (estadoBotao == HIGH) {//se botão pressionado
        digitalWrite(LED, HIGH); // acende o LED.
     }
     else {// se não estiver pressionado
        digitalWrite(LED, LOW); // apaga o LED.
   }
}
```

O próximo sketch é a mesma aplicação, porém, utiliza o botão como uma chave interruptor, e não precisa segurar o botão.

Assim quando o botão é pressionado o led acende e se mantem aceso até que o botão é pressionado novamente.

```
/************************Liga Led 2**********************************
*********************Samuel Serqueira 2014*****************************
*************************************************************************/
//Declaração das variáveis
const byte LED = 9 ;//refere-se ao pino digital 10
como byte const byte botao = 10;//refere-se ao
pino digital 9 como byte
byte estadoBotao = 0;//variável do tipo byte para ler o
estado do botão byte Aux = 0;//variável auxiliar do tipo
byte, que vai alternar de estado. void setup() {
        pinMode(LED,OUTPUT);//definindo o pino
        como saída.
        pinMode(botao,INPUT);//definindo o pino
        como entrada Serial.begin (9600);//iniciando
        uma comunicação serial
}
void loop() {
        estadoBotao = digitalRead(botao);// Lendo o estado do pino 9, e
atribuindo o resultado a variável estadoBotao
        //Verificando o estado do botão para definir se acenderá ou
        apagará o LED. if (estadoBotao == 1) {//se botão pressionado
                Aux = !Aux;//muda o estado da variável auxiliar
        }//fim if
  digitalWrite (LED, Aux);//acende ou apaga o led conforme o estado da variável auxiliar
//imprimindo o estado do
        botão e Aux
        Serial.println(esta
        doBotao);
        Serial.println("

        ");
        Serial.println(Aux
        );
delay (500);//aguarda meio segundo
}//fim loop
```

Para configurar o pino do botão como entrada, utiliza a função pinMode, com INPUT.

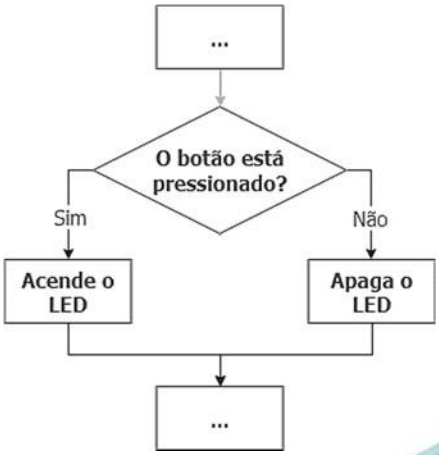
E para realizar a "escrita digital" no pino do LED usa digitalWrite, enquanto que para realizar a leitura digital usa digitalRead. Ao contrário da função de escrita, que não retorna nenhum valor, a função de leitura nos retorna o valor atual do pino entre parêntesis (nesse caso, o botão). Isso significa que se o botão estiver pressionado, o valor da tensão no pino é de 5 V e a função retornará HIGH. Enquanto o botão estiver solto, o valor é de 0 V, retornando então LOW.

Este sketch não foi construído com códigos sequenciais, que sempre executavam uma linha atrás da outra. Agora foi preciso realizar algum tipo de desvio, pois acender ou apagar o LED depende de uma condição: o botão estar ou não pressionado.

É para isso que entra a estrutura condicional if and else.

Ela realiza um teste lógico para decidir entre dois trechos de código. Ela executa o primeiro se a condição for verdadeira e o segundo se a condição for falsa.

Figura 5 - Fluxograma de condicional Fonte: www.robocore.net/tutoriais/kit- iniciante-v8-lendo-um-botao.html




```
if (condição){
        // código executado se a condição for verdadeira (HIGH)
}
else{
        // código executado se a condição for falsa (LOW)
}
```

Quando é realizado testes lógicos, é natural que precise comparar as coisas. Seja dois valores, resultados de funções, variáveis ou quaisquer outras grandezas, pode utilizar os operadores de comparação para definir qual é a relação entre eles. Na tabela abaixo tem quais são os operadores de comparação aceitos pela linguagem do Arduino e alguns exemplos.

Símbolo	Comparação	Exemplo
==	igual a	HIGH == HIGH
<	menor que	5 < 6
>	maior que	17 > 16
<=	menor ou igual a	7 <= 8
>=	maior ou igual a	65 >= 65
!=	não igual a	HIGH!= LOW

Este código utiliza o operador == para verificar se o resultado da função digitalRead(botao) é igual a HIGH. Somente quando essa comparação é verdadeira é que o LED acende.

4.2 Projeto Semáforo de Trânsito de Carros e Pedestres

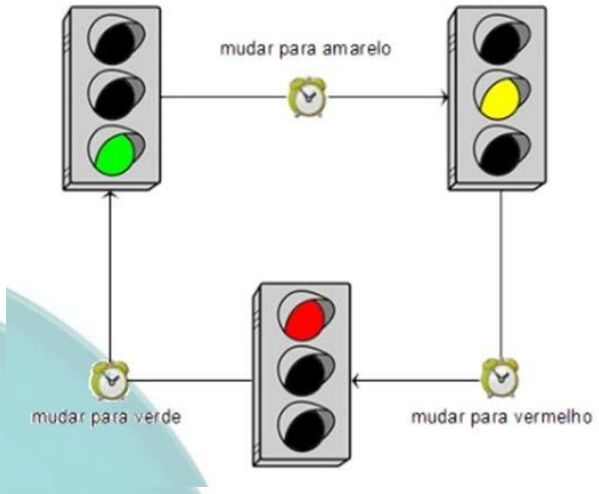


Figura 6 - Semáforo
Fonte: https://pt.wikipedia.org/wiki/Sem%C3%A1foro

Componentes:

- ✓ 1 Arduíno uno;
- ✓ 1 Placa Protoboard;
- ✓ 2 Leds Vermelho;
- ✓ 2 Leds Verde;
- ✓ 1 Led amarelo;
- ✓ 5 Resistores de 300Ω ou 330Ω;
- ✓ 1 Resistor de 10kΩ;
- ✓ 10 Jumpers;
- ✓ 1 Botão (push-button);

Semáforo de Trânsito de Carros e Pedestres.

Semáforo é um instrumento utilizado para controlar o tráfego de veículos e de pedestres nas grandes cidades em quase todo o mundo. Utiliza uma linguagem simples e, por isso, de fácil assimilação. É composto geralmente por três círculos de luzes coloridas. O controle semafórico permite alternar o direito de passagem na zona de conflito de uma interseção.

Este projeto vai desenvolver um "simulador" de sistema de semáforos (como os utilizados e que é "funciona" no mundo real).

4.2.1 Montagem 01

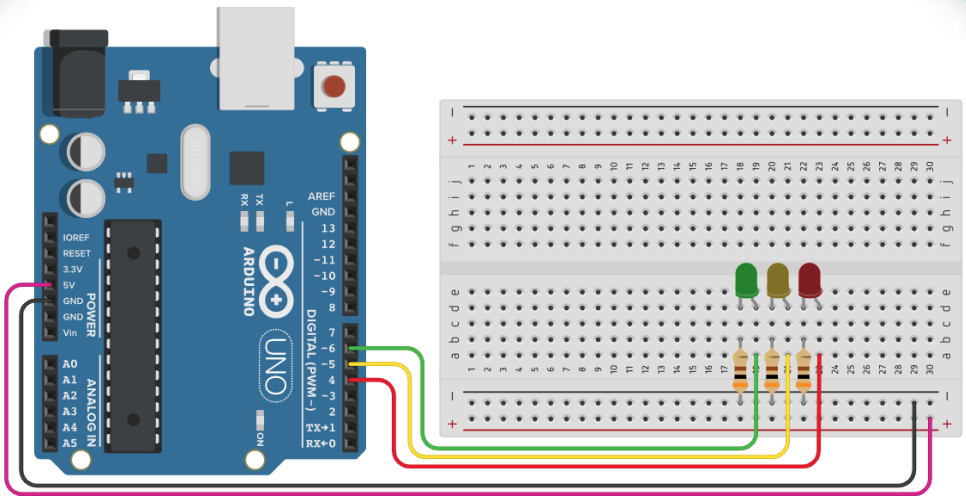


Figura 35 - Montagem do projeto Semáforo para veículos 1 lado - Fonte: Acervo pessoal João Marcos (2022)

Realizar a configuração dos pinos declarando as constantes, definindo os pinos como saída no SETUP. No LOOP iniciar a programação do semáforo.

```
  /*******************Projeto Semáforo 1 Lado*****************************
   *******************Samuel Serqueira 2014*****************************
  ************************************************************************\
//Declaração das constantes
const int LedVerdeE = 6;//Define o pino 6 como
LedVerde const int LedAmareloE = 5;//Define o pino
5 como LedAmarelo
const int LedVermelhoE = 4;//Define o pino 4 como
LedVermelho void setup(){
 //Configura os pinos dos LEDs
 como saída
 pinMode(LedVerdeE,
 OUTPUT);
 pinMode(LedAmareloE,
 OUTPUT);
 pinMode(LedVermelhoE,
 OUTPUT);
}
void loop(){
//Apaga Verde, apaga o Amarelo, acende o Vermelho, e aguarda
 5 segundos digitalWrite(LedVerdeE, LOW);
 digitalWrite(LedAmareloE
 , LOW);
 digitalWrite(LedVermelho
 E, HIGH); delay(5000);
 //Acende Verde, apaga o Amarelo, apaga o Vermelho, e aguarda
 3 segundos digitalWrite(LedVerdeE, HIGH);
 digitalWrite(LedAmarelo
 E, LOW);
 digitalWrite(LedVermelh
 oE, LOW); delay(3000);
 //Apaga Verde, acende o Amarelo, apaga o Vermelho, e aguarda
 2 segundos digitalWrite(LedVerdeE, LOW);
 digitalWrite(LedAmarelo
 E, HIGH);
 digitalWrite(LedVermelh
 oE, LOW); delay(2000);
}
```

O semáforo inicia ligando o led verde, e apagando os leds amarelo e vermelho, em seguida, tem um delay (pausa) que garante esse estado por 3000 milissegundos, ou 3 segundos. Passado o tempo, ele muda para o estado de advertência com um delay (pausa) de 2 segundos. Avançado para o estado em que o semáforo se encontra fechado, e assim se mantém por 5 segundos.

OBS.: Como o Arduino trabalha com um Loop após a execução da última linha, o processo automaticamente retorna para a 1º linha do VOID LOOP, isso se repete até que a placa seja desligada. Esta montagem simula apenas um dos lados, e o real tem um cruzamento simples com 2 semáforos, para que os carros da outra via sejam sinalizados também.

4.2.2 Montagem 02

Esta montagem simula apenas um dos lados, e o real tem um cruzamento simples com 2 semáforos, para que os carros da outra via sejam sinalizados também. Abaixo, uma montagem de um simulador de semáforo simples para um cruzamento e o sketch da programação.

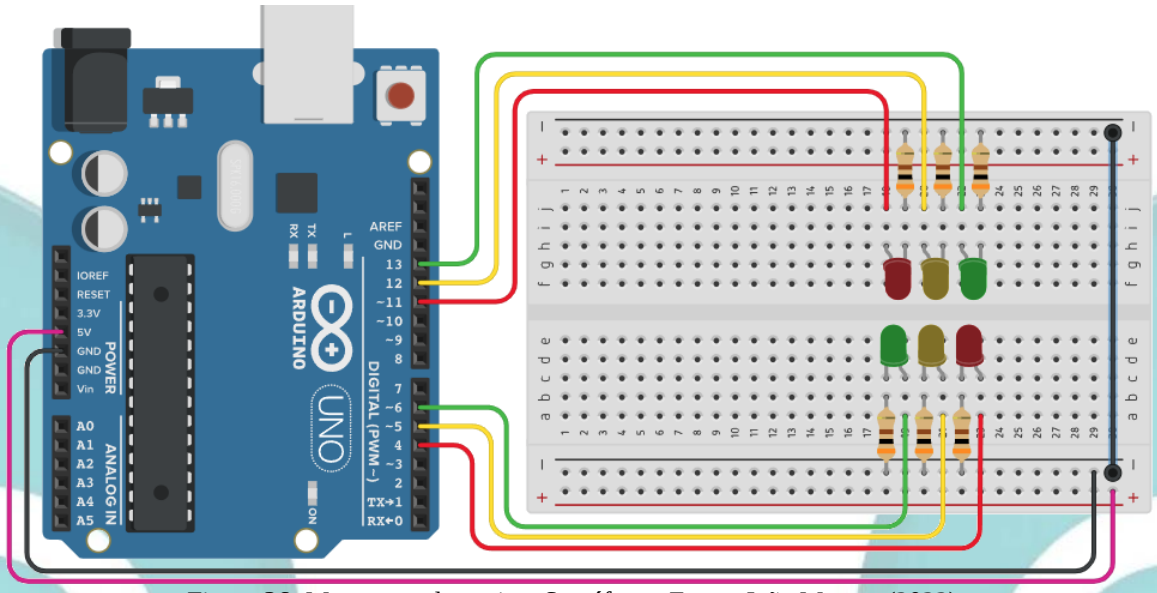


Figura 36- Montagem do projeto Semáforo - Fonte: João Marcos (2022)

```
   /*******************Projeto Semáforo 2 Lado*****************************
    *******************Samuel Serqueira 2014*******************************
    ***********************************************************************/
//Declaração das constantes
const int LedVerdeE = 6;//Define o pino 6 como LedVerdeE
const int LedAmareloE = 5;//Define o pino 5 como LedAmareloE
const int LedVermelhoE = 4;//Define o pino 4 como LedVermelhoE
const int LedVerdeD = 13;//Define o pino 13 como LedVerdeD
const int LedAmareloD = 12;//Define o pino 12 como LedAmareloD
const int LedVermelhoD = 11;//Define o pino 11 como LedVermelhoD
void setup(){
  //Configura os pinos dos LEDs como saída
  pinMode(LedVerdeD, OUTPUT);
  pinMode(LedAmareloD, OUTPUT);
  pinMode(LedVermelhoD, OUTPUT);
  pinMode(LedVerdeE, OUTPUT);
  pinMode(LedAmareloE, OUTPUT);
  pinMode(LedVermelhoE, OUTPUT);
}
void loop(){
  sigaVeiculo();
  atenVeiculo1();
  pareVeiculo();
  atenVeiculo2();
}
```

```
void sigaVeiculo(){
  //função que apaga Verde, apaga o Amarelo, acende o Vermelho da esquerda,
  //acende Verde, apaga o Amarelo, apaga o Vermelho da direita, e aguarda 3 segundos
  digitalWrite(LedVerdeE, LOW);
  digitalWrite(LedAmareloE, LOW);
  digitalWrite(LedVermelhoE, HIGH);
  digitalWrite(LedVerdeD, HIGH);
  digitalWrite(LedAmareloD, LOW);
  digitalWrite(LedVermelhoD, LOW);
  delay(3000);
}
void atenVeiculo1(){
  //função que apaga Verde, apaga o Amarelo, acende o Vermelho da esquerda,
  //apaga Verde, acende o Amarelo, apaga o Vermelho da direita, e aguarda 2 segundos
  digitalWrite(LedVerdeE, LOW);
  digitalWrite(LedAmareloE, LOW);
  digitalWrite(LedVermelhoE, HIGH);
  digitalWrite(LedVerdeD, LOW);
  digitalWrite(LedAmareloD, HIGH);
  digitalWrite(LedVermelhoD, LOW);
  delay(2000);
}
void pareVeiculo(){
  //função que acende Verde, apaga o Amarelo, apaga o Vermelho da esquerda,
  //apaga Verde, apaga o Amarelo, acende o Vermelho da direita, e aguarda 3 segundos
  digitalWrite(LedVerdeE, HIGH);
  digitalWrite(LedAmareloE, LOW);
  digitalWrite(LedVermelhoE, LOW);
  digitalWrite(LedVerdeD, LOW);
  digitalWrite(LedAmareloD, LOW);
  digitalWrite(LedVermelhoD, HIGH);
  delay(3000);
}
void atenVeiculo2(){
  //função que apaga Verde, acende o Amarelo, apaga o Vermelho da esquerda,
  //apaga Verde, apaga o Amarelo, acende o Vermelho da direita, e aguarda 2 segundos
  digitalWrite(LedVerdeE, LOW);;
  digitalWrite(LedAmareloE, HIGH);
  digitalWrite(LedVermelhoE, LOW);
  digitalWrite(LedVerdeD, LOW);
  digitalWrite(LedAmareloD, LOW);
  digitalWrite(LedVermelhoD, HIGH);
  delay(2000);
}
```

Configurar os pinos novos declarando as constantes novas, definir os novos pinos como saída no SETUP. No LOOP, a programação do semáforo é apresentada chamando as funções.

As funções apagam ou acendem os LEDS seguindo uma linguagem onde a ordem de um é o inverso do outro, mantendo o vermelho aceso por 5 segundos, o verde por 3 segundos e o amarelo por 2 segundos.

A ordem é verde esquerdo e vermelho direito aceso, depois apaga o verde esquerdo e acende o amarelo esquerdo mantendo o vermelho direito aceso, após apaga o amarelo esquerdo e acende o vermelho esquerdo, apaga o vermelho direito e acende o verde direito. Assim inicia o ciclo pelo verde direito, indo para o amarelo e depois vermelho, enquanto o da esquerda vai de vermelho direto para o verde.

Para programar estas ações foram criadas 4 funções, sem retorno - “void”, sem parâmetro -“()”, estas funções são chamadas no loop.

Em alguns cruzamentos tem o sinal semafórico para pedestre, a implementação do projeto é adicionar um par de LEDs, um vermelho e um verde, que indicarão para o pedestre quando ele poderá ou não atravessar a rua (na faixa!). E isso deve obrigatoriamente estar sincronizado com o semáforo dos carros, pois um só deve abrir quando o outro estiver fechado. As tabelas abaixo ilustram a lógica:

CARRO 1	**SIGA**	**ATENÇÃO**	**PARE**
VERDE	LIGADO	DESLIGADO	DESLIGADO
AMARELO	DESLIGADO	LIGADO	DESLIGADO
VERMELHO	DESLIGADO	DESLIGADO	LIGADO
PEDESTRE 1	**PARE**	**-**	**SIGA**
VERDE	DESLIGADO	DESLIGADO	LIGADO
VERMELHO	LIGADO	LIGADO	DESLIGADO

Figura 37 - Lógica do Semáforo – Fonte: Serqueira (2019)

CARRO 2	**PARE**	**ATENÇÃO**	**SIGA**
VERDE	DESLIGADO	DESLIGADO	LIGADO
AMARELO	DESLIGADO	LIGADO	DESLIGADO
VERMELHO	LIGADO	DESLIGADO	DESLIGADO
PEDESTRE 2	**SIGA**	**-**	**PARE**
VERDE	LIGADO	DESLIGADO	DESLIGADO
VERMELHO	DESLIGADO	LIGADO	LIGADO

Figura 38 - Lógica do Semáforo – Fonte: Serqueira (2019)

Na figura 39 uma sugestão de circuito com o sinal para pedestre do simulador de semáforo.

4.2.3 Montagem 03

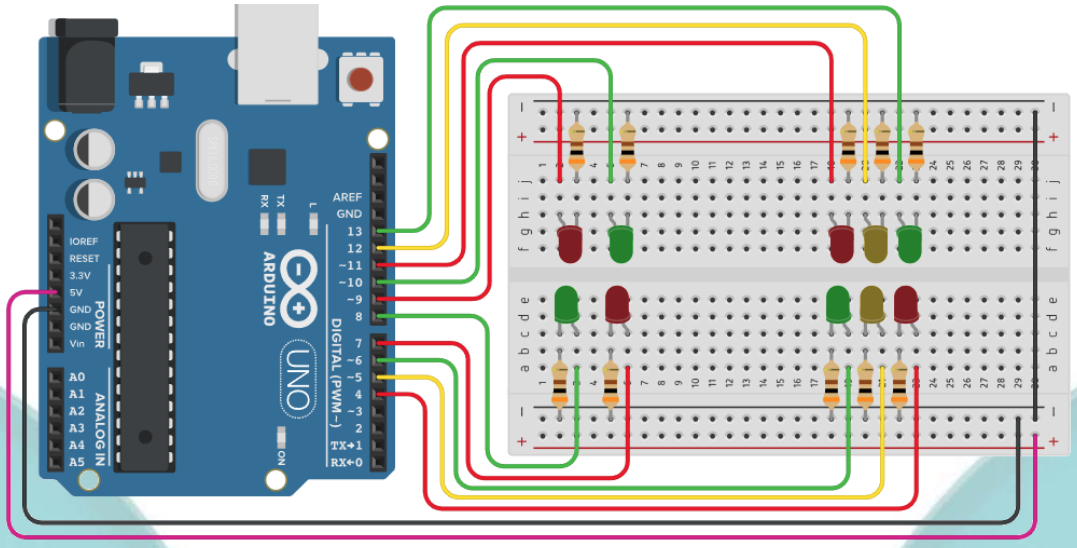


Figura 39 - Semáforo completo – Fonte: João Marcos (2022)

```
    /**************Projeto Semáforo 3 Lados*****************************
     ********************Samuel Serqueira 2014*******************************
   ***************************************************************************\
const int ledVerdeE = 6;//Define o pino 6 como LedVerdeE
const int ledAmareloE = 5;//Define o pino 5 como LedAmareloE
const int ledVermelhoE = 4;//Define o pino 4 como LedVermelhoE
const int ledVerdeD = 13;//Define o pino 13 como LedVerdeD
const int ledAmareloD = 12;//Define o pino 12 como LedAmareloD
const int ledVermelhoD = 11;//Define o pino 11 como LedVermelhoD
const int ledVerdePE = 10;//Define o pino 10 como LedVerdePE
const int ledVermelhoPE = 9;//Define o pino 9 como LedVermelhoPE
const int ledVerdePD = 8;//Define o pino 8 como LedVerdePD
const int ledVermelhoPD = 7;//Define o pino 7 como LedVermelhoPD
void setup(){
  //Configura os pinos dos LEDs como saída
  pinMode(ledVerdeD, OUTPUT);
  pinMode(ledAmareloD, OUTPUT);
  pinMode(ledVermelhoD, OUTPUT);
  pinMode(ledVerdeE, OUTPUT);
  pinMode(ledAmareloE, OUTPUT);
  pinMode(ledVermelhoE, OUTPUT);
  pinMode(ledVerdePE, OUTPUT);
  pinMode(ledVermelhoPE, OUTPUT);
  pinMode(ledVerdePD, OUTPUT);
  pinMode(ledVermelhoPD, OUTPUT);
}
void loop(){
  //acende Verde, apaga o Amarelo, apaga o Vermelho da esquerda
  semaVeiculo1(1, 0, 0);
  //apaga Verde, apaga o Amarelo, acende o Vermelho da direita
```

```
  semaVeiculo2(0, 0, 1);
  //acende Verde, apaga o Vermelho do pedestre da esquerda
  semaPedestre1(1, 0);
  //apaga Verde, acende o Vermelho do pedeste da direita
  semaPedestre2(0, 1);
  //aguarda 3 segundos
  delay (3000);
  //apenas muda de verde para amarelo
  semaVeiculo1(0, 1, 0);
  semaVeiculo2(0, 0, 1);
  delay (2000);
  //libera o fluxo ao contrario
  semaVeiculo1(0, 0, 1);
  semaVeiculo2(1, 0, 0);
  semaPedestre1(0, 1);
  semaPedestre2(1, 0);
  delay (3000);
  //apenas muda de verde para amarelo
  semaVeiculo1(0, 0, 1);
  semaVeiculo2(0, 1, 0);
  delay (2000);
}
void semaVeiculo1(byte verde, byte amarelo, byte vermelho){
  //função do semáforo Esquerdo
  digitalWrite(ledVerdeE, verde);
  digitalWrite(ledAmareloE, amarelo);
  digitalWrite(ledVermelhoE, vermelho);
}
void semaVeiculo2(byte verde, byte amarelo, byte vermelho){
  //função do semáforo Direito
  digitalWrite(ledVerdeD, verde);
  digitalWrite(ledAmareloD, amarelo);
  digitalWrite(ledVermelhoD, vermelho);
}
void semaPedestre1(byte verde, byte vermelho){
  //função do semáforo esquerdo pedestre
  digitalWrite(ledVerdePE, verde);
  digitalWrite(ledVermelhoPE, vermelho);
}
void semaPedestre2(byte verde, byte vermelho){
  //função do semáforo direito pedestre
  digitalWrite(ledVerdePD, verde);
  digitalWrite(ledVermelhoPD, vermelho);
}
```

Configurar novos pinos declarando as novas constantes, definir os novos pinos como saída no SETUP. No LOOP, a programação usa funções sem retorno – “void”, mas com parâmetros “(x ,x, x)”.

As funções fazem a mesma coisa das anteriores, mas através de seus parâmetros diminuímos as linhas de códigos, mesmo aumentando mais duas ações (a dos pedestres).

4.3 Projeto produzindo som

Este projeto mostra como fazer algumas notas musicais com buzzer no Arduino. Cada botão toca uma nota musical diferente e acende um led. É expansível para mais uma nota musical com mais botão (e mais led).

Cada um dos 3 botões tocará uma nota musical diferente. Este projeto usa um novo componente: o Buzzer. Um Buzzer nada mais é do que um pequeno alto-falante. Obviamente que ele não consegue tocar músicas, mas consegue fazer apitos soarem, como sirenes ou alarmes. A maioria dos alarmes de pequenos equipamentos eletrônicos é feito através de um buzzer. Ele funciona da seguinte maneira: quando alimentado por uma fonte, componentes metálicos internos vibram da frequência da fonte, produzindo assim um som.

Para este experimento, também pode utilizar um pequeno alto-falante (o som sai mais puro e a diferença entre as notas musicais é mais nítida). Último detalhe sobre o Buzzer: ele tem polaridade.

Na parte superior do buzzer tem um sinal de positivo (+). Este sinal mostra onde está o pino positivo do componente. Sempre ligue este a uma saída digital do Arduino e o outro em GND.

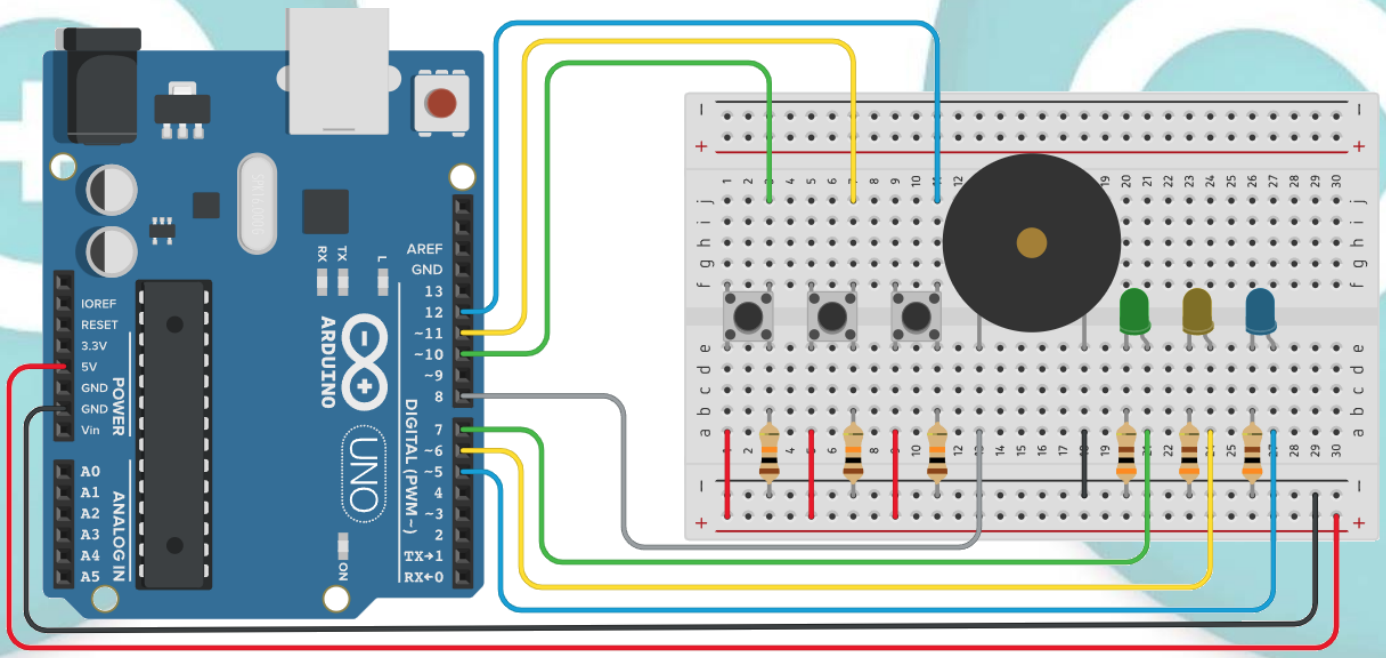


Figura 41 - Produzindo Som - Fonte: Acervo Pessoal João Marcos (2022)

Componentes:

- ✓ 1 Arduíno;
- ✓ 1 Placa Protoboard;
- ✓ 16 Cabos Jumpers;
- ✓ 1 Led difuso vermelho;
- ✓ 1 Led difuso amarelo;
- ✓ 1 Led difuso verde;
- ✓ 1 Resistor de 100 Ω;

- ✓ 3 Resistores de 300 ou 330 Ω;
- ✓ 3 Resistores de 10 KΩ;
- ✓ 1 Buzzer ativo 5V 12mm;
- ✓ 3 Botão (push-button);

Note que são usados três tipos de resistores, por mais que pareçam ter o mesmo valor eles são diferentes!

```
   /*****************Projeto produzindo som*****************************
******************Samuel Serqueira 2014*****************************
*************************************************************************\
//Declaração das constantes
const int ledVerde = 7;
const int ledAmarelo = 6;
const int ledAzul = 5;
const int botaoVerde = 10;
const int botaoAmarelo = 11;
const int botaoAzul = 12;
const int buzzer = 8; //O buzzer está colocado no pino 8
int estadobotaoVerde = 0;
int estadobotaoAmarelo = 0;
int estadobotaoAzul = 0;
int tom = 0; //Variavel para armazenar a nota musical
void setup() {
  pinMode(ledVerde, OUTPUT);
  pinMode(ledAmarelo, OUTPUT);
  pinMode(ledAzul, OUTPUT);
  pinMode(botaoVerde, INPUT);
  pinMode(botaoAmarelo, INPUT);
  pinMode(botaoAzul, INPUT);
  pinMode(buzzer, OUTPUT) }
void loop(){
  estadobotaoVerde = digitalRead(botaoVerde);
  estadobotaoAmarelo = digitalRead(botaoAmarelo);
  estadobotaoAzul = digitalRead(botaoAzul);
  if(estadobotaoVerde && !estadobotaoAmarelo &&
!estadobotaoAzul){ //se estadoBo-tao1 for verdadeiro e
estadobotaoAmarelo for verdadeiro e estadobotaoAzul for verdadeiro
faça
    tom = 100;
    digitalWrite(ledVerde, HIGH);
  }
  if(!estadobotaoVerde && estadobotaoAmarelo &&
!estadobotaoAzul){
tom = 200;
```

```
    digitalWrite(ledAmarelo, HIGH);
  }
  if(!estadobotaoVerde && !estadobotaoAmarelo && estadobotaoAzul){
    tom = 500;
    digitalWrite(ledAzul, HIGH);
  }
  if(tom > 0) { //se tom for maior que zero faça
    digitalWrite(buzzer, HIGH); // Liga buzzer
    delayMicroseconds(tom); // Espera o tempo proporcional ao comprimento de onda da
nota musical em microsegundos
    digitalWrite(buzzer, LOW); // Desliga buzzer
    delayMicroseconds(tom); // Espera o tempo proporcional ao comprimento de onda da
nota musical em microsegundos
    tom = 0; // Reseta o Tom para zero, para sair do if e nao to-car o som constantemente
    digitalWrite(ledVerde, LOW);
    digitalWrite(ledAmarelo, LOW);
    digitalWrite(ledAzul, LOW);
  }
}
```

Neste sketch foi apresentado um novo operador o “&&” (e lógico), ele devolve “true” se e somente se as duas condições forem “true”. No sketch, o estadoBotao armazena “1” para botão pressionado e “0” para não pressionado. No se (if) o estadoBotao assume true para 1 e false para 0. Assim no !estadoBotao inverte o estado que ele assume. Então se o estado do botão for “false” e existir o operador “!” (não) na variável estadoBotao, ela assume o estado “true”. Isto acontece no primeiro “if”.

4.4 Projeto LDR – sensor de luz

Um sensor fotoelétrico é um componente eletrônico que responde eletricamente às variações de intensidade de luz incidente. Um bom exemplo desse tipo de sensor são os LDRs ou Light-Dependent Resistor. Este componente varia sua resistência em função da intensidade da luz, respondendo com máxima resistência em situações de pouca luminosidade e mínima resistência em casos com alta luminosidade.

4.4.1 Divisor de tensão

Como as placas Arduino não medem resistência, pois a única coisa que consegue interpretar é tensão, é preciso usar um artifício para efetuar a leitura dos sensores, o circuito divisor de tensão. Este circuito faz com que a tensão de entrada seja fracionada, permitindo que a leitura possa ser efetuada.

Sem o resistor de 10kΩ a tensão no pino analógico seria sempre de 0 V (GND), independente da resistência do LDR.

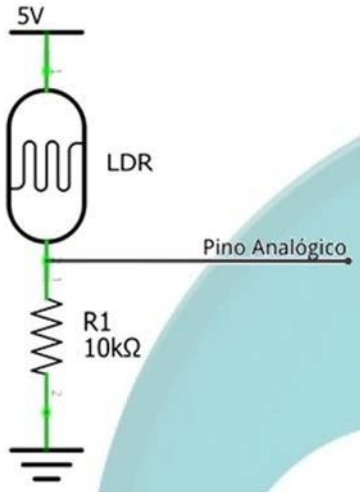


Figura 42 - Divisor de Tensão
Fonte: Samuel Serqueira

4.4.2 Conversor Analógico/Digital

Aparelhos eletrônicos só entendem 1 e 0, ou ligado e desligado. Como a tensão de saída do circuito de divisor de tensão pode variar de 0 a 5 V, não tem como usar uma entrada digital do micro controlador, que lê somente 0 V (LOW) ou 5 V (HIGH). Por isso se lê esse sinal com uma entrada analógica do Arduino.

As entradas analógicas são capazes de interpretar variações de 0 a 5 V, pois são ligadas a conversores analógico-digital (ADC), que resumidamente são circuitos que convertem as medidas do mundo real (leitura analógica) para valores que dispositivos eletrônicos entendem (leitura digital). Sua resolução é dada pelo número de bits.

No caso da maioria das placas Arduino, suas entradas analógicas podem efetuar leituras de 10 bits, ou seja, 2^10 = 1024 divisões de 0 a 5 V. Isso significa que a menor variação que o Arduino consegue interpretar é de 0,005 V.

Componentes:

- ✓ 1 Arduino Uno Rev3 ou similar
- ✓ 1 Placa Protoboard
- ✓ 1 LDR – Sensor de Luz
- ✓ 1 Led de Alto Brilho
- ✓ 1 Resistor 300 Ohm

- ✓ 1 Resistor 10 kOhm
- ✓ 5 Fios Jumpers

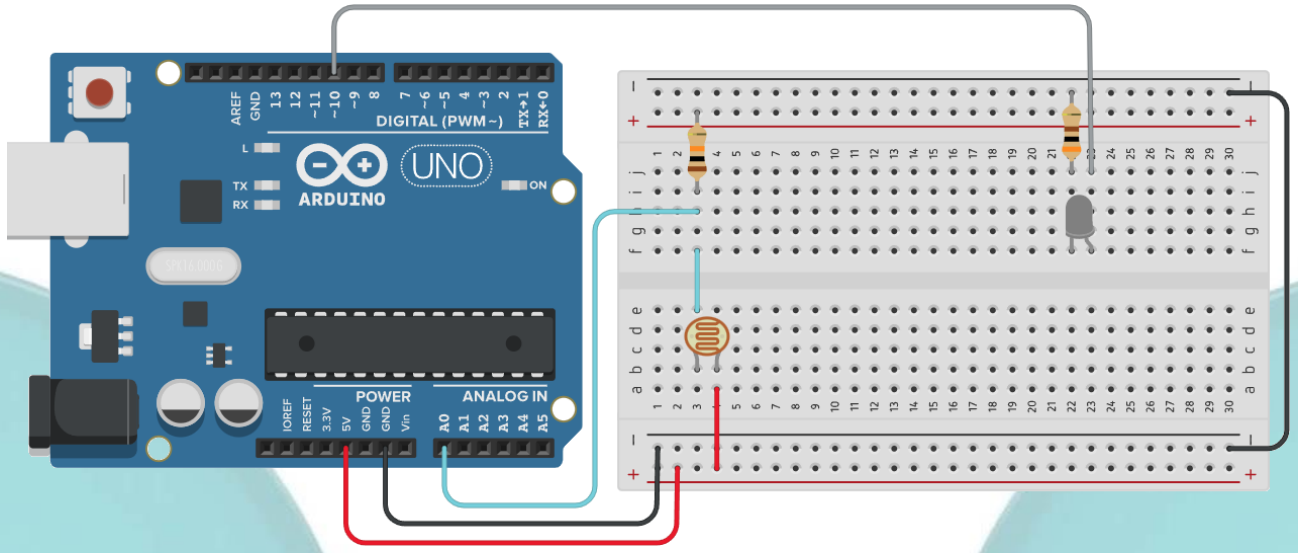


Figura 43 - Iluminação Automática - Fonte: Acervo pessoal Samuel Serqueira

```
/*************Exercício Iluminação Automatizada**********************
******************Samuel Serqueira 2014*****************************
*****************************************************************/
//Declaração das constantes const int LDR =
A0;
const int Led = 10;
//variáveis
int valorLido = 0;
int intensidade = 0; //Variável que controla a intensidade do led void setup() {
pinMode(Led, OUTPUT); Serial.begin
(9600);
}
void loop() {
valorLido = analogRead(LDR);
valorLido = map(valorLido, 1023, 0, 0, 255);
                        //modifica a variável conforme o mapa de 0 a 1023 para 255 a 0.
  Serial.print("Valor lido pelo LDR = ");
  Serial.println(valorLido);
if (valorLido > 50){//base de disparo para ligar o LED intensidade =
   valorLido;
  }
else {
  digitalWrite(Led, LOW); intensidade = 0;
}
Serial.print("Valor Intensidade = ");
Serial.println(intensidade); analogWrite(Led,
intensidade); delay(10);
}
```

Fazer o upload do código para a placa e abrir o monitor serial. O monitor serial é a tela na qual são exibidas as mensagens que o Arduino envia para o computador, e também a tela pela qual envia comandos em tempo real para o Arduino.

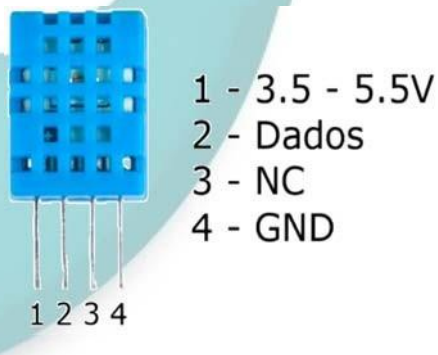
Para ver os dados no monitor serial, basta clicar no botão indicado com a seta vermelha na figura a seguir, no ambiente de desenvolvimento do Arduino.

Figura 44 - Serial na IDE - Fonte: acervo pessoal Samuel Serqueira

4.5 Projeto Sensor de Temperatura e Umidade

Temperatura e umidade são grandezas diretamente ligadas à sensação térmica. A alta umidade durante dias quentes, faz a sensação térmica aumentar, ou seja, a impressão de que está mais calor. Esses valores geralmente são relacionados com outras grandezas, como direção e velocidade do vento, e fornecem uma boa base para a previsão do tempo. Esse projeto faz a leitura desses dois indicadores utilizando apenas um sensor: o sensor de temperatura e umidade DHT11.

O sensor DHT11 usa apenas um fio de sinal para transmitir dados ao Arduino. A alimentação vem de fios separados de VCC e GND. É necessário um resistor de pull-up de 10k ohm entre a linha de sinal (pino 2) e a linha de 5 V (pino 1) para garantir que o nível do sinal permaneça alto por padrão.



Dica: NC significa "não conectado", quando dá nome a terminais de componentes em eletrônica.

Figura 45 - Sensor DHT11
Fonte: https://www.robocore.net/tutoriais/kit-iniciante-v8

Componentes:

- ✓ 1 Arduino UNO R3;
- ✓ 1 Placa Protoboard;
- ✓ 1 Sensor de Temperatura e Umidade DH11;
- ✓ 1 Resistor 10KΩ
- ✓ 10 Jumpers

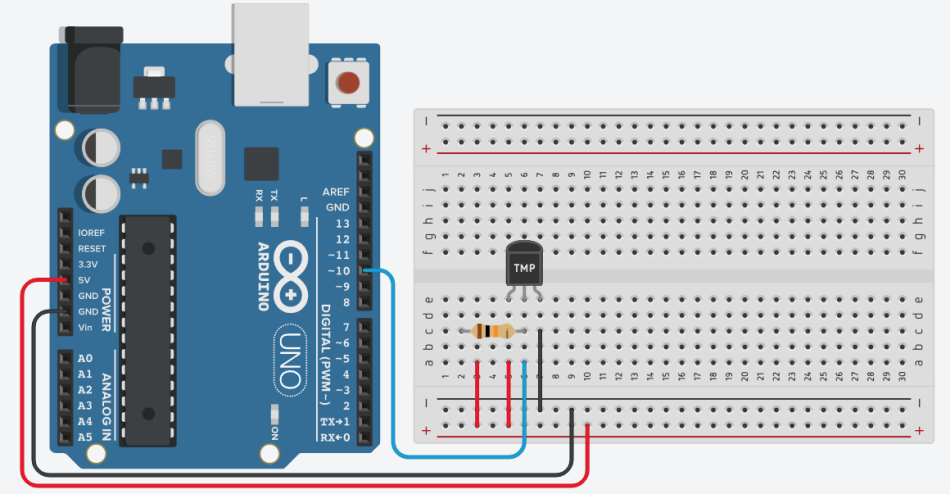


Figura 46 - Sensor de Temperatura e Humidade Fonte: acervo pessoal João Marcos (2022)

Para implementar o protocolo de comunicação com o sensor é necessário utilizar uma biblioteca chamada DHT.h. Internamente a biblioteca DHT.h faz referência à biblioteca Adafruit_Sensor.h, por esse motivo é preciso instalar as duas bibliotecas para que o código- go possa ser compilado. As bibliotecas se encontram para download nos endereços abaixo:

Biblioteca DHT - https://codeload.github.com/adafruit/DHT-sensor-library/zip/master

Biblioteca Adafruit Sensors – https://codeload.github.com/adafruit/Adafruit_Sensor/zip/master

Para adicionar as bibliotecas, após o download, abrir a IDE do Arduino no menu Sketch -> Incluir Biblioteca (Include Library) -> Adicionar Biblioteca .ZIP (Add .ZIP Library...), veja a Figura 47:

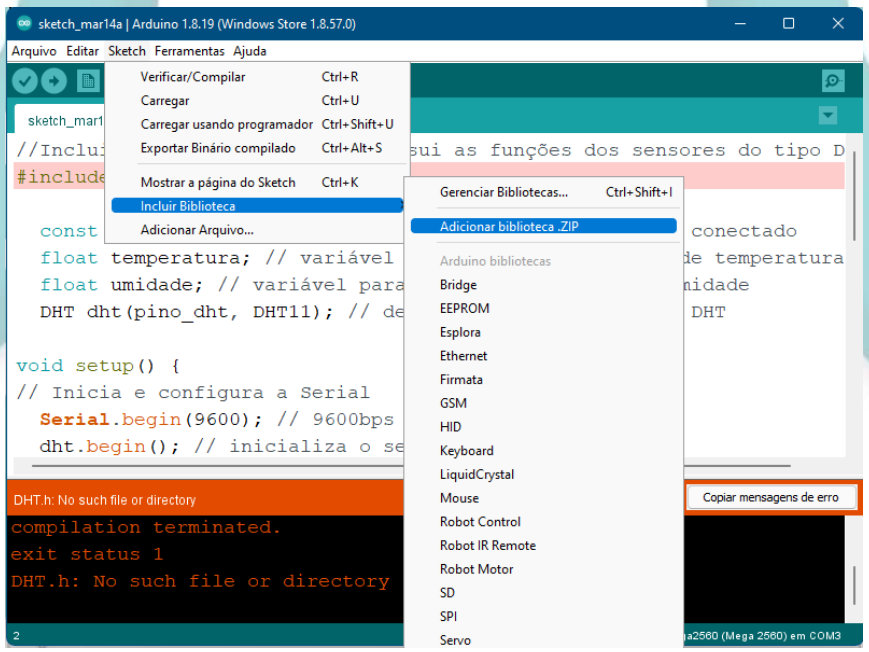


Figura 47 – Adicionando Bibliotecas Fonte: acervo pessoal João Marcos (2022)

Selecionar a biblioteca na pasta onde foi feito o download e clicar em Abrir. Se todos os passos foram seguidos retornará a mensagem "biblioteca adicionada às suas bibliotecas". Repetir os passos para a outra biblioteca.

Carregue o código a seguir no Arduino.

```
      /********Kit Iniciante para Arduino v8 - Utilizando o DHT11**************
 ********Mostrar os valores de Temperatura e Umidade no Monitor Serial********
*********************************************************************\
//Inclui a biblioteca DHT que possui as funções dos sensores do tipo DHT
#include "DHT.h"
const int pino_dht = 9; // pino onde o sensor DHT está conectado
float temperatura; // variável para armazenar o valor de temperatura
float umidade; // variável para armazenar o valor de umidade
DHT dht(pino_dht, DHT11); // define o pino e o tipo de DHT
void setup() {
  // Inicia e configura a Serial
  Serial.begin(9600); // 9600bps
  dht.begin(); // inicializa o sensor DHT
}
void loop() {
  // Aguarda alguns segundos entre uma leitura e outra
  delay(2000); // 2 segundos (Datasheet)
  // A leitura da temperatura ou umidade pode levar 250ms
  // O atraso do sensor pode chegar a 2 segundos
  temperatura = dht.readTemperature(); // lê a temperatura em Celsius
  umidade = dht.readHumidity(); // lê a umidade
  // Se ocorreu alguma falha durante a leitura
  if (isnan(umidade) || isnan(temperatura)) {
    Serial.println("Falha na leitura do Sensor DHT!");
  }
  else { // Se não
    // Imprime o valor de temperatura
    Serial.print("Temperatura: ");
    Serial.print(temperatura);
    Serial.print(" *C ");
    Serial.print("\t"); // tabulação
    // Imprime o valor de umidade
    Serial.print("Umidade: ");
    Serial.print(umidade);
    Serial.print(" %\t");
    Serial.println(); // nova linha
  }
}
```

Primeiro declarar a inclusão da biblioteca DHT.h, responsável por prover uma série de funções que permitem obter os valores de temperatura e umidade. Em seguida são criadas as variáveis que serão utilizadas para armazenar os valores necessários, como o pino onde o sensor está conectado, a temperatura e a umidade.

A instrução dht(pino_dht, DHT11) é responsável por criar um objeto, ou seja, uma estrutura contendo todas as características e funções do sensor que são definidas na biblioteca. Nessa linha é passado em qual pino o sensor está conectado e o modelo de sensor, nesse caso pino 9 e modelo DHT11.

Com as variáveis criadas e inicializadas, são realizadas as configurações iniciais. Após a Serial ser inicializada, o sensor é inicializado através da função dht.begin().

Por último o loop, onde tudo acontece. Ele já inicia com um delay de dois segundos. Isso se faz necessário para respeitar o tempo de amostragem definido pelo fabricante na folha de dados do sensor.

Em seguida, as leituras das duas grandezas procuradas são realizadas e armazenadas através das funções dht.readTemperature() e dht.readHumidity().

Antes de imprimir os valores na serial, uma instrução é utilizada para verificar se os valores obtidos são plausíveis ou se ocorreu alguma falha na comunicação. Essa função é a isnan() . O que ela faz basicamente é retornar verdadeiro se o valor dentro dos parêntesis não for um número e falso caso contrário. Nesse caso, a lógica utilizada é: se o valor de umidade ou o valor de temperatura não forem um número válido, imprima uma mensagem de erro; caso contrário, imprima os valores obtidos.

Agora basta abrir o monitor serial e ver as leituras dos valores de temperatura e umidade. Exponha o sensor a diferentes condições de temperatura e umidade para observar o comportamento das leituras. Uma maneira bem simples de observar mudanças nas leituras é soprar o sensor.

4.6 Projeto Sensor de Chuva

O sensor de chuva é usado para detectar a água, é configurado por duas peças: a placa eletrônica e a placa coletor que coleta as gotas de água.

Figura 48 - Sensor de chuva - Fonte: acervo pessoal Abel Alves

Possui um potenciômetro embutido para ajuste de sensibilidade da saída digital (D0). Também tem um LED de energia que acende quando o sensor é ligado e um LED de saída digital.

Basicamente, a resistência da placa coletor varia de acordo com a quantidade de água em sua superfície. Molhada: a resistência aumenta e a tensão de saída diminui, Seco: a resistência é menor, e a tensão de saída é maior.

Componentes:

- ✓ 1 Arduíno;
- ✓ 1 Módulo sensor de chuva;
- ✓ 1 Led Vermelho;
- ✓ 1 Led Verde;
- ✓ 3 Jumpers;
- ✓ 1 Placa Protoboard;
- ✓ 2 Resistores 220 Ohm;

Este é um exemplo simples para entender como usar o sensor de chuva. Ele apenas mostra os valores do sensor analógico imprimindo essas leituras no monitor serial do IDE.

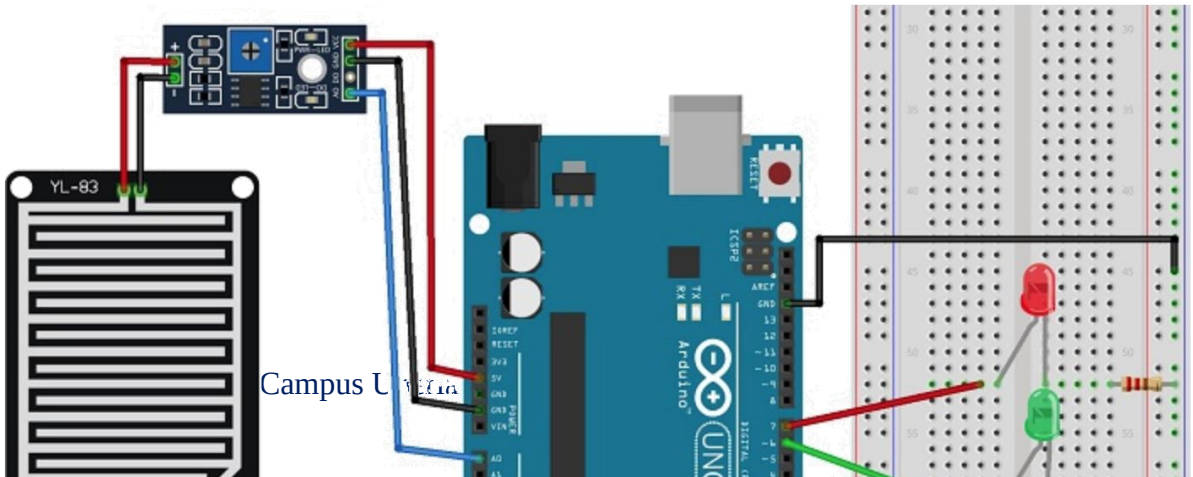


Figura 49 - Sensor de Chuva – Fonte: https://randomnerdtutorials.com/guide-for-rain-sensor-fc-37-or-yl-83-with-arduino/#more-22600

Faça o upload do sketch para o Arduino (fique à vontade para ajustar a variável thresholdValue com um valor limite diferente):

```
    /*********Kit Iniciante para Arduino v8 - Utilizando o DHT11*************
  *************Projeto sensor de chuva e a porcentagem da umidade*************
 ************************************************************************\
//Declaração das variáveis
int rainPin = A0;              // Pin da chuva
int Ledverde = 6;              //Led verde no pino 6
int Ledvermelho = 7;           //Led vermelho no pino 7
//pode ajustar o valor limite
int thresholdValue = 500;      // valor limitado
void setup(){
  pinMode(rainPin, INPUT);
  pinMode(Ledverde, OUTPUT);
  pinMode(Ledvermelho, OUTPUT);
  digitalWrite(Ledverde, LOW);
  digitalWrite(Ledvermelho, LOW);
  Serial.begin(9600);
}
void loop() {
```

```
  //leia a entrada no pino analógico 0:
  int sensorValue = analogRead(rainPin);
  Serial.print(sensorValue);
  if(sensorValue < thresholdValue){
  Serial.println(" – Está chovendo");
    digitalWrite(Ledverde, LOW);          // apaga o Led verde
    digitalWrite(Ledvermelho, HIGH);      // acende o Led vermelho
  }
  else {
    Serial.println(" – Não vai chover");
    digitalWrite(Ledverde, HIGH);         // acende o Led verde
    digitalWrite(Ledvermelho, LOW);       // apaga o Led vermelho
  }
  delay(500);                             // pausa de 0.5 segundos
}
```

4.7 Projeto Variando as Cores do LED RGB com Potenciômetro

Este projeto demonstra uma variedade de cores do LED RGB com o potenciômetro. Movimentando o potenciômetro da direita para a esquerda, o LED-RGB vai alterando a sua cor de branco para o preto. Apresentando sete cores diferentes.

Componentes:

- ✓ 1 Arduíno;
- ✓ 3 Resistores 330Ω;
- ✓ 3 Resistores 10 KΩ;
- ✓ 1 Potenciômetro 10 KΩ;
- ✓ 1 Led RGB (catodo comum);
- ✓ 11 Cabos jumpers;
- ✓ 1 Placa Protoboard;

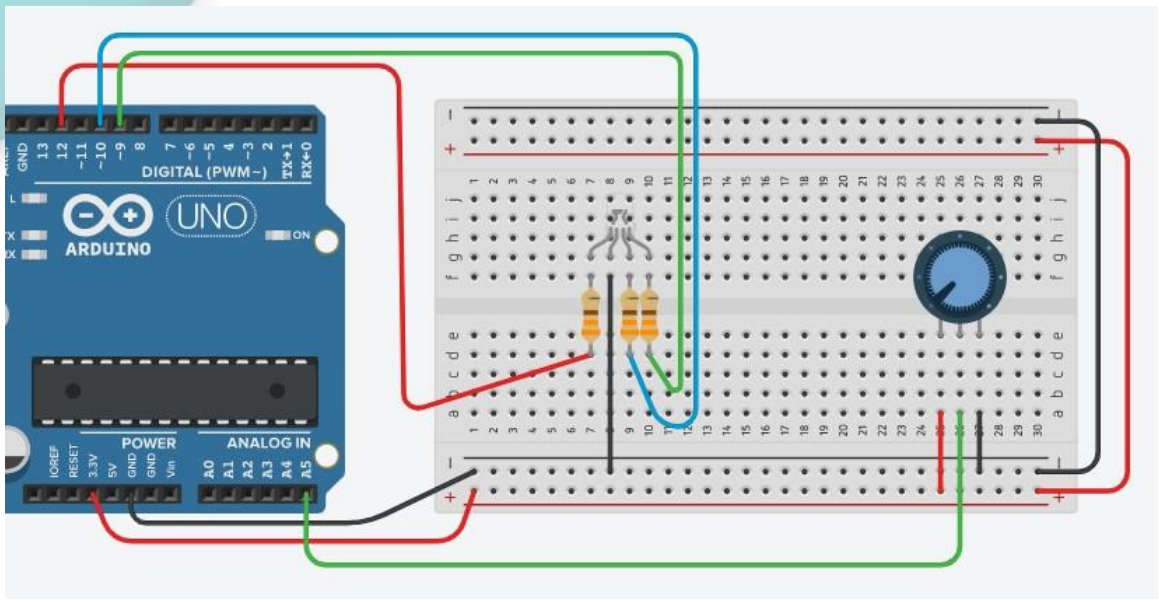


Figura 50 – LED-RGB e Potenciômetro – Fonte: acervo pessoal de Walteno Martins Parreira Júnior (2020)/

O LED RGB nada mais é que três Leds em um só, ele é formado por um vermelho (R de red), um verde (G de green) e um azul (B de blue) que são cores primarias da cor-luz. Associando as cores e intensidade do brilho dos três Leds é possível se obter várias possibilidades de cores.

```
int R = 12;
int B = 10;
int G = 9;
int potenciometro = 5;

void setup()
{
  Serial.begin(9600);
  pinMode(R, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(G, OUTPUT);
}

//Funções para executar o brilho
selecionado
void redFunc(){          //vermelho
  digitalWrite(G,LOW);
  digitalWrite(B,LOW);
  digitalWrite(R,HIGH);
}
void blueFunc(){          //azul
  digitalWrite(G,LOW);
  digitalWrite(B,HIGH);
  digitalWrite(R,LOW);
}
void greenFunc(){          //verde
  digitalWrite(G,HIGH);
  digitalWrite(B,LOW);
  digitalWrite(R,LOW);
}
void yellowFunc(){          //amarelo
  digitalWrite(G,50);
  digitalWrite(B,0);
  digitalWrite(R,255);
}
void purpleFunc(){          //roxo
  digitalWrite(G,0);
  digitalWrite(B,207);
  digitalWrite(R,255);
}
void whiteFunc(){          //branco
  digitalWrite(G,HIGH);
  digitalWrite(B,HIGH);
  digitalWrite(R,HIGH);
}
// final das funções
```

```
void loop()
{
  //potenciometro simula a leitura da
temperatura
  float sinal;
  sinal = analogRead(potenciometro);
  Serial.println(sinal);
  if(sinal>=0 && sinal <=150)
    whiteFunc();
  else if(sinal>150 && sinal <=300)
                    blueFunc();
    else if(sinal>300 && sinal<=450)
                    greenFunc();
      else if(sinal>450 && sinal<=600)
                     yellowFunc();
        else if(sinal>600 && sinal<=750)
                    redFunc();
             else whiteFunc();
}
```

4.8 Projeto Variando tonalidade da Cor do LED RGB com Potenciômetro

Este projeto demonstra as cores do LED RGB com o potenciômetro.
O projeto tem três botões para selecionar as cores vermelho, verde ou azul em um Led RGB, e os seus brilhos são ajustados por um potenciômetro. Quando nenhum botão é acionado, o Led RGB exibe a combinação de cores e brilhos definidos anteriormente pelo potenciômetro.
Fazendo o controle do brilho de cada uma das cores de um Led RGB obtém diversas cores diferentes. Combinando os valores de cada cor é possível criar até 16,7 milhões de cores diferentes (256 X 256 X 256).

Componentes:

- ✓ 1 Arduino;
- ✓ 3 Resistores 330Ω;
- ✓ 3 Resistores 10 KΩ;
- ✓ 3 Botão (push-button);
- ✓ 1 Potenciômetro 10 KΩ;
- ✓ 1 Led RGB (catodo comum);
- ✓ 20 Cabos jumpers;
- ✓ 1 Placa Protoboard;

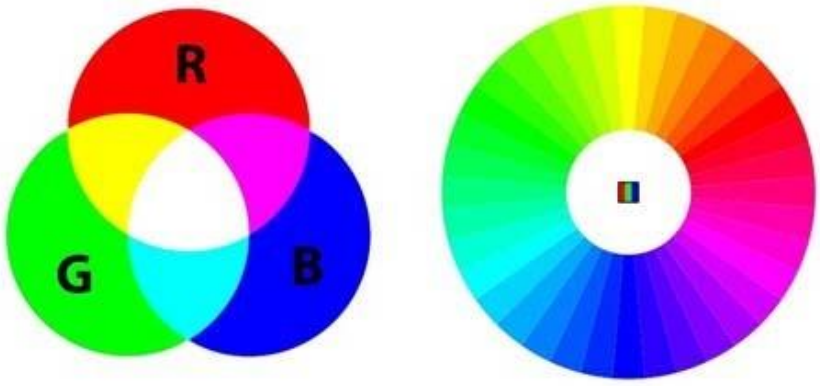
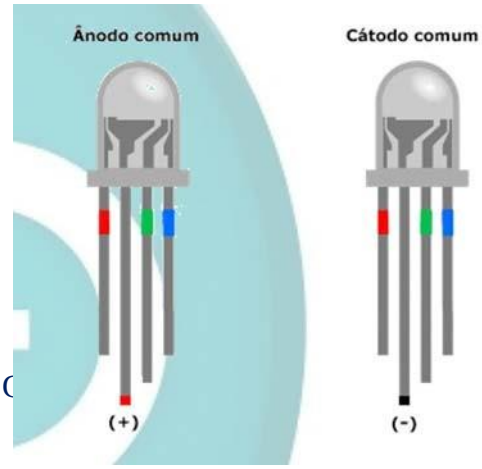


Figura 50 - Escalas de cores RGB - Fonte: acervo pessoal Samuel Serqueira

Existem dois tipos de LED RGB, o ânodo comum e o cátodo comum. Em cada uma das cores existe um terminal, e além disso o quarto terminal (maior) deverá ser conectado ao polo positivo (ânodo comum) ou ao polo negativo (cátodo comum).



Podem existir no mercado outros tipos de LED RGB com configurações dos terminais diferentes dos LEDs RGB comuns. Por isso é importante verificar com o vendedor o tipo de LED adquirido. Caso o LED RGB tenha os pinos configurados conforme figura abaixo ou de outro tipo, basta ajustar as variáveis globais R, G e B do sketch do projeto.
Este projeto há um botão para selecionar as cores em um Led RGB, vermelho, verde ou

azul, que terão os seus brilhos ajustados por um potenciômetro. Outro botão seletor acionará o segundo Led RGB que exibirá a cor definida pelos valores ajustados anteriormente pelo potenciômetro.

Figura 51 - LEDs RGB
Fonte: Acervo pessoal Samuel Serqueira

Com o controle do brilho de cada um dos leds encapsulado em um Led RGB e misturando estes brilhos, produz diversas cores. Combinando os valores de cada cor é possível criar até 16,7 milhões de cores diferentes (256 X 256 X 256).

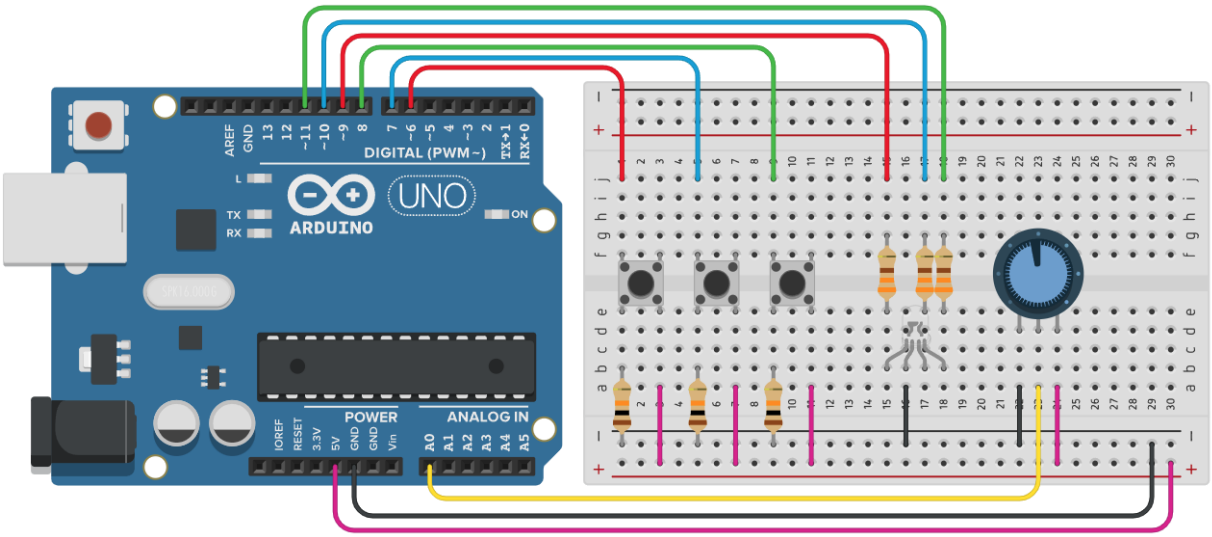


Figura 52 - Variando Cores do LED RGB com o Potenciômetro - Fonte: acervo pessoal João Marcos (2022)

```
/**********Variando Cores do LED RGB com o Potenciômetro**************
**************Samuel Serqueira 2019*****************************
*****************************************************************/
// Define as conexões para cada cor do RGB
const int verde = 11;
const int azul = 10;
const int vermelho = 9;
// Armazena os valores para cada cor entre (0-255)
int valorVermelho = 255;
int valorVerde;
int valorAzul;
// Define as conexões dos botões seletores
const int botVerde = 8;
const int botAzul = 7;
const int botVermelho = 6;
int estadoBoVermelho = 0;
int estadoBoVerde = 0;
int estadoBoAzul = 0;
int auxVm = 0;
int auxVd = 0;
int auxAz = 0;
// Define a conexão do potenciômetro
const int pot = A0;
//Define a cor que está sendo controlada
char cor;
void setup(){
// Define Monitor Serial (visualização de dados)
  Serial.begin(9600); // taxa de comunicação entre Arduíno e PC
// Indica que os pinos dos botões são de SAÍDA do Arduíno
    pinMode(vermelho, OUTPUT);
    pinMode(verde, OUTPUT);
    pinMode(azul, OUTPUT);
// Indica que os pinos são ENTRADA do Arduíno
    pinMode(botVermelho, INPUT);
    pinMode(botVerde, INPUT);
    pinMode(botAzul, INPUT);
    pinMode(pot, INPUT);
}
void loop(){
  //Verifica se os botões seletores estão acionados
  estadoBoVermelho = digitalRead(botVermelho);
  estadoBoVerde = digitalRead(botVerde);
  estadoBoAzul = digitalRead(botAzul);
  //se algum botão for acionado
  if (estadoBoVermelho == HIGH) {//se botão pressionado
    auxVm = !auxVm;//muda o estado da variável auxiliar
  }
  if (estadoBoVerde == HIGH) {//se botão pressionado
    auxVd = !auxVd;//muda o estado da variável auxiliar
  }
```

```
 if (estadoBoAzul == HIGH) {//se botão pressionado
   auxAz = !auxAz;//muda o estado da variável auxiliar
 }
 if (auxVm == HIGH){// enquanto o estado do botão vermelho estiver alto
   valorVermelho = map(analogRead(pot),0,1023,0,255);
   analogWrite(vermelho,valorVermelho);
   analogWrite(verde,0);
   analogWrite(azul,0);
   cor = 'V';
 }
 else if (auxVd == HIGH){// enquanto o estado do botão verde estiver alto
   valorVerde = map(analogRead(pot),0,1023,0,255);
   analogWrite(vermelho,0);
   analogWrite(verde,valorVerde);
   analogWrite(azul,0);
   cor = 'v';
 }
 else if (auxAz == HIGH){// enquanto o estado do botão azul estiver alto
   valorAzul = map(analogRead(pot),0,1023,0,255);
   analogWrite(vermelho,0);
   analogWrite(verde,0);
   analogWrite(azul,valorAzul);
   cor = 'A';
 }
 else if (auxVm == LOW && auxVd == LOW && auxAz == LOW){
   // Atualiza o Led RGB com a cor da mistura
   analogWrite(vermelho, valorVermelho);
   analogWrite(verde, valorVerde);
   analogWrite(azul, valorAzul);
   cor = 'T';
 }
 monitorSerial();
}
void monitorSerial(){
//mostra os valores no PC - Monitor Serial valor do potenciô-metro

  Serial.print("Vermelho = ");
  Serial.print(valorVermelho);
  Serial.print(" | Verde = ");
  Serial.print(valorVerde);
  Serial.print(" | Azul= ");
  Serial.println(valorAzul);

 /*Serial.print("Estado Botão Vermelho = ");
  Serial.print(estadoBoVermelho);
  Serial.print(" | Estado Botão Verde = ");
  Serial.print(estadoBoVerde);
  Serial.print(" | Estado Botão Azul= ");
  Serial.println(estadoBoAzul);
  Serial.print("Auxiliar Vermelho = ");
```

```
  Serial.print(auxVm);
  Serial.print(" | Auxiliar Verde = ");
  Serial.print(auxVd);
  Serial.print(" | Auxiliar Azul= ");
  Serial.println(auxAz);
  Serial.println(" ");*/
  Serial.print ("A cor = ");
  switch (cor){
   case 'V':
     Serial.println("Vermelho");
     break;
   case 'v':
     Serial.println("Verde");
     break;
   case 'A':
     Serial.println("Azul");
     break;
   case 'T':
     Serial.println("Todas");
     break;
  }
  delay(100);
}
```

Uma forma de testar é, após iniciar o sketch, o Led RGB ficará aceso na cor vermelho. Para ajustar o brilho do led, selecionar pressionando o primeiro botão, e depois girando o eixo do potenciômetro. O sentido do eixo do potenciômetro será definido pela polaridade, ou seja, no sentido horário o brilho do led aumentará.

Para visualizar os valores definidos pelo potenciômetro, clique no ícone monitor serial do IDE do Arduino.
Girando o eixo do potenciômetro altera o brilho e o valor da variável. Para criar uma cor, por exemplo a cor Orchild2, ou seja R = 238, G = 122 e B = 233. Então, vamos ajustar primeiro o vermelho para 238.
Ao clicar no botão novamente, seleção é desliga e a cor é acrescentada as outras no RGB. Clicar no seletor verde (botão no centro), alterna a cor para verde e ao clicar de novo desliga a seleção, e o outro (botão mais à direita) alterna para o azul. Através do potenciômetro, faz os ajustes de brilho nas cores até atingir os valores da cor desejada (R = vermelho, V= Verde e B = azul).

Utilizar um difusor (cilindro de papel A4, por exemplo) para que a mistura de cores fique mais agradável, fazendo com que a visualização fique melhor para a cor resultante.

Referências

ARAÚJO, Thayron. **Sensor Ultrassônico com Arduino**. Disponível em: <http://blog.fazedores.com/sensor-ultrassonico-com-arduino/http://squids.com.br/>, Acesso em: 29 mai. 2018.

ELETROGATE. **Apostila Arduino Básico**. Disponível em: <http://apostilas.eletrogate.com/Apostila_Arduino_Basico-V1.0-Eletrogate.pdf/>, Acesso em: 18 mai. 2018.

HACHOUCHE, Anwar S. **Apostila Arduíno Básico V.0**. Eletrogate, Disponível http://apostilas.eletrogate.com, acesso em 20 jun 2017

MCROBERTS, Michael. **Arduino Básico**, tradução Rafael Zanolli - São Paulo, Novatec Editora, 2011.

MECATRONIZANDO. Disponível em: <http://www.mecatronizando.com.br/2016/08//>, Acesso em: 22 mai. 2018.

NGELO, Projeto 26. **Criando cores com Led RGB com 1 potenciômetro** (simplificado), 25 set 2017, disponível http://www.squids.com.br/arduino/index.php/projetos-arduino/projetos-basicos/106-criando-cores-com-led-rgb-com-1-potenciometro-simplificado, acesso em 15 mar 2019.

RANDOM Nerd Tutorials. **Guide for Rain Sensor FC-37 or YL-83 with Arduino**. Disponível em: <http://randomnerdtutorials.com/guide-for-rain-sensor-fc-37-or-yl-83-with-arduino/>, Acesso em: 25 mai. 2018.

REVISTA do Arduino. Disponível www.revistadoarduino.com.br, acesso em 11 fev 2013.

ROBOCORE. **Apostila Kit Iniciante para Robótica**. disponível https://www.robocore.net/tutoriais, acesso em 01 fev 2019.

ROBOCORE. **Apostila Kit Iniciante V7.1 para Arduino**. 2018.

ROBOCORE. **Apostila Kit Iniciante V8 para Arduino**. disponível https://www.robocore.net/tutoriais, acesso em 01 fev 2019.

ROBOCORE. **Lendo um botão**. Disponível em: https://www.robocore.net/tutoriais/kit-iniciante-v8-lendo-um-botao.html

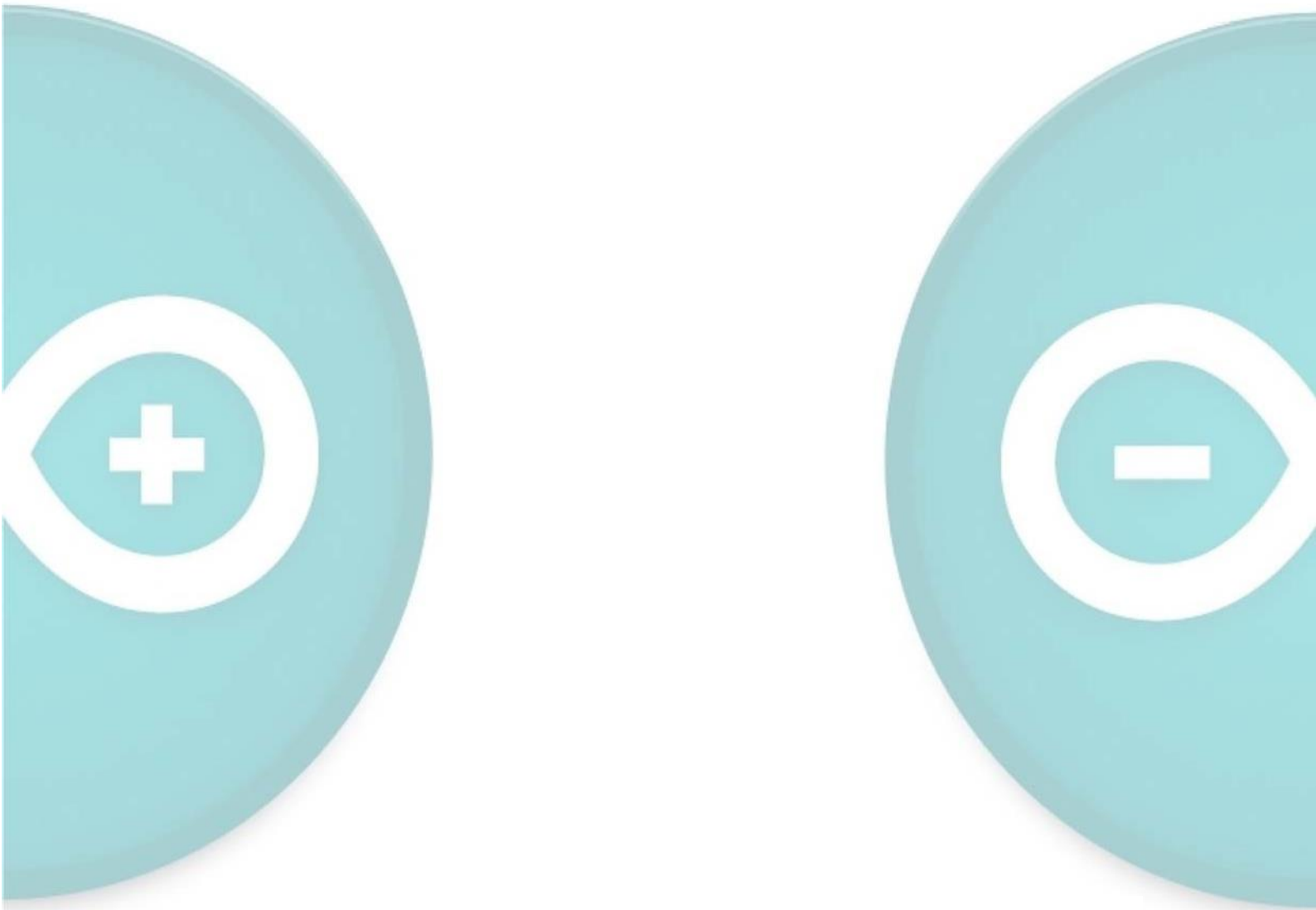
SANTOS, Nuno Pessanha, **Arduíno** – Introdução e Recursos Avançados, Escola Naval, Departamento de Engenheiros Navais, Ramo de Armas e Electrónica, 2009;

SANTOS, Rui; SANTOS, Sara. **Guia para o sensor de chuva FC-37 ou YL-83 com Arduino**. Random Nerd Tutorials, disponível http://randomnerdtutorials.com/guide-for-rain-sensor-fc-37-or-yl-83-with-arduino/, Acesso em 25 mai 2018;

SILVEIRA, João Alexandre da. **Arduino**: Cartilha para programação em C.

SOUZA, Fábio. **Arduino UNO**. 2013. Disponível em: <https://www.embarcados.com.br/arduino-uno/>, Acesso em: 29 mai. 2018.

INSTITUTO FEDERAL
Triângulo Mineiro
Campus Uberlândia Centro



GPETEC
Grupo de Pesquisa em Educação, Tecnologia e Ciências.

CRIAR