

The background of the cover is a complex, abstract digital pattern. It features a dense network of intersecting lines in vibrant green, blue, and red. These lines are overlaid on a grid of small, semi-transparent squares. The overall effect is one of high-tech connectivity and data flow, with a perspective that suggests depth and movement.

Aprenda Arduino

Uma abordagem prática

Cláudio Luís Vieira Oliveira
Humberto Augusto Piovesana Zanetti
Cristina Becker Matos Nabarro
Júlio Alberto Vansan Gonçalves

Cláudio Luís Vieira Oliveira
Humberto Augusto Piovesana Zanetti
Cristina Becker Matos Nabarro
Júlio Alberto Vansan Gonçalves

Aprenda Arduino

Uma abordagem prática

Copyright © 2018, Cláudio Luís Vieira Oliveira, Humberto Augusto Piovesana Zanetti, Cristina Becker Matos Nabarro e Júlio Alberto Vansan Gonçalves

Editoração, fotografias, ilustrações e revisão ortográfica:
Cláudio Luís Vieira Oliveira, Humberto Augusto Piovesana Zanetti,
Cristina Becker Matos Nabarro e Júlio Alberto Vansan Gonçalves

Capa:
Claudia Baptistella Oliveira

Todos os direitos reservados e protegidos pela Lei 9610 de 19/02/1998. Todas as informações contidas nesta obra são de exclusiva responsabilidade dos autores. Nenhuma parte desta obra pode ser reproduzida ou transmitida por qualquer meio, sem prévia autorização por escrito dos autores. O mesmo se aplica às características gráficas e à editoração eletrônica desta obra.

Alguns nomes de empresas e respectivos produtos e/ou marcas foram citados apenas para fins didáticos, não havendo qualquer vínculo dos mesmos com a obra.

Dentro dos códigos de programação, **algumas palavras não foram acentuadas por questões técnicas** relacionadas à/s linguagens de programação utilizadas. Os autores acreditam que todas as informações apresentadas nesta obra estão corretas. Contudo, não há qualquer tipo de garantia de que o uso das mesmas resultará no esperado pelo leitor, principalmente quando se tratar de códigos de programação. Caso seja(m) necessária(s), os autores disponibilizarão errata(s) no site www.profclaudio.com.br.

Dados Internacionais de Catalogação na Publicação (CIP)

O482a Oliveira, Cláudio Luís Vieira Oliveira

Aprenda Arduino – Uma abordagem prática / [texto de Cláudio Luís Vieira Oliveira, Humberto Augusto Piovesana Zanetti, Cristina Becker Matos Nabarro e Júlio Alberto Vansan Gonçalves]. – Duque de Caixas: Katzen Editora, 2018.

181p.

ISBN: 978-85-52946-03-8

1. Informática. 2. Arduino (Controlador Programável). I. Título.

CDD: 005.133

Impresso no Brasil / *Printed in Brazil*

Sobre os Autores

Cláudio Luís Vieira Oliveira

Mestre em Sistemas de Computação pela Pontifícia Universidade Católica de Campinas e bacharel em Análise de Sistemas pela Universidade Metodista de Piracicaba. Possui mais de 26 anos de experiência na área de Ciência da Computação. Coordenador de Curso e Professor da Faculdade de Tecnologia (FATEC) de Jundiaí é também Professor nas Faculdades de Tecnologia (FATEC) de Bragança Paulista e Campinas.

Humberto Augusto Piovesana Zanetti

Doutorando em Tecnologia pela Faculdade de Tecnologia da Universidade Estadual de Campinas (FT - UNICAMP) e Mestre em Ciência da Computação pela Faculdade de Campo Limpo Paulista (FACCAMP). Desde 2005 atuando no ensino técnico e superior. Atualmente professor na Escola Técnica Rosa Perrone Scavone (Itatiba, SP) e na Faculdade de Tecnologia de Jundiaí (FATEC). Na UNICAMP é integrante do LIAG (Laboratório de Informática, Aprendizagem e Gestão).

Cristina Becker Matos Nabarro

Mestranda em Ciência da Computação pela FACCAMP (2017), especialista em Engenharia de Projetos em Sistemas de Informação pela Faculdade e Centro de Educação Tecnológica Radial. Bacharel em Administração com ênfase em Análise de Sistemas pela Faculdade Radial São Paulo, atua há mais de 20 anos na área de TI. É docente da Fatec de Bragança Paulista e da Fatec de Guarulhos e,

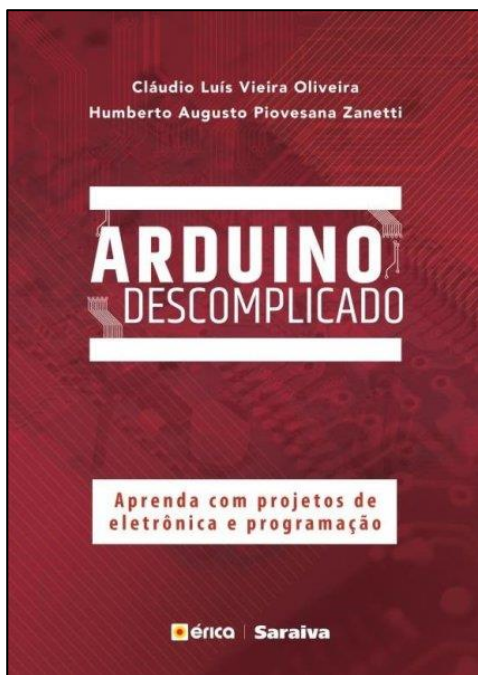
atualmente, também é coordenadora do curso de Gestão de Tecnologia da Informação da Fatec Bragança Paulista.

Júlio Alberto Vansan Golçalves

Mestrando em Ciência da Computação pela FACCAMP (2018), Graduado em Ciência da Computação pelo Centro Universitário Anhanguera (2005). Pós graduado pela UNIP (2013) MBA em Gestão Empresarial, Programa Especial de Formação Pedagógica em Informática - Equivalente à Licenciatura Plena, pela Faculdade Paulista São José (2015). Atua há mais de 20 anos na área de TI. É docente na ETEC Prof. Carmine Biagio Tundisi (Atibaia, SP), ETEC Vasco Antônio Venchiarutti (Jundiaí, SP), ETEC Benedito Storani (Jundiaí, SP), ETEC Rosa Perrone Scavone (Itatiba, SP) e na Faculdade de Tecnologia de Jundiaí (FATEC).

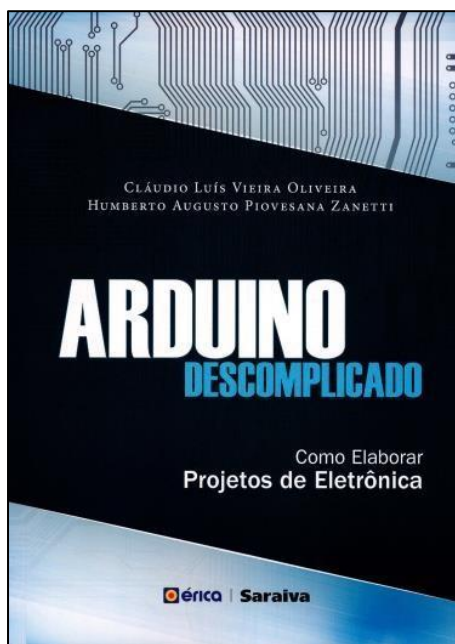
Conheça Também...

Escrito por Cláudio Luís Vieira Oliveira e Humberto Augusto Piovesana Zanetti, o livro **Arduino Descomplicado – Aprenda com projetos de eletrônica e programação**, apresenta aos leitores uma abordagem prática, descomplicada e divertida. Para ajudar ainda mais a compreensão e a execução dos 37 projetos propostos nesta obra, utilizaremos módulos, cujo objetivo é trazer uma solução pronta e com baixa abstração, sem que haja a necessidade de ter conhecimentos sobre a montagem de circuitos para usar os componentes eletrônicos. Assim, o foco passa a ser as funcionalidades e os recursos de programação. Para a programação, duas abordagens são adotadas: linhas de código e programação em blocos. Na programação em linhas de código, aplicaremos a linguagem padrão do Arduino, a linguagem Wiring. Já na programação em blocos, adotaremos a ferramenta Snap!, que cada vez mais ganha adeptos e está adaptada à plataforma Arduino.



Conheça Também...

O livro **Arduino Descomplicado – Como Elaborar Projetos de Eletrônica**, escrito por Cláudio Luís Vieira Oliveira e Humberto Augusto Piovesana Zanetti, apresenta os conceitos básicos e avançados que envolvem o Arduino, considerado o principal marco na história do hardware livre. O Arduino cria uma ponte entre os mundos das ideias, da eletrônica e computação, possibilitando que uma ideia saia do papel e se transforme em realidade de maneira rápida e simples. A prototipação rápida é um convite à experimentação. Este livro traz, ainda, a história da ferramenta e analisa conhecimentos de eletrônica



básica. Orienta quanto à criação do primeiro projeto, destacando os principais componentes e o processo de montagem do circuito. Discorre sobre entradas e saídas digitais e analógicas, porta serial, displays, sensores e módulos. Por fim, explica os conceitos de motores e servo-motores, Android e a documentação de projetos com o programa Fritzing.

Conheça Também...

“O Fantástico Mundo do Arduino”, escrito por Ângela Cristina de Oliveira Lühmann e Cláudio Luís Vieira Oliveira, é um livro de tecnologia desenvolvido para o público infantojuvenil e também adultos que estão iniciando seus estudos em lógica de programação e eletrônica. Ele conta a história de três crianças e seu pequeno robô, construído por eles

mesmos. No decorrer da história, a Turma da Casa da Árvore como são chamados, cria projetos que ensinam de uma forma divertida, interativa e didática, as primeiras noções de programação e eletrônica, além de auxiliar no desenvolvimento do raciocínio lógico e matemático.

Nesta obra são explorados conceitos de Computação Física, através do mundialmente conhecido Arduino, sendo este utilizado em conjunto com o ambiente de programação Scratch for Arduino (S4A), que está fundamentado sobre o intuitivo conceito de blocos de montagem criado pelo Massachusetts Institute of Technology (MIT). O S4A não exige conhecimento prévio de outras linguagens de programação, sendo ideal para pessoas que estão começando a programar.



Conheça Também...



O livro **Arduino Simples e Divertido**, também de autoria de Cláudio Luís Vieira Oliveira e Humberto Augusto Piovesana Zanetti, permite explorar todo o potencial do Arduino através de 40 projetos desenvolvidos com os módulos da GBK Robotics.

Os módulos da GBK Robotics simplificam a montagem dos projetos permitindo, desta forma, que se dê um foco maior

nas funcionalidades e nos recursos de programação. Serão utilizados diversos módulos durante todo o livro, iniciando com um simples pisca-pisca, e evoluindo para entradas por botões, sensores (de luminosidade, termômetro, infravermelho, entre outros), controle remoto, display de led e, até mesmo, um pequeno robô.

Desperte seu espírito criativo e comece a ler este livro, que em poucas páginas, você irá perceber o quanto o Arduino é simples de usar e, principalmente, muito divertido!

Acesse www.profclaudio.com.br para conhecer mais sobre estes livros e também para saber como adquirir seu exemplar!

Índice

O que é o Arduino?	13
Conceitos de Eletrônica.....	17
Tensão, Corrente e Resistência	17
Principais Componentes.....	19
Projetos e Desafios.....	25
Projeto 1 – Controle de um LED	25
Desafio 1 – Controle de Semáforo	29
Projeto 2 – Potenciômetro	35
Projeto 3 – LDR	43
Projeto 4 – Buzzer.....	50
Projeto 5 – Botão.....	55
Desafio 2 – Controle de Estufa	61
Projeto 6 – LCD (<i>Liquid Crystal Display</i> 16x2).....	63
Desafio 3 – Controle de Estufa com LCD	70
Projeto 7 – Uso do Sensor de Temperatura	72
Projeto 8 – Termistor	78
Desafio 4 – Termômetro Digital Completo	83
Projeto 9 – Sensor Ultrassônico (HC-SR04)	85
Desafio 5 – Sensor de Estacionamento	89
Projeto 10 – Piezo Elétrico	91
Projeto 11 – Relógio com LCD	96

Projeto 12 – Display de Led de 7 Segmentos	104
Projeto 13 – Display de Led de 7 Segmentos e 4 Dígitos	114
Projeto 14 – Servo Motor	120
Projeto 15 – Sensor Óptico Reflexivo	126
Projeto 16 – Teclado com Divisor de Tensão	133
Projeto 17 – Infravermelho	136
Projeto 18 – Contador Binário	148
Projeto 19 – Contador Binário com Chave Rotativa	153
Desafio 6 – Contador Hexadecimal	163
Projeto 20 – Utilizando Entradas Analógicas como Portas Digitais	164
Projeto 21 – Utilizando INPUT_PULLUP	169
Projeto 22 – Sensor de Presença	173
Desafio 7 – Sensor de Presença com Temporizador	176
Projeto 23 – LED RGB	177
Referências Bibliográficas	180



Capítulo 1: O que é o Arduino?

O Arduino é uma versátil plataforma de prototipagem eletrônica, de hardware e software aberto, de baixo custo e muito fácil de usar, mesmo para pessoas que possuem pouco ou nenhum conhecimento de eletrônica. Quando falamos em Arduino devemos ter em mente três conceitos: **hardware** que é a placa que possui como componente central um microcontrolador da família ATmega; **software** que consiste em uma linguagem de programação e um ambiente de desenvolvimento derivados do Processing. O terceiro conceito é a **comunidade**, composta por um grande número de pessoas que compartilham os seus conhecimentos e projetos na Internet, disseminando a plataforma.

A placa **Arduino** é muito parecida com um computador de pequeno porte, sendo composta pelo microcontrolador, memória RAM, memória secundária (memória flash), clock e comunicação USB entre outras funcionalidades. Na Figura 1.1 temos o modelo mais popular dos Arduinos que é o Uno R3, porém, para os projetos deste livro qualquer outro modelo de Arduino pode ser usado sem restrições.

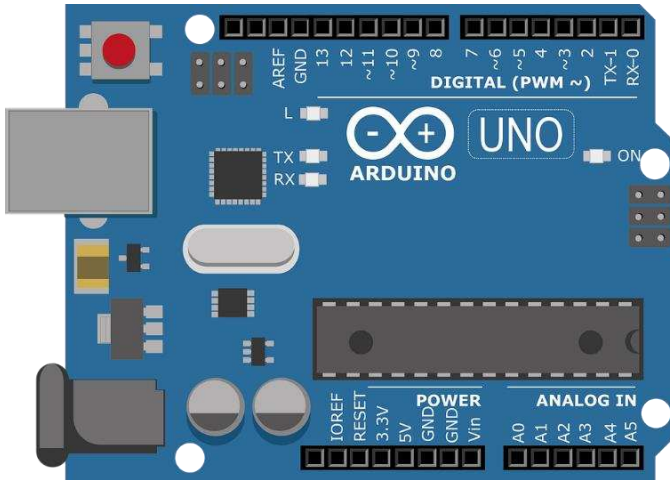


Figura 1.1: Placa Arduino Uno R3

O Arduino Uno R3 apresenta 14 pinos que podem ser utilizados como entradas ou saídas digitais (pinos 0 a 13), sendo que os pinos 3, 5, 6, 9, 10 e 11 também podem utilizar Pulse Width Modulation (PWM) para gerar um conjunto de valores inteiros entre 0 e 255. Os pinos de A0 a A5 correspondem às entradas analógicas, que recebem uma tensão entre 0 e 5V e o produzem em uma escala de 0 a 1023. Também temos os pinos 3,3V, 5V e GND (Terra) permitem alimentar os componentes dos circuitos conectados ao Arduino. Possui um microprocessador ATmega328, com uma memória RAM de 2KB, memória Flash de 32KB e clock de 16MHz.

O ambiente de desenvolvimento do Arduino pode ser gratuitamente baixado do site www.arduino.cc. Neste site e em muitos outros, também estão disponíveis as instruções para realizar a instalação em diversos sistemas operacionais, além de fóruns para tirar dúvidas e obter maiores informações.

O ambiente de programação (Figura 1.2) é muito simples e intuitivo. Um programa, que no Arduino é chamado de **sketch**, apresenta duas funções básicas: **setup()** e **loop()**.

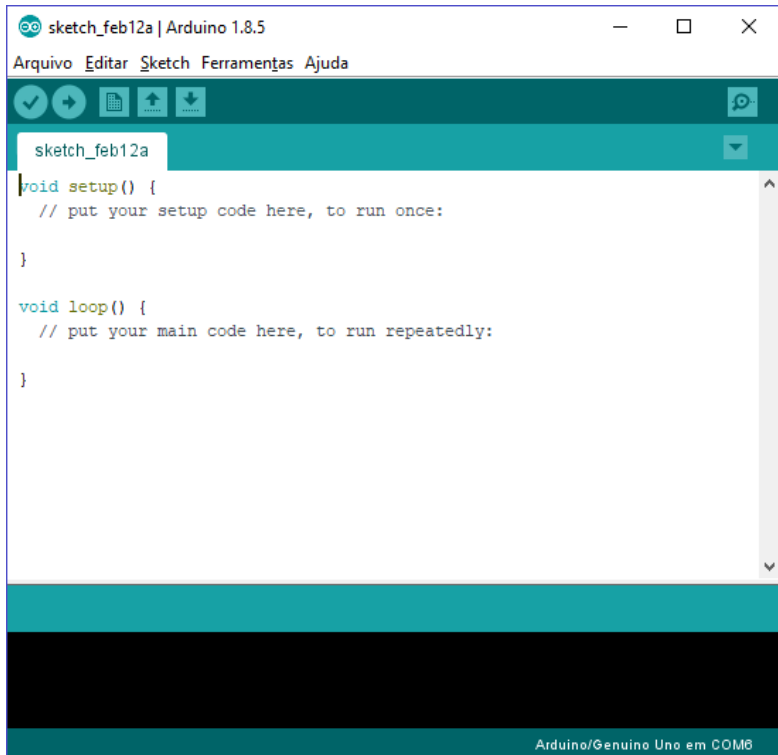


Figura 1.2: Ambiente de Desenvolvimento do Arduino

A função **setup()** deverá conter o código que irá executar apenas uma vez, quando o sketch iniciar. Normalmente colocamos nesta função as definições iniciais do programa.

A função **loop()** irá executar continuamente as instruções que estão lá até que outro sketch seja carregado

na memória “flash” do Arduino. É importante notar que no Arduino é possível armazenar e executar um sketch por vez, desta forma, sempre quando transferimos um sketch esse irá substituir o programa que estava anteriormente carregado na memória. Também observe que como o sketch fica armazenado na memória “flash”, que é permanente, mesmo quando desligamos o Arduino, o programa continua armazenado e irá entrar novamente em execução quando o Arduino for ligado novamente.

Note também que, nestas duas funções, a palavra reservada **void** indica que as funções não apresentam um valor de retorno, sendo usadas exclusivamente para realizar a execução de um conjunto de instruções.



Capítulo 2: Conceitos de Eletrônica

A construção de projetos com o Arduino envolve o conhecimento básico de Eletrônica, o qual irá permitir identificar os componentes que serão utilizados e também entender o seu funcionamento.

Tensão, Corrente e Resistência

A eletrônica está fundamentada sobre os conceitos de tensão, corrente e resistência. Podemos entender como tensão a energia potencial armazenada em uma pilha ou bateria que irá fluir quando um circuito for fechado entre os polos de maior e menor potencial (sentido convencional). Observe na Figura 2.1 que, como analogia, podemos pensar na água armazenada em dois recipientes conectados por um cano. A água irá fluir do recipiente com maior quantidade de água para o menor.

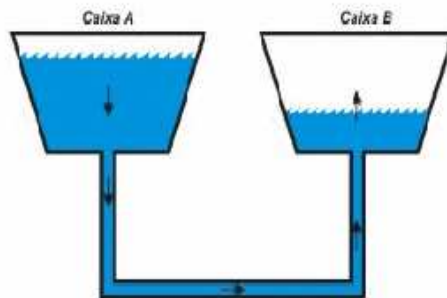


Figura 2.1: Diferença de Potencial

Em eletrônica o princípio é o mesmo, por exemplo, os polos positivos e negativos de uma pilha indicam o sentido na qual a corrente elétrica irá fluir. Desta forma, podemos definir que a corrente elétrica é a movimentação ordenada de cargas elétricas num condutor. Para fins de análise, podemos notar na Figura 2.2 que a corrente elétrica poderá circular em dois sentidos: a) sentido real, que é resultante do movimento de cargas negativas ou; b) sentido convencional – resultante do movimento de cargas positivas.

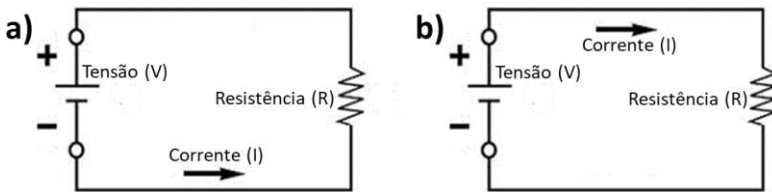


Figura 2.2: Sentido Real (a) e Sentido Convencional (b)

A movimentação das cargas elétricas através do condutor pode encontrar elementos que oferecem certa resistência a passagem dessas cargas. Na figura acima, por exemplo, a resistência a passagem da corrente elétrica faz com que a lâmpada gere calor no seu filamento e fique incandescente. É esse mesmo efeito que permite que a água de um chuveiro seja aquecida ao passar pela resistência.

A Lei de Ohm estabelece a relação entre tensão (V), corrente (I) e resistência (R), onde:

$$I = V / R$$

A tensão é expressa em Volts (V), a corrente em Amperes (A) enquanto a resistência em Ohms (Ω). Desta forma, considerando o circuito elétrico (Figura 2.3) no qual temos uma tensão de 5V aplicada sobre uma resistência de $220\ \Omega$ o que irá produzir uma corrente de 0,022A ou 22mA.

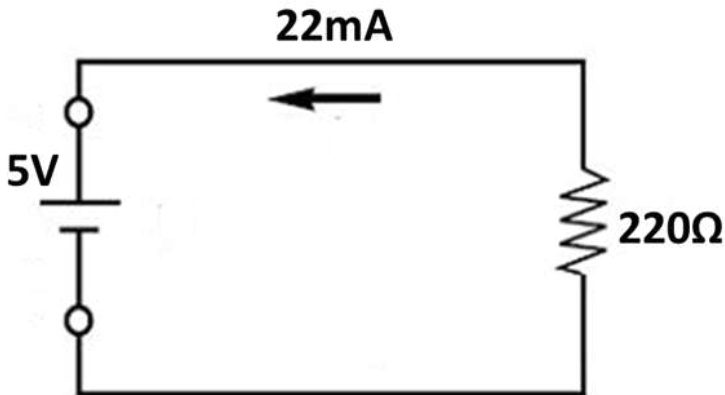


Figura 2.3: Aplicação da Lei de Ohm

Principais Componentes

Um circuito eletrônico é formado por diversos componentes com finalidades distintas, vamos a seguir aprender a identificar os mais utilizados.

Os **resistores** (Figura 2.4), conforme já explicado acima, tem a função limitar a corrente elétrica, eles são necessários do modo a evitar que determinados componentes eletrônicos recebam uma tensão ou corrente maior do que eles podem suportar evitando, desta forma, que os mesmos sejam danificados. São componentes não polarizados, ou seja, podem ser instalados em qualquer

sentido no circuito elétrico, sem preocupação com os polos negativos ou positivos.

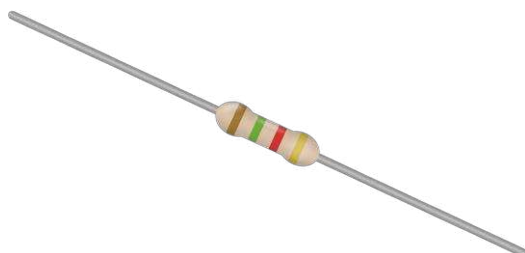
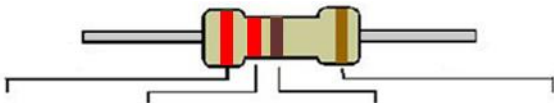


Figura 2.4: Resistor

O valor de um resistor pode ser determinado através de uma tabela código de cores, conforme ilustra a Figura 2.5.



Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	1Ω	
Marrom	1	1	1	10Ω	± 1% (F)
Vermelho	2	2	2	100Ω	± 2% (G)
Laranja	3	3	3	1kΩ	
Amarelo	4	4	4	10kΩ	
Verde	5	5	5	100kΩ	± 0.5% (D)
Azul	6	6	6	1MΩ	± 0.25% (C)
Violeta	7	7	7	10MΩ	± 0.10% (B)
Cinza	8	8	8		± 0.05%
Branco	9	9	9		
Ouro				0.1	± 5% (J)
Prata				0.01	± 10% (K)

Figura 2.5: Código de Cores para Resistores

Os **capacitores** (Figura 2.6) são componentes que permitem armazenar energia para uma utilização rápida. Por exemplo, se compararmos um capacitor com uma pilha temos que o capacitor pode descarregar toda sua carga em uma pequena fração de segundo, já a pilha demoraria vários minutos para descarregar. Uma aplicação típica de capacitores é no flash de câmera, a pilha (ou bateria) carrega o capacitor por vários segundos, e então o capacitor descarrega toda a carga armazenada para que a lâmpada do flash seja acionada imediatamente. Existem diversos tipos de capacitores sendo alguns polarizados e outros não, a unidade de medida de um capacitor é o Farad (F).

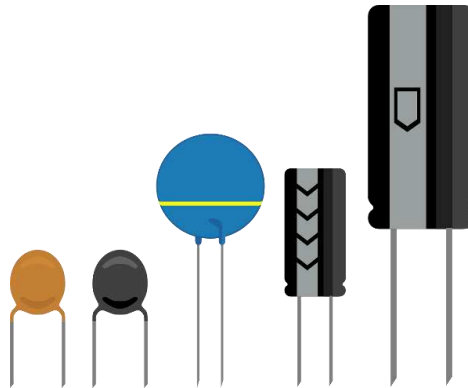


Figura 2.6: Tipos de Capacitores

Um **diodo** (Figura 2.7) é um componente semicondutor que permite que a corrente flua em apenas um sentido. É um componente polarizado, desta forma, o terminal Cátodo que é identificado por uma faixa deve estar sempre conectado ao polo negativo (ou terra) do circuito, enquanto o Ânodo que é o outro terminal deverá estar conectado ao polo positivo.

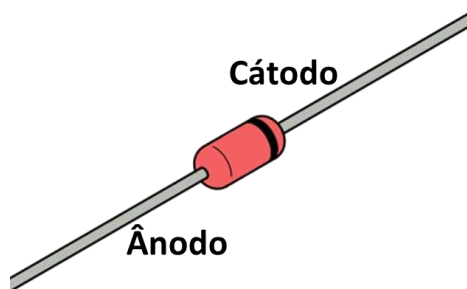


Figura 2.7: Diodo

O **diodo emissor de luz** ou simplesmente **LED** (Figura 2.8) é uma variação do diodo e apresenta, como principal característica, a emissão de luz quando uma corrente flui através do mesmo. É um componente polarizado, desta forma, o Cátodo (lado chanfrado) sempre deve estar conectado ao polo negativo (ou terra) do circuito, se conectado invertido pode ser danificado.

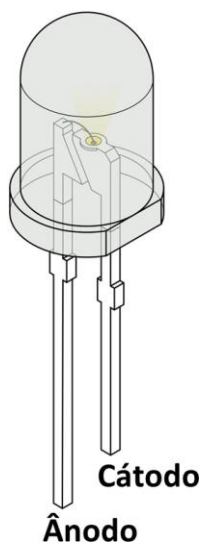


Figura 2.8: Diodo Emissor de Luz (LED)

Os **transistores** (Figura 2.9) são componentes semicondutores e foram os principais responsáveis pela revolução da eletrônica e da informática na década de 1960, pois, permitiram substituir as válvulas nos equipamentos. Um transistor é praticamente cem vezes menor que uma válvula, não necessita de tempo para aquecimento, consome menos energia, sendo muito mais rápido e confiável. Apresenta inúmeras aplicações, sendo as principais, a atuação como uma “chave” eletrônica e amplificador de sinal.

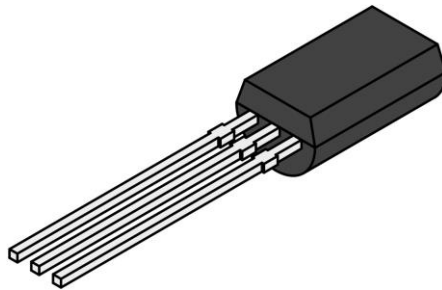


Figura 2.9: Transistor

Um transistor possui três terminais, sendo nomeados como base, coletor e emissor e para identificá-los devemos ter o modelo utilizado e consultar as respectivas especificações (datasheet).

Os **circuitos integrados** (Figura 2.10) consistem em transistores e vários outros componentes eletrônicos miniaturizados e montados num único chip. A integração em larga escala permite colocar cerca de 1 milhão de transistores por mm^2 proporcionando um alto nível de miniaturização dos circuitos eletrônicos além de uma grande redução de custos.

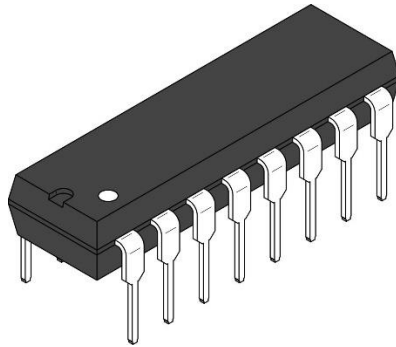


Figura 2.10: Circuito Integrado

Uma **protoboard** (Figura 2.11) permite a montagem provisória de circuitos eletrônicos permitindo a reutilização dos componentes, consiste em uma matriz de contatos interconectados através dos quais os componentes são interligados.

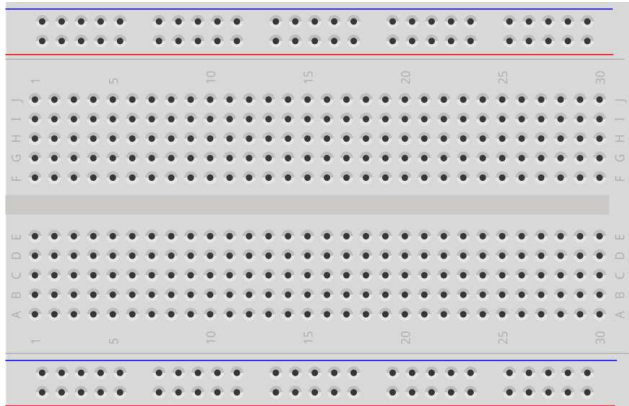


Figura 2.11: Protoboard

Os pinos dos componentes devem ser conectados sempre em linhas diferentes, enquanto conexões entre componentes diferentes devem ocorrer em uma mesma linha.



Capítulo 3: Projetos e Desafios

Neste capítulo desenvolvemos uma sequência de projetos e desafios que abordam os conceitos de entradas e saídas digitais e analógicas e o uso de dispositivos sensores e atuadores.

Projeto 1 – Controle de um LED

O objetivo deste projeto é utilizar uma porta digital do Arduino para controlar o funcionamento de um Diodo Emissor de Luz (LED). Um nível 1 (HIGH) colocado no pino irá acender o LED, enquanto um nível 0 (LOW) vai apagar o LED.

Material necessário:

- 1 Arduino.
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom).
- 1 Led (qualquer cor).
- 1 Protoboard.
- Jumper cable.

Passo 1: Montagem do circuito

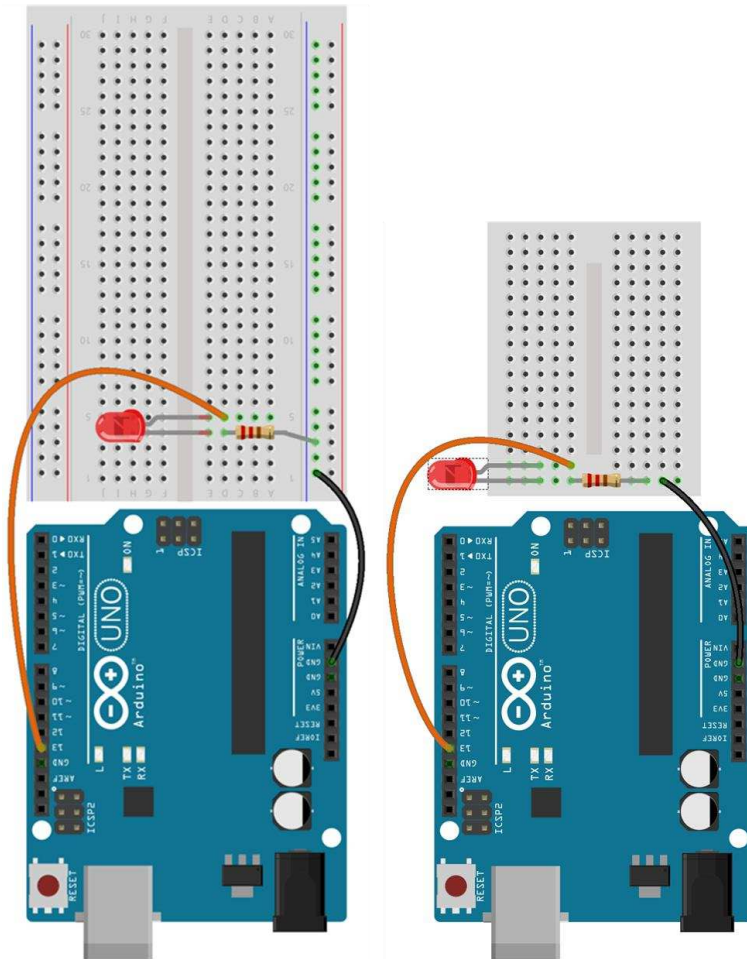


Figura 3.1: Opções de Montagem do Projeto

Observe a Figura 3.1 e de acordo com o modelo de protoboard que estiver usando:

- a. Conecte o pino GND do Arduino à linha de alimentação negativa (preta ou azul) da protoboard.
- b. Coloque o resistor de 220 ohms (ou 330 ohms) entre a linha de alimentação negativa e qualquer outra linha da protoboard.
- c. Coloque o LED com o Cátodo (lado chanfrado) conectado ao resistor.
- d. Conecte o Ânodo do LED ao pino 13 do Arduino.

Passo 2: Programa

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
int LED = 13; // Pino onde o LED foi conectado

void setup() {
  pinMode(LED, OUTPUT); // Definir pino como
                        // saída
}

void loop() {
  digitalWrite(LED, HIGH); // Colocar nível 1 no
                          // pino (liga o LED)
  delay(2000); // Aguardar por 2 segundos
  digitalWrite(LED, LOW); // Colocar nível 0 no
                          // pino (apaga o LED)
  delay(2000);
}
```

Passo 3: Compilação e transferência do programa para o Arduino

Observe a Figura 3.2 e após salvar o sketch (programa), faça a compilação e, em seguida, conecte o Arduino à porta USB do computador. Finalizando, pressione o botão Carregar (Transferir) para fazer a transferência do sketch para o Arduino.

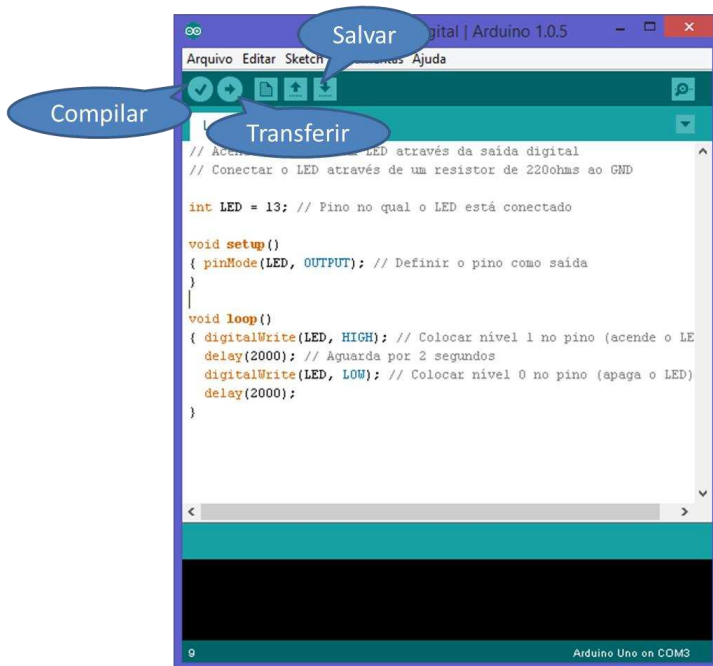


Figura 3.2: Ambiente de Desenvolvimento do Arduino

Desafio 1 – Controle de Semáforo

a) Semáforo de Veículos

Objetivo:

Aplicando o conceito de Saídas Digitais, abordado no Projeto 1, reproduzir o funcionamento de um sinal de trânsito.

Material necessário:

- 1 Arduino.
- 3 Resistores de 220 ohms (vermelho, vermelho, marrom) ou de 330 ohms (laranja, laranja, marrom).
- 3 LEDs (1 vermelho, 1 verde e 1 amarelo).
- 1 Protoboard.
- Jumper cable.

Montagem do circuito

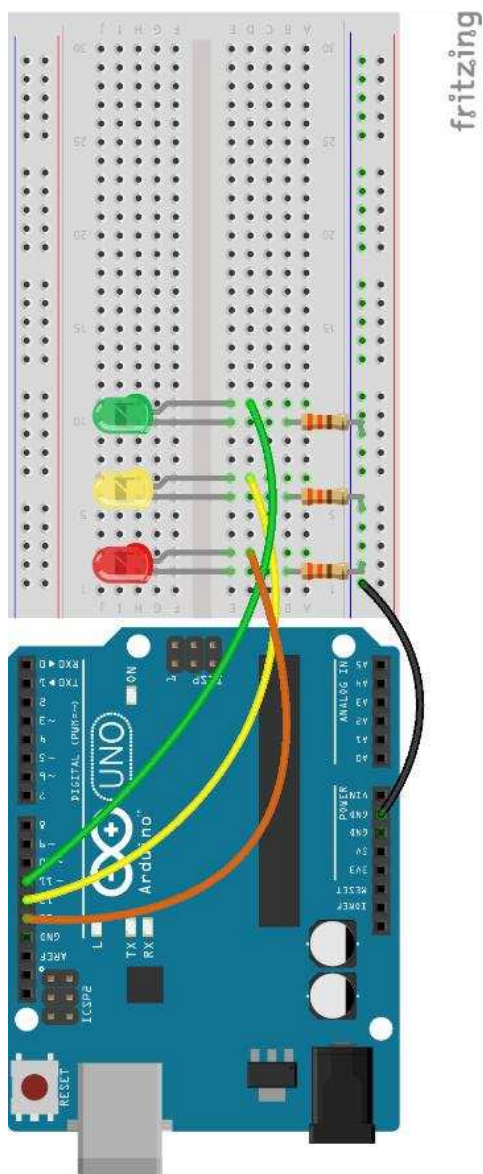


Figura 3.3: Semáforo de Veículos

Realize a montagem do circuito, conforme ilustra a Figura 3.3:

- a. Conecte o pino GND do Arduino à linha de alimentação negativa (azul) do protoboard.
- b. Coloque 5 resistores de 220 ohms (ou 330 ohms) entre com uma ligação na linha de alimentação negativa e qualquer outra linha do protoboard.
- c. Coloque os 3 LEDs com o Cátodo (lado chanfrado) conectado em cada um dos resistores.
- d. Conecte o Ânodo dos LEDs na seguinte ordem: pino 13 do Arduino em um LED vermelho, pino 12 do Arduino em um LED amarelo e pino 11 do Arduino em um LED verde.

b) Semáforo de Veículos e Pedestres

Objetivo:

Reproduzir um cenário similar ao de um semáforo de veículos e pedestres. Supondo o estado inicial do cenário com semáforo de veículos (VEÍCULO) sendo vermelho (PARE) e o semáforo de pedestres (PEDESTRE) sendo verde (SIGA), deve-se programar a sequência de luzes indicando os estados do semáforo de veículos sincronizado com os estados do semáforo de pedestres. Algumas especificações a serem seguidas:

- O sinal vermelho e sinal verde de VEÍCULO tem duração de 10 segundos cada.
- O sinal amarelo de VEÍCULO tem duração de 2 segundos.

- O sinal vermelho de PEDESTRE ficará acesso durante todo o tempo que o sinal vermelho e sinal amarelo de VEÍCULO estiverem acessos, impedindo a passagem de pedestres enquanto os carros transitam.
- O sinal verde de PEDESTRE ficará acesso durante todo o tempo que o sinal vermelho de VEÍCULO ficar acesso, indicando que os pedestres estão livres para atravessar.
- Antes transição do sinal verde para o vermelho de PEDESTRE, faltando 2 segundos para a transição, o sinal verde pisca rapidamente 2 vezes indicando aos pedestres que se tornará vermelho.

Material necessário:

- 1 Arduino.
- 5 Resistores de 220 ohms (vermelho, vermelho, marrom) ou de 330 ohms (laranja, laranja, marrom).
- 5 Leds (2 vermelhos, 2 verdes e 1 amarelo).
- 1 Protoboard.
- Jumper cable.

Montagem do circuito

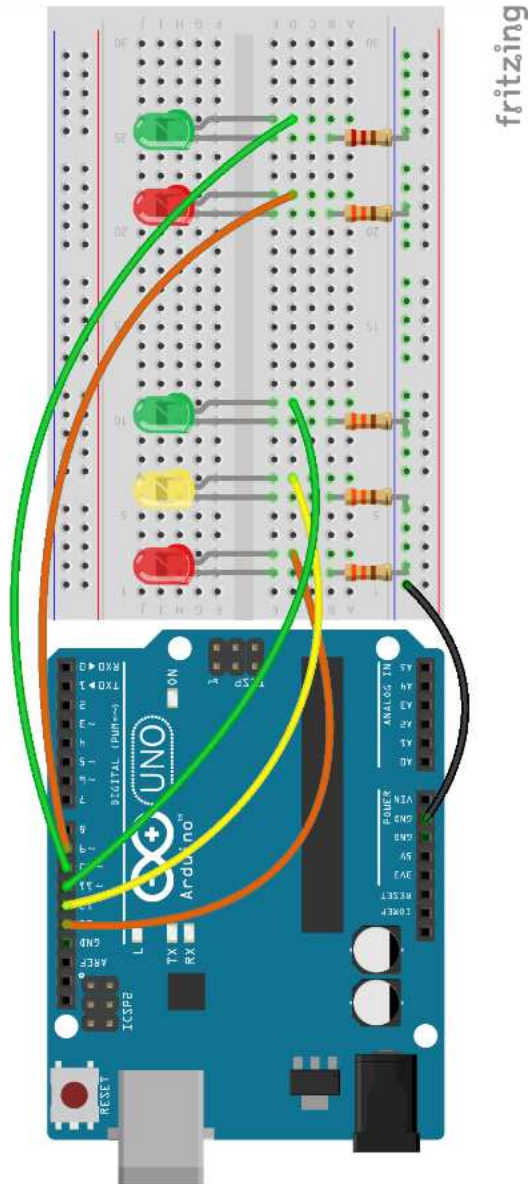


Figura 3.4: Semáforo de Veículos e Pedestres

Conforme ilustra a Figura 3.4 realize os seguintes passos:

- a. Conecte o pino GND do Arduino à linha de alimentação negativa (azul) do protoboard.
- b. Coloque 5 resistores de 220 ohms (ou 330 ohms) entre com uma ligação na linha de alimentação negativa e qualquer outra linha do protoboard.
- c. Coloque 5 LEDs com o Cátodo (lado chanfrado) conectado em cada um dos resistores.
- d. Conecte o Ânodo dos LEDs na seguinte ordem: pino 13 do Arduino no LED vermelho (vermelho de VEÍCULO), pino 12 do Arduino no LED amarelo (amarelo de VEÍCULO), pino 11 do Arduino no LED verde (verde de VEÍCULO), pino 9 do Arduino no LED vermelho (vermelho de PEDESTRE) e pino 10 do Arduino no LED verde (verde de PEDESTRE).

Projeto 2 – Potenciômetro

O objetivo deste projeto é controlar a frequência de acender e apagar (frequência de pisca-pisca) e a intensidade da luminosidade de um LED. Nesse workshop teremos dois experimentos para alcançar esses objetivos. Um potenciômetro um resistor variável no formato de um botão giratório que fornece um valor analógico. Se girarmos o potenciômetro, alteramos a resistência em cada lado do contato elétrico que vai conectado ao terminal central do botão. Essa mudança implica em uma mudança no valor analógico de entrada. Quando o cursor for levado até o final da escala, teremos 0 volts e assim obtendo o valor 0 na entrada analógica. Quando giramos o cursor até o outro extremo da escala, teremos 5 volts e assim tendo o valor 1023 na entrada analógica. Outro conceito que podemos notar é a utilização dos pinos digitais com a marcação “~” (til) como, por exemplo, o pino digital “~9” usado no Programa N° 2.

Material necessário:

- 1 Arduino.
- 1 Potenciômetro.
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom) para o Led.
- 1 LED de qualquer cor.
- 1 Protoboard.
- Jumper cable.

Passo 1: Montagem do circuito

Conforme ilustra a Figura 3.5:

- a. Conecte o pino 5v do Arduino à linha de alimentação positiva (vermelha) do protoboard.
- b. Conecte o pino GND do Arduino à linha de alimentação negativa (preta) do protoboard.
- c. Conecte um LED utilizando um resistor de 220 ohms (ou 330 ohms).
- d. Conecte o LED no pino digital 13.
- e. Conecte o potenciômetro na protoboard com o botão de girar virado para você.
- f. Conecte o pino da esquerda do potenciômetro na linha de alimentação GND.
- g. Conecte o pino da direita do potenciômetro na linha de alimentação positiva.
- h. Conecte o pino do centro do potenciômetro no pino analógico A1 do Arduino.

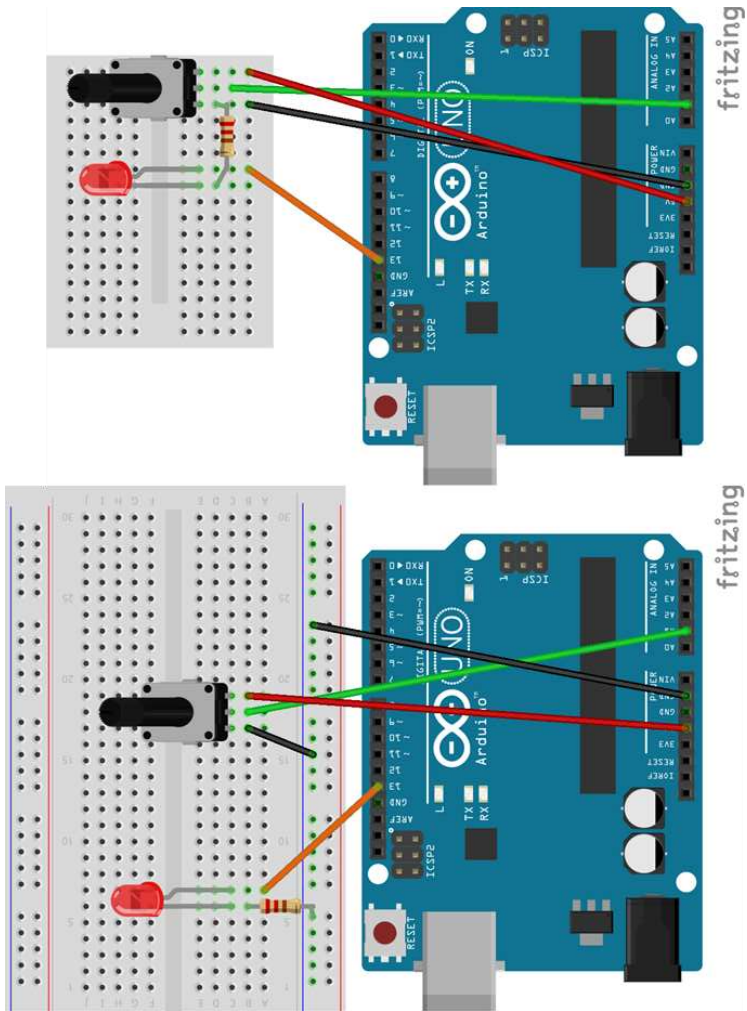


Figura 3.5: Possibilidades de Montagem do Projeto

Passo 2: Programa N° 1 – Controlando a frequência do pisca-pisca

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
// Liga e desliga um LED na frequência
// determinada pelo potenciômetro.

// Pino de entrada ligado ao potenciômetro
int POT = A1;

// Pino conectado ao LED
int LED = 13;

// Variável que armazenará o valor do
// obtido do potenciômetro
int valor = 0;

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

void loop() {
  valor = analogRead(POT);
  digitalWrite(LED, HIGH);
  delay(valor);
  digitalWrite(LED, LOW);
  delay(valor);
}
```

Passo 3: Montagem do circuito

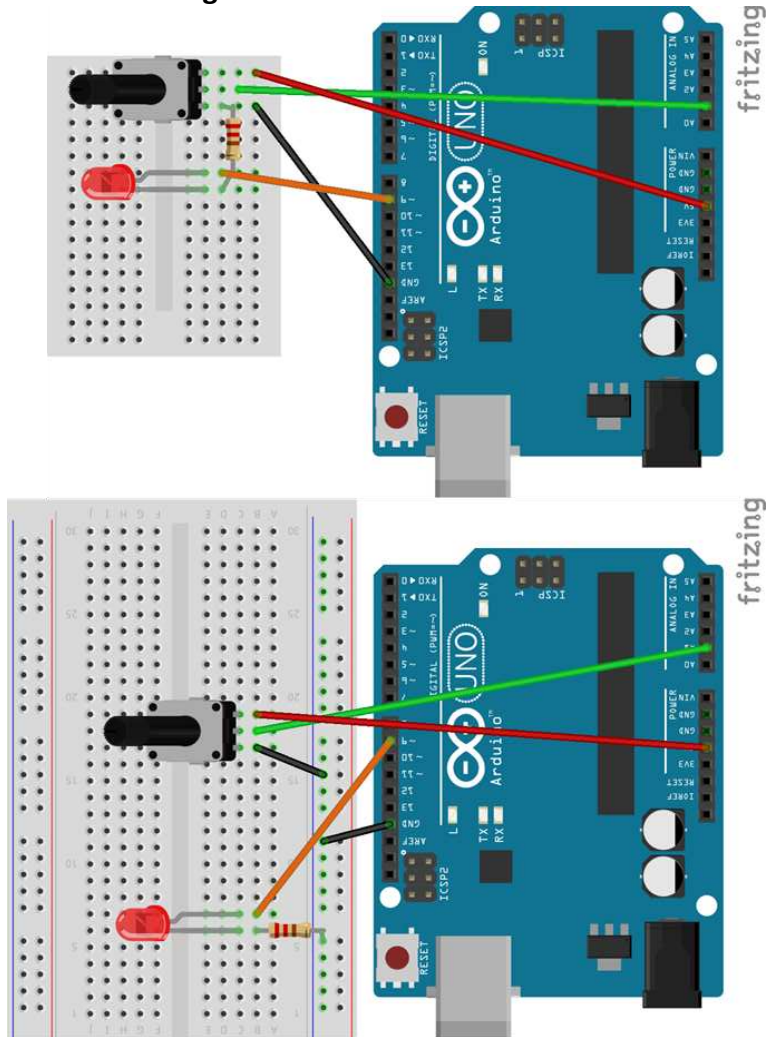


Figura 3.6: Opções de Montagem do Projeto

Conforme ilustra a Figura 3.6:

- a. Conecte o pino 5v do Arduino à linha de alimentação positiva (vermelha) do protoboard.
- b. Conecte o pino GND do Arduino à linha de alimentação negativa (preta) do protoboard.
- c. Conecte um LED utilizando um resistor de 220 ohms (ou 330 ohms).
- d. Conecte o LED no pino digital 9.
- e. Conecte o potenciômetro na protoboard com o botão de girar virado para você.
- f. Conecte o pino da esquerda do potenciômetro na linha de alimentação GND.
- g. Conecte o pino da direita do potenciômetro na linha de alimentação positiva.
- h. Conecte o pino do centro do potenciômetro no pino analógico A1 do Arduino.

Passo 4: Programa Nº 2 – Controle da intensidade da luminosidade

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
// Controla a intensidade da luminosidade de um
// LED através de frequência determinada pelo
// potenciômetro.

int POT = A1;
int LED = 9;
int valor = 0;

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



```
pinMode(LED, OUTPUT);  
}  
  
void loop() {  
  valor = analogRead(POT);  
  if(valor > 0) {  
    // Acende o led com intensidade proporcional  
    // ao valor obtido  
    analogWrite(LED, (valor/4));  
  
    // Exibe no Serial Monitor o valor obtido  
    // do potenciômetro  
    Serial.println(valor);  
  }  
}
```

Dicas - Como funciona a PWM?

A Modulação por Largura de Pulso (*Pulse Width Modulation* - **PWM**) é uma técnica que consiste em fornecer um sinal analógico através de meios digitais. A forma de onda do sinal digital consiste em uma onda quadrada que alterna seu estado em nível lógico alto e um nível lógico baixo (pode ser representado por ligado/desligado ou pelo sistema binário 1 e 0).

A razão entre o período de pico e o período total da onda é chamada de *Duty Cycle* (Figura 3.7). Podemos, então, entender que para termos uma onda quadrada real (que possui picos e vales iguais) é necessário que o *Duty Cycle* seja de 50%, ou seja, 50% de pico e 50% de vale.

Pulse Width Modulation

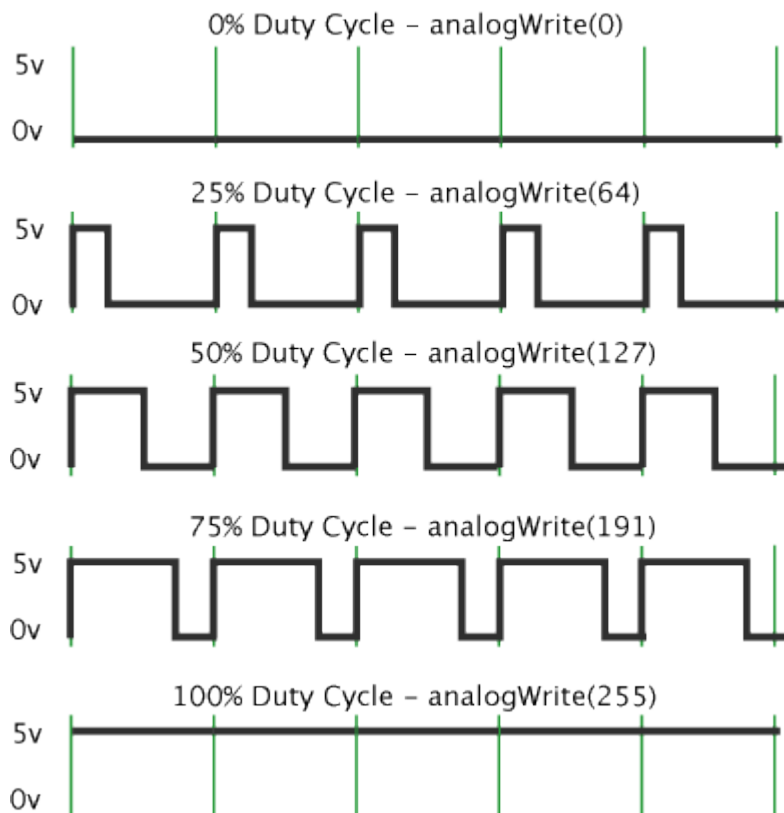


Figura 3.7: Razão de Ciclo

No Arduino UNO, as portas digitais que permitem PWM são as portas 3, 5, 6, 9, 10 e 11. Essas portas são facilmente identificadas pelo símbolo "~" abaixo de cada porta.

Fonte: Tradução e imagem de "PWM" disponível em <http://arduino.cc/en/Tutorial/PWM>.

Projeto 3 – LDR

O objetivo deste projeto é controlar o estado de um LED (aceso ou apagado) através da verificação de luminosidade do ambiente utilizando um sensor de luminosidade LDR. O LDR (*Light Dependent Resistor*) é um componente que varia a sua resistência conforme o nível de luminosidade que incide sobre ele. A resistência do LDR varia de forma inversamente proporcional à quantidade de luz incidente sobre ele. Quanto maior a luminosidade, menor será a resistência, por outro lado, no Arduino, **maior** será o valor presente na entrada analógica. Quanto menor for a luminosidade, maior será a resistência, ou seja, **menor** será o valor na entrada analógica do Arduino. Para essa experiência, vamos supor que o nível limite de luminosidade para que o LED se acenda como sendo um valor menor que 100 na entrada analógica. Para monitorar o valor gerado pelo LDR vamos estabelecer a comunicação serial entre o Arduino e o computador e, em seguida, utilizar o Serial Monitor da IDE do Arduino para monitorar.

Material necessário:

- 1 Arduino.
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom)².
- 1 LED (qualquer cor)².
- 1 LDR^{1 2}.
- 1 Resistor de 10k ohms (marrom, preto laranja) para o LDR^{1 2}.
- 1 Protoboard².
- Jumper cable.

¹ Podem ser substituídos pelo módulo P13-LDR da GBK Robotics.

² Podem ser substituídos pelo módulo P7-Sensor de Luminosidade da GBK Robotics.

Passo 1: Montagem do circuito

Conforme ilustra a Figura 3.8:

- a. Conecte o pino 5v do Arduino à linha de alimentação positiva (vermelha) da protoboard.
- b. Conecte o pino GND do Arduino à linha de alimentação negativa (preta) da protoboard.
- c. Coloque o resistor de 220 ohms (ou 330 ohms) entre a linha de alimentação negativa e qualquer outra linha da protoboard.
- d. Coloque o LED com o Cátodo (lado chanfrado) conectado ao resistor de 220 ohms (ou 330 ohms);
- e. Conecte o Ânodo do LED ao pino 13 do Arduino.
- f. Coloque o resistor de 10k ohms entre a linha de alimentação negativa e qualquer outra linha da protoboard.
- g. Conecte uma das extremidades do LDR na linha o resistor de 10k ohms.
- h. Conecte uma extremidade do jumper entre o LDR e o resistor. A outra extremidade conecte no pino analógico A0;
- i. Conecte a outra extremidade do LDR à linha de alimentação positiva (vermelha).

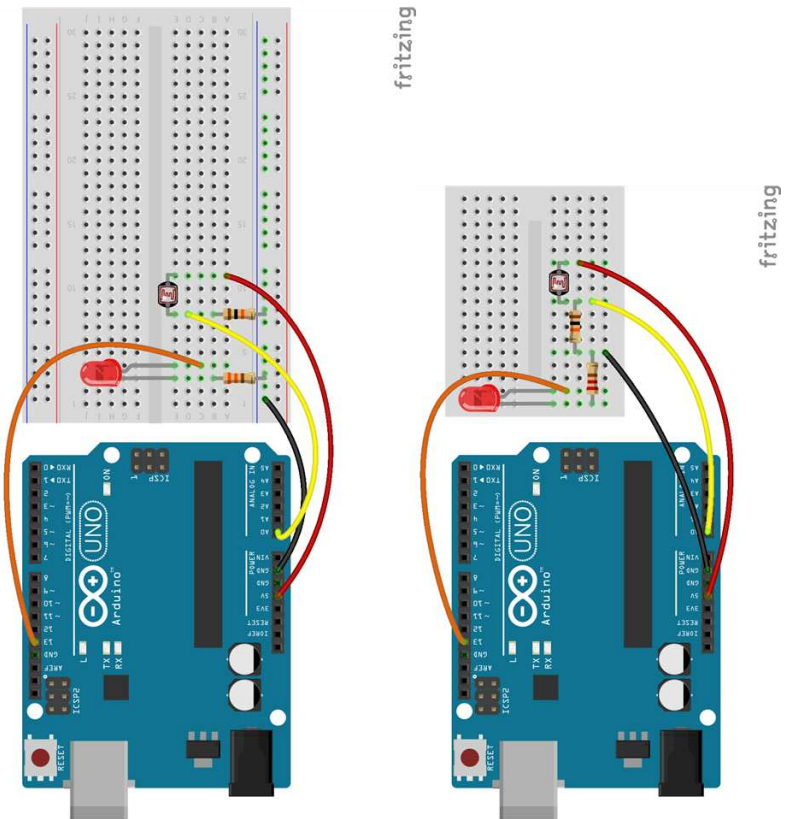


Figura 3.8: Disposição dos Componentes e Ligações

Variação de Montagem 1

Módulo P13-LDR da GBK Robotics

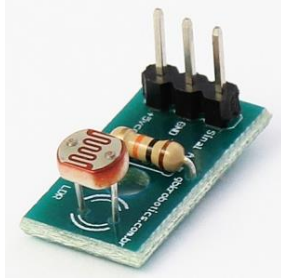


Figura 3.9: Módulo P13-LDR

Este projeto pode ser montado substituindo o LDR e o Resistor de 10k ohms pelo módulo P13-LDR (Figura 3.9) da GBK Robotics, neste caso:

- Conecte o pino GND do Arduino à linha de alimentação negativa (preta) da protoboard.
- Coloque o resistor de 220 ohms (ou 330 ohms) entre a linha de alimentação negativa e qualquer outra linha da protoboard.
- Coloque o LED com o Cátodo (lado chanfrado) conectado ao resistor de 220 ohms (ou 330 ohms).
- Conecte o Ânodo do LED ao pino 13 do Arduino.
- Conecte o pino 5v do Arduino ao pino +5Vcc do módulo P13.
- Conecte o pino GND do módulo P13 à linha de alimentação negativa da protoboard.
- Conecte o pino analógico A0 do Arduino ao pino Sinal A do módulo P13.

IMPORTANTE: Não há alterações no sketch (programa).

Variação de Montagem 2

Módulo P7-Sensor de Luminosidade da GBK Robotics



Figura 3.10: Módulo P7-Sensor de Luminosidade

Este projeto pode ser montado substituindo o LED, o LDR, os Resistores de 220 ohms (ou 330 ohms) e 10k ohms e a Protoboard pelo módulo P7-Sensor de Luminosidade (Figura 3.10) da GBK Robotics, neste caso:

- Conecte o pino 5v do Arduino ao pino 5Vcc do módulo P7.
- Conecte o pino GND do módulo P7 a um dos pinos de GND do Arduino.
- Conecte o pino analógico A0 do Arduino ao pino Sinal Analog. do módulo P7.
- Conecte o pino Led1 do módulo P7 ao pino digital 13 do Arduino.

IMPORTANTE: Os pinos Led2 e Led3 no módulo P7 não são ligados e não há alterações no sketch (programa).

Passo 2: Programa

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
// Pino no qual o LED está conectado
int LED = 13;

// Pino no qual o LDR está conectado
int LDR = A0;

// Variável que receberá o valor lido do LDR
int entrada = 0;

void setup() {
  // Iniciar e definir a velocidade de
  // comunicação da porta serial
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
}

void loop() {
  entrada = analogRead(LDR);

  // Valor que será exibido no Serial Monitor do
  // ambiente de desenvolvimento do Arduino
  Serial.println(entrada);

  if (entrada < 500)
    digitalWrite(LED, HIGH); // Acender o LED
  else
    digitalWrite(LED, LOW); // Apagar o LED
  delay(100);
}
```


Passo 3: Serial Monitor

Clique no botão mostrado em destaque na Figura 3.11 para abrir a janela do Monitor Serial.

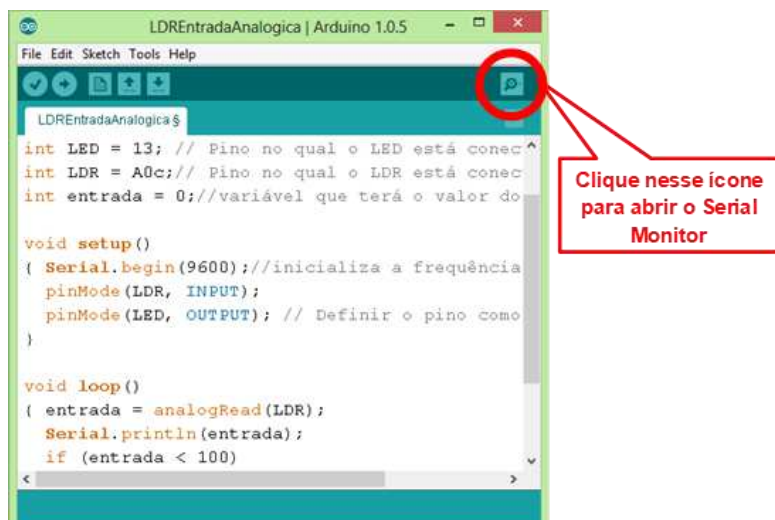


Figura 3.11: Botão para Abrir o Monitor Serial

Projeto 4 – Buzzer

O objetivo deste projeto é acionar um buzzer utilizando as funções `tone()` e `noTone()`. Buzzer é um dispositivo para geração de sinais sonoros (*beeps*), como aqueles encontrados em computadores. Para a emissão do som, o buzzer vibra através de um oscilador. Essa oscilação é determinada por uma frequência, que por sua vez define um som específico. Nesse experimento será usado 2 modos de uso da função `tone()` e uma pequena variação de sons representando notas musicais. A função `tone()` possui 2 sintaxes: `tone(pino, frequência)` e `tone(pino, frequência, duração)`, onde **pino** referencia qual é o pino que irá gerar a frequência (ligado ao positivo do buzzer), a **frequência** é definida em hertz e a **duração** (opcional) é em milissegundos. Caso opte pela sintaxe sem duração é necessário usar a função `noTone(pino)` para parar a frequência enviada pelo pino definido.

Material necessário:

- 1 Arduino.
- 1 Buzzer*.
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom) para o buzzer*.
- 1 Protoboard*.
- Jumper cable.

** Podem ser substituídos pelo módulo P15-Buzzer da GBK Robotics.*

Passo 1: Montagem do circuito

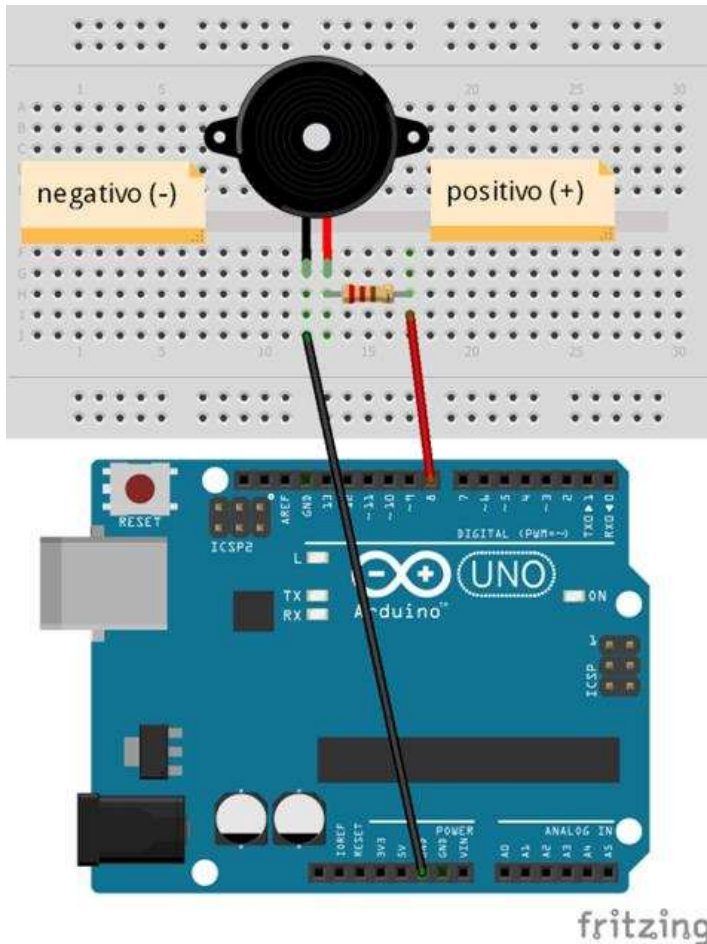


Figura 3.12: Ligação do Buzzer ao Arduino

Conforme ilustra a Figura 3.12:

- Coloque o buzzer na protoboard.
- Conecte o pino GND do Arduino ao pino negativo do buzzer.

- c. Coloque o resistor de 220 ohms (ou 330 ohms) ligado ao pino positivo do buzzer.
- d. Nesse resistor, conecte um jumper até a porta digital 8 do Arduino.

Variação de Montagem

Módulo P15-Buzzer da GBK Robotics



Figura 3.13: Módulo P15-Buzzer

Este projeto pode ser montado substituindo o buzzer, o resistor de 220 ohms (ou 330 ohms) e a protoboard pelo módulo P15-Buzzer (Figura 3.13) da GBK Robotics, neste caso:

- a. Conecte o pino GND do Arduino ao pino GND do módulo P15.
- b. Conecte o pino 8 do Arduino ao pino Sinal do módulo P15.

IMPORTANTE: Não há alterações no sketch (programa).

Passo 2: Programa N° 1 – Uso das funções *tone(pino, frequência)* e *noTone(pino)*

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
// Pino ao qual o buzzer está conectado
int buzzer = 8;

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

void loop() {
  tone(buzzer, 1200); //Define pino e frequência
  delay(500);
  noTone(buzzer); //Desliga o buzzer
  delay(500);
}
```

Passo 3: Programa N° 2 – Uso da função *tone(pino, frequência, duração)*

No ambiente de desenvolvimento do Arduino digite o sketch (programa) a seguir:

```
// Pino ao qual o buzzer está conectado
int buzzer = 8;

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

void loop() {
```

```
// Define pino, a frequência e duração
tone(buzzer, 1200, 500);
delay(1000);
}
```

Passo 4: Programa N° 3 – Notas musicais com buzzer

No ambiente de desenvolvimento do Arduino digite o sketch (programa) a seguir:

```
// Pino ao qual o buzzer está conectado
int buzzer = 8;

int numNotas = 10;
// Vetor contendo a frequência de cada nota
int notas[] = {261, 277, 294, 311, 330, 349, 370, 392, 415, 440};
// Notas: C, C#, D, D#, E, F, F#, G, G#, A
/*
C=Dó D=Ré E=Mí F=Fá G=Sol A=Lá B=Si
As notas sem o “#”, são as notas naturais
(fundamentais).
Aquelas com o “#”, são chamadas “notas
sustenidas” (por exemplo: C#=Dó Sustenido).
*/
void setup() {
  pinMode(buzzer, OUTPUT);
}

void loop() {
  for (int i = 0; i < numNotas; i++) {
    tone(buzzer, notas[i]);
    delay(500);
  }
  noTone(buzzer);
}
```

Projeto 5 – Botão

O objetivo deste projeto é utilizar um botão para acender, apagar e, posteriormente, também controlar a luminosidade de um LED.

Material necessário:

- 1 Arduino.
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom) para o LED.
- 1 Resistor de 10k ohms (marrom, preto laranja) para o botão.
- 1 LED (qualquer cor) .
- 1 Protoboard.
- Jumper cable.

Passo 1: Montagem do circuito

Conforme o tipo de protoboard usado realize a montagem adotando, como referência, a Figura 3.14.

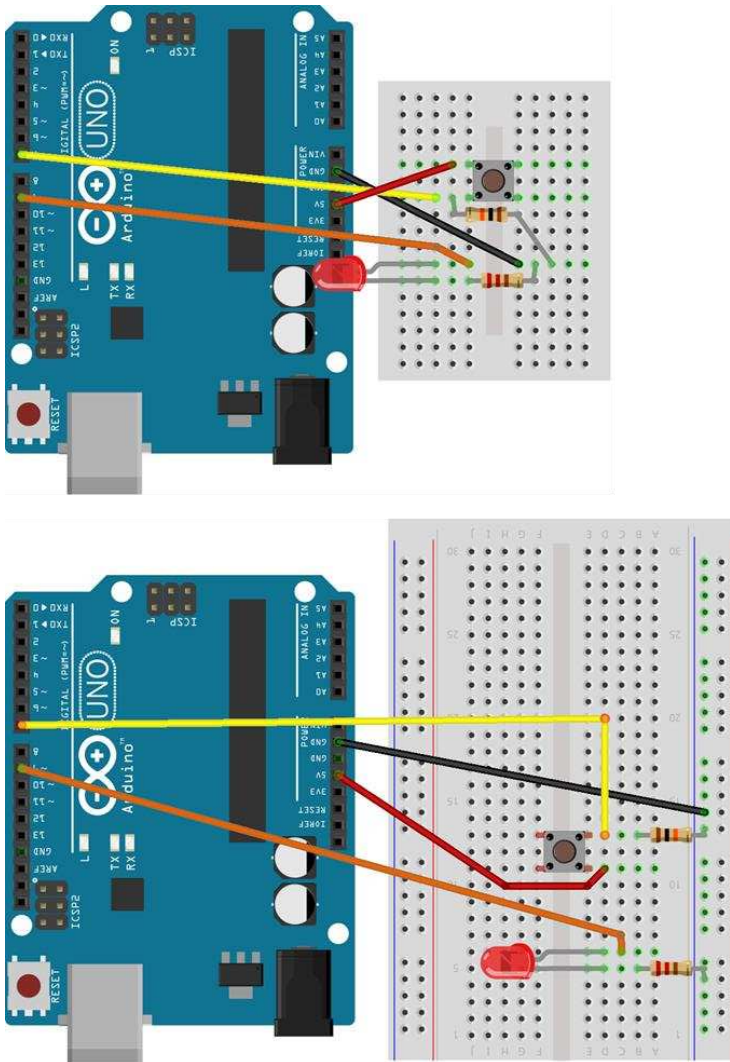


Figura 3.14: Conexões Necessárias

Passo 2: Programa-Ligando e desligando um LED através do botão

Neste programa, enquanto o botão estiver pressionado o LED ficará acesso, caso contrário, o LED fica apagado. Para implementá-lo, acesse o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
// Pino ao qual o LED está conectado
int LED = 9;
// Pino ao qual o Botão está conectado
int BOTAO = 7;
int valor;

void setup() {
  // Definir o pino como saída
  pinMode(LED, OUTPUT);
  // Definir o pino com entrada
  pinMode(BOTAO, INPUT);
}

void loop() {
  // Obtém LOW (botão não pressionado) ou
  // HIGH (botão pressionado)
  valor = digitalRead(BOTAO);

  digitalWrite(LED, valor);
  delay (500);
}
```

Passo 3: Programa-Ligando e desligando um LED através do botão (com troca de estado)

Neste outro programa ao pressionar e soltar o botão o LED acenderá, a pressionar e soltar o botão novamente o LED vai apagar e assim sucessivamente. Desenvolva o programa acessando o ambiente de desenvolvimento do Arduino e digitando o seguinte sketch.

```
// Pino ao qual o LED está conectado
int LED = 9;
// Pino ao qual o Botão está conectado
int BOTAO = 7;
int valor;
int anterior = 0;
int estado = LOW;
void setup() {
  pinMode(LED, OUTPUT);
  pinMode(BOTAO, INPUT);
}

void loop() {
  valor = digitalRead(BOTAO);
  if (valor == HIGH && anterior == LOW) {
    estado = !estado;
  }
  digitalWrite(LED, estado);
  anterior = valor;
  delay (50);
}
```

Passo 4: Programa-Ligando, desligando e alterando a intensidade luminosa de um LED

Nesta implementação, através de um pino capaz de utilizar valores analógicos (PWM), poderemos ligar, desligar e obter uma variação de luminosidade. O LED começa com seu estado “apagado”. Com um pressionar no botão, altera-se o estado do LED para “aceso”. Caso permaneça com o botão pressionado por mais de 5 segundos, poderá ser identificada uma variação de luminosidade. Crie o sketch a seguir no ambiente de desenvolvimento do Arduino.

```
// Pino ao qual o LED está conectado
int LED = 9;
// Pino ao qual o Botão está conectado
int BOTAO = 7;
int valor = LOW;
int valorAnterior = LOW;
// 0 = LED apagado, 1 = LED aceso
int estado = 0;
int brilho = 128;
unsigned long inicio;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(BOTAO, INPUT);
}

void loop() {
  valor = digitalRead(BOTAO);
  if ((valor == HIGH) && (valorAnterior==LOW)) {
    estado = 1 - estado;
    // Obtém a quantidade de milisegundos após
    // o Arduino ser inicializado
    inicio = millis();
```

```

    delay (10);
}

// Verifica se o botão está pressionado
if ((valor==HIGH) && (valorAnterior==HIGH)) {
    // Verifica se o botão está pressionado por
    // mais de 0,5 segundos
    if (estado==1 && (millis()-inicio) > 500) {
        brilho++;
        delay(10);
        if (brilho > 255)
            brilho = 0;
    }
}
valorAnterior = valor;

if (estado == 1) {
    // Define o nível de luminosidade do LED
    analogWrite(LED, brilho);
}
else {
    analogWrite(LED, 0); // Apaga o LED
}
}

```

Desafio 2 – Controle de Estufa

Objetivo:

O objetivo deste projeto é controlar a luminosidade de uma estufa através de sensores e controles de abertura e fechamento de telas que controlam a entrada de luz solar em um ambiente simulado. Essa estufa foi projetada para plantas ornamentais, que precisam de baixa luminosidade para seu crescimento ideal. Antes da construção real da estufa é necessário criar uma base de teste. Para essa base de testes são definidos os seguintes parâmetros:

- Um LDR será utilizado para verificar a luminosidade do local. A partir dele deverá ser avaliada a luminosidade mais alta possível do local (o valor mais alto alcançado). Esse valor tem que ser verificado através do Serial Monitor durante algum tempo e será usado como referência (valor máximo de luminosidade do local). Por exemplo, se o maior valor captado pelo LDR for 835, esse será o valor máximo (referente a 100% de luminosidade).
- Três LEDs irão representar o status de luminosidade do local: o LED vermelho será acesso quando a luminosidade do local estiver entre 80% a 100% da luminosidade máxima do local; o LED amarelo será acesso quando a luminosidade estiver entre 50% a 79% da luminosidade máxima do local; e o LED verde indica uma luminosidade ideal, abaixo dos 50%.
- Um buzzer irá representar o controle das telas. Quando o LED estiver vermelho, as 3 telas de

proteção deverão ser fechadas, para impedir a entrada de luz solar e assim reduzindo a luminosidade. Com isso, o buzzer deve apitar 3 sinais breves, sendo cada um dos sinais indicativo que uma tela está fechada. Se o LED estiver amarelo, serão emitidos 2 apitos breves (2 telas fechadas). Quando o LED estiver verde, 1 apito será ouvido (1 tela fechada).

- O sistema também terá um botão de liga/desliga. Durante o dia (período que o sistema deverá estar ligado) é necessário clicar uma vez no botão (*push button*). Pressionar novamente o botão indica o desligamento do sistema, ou seja, o sistema não estará mais ativo.

Projeto 6 – LCD (*Liquid Crystal Display* 16x2)

O objetivo deste projeto é aprender a montagem de um display LCD 16x2 controlado pelo Arduino utilizando a biblioteca ***LiquidCrystal.h***. Essa biblioteca possui funções que auxiliam nas configurações e tratamento dos dados a serem enviados ao LCD. A montagem do display deve ser de acordo com sua especificação (*datasheet*), onde cada um dos pinos possui uma função específica (ver no passo 1 – Montagem do circuito). Para ver todas as funções disponíveis na biblioteca ***LiquidCrystal.h*** acesse o site oficial da biblioteca, que está disponível em <http://arduino.cc/en/Reference/LiquidCrystal>.

Material necessário:

- 1 Arduino.
- 1 LCD 16x2.
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom).
- 1 Protoboard.
- Jumper cable.



64

Conforme ilustra a Figura 3.15, realizar a seguinte sequência de montagem:

- Pino 1 do LCD ligado ao GND do Arduino.
- Pino 2 do LCD ligado ao 5V do Arduino.
- Pino 3 do LCD ligado ao pino central do potenciômetro (controle de contraste).
- Pino 4 do LCD ligado ao pino digital 12 do Arduino.
- Pino 5 do LCD ligado ao GND do Arduino.
- Pino 6 do LCD ligado ao pino digital 11 do Arduino.
- Pino 11 do LCD ligado ao pino digital 5 do Arduino.
- Pino 12 do LCD ligado ao pino digital 4 do Arduino.
- Pino 13 do LCD ligado ao pino digital 3 do Arduino.
- Pino 14 do LCD ligado ao pino digital 2 do Arduino.
- Pino 15 do LCD ligado ao 5v do Arduino com um resistor de 220 ohms (controle do brilho).
- Pino 16 do LCD ligado ao GND do Arduino.

Na tabela a seguir apresentamos as funções de cada um dos pinos do Display de Cristal Líquido:

Pino	Símbolo	Função
1	VSS	GND(Alimentação)
2	VDD	5V(Alimentação)
3	VO	Ajuste de Contraste
4	RS	Habilita/Desabilita Seletor de Registrador
5	R/W	Leitura/Escrita
6	E	Habilita Escrita no LCD
7	DB0	Dado
8	DB1	Dado
9	DB2	Dado
10	DB3	Dado
11	DB4	Dado
12	DB5	Dado
13	DB6	Dado
14	DB7	Dado
15	A	5V(Backlight)
16	K	GND(BackLight)

Fonte: labdegaragem.com

Passo 2: Programa N° 1 – Exibição simples de texto

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
#include <LiquidCrystal.h>

// LCD - Modelo ACM 1602K
// Pin Sigla Função          Conectar
// 1 Vss   Ground           GND
// 2 Vdd   +5V              VCC
// 3 Vo    LCD contrast adjust Potenciômetro
// 4 RS    Register select   Arduino 12
```

```
// 5  R/W  Read/write      GND
// 6  E    Enable         Arduino 11
// 7  DB0  Data bit 0     NC
// 8  DB1  Data bit 1     NC
// 9  DB2  Data bit 2     NC
// 10 DB3  Data bit 3     NC
// 11 DB4  Data bit 4     Arduino 5
// 12 DB5  Data bit 5     Arduino 4
// 13 DB6  Data bit 6     Arduino 3
// 14 DB7  Data bit 7     Arduino 2
// +    BL+  Alimentação (+)  Resistor de 1k
//                               para VCC
// -    BL-  Alimentação (-)  GND

#define TEMPO_ATUALIZACAO 500

LiquidCrystal lcd (12, 11, 5, 4, 3, 2);

void setup() {
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);

  // Inicia o LCD com dimensões 16 x 2 (Colunas
  // x Linhas)
  lcd.begin (16, 2);
}

void loop() {
  lcd.clear();
  lcd.setCursor(0, 0); // Linha 0 e Coluna 0
  lcd.print("Ola");
  lcd.setCursor(0, 1); // Linha 1 e Coluna 0
  lcd.print("FATECINO");
  delay(TEMPO_ATUALIZACAO);
}
```

Passo 3: Programa N° 2 – Rolagem (scroll) do texto

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
#include <LiquidCrystal.h>

// LCD - Modelo ACM 1602K
// Pin Sigla Função          Conectar
// 1  Vss   Ground          GND
// 2  Vdd   +5V             VCC
// 3  Vo    LCD contrast adjust Potenciômetro
// 4  RS    Register select  Arduino 12
// 5  R/W   Read/write       GND
// 6  E     Enable           Arduino 11
// 7  DB0   Data bit 0       NC
// 8  DB1   Data bit 1       NC
// 9  DB2   Data bit 2       NC
// 10 DB3   Data bit 3       NC
// 11 DB4   Data bit 4       Arduino 5
// 12 DB5   Data bit 5       Arduino 4
// 13 DB6   Data bit 6       Arduino 3
// 14 DB7   Data bit 7       Arduino 2
// +  BL+   Alimentação (+)  Resistor de 1k
//                               para VCC
// -  BL-   Alimentação (-)  GND

#define TEMPO_ATUALIZACAO 500

LiquidCrystal lcd (12, 11, 5, 4, 3, 2);
int inicio = 0, tamanho = 1;
boolean alterar = false;

void setup() {
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}
```

```
// Inicia o LCD com dimensões 16 x 2 (Colunas
// x Linhas)
lcd.begin (16, 2);
}

void loop() {
  lcd.clear(); // Limpa o display de LCD
  String nome = "Fatecino-Clube de Arduino";
  if (tamanho < 16) {
    // Posiciona o cursor nas coordenadas
    // especificadas
    lcd.setCursor(16 - tamanho, 0);

    // Exibe no LCD
    lcd.print(nome.substring(inicio, tamanho));
    tamanho++;
  }
  else {
    if (!alterar) {
      alterar = !alterar;
      tamanho = 16;
      lcd.setCursor(0, 0);
    }
    lcd.print(nome.substring(inicio, inicio +
tamanho));
    inicio++;
  }
  if (inicio > nome.length()) {
    inicio = 0;
    tamanho = 1;
    alterar = !alterar;
  }
  delay(TEMPO_ATUALIZACAO);
}
```

Desafio 3 – Controle de Estufa com LCD

O objetivo deste projeto é controlar a luminosidade de uma estufa através de sensores e controles de abertura e fechamento de telas que controlam a entrada de luz solar em um ambiente simulado. Essa estufa foi projetada para plantas ornamentais, que precisam de baixa luminosidade para seu crescimento ideal. Antes da construção real da estufa é necessário criar uma base de teste. Para essa base de testes são definidos os seguintes parâmetros:

- Um LDR será utilizado para verificar a luminosidade do local. A partir dele deverá ser avaliada a luminosidade mais alta possível do local (o valor mais alto alcançado). Esse valor tem que ser verificado através do Serial Monitor durante algum tempo e será usado como referência (valor máximo de luminosidade do local). Por exemplo, se o maior valor captado pelo LDR for 835, esse será o valor máximo (referente a 100% de luminosidade).
- Três LEDs irão representar o status de luminosidade do local: o LED vermelho será acesso quando a luminosidade do local estiver entre 80% a 100% da luminosidade máxima do local; o LED amarelo será acesso quando a luminosidade estiver entre 50% a 79% da luminosidade máxima do local; e o LED verde indica uma luminosidade ideal, abaixo dos 50%.
- Será utilizado um display LCD 16x2 para mostrar todos os estados do sistema. Quando o LED vermelho estiver acesso no display mostrará “Luminosidade Alta. Valor do LDR: XXX”. Caso o LED

amarelo esteja acesso, o display terá que mostrar “Luminosidade Média. Valor do LDR: XXX”. E quando o LED verde estiver acesso, mostrará “Luminosidade Baixa. Valor do LDR: XXX”. **OBS.:** No lugar de “XXX” deverá aparecer o valor atual do LDR.

- Um buzzer irá representar o controle das telas. Quando o LED estiver vermelho, as 3 telas de proteção deverão ser fechadas, para impedir a entrada de luz solar e assim reduzindo a luminosidade e o display mostrará “3 telas fechadas”. Com isso, o buzzer deve apitar 3 sinais breves, sendo cada um dos sinais indicativo que uma tela está fechada. Se o LED estiver amarelo, serão emitidos 2 apitos breves (2 telas fechadas) e o display mostrará “2 telas fechadas”. Quando o LED estiver verde, 1 apito será ouvido (1 tela fechada) e o display mostrará “1 tela fechada”.
- O sistema também terá um botão de liga/desliga. Durante o dia (período que o sistema deverá estar ligado) é necessário clicar uma vez no botão (*push button*). Pressionar novamente o botão indica o desligamento do sistema, ou seja, o sistema não estará mais ativo.

Projeto 7 – Uso do Sensor de Temperatura

O objetivo deste projeto é enviar os dados de um sensor de temperatura (LM35 ou DHT11) para a saída serial.

Material necessário:

- 1 Arduino.
- 1 LM35 (ou DHT11).
- 1 Protoboard.
- Jumper cable.

Passo 1: Montagem do circuito (com o LM35)

Realize as conexões conforme ilustra a Figura 3.16.

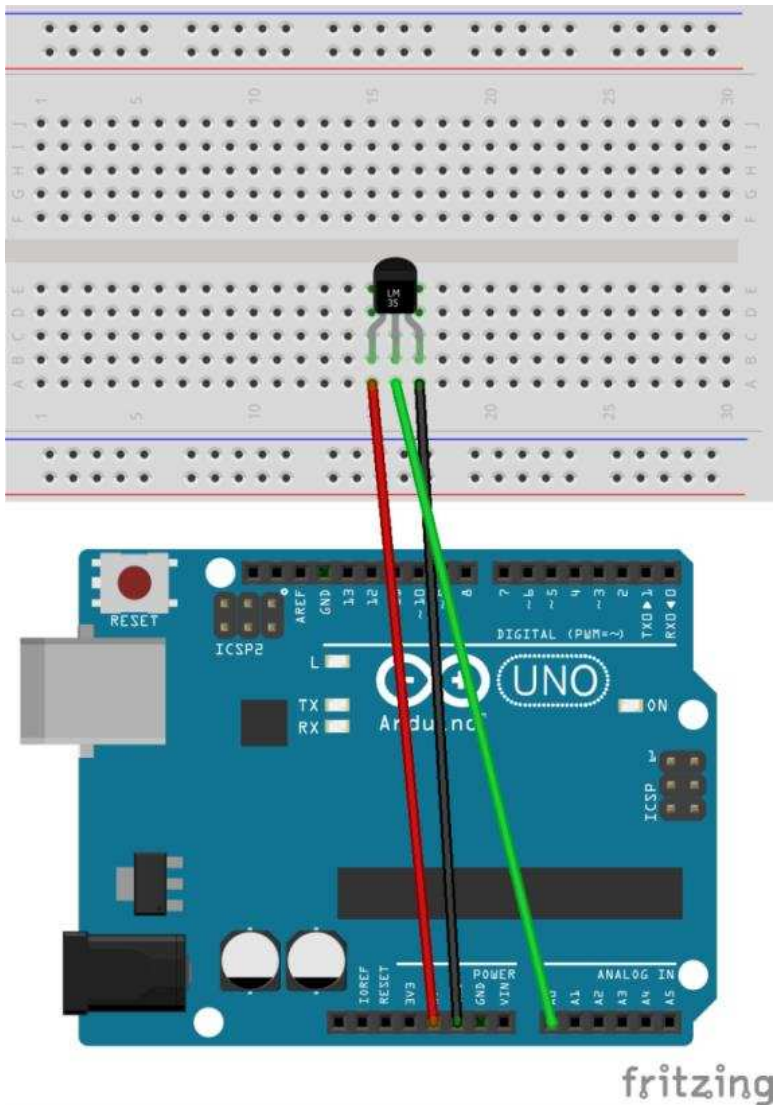


Figura 3.16: Ligação do Arduino ao LM35

Passo 2: Programa para o LM35

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
// Pino Analógico que vai ser ligado ao
// terminal 2 do LM35
const int LM35 = A0;

// Tempo de atualização em ms (milissegundos)
const int ATRASO = 5000;

// Base de conversão para Graus Celsius:
// ((5/1023) * 100)
const float BASE_CELSIUS =
    0.4887585532746823069403714565;

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

void loop() {
    Serial.print("Temperatura: ");
    Serial.print(lerTemperatura());
    Serial.println("\260C");
    delay(ATRASO);
}

float lerTemperatura() {
    return (analogRead(LM35) * BASE_CELSIUS);
}
```

Passo 3: Montagem do circuito (com o DH11)

Monte o projeto conforme mostra a Figura 3.17.

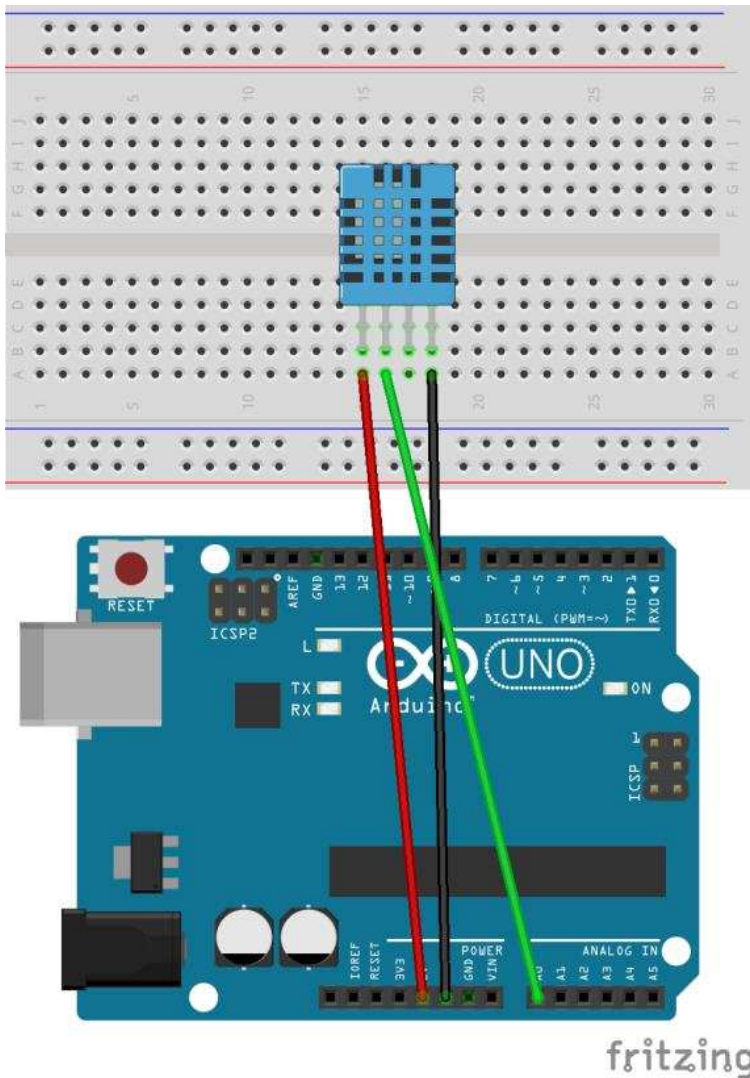


Figura 3.17: Conexão do Arduino ao DHT11

Passo 4: Programa para o DHT11

Baixe a biblioteca a partir do link: <http://hobbyist.co.nz/sites/default/files/WeatherStation/DHT.zip>. Descompacte o conteúdo do arquivo DHT.zip na pasta Arduino\libraries\ localizada na pasta Documentos. Em seguida, inicie o ambiente de desenvolvimento do Arduino e digite o sketch.

```
#include <dht.h>

// Pino analógico ligado ao terminal 2 do DHT11
const int DHT11 = A0;

//Tempo de atualização entre as leituras em ms
const int ATRASO = 2000;

dht sensor;

float temperatura, umidade;

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

void loop() {
  // Obtém os dados do sensor
  sensor.read11(DHT11);

  // Obtém a temperatura
  temperatura = sensor.temperature;

  // Obtém a umidade
  umidade = sensor.humidity;

  // Enviar a temperatura e a umidade através
```

```
// da saída serial
Serial.print("Temperatura: ");
Serial.print(temperatura);
Serial.print(';');
Serial.print("\260C, Umidade: ");
Serial.print(umidade);
Serial.print(';');
Serial.println("%");
delay(ATRASO);
}
```

Passo 5: Conversão para Fahrenheit e Kelvin

Utilizando as fórmulas a seguir, alterar o programa para exibir a temperatura em Fahrenheit e Kelvin.

- $F = (C * 9) / 5 + 32$
- $K = C + 273.15$

Passo 6: Exibição da temperatura em um display de LCD

Utilizando como base o Projeto-6 (Uso do LCD), enviar os dados de temperatura para um display de LCD.

Dica: Como imprimir o símbolo de graus no display de LCD? **Tente usar `lcd.write(B11011111)`;**

Projeto 8 – Termistor

O objetivo deste projeto é obter a temperatura ambiente através da leitura dos dados recebidos de um termistor. Há dois tipos de termistores:

- Termistor PTC (Positive Temperature Coefficient): Este tipo de termistor tem o coeficiente de temperatura positivo, ou seja, a resistência aumenta com o aumento da temperatura.
- Termistor NTC (Negative Temperature Coefficient): Já este é o inverso do anterior e seu coeficiente de temperatura é negativo. Com isto sua resistência diminui com o aumento da temperatura.

Material necessário:

- 1 Arduino.
- 1 Termistor NTC de 10k ohms*.
- 1 Resistor de 10k ohms (marrom, preto, laranja) para o termistor*.
- 1 Protoboard*.
- Jumper cable.

** Podem ser substituídos pelo módulo P10-Sensor de Temperatura com NTC da GBK Robotics.*

Passo 1: Montagem do circuito

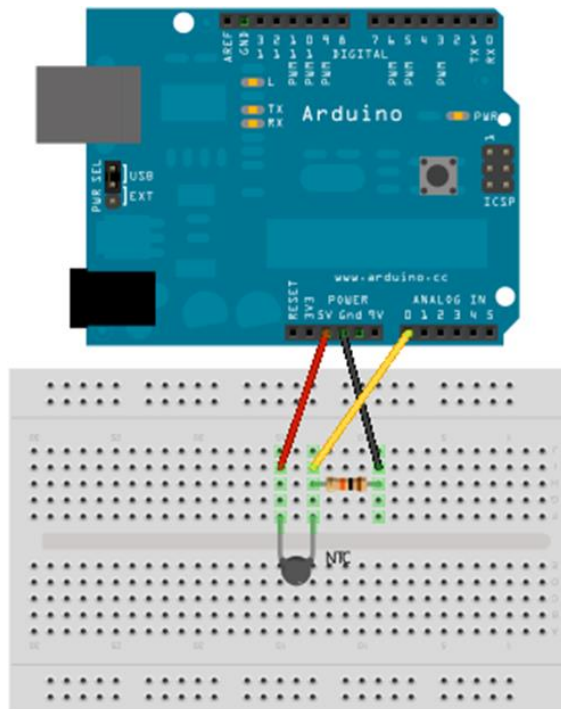


Figura 3.18: Ligação do Termistor ao Arduino

Conforme ilustra a Figura 3.18:

- Coloque o termistor na protoboard.
- Conecte o pino 5V do Arduino a um dos terminais do termistor.
- Conecte o resistor de 10k ohms ao outro pino do termistor.
- Conecte o pino GND do Arduino ao outro pino do resistor de 10k ohms.
- Ligue o pino A0 do Arduino junto com o resistor de 10k ohms e o terminal do termistor.

Variação de Montagem

Módulo P10-Sensor de Temperatura com NTC da GBK Robotics

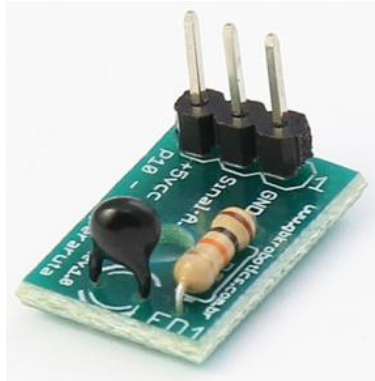


Figura 3.19: Módulo P10-Sensor de Temperatura

Este projeto pode ser montado substituindo o termistor, o resistor de 10k ohms e a protoboard pelo módulo P10-Sensor de Temperatura com NTC (Figura 3.19) da GBK Robotics, neste caso:

- Conecte o pino GND do Arduino ao pino GND do módulo P10.
- Conecte o pino 5V do Arduino ao pino +5Vcc do módulo P10.
- Conecte o pino A0 do Arduino ao pino Sinal-A do módulo P10.

IMPORTANTE: Não há alterações no Sketch (programa).

Passo 2: Programa

A biblioteca Thermistor pode ser baixada através do link: http://www.fatecjd.edu.br/fatecino/arg_projetos/biblioteca-thermistor.zip. Após sua instalação, inicie o ambiente de desenvolvimento do Arduino e digite o sketch mostrado a seguir.

```
#include <Thermistor.h>

Thermistor termistor(A0);

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

void loop() {
  int temperatura = termistor.getTemp();
  Serial.print("A temperatura e: ");
  Serial.print(temperatura);
  Serial.println("°C");
  delay(1000);
}
```

Passo 3: Conversão para Fahrenheit e Kelvin

Utilizando as fórmulas a seguir, alterar o programa para exibir a temperatura em Fahrenheit e Kelvin.

- $F = (C * 9) / 5 + 32$
- $K = C + 273.15$

Passo 4: Exibição da temperatura em um display de LCD

Utilizando como base o Projeto-6 (Uso do LCD), enviar os dados de temperatura para um display de LCD.

Dica: Como imprimir o símbolo de graus no display de LCD? **Tente usar `lcd.write(B11011111);`**

Desafio 4 – Termômetro Digital Completo

A proposta deste projeto é unir os conceitos já abordados sobre o uso de **sensores de temperatura** (Projetos 7 e 8), **display de LCD** (Projeto 6) e **botão** (Projeto 5) para montar um termômetro digital que permita escolher a escala de temperatura a ser exibida (Celsius, Fahrenheit ou Kelvin). Quando o sensor DHT-11 for utilizado no projeto também poderá ser exibida a umidade relativa do ar.

Na Figura 3.20 é mostrada uma sugestão de montagem utilizando o sensor DHT-11, podendo ser adaptada para uso do LM-35 ou também para termistores.

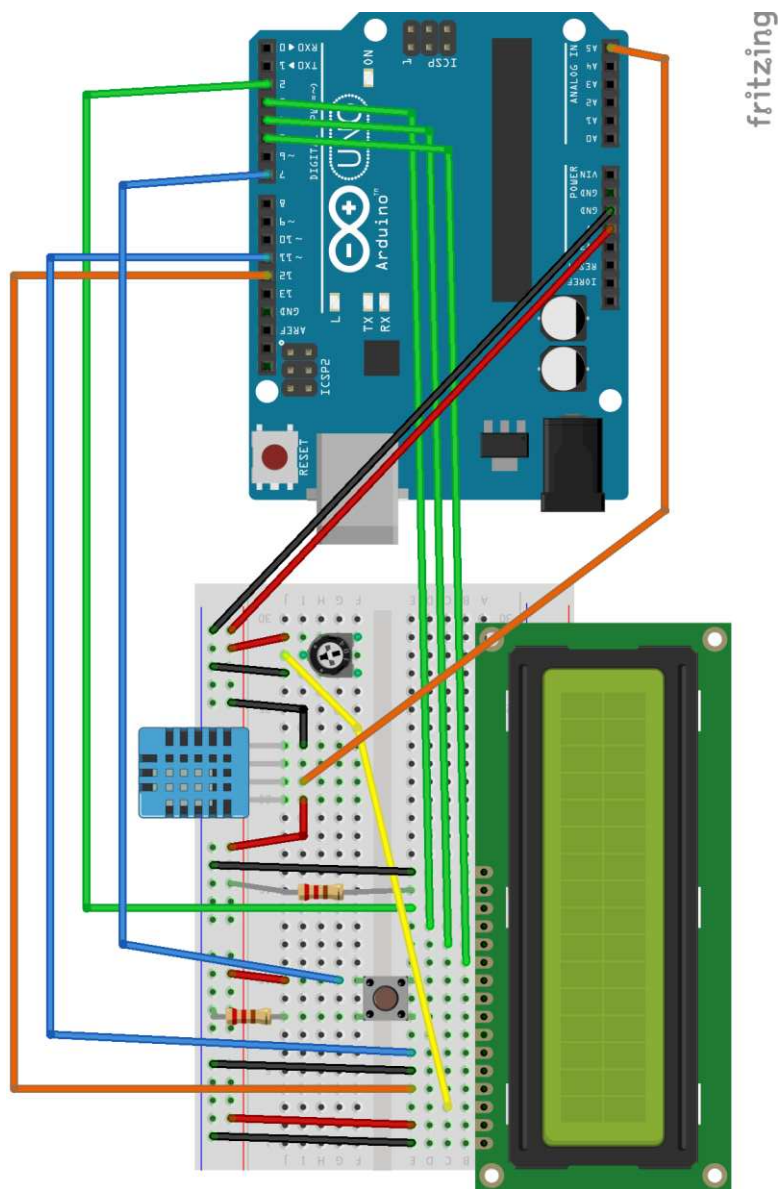


Figura 3.20: Ligações do Termômetro Digital

Projeto 9 – Sensor Ultrassônico (HC-SR04)

O objetivo deste projeto é utilizar o sensor ultrassônico HC-SR04 para medir distâncias entre o sensor e um objeto. O sensor HC-SR04 permite detectar objetos que lhe estão distantes entre 1 cm e 200 cm. Este sensor emite um sinal ultrassônico que reflete em um objeto e retorna ao sensor, permitindo deduzir a distância do objeto ao sensor tomando o tempo da trajetória do sinal. A velocidade do sinal no ar é de aproximadamente 340 m/s (velocidade do som). O sensor possui 4 pinos, sendo:

- VCC - Alimentação de 5V (+).
- TRIG - Pino de gatilho (*trigger*) .
- ECHO - Pino de eco (*echo*) .
- GND – Terra (-).

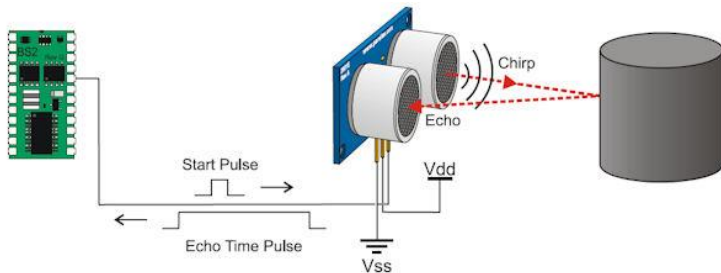


Figura 3.21: Funcionamento do Sensor Ultrassônico

O pino ligado ao *trigger* (TRIG) normalmente deve estar em nível baixo. Para iniciar uma leitura de distância, o mesmo deve ser colocado em nível alto por 10 microsegundos e retornar para nível baixo em seguida. Neste momento, 8 pulsos de 40kHz são emitidos e no pino de eco (ECHO) será gerado um sinal em nível alto proporcional à distância do sensor ao objeto. Em seguida,

basta verificar o tempo em que o pino ECHO permaneceu em nível alto e utilizar a fórmula de cálculo de distância (em centímetros): **distância = duração/58**.

Material necessário:

- 1 Arduino.
- 1 sensor HC-SR04.
- 1 Protoboard.
- Jumper cable.

Passo 1: Montagem do circuito

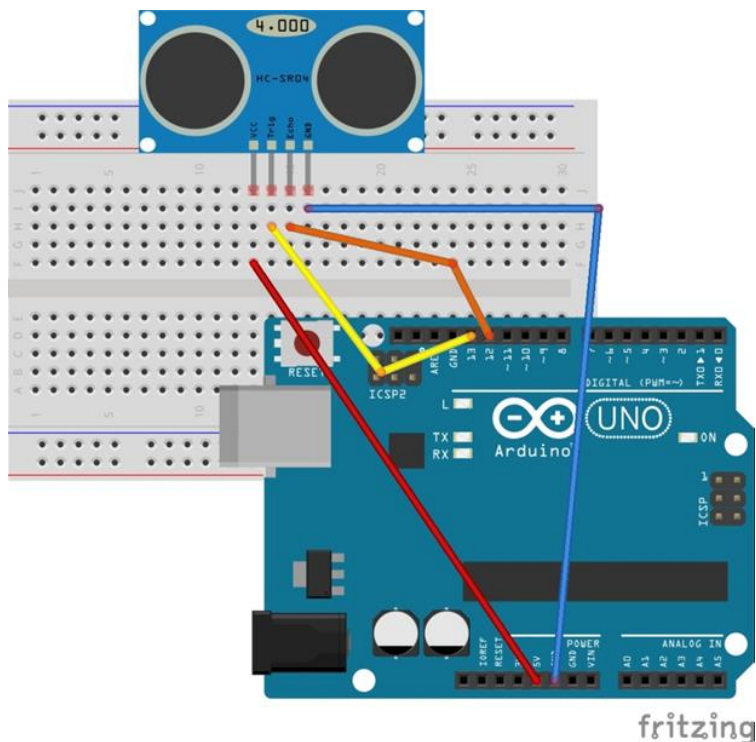


Figura 3.22: Conexão do HC-SR04 ao Arduino

Conforme ilustra a Figura 3.22, realize as seguintes ligações:

- Pino GND do HC-SR04 ligado ao GND do Arduino.
- Pino VCC do HC-SR04 ligado ao 5V do Arduino.
- Pino TRIG do HC-SR04 ligado ao pino 13 do Arduino.
- Pino ECHO do HC-SR04 ligado ao pino 12 do Arduino.

Passo 2: Programa

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
// Pino 12 irá receber o pulso do echo
int echoPino = 12;

// Pino 13 vai enviar o pulso para gerar o echo
int trigPino = 13;

long duracao = 0;
long distancia = 0;

void setup() {
  // Iniciar a porta serial com velocidade de
  // 9600 bits por segundo
  Serial.begin(9600);
  pinMode(echoPino, INPUT);
  pinMode(trigPino, OUTPUT);
}

void loop() {
  // Pino trigger com um pulso baixo LOW
  // (desligado)
  digitalWrite(trigPino, LOW);
```

```
// Delay (atraso) de 10 microssegundos
delayMicroseconds(10);

// Pino trigger com pulso HIGH (ligado)
digitalWrite(trigPino, HIGH);

// Delay (atraso) de 10 microssegundos
delayMicroseconds(10);

// Pino trigger com um pulso baixo LOW
// (desligado) novamente
digitalWrite(trigPino, LOW);

// A função pulseIn verifica o tempo que o
// pino ECHO ficou HIGH
// Calculando, desta forma, a duração do
// tráfego do sinal
duracao = pulseIn(echoPino,HIGH);

// Cálculo: distância = duração / 58.
distancia = duracao / 58;

Serial.print("Distancia em cm: ");
Serial.println(distancia);
delay(100);
}
```

Passo 3: Exibição da distância em um display de LCD

Utilizando como base o Projeto-6 (Uso do LCD), enviar os dados da distância para um display de LCD, tanto em centímetros quanto em polegadas. Para o cálculo de polegadas utilize: **distância = duração/37**.

Desafio 5 – Sensor de Estacionamento

O objetivo deste projeto é criar um sistema de estacionamento para carros (conhecidos também como *parking sensor system*). Esse sistema auxilia o motorista no momento de estacionar seu carro em vagas que exijam manobras em marcha ré. Na traseira do carro existem sensores (em nosso projeto será APENAS UM SENSOR) que medem a distância de objetos e/ou carros estacionados, mostrando ao motorista a distância em centímetro, além de sinais luminosos e sonoros. O projeto deverá ter as seguintes especificações:

- Um sensor ultrassônico HC-SR04 será usado para medir a distância entre o carro e o obstáculo. Para uma visualização dessa distância, terá que ser feita uma medição em centímetros, pois terá o valor sendo exibido em um display e utilizado para conferência de vários status do sistema.
- Três LEDs irão representar o status de proximidade com obstáculos: o LED vermelho será acesso quando o objeto estiver a uma distância menor do que 5 cm do sensor; o LED amarelo quando o obstáculo estiver em uma distância maior ou igual a 5 cm até igual a 10 cm de distância do sensor; e o LED verde acesso indicará uma distância ainda segura, acima de 10 cm de distância do sensor.
- Um buzzer irá dar os sinais sonoros de proximidade. Quanto mais perto do obstáculo mais curto será o intervalo dos “bips” emitidos pelo buzzer. Por exemplo, se o obstáculo estiver à 10 cm de distância, serão emitidos “bips” a cada 1000 ms

(emite um “bip” – aguarda 1000 ms – emite um “bip” – aguarda 1000 ms -...). O intervalo dos “bips” irá diminuir quanto mais próximo o obstáculo estiver do sensor. Se distância for de 9 cm o intervalo será de 900ms, se for 8 cm será de 800 ms, e assim por diante. Quanto menor o intervalo da emissão de som, mais alerta o motorista terá que ficar, pois indicará uma possível colisão. Esse sinal sonoro só se iniciará quando a distância for menor ou igual a 10 cm.

- Será utilizado um display LCD 16x2 para mostrar todos os estados do sistema. Quando o LED verde estiver acesso o display mostrará “SEGURO: XX cm”. Caso o LED amarelo esteja acesso, o display terá que mostrar “ATENCAO: XX cm”. E quando o LED vermelho estiver acesso e a distância for maior que 2 cm, mostrará “CUIDADO: XX cm”. E por fim, se o LED vermelho estiver acesso e a distância for menor ou igual a 2 cm, o display mostrará “PARE!!!”.

Projeto 10 – Piezo Elétrico

O objetivo desse projeto é utilizar um piezo elétrico para identificar uma batida (pressão sobre o piezo) e acionar o toque em um buzzer. Um piezo elétrico possui a capacidade de converter uma oscilação mecânica (pressão sobre ele) em um sinal elétrico. O inverso desse princípio, ou seja, converter um sinal elétrico em oscilação mecânica é o que ocorre em um buzzer. A pressão sobre o piezo gera uma pequena corrente elétrica, por isso possuem polaridade, sendo o centro prateado polo positivo (ligado ao fio vermelho) e a borda dourada, o negativo (ligado ao fio preto). Essa corrente elétrica atinge um limiar, dependendo da pressão, e é a partir dele é que verificamos se houve ou não um toque nele. Esse limiar deve ser ajustado para definir a sensibilidade necessária do toque e um valor analógico será lido pelo Arduino.

Material necessário:

- 1 Arduino.
- 1 Buzzer*.
- 1 Piezo elétrico.
- 1 Resistor de 1k ohms (marrom, preto, vermelho) .
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom)* .
- 1 Protoboard.
- Jumper cable.

** Podem ser substituídos pelo módulo P15-Buzzer da GBK Robotics.*

Passo 1: Montagem do circuito

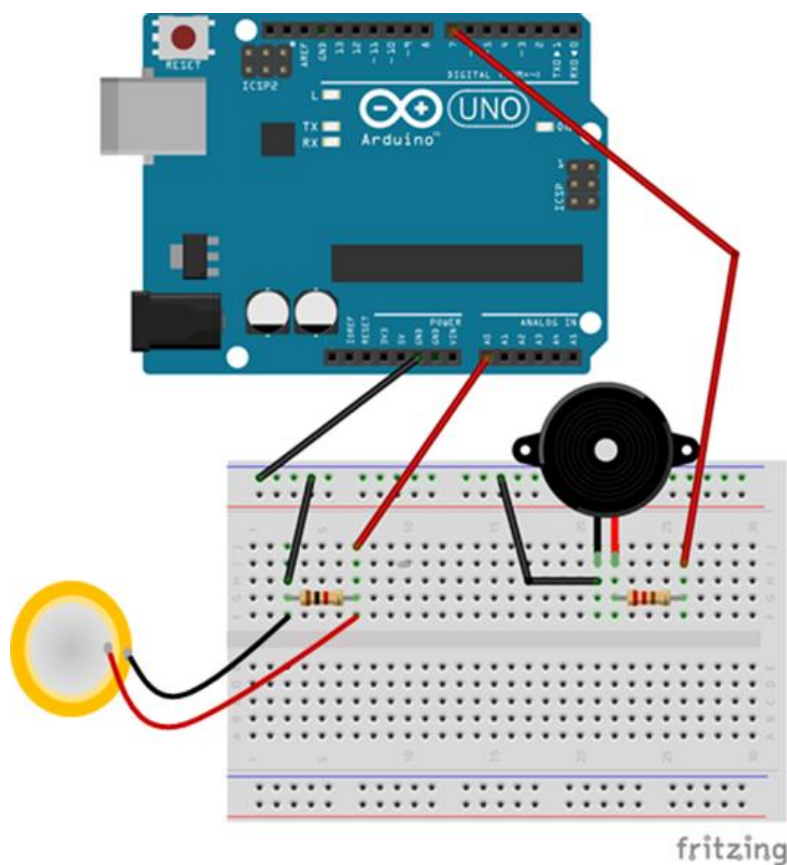


Figura 3.23: Ligação do Arduino ao Buzzer e ao Piezo

Realize a sequência de montagem ilustrada pela Figura 3.23:

- a. Conecte o GND do Arduino na alimentação GND da protoboard.
- b. Conecte um resistor de 1k ohms no sentido horizontal da protoboard.
- c. Conecte os fios do piezo nas extremidades desse resistor.
- d. Conecte um jumper no GND da protoboard e na linha do negativo do piezo.
- e. Conecte um jumper no positivo do piezo e no pino A0 do Arduino.
- f. Conecte o pino GND do Arduino ao pino negativo do buzzer já colocado na protoboard.
- g. Coloque o resistor de 220 ohms (ou 330 ohms) ligado ao pino positivo do buzzer.
- h. Nesse resistor, conecte um jumper até a porta digital 7 do Arduino.

Variação de Montagem

Módulo P15-Buzzer da GBK Robotics



Figura 3.24: Módulo P15-Buzzer

Este projeto pode ser montado substituindo o buzzer e o resistor de 220 ohms (ou 330 ohms) pelo módulo P15-Buzzer (Figura 3.24) da GBK Robotics, neste caso:

- Conecte o GND do Arduino na alimentação GND da protoboard.
- Conecte um resistor de 1k ohms no sentido horizontal da protoboard.
- Conecte os fios do piezo nas extremidades desse resistor.
- Conecte um jumper no GND da protoboard e na linha do negativo do piezo.
- Conecte um jumper no positivo do piezo e no pino A0 do Arduino.
- Conecte o pino GND do módulo P15 na linha de alimentação GND da protoboard.
- Conecte o pino 7 do Arduino ao pino Sinal do módulo P15.

IMPORTANTE: Não há alterações no sketch (programa).

Passo 2: Programa

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
int BUZZER = 7 ;
int PIEZO = A0;
int LIMIAR = 50;          // Valor do limiar, que
                           // define se houve toque ou não!
int VALOR_PIEZO = A0; // Valor que será lido
                           // pelo piezo

void setup() {
  pinMode(BUZZER, OUTPUT);
  pinMode(BUZZER, INPUT);
  Serial.begin(9600);
}

void loop() {
  VALOR_PIEZO = analogRead(PIEZO);

  // Verifica se o valor lido pela pressão no
  // piezo ultrapassa o limiar
  if (VALOR_PIEZO >= LIMIAR) {
    // Ativa o buzzer e mostra uma mensagem no
    // Monitor Serial
    tone(BUZZER, 1600, 200);
    Serial.println("Bip!");
  }
  delay(100);
}
```

Projeto 11 – Relógio com LCD

O objetivo deste projeto é criar um relógio digital a partir de um módulo Real Time Clock (RTC) e um display LCD 16x2. Neste projeto usaremos as bibliotecas **RTCLib.h** e **LiquidCrystal.h**. A biblioteca **RTCLib.h**, disponível para download em <https://github.com/adafruit/RTCLib>, irá fornecer as funções necessárias para a utilização do módulo RTC DS-1307. Por outro lado, a biblioteca **LiquidCrystal.h** possui funções que auxiliam nas configurações e tratamento dos dados a serem enviados ao LCD. A montagem do display deve ser de acordo com sua especificação (*datasheet*), onde cada um dos pinos apresenta uma função específica (ver no Passo 2 – Montagem do circuito). Para ver todas as funções disponíveis na biblioteca **LiquidCrystal.h** acesse o site oficial da biblioteca, que está disponível em <http://arduino.cc/en/Reference/LiquidCrystal>.

Material necessário:

- 1 Arduino.
- 1 Real Time Clock (RTC DS-1307).
- 1 LCD 16x2.
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) .
- 1 Protoboard.
- Jumper cable.

Passo 1: Importar a Biblioteca RTCLib

A biblioteca RTCLib não faz parte da distribuição padrão da IDE do Arduino, desta forma, torna-se necessário importá-la antes de utilizar o RTC pela primeira vez. Para isso, no menu “Sketch”, escolha a opção “Importar Biblioteca” e, “Add Library”, conforme ilustrado pela Figura 3.25.

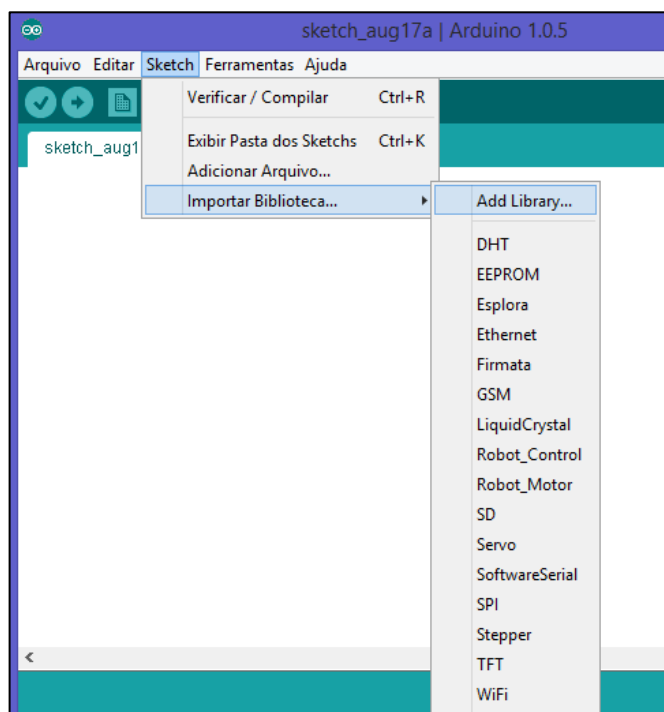


Figura 3.25: Adicionar uma Biblioteca

Em seguida, observe a Figura 3.26 e selecione o arquivo compactado (ZIP) que contém a biblioteca a ser importada, ou seja:

Aprenda Arduino – Uma abordagem prática

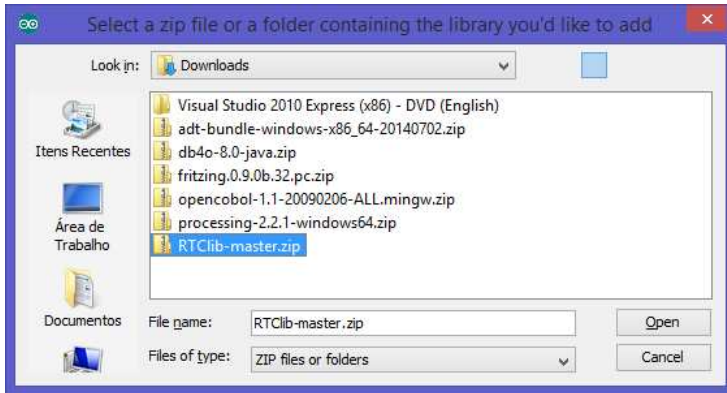


Figura 3.26: Selecionar o Arquivo Zip ou a Pasta

Após importar a biblioteca a mesma estará disponível para uso dentro da opção do menu “Importar Biblioteca”, conforme podemos observar na Figura 3.27.

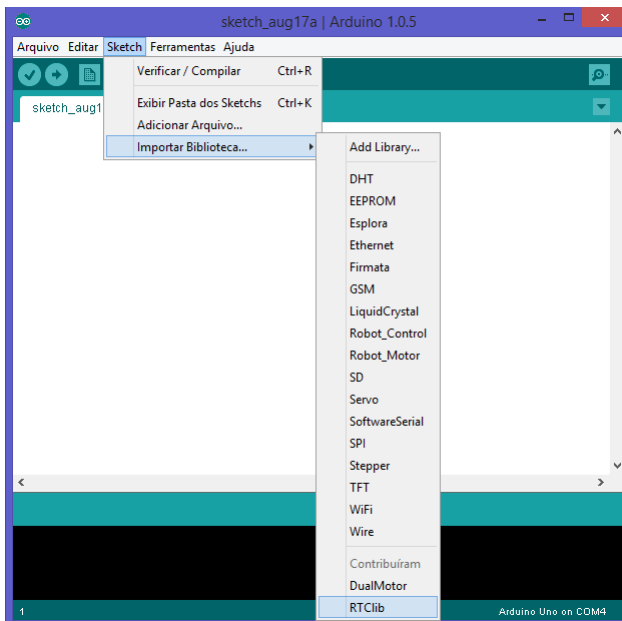


Figura 3.27: Seleção da Biblioteca RTClib

Passo 2: Montagem do circuito

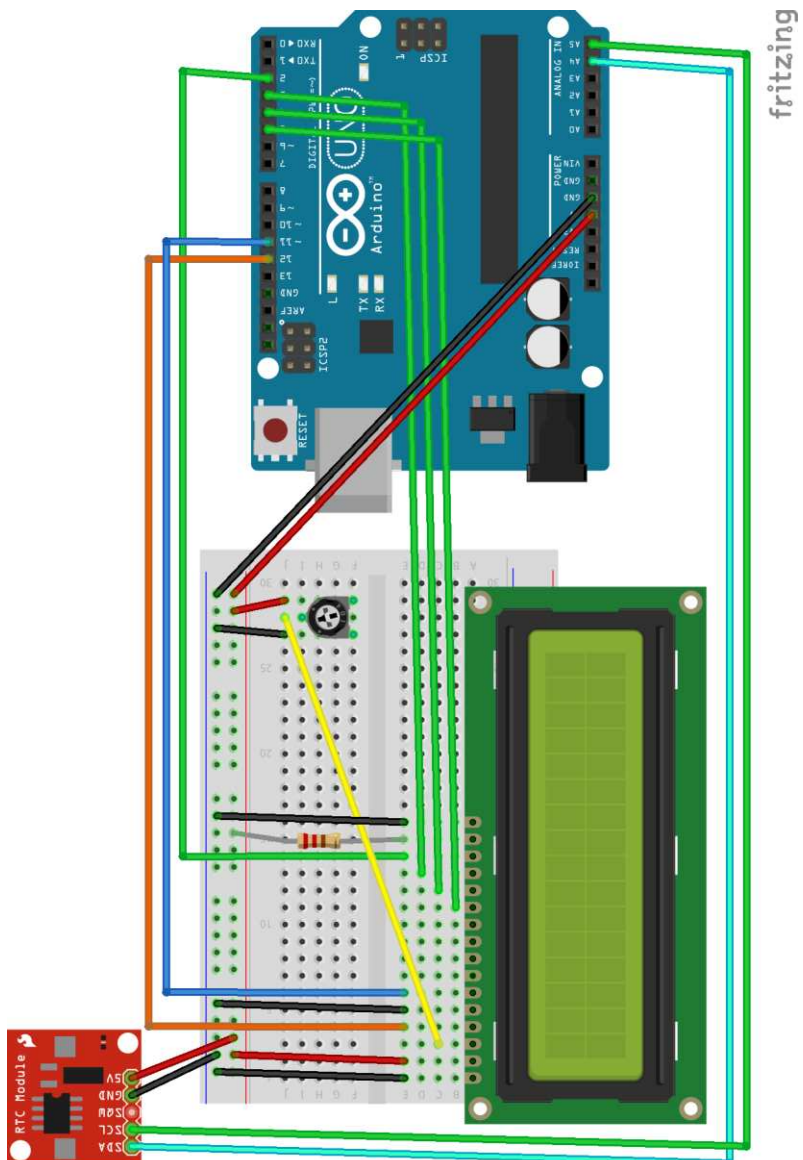


Figura 3.28: Ligação do Relógio em Tempo Real (RTC)

Com base na Figura 3.28, realizar esta sequência de montagem:

- Pino 1 do LCD ligado ao GND do Arduino.
- Pino 2 do LCD ligado ao 5V do Arduino.
- Pino 3 do LCD ligado ao pino central do potenciômetro (controle de contraste).
- Pino 4 do LCD ligado ao pino digital 12 do Arduino.
- Pino 5 do LCD ligado ao GND do Arduino.
- Pino 6 do LCD ligado ao pino digital 11 do Arduino.
- Pino 11 do LCD ligado ao pino digital 5 do Arduino.
- Pino 12 do LCD ligado ao pino digital 4 do Arduino.
- Pino 13 do LCD ligado ao pino digital 3 do Arduino.
- Pino 14 do LCD ligado ao pino digital 2 do Arduino.
- Pino 15 do LCD ligado ao 5V do Arduino com um resistor de 220 ohms (controle do brilho).
- Pino 16 do LCD ligado ao GND do Arduino.
- Pino SDA do RTC ligado ao pino analógico A4 do Arduino.
- Pino SCL do RTC ligado ao pino analógico A5 do Arduino.
- Pino GND do RTC ligado ao pino GND do Arduino.
- Pino VCC do RTC ligado ao 5V do Arduino.

Passo 3: Programa

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

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

#define TEMPO_ATUALIZACAO 1000

// RTC - Real Time Clock
// Conectar SCL (RTC) em A5 (Arduino) e
// SDA (RTC) em A4 (Arduino).

RTC_DS1307 RTC;

// LCD - Modelo ACM 1602K
// Pin Sigla Função                               Conectar
// 1  Vss   Ground                                GND
// 2  Vdd   +5V                                    VCC
// 3  Vo    LCD contrast adjust Potenciômetro
// 4  RS    Register select                       Arduino 12
// 5  R/W   Read/write                             GND
// 6  E     Enable                                 Arduino 11
// 7  DB0   Data bit 0                             NC
// 8  DB1   Data bit 1                             NC
// 9  DB2   Data bit 2                             NC
// 10 DB3   Data bit 3                             NC
// 11 DB4   Data bit 4                             Arduino 5
// 12 DB5   Data bit 5                             Arduino 4
// 13 DB6   Data bit 6                             Arduino 3
// 14 DB7   Data bit 7                             Arduino 2
// +  BL+   Alimentação (+)                       Resistor de 1k
//                                     para VCC
// -  BL-   Alimentação (-)                       GND
```

```

LiquidCrystal lcd (12, 11, 5, 4, 3, 2);

int dia, mes, ano, hora, minuto, segundo,
dia_semana;
char semana[][4] = {"DOM", "SEG", "TER", "QUA",
"QUI", "SEX", "SAB"};

void setup () {
  // Inicializa comunicação serial
  Serial.begin(9600);

  // Inicializa o protocolo Wire
  Wire.begin();

  // Inicializa o módulo RTC
  RTC.begin();

  // Verifica se o modulo esta funcionando
  if (! RTC.isrunning()) {
    Serial.println("O      RTC      não      está
executando!");
  }

  // Ajusta o relógio com a data e hora na qual
  // o programa foi compilado
  // RTC.adjust(DateTime(__DATE__, __TIME__));

  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  lcd.begin (16, 2);
}

void loop () {
  //Recupera data e hora atual
  DateTime now = RTC.now();
  dia = now.day();
  mes = now.month();

```

```
ano = now.year();
hora = now.hour();
minuto = now.minute();
segundo = now.second();
dia_semana = now.dayOfWeek();

lcd.clear();
if (dia < 10)
    lcd.print("0");
lcd.print(dia, DEC);
lcd.print("/");
if (mes < 10)
    lcd.print("0");
lcd.print(mes, DEC);
lcd.print("/");
lcd.print(now.year(), DEC);
lcd.setCursor(13, 0); // Coluna, Linha
lcd.print(semana[dia_semana]);
lcd.setCursor(0, 1); // Coluna, Linha
if (hora < 10)
    lcd.print("0");
lcd.print(hora, DEC);
lcd.print(":");
if (minuto < 10)
    lcd.print("0");
lcd.print(minuto, DEC);
lcd.print(":");
if (segundo < 10)
    lcd.print("0");
lcd.print(segundo, DEC);
delay(TEMPO_ATUALIZACAO);
}
```

Projeto 12 – Display de Led de 7 Segmentos

O objetivo deste projeto é demonstrar a utilização de um display de led de 7 segmentos controlado diretamente a partir das portas digitais do Arduino.

Material necessário:

- 1 Arduino.
- 1 Display de Led de 7 Segmentos Cátodo ou Ânodo Comum (1 dígito)*.
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom)*.
- 1 Protoboard*.
- Jumper cable.

** Podem ser substituídos pelo módulo P11-Display Simples da GBK Robotics.*

Passo 1: Displays de Led de 7 Segmentos

Os displays de led de sete segmentos (Figura 3.29) são bastante comuns e muitos utilizados para exibir, principalmente, informações numéricas. Podem ser de dois tipos:

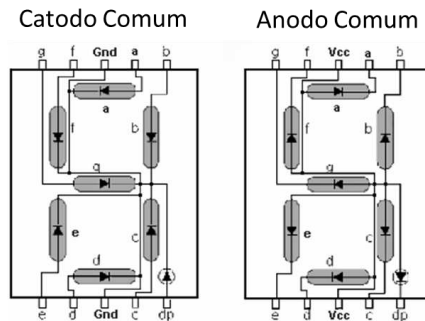


Figura 3.29: Tipos de Displays de LEDs de 7 Segmentos

Passo 2: Montagem do circuito

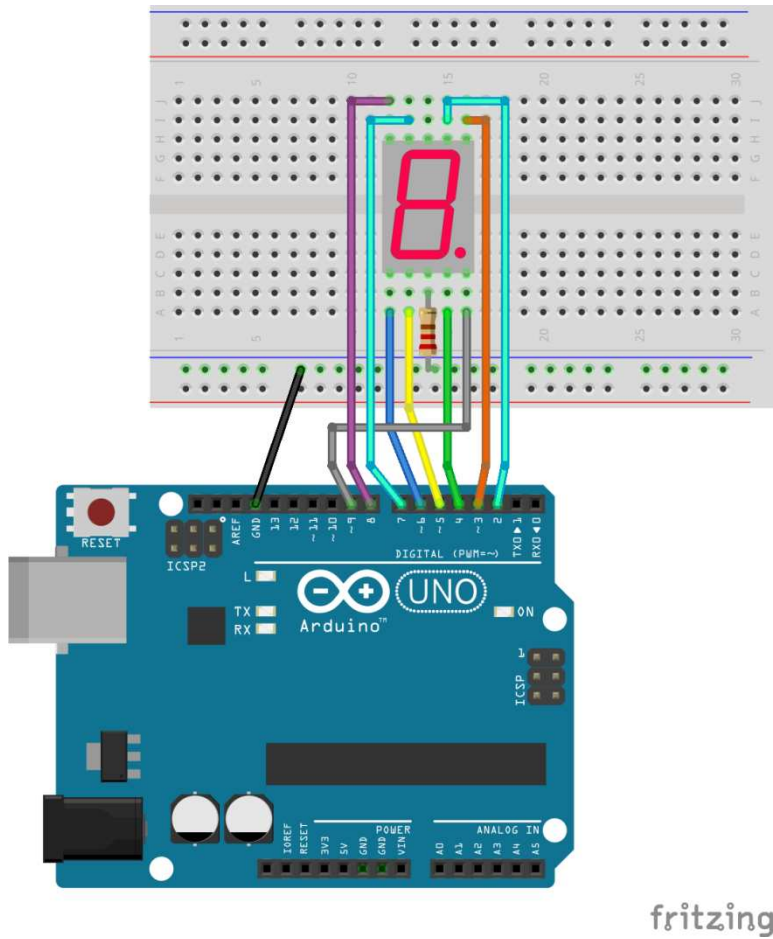


Figura 3.30: Ligação do Display de 7 Segmentos

Adotando como referência a Figura 3.30, realizar a sequência de montagem:

- Pino 1 (segmento e) do display ligado ao 6 do Arduino.
- Pino 2 (segmento d) do display ligado ao 5 do Arduino.
- Pino 3 (Gnd, se Cátodo comum ou Vcc se Ânodo comum) do display ligado ao resistor de 220 ohms.
- Resistor de 220 ohms ligado ao Gnd (se Cátodo comum) ou Vcc (se Ânodo comum) do Arduino.
- Pino 4 (segmento c) do display ligado ao 4 do Arduino.
- Pino 5 (ponto decimal) do display ligado ao 9 do Arduino.
- Pino 6 (segmento b) do display ligado ao 3 do Arduino.
- Pino 7 (segmento a) do display ligado ao 2 do Arduino.
- Pino 9 (segmento f) do display ligado ao 7 do Arduino.
- Pino 10 (segmento g) do display ligado ao 8 do Arduino.

Variação de Montagem

Módulo P11-Display Simples da GBK Robotics

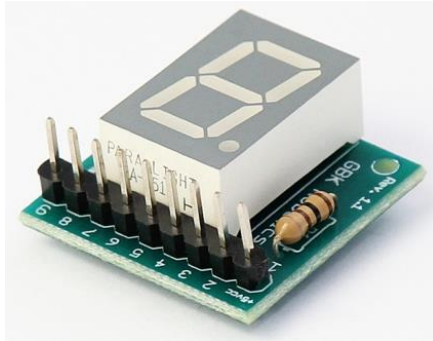


Figura 3.31: Módulo P11-Display Simples

Este projeto pode ser montado substituindo o Display de Led de 7 Segmentos, o resistor de 220 ohms (ou 330 ohms) e a protoboard pelo módulo P11-Display Simples (Figura 3.31) da GBK Robotics, neste caso:

- Conecte o pino 5V do Arduino ao pino +5Vcc do módulo P11.
- Conecte o pino 2 do Arduino ao pino 2 do módulo P11.
- Conecte o pino 3 do Arduino ao pino 3 do módulo P11.
- Conecte o pino 4 do Arduino ao pino 4 do módulo P11.
- Conecte o pino 5 do Arduino ao pino 5 do módulo P11.
- Conecte o pino 6 do Arduino ao pino 6 do módulo P11.
- Conecte o pino 7 do Arduino ao pino 7 do módulo P11.
- Conecte o pino 8 do Arduino ao pino 8 do módulo P11.
- Conecte o pino 9 do Arduino ao pino 9 do módulo P11.

IMPORTANTE: Para o módulo P11 nos sketches (programas) a seguir adotar a configuração para Ânodo comum.

Passo 3: Programa N° 1 – Exibindo a Letra “A”

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
int SEG_A = 2;
int SEG_B = 3;
int SEG_C = 4;
int SEG_D = 5;
int SEG_E = 6;
int SEG_F = 7;
int SEG_G = 8;
int PONTO = 9;
int ATRASO = 1000;

void setup() {
  for (int pino = SEG_A; pino <= PONTO; pino++)
  {
    pinMode(pino, OUTPUT);
    // Para displays de Cátodo comum:
    digitalWrite(pino, LOW);
    // Para displays de Ânodo comum:
    // digitalWrite(pino, HIGH);
  }
}

void loop() {
  // Para displays de Cátodo comum:
  digitalWrite(SEG_A, HIGH);
  digitalWrite(SEG_B, HIGH);
  digitalWrite(SEG_C, HIGH);
  digitalWrite(SEG_E, HIGH);
  digitalWrite(SEG_F, HIGH);
  digitalWrite(SEG_G, HIGH);

  // Para displays de Ânodo comum:
  // digitalWrite(SEG_A, LOW);
```

```
// digitalWrite(SEG_B, LOW);  
// digitalWrite(SEG_C, LOW);  
// digitalWrite(SEG_E, LOW);  
// digitalWrite(SEG_F, LOW);  
// digitalWrite(SEG_G, LOW);  
  
delay(ATRASO);  
}
```

Passo 4: Programa N° 2 – Animação Simples

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
int SEG_A = 2;  
int SEG_B = 3;  
int SEG_C = 4;  
int SEG_D = 5;  
int SEG_E = 6;  
int SEG_F = 7;  
int SEG_G = 8;  
int PONTO = 9;  
  
int ATRASO = 150;  
  
void setup() {  
  for (int pino = SEG_A; pino <= PONTO; pino++)  
  {  
    pinMode(pino, OUTPUT);  
    // Para displays de Cátodo comum:  
    digitalWrite(pino, LOW);  
    // Para displays de Ânodo comum:  
    // digitalWrite(pino, HIGH);  
  }  
}  
  
void loop() {
```

```

for (int pino = SEG_A; pino < SEG_G; pino++) {
  // Para displays de Cátodo comum:
  digitalWrite(pino, HIGH);
  // Para displays de Ânodo comum:
  // digitalWrite(pino, LOW);
  if (pino > SEG_A) {
    // Para displays de Cátodo comum:
    digitalWrite(pino - 1, LOW);
    // Para displays de Ânodo comum:
    // digitalWrite(pino - 1, HIGH);
  }
  else {
    // Para displays de Cátodo comum:
    digitalWrite(SEG_F, LOW);
    // Para displays de Ânodo comum:
    // digitalWrite(SEG_F, HIGH);
  }
  delay(ATRASO);
}
}

```

Passo 5: Programa N° 3 – Contagem Regressiva

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```

// Matriz com os dígitos de 0 a 9.
byte digitos[10][7] = {
  { 1,1,1,1,1,1,0 }, // = 0
  { 0,1,1,0,0,0,0 }, // = 1
  { 1,1,0,1,1,0,1 }, // = 2
  { 1,1,1,1,0,0,1 }, // = 3
  { 0,1,1,0,0,1,1 }, // = 4
  { 1,0,1,1,0,1,1 }, // = 5
  { 1,0,1,1,1,1,1 }, // = 6
  { 1,1,1,0,0,0,0 }, // = 7
  { 1,1,1,1,1,1,1 }, // = 8

```

```
{ 1,1,1,0,0,1,1 } // = 9
};

void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pontoDecimal(false);
}

void pontoDecimal(boolean ponto) {
  digitalWrite(9, ponto);
}

void escrever(int digito) {
  int pino = 2;
  for (int segmento = 0; segmento < 7;
segmento++) {
    // Para Cátodo Comum:
    digitalWrite(pino,
digitos[digito][segmento]);
    // Para Ânodo Comum - apenas inverter o
    // valor através do operador not (!):
    // digitalWrite(pino,
    // !digitos[digito][segmento]);
    pino++;
  }
  pontoDecimal(false);
}

void limpar() {
  byte pino = 2;
```



```

    for (int segmento = 0; segmento < 7;
segmento++) {
        // Para Cátodo Comum:
        digitalWrite(pino, LOW);
        // Para Ânodo Comum:
        // digitalWrite(pino, HIGH);
        pino++;
    }
}

void loop() {
    for (int cont = 9; cont >= 0; cont--) {
        escrever(cont);
        boolean ponto = true;
        for (int i = 0; i < 4; i++) {
            delay(250);
            pontoDecimal(ponto);
            ponto = !ponto;
        }
    }
    limpar();
    delay(1000);
}

```

Projeto 13 – Display de Led de 7 Segmentos e 4 Dígitos

O objetivo deste projeto é demonstrar a utilização do display de led de 7 segmentos e 4 dígitos controlado através do CI MAX 7219 ou 7221.

Material necessário:

- 1 Arduino.
- 1 Display de Led de 7 Segmentos (4 dígitos).
- 1 Circuito Integrado (CI) MAX 7219 ou 7221.
- 1 Resistor de 100k ohms (marrom, preto, amarelo).
- 1 Protoboard.
- Jumper cable.

Passo 1: Uso de displays com múltiplos dígitos

Você já deve ter observado que quando precisamos utilizar displays de leds que apresentam mais do que um dígito, as portas disponíveis no Arduino não serão suficientes ou mesmo que sejam suficientes, não permitirão colocar novas funcionalidades ao seu projeto como, por exemplo, um sensor de temperatura ou um módulo de relógio em tempo real (RTC). Desta maneira, para otimizar o uso das portas do Arduino devemos utilizar um driver para displays de led sendo, o mais popular, o Maxim 7219 ou 7221, cuja pinagem pode ser observada na Figura 3.32.

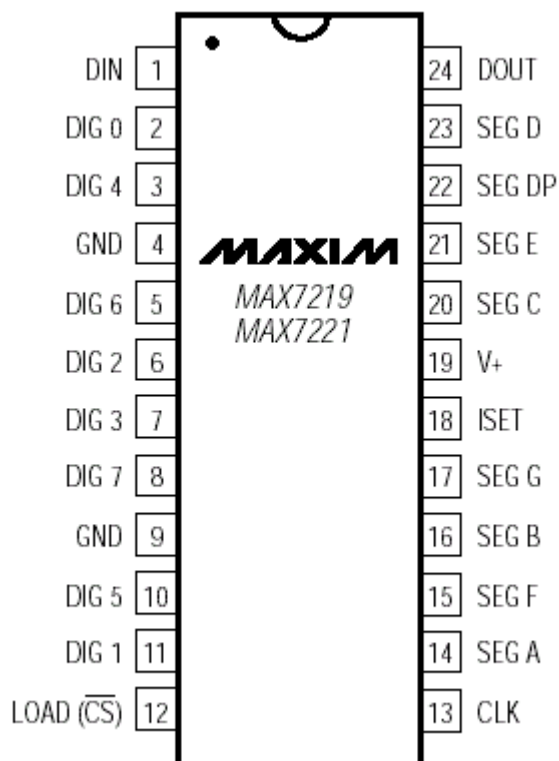


Figura 3.32: Pinos do Maxim 7219 ou 7221

Passo 2: Montagem do circuito

Realize as ligações indicadas na Figura 3.33.

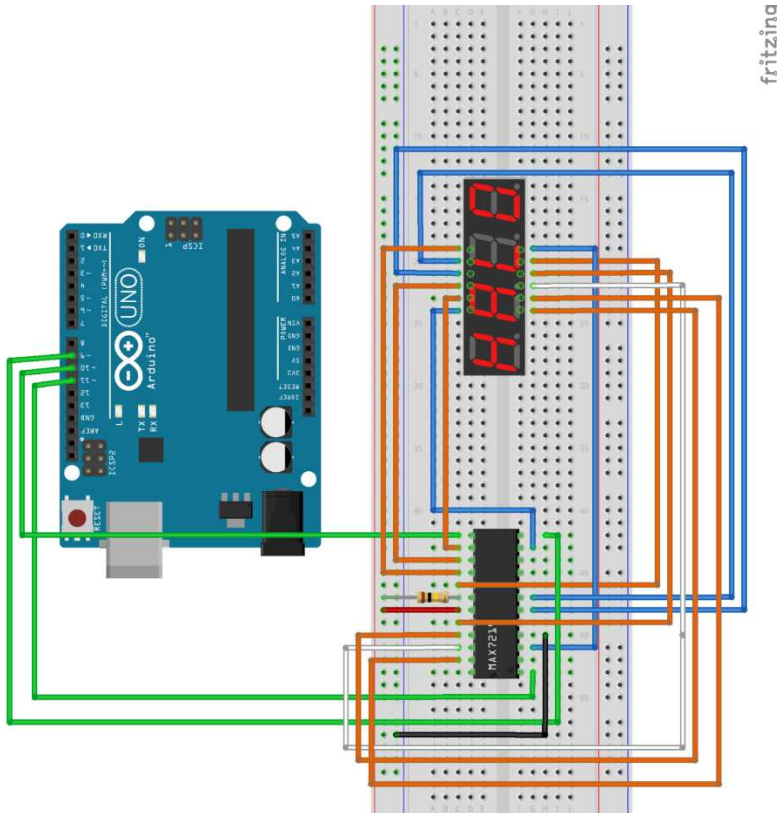


Figura 3.33: Ligação do Display com Múltiplos Dígitos

Passo 3: Programa N° 1

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
// Este sketch exibe o número 1234 em um display
// de led de 7 segmentos com 4 dígitos

// Utilizar a biblioteca LedControl
#include "LedControl.h"

/* 0 pino 11 do Arduino deve ser conectado ao
 * pino DATA IN do primeiro MAX7219/21
 * 0 pino 10 do Arduino deve ser conectado ao
 * pino CLK do primeiro MAX7219/21
 * 0 pino 9 do Arduino deve ser conectado ao
 * pino LOAD (/CS) do primeiro MAX7219/21
 * 0 quarto parâmetro indica que há apenas um
 * MAX7219/21 conectado ao Arduino
 */
LedControl lc = LedControl(11, 10, 9, 1);

void setup() {
  // Retira o MAX7219/21 do modo de economia
  // de energia
  lc.shutdown(0, false);
  // Define a intensidade do brilho dos leds
  lc.setIntensity(0, 2);
  lc.clearDisplay(0);
}

void loop() {
  lc.setChar(0, 2, '0', false);
  lc.setChar(0, 2, '1', false);
  lc.setChar(0, 3, '2', false);
  lc.setChar(0, 4, '3', false);
}
```

Passo 4: Programa N° 2

Acesse o ambiente de desenvolvimento do Arduino e digite o seguinte sketch.

```
// Este sketch exibe os números inteiros entre
// -999 e 999 em um display de led de
// 7 segmentos com 4 dígitos

// Utilizar a biblioteca LedControl
#include "LedControl.h"

/* O pino 11 do Arduino deve ser conectado ao
 * pino DATA IN do primeiro MAX7219/21
 * O pino 10 do Arduino deve ser conectado ao
 * pino CLK do primeiro MAX7219/21
 * O pino 9 do Arduino deve ser conectado ao
 * pino LOAD (/CS) do primeiro MAX7219/21
 * O quarto parâmetro indica que há apenas um
 * MAX7219/21 conectado ao Arduino
 */
LedControl lc = LedControl(11, 10, 9, 1);
int i = -999;

void setup() {
    lc.shutdown(0, false);
    lc.setIntensity(0, 2);
    lc.clearDisplay(0);
}

void loop() {
    exibirInteiro(i++);
}

void exibirInteiro(int valor) {
    int unidade;
    int dezena;
```

```
int centena;
boolean negativo = false;

if(valor < -999 || valor > 999)
    return;

if(valor < 0) {
    negativo = true;
    valor = valor * (-1);
}

unidade = valor % 10;
valor = valor / 10;
dezena = valor % 10;
valor = valor / 10;
centena = valor;

if (negativo) {
    // Imprime o sinal negativo no display
    // que está mais a esquerda
    lc.setChar(0, 1, '-', false);
}
else {
    // Imprime um espaço no lugar do sinal
    // negativo
    lc.setChar(0, 1, ' ', false);
}

// Exibe o número dígito a dígito
lc.setDigit(0, 2, (byte)centena, false);
lc.setDigit(0, 3, (byte)dezena, false);
lc.setDigit(0, 4, (byte)unidade, false);
delay(100);
}
```

Projeto 14 – Servo Motor

O objetivo destes projetos é mostrar o uso da biblioteca “Servo.h” (já existente na IDE Arduino) para a manipulação de servo motores. Um servo motor é um motor que tem sua rotação limitada através de uma angulação, podendo variar entre de 0º a 180º para indicar seu posicionamento. São muito utilizados em modelos de controle remoto, como em carrinhos, para girar o eixo de direção ou em barcos, para direcionamento do leme. Outras aplicações interessantes são em direcionamento automatizado de antenas e em articulações de braços robóticos.

A biblioteca “Servo.h” apresenta um conjunto de métodos, como o ***attach()***, para definir qual é o pino está conectado, e o ***write()***, para “escrever” o valor do ângulo no motor. Para a utilização desses métodos da biblioteca é preciso criar um objeto do tipo “servo”, procedimento similar ao do experimento usando o display LCD e a biblioteca “LiquidCrystal.h”.

Material necessário:

- 1 Arduino.
- 1 Servo motor RC padrão.
- 1 Potenciômetro 10K (resistor variável).
- 1 Protoboard.
- Jumper cable.

Passo 1: Montagem do circuito

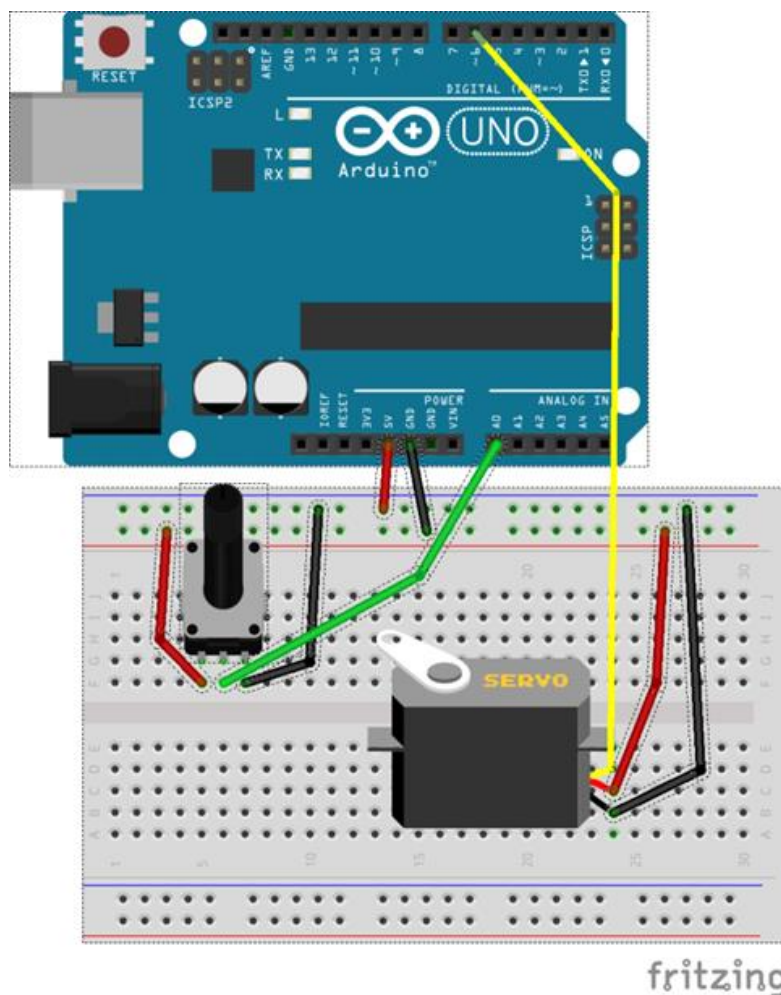


Figura 3.34: Conexão do Servo Motor

Adotando, como referência, a Figura 3.34 realize as seguintes ligações:

- Cabo de alimentação GND do servo motor (cabo preto) na alimentação GND da protoboard.
- Cabo de alimentação de 5V do servo motor (vermelho ou outra cor, logo ao lado do preto) na alimentação 5V da protoboard.
- Cabo de sinal do servo motor (amarelo) no pino digital 6 do Arduino.
- Pino positivo do potenciômetro na alimentação 5V da protoboard.
- Pino negativo do potenciômetro na alimentação GND da protoboard.
- Pino do meio (sinal) do potenciômetro no pino analógico A0 do Arduino.
- Pino 5V do Arduino na trilha positiva da protoboard.
- Pino GND do Arduino na trilha negativa da protoboard.

Passo 2: Programa N° 1

Neste programa vamos utilizar o potenciômetro para variar a angulação do servo motor. Quanto maior o valor proveniente do sinal do potenciômetro maior o ângulo, e vice-versa. Como a variação do potenciômetro é de 0 a 1023 e o do servo motor é de 0 a 180, utilizamos a função **map()**, como o objetivo de converter o *range* (intervalo de variação) do potenciômetro com o do servo motor. A função **map()** é definida como:

map(valor, deMin, deMax, paraMin, paraBaixo)

Onde:

- **valor**: valor a ser convertido.
- **deMin**: valor mínimo do intervalo do valor.
- **deMax**: valor máximo do intervalo do valor.
- **paraMin**: valor mínimo do intervalo a ser convertido.
- **paraMax**: valor máximo do intervalo a ser convertido.

Basicamente a função **map()** faz uma “regra de 3”, para determinar o valor de um intervalo em outro intervalo. Em nosso programa usamos **map(valorPot, 0, 1023, 0, 180)**, e podemos entender como *“o valor do potenciômetro que está no intervalo de 0 a 1023, será em convertido em um valor equivalente no intervalo de 0 a 180”*. Vamos implementar o sketch a seguir para demonstrar o uso do servo motor.

```
#include "Servo.h"

Servo servo;
int Pinpotenciometro = 0;
int PinservoMotor = 6;
int valorPot;
int valorMotor=0;

void setup() {
  servo.attach(PinservoMotor);
  Serial.begin(9600);
}

void loop() {
  valorPot = analogRead(Pinpotenciometro);
  valorMotor = map(valorPot, 0, 1023, 0, 180);
```

```
servo.write (valorMotor);  
Serial.print(valorMotor);  
delay(20);  
}
```

Passo 3: Programa N° 2

Mantenha a mesma montagem, apenas despreze o potenciômetro, já que não será usado. Nesse próximo programa, quando digitado a letra 'D' ou 'd' no Serial Monitor para fazer com que o servo motor diminua sua angulação, de 15º em 15º. Caso digitado 'A' ou 'a' aumenta sua angulação. No Serial Monitor, digite a letra e aperte ENTER ou o botão "Enviar". Acesse o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir para implementá-lo.

```
#include "Servo.h"  
  
Servo servo;  
int PinservoMotor = 6;  
char tecla;  
int valorMotor=0;  
  
void setup() {  
  servo.attach(PinservoMotor);  
}  
  
void loop() {  
  tecla = Serial.read();  
  if (tecla == 'D' || tecla == 'd') {  
    valorMotor = valorMotor - 15;  
    if (valorMotor >=180) {  
      valorMotor = 180;  
    }  
  }
```

```
}  
else if (tecla == 'A' || tecla == 'a') {  
    valorMotor = valorMotor + 15;  
    if (valorMotor <= 0) {  
        valorMotor = 0;  
    }  
}  
servo.write(valorMotor);  
delay(20);  
}
```

Projeto 15 – Sensor Óptico Reflexivo

Este projeto irá utilizar um Sensor Óptico Reflexivo TCRT5000 para implementar um interruptor de proximidade. Desta forma, não será necessário que a pessoa toque o sensor para acender ou apagar um Led. Esse tipo de circuito é muito útil quando desejamos manter a pessoa “isolada” do circuito elétrico evitando choques indesejáveis.

Um Sensor Óptico Reflexivo consiste em um diodo emissor (ou Led) de infravermelho, igual ao utilizado em controles remotos e um foto transistor que irá receber o sinal quando houver uma reflexão, ou seja, quando um obstáculo estiver à frente do sensor. No exemplo que iremos desenvolver vamos utilizar o modelo TCRT5000 (Figura 3.35), porém, o projeto pode ser facilmente alterado para utilizar outro modelo similar.

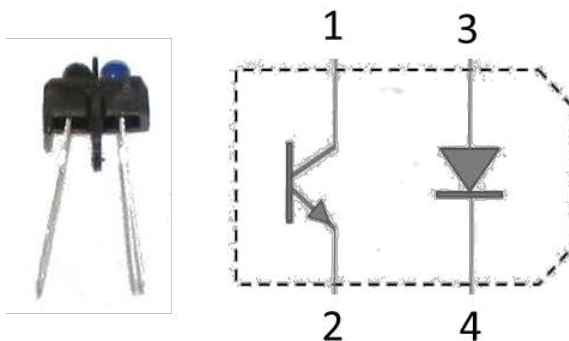


Figura 3.35: Vista lateral e superior do Sensor Óptico Reflexivo TCRT5000

No quadro a seguir estão relacionados os pinos do sensor com as respectivas funções.

Pino	Nome	Função
1	Coletor (T+)	Coletor do fototransistor
2	Emissor (T-)	Emissor do fototransistor
3	Ânodo (D+)	Ânodo do led infravermelho
4	Cátodo (D-)	Cátodo do led intravermelho

Material necessário:

- 1 Arduino.
- 1 Sensor Óptico Reflexivo TCRT5000*.
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) ou de 330 ohms (laranja, laranja, marrom) para a ligação do LED.
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) ou de 330 ohms (laranja, laranja, marrom) para ligação ao Sensor Óptico Reflexivo*.
- 1 Resistor de 10k ohms (marrom, preto, laranja)*.
- 1 Protoboard.
- Jumper cable.

** Podem ser substituídos pelo módulo P12-Sensor de Obstáculos da GBK Robotics.*

Montagem do circuito

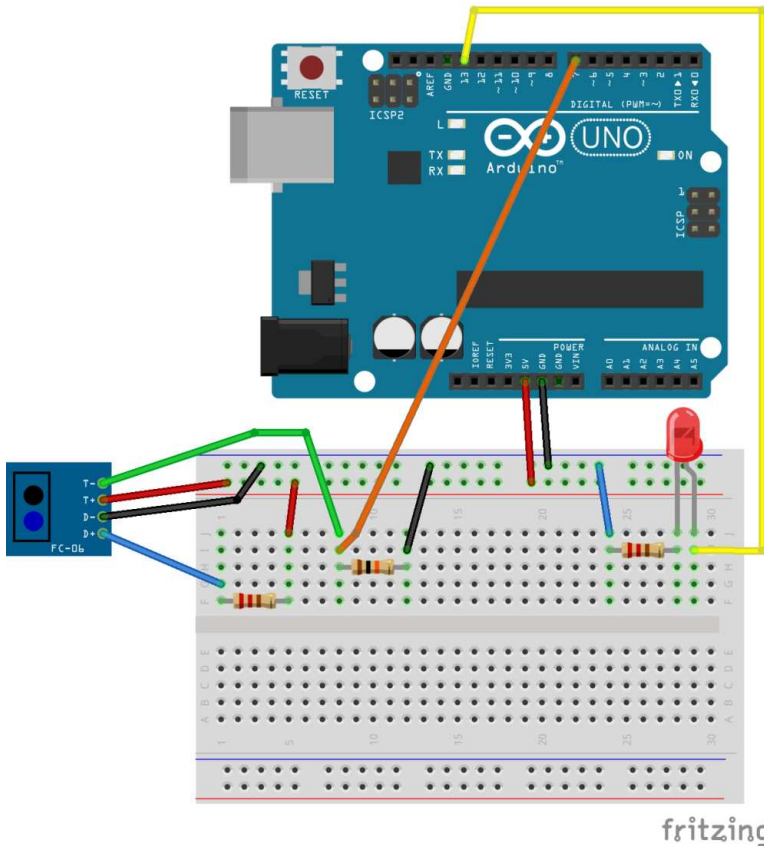


Figura 3.36: Interruptor de proximidade

Adotando como referência a Figura 3.36 realize os seguintes passos para montar o hardware que será usado neste projeto:

- Inserir o sensor óptico reflexivo na protoboard.
- Conectar o pino 4 (D-) do sensor na linha de alimentação negativa (preta ou azul) da protoboard.

- c) Inserir um resistor de 220 ohms (ou 330 ohms) na protoboard e conecte um dos seus terminais no pino 3 (D+) do sensor.
- d) Conecte o outro terminal do resistor de 220 ohms (ou 330 ohms) na linha de alimentação positiva (vermelha) da protoboard.
- e) Insira o resistor de 10k ohms na protoboard e conecte um dos seus terminais ao pino 2 (T-) do sensor.
- f) Conecte o mesmo terminal do resistor de 10k ohms ao pino digital 7 do Arduino.
- g) Conecte o outro terminal do resistor de 10k ohms à linha de alimentação negativa (preta ou azul) da protoboard.
- h) Conecte o pino 1 (T+) do sensor a linha de alimentação positiva (vermelha) da protoboard.
- i) Insira o outro resistor de 220 ohms (ou 330 ohms) na protoboard e conecte um dos seus terminais na linha de alimentação negativa (preta ou azul) da protoboard.
- j) Insira na protoboard o led com o cátodo (lado chanfrado e que possui o terminal mais curto) conectado ao outro terminal do resistor de 220 ohms (ou 330 ohms).
- k) Conecte o ânodo do led ao pino digital 13 do Arduino.
- l) Conecte o pino 5 Volts do Arduino à linha de alimentação positiva (vermelha) da protoboard.
- m) Conecte o pino Gnd do Arduino à linha de alimentação negativa (preta ou azul) da protoboard.

Variação de Montagem

Módulo P12-Sensor de Obstáculos da GBK Robotics



Figura 3.37: Módulo P12-Sensor de Obstáculos

Este projeto pode ser montado substituindo o Sensor Óptico Reflexivo TCRT5000, um dos resistores de 220 ohms (ou 330 ohms) e o resistor 10k ohms pelo módulo P12-Sensor de Obstáculos (Figura 3.37), neste caso:

- Conecte o pino Gnd do Arduino à linha de alimentação negativa (preta ou azul) da protoboard.
- Conecte o pino 5+ do módulo P12 ao pino 5V do Arduino.
- Conecte o pino GND do módulo P12 à linha de alimentação negativa (preta ou azul) da protoboard.
- Ligue o pino 7 do Arduino ao pino A0 do módulo P12.
- Insira o resistor de 220 ohms (ou 330 ohms) na protoboard e conecte um dos seus terminais na linha de alimentação negativa (preta ou azul) da protoboard.
- Insira na protoboard o led com o cátodo (lado chanfrado e que possui o terminal mais curto) conectado ao outro terminal do resistor de 220 ohms (ou 330 ohms).
- Conecte o ânodo do led ao pino digital 13 do Arduino.

IMPORTANTE: Não há alterações no sketch (programa).

Programa N° 1: Obtendo o valor através da entrada digital

Após montar o circuito, entre no ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir.

```
int LED = 13;
// Pino digital que irá receber o sinal do
// fototransistor
int SENSOR = 7;
int valor;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(SENSOR, INPUT);
}

void loop() {
  //Obter o valor do sensor, LOW ou HIGH
  valor = digitalRead(SENSOR);

  digitalWrite(LED, valor);
  delay (100);
}
```

Programa N° 2: Obtendo o valor através da entrada analógica

Entre no ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir.

```
int LED = 13;
// Pino analógico que irá receber o sinal do
// fototransistor
int SENSOR = A0;
```

```
int valor;

void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
}

void loop() {
  // Obter o valor do sensor
  // Que consiste em um valor entre 0 e 1023
  valor = analogRead(SENSOR);

  Serial.print("Valor: ");
  Serial.println(valor);

  if (valor > 300)
    digitalWrite(LED, HIGH);
  else
    digitalWrite(LED, LOW);
  delay (500);
}
```

Neste exemplo, observe que precisamos definir um valor de liminar (300) para determinar se o LED será ou não aceso. Se necessário, ajuste esse valor para as condições do ambiente e também em relação à distância desejada.

Projeto 16 – Teclado com Divisor de Tensão

Neste projeto será aplicado o conceito de divisor de tensão para obter o valor de vários botões através de uma única porta analógica. A aplicação deste conceito permite “poupar” o uso de portas do Arduino em projetos que requerem a utilização de muitos botões.

Material necessário:

- 1 Arduino.
- 3 Interruptores táteis (push button).
- 3 Resistores de 10 kohms (marrom, preto, laranja).
- 1 Protoboard.
- Jumper cable.

Montagem do circuito

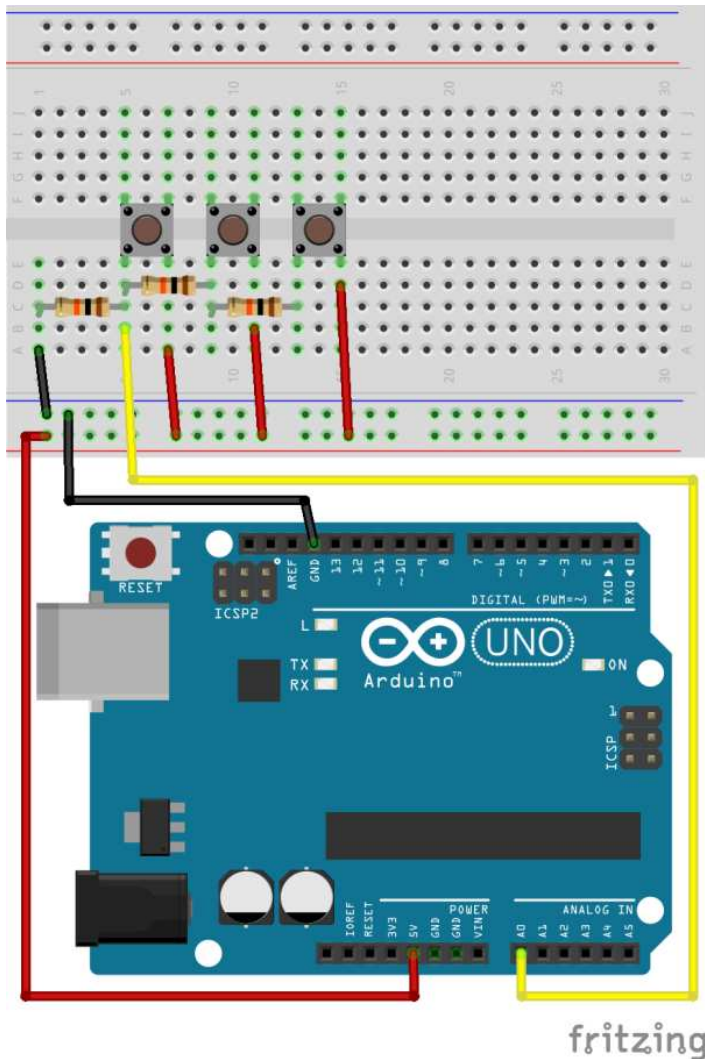


Figura 3.38: Teclado com divisor de tensão

Adotando como referência a Figura 3.38 realize a montagem do circuito que será usado neste projeto.

Programa

No ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir.

```
const int TECLADO = A0;

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

void loop() {
  long valor = 0;

  for(int i = 0; i < 20; i++) {
    valor += analogRead(TECLADO);
  }
  valor /= 20;

  if (valor > 0) {
    Serial.print("Teclado = ");
    Serial.print(valor);
    if (valor > 1020)
      Serial.println(", Tecla 1");
    else if (valor > 505 && valor < 515)
      Serial.println(", Tecla 2");
    else if (valor > 335 && valor < 345)
      Serial.println(", Tecla 3");
    else
      Serial.println("");
  }
  delay(200);
}
```

Projeto 17 – Infravermelho

O objetivo deste projeto é demonstrar a utilização de um receptor de infravermelho. O mesmo irá receber um sinal de um controle remoto e controlará o acendimento de um Led.

Material necessário:

- 1 Arduino.
- 1 Led (qualquer cor).
- 1 Receptor IR (infravermelho)*.
- 1 Transmissor IR (qualquer controle remoto).
- 1 Resistor de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom) para o LED.
- 1 Protoboard.
- Jumper cable.

** Pode ser substituído pelo módulo P14-IR Receiver da GBK Robotics.*

Passo 1: Montagem do Circuito N° 1

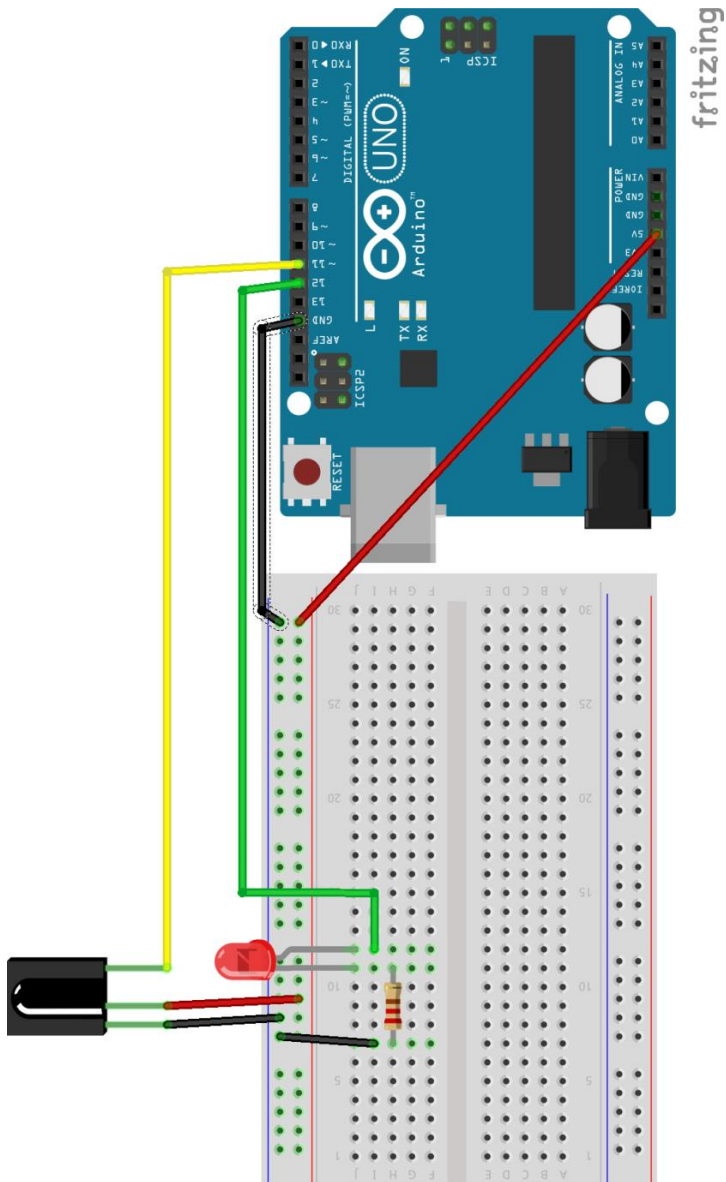


Figura 3.39: Ligação do Receptor Infravermelho

Conforme ilustra a Figura 3.39:

- a. Conecte o pino GND do Arduino à linha de alimentação negativa (preta) da protoboard.
- b. Conecte o pino 5 volts do Arduino à linha de alimentação positiva (vermelha) da protoboard.
- c. Coloque o resistor de 220 ohms (ou 330 ohms) entre a linha de alimentação negativa e qualquer outra linha da protoboard.
- d. Coloque o led com o cátodo (lado chanfrado) conectado ao resistor.
- e. Conecte o ânodo do led ao pino 12 do Arduino.
- f. Pino GND do receptor infravermelho na linha de alimentação GND da protoboard.
- g. Pino VCC do receptor infravermelho na linha de alimentação positiva (5 volts) da protoboard.
- h. Pino de dados do receptor infravermelho no pino 11 do Arduino.

Variação de Montagem

Módulo P14-IR Receiver da GBK Robotics

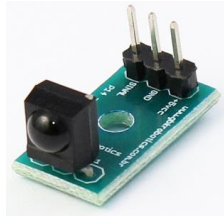


Figura 3.40: Módulo P14-IR Receiver

Este projeto pode ser montado substituindo o receptor de IR pelo módulo P14-IR Receiver (Figura 3.40), neste caso:

- Conecte o pino GND do Arduino à linha de alimentação negativa (preta) da protoboard.
- Conecte o pino 5 volts do Arduino à linha de alimentação positiva (vermelha) da protoboard.
- Coloque o resistor de 220 ohms (ou 330 ohms) entre a linha de alimentação negativa e qualquer outra linha da protoboard.
- Coloque o led com o cátodo (lado chanfrado) conectado ao resistor.
- Conecte o ânodo do led ao pino 12 do Arduino.
- Conecte o pino GND do Arduino ao pino GND do módulo P14.
- Conecte o pino 5V do Arduino ao pino +5Vcc do módulo P14.
- Conecte o pino 11 do Arduino ao pino Sinal do módulo P14.

IMPORTANTE: Não há alterações no sketch (programa).

Passo 2: Programa N° 1 – Acendendo e apagando um Led através do controle remoto

Neste projeto utilizaremos a biblioteca IRremote a qual está disponível para download em <http://vansan.com.br/arduino/IRremote.zip>. Em seguida, descompacte o conteúdo do arquivo IRremote.zip na pasta Arduino\libraries que está dentro da pasta Documentos. Note também que em algumas versões do ambiente de desenvolvimento do Arduino podem ocorrer incompatibilidades e conflitos com esta biblioteca. Nas versões 1.6, 1.01 e 0022 a biblioteca funcionará corretamente, porém a partir da versão 1.0.6 após a importação da biblioteca pode ocorrer um erro no momento da compilação. Este erro é ocasionado por uma incompatibilidade com a biblioteca RobotIRremote e, para resolver este problema, apague esta biblioteca acessando a pasta onde o Arduino está instalado e excluindo a pasta RobotIRremote.

No ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
#include "IRremote.h"

int RECEPTOR = 11;
int LED = 12;
IRrecv controle(RECEPTOR);
decode_results resultado;
int estado = LOW;

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

```
// Iniciar a recepção
controle.enableIRIn();

// O LED conectado ao pino 13 irá piscar
// sempre que um sinal for recebido
controle.blink13(true);

pinMode(LED, OUTPUT);
}

void loop() {
  if (controle.decode(&resultado)) {
    if (resultado.decode_type == NEC) {
      Serial.print("NEC: ");
    }
    else if (resultado.decode_type == SONY) {
      Serial.print("SONY: ");
    }
    else if (resultado.decode_type == RC5) {
      Serial.print("RC5: ");
    }
    else if (resultado.decode_type == RC6) {
      Serial.print("RC6: ");
    }
    else if (resultado.decode_type == UNKNOWN) {
      Serial.print("Desconhecido ");
    }
    Serial.println(resultado.value, HEX);

    // Se necessário, alterar o valor das teclas
    // que irão apagar e acender o LED
    // respectivamente, conforme o modelo do
    // controle remoto
    if (resultado.value == 0x8B7D22D)
      estado = LOW;
    else if (resultado.value == 0x8B752AD)
      estado = HIGH;
  }
}
```

```
digitalWrite(LED, estado);

// Obter o próximo valor
controle.resume();
}
}
```

Passo 3: Programa N° 2 – Alterando a frequência de acendimento do LED

Após a leitura do controle remoto, podemos aplicar este código que irá alterar a velocidade (frequência) que o Led conectado ao pino 13 do Arduino irá piscar. Na linha dos desvios condicionais, estamos usando valores em hexadecimal, por isso colocamos o prefixo **0x**. Por exemplo, se o valor lido foi FF6897 o comando ficará `if (results.value==0xFF6897)`.

```
#include <IRremote.h>

int LED = 12;
int RECV_PIN = 11;

IRrecv irrecv(RECV_PIN);
decode_results results;
int estadoLed = LOW;
long tempoAnterior = 0;
long intervalo = 10000;

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(LED, OUTPUT);
}
```

```
void loop() {  
  if (irrecv.decode(&results)) {  
    Serial.println(results.value, HEX);  
    // Se necessário, alterar o valor das teclas  
    // de acordo com o modelo do controle remoto  
    // Tecla 0  
    if (results.value==0xFF6897)  
      intervalo=50;  
    // Tecla 1  
    if (results.value==0xFF30CF)  
      intervalo=100;  
    // Tecla 2  
    if (results.value==0xFF18E7)  
      intervalo=300;  
    // Tecla 3  
    if (results.value==0xFF7A85)  
      intervalo=500;  
    // Tecla 4  
    if (results.value==0xFF10EF)  
      intervalo=1000;  
    // Tecla 5  
    if (results.value==0xFF38C7)  
      intervalo=2000;  
    // Tecla 6  
    if (results.value==0xFF5AA5)  
      intervalo=3000;  
    // Tecla 7  
    if (results.value==0xFF42BD)  
      intervalo=4000;  
    // Tecla 8  
    if (results.value==0xFF4AB5)  
      intervalo=5000;  
    // Tecla 9  
    if (results.value==0xFF52AD)  
      intervalo=10000;  
    irrecv.resume();  
  }  
}
```

```
// Obtém o tempo atual em milissegundos
unsigned long tempoAgora = millis();

// Piscar o LED
if(tempoAgora - tempoAnterior > intervalo) {
    tempoAnterior = tempoAgora;
    if (estadoLed == LOW)
        estadoLed = HIGH;
    else
        estadoLed = LOW;
    digitalWrite(LED, estadoLed);
}
}
```

Passo 4: Montagem do Circuito N° 2

Monte o circuito conforme ilustra a Figura 3.41.

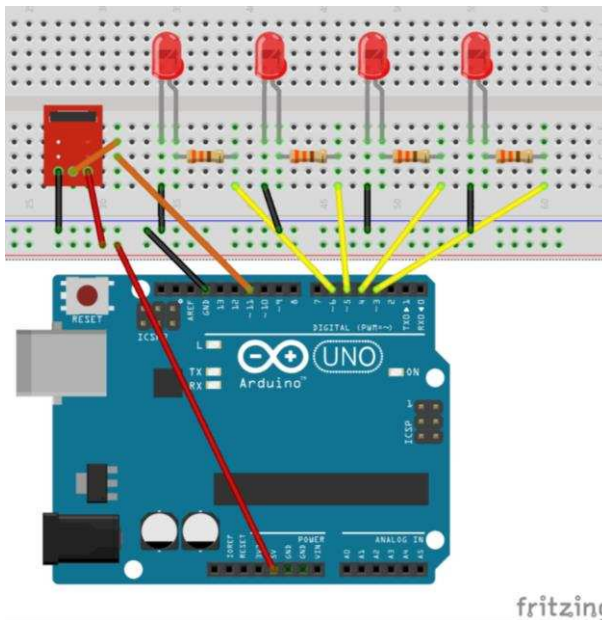


Figura 3.41: Receptor de IR Controlando 4 LEDs

Passo 5: Programa N° 3 – Controlando vários Leds

Observe que, neste programa, vamos utilizar um controle remoto para determinar o funcionamento de 4 Leds.

```
#include <IRremote.h>

int led01 = 6;
int led02 = 5;
int led03 = 4;
int led04 = 3;
int RECV_PIN = 11;

IRrecv irrecv(RECV_PIN);
decode_results results;

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(led01, OUTPUT);
  pinMode(led02, OUTPUT);
  pinMode(led03, OUTPUT);
  pinMode(led04, OUTPUT);
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);

    // Se necessário, alterar o valor das teclas
    // de acordo com o modelo do controle remoto
    if (results.value==0xFF6897){
      // Tecla 0 desliga todos os Leds
      digitalWrite(led01, LOW);
      digitalWrite(led02, LOW);
      digitalWrite(led03, LOW);
    }
  }
}
```

```

        digitalWrite(led04, LOW);
    }
    else if (results.value==0xFF30CF) {
        // Tecla 1 Liga Led 1
        digitalWrite(led01,HIGH);
    }
    else if (results.value==0xFF18E7)
        // Tecla 2 Liga Led 2
        digitalWrite(led02,HIGH);
    }
    else if (results.value==0xFF7A85) {
        // Tecla 3 Liga Led 3
        digitalWrite(led03,HIGH);
    }
    else if (results.value==0xFF10EF) {
        // Tecla 4 Liga Led 4
        digitalWrite(led04,HIGH);
    }
    else if (results.value==0xFF38C7) {
        // Tecla 5 Desliga Led 1
        digitalWrite(led01,LOW);
    }
    else if (results.value==0xFF5AA5) {
        // Tecla 6 Desliga Led 2
        digitalWrite(led02,LOW);
    }
    else if (results.value==0xFF42BD) {
        // Tecla 7 Desliga Led 3
        digitalWrite(led03,LOW);
    }
    else if (results.value==0xFF4AB5) {
        // Tecla 8 Desliga Led 4
        digitalWrite(led04,LOW);
    }
    else if (results.value==0xFF52AD) {
        // Tecla 9 Liga Todos LEDs
        digitalWrite(led01, HIGH);
    }

```

```
        digitalWrite(led02, HIGH);  
        digitalWrite(led03, HIGH);  
        digitalWrite(led04, HIGH);  
    }  
    irrecv.resume();  
}  
}
```

Projeto 18 – Contador Binário

O objetivo deste projeto é utilizar três LEDs para mostrar os números entre 0 e 7 no sistema de numeração binário, ou seja 0 (Desligado - LOW) ou 1 (Ligado - HIGH).

Número	LED 1	LED 2	LED 3
0	Low	Low	Low
1	Low	Low	High
2	Low	High	Low
3	Low	High	High
4	High	Low	Low
5	High	Low	High
6	High	High	Low
7	High	High	High

Material necessário:

- 1 Arduino.
- 3 Resistores de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom) para o LED*.
- 3 LEDs (qualquer cor)*.
- 1 Protoboard*.
- Jumper cable.

** Podem ser substituídos pelo módulo P7-Sensor de Luminosidade da GBK Robotics.*

Passo 1: Montagem do circuito

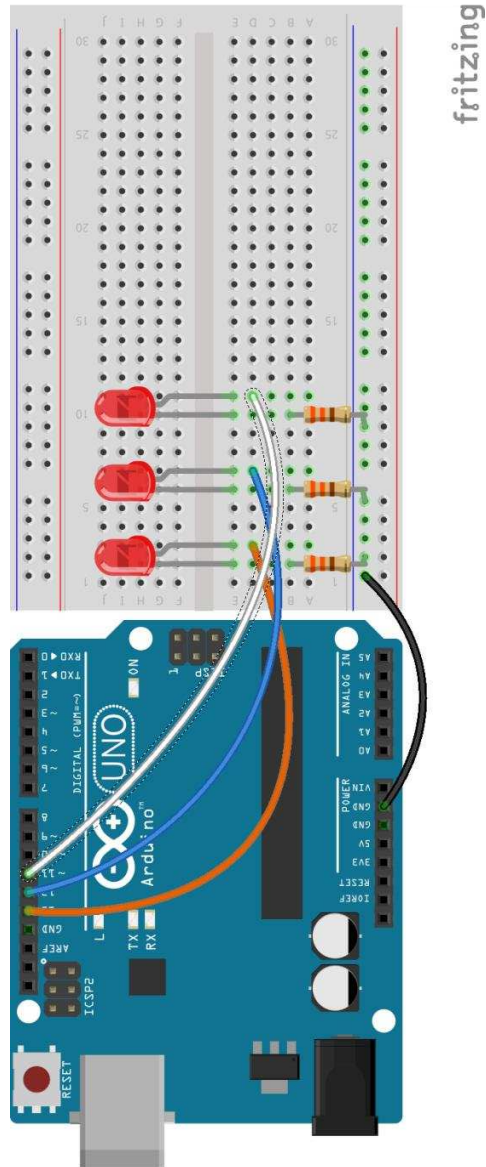


Figura 3.42: Montagem do Contador Binário

Conforme ilustra a Figura 3.42:

- a. Conecte o pino GND do Arduino à linha de alimentação negativa (preta ou azul) da protoboard.
- b. Coloque os três resistores de 220 ohms (ou 330 ohms) entre a linha de alimentação negativa e qualquer outra linha da protoboard.
- c. Coloque cada um dos três LEDs com o Cátodo (lado chanfrado) conectado a um dos resistores de 220 ohms (ou 330 ohms).
- d. Conecte o Ânodo do primeiro LED ao pino 11 do Arduino.
- e. Conecte o Ânodo do segundo LED ao pino 12 do Arduino.
- f. Conecte o Ânodo do terceiro LED ao pino 13 do Arduino.

Variação de Montagem 1

Módulo P7-Sensor de Luminosidade da GBK Robotics



Figura 3.43: Módulo P7-Sensor de Luminosidade

Este projeto pode ser montado substituindo os LEDs, os Resistores de 220 ohms (ou 330 ohms) e a Protoboard pelo módulo P7-Sensor de Luminosidade (Figura 3.43), neste caso:

- Conecte o pino GND do módulo P7 a um dos pinos de GND do Arduino.
- Conecte o pino Led1 do módulo P7 ao pino digital 11 do Arduino.
- Conecte o pino Led2 do módulo P7 ao pino digital 12 do Arduino.
- Conecte o pino Led3 do módulo P7 ao pino digital 13 do Arduino.

IMPORTANTE: Não há alterações no sketch (programa).

Passo 2: Programa

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
int digito[8][3] = {
  { LOW,  LOW,  LOW }, // 0
  { LOW,  LOW,  HIGH }, // 1
  { LOW,  HIGH, LOW }, // 2
  { LOW,  HIGH, HIGH }, // 3
  { HIGH, LOW,  LOW }, // 4
  { HIGH, LOW,  HIGH }, // 5
  { HIGH, HIGH, LOW }, // 6
  { HIGH, HIGH, HIGH } // 7
};

int LED1 = 11;
int LED2 = 12;
int LED3 = 13;

int num = 0;

void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
}

void loop() {
  digitalWrite(LED1, digito[num][0]);
  digitalWrite(LED2, digito[num][1]);
  digitalWrite(LED3, digito[num][2]);
  num++;
  if (num > 7)
    num = 0;
  delay (1000);
}
```


Projeto 19 – Contador Binário com Chave Rotativa

O objetivo deste projeto é utilizar três LEDs para mostrar os números entre 0 e 7 no sistema de numeração binário, ou seja 0 (Desligado - LOW) ou 1 (Ligado - HIGH). A chave rotativa (encoder) será utilizada para incrementar ou decrementar o valor binário exibido pelos LEDs.

Material necessário:

- 1 Arduino.
- 3 Resistores de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom) para o LED¹.
- 3 LEDs (qualquer cor)¹.
- 1 Chave Rotativa (Encoder)².
- 1 Protoboard³.
- Jumper cable.

¹ Podem ser substituídos pelo módulo P7-Sensor de Luminosidade da GBK Robotics.

² Podem ser substituídos pelo módulo P17-Encoder da GBK Robotics.

³ Caso os módulos P7 e P17 da GBK Robotics forem usados em conjunto, a protoboard poderá ser substituída.

Passo 1: Montagem do circuito

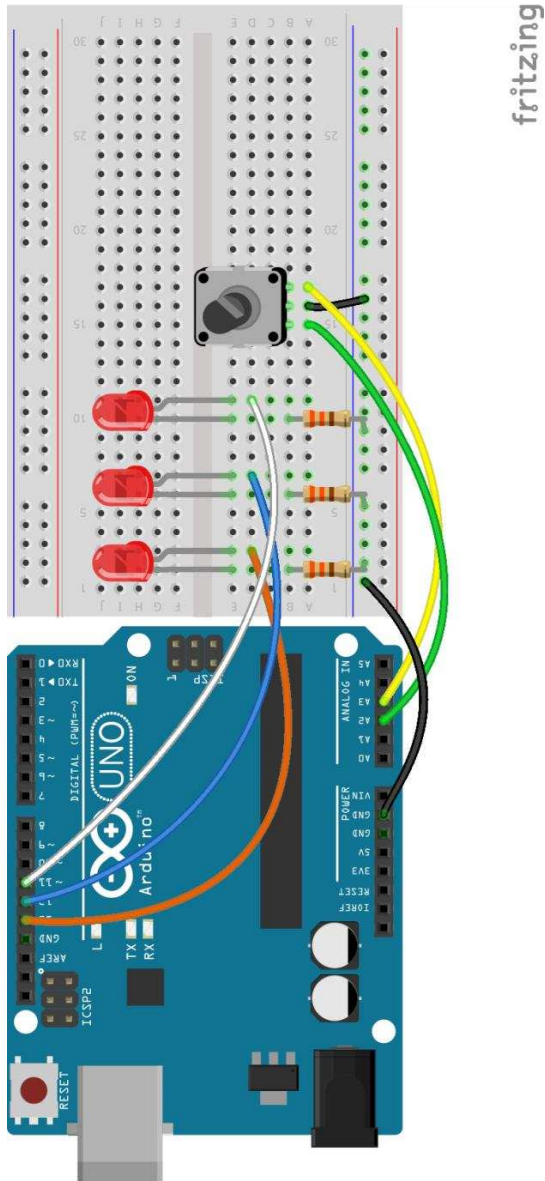


Figura 3.44: Ligações dos LEDs e Chave Rotativa ao Arduino

Conforme mostrado na Figura 3.44 execute a seguinte montagem:

- a. Conecte o pino GND do Arduino à linha de alimentação negativa (preta ou azul) da protoboard.
- b. Coloque os três resistores de 220 ohms (ou 330 ohms) entre a linha de alimentação negativa e qualquer outra linha da protoboard.
- c. Coloque cada um dos três LEDs com o Cátodo (lado chanfrado) conectado a um dos resistores de 220 ohms (ou 330 ohms).
- d. Conecte o Ânodo do primeiro LED ao pino 11 do Arduino.
- e. Conecte o Ânodo do segundo LED ao pino 12 do Arduino.
- f. Conecte o Ânodo do terceiro LED ao pino 13 do Arduino.
- g. Coloque a chave rotativa (encoder) na protoboard e ligue o pino 1 da na entrada analógica A2 do Arduino.
- h. Ligue o pino 2 da chave rotativa (encoder) à linha de alimentação negativa (preta ou azul) da protoboard.
- i. Ligue o pino 3 da chave rotativa (encoder) na entrada analógica A3 do Arduino.

Variação de Montagem 1

Módulo P7-Sensor de Luminosidade da GBK Robotics



Figura 3.45: Módulo P7-Sensor de Luminosidade

Este projeto pode ser montado substituindo os LEDs, os Resistores de 220 ohms (ou 330 ohms) pelo módulo P7-Sensor de Luminosidade (Figura 3.45), neste caso:

- Conecte o pino GND do módulo P7 a um dos pinos de GND do Arduino.
- Conecte o pino Led1 do módulo P7 ao pino 11 do Arduino.
- Conecte o pino Led2 do P7 ao pino 12 do Arduino.
- Conecte o pino Led3 do P7 ao pino 13 do Arduino.
- Coloque a chave rotativa (encoder) na protoboard e ligue o pino 1 na entrada analógica A2 do Arduino.
- Ligue o pino 2 da chave rotativa (encoder) a um dos pinos de GND do Arduino.
- Ligue o pino 3 da chave rotativa (encoder) na entrada analógica A3 do Arduino.

IMPORTANTE: Não há alterações no sketch (programa).

Variação de Montagem 2

Módulo P17-Encoder da GBK Robotics



Figura 3.46: Módulo P17-Encoder

Este projeto pode ser montado substituindo a chave rotativa (encoder) pelo módulo P17-Encoder (Figura 3.46) da GBK Robotics, neste caso:

- Conecte o pino GND do Arduino à linha de alimentação negativa (preta ou azul) da protoboard.
- Coloque os três resistores de 220 ohms (ou 330 ohms) entre a linha de alimentação negativa e qualquer outra linha da protoboard.
- Coloque cada um dos três LEDs com o Cátodo (lado chanfrado) conectado a um dos resistores de 220 ohms (ou 330 ohms).
- Conecte o Ânodo do primeiro LED ao pino 11 do Arduino.
- Conecte o Ânodo do segundo LED ao pino 12 do Arduino.

- f. Conecte o Ânodo do terceiro LED ao pino 13 do Arduino.
- g. Ligue o pino Sinal 1 do módulo P17 na entrada analógica A2 do Arduino.
- h. Ligue o pino GND do módulo P17 a um dos pinos de GND do Arduino.
- i. Ligue o pino Sinal 2 do módulo P17 na entrada analógica A3 do Arduino.

IMPORTANTE: Não há alterações no sketch (programa).

Variação de Montagem 3

Módulo P7-Sensor de Luminosidade e Módulo P17-Encoder da GBK Robotics

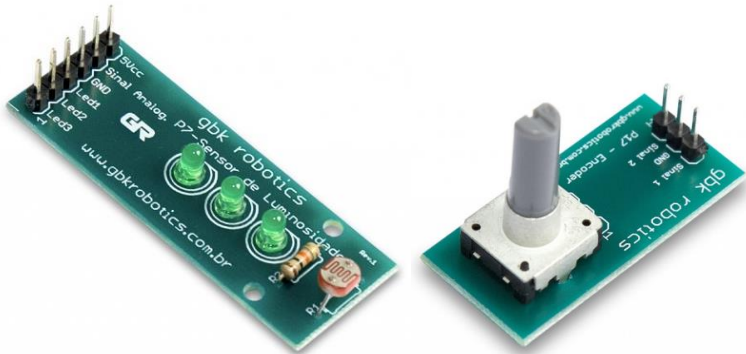


Figura 3.47: Módulos P7 e P17

Este projeto pode ser montado substituindo os LEDs, os Resistores de 220 ohms (ou 330 ohms), a chave rotativa (encoder) e a protoboard pelos módulos P7-Sensor de Luminosidade e P17-Encoder (Figura 3.47), neste caso:

- Conecte o pino GND do módulo P7 a um dos pinos de GND do Arduino.
- Conecte o pino Led1 do módulo P7 ao pino digital 11 do Arduino.
- Conecte o pino Led2 do módulo P7 ao pino digital 12 do Arduino.
- Conecte o pino Led3 do módulo P7 ao pino digital 13 do Arduino.

- e. Ligue o pino Sinal 1 do módulo P17 na entrada analógica A2 do Arduino.
- f. Ligue o pino GND do módulo P17 a um dos pinos de GND do Arduino.
- g. Ligue o pino Sinal 2 do módulo P17 na entrada analógica A3 do Arduino.

IMPORTANTE: Não há alterações no sketch (programa).

Passo 2: Programa

Inicialmente, realize o download da biblioteca RotaryEncoder disponível em <http://vansan.com.br/arduino/RotatoryEncoder.zip>.

Em seguida, descompacte o conteúdo do arquivo RotatoryEncoder.zip na pasta Arduino\libraries que está dentro da pasta Documentos. Depois inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
#include <RotaryEncoder.h>

int digito[8][3] = {
  { LOW,  LOW,  LOW  }, // 0
  { LOW,  LOW,  HIGH }, // 1
  { LOW,  HIGH, LOW  }, // 2
  { LOW,  HIGH, HIGH }, // 3
  { HIGH, LOW,  LOW  }, // 4
  { HIGH, LOW,  HIGH }, // 5
  { HIGH, HIGH, LOW  }, // 6
  { HIGH, HIGH, HIGH }, // 7
};

int LED1 = 11;
int LED2 = 12;
int LED3 = 13;

RotaryEncoder encoder(A2, A3);

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

  // Habilitar a Interrupção 1 (Pin Change) para
  // as entradas analógicas.
  PCICR |= (1 << PCIE1);
```

```
// Habilita a interrupção para os pinos
// analógicos 2 e 3.
PCMSK1 |= (1 << PCINT10) | (1 << PCINT11);

pinMode(LED1, OUTPUT);
pinMode(LED2, OUTPUT);
pinMode(LED3, OUTPUT);
}

// Rotina do Serviço de Interrupção 1 (Pin
// Change)
// Esta rotina apenas é chamada quando ocorre
// uma mudança de sinal nos pinos A2 e A3
ISR(PCINT1_vect) {
    encoder.tick();
}

void loop() {
    static int pos = 0;
    int novaPos = encoder.getPosition();
    if (pos != novaPos) {
        Serial.println(novaPos);
        pos = novaPos;
        if (pos >= 0 && pos <= 7) {
            digitalWrite(LED1, digito[pos][0]);
            digitalWrite(LED2, digito[pos][1]);
            digitalWrite(LED3, digito[pos][2]);
        }
    }
}
```

Desafio 6 – Contador Hexadecimal

Unindo os conceitos abordados no Projeto 12: Display de LED e no Projeto 19: Chave Rotativa (Encoder) elaborar um contador hexadecimal (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) de modo que o dígito exibido no display de led de 7 segmentos seja incrementado ou decrementado conforme a variação de posição da chave rotativa (encoder).

Projeto 20 – Utilizando Entradas Analógicas como Portas Digitais

Neste projeto será mostrado um recurso muito útil no Arduino e um pouco fora dos padrões das características divulgadas da placa, que é usar as entradas analógicas (Analog In) como portas digitais. Esta situação é muito prática quando precisamos de mais portas digitais do que as 14 que o Arduino nos oferece. Para o Arduino Uno, utilize os seguintes números para cada um dos terminais analógicos: A0: 14, A1: 15, A2: 16, A3: 17, A4: 18 e A5: 19.

Material necessário:

- 1 Arduino.
- 1 Protoboard.
- Jumper cable.
- Resistores de 220 ohms à um 1k ohms.
- Leds (qualquer cor).
- 1 Resistor de 10k ohms (marrom, preto laranja) para o botão.
- 1 Botão.

Montagem do Circuito N° 1

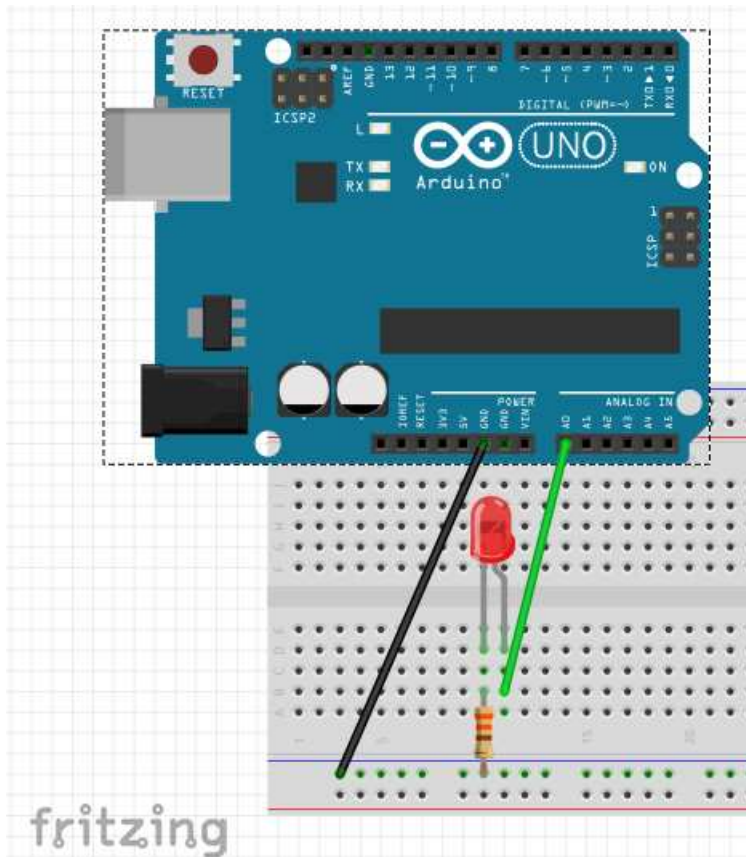


Figura 3.48: LED Conectado ao Pino A0 (ou 14) do Arduino

Adotando, como referência, a Figura 3.48 realize a montagem do circuito que será usado neste projeto.

Programa N ° 1

No ambiente de desenvolvimento do Arduino digite o seguinte programa:

```
int LED = A0; // A0 ou 14

// Tempo que o LED ficará aceso ou apagado.
int espera = 500;

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

void loop() {
  digitalWrite(LED, HIGH);
  delay(espera);
  digitalWrite(LED, LOW);
  delay(espera);
}
```

Observe que na variável LED colocamos o valor A0, mas também podemos usar o valor 14.

Programa N° 2

Digite o seguinte programa no ambiente de desenvolvimento do Arduino:

```
int LED = A0; // A0 ou 14
int BOTAO = A3; // A3 ou 17

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(BOTAO, INPUT);
  Serial.begin(9600);
}
void loop() {
  // Obtém o estado do Botão
  int estado = digitalRead(BOTAO);
  Serial.print(estado);
  if (estado==HIGH) {
    digitalWrite(LED, HIGH);
    Serial.println(" - Led Ligado");
  }
  else {
    digitalWrite(LED, LOW);
    Serial.println(" - Led Desligado");
  }
  delay(100);
}
```

Observe neste programa que a variável BOTAO está com o valor A3 e na função setup foi declarada como INPUT, ou seja uma entrada digital. Se habilitarmos o monitor serial poderemos verificar o valor da variável estado e uma mensagem informativa indicando se o LED está aceso ou apagado.

Projeto 21 – Utilizando INPUT_PULLUP

Neste projeto vamos utilizar um recurso bastante versátil e pouco conhecido, o Arduino apresenta resistores internos para as entradas digitais, desta forma não precisaremos utilizar resistores externos para a ligação de botões e outros sensores, assim economizamos componentes nas montagens e projetos, criando projetos mais simples. Esta funcionalidade pode ser ativada via software, na função `pinMode()` onde ao invés de declarar um botão como `INPUT`, colocaremos `INPUT_PULLUP`.

Material necessário:

- 1 Arduino.
- 1 Protoboard.
- Jumper cable.
- Resistores de 220 ohms à um 1k ohms para os Leds.
- Leds (qualquer cor).
- 1 Botão.

Montagem do Circuito

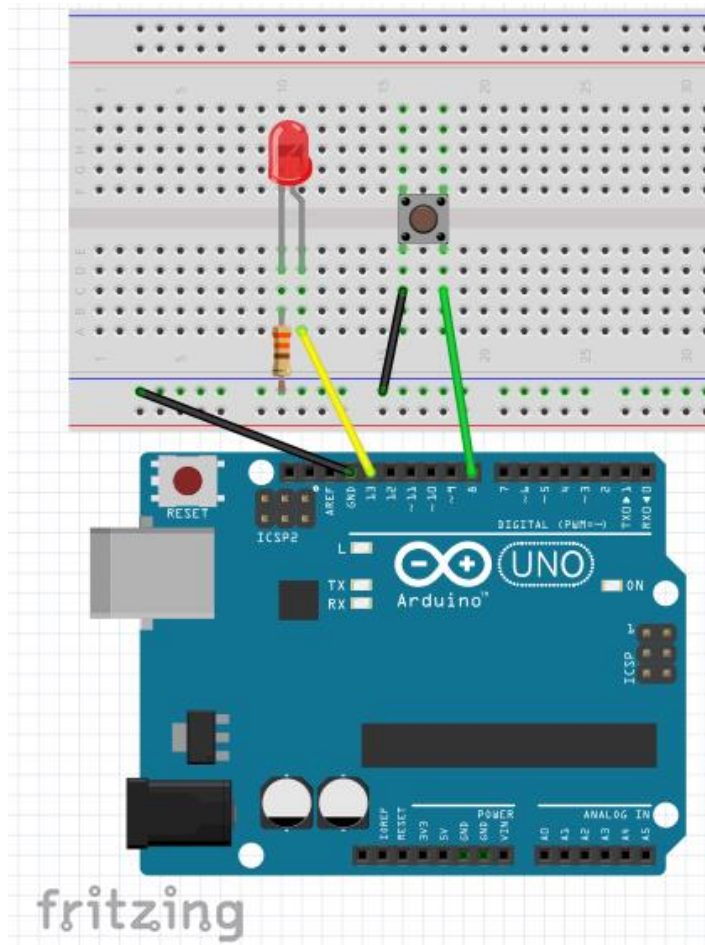


Figura 3.50: Ligação do LED e do Botão ao Arduino

Adotando como referência a Figura 3.50 realize a montagem do circuito que será usado neste projeto.

Programa N° 1

No ambiente de desenvolvimento do Arduino digite o seguinte sketch:

```
int LED = 13;
int BOTA0 = 8;

void setup() {
  pinMode(LED, OUTPUT);

  //Habilitar os resistores internos
  pinMode(BOTA0, INPUT_PULLUP);

  Serial.begin(9600);
}

void loop() {
  // Obtém o estado do botão
  int estado = digitalRead(BOTA0);

  Serial.print("Valor da variável estado: ");
  Serial.print(estado);
  if (estado == LOW) {
    digitalWrite(LED, HIGH);
    Serial.println(" - Led Ligado");
  }
  else {
    digitalWrite(LED, LOW);
    Serial.println(" - Led Desligado");
  }
  delay(100);
}
```

Utilizando o monitor Serial, teremos a seguinte saída, sem pressionar o botão:

<i>Valor da variável estado: 1 - Led Desligado</i>
--

Quando pressionar o botão:

<i>Valor da variável estado: 0 - Led Ligado</i>

Isto ocorre pois quando não estivermos pressionando o botão, o resistor interno do micro controlador, manterá a entrada em 1 ou HIGH e ao pressionar, estaremos ligando o GND ou 0 diretamente na porta, estamos aterrando a porta, e desta forma a variável estadoBotao receberá o valor 0 ou LOW.

Projeto 22 – Sensor de Presença

Neste projeto apresentamos o sensor de presença (Figura 3.51), este tipo de sensor utiliza infravermelho para detectar algum movimento, no módulo pode-se ajustar a sensibilidade e o tempo que o sinal será enviado ao Arduino, ao se detectar algum movimento o sensor envia o sinal 1 (HIGH) para o Arduino.



Figura 3.51: Sensor de Presença (PIR)

Material necessário:

- 1 Arduino.
- 1 Protoboard.
- Jumper cable.
- Resistores de 220 ohms à um 1k ohms para os LEDs.
- Leds (qualquer cor).
- 1 Modulo Sensor de Movimento Presença (PIR).

Montagem do Circuito

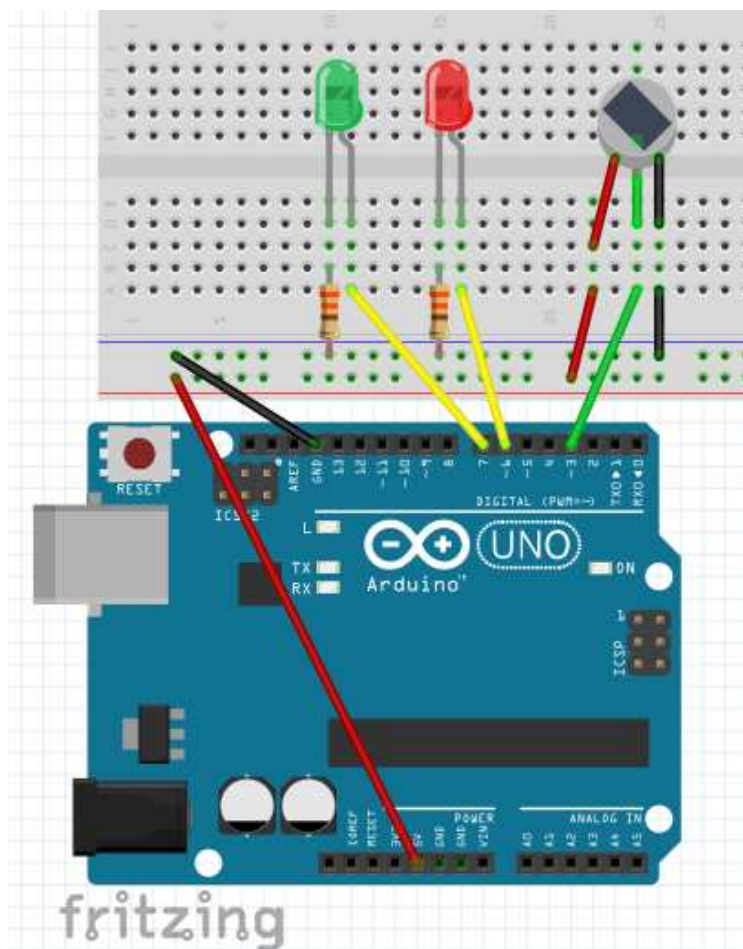


Figura 3.52: Ligação do Sensor e Outros Componentes ao Arduino

Adotando como referência a Figura 3.52 realize a montagem do circuito que será usado neste projeto.

Programa

Implemente o programa a seguir no ambiente de desenvolvimento do Arduino.

```
int ledMovimento = 6;
int ledParado = 7;
int pinoPIR = 3; //Pino ligado ao sensor PIR
int acionamento;

void setup() {
  pinMode(ledMovimento, OUTPUT);
  pinMode(ledParado, OUTPUT);
  // Define pino onde o sensor foi ligado como
  // entrada
  pinMode(pinoPIR, INPUT);
  Serial.begin(9600);
}

void loop() {
  acionamento = digitalRead(pinoPIR);
  Serial.print("Acionamento: ");
  Serial.print(acionamento);
  if (acionamento == LOW) {
    // Sem movimento, LED verde é ligado
    digitalWrite(ledMovimento, LOW);
    digitalWrite(ledParado, HIGH);
    Serial.println(" - Sem Movimento");
  }
  else {
    // Em caso de movimento, ligar o LED
    // vermelho
    digitalWrite(ledMovimento, HIGH);
    digitalWrite(ledParado, LOW);
    Serial.println(" - Movimento Detectado");
  }
}
```

Desafio 7 – Sensor de Presença com Temporizador

Neste desafio utilize os conhecimentos adquiridos no Projeto 22 - Sensor de Presença, Projeto 21 - Utilizando Entradas com INPUT_PULLUP e Projeto 3 - LDR para criar um sensor de presença que acenda uma lâmpada de uma área externa que será representando por um LED, onde este LED apenas acenderá quando estiver escuro e for detectado algum movimento ou for pressionado um botão. O LED deverá ficar acesso por 10 segundos antes de desligar.

Projeto 23 – LED RGB

O objetivo deste projeto é utilizar três portas PWM do Arduino para determinar a intensidade de acendimento de cada uma das cores (vermelho, verde e azul) de um LED RGB.

Material necessário:

- 1 Arduino.
- 3 Resistores de 220 ohms (vermelho, vermelho, marrom) ou 330 ohms (laranja, laranja, marrom).
- 1 Led RGB.
- 1 Protoboard.
- Jumper cable.

Passo 1: Montagem do circuito

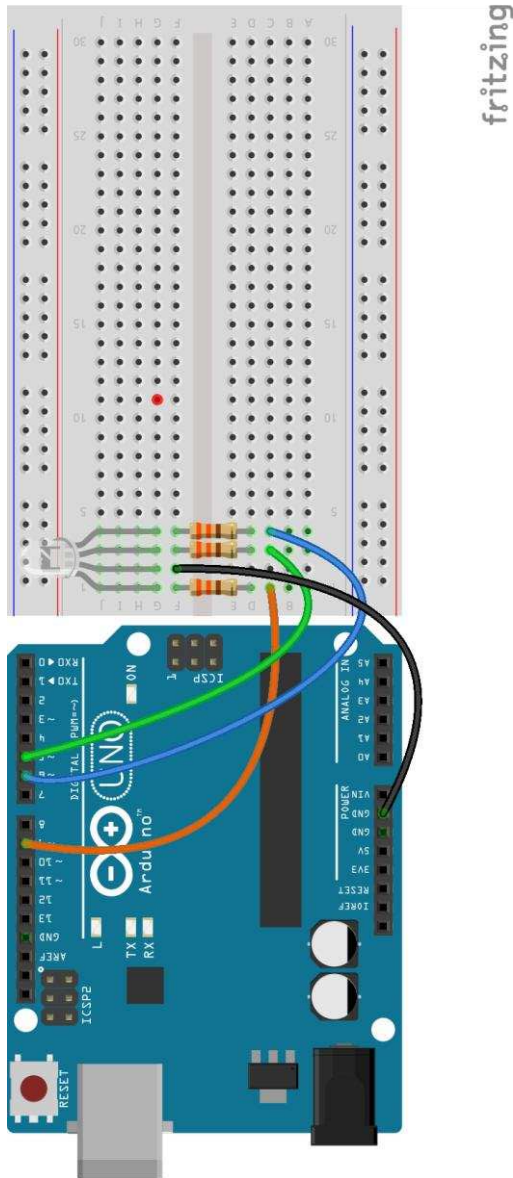


Figura 3.53: Ligação do LED RGB ao Arduino

A montagem do circuito deve ser realizada conforme ilustra a Figura 3.53.

Passo 2: Programa

Inicie o ambiente de desenvolvimento do Arduino e digite o sketch (programa) a seguir:

```
// LED RGB
int VERMELHO = 9, AZUL = 6, VERDE = 5;

int iVermelho, iVerde, iAzul;

void setup() {
  pinMode (VERMELHO, OUTPUT);
  pinMode (VERDE, OUTPUT);
  pinMode (AZUL, OUTPUT);
}

void loop() {
  iVermelho = random(256);
  iVerde = random(256);
  iAzul = random(256);
  analogWrite(VERMELHO, iVermelho);
  analogWrite(VERDE, iVerde);
  analogWrite(AZUL, iAzul);
  delay(500);
}
```



Referências Bibliográficas

ARDUINO. **Language Reference**. Disponível em: <<https://www.arduino.cc/reference/en/>>. Acesso em 09/08/2018.

OLIVEIRA, C.; ZANETTI, H. **Arduino descomplicado: como elaborar projetos de eletrônica**. São Paulo: Érica, 2015.

_____. **Arduino descomplicado: aprenda com projetos de eletrônica e programação**. São Paulo: Érica/Saraiva, 2017.

_____. **Arduino simples e divertido: como elaborar projetos de eletrônica**. Salvador: Asè Editorial, 2016.

