Bernardo Victoria Escobar Silva
Josiney de Souza
Leonardo Felipe de Avila Calbusch
Otavio Henrique da Silva
Tiago Rafael de Almeida Alves
(organizadores)

IFMAKER IFC CAMPUS BRUSQUE:

INICIANDO OS TRABALHOS COM IDENTIFICAÇÃO E USO DOS COMPONENTES COMPUTACIONAIS DO LABORATÓRIO MAKER

editora IFC

Bernardo Victoria Escobar Silva
Josiney de Souza
Leonardo Felipe de Avila Calbusch
Otavio Henrique da Silva
Tiago Rafael de Almeida Alves
(organizadores)

IFMAKER IFC CAMPUS BRUSQUE:

INICIANDO OS TRABALHOS COM IDENTIFICAÇÃO E USO DOS COMPONENTES COMPUTACIONAIS DO LABORATÓRIO MAKER

IFC
Blumenau, 2022

CONTATO:
Rua das Missões, nº 100 – Ponta Aguda – Blumenau/SC – CEP: 89.051-000
Fone: (47) 3331-7850 | E-mail: editora@ifc.edu.br

PROJETO GRÁFICO
Paolo Malorgio Studio Ltda

DIAGRAMAÇÃO
Paolo Malorgio Studio Ltda

REVISÃO TEXTUAL
Alessandra Klein

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

IFMAKER IFC [livro eletrônico] : Campus Brusque : iniciando os trabalhos com identificação e uso dos componentes computacionais do laboratório maker / organização Bernardo Victoria Escobar Silva...[et al.]. -- Blumenau, SC : Editora do Instituto Federal Catarinense, 2022.
PDF

Outros autores.
Outros organizadores: Josiney de Souza, Leonardo Felipe de Avila Calbusch, Otavio Henrique da Silva, Tiago Rafael de Almeida Alves.
Bibliografia.
ISBN 978-65-88089-16-3

1. Arduino (Linguagem de programação para computadores) 2. Hardware 3. Robótica 4. Sistemas operacionais (Computadores) 5. Trabalho integrado I. Silva, Bernardo Victoria Escobar. II. Souza, Josiney de. III. Calbusch, Leonardo Felipe de Avila. IV. Silva, Otavio Henrique da. V. Alves, Tiago Rafael de Almeida.

22-129922 CDD-005.43

Índices para catálogo sistemático:

1. Sistemas operacionais : Computadores : Processamento de dados 005.43

Eliete Marques da Silva - Bibliotecária - CRB-8/9380

Sumário

1 INTRODUÇÃO

Josiney de Souza - ⟨josiney.souza@ifc.edu.br⟩

Tudo começou em 2018 quando fui para o Fórum Internacional de Software Livre (FISL), em Porto Alegre, apresentar um trabalho chamado "Claquete". Esse é um sistema de scripts (por isso claquete - para lembrar os roteiros/*scripts* de filmes - estava sem ideias) em linguagem BASH que criei para ajudar na criação e envio por e-mail de certificados de eventos realizados no *campus*. Caso tenha se interessado, pode acessar os arquivos em: ⟨https://github.com/josiney-souza/claquete⟩. Mas já alerto que terá que descobrir tudo sozinho, pois coloquei tudo em um arquivo compactado e não coloquei mais detalhes sobre ele. Mas quem conhece BASH vai se dar bem. E eis que se começa a saga...

1.1 Labrador

Nesse evento, enquanto não chegava o momento da minha apresentação, assistia a outras palestras. E, em uma dessas palestras que fui apresentado a um computador de placa única (*Single Board Computer* - SBC) chamado Labrador. Como assim um computador com nome de cachorro? Pensando bem, por que não? Já existem os *Beaglebone* mesmo...

Então me despertou um interesse maior quando soube de que se tratava de uma fabricante brasileira (Caninos Loucos), alocada na USP e de fabricação nacional. Sempre ouvimos "vamos prestigiar a produção nacional" e, se tratando de uma novidade, pensei: e se eu conseguisse uma placa dessas para o *campus*? Está certo que é tudo novidade mas podemos ser um dos primeiros a fazer esses testes.

Pelo volume de trabalho, acabei me esquecendo dessa placa em 2018. Porém, em 2019, quando estávamos pensando em compras para o *campus* para uso em 2020, me lembrei dessa placa. Fui atrás de informações e descobri que a empresa Robocore era a responsável pela distribuição da placa. Apesar de estarem em estágio de testes, ou seja, ainda não era possível comprar o equipamento, poderíamos receber o equipamento gratuitamente em nosso *campus* para 50 dias de testes. Escrevi a eles, solicitei o equipamento e o recebi nos primeiros dias março de 2020, como na Figura 1.

Ao receber a placa, fui contar a novidade aos servidores. Infelizmente, parece que ninguém se animou tanto quanto eu. Pensei: vou contar aos alunos; mesmo porque alguns alunos estavam no momento de decidir seus temas para o Trabalho de Conclusão de Curso (TCC) na graduação de Redes de Computadores. Novamente, sem muito interesse.

Figura 1 – Placa Labrador v2

Fonte: Arquivo pessoal

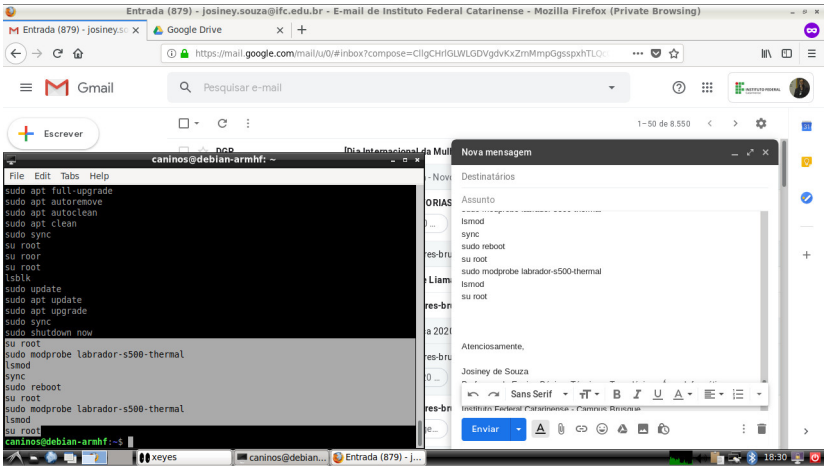
Então eu mesmo fiz alguns poucos testes com a placa mas, sem muito auxílio, não consegui avançar muito além de:

- instalar um sistema operacional em um cartão de memória;
- projetar a saída em um monitor via HDMI;
- instalar alguns programas/software;
- tentar instalar o RetroPie (⟨https://retropie.org.uk/⟩) e tentar consertar os erros que impediram a completa instalação;
- tirar algumas capturas de tela como a Figura 2.

Até pensei em criar um livro ou e-book para relatar a experiência, como se fosse um diário dos 50 dias de testes, mas sem ajuda (e estava sem) não conseguiria fazê-lo. Para piorar, depois veio a pandemia de COVID-19 e então muito do ensino, da pesquisa e da extensão ficou travado por um bom tempo, de modo que tive que devolver a placa sem muitos avanços. Mas não desisti.

Desde então, fiquei com a pulga atrás da orelha: por que ninguém mais se interessou? Será que esse é um assunto interessante? Como poderíamos ter mais adeptos? Será que poderia ser medo? Será que podemos fazer algo para que não percamos mais importunidades únicas como essa? Assim entramos em uma nova etapa.

Figura 2 – Captura de tela do ambiente Debian instalado na placa Labrador



Fonte: Arquivo pessoal

1.2 Hora das compras

Dizem que são as perguntas que movem os pesquisadores. As perguntas acima realmente me fizeram pensar e tentar agir para mudar a realidade onde estou. Foi a partir disso que pensei em conciliar um momento da nossa instituição de curricularizar a pesquisa e a extensão nos planos de ensino das disciplinas. Essa poderia ser a oportunidade de angariar mais interessados para esse mundo.

Além do mais, podemos ver como o mundo atualmente se interessa por essa cultura do "faça você mesmo": hoje temos exemplos no nosso cotidiano (com drones e sensores em todos os tipos de equipamentos, como carros) e até programas de conhecimento e entretenimento (exemplo: Conexão *Maker* e Batalha *Makers* Brasil). É justo que uma instituição de ensino, principalmente uma que se propõe a trabalhar na área de informática, comece a trabalhar e ter esses conhecimentos também.

A verdade é que o *campus* já possuía algum conhecimento nesse universo: em 2020 mesmo, durante a pandemia, a reitoria doou impressoras 3D ao *campus* para auxiliar na confecção de máscaras protetoras faciais do tipo *face shield* (⟨https://brusque.portaldacidade.com/noticias/saude/ifc-brusque-produzira-protetores-faciais-para-doacao-1519⟩). Contudo, devido ao distanciamento social, nem todos os servidores tiveram acesso ou manuseio dessa tecnologia; ainda era necessário criar um plano para envolver os docentes de informática.

Iniciei tentando realizar a compra da placa Labrador, que estava às vésperas de ser lançada, bem como de vários outros componentes do tipo *maker* por meio do plano anual de compras do *campus*. Tínhamos um quantitativo reduzido de componentes eletrônicos em nosso laboratório de hardware que não permitia desenvolver projetos com os alunos.

Pelo esforço realizado, no começo de 2021 conseguimos receber diversos equipamentos e componentes *maker*. Abaixo há a lista de todas nossas aquisições:

- Arduino Uno R3
- Módulo *Bluetooth* HC-06
- *Protoboard* (1280 pontos e 170 pontos)
- *Raspberry* Pi W
- Shield Ethernet
- *Raspberry* PI 3 Model B
- Cartão Micro SD 16 GB
- Fonte chaveada para *Raspberry*
- Cabo HDMI
- Case *Raspberry* Pi 3
- Módulo *wireless*
- Módulo giroscópio
- Módulo relé
- Adaptador ESP8266
- Módulo sensor som
- Sensor movimento
- Fonte chaveada para Arduino
- Controle remoto IR
- Bateria CR2032
- Motor de passo 5V
- Módulo *drive* motor de passo
- Motor DC 5V
- *Display LCD* 16x2

- *LED* emissor Infravermelho
- *LED* receptor Infravermelho
- Fio *jumper* macho-macho
- Fio *jumper* macho-fêmea
- Potenciômetro
- Resistor carbono
- Resistor carbono
- Transistor tip120
- Diodo retificador
- Kit Makey-Makey
- Sensor pressão
- Módulo sensor PH (Sensor de pH Arduino + Módulo de Leitura)
- *Protoboard* 840
- Fios *Jumper* Macho-Macho de 20 cm
- *Display de LCD* 16x2 com backlight
- HC - SR04 - Sensor Ultrassom
- *LED* alto brilho
- *LEDs* Amarelos
- *LEDs* Verdes
- *LEDs* Vermelhos
- Chave Momentânea (PushButton)
- Resistores 10k
- Resistores 470
- Potenciômetro 10k

- Sensor de Luminosidade (LDR 5mm)
- *Buzzer*
- Sensor de Temperatura NTC 10K 3mm
- Caixa Organizadora
- Sensor de chuva
- Sensor Capacitivo de Umidade do Solo
- Sensor de Gás MQ-135 para Gases Tóxicos
- Sensor de Umidade do Solo Higrômetro
- Módulo TM1637 com *Display* 7 Segmentos 4 Dígitos
- Cabos garra de jacaré
- *Case* Arduino
- *Jumper* macho-fêmea
- Sensor de nível de água
- Sensor DHT22
- *Display* 7 segmentos anodo
- Módulo ESP01S
- Módulo NRF24L01
- HC - SR04 - Sensor Ultrassom
- Sensor DHT11
- Matriz de *LEDs*
- Teclado matricial 16 teclas

Faltava então a aplicação e uso desses componentes. Assim, aproveitando o momento de possibilidade de se conseguir bolsas de projeto, montei o projeto de pesquisa intitulado "IF-maker IFC *campus* Brusque: iniciando os trabalhos com identificação e uso dos componentes computacionais do laboratório *maker*" e submeti ao edital de bolsas de pesquisa do *campus*. E assim se iniciou mais uma etapa.

1.3 Projeto de pesquisa

Fomos contemplados com uma bolsa discente para o projeto. Se já tínhamos os equipamentos, o bolsista e estudantes que colaborariam, só falta ligar todos esses fatores a favor da execução. Aproveitei o momento de início de ano letivo de 2021, o qual pudemos compartilhar experiências exitosas de projetos e métodos de trabalhos integrado nas disciplinas com outros *campi*, para apresentar minha proposta de trabalho aos demais colegas do *campus*. Mais uma vez, ao menos naquele momento, não houve interessados.

Independente de se ter mais colegas servidores, estava decidido a iniciar esse trabalho de pesquisa integrado ao ensino. O desejo também era de um legado de conhecimento para a comunidade interna e externa disponibilizando sugestões de projetos possíveis de serem estudados e replicado no *campus*. E digo mais: nesse projeto, acreditei que pudesse contemplar o tripé trabalhado nos Institutos Federais (ensino-pesquisa-extensão), por entender que conseguiria associar as dimensões a seguir:

Ensino: o projeto faria/faz parte da disciplina de Hardware e Sistemas Operacionais do curso Técnico em Informática Integrado ao Ensino Médio. Dentro da disciplina, os estudantes poderiam pesquisar sobre os componentes *maker* adquiridos, estudá-los e propor projetos de uso. Isso contaria como trabalhos e notas da disciplina. Ainda, tanto o professor e bolsista estariam à disposição dos demais estudantes da disciplina como mediadores/orientadores/monitores;

Pesquisa: conhecer os componentes e suas interações, pensar em projetos possíveis, dimensionar cronogramas, se organizar para cumprir os objetivos, escrever os resultados nas normas vigentes (de português e de escrita de trabalhos)... todas essas atividades a serem desenvolvidas contemplam o método científico propiciado pelo trabalho integrado;

Extensão: como resultado das demais dimensões, o produto final seria este livro que pode atender as demandas locais bem como demandas de outros locais e instituições. Além desse produto, os estudantes também realizaram apresentações no *YouTube* mostrando os resultados pesquisados, ficando assim as apresentações gravadas e disponíveis na Internet através do canal do *campus*.

Vale dizer que os estudantes se organizaram em grupos e puderam escolher com que componentes eletrônicos desejariam trabalhar. E suas apresentações no *YouTube* foram em formato de *pitch*: foram 5 minutos de apresentação ininterrupta e mais 5 minutos de perguntas e respostas dos telespectadores. Mais uma vez, os estudantes puderam ter contato com formatos usados no mundo do trabalho, também foco dos Institutos Federais.

1.4 Resultados

Este projeto rendeu alguns resultados aos seus membros e demais envolvidos:

Anais de evento 1: o projeto participou da IV Semana de Formação Acadêmica e Científica e Cultural e Humanística e... (FACCHU) do IFC *campus* Brusque em 2021 (⟨http://brusque.ifc.edu.br/facchu/⟩) apresentando seu trabalho desenvolvido (⟨https://youtu.be/uSv3n-ZUyxg?t=966⟩) e publicou texto nos anais desse evento (⟨http://brusque.ifc.edu.br/facchu/facchu-anais/⟩);

Anais de evento 2: a partir deste projeto, um projeto derivado e integrado a este, na categoria de projeto de ensino integrado à pesquisa, denominado de "Conexões da Física com a cultura *maker*" foi apresentado na XIV Mostra Nacional de Iniciação Científica e Tecnológica Interdisciplinar (MICTI)/ III Epromundo (⟨https://eventos.ifc.edu.br/micti2021/⟩) do IFC (vídeo em ⟨https://youtu.be/MKBVMclh83c?t=223⟩) e teve texto publicado nos anais desse evento (⟨https://publicacoes.ifc.edu.br/index.php/micti/article/view/2626⟩);

Apresentações no *YouTube*: além das apresentações realizadas pelos estudantes no canal do *campus* no *YouTube* (⟨https://www.youtube.com/watch?v=HF8c3yjI0D0⟩ com 127 visualizações, ⟨https://www.youtube.com/watch?v=DzjiyOzZOTs⟩ com 87 visualizações e ⟨https://www.youtube.com/watch?v=iqQwer30uSY⟩ com 82 visualizações), este projeto foi convidado a participar do evento on-line Semana de Empreendedorismo e Inovação - Inova Brusque e região e o 2º Hackathon Shift Indústria 4.0 reapresentando seu método de trabalho (⟨https://www.youtube.com/watch?v=L8bw4uWKxsI⟩ com 122 visualizações);

Livro: este produto final que ficará disponível tanto interna quanto externamente;

Premiação 1: também na XIV MICTI, o projeto "Conexões da Física com a cultura *maker*" (derivado e integrado a este projeto) configurou como trabalho destaque na categoria "criatividade";

Premiação 2: além da participação na MICTI, o coordenador Josiney foi premiado como destaque pelo fomento e interação no 3º Prêmio Comunidade Hackathon Shift pelo trabalho apresentado na Semana de Empreendedorismo e Inovação - Inova Brusque e região e o 2º Hackathon Shift Indústria 4.0.

1.5 Próximos capítulos

Agora que a história do projeto e deste livro foi contada, o próximo capítulo apresentará conceitos importantes para entendimento dos trabalhos, bem como os demais capítulos apresentarão os resultados das pesquisas desenvolvidas pelas equipes de estudantes e demais membros do projeto de pesquisa. Aproveite a jornada.

2 CONCEITOS IMPORTANTES

Josiney de Souza - ⟨josiney.souza@ifc.edu.br⟩
Otavio Henrique da Silva - ⟨otaviomdie@gmail.com⟩

Antes de ver os projetos vamos ver o que são alguns dos componentes.

2.1 Arduino

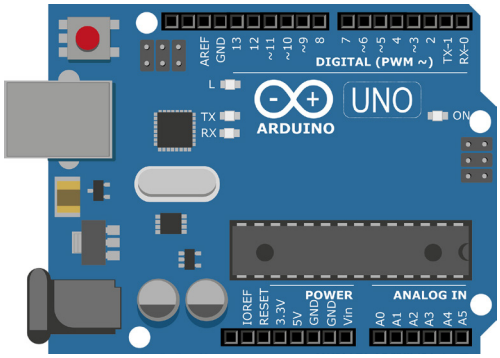
É uma plataforma de prototipagem eletrônica de hardware livre e de placa única. Ela se baseia na linguagem de programação C/C++. Essa placa é geralmente usada para criar projetos acessíveis, com baixo custo e fácil de ser usar tanto para iniciante quanto para profissionais. Na Figura 3, o símbolo do Arduino:

Figura 3 – Símbolo do Arduino

Fonte: ⟨https://www.arduino.cc/⟩

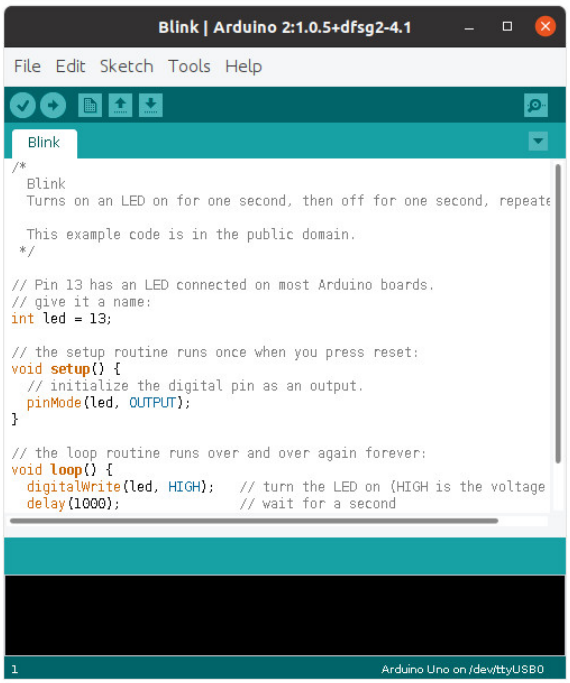
Ele não pode ser usado como um computador como o *Raspberry* Pi, mas pode ser usado para automatização, robótica, para ajudar as pessoas no seu dia a dia e entre outros. O conjunto Arduino possui basicamente duas partes: o hardware (Figura 4), placa onde estão todos os circuitos integrados; e o software (Figura 5), um ambiente integrado de desenvolvimento (*IDE - Integrated Developer Environment*) para se desenvolver os *sketches* - programas desenvolvidos para se executar no Arduino.

Figura 4 – Uma das placas Arduino - versão Uno



Fonte: ⟨https://pixabay.com/pt/illustrations/arduino-arduino-uno-tecnologia-2168193/⟩

Figura 5 – Ambiente Integrado de Desenvolvimento - IDE - do Arduino



Fonte: Reprodução/Próprio autor

Ele foi criado em 2005 por 5 pesquisadores com o objetivo de fazer um produto barato e ao mesmo tempo funcional e fácil de programar, sendo dessa forma acessível para estudantes e projetistas amadores.

2.2 *Raspberry* Pi

O *Raspberry* Pi é quase igual o Arduino: é uma placa única, mas com a diferença que o *Raspberry* Pi pode ser utilizado como um computador, pode-se instalar um sistema operacional no *Raspberry* Pi. Geralmente o *Raspberry* é usado para robótica, usar emuladores de consoles antigos, e outros projetos muitos legais. Na Figura 6, o símbolo do *Raspberry* Pi:

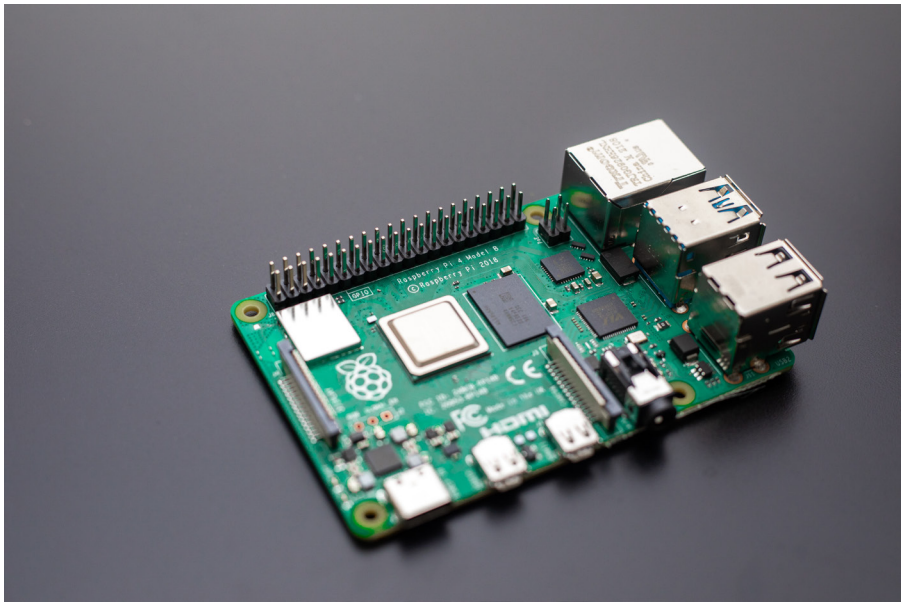
Figura 6 – Símbolo do *Raspberry* Pi

Fonte: ⟨https://www.raspberrypi.org/⟩

O *Raspberry* Pi não tem HD interno e externo; ele usa como armazenamento cartão de

memória Micro SD. Também não tem mouse, teclado e monitor; podendo ser usado teclados e mouses comuns com entrada USB e monitores com entrada HDMI. Na Figura 7, uma das placas *Raspberry* Pi:

Figura 7 – Uma das placas *Raspberry* Pi - modelo 4B



Fonte: ⟨https://unsplash.com/photos/BIHgNEaM394?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

Ele foi criado pela *Raspberry* Pi *Foundation* que começou a vender eles em 2012 por US$ 35,00 (35 dólares).

2.3 Makey Makey

É um kit de interação criativa, com ele você pode transformar objetos em comandos como por exemplo um piano usando frutas ou ser um controle de um jogo. Ele é utilizado para ensinar programação a crianças de uma forma divertida, ele é usado também em robótica. Na Figura 8, o símbolo do Makey Makey:

Figura 8 – Símbolo do Makey Makey

Fonte: ⟨https://makeymakey.com/⟩

Makey Makey foi feito por dois estudantes do Media Lab do MIT (Massachusetts Institute of Technology) e contaram com a orientação de Mitch Resnick, que tem grande contribuição no campo da educação tecnológica para crianças. Na Figura 9, o kit básico do Makey Makey:

Figura 9 – Kit Makey Makey

Fonte: ⟨https://makeymakey.com/⟩

2.4 Fritzing

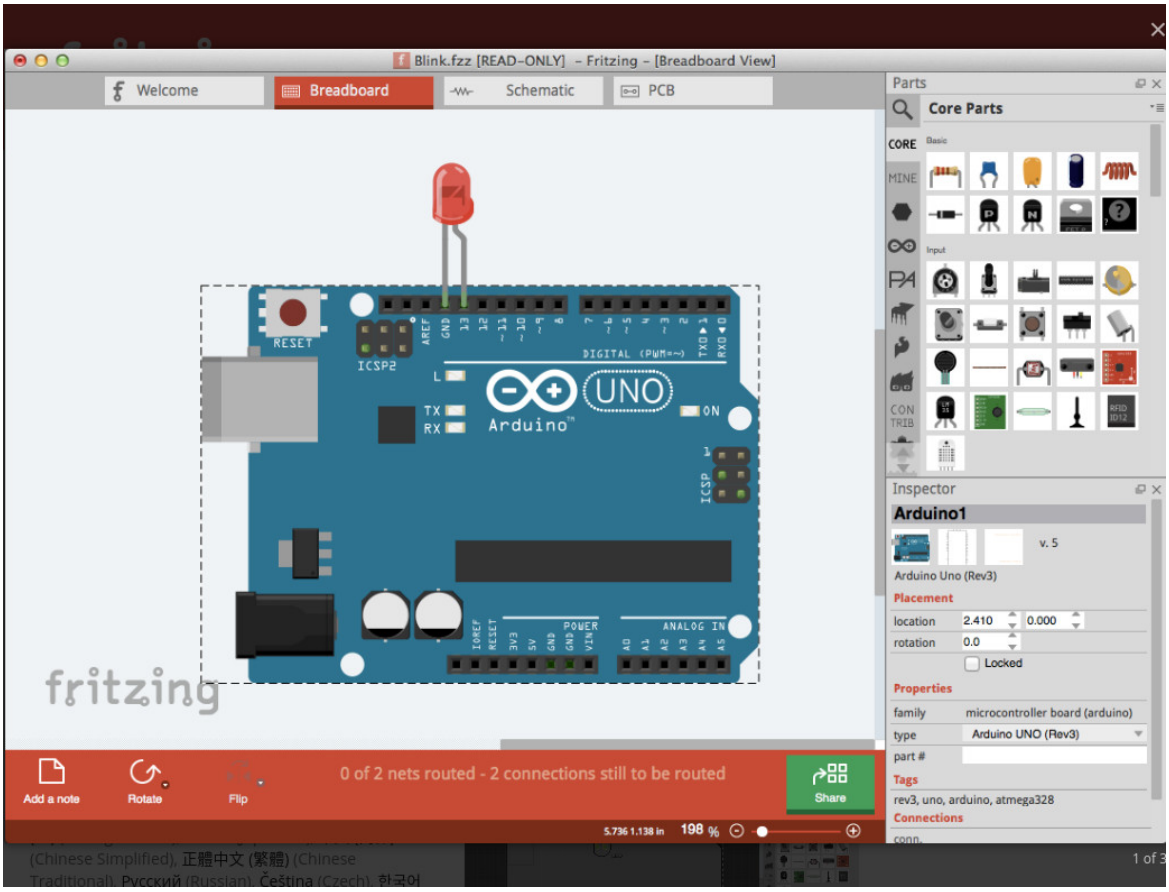
É um programa usado para criar protótipos, podendo ser usado para criar protótipos de montagens para o Arduino. Esse programa serve para as pessoas que estão fazendo projetos vejam como vai ser a ligação de cada cabo em 2D para fazer o projeto funcionar para outras pessoas. Na Figura 10, o símbolo do Fritzing:

Figura 10 – Símbolo do Fritzing

Fonte: ⟨https://fritzing.org/⟩

Esse programa foi criado na Alemanha na Universidade de Ciências Aplicadas de Potsdam, que agora é mantida pela Friends-of-Fritzing Foundation. Na Figura 11, é mostrada uma captura de tela de uso do programa:

Figura 11 – Exemplo de tela da ferramenta Fritzing



Fonte: ⟨https://fritzing.org/⟩

É importante mencionar que o Fritzing é um programa apenas para modelagens e criação de protótipos virtuais; ele não possui suporte a execução dos códigos-fontes (*sketches*) construídos.

2.5 Tinkercad

O Tinkercad é um programa de modelagem 3D gratuito que roda no navegador e conhecido por sua facilidade e simplicidade de usar. Ele serve também para criar um modelo conceitual de ligação de cabos para mostrar onde cada cabo fica ligado, assim como o Fritzing; além também ser utilizado para fazer modelagem em 3D. Na Figura 12, o símbolo do Tinkercad:

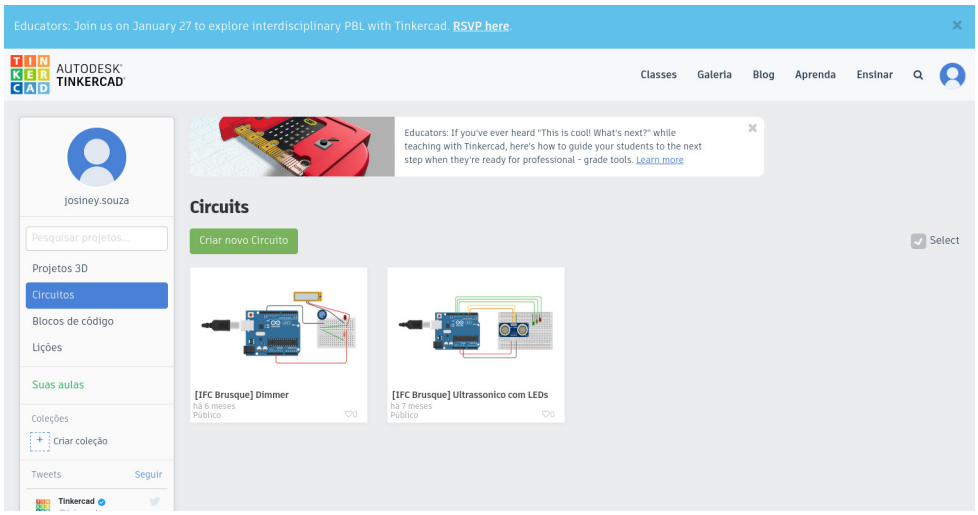
Ele foi criado por um ex-engenheiro do Google, Kai Backman, e seu cofundador, Mikko Mononen. Esse programa foi lançando em 2011 e foi adquirido pela Autodesk em 2013. Na Figura 13, segue uma captura de tela do ambiente Tinkercad:

O Tinkercad permite simulações dos projetos criados. É possível criar códigos-fonte/sketches

Figura 12 – Símbolo do Tinkercad

Fonte: ⟨https://www.tinkercad.com/⟩

Figura 13 – Captura de tela do ambiente Tinkercad



Fonte: ⟨https://www.tinkercad.com/⟩

e testá-los na própria ferramenta. Esses projetos também podem ser compartilhados com outras pessoas para visualização e para simulações, tal qual neste exemplo: ⟨https://www.tinkercad.com/things/68Coy0hxK0S⟩.

3 SENSOR DE LUMINOSIDADE

Grupo 01:
Guilherme Pozzan da Silva - ⟨gui_pozzan31@hotmail.com⟩
Pedro Paulo Gonçalvez de Souza - ⟨pedropaulogoncalvezdesouza9@gmail.com⟩

3.1 Este projeto: O que é

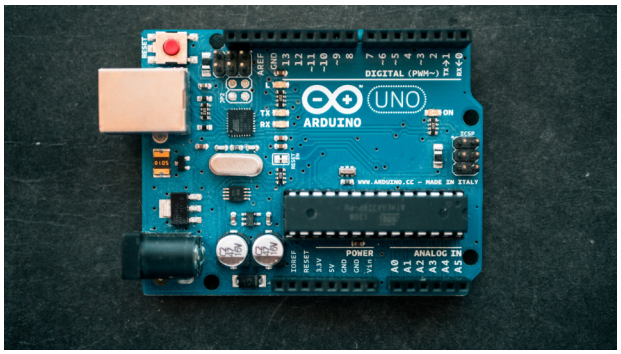
Esse projeto tem como objetivo fazer com que o sensor de luminosidade detecte uma certa quantidade de luz no ambiente onde ele encontra-se localizado. A partir dessa detecção o programa vai checar se essa intensidade é maior ou menor que a designada, caso seja maior o *LED* vai apagar e se for menor, o *LED* vai acender.

Este capítulo é baseado no trabalho de FILIPEFLOP (2022a).

3.2 Componentes usados

1. **Placa Arduino UNO e Cabo de Alimentação/Dados:** É uma placa microcontroladora que foi feita pensando na realização de projetos. O cabo será usado para gerar uma alimentação a placa e também para transmitir dados, como por exemplo o código-fonte, para o Arduino;

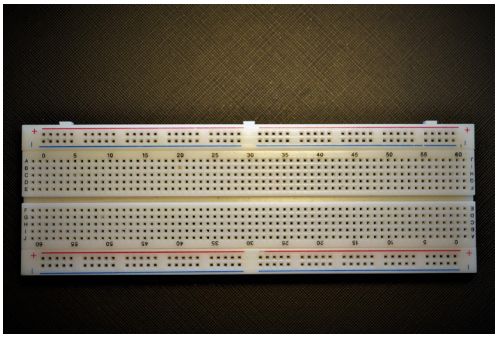
Figura 14 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

2. ***Protoboard*:** é uma placa com furos e conexões condutoras que vai ser usada para montar protótipos e projetos;

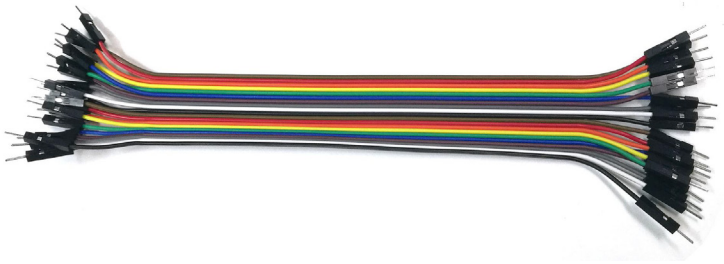
Figura 15 – *Protoboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

3. **Fios *Jumper* Macho-Macho:** é um pequeno condutor que vai ser utilizado para ligar dois pontos de um circuito eletrônico. Basicamente, vai ser usado para fazer a transmissão de dados;

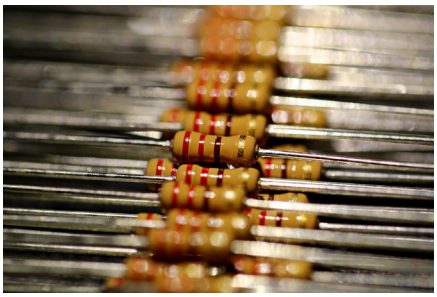
Figura 16 – Fios *jumper* macho-macho



Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

4. **Resistor 220 Ohm/Resistor 10k Ohm:** os dois têm como função limitar o fluxo de cargas eletrônicas;

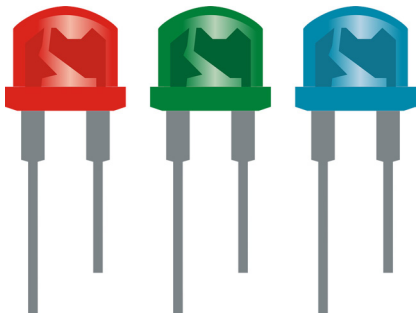
Figura 17 – Resistor



Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

5. **Lâmpada *LED*:** é uma lâmpada que vai acender ou desligar de acordo com a intensidade da luz presente no ambiente;

Figura 18 – *LED*



Fonte: ⟨https://pixabay.com/pt/vectors/rgb-conduziu-8mm-l%c3%a2mpadas-luz-2270087/⟩

6. **Sensor de Luminosidade:** vai detectar e fornecer valores da intensidade da luz do ambiente. A partir desses valores o programa vai ver se necessita ligar ou não a lâmpada *LED*.

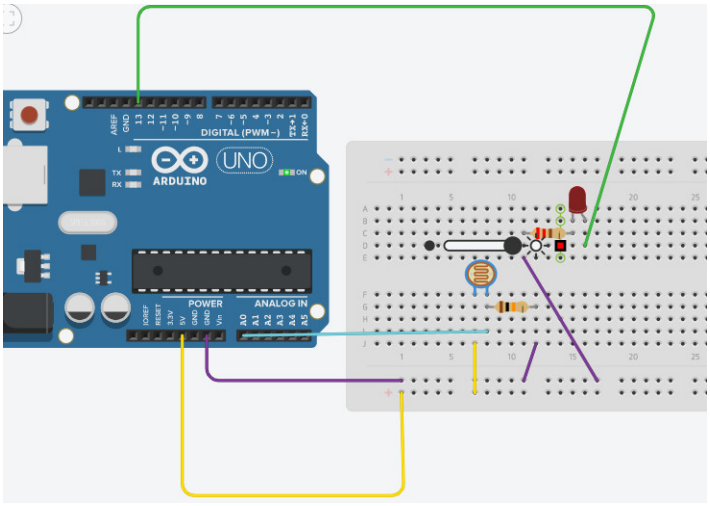
Figura 19 – Sensor de luminosidade LDR

Fonte: ⟨https://pixabay.com/pt/vectors/resistor-eletr%c3%b4nico-ldr-40611/⟩

3.3 Esquema de ligações

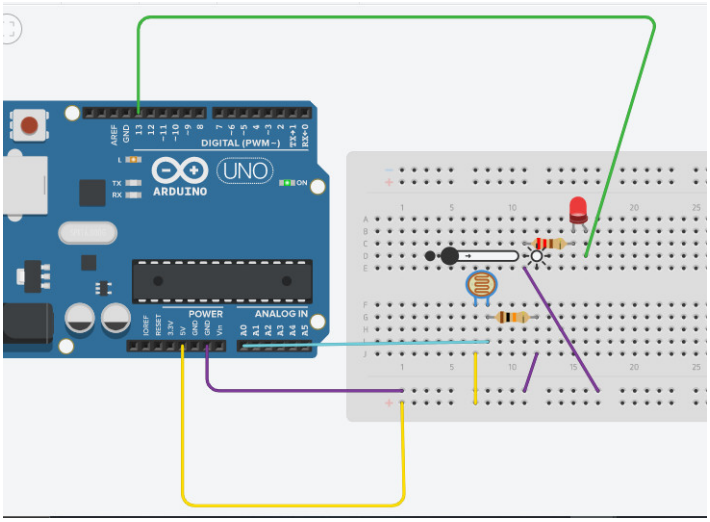
As duas imagens mostram como devem ser feitas as ligações para que o projeto funcione corretamente. As duas figuras foram criadas usando o site Tinkercad. Temos que nos atentar à lâmpada *LED*, devido ao fato dela ter lado positivo e negativo. Já o sensor de luminosidade não tem lado, o que facilita muito. Como podemos ver, a lâmpada está recebendo um transistor de 220 Ohm e o sensor um de 10k Ohm.

Figura 20 – *LED* apagado



Fonte: Criação própria / Tinkercad

Figura 21 – *LED* aceso



Fonte: Criação própria / Tinkercad

A Figura 20 mostra o *LED* apagado, ou seja, na simulação, como podemos ver na barrinha acima do sensor, está com uma alta intensidade de luz. Na Figura 21 o *LED* aparece ligado, então podemos dizer que nesse caso esse ambiente simulado está com uma baixa intensidade de luz.

3.4 Código-fonte

Logo abaixo vai estar localizado o código-fonte, que é utilizado para que esse projeto consiga ser capaz de identificar a intensidade e com isso dar a instrução de ligar ou desligar a lâmpada *LED*.

```
int pinoLed = 13;
int pinoSensorLuz = A0;
int valorLuz = 0;

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

void loop()
{
  valorLuz = analogRead(pinoSensorLuz);
  if(valorLuz<750)
  {
    digitalWrite(pinoLed,HIGH);
  }
  else
  {
    digitalWrite(pinoLed,LOW);
  }
  delay(10);
}
```

3.5 Sugestões de uso

Esse projeto pode ser utilizado em lugares ao ar livre para que seja possível acender as luzes automaticamente quando anoitece. Esse sistema é utilizado nos postes e em algumas luzes de jardim, facilitando muito e gerando certa economia.

3.6 Apresentação no canal de *YouTube* do *campus*

O projeto do sensor de luminosidade foi apresentado nas atividades avaliativas da disciplina de Hardware e Sistemas Operacionais do professor Josiney em 05/08/2021. Essa transmissão foi realizada no canal do *Youtube* do *campus* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/HF8c3yjI0D0?t=2513⟩.

4 INSTALAÇÃO DE S.O. EM UM *RASPBERRY* PI

Grupo 02:
Clara Haag Rodrigues - ⟨cla.hrd@gmail.com⟩
Davi Lima de Almeida - ⟨davimojang9@gmail.com⟩
Filipe Buttchevitz - ⟨lipebrusque@gmail.com⟩
Leonardo Gabriel Rohling - ⟨leogabriel.rg@gmail.com⟩
Maria Luiza de Santana Guedes - ⟨lisandrarodrigues13@icloud.com⟩

4.1 Este projeto: O que é

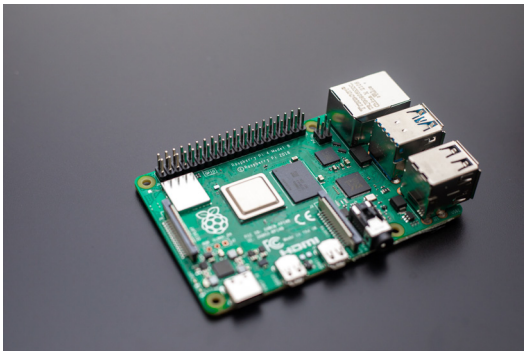
Este projeto é sobre o processo de instalação de um sistema operacional em um *Raspberry* Pi. O sistema utilizado será o Raspbian, que é o sistema oficial do *Raspberry* Pi, e também será utilizado um monitor e um teclado para a navegação no sistema.

Este capítulo é baseado nos trabalhos de RASPBIAN (2012) e FROMAGET (2022).

4.2 Componentes usados

1. ***Raspberry* Pi 4 model B e fonte DC chaveada USB tipo C:** mini computador desenvolvido pela Fundação *Raspberry* Pi com o objetivo de ser um computador de entrada mais barato. A fonte serve para manter o computador ligado;

Figura 22 – Uma das placas *Raspberry* Pi - modelo 4B



Fonte: ⟨https://unsplash.com/photos/BIHgNEaM394?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

2. **Periféricos (monitor, mouse, teclado):** serão utilizados na navegação do sistema operacional e não há necessidade de uma marca específica;

3. **Cartão de Memória Classe 10 16GB MicroSd SanDisk:** utilizado para instalar o sistema operacional e como armazenamento (assim como um HD em um computador normal). O *Raspberry* Pi suporta cartões de até 32GB, mas você consegue colocar cartões com uma capacidade um pouco maior se souber. Não é necessário utilizar apenas a marca SanDisk;

4. **Cabo HDMI:** utilizado para conectar o *Raspberry* Pi ao monitor;

4.3 Esquema de ligações

Na Figura 23 podemos ver o esquema de ligações do projeto. O monitor se conecta à entrada HDMI por meio de um cabo HDMI, o mouse e o teclado se conectando às duas entradas USB e o cartão SD entra em sua entrada.

Figura 23 – Esquema de ligações

Fonte: ⟨https://www.flickr.com/photos/robives/7427332488/in/photolist-cjk2bG-bUsch6-dxAdsv-bUF6mg-d7Hc7A-eE2yZM-bxhQoN-dd7qHJ-zhBbth-rsNTJK-2jouFDk-yb6BHa-qr6NbR-cbNMNG-onZM26-cbNPvd-dJFR2X-cbNNGJ-nFSWb5-oCswfb-onZW9o-nHRYKe-cjjZtU-cbNR5y-onZLPH-nFSVqY-cbNt6L-npA9Du-cbNwkh-npAeLF-g3tsLo-z29iQc-2kXEy6y-nG4bww-cbNuKy-dzWp1i-nHRXKi-cjk1ho-dmjFBz-oGf9TR-cbNxUo-nHRY6P-onZVUL-dWBZAA-dYKjXr-c6SRZC-npAfXD-oo1m8V-eyFVtF-CyGEVS⟩

4.4 Sugestões de uso

O *Raspberry* Pi não é um computador tão potente como um comum, porém pode ser facilmente utilizado para:

- Navegar na internet;
- Editar slides, documentos de texto e planilhas;
- Usar aplicativos de desenho;

Também pode-se usá-lo como um videogame, alguns emuladores como o RetroPie.

4.5 Processo de instalação do S.O.

Há vários sistemas operacionais disponíveis para o *Raspberry*, mas o oficial é o Raspbian. Para todos os sistemas o processo de instalação é o mesmo, assim como em um computador normal.

1. **Baixar a imagem:** vá até o site oficial, neste caso seria ⟨https://www.raspbian.org/⟩ para o Raspbian, e baixe a imagem;
2. **Baixar o Etcher:** Etcher é um software utilizado para instalar arquivos .img em um cartão SD por exemplo. O *link* para o site é: ⟨https://www.balena.io/etcher/⟩;
3. **Instalar a imagem .iso no cartão SD:** coloque o cartão em seu computador e abra o Etcher. Selecione o arquivo .img e o cartão SD caso já não tenha detectado, clique em *Flash!* para instalar o sistema;
4. **Ejete o cartão e coloque-o em seu *Raspberry*:** agora é só conectar os periféricos, como um monitor e um teclado, e aproveitar seu novo computador!

4.6 Outros S.O. para *Raspberry*

Além do Raspbian, há outras distribuições Linux que podem ser utilizadas no seu *Raspberry* Pi. Como por exemplo:

Ubuntu a distribuição mais utilizada no mundo tem sua versão para o pequeno computador. *Link* para download em: ⟨https://ubuntu.com/download/raspberry-pi⟩

Kali Linux o sistema operacional muito utilizado para achar vulnerabilidades e ataques na Internet possui uma versão compatível para o modelo mais atual, *link* do download aqui: ⟨https://www.kali.org/get-kali/#kali-platforms⟩

Kano OS voltado ao público infantil, possui interface gráfica simples e bem colorida. Foi desenvolvida pela Kano, uma empresa que produz kits de computador para ensinar sobre hardware. *Link* do download: ⟨https://kano.me/eu/downloadable⟩

DietPi desenvolvido para aqueles que querem um sistema leve e com apenas o essencial, *link* do download: ⟨https://dietpi.com/⟩

Fedora outra distribuição muito popular entre os usuários de Linux. *Link* do download em: ⟨https://fedoraproject.org/wiki/Architectures/ARM/Raspberry_Pi#Downloading_the_Fedora_ARM_image⟩

4.7 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 05/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/HF8c3yjI0D0?t=3032⟩.

5 MAKEY MAKEY

Grupo 03:
Camila Vitória Taboni - ⟨camilavtaboni@gmail.com⟩
Francisco Gabriel Hadwig Werle - ⟨franciscohwerle@gmail.com⟩
Gabrielly Correa - ⟨correa.gabrielly1010@gmail.com⟩
Larissa dos Santos Silva - ⟨larisantosilva05@gmail.com⟩

5.1 Este projeto: O que é

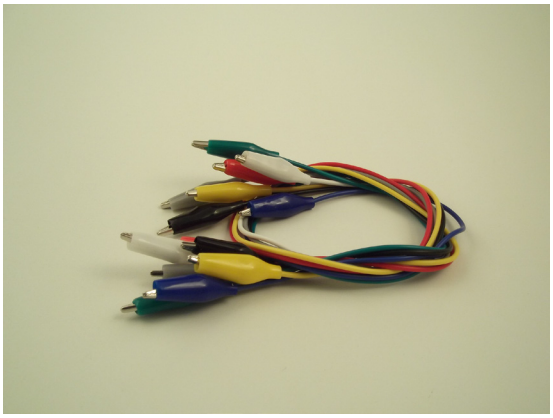
O nome Makey Makey vem do inglês Make: fazer, e key: tecla, ou seja, fazer + tecla. O Makey Makey foi criado por Jay Silver e Eric Rosenbaum e é um kit simplificado para que objetos cotidianos virem touchpads. O nosso projeto é um piano de bananas, isso mesmo, um piano com teclas de bananas.

Este capítulo é baseado no trabalho de KENSHIMA (2016).

5.2 Componentes usados

1. **Cabos garras de jacaré:** é utilizada na conexão de circuitos eletrônicos;

Figura 24 – Cabos garras de jacaré

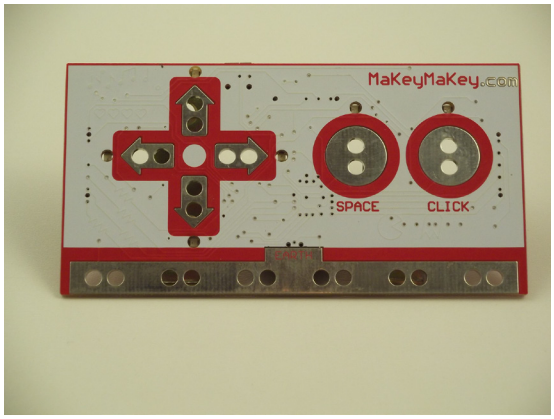


Fonte: ⟨https://www.flickr.com/photos/epicstuff_de/8212363964/in/photolist-dvGvG9-dvAWZa-dvGvff-dvGvbN-dwUKwH-dvGvuE-dvGw5W-dvAVQB-nUgEH5-nBS1M1-nUe4k3-d1WNks-dCkvMd-fm2aRg-fm1XK2-fm28Mc-fmganA-fmgjWE-ccJUzu-fm23Fa-fmgb1C-fmgbt7-fmgbLh-fmgaQu-fmghGd-ccJUBE-fm25m6-fm1SRB-fmg7X3-fm28fH-fm23kF-fm224k-fmg6AA-fmgf1o-fmgcRw-bVnEcK-fmge3w-ccJV1E-fmg5Bu-bVnE92-ccJUKu-bVnEnT-bVnEdg-bVnEoB-Qezxyq-2g1U7wh-bVnEhK-fmgghj-ccJUN9-ccJUDE⟩

2. **Cabo mini USB:** Este tipo de conexão permite que os dados sejam lidos sem a necessidade de um computador;

3. **Placa Makey Makey:** É uma placa de circuito e executa os processos programados;

Figura 25 – Placa Makey Makey



Fonte: ⟨https://www.flickr.com/photos/epicstuff_de/8212367990/in/photolist-dvGwTy-dvAXbt-dG4THJ-dvGvUY-dvGwhU-dvGvG9-dvAWZa-dvGvff-dvGvbN-dwUKwH-dvGvuE-dvGw5W-dvAVQB-nUgEH5-nBS1M1-nUe4k3-d1WNks-dCkvMd-fm2aRg-fm1XK2-fm28Mc-fmganA-fmgjWE-ccJUzu-fm23Fa-fmgb1C-fmgbt7-fmgbLh-fmgaQu-fmghGd-ccJUBE-fm25m6-fm1SRB-fmg7X3-fm28fH-fm23kF-fm224k-fmg6AA-fmgf1o-fmgcRw-bVnEcK-fmge3w-ccJV1E-fmg5Bu-bVnE92-ccJUKu-bVnEnT-bVnEdg-bVnEoB-Qezxyq⟩

4. **Bananas:** Objeto condutivo que junto ao Makey Makey pode virar um piano de bananas;

5.3 Esquema de ligações

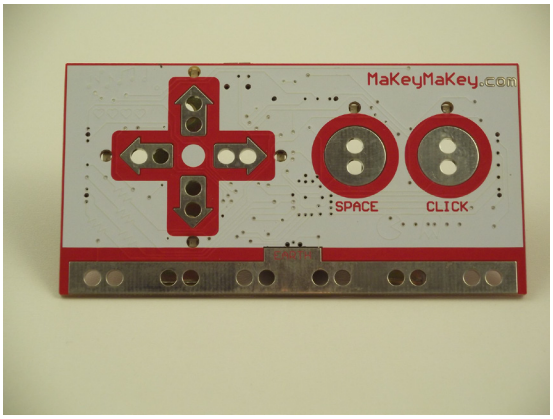
Plugando a placa em seu computador, as garras de jacaré nos *touch pads* do Makey Makey e em objetos condutores, formando um circuito, que realiza as ações programadas. Há uma barra inferior chamada Earth, que é o referencial Terra. O seu circuito necessita de aterramento, e neste caso, o aterramento será quem estiver interagindo com o circuito.

A parte frontal do Makey Makey contém os seguintes botões destacados e suas respectivas funções, também ilustrado na Figura 26:

- Alto
- Direita
- Baixo

- Esquerda
- A barra de espaço
- O click esquerdo do mouse

Figura 26 – Placa Makey Makey - Frente

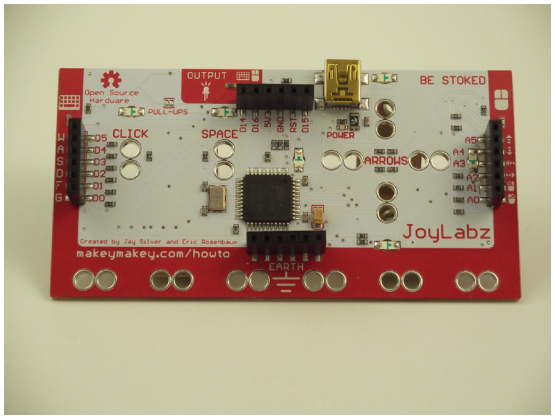


Fonte: ⟨https://www.flickr.com/photos/epicstuff_de/8212367990/in/photolist-dvGwTy-dvAXbt-dG4THJ-dvGvUY-dvGwhU-dvGvG9-dvAWZa-dvGvff-dvGvbN-dwUKwH-dvGvuE-dvGw5W-dvAVQB-nUgEH5-nBS1M1-nUe4k3-d1WNks-dCkvMd-fm2aRg-fm1XK2-fm28Mc-fmganA-fmgjWE-ccJUzu-fm23Fa-fmgb1C-fmgbt7-fmgbLh-fmgaQu-fmghGd-ccJUBE-fm25m6-fm1SRB-fmg7X3-fm28fH-fm23kF-fm224k-fmg6AA-fmgf1o-fmgcRw-bVnEcK-fmge3w-ccJV1E-fmg5Bu-bVnE92-ccJUKu-bVnEnT-bVnEdg-bVnEoB-Qezxyq⟩

Em sua parte de trás há as seguintes entradas, também ilustrado na Figura 27:

- Conector USB
- Entrada para o teclado
- Entrada para o mouse
- Barra do referencial Terra (Earth)
- Cabeçalho de saída/expansão
- *LEDs* do teclado
- *LEDs* do mouse

Figura 27 – Placa Makey Makey - Verso



Fonte: ⟨https://www.flickr.com/photos/epicstuff_de/8211279035/in/photolist-dvAXbt-dG4THJ-dvGvUY-dvGwhU-dvGvG9-dvAWZa-dvGvff-dvGvbN-dwUKwH-dvGvuE-dvGw5W-dvAVQB-nUgEH5-nBS1M1-nUe4k3-d1WNks-dCkvMd-fm2aRg-fm1XK2-fm28Mc-fmganA-fmgjWE-ccJUzu-fm23Fa-fmgb1C-fmgbt7-fmgbLh-fmgaQu-fmghGd-ccJUBE-fm25m6-fm1SRB-fmg7X3-fm28fH-fm23kF-fm224k-fmg6AA-fmgf1o-fmgcRw-bVnEcK-fmge3w-ccJV1E-fmg5Bu-bVnE92-ccJUKu-bVnEnT-bVnEdg-bVnEoB-Qezxyq-2g1U7wh⟩

5.4 Sugestões de uso

Como o Makey Makey é bem divertido, pode vir a ser utilizado como uma maneira de diversão e interação, despertando assim o interesse e a curiosidade nos alunos do *Campus*.

5.5 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 05/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://www.youtube.com/watch?v=HF8c3yjI0D0&t=1909⟩.

6 O USO DO *BLUETOOTH*

Grupo 04:
Fernanda Lanznaster - ⟨fernandaaaalanznaster@gmail.com⟩
Giovana Da Silva Dias - ⟨giovanadias580@gmail.com⟩
Rian Bormanieri - ⟨rian.borma@gmail.com⟩
Taynan Vila Nova - ⟨taynanvilanovasbj@gmail.com⟩
Thalia Schorner - ⟨thaliaschorner291@gmail.com⟩

6.1 Este projeto: O que é

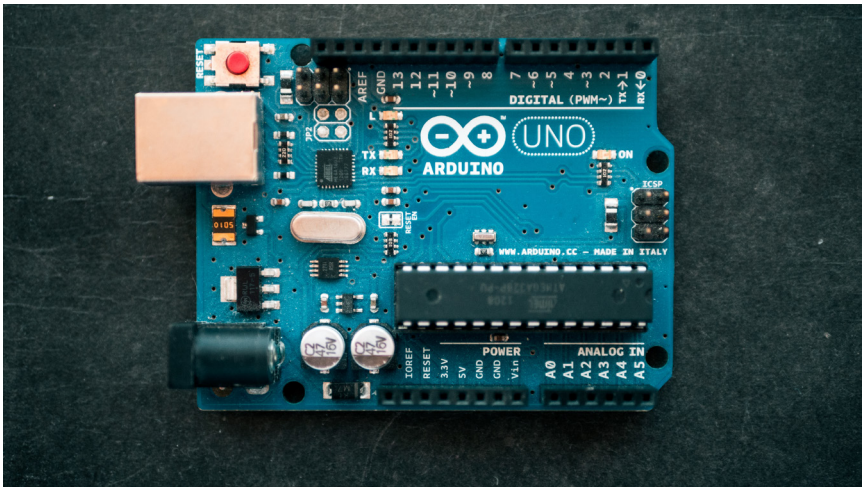
Este projeto visa fazer a conexão entre o módulo *Bluetooth* HC-05 ao Arduino. Veremos a seguir, como enviar informações de temperatura utilizando módulo *Bluetooth* com Arduino. Como o módulo *Bluetooth* HC-05 trabalha no modo mestre, ou seja, pode parear com outros dispositivos *bluetooth*, ele aceita pareamento.

Este capítulo é baseado no trabalho de THOMSEN (2015).

6.2 Componentes usados

1. **Placa Uno R3 e o Cabo USB para Arduino:**

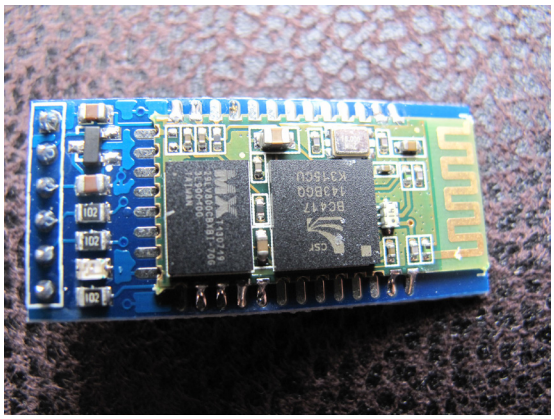
Figura 28 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

2. **Módulo *Bluetooth* RS232 HC-05:**

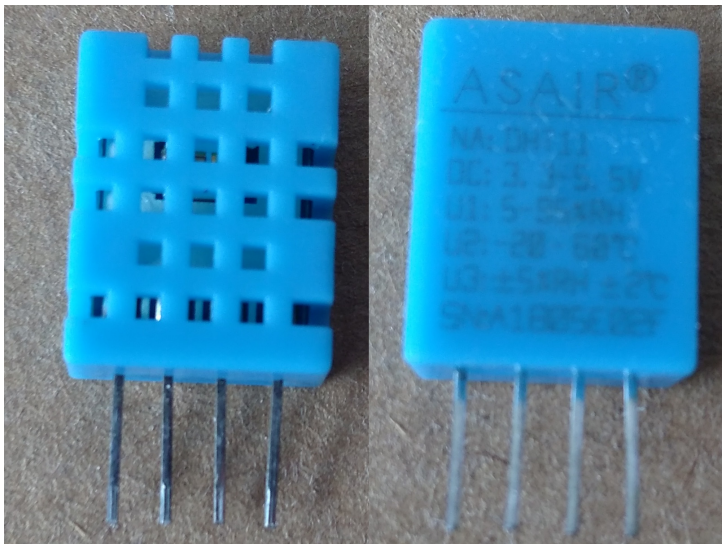
Figura 29 – Módulo *Bluetooth*



Fonte: ⟨https://www.flickr.com/photos/lilspikey/10097706146/in/photolist-goioeQ-ahqw77-24CaqRo-2xdxor-crYRZG-275yJ4S-pAGUm-JuaWb-crYSdN-6mheMm-7nfThF-4WwckD-d4fNVy-d4fP6U-6md6fz-6md6bF-86kjwy-6mheqS-d4fNfJ-49UhLV-d4fNzS-d4fNtd-d4fNM5-d4fMFC-d4fMvE-d4fN5C-d4fM23-d4fMRE-d4fM9w-d4fMeC-d4fMm9-d4fM7u-d4fM57-d4fMcj-49Ymvm-49Ympy-2mgEGPY-49UhYP-pHLfg-2m8y7bb-49YmsY-dT7Gfo-dyA9gx-49Ui3D-49UhWM-aQZHmg-e5dZ6i-ahms8B-ceaiPo-49UhQM⟩

3. **Sensor de Umidade e Temperatura DHT11:**

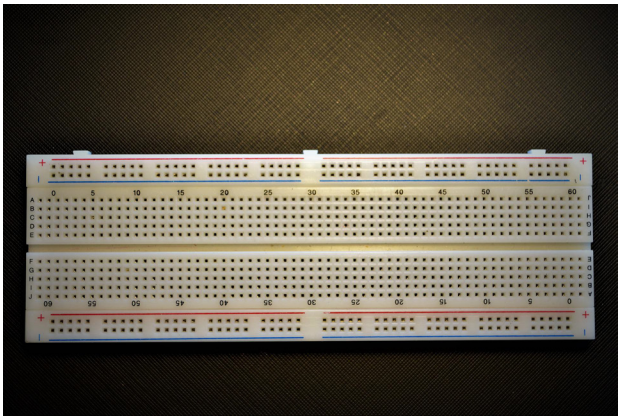
Figura 30 – Sensor DHT-11



Fonte: Arquivo pessoal de Josiney de Souza

4. ***Protoboard* 400 Pontos:**

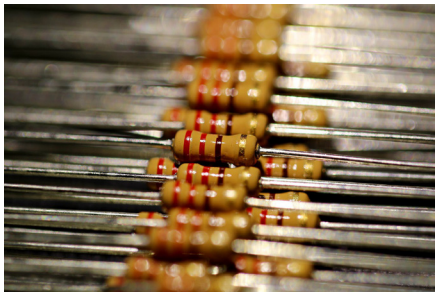
Figura 31 – *Protoboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

5. **Resistores 2.2KΩ 1/4W, 1KΩ 1/4W:**

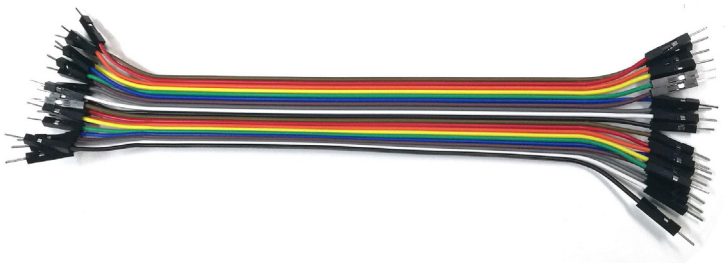
Figura 32 – Resistor



Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

6. **Kit *Jumpers* Macho-Macho:**

Figura 33 – Fios *jumper* macho-macho

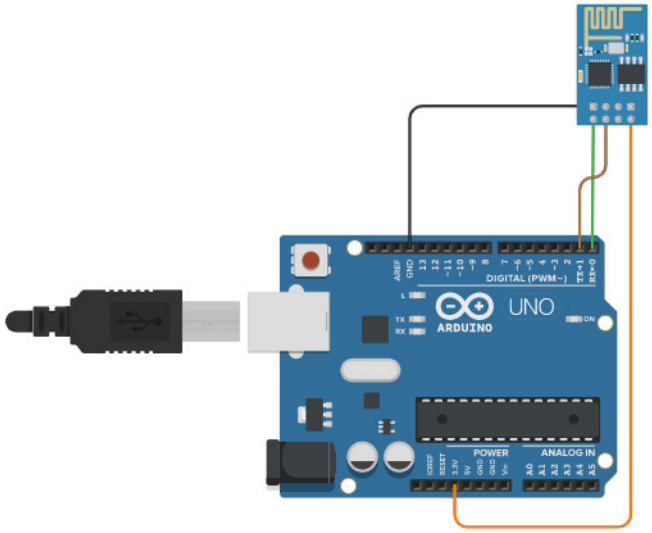


Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

6.3 Esquema de ligações

A Figura 34 mostra o esquema de ligações para o projeto, utilizando 2 resistores ligados ao pino RX do módulo *Bluetooth*. No nosso circuito usamos um de 1,5 K e outro de 2,2 K, o que gerou um nível de sinal de aproximadamente 3.1 V, suficiente para os testes. Monta-se o circuito deixando por enquanto o pino Vcc do módulo *Bluetooth* desconectado, já que tanto a comunicação com o computador como a comunicação do módulo *Bluetooth* com Arduino utilizam a mesma interface serial.

Figura 34 – Esquema de ligações de componentes



Fonte: ⟨https://www.tinkercad.com/things/032rz7FaRzX-bluetooth-hc-06⟩

6.4 Código-fonte

Abaixo está o código-fonte criado para a IDE do Arduino que faz a coleta e exibição de dados.

```
// Programa: Sensor DHT11 - Envio de dados via Bluetooth
// Autor: FILIPEFLOP

#include "DHT.h"

#define dht_pin A5 //Pino DATA do Sensor ligado na porta Analogica A5
```

```
#define DHTTYPE DHT11

DHT dht(dht_pin, DHTTYPE);

void setup()
{
  Serial.begin(9600);
  // Aguarda 1 seg antes de acessar as informações do sensor
  delay(1000);
  dht.begin();
}

void loop()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Mostra os valores lidos, na serial
  Serial.print("Temp. = ");
  Serial.print(t);
  Serial.print(" C ");
  Serial.print("Um. = ");
  Serial.print(h);
  Serial.println(" % ");

  // Nao diminuir muito o valor abaixo
  // O ideal e a leitura a cada 2 segundos
  delay(2000);
}
```

6.5 Exemplo da Comunicação *Bluetooth* com Arduino e celular

Nos testes, utilizamos um celular com Android, juntamente com o aplicativo *Bluetooth* SPP, disponível neste link (⟨https://play.google.com/store/apps/details?id=mobi.dzs.android.BLE_SPP_PRO⟩) do Google Play. É um aplicativo com vários recursos, mas o que vamos utilizar no momento é a apresentação em tempo real dos dados recebidos via *Bluetooth*.

Antes de utilizar o aplicativo, faça o pareamento entre o seu celular e o módulo *Bluetooth*. Procure pelo dispositivo *Bluetooth* (geralmente com o nome LINVOR), e utilize a senha 1234.

Execute o aplicativo *Bluetooth* SPP. Será apresentada a tela principal. Selecione o botão MENU do celular e escolha a opção *Connected*. O celular inicia uma varredura para detectar os dispositivos *Bluetooth*, e o módulo LINVOR que você pareou anteriormente será exibido na tela.

Clique no dispositivo detectado e na tela de seleção de modo de operação, selecione *REAL TIME MODE* (Modo de tempo real). Será mostrada, então, uma nova tela, com as informações de temperatura atualizadas a cada 2 segundos.

6.6 Sugestões de uso

Fazer transferência de dados rapidamente (dependendo do tamanho do arquivo) com um alcance de 10 metros dependendo do aparelho. Também poderia ser usado para enviar informações para acender uma lâmpada em determinado ambiente, usando o celular como um controle remoto.

6.7 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 19/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/iqQwer30uSY?t=2612⟩.

7 O SENSOR DE MOVIMENTO EM CONJUNTO COM O ARDUINO PARA ACENDER UM *LED*

Grupo 05:
Eduardo Mateus Teixeira Rodrigues Farias - ⟨fariasedu54@gmail.com⟩
Isabelli do Amaral - ⟨doamaralribeiroisabelli@gmail.com⟩
Isabel Lyssa Zimmermann Lima - ⟨isabellyssa764@gmail.com⟩
Nathalia Morais Gregorio - ⟨natymorais764@gmail.com⟩
Polhiane Hey - ⟨polhiii.anee@gmail.com⟩

7.1 Este projeto: O que é

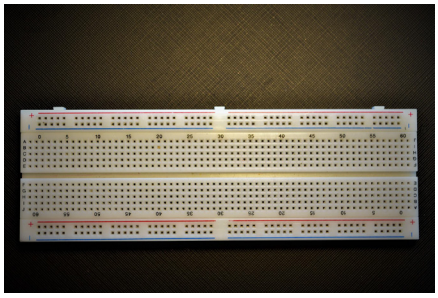
Este projeto tenciona articular sobre o sensor de movimento e utilizá-lo em companhia de um Arduino com o intuito de acender um *LED* por dois segundos ao detectar algum movimento próximo.

Este capítulo é baseado no trabalho de REIS (2018).

7.2 Componentes usados

1. ***PROTOBOARD*:** placa extensora usada para prototipação. ESPECIFICAÇÕES:
   - 400 Pontos
   - Furos Prototipagem

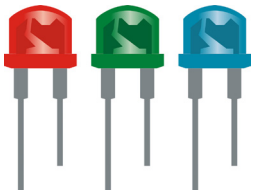
Figura 35 – *Protoboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

2. **UM *LED*:**

Figura 36 – *LED*

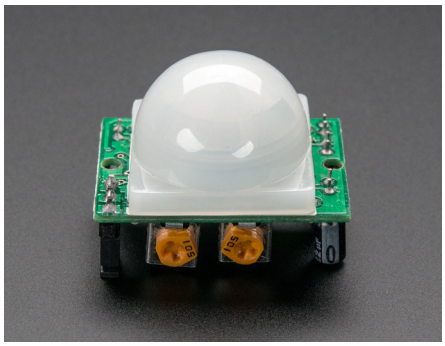


Fonte: ⟨https://pixabay.com/pt/vectors/rgb-conduziu-8mm-l%c3%a2mpadas-luz-2270087/⟩

3. **SENSOR PIR HC-SR501:** utiliza um sensor piroelétrico, quando em uso é capaz de detectar movimento em uma área de até 7 metros trabalhando com um ângulo de até 100°. Ele emite um sinal de 3.3V através de seu pino OUT, sua parte traseira possui um potenciômetro que permite uma configuração mínima de 3 segundos. Tem a capacidade de ajuste para o tempo de retardo (de até 200 segundos). ESPECIFICAÇÕES:

   - Ele é passivo e sua Alimentação é de 5V – 20V
   - Distância detectável: 3-7m
   - Raio de alcance de aproximadamente 100°

Figura 37 – Sensor de presença/movimento

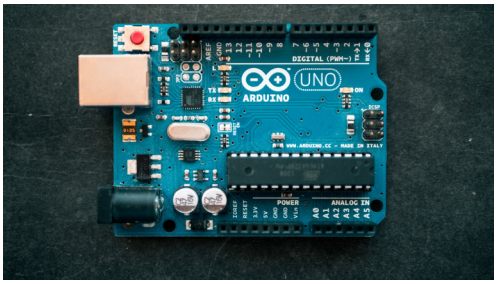


Fonte: ⟨https://www.flickr.com/photos/adafruit/11240262523/in/photolist-i8ghrH-i8fKVe-i8fKze-Jga6Tb-oq4GyK-Jd9SFB-2gLHHfB-2crm4eC-aa1tiM-4Wo6mM-4kDWnm-4kG1NB-4kzUBi-26WBTPA-4kFSGp-dUavQo-dUGuRk-mNmjRX-2dkbk25-CTAiLz-NL4zzT-2mYLDrP-dUav33-7PFhHV-nphaoi-2jCJmVw-byBpYq-fMNcFB-Maw68H-2h6KWi7-STd9RQ-T84fea-meFYH6-u8zJP3-P1Rrc1-tt9FTh-meH1ss-2jCJmVm-aw8Etg-bRnZQr-2mPux9D-dbqGRv-zb37kX-7BZQWK-cVyjwm-dbqKA5-ryt5Ze-rh1un5-rwhmpm-qBMzVg⟩

4. **ARDUINO UNO R3:** é uma plataforma de prototipagem eletrônica *open-source* que se baseia em hardware e software. ESPECIFICAÇÕES:

- Ele tem 20 pinos de entrada / saída digital
- Um ressonador de 16MHz
- Uma conexão USB
- Um botão de reiniciar

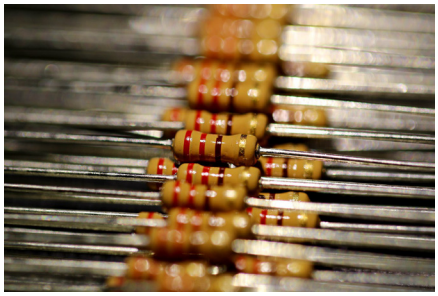
Figura 38 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

5. **RESISTOR de 220 Ohms:** o resistor apresenta resistência à passagem de eletricidade. Na prática eles limitam a intensidade de corrente elétrica. ESPECIFICAÇÕES:
   - Filme Carbono 1/4 watts
   - Tolerância 5

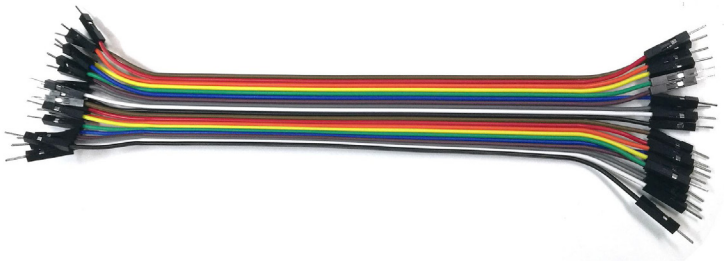
Figura 39 – Resistor



Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

6. **FIOS E CABOS *JUMPER*:** usados para transmitir dados de sensores e atuadores. ESPECIFICAÇÕES:
   - Modelo: FEMEA X FEMEA
   - Dimensão: 20 CM
   - C/ 40 PEÇAS

Figura 40 – Fios *jumper* macho-macho



Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

7.3 O sensor de movimento

Um sensor de movimento é um tipo de instrumento elétrico que usa um detector ou um sensor para detectar o movimento próximo.

7.3.1 Como funciona

Os sensores de movimento são capazes de detectar corpos tanto em ambientes iluminados quanto em uma completa escuridão.

7.3.2 Resumo

Em geral os sensores de movimentos se dividem em ativos e passivos, os quais os ativos trabalham com a emissão de um feixe de luz infravermelha e se algo cortar esse feixe o sensor dispara, já os passivos fazem uma leitura das mudanças de luz no ambiente que está sendo monitorado, eles detectam a luz infravermelha de corpos quentes.

7.3.3 Sensor de movimento ativo

São utilizados em locais que podem ser facilmente invadidos, como por exemplo, muros de residência, portas, janelas ou áreas de acesso restrito. Esse sensor depende de dois tipos de dispositivos para funcionar, um deles vai emitir um feixe de luz infravermelha enquanto o outro vai receber. Esse sinal é comparado constantemente entre o tempo da emissão e o do recebimento do feixe. Quando há uma certa diferença entre esses tempos significa que algo ou alguém se interpôs sobre a luz. Os sensores mais modernos são programados para não disparar quando algo de pequeno porte passa sobre eles, como por exemplo os animais. Alguns modelos do sensor de movimento ativo são os mais indicados para portas e janelas, pois emitem e recebem o sinal da luz infravermelha a poucos metros de distância, já outros

modelos são mais indicados para terrenos e áreas abertas, pois emitem e recebem o sinal do feixe de luz a dezenas de metros.

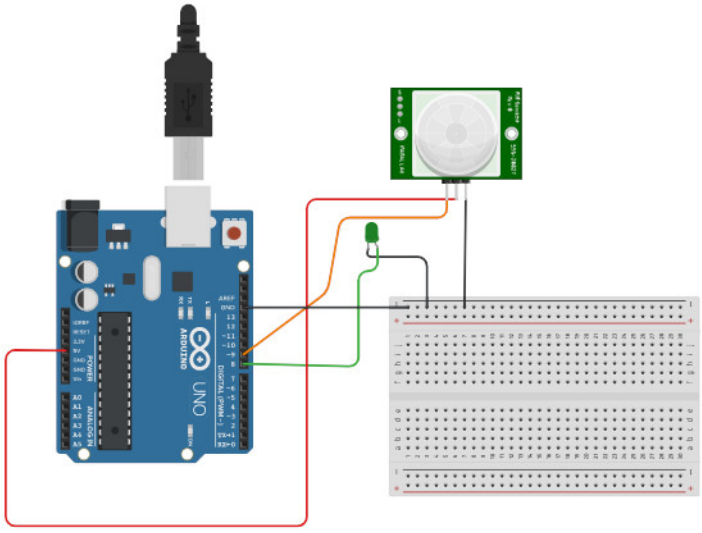
7.3.4 Sensor de movimento passivo

São utilizados mais em residências, como salas, quartos, corredores, varandas ou até mesmo pequenas áreas de acesso restrito. Esses sensores fazem uma leitura das mudanças da luz infravermelha no ambiente monitorado, já que os humanos e animais tem um corpo mais quente do que os objetos, sendo assim, acaba emitindo a radiação infravermelha. Quando alguém passa diante do sensor isso faz com que a intensidade de radiação infravermelha aumente e assim, o alarme é disparado. Existem variações nesses sensores, alguns detectam movimentos num ângulo de 90° enquanto outros podem detectar interações em um raio de até 10 metros. Fontes de calor intenso ou equipamentos de refrigeração podem afetar o desempenho dos sensores.

Curiosidade: Algumas câmeras contam com sensores de movimento como forma de ativação da gravação de imagens.

7.4 Esquema de ligações

Figura 41 – Esquema de ligações



Fonte: ⟨https://www.tinkercad.com/things/2vBZpZ4Bqxx-sensor-de-movimento⟩

A entrada de 5V do Arduino se conecta à coluna positiva da *protoboard*, enquanto a entrada GND (Terra) conecta-se a negativa, o sensor recebe energia Terra e 5V destas entradas.

O *LED* apenas se alimenta da entrada Terra já que, o seu Anodo conecta-se a uma das entradas do Arduino, que trata de ser, no caso da nossa imagem, a entrada 10 sem antes atravessar um resistor que garante que o mesmo não queime, já o sensor, se conecta a entrada 4, através da saída (Out) do sensor.

Ao receber energia de uma entrada USB, o Arduino alimenta, o sensor com 5V e energia Terra, quando o mesmo recebe um sinal de movimento, este envia um sinal direto para o Arduino através da entrada 4, quando detecta tal sinal, o Arduino envia energia através da entrada 10 para o *LED* que está sendo alimentado apenas pelo Terra. Assim acendendo o *LED*, que depois de passado uma determinada quantia de tempo, se desliga.

7.5 Código-fonte

```
const int pinoLED = 10;
const int pinoPIR = 4;
void setup() {
pinMode(pinoLED, OUTPUT);
pinMode(pinoPIR, INPUT);
}
void loop() {
int valor = digitalRead(pinoPIR);
if (valor == HIGH) {
digitalWrite(pinoLED, HIGH);
delay(2000);
digitalWrite(pinoLED, LOW);
}
}
```

As constantes pinoLED e pinoPIR são criadas com a função de configurar, respectivamente, os pinos de conexão do *LED* e de saída do sensor PIR. Na função de setup, os pinos são ajustados como saída (*LED*) e como entrada (PIR).

Na função principal, *loop*, é criada dinamicamente uma variável de nome valor que receberá o nível lido no pin do PIR, isso por meio do método digitalRead no pino correspondente. Caso o valor lido for nível alto (*HIGH*), o *LED* acenderá, permanecendo dessa forma por 2 segundos, ou 2000 milissegundos - função *digitalWrite* -, e logo após apagando, se acendendo novamente caso o sensor detecte movimento.

7.6 Sugestões de uso

O sensor de movimento pode ser integrado em diversos ideais:

- Para abrir ou fechar as portas, como nos elevadores.
- Para aumentar a segurança de ambientes com acesso restrito, incorporado, por exemplo, com um alarme.
- Como já dito anteriormente, com câmeras de vídeo.

No projeto aqui apresentado, está mais ligado com o controle automático de iluminação, ou seja, acender alguma luz ao detectar movimento (tendo como exemplo comum as luzes dos corredores de prédios).

Os sensores de movimento proporcionam economia de energia (ao impedir luzes ligadas sem motivo por muito tempo) e também praticidade.

7.7 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 12/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/DzjiyOzZOTs?t=816⟩.

8 USO DE CONTROLE REMOTO - IR

Grupo 06:
Alice Pereira - ⟨alice2005pereira@gmail.com⟩
Giovana Schmitt - ⟨giovanaschmitt10@hotmail.com⟩
Hérika Smaniotto - ⟨herika3242@gmail.com⟩
Heloysa Smaniotto - ⟨heloysa3242@gmail.com⟩

8.1 Este projeto: O que é

Este projeto diz respeito ao controle remoto IR, mais conhecido no Brasil como infravermelho, de aparelhos eletrônicos. Consiste em um pequeno dispositivo que contém um chip de microcontrolador, um ou mais *LEDs* emissores de infravermelho e um teclado acoplado. Quando o usuário pressiona uma das teclas do controle, uma sequência de pulsos de luz infravermelha é transmitida pelos *LEDs*, acendendo-os. Esses pulsos formam um código que é único para cada tecla acionada.

Este capítulo é baseado no trabalho de MURTA (2021).

8.2 Componentes usados

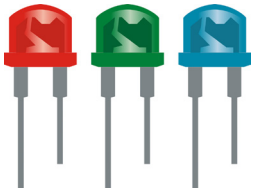
1. **Um controle remoto infravermelho (receptor e controle):**

Figura 42 – Controle remoto infravermelho

Fonte: ⟨https://www.flickr.com/photos/169136682@N02/46883118884/in/photolist-2eqUeRW-2gwuH4k-DkwLwv-2hjmGVq-2mFEGEw-2mKMG3B-2i9R6iv-2neABus-2njLoqJ-2mNY2cv-2niSE3F-2n5fTYQ-a92EzK-bSLTE6-bqwuPq-gL2PT-21Ao9Wq-23gup3m-LwYKKJ-hq4p4y-btd4a6-2n4TCi4-LefoRo-7mxGs9-aJffn-2nc7Njo-2niogML-2nfDzdX-2n4jArp-25paerZ-2mTSvv1-2n2AtbR-2ndftES-2nkt4sV-2n5Qude-2mZZiRv-2n3eDm7-2n1XkgW-2mVu1qN-2mGSuta-2kbwNa6-2mPDgLG-NFbzxx-6TxHqg-q9oTYV-7e4MDP-4ede23-sNwFw-3fw524-sNwFq⟩

2. ***LEDs* de cores variadas:** é um componente eletrônico semicondutor, ou seja, um diodo emissor de luz;

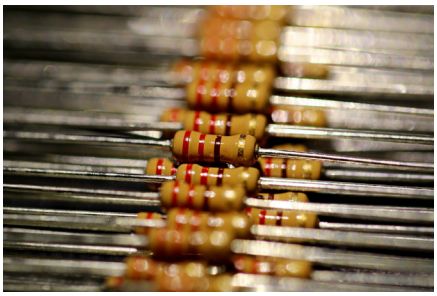
Figura 43 – *LED*



Fonte: ⟨https://pixabay.com/pt/vectors/rgb-conduziu-8mm-l%c3%a2mpadas-luz-2270087/⟩

3. **Resistores de 220 Ohms:** os resistores são elementos que apresentam resistência à passagem de eletricidade;

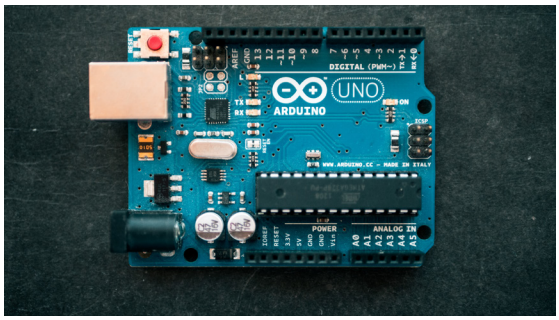
Figura 44 – Resistor



Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

4. **Arduino Uno R3:** é uma plataforma de prototipagem eletrônica;

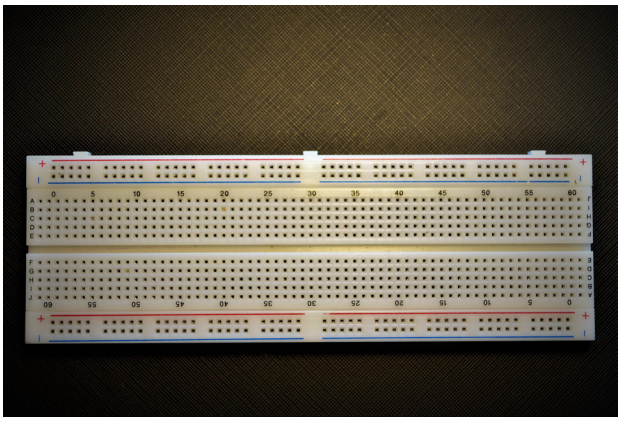
Figura 45 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

5. ***Protoboard*:** a *protoboard* é uma placa que possui furos e conexões internas para montagem de circuitos;

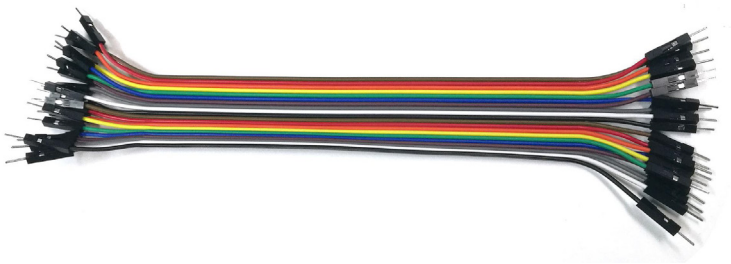
Figura 46 – *Protoboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

6. **Cabos *jumper*:** têm a função de transmitir dados para o arduino para os *LEDs*.

Figura 47 – Fios *jumper* macho-macho

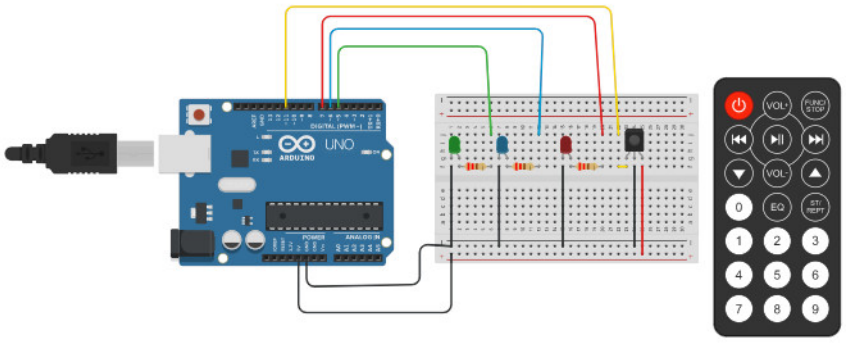


Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

8.3 Esquema de ligações

A montagem do circuito é simples, interligando um módulo Receptor IR AX-1838Hs ao Arduino. A alimentação do módulo é fornecida pelo 5V do Arduino, e o pino de saída dos pulsos do receptor é conectado ao pino D11 do Arduino. O *LED* amarelo está conectado no pino D05, o *LED* verde no pino D06 e o *LED* vermelho no pino D07 do Arduino. Todos os *LEDs* tem um resistor de 220 ohms em série com as ligações. O lado chanfrado do *LED* é o catodo, que deve ser conectado no terra (GND).

Figura 48 – Esquema de ligação



Fonte: ⟨https://www.tinkercad.com/things/e7fAxIYdlql-controle-remoto-ir⟩

8.4 Código-fonte

A biblioteca para trabalhar com o controle remoto IR é IRremote. O código necessário para o funcionamento do sensor é o seguinte:

Primeiro código:

```
#include <IRremote.h> //inclui a biblioteca do IR

int receiver = 13;
//variável que armazena o pino de o sensor está

IRrecv irrecv(receiver); //define o pino do sensor
decode_results results;

void setup()
{
    Serial.begin(9600); //Inicia o Serial
    irrecv.enableIRIn();
}

void loop()
```

```
{
    if (irrecv.decode(&results)) //Verifica valor recebido
    {
        //mostra o valor em hexadecimais no monitor serial
        Serial.print("Valor lido: ");
        Serial.println(results.value, HEX);
        irrecv.resume(); //lê o próximo valor
    }
}
```

O código anterior é implementado para que funcione da seguinte forma:

- Tecla 1 - Acende o *LED* Vermelho
- Tecla 2 - Apaga o *LED* Vermelho
- Tecla 3 - Acende o *LED* Verde
- Tecla 4 - Apaga o *LED* Verde
- Tecla 5 - Acende o *LED* Amarelo
- Tecla 6 - Apaga o *LED* Amarelo

Segundo código:

```
#include <IRremote.h> //inclui a biblioteca do IR

float valor; //servira para armazenar o valor recebido pelo sensor
int pino_receiver = 13; //variavel que armazena o pino que o sensor esta conectado
int ledvermelho = 7; //armazena o pino que o LED vermelho esta conectado
int ledverde = 6; //armazena o pino que o LED verde esta conectado
int ledamarelo = 5; //armazena o pino que o LED amarelo esta conectado

IRrecv irrecv(pino_receiver); // define o pino do sensor
decode_results results;

void setup()
{
  pinMode(ledvermelho, OUTPUT); //define os pinos dos leds como saida
  pinMode(ledverde, OUTPUT);
  pinMode(ledamarelo, OUTPUT);
  Serial.begin(9600); //inicia o serial
```

```
   irrecv.enableIRIn(); //inicia o receptor ir
}

void loop()
{
   if (irrecv.decode(&results)) //verifica o valor recebido
   {
       Serial.print("Valor lido: "); //mostra o valor no monitor serial
       Serial.println(results.value, HEX);
       valor = (results.value); //variavel "valor" recebe o codigo do botao
           pressionado
       if (valor == 0xFFA25D) //executa alguma funcao, de acordo com qual botao
           foi pressionado
       {
           digitalWrite(ledvermelho, HIGH);
       }
       if (valor == 0xFF629D)
       {
           digitalWrite(ledvermelho, LOW);
       }
       if (valor == 0xFF22DD)
       {
           digitalWrite(ledverde, HIGH);
       }
       if (valor == 0xFF02FD)
       {
           digitalWrite(ledverde, LOW);
       }
       if (valor == 0xFFE01F)
       {
           digitalWrite(ledamarelo, HIGH);
       }
       if (valor == 0xFFA857)
       {
           digitalWrite(ledamarelo, LOW);
       }
       if (valor == 0xFFE21D)
       {
           digitalWrite(ledvermelho, LOW);
```

```
            digitalWrite(ledverde, LOW);
            digitalWrite(ledamarelo, LOW);
        }
        irrecv.resume(); //espera o proximo botao ser pressionado
    }
}
```

8.5 Sugestões de uso

Utilizado para controlar de forma remota algum tipo de aparelho compatível com tecnologia infravermelha. Sugestões de uso incluem: acender e apagar lâmpadas, sejam as convencionais ou de *LED*.

8.6 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 05/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC Campus Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/HF8c3yjI0D0?t=3917⟩.

9 EXIBIÇÃO DE UMA MENSAGEM EM *DISPLAY*

Grupo 08:
José Enrico B. Belli - ⟨joseenrico1811@gmail.com⟩

9.1 Este projeto: O que é

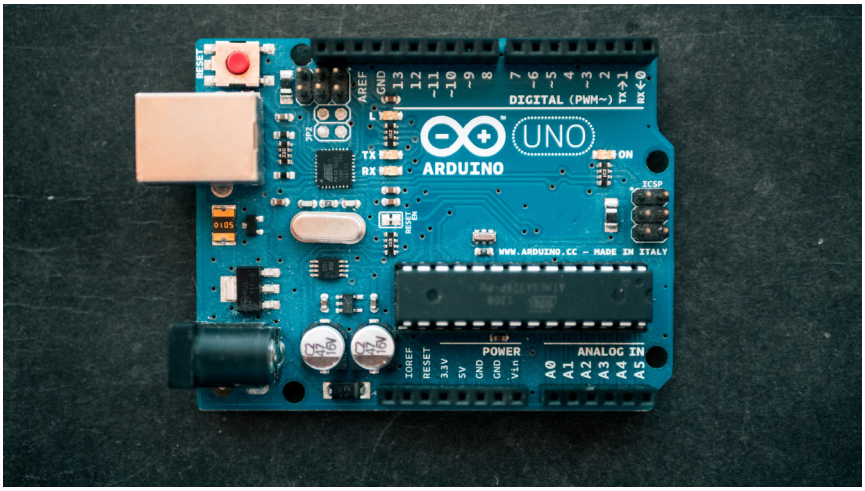
Este projeto tem como objetivo explicar como utilizar um *display LCD* com Arduino, explicando duas funções: a de escrever na tela e mover o texto escrito para a direita e depois para a esquerda, e depois como criar e utilizar um caractere personalizado.

Este capítulo é baseado no trabalho de MURTA (2018).

9.2 Componentes usados

1. **Arduino e cabo de alimentação/dados:** placa microcontroladora usada para se trabalhar com protótipos. O cabo USB é usado para transmitir o código-fonte para a placa e para alimentar a placa;

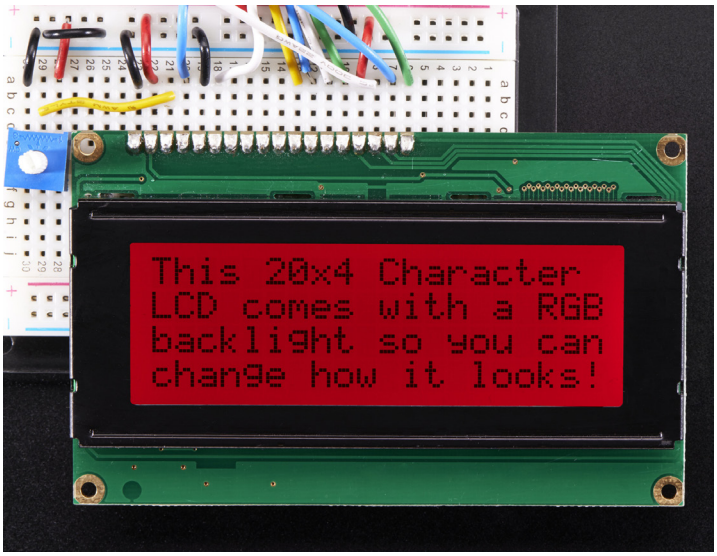
Figura 49 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

2. ***Display LCD* 16x2:** *display* de cristal líquido, um *display* fino usado para mostrar informações de forma eletrônica, como imagens e textos;

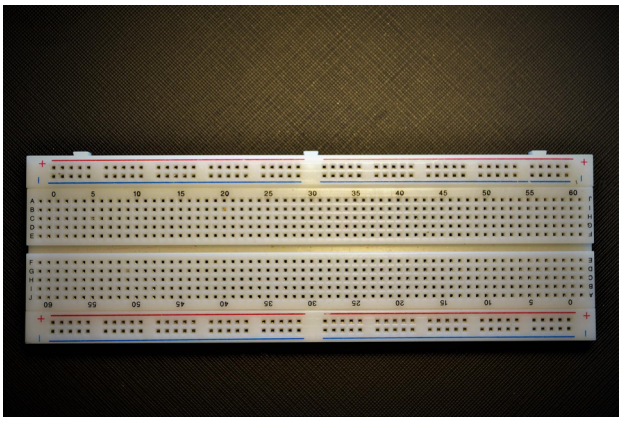
Figura 50 – *Display LCD*



Fonte: ⟨https://www.flickr.com/photos/adafruit/17572014992/in/photolist-sLMaRJ-rQaMVe-sLMphG-suoCZu-pqdMbd-5bPtJj-kaLZX-5uL15p-28W72tR-rPYb5j-wFKKcj-suorFs-sLZ9MD-sLZ324-w2mcyE-suwagD-suw1nr-46TgVD-9fRJhz-fJpNvP-pYAajq-butLFK-sLZ73v-4m4igU-suopyw-ydrRfY-sJDZBL-suw8Mr-5uQn5s-5KV3Kk-pG4NaJ-sLZ8hV-rQaFu8-suw8YZ-sLYHMz-2kxPHVK-2nm4WD6-2j1SfCA-2mZYuwP-KJjLBv-4mGuxe-562gNk-2gm61nq-5xcWHj-suoxDj-sLYQWZ-suvMV2-sLZ5Ln-suor69-46TgJv⟩

3. **400-*Point Breadboard*:** placa externa que aumenta o número de entradas do Arduino;

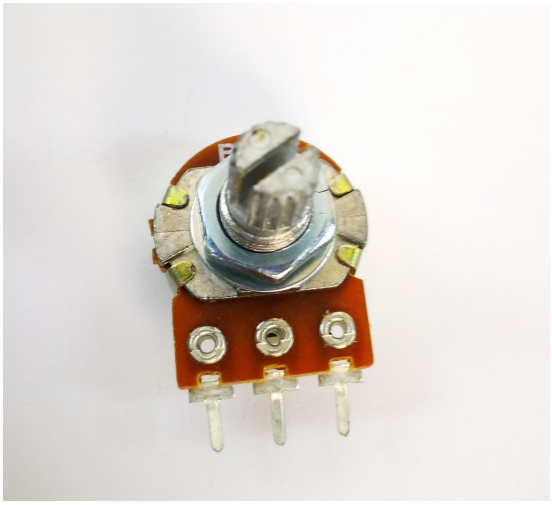
Figura 51 – *Protoboard*/*Breadboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

4. **Potenciômetro:** componente eletrônico com resistência eletrônica ajustável;

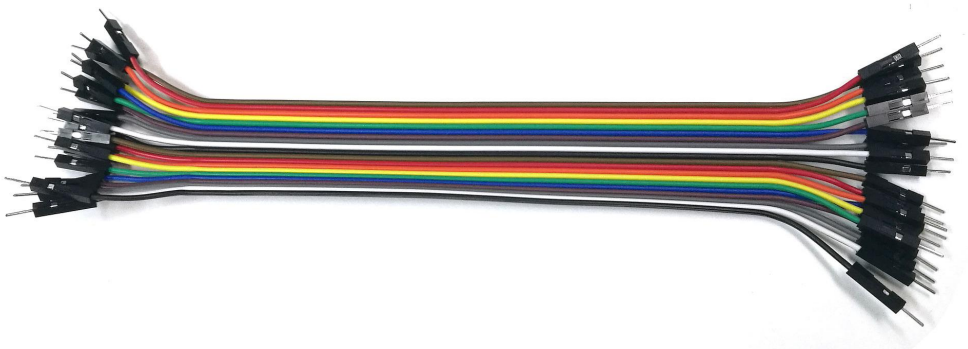
Figura 52 – Potenciômetro



Fonte: ⟨https://www.flickr.com/photos/161851377@N08/42193851721/in/photolist-FUsduJ-Fp7uL9-5NwodA-gWfSHA-CsNbQ-28WEbu-kmYyju-kmWc5R-aNLWzM-kmWa9B-29symT-kmYz5C-kmX5Zk-BBVpkz-kmX5v4-dpSUth-kmW9kc-dvziTU-kmWbdR-7h4fK2-27hww76-7bnFsu-4WeyN9-6t4Q9o-kmYEnw-4RVkax-7NCgto-2tdRFu-9t8yuz-5BeEv2-pwQCxC-2c9NnWt-VjtLEE-pwToUw-HJmon1-kntyRV-Yj8RJH-2aveXcZ-x2gpRE-uAJD2S-Rw1ygG-eimuiF-ai1Rp5-a83EzQ-4RZwFS-knvQfA-kmWfVM-4RVkVa-7qUWQj-knujDc⟩

5. **Cabos *jumper*:** cabos utilizados para a transmissão do Arduino pros componentes e vice-versa;

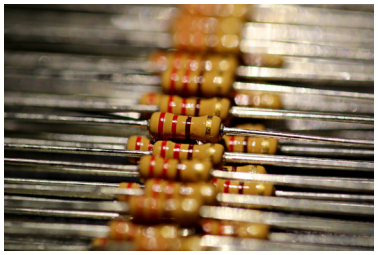
Figura 53 – Fios *jumper* macho-macho



Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

6. **Resistor:** é utilizado para limitar a carga elétrica em um circuito;

Figura 54 – Resistor

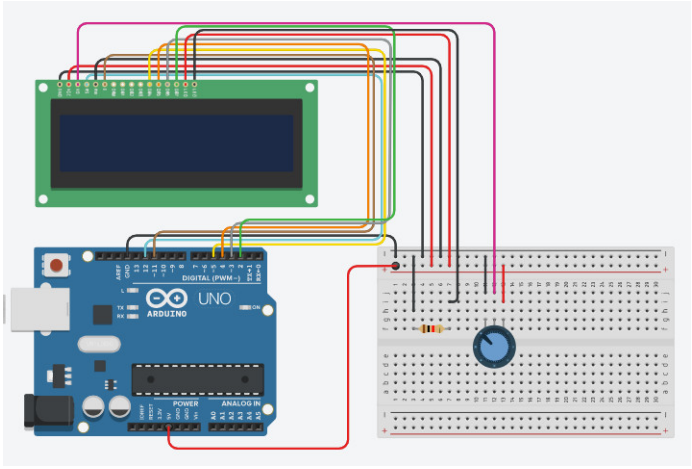


Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

9.3 Esquema de ligações

No esquema de ligações iremos utilizar 4 pinos de dados(2,3,4 e 5), dois pinos de controle(11 e 12), três pinos negativos(GND), um pino para ligar ao pino central do potenciômetro, dois pinos de 5V, e os outros 4 desconectados. Como mostrado na Figura 55 abaixo.

Figura 55 – Esquema de ligações



Fonte: Próprio autor

9.4 Código-fonte

O código abaixo é o código responsável por exibir uma mensagem e movê-la para o lado esquerdo e direito.

```
//Importa a biblioteca LiquidCrystal, a biblioteca utilizada para o
funcionamento do display
#include <LiquidCrystal.h>
```

```
//Declara uma variável que fala para o arduino os pinos que o display está
utilizando.
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
//Declara o tamanho do display, sendo o primeiro valor as colunas, e o
segundo as linhas.
lcd.begin(16,2);
}

void loop() {
//Limpa tudo escrito no display.
lcd.clear();
//Decide o ponto de início da mensagem.
lcd.setCursor(4,0);
//Escreve a mensagem.
lcd.print("Mensagem");
delay(700);
//Move a mensagem para a esquerda.
lcd.scrollDisplayLeft();
delay(300);
lcd.scrollDisplayRight();
//Move a mensagem para a direita
delay(300);
lcd.scrollDisplayRight();
delay(300);
}
```

Figura 56 – Exemplo de exibição de mensagem

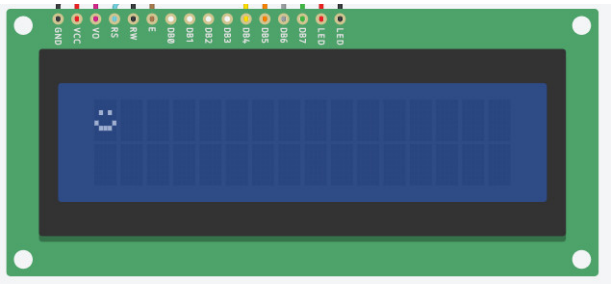
Fonte: Próprio autor

Este outro código é relevante a parte do projeto que cria e utiliza um caractere personalizado.

```
#include <LiquidCrystal.h>

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

//Declara uma variável que especifica quais pixels ficam acesos e quais ficam
apagados no caractere criado.
byte smile[8] = {
B00000,
B00000,
B01010,
B00000,
B10001,
B01110,
B00000,
B00000
};

void setup() {
lcd.clear();
lcd.setCursor(0,0);
//Cria o caractere para ser utilizado no código.
lcd.createChar(1, smile);
lcd.begin(16,2);
//Escreve o caractere no display.
lcd.write((byte)1);
delay(5000);
}

void loop() {
}
```

Dica: para criar o caractere com mais facilidade utilize um site gerador de caracteres, nele você clica nos quadrados que deseja acender ou apagar, e o próprio site gera o código que você copia e cola na variável.

Figura 57 – Exemplo de exibição de caractere especial



Fonte: Próprio autor

9.5 Sugestões de uso

Agora que você aprendeu a utilizar o *display*, chegou a sua hora de utilizá-lo do seu jeito, ele é um componente de apoio, você pode utilizá-lo para monitorar informações de outros componentes como sensores, ou para montar um menu para um projeto mais complexo.

9.6 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 19/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/iqQwer30uSY?t=4589⟩.

10 MEDIÇÃO DE PH

Grupo 09:
Allan de Oliveira Santos - ⟨oliveira.allan1568@gmail.com⟩
Bruna Fontana Vieira - ⟨brubsfontana@outlook.com⟩
Murilo dos Santos Costa - ⟨murilocosta2707@gmail.com⟩
Taina Costa Dias - ⟨tainadiass10@gmail.com⟩

10.1 Este projeto: O que é

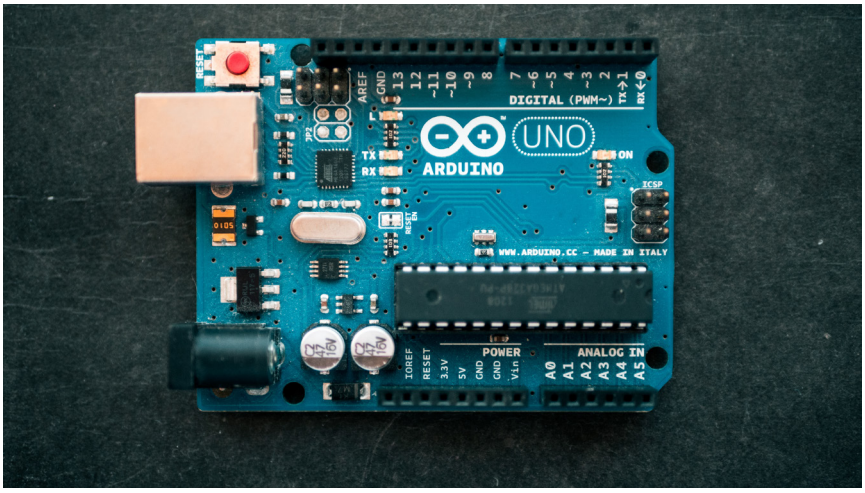
Este projeto é sobre a medição do PH em água e solo e a relação que ele pode ter com o Arduino, que por sua vez possibilita o desenvolvimento de vários projetos robóticos.

Este capítulo é baseado no trabalho de NATIVA (2018).

10.2 Componentes usados

1. **Arduino Uno:** é uma placa de prototipagem;

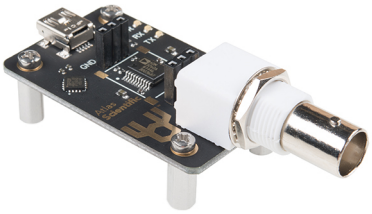
Figura 58 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

2. **Kit Sensor pH (que possui: Interface EZO, Eletrodo de pH para água e solo, Placa com Conector BNC e Soluções de Calibração 4,00; 7,00; 10,00 pH)**;

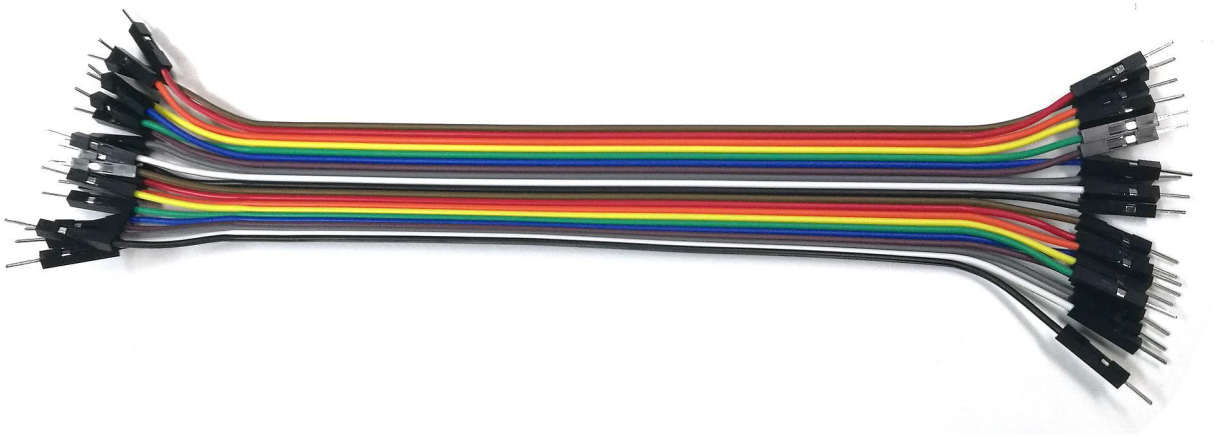
Figura 59 – Placa EZO para sensor de pH



Fonte: ⟨https://www.flickr.com/photos/sparkfun/32401502832/in/photolist-26rMsvB-2iw6Ywp-dVqLn2-dVqkGX-owTYMn-dVwuaQ-RndaWy-Rx5UTA-RndaNs-Rx5UHW-RndaN7-ahNCcj-Kj7bjQ-WYPvoi-ouwXKi-btXXvi-5dMXdy-btXXEM-btXXQz-btXXhD-btXX4z⟩

3. **Cabo *Jumper*:** o cabo *jumper* é um cabo com um conector ou pino em cada extremidade, que é normalmente usado para interconectar os componentes de uma protoboard com um outro componente, neste caso, o Arduino;

Figura 60 – Fios *jumper* macho-macho

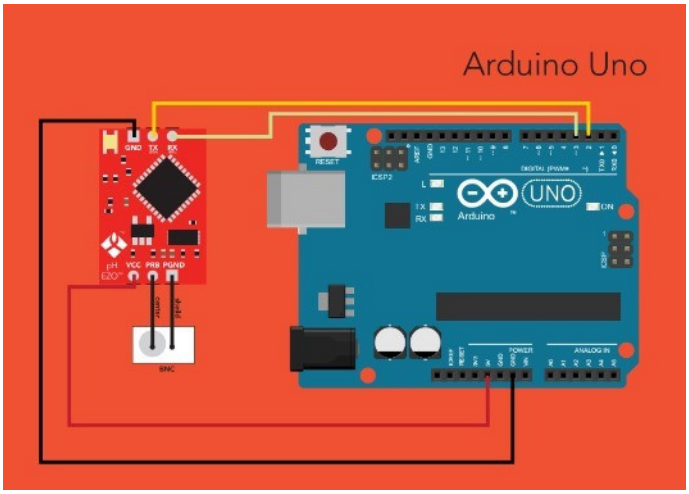


Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

10.3 Esquema de ligações

O Arduino Uno é interligado a placa EZO por meio do cabo *jumper*, essa ligação não utiliza um *protoboard*.

Figura 61 – Esquema de ligações



Fonte: Próprio autor/Adaptado de ⟨https://www.acquanativa.com.br/aplicacoes/kit-sensor-ph-com-arduino-5-passos.html⟩

10.4 Código-fonte

```
#include <SoftwareSerial.h>
#define rx 2
 #define tx 3

SoftwareSerial myserial(rx, tx);
String inputstring = "";
boolean input_string_complete = false;
boolean sensor_string_complete = false;
float pH;
void setup()
 { Serial.begin(9600);
 myserial.begin(9600);
inputstring.reserve(10);
sensorstring.reserve(30);
 }

void serialEvent() {
 inputstring = Serial.readStringUntil(13);
 input_string_complete = true;
 }
```

```
void loop() {
f (input_string_complete) { myserial.print(inputstring);
 myserial.print('\r');
inputstring = "";
input_string_complete = false;
 }

if (myserial.available() > 0) {         /
 char inchar = (char)myserial.read();
 sensorstring += inchar;
 if (inchar == '\r') {
sensor_string_complete = true;
  }
}

if (sensor_string_complete == true) {
 Serial.println(sensorstring);
 /*
 if (isdigit(sensorstring[0])) {
 pH = sensorstring.toFloat();
 if (pH >= 7.0) {
  Serial.println("high");
}
if (pH <= 6.999){
  Serial.println("low");
      }
 }
 */
 sensorstring = "";
sensor_string_complete = false;
  }
 }
```

Definições: no cabeçalho, estão incluídas a biblioteca para comunicação serial SoftwareSerial.h e definidas as funções para os pinos 2 e 3, RX e TX respectivamente. Também são criadas variáveis para adquirir informações do sensor e confirmar seu recebimento.

Parâmetros: abaixo do cabeçalho, são criadas rotinas de setup e inicialização da porta serial. Estas são rotinas importantes e que não devem ser alteradas, inicialmente.

Troca de Informações: no *Loop* do código são tratadas as informações enviadas pelo *Serial Monitor* e pela placa EZO. Caso um comando enviado pelo *Serial Monitor* seja aceito pela placa EZO, ela retorna uma informação. Você pode conferir a lista de comandos aceitos na página 21 do Manual de Operações.

Compilação: clicando em Verificar (*Verify*) no canto superior esquerdo da IDE do Arduino para assegurar que o código será compilado corretamente. Na sequência, em ferramentas (*Tools*), selecione e confirme a porta COM na qual está conectada o Arduino. Faça upload do código para o Arduino.

10.5 Sugestões de uso

Pode ser usado em aulas de Química, e em projetos para auxiliar na medição do pH da água.

10.6 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 12/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/DzjiyOzZOTs?t=3406⟩.

11 SENSOR DE SOM

Grupo 10:
Mateus Tamasia - ⟨mateustamasia3@gmail.com⟩
Eduardo Kohler - ⟨duduflluminenseomelhor@gmail.com⟩
Lucas da Silva Valim - ⟨lucasvalimdas@gmail.com⟩
Iago Haveroth Goerttmann - ⟨991870407iago@gmail.com⟩

11.1 Este projeto: O que é

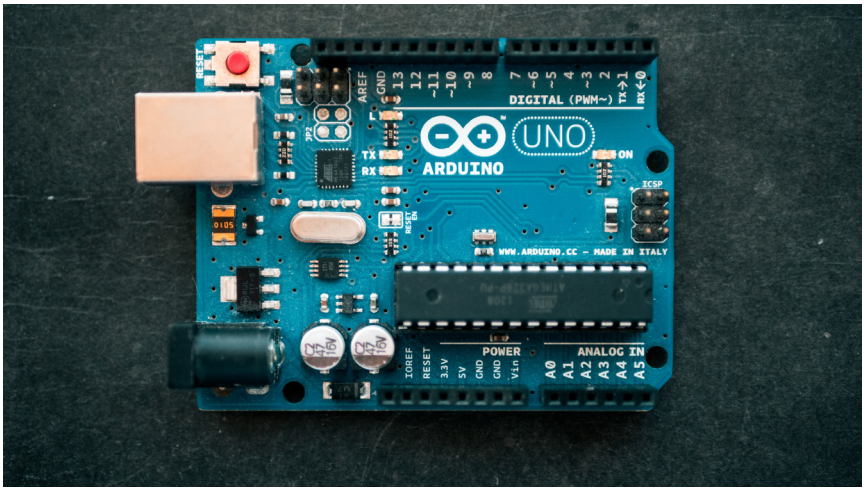
O objetivo desse projeto é detectar o som no ambiente e medir a intensidade do mesmo, através do sensor de som KY-038 e mostrar a intensidade e a presença do mesmo no monitor serial, através do cabo USB. O tempo que as informações são exibidas podem ser alterados.

Este capítulo é baseado no trabalho de CASTRO; CASSIOLI (2020).

11.2 Componentes usados

1. **Placa Arduino:** função de placa micro controladora. O cabo USB é usado para mostrar o resultado no monitor serial. A placa também funciona como fonte de alimentação;

Figura 62 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

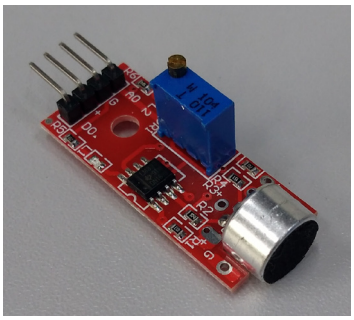
2. **Cabo USB:** ler os dados através da serial;

Figura 63 – Cabo USB tipo B

Fonte: ⟨https://unsplash.com/photos/oXOZAx_MJM4?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

3. **Sensor de som (microfone, etc.):**

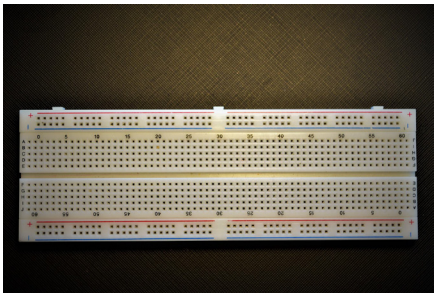
Figura 64 – Sensor de som



Fonte: Acervo pessoal de Josiney de Souza

4. ***Protoboard*:** colocar o sensor;

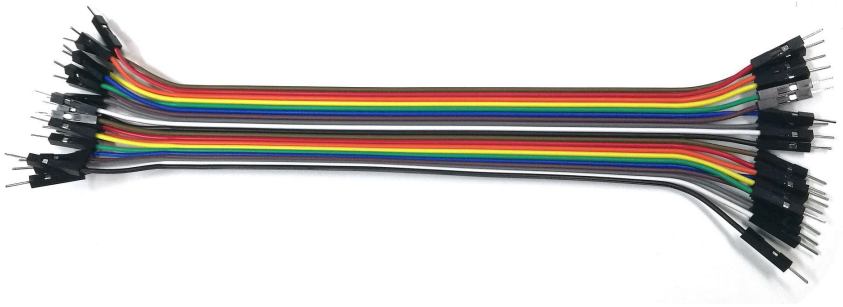
Figura 65 – *Protoboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

5. **Cabo *jumper*:** fazer as ligações;

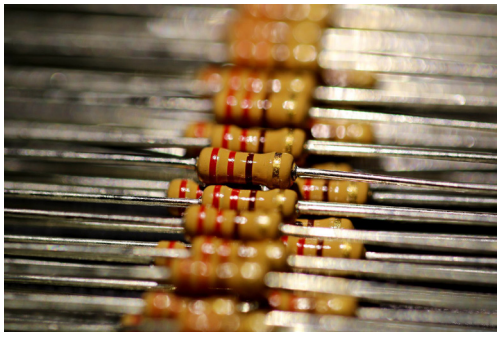
Figura 66 – Fios *jumper* macho-macho



Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

6. **Resistor:** limitar fluxo de cargas elétricas;

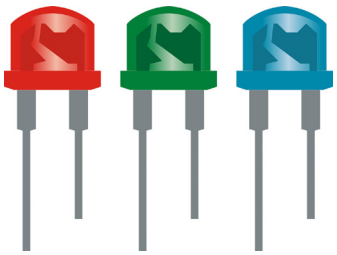
Figura 67 – Resistor



Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

7. ***LED* 5mm vermelho:** mostrar se o Arduino está ligado, funcionando, etc.;

Figura 68 – *LED*

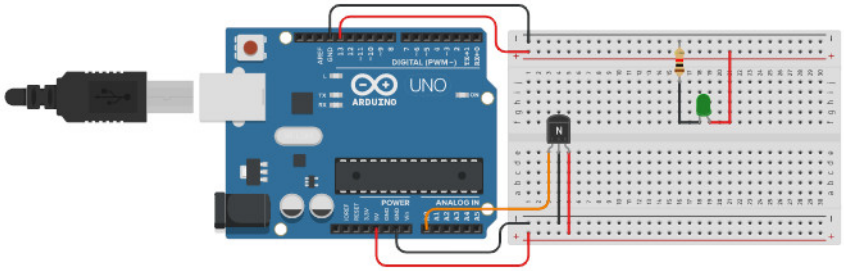


Fonte: ⟨https://pixabay.com/pt/vectors/rgb-conduziu-8mm-l%c3%a2mpadas-luz-2270087/⟩

11.3 Esquema de ligações

Na Figura 69 podemos ver a ligação dos cabos do Arduino, em que o primeiro é a voltagem com o cabo VCC, o segundo é o cabo GND, e o terceiro é a saída analógica.

Figura 69 – Esquema de ligações



Fonte: ⟨https://www.tinkercad.com/things/7oI2iOKaZtc-sensor-de-somled⟩

Caso seja necessário identificar os pinos do sensor, estes são seus significados:

GND terra

VCC tensão 3.3 - 5V = alimentação

DO saída digital, retorna ligado ou desligado (*HIGH* ou *LOW*)

AO saída analógica, retorna intensidade do som identificado (quantidade)

O sensor de som ainda tem a seguinte estrutura:

Microfone utilizado para identificar o som

Potenciômetro utilizado para aumentar ou diminuir a sensibilidade do som

***LED* de alimentação** utilizado para indicar se o sensor está sendo alimentado, se estiver sua luz se acende

***LED* de saída digital** utilizado para indicar se o sensor está captando algum som, se estiver sua luz se acende

11.4 Código-fonte

Na primeira parte definimos a variável PINO_SENSOR que armazena o pino que o sensor está conectado

```
const int PINO\_SENSOR = (PINO que o sensor está conectado);

pinMode(PINO_SENSOR, INPUT); = Configuramos a variável PINO_SENSOR como
entrada do nosso microcontrolador.

Serial.begin(9600); = inicializa a comunicação serial no monitor

Serial.println(analogRead(PINO_SENSOR)); = com isso imprimimos no Plotter
serial a leitura analógica do pino do sensor

delay(50); = botamos um intervalo de tempo entre as leituras de som para
que as informações sejam atualizadas
```

Assim configuramos o Arduino para captar a intensidade do som do ambiente. Podemos utilizar isso na IDE do Arduino desde a versão 1.6.7.x.

11.5 Sugestões de uso

Pode ser utilizado numa biblioteca, para medir a intensidade do som falado, por exemplo, quando a pessoa ultrapassar o volume de voz permitido o sensor aciona.

11.6 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 12/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/DzjiyOzZOTs?t=4683⟩.

12 ÁGUA MUSICAL COM MAKEY MAKEY

Grupo 11:
Ana Flavia Stedile - ⟨anaflaviastedile4321@gmail.com⟩
Sofia Orthmann - ⟨sofiaorthmann15@gmail.com⟩
Irlly Nascimento - ⟨irllynascimento1000@gmail.com⟩

12.1 Este projeto: O que é

Todos nós sabemos que o som da água é um som agradável e relaxante, mas você sabia que a água pode se tornar um piano? Neste projeto vamos lhe mostrar que isso é possível usando o Makey Makey e alguns outros componentes.

Este capítulo é baseado no trabalho de TERRA (2019).

12.2 Componentes usados

1. **Kit Makey Makey:**
   - Placa de controle Makey Makey;
   - Cabo USB A X USB mini;
   - *Jumpers* Garras Jacaré.

Figura 70 – Kit Makey Makey

Fonte: ⟨https://makeymakey.com/⟩

2. **Computador/Notebook:** qualquer um dos dois desde que tenha uma entrada para cabo USB A;

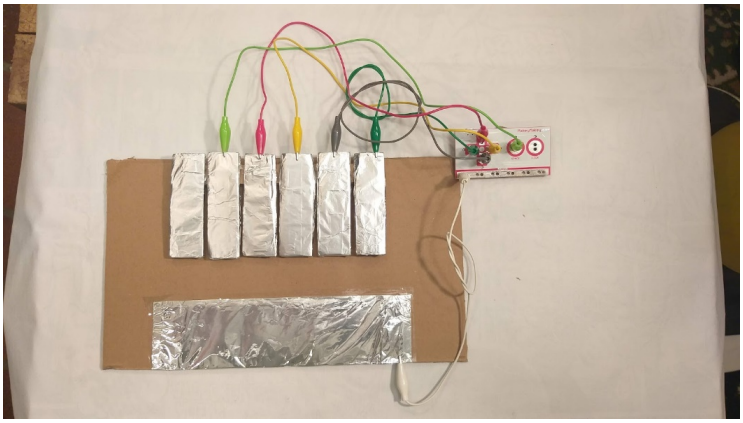
3. **Água:** para colocar dentro das bandejas;

4. **Bandejas de Alumínio:** ou você pode usar pratos revestidos de alumínio;

5. **App On-line de piano Makey Makey:** aplicativo on-line, via ⟨https://apps.makeymakey.com/piano/⟩, que simula um piano a partir do pressionamento de teclas.

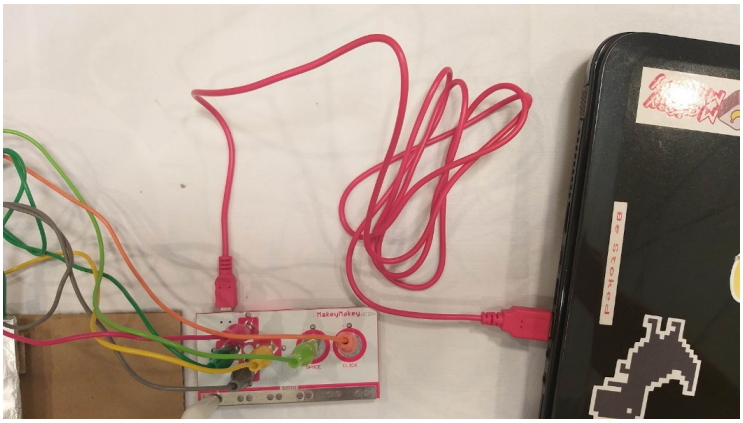
12.3 Esquema de ligações

Figura 71 – Esquema de ligações 1



Fonte: Próprio autor/Adaptado de ⟨https://www.makerzine.com.br/educacao/piano-com-makey-makey-e-scratch/⟩

Figura 72 – Esquema de ligações 2



Fonte: Próprio autor/Adaptado de ⟨https://www.makerzine.com.br/educacao/piano-com-makey-makey-e-scratch/⟩

12.4 Código-fonte

Nosso projeto não apresenta código-fonte, pois a "programação" é feita manualmente na placa do Makey Makey.

12.5 Sugestões de uso

Este trabalho por ser usado tanto para lazer em jogos, quanto para a função educacional.

12.6 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 12/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/DzjiyOzZOTs?t=4062⟩.

13 SENSOR DE NÍVEL DE ÁGUA COM *SHIELD* ETHERNET

Grupo 12:
Jenifer Pühler Suavi - ⟨jeniferpsuavi@gmail.com⟩
Maria Eduarda Kohler Ramos - ⟨kohlerramosm@gmail.com⟩
Mayra Carolina Habitzreuter - ⟨mayrachabitz19@gmail.com⟩
Nicolle Lira - ⟨ellocin2015@gmail.com⟩

13.1 Este projeto: O que é

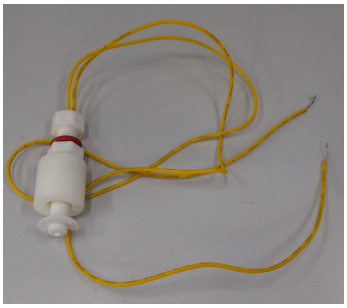
O projeto visa controlar de forma simplificada o nível de água dos reservatórios localizados em ambientes residenciais, estes que geralmente se encontram em locais altos e de difícil acesso. Este projeto combinado com um microcontrolador (Arduino) também é capaz de controlar o nível de água de forma elétrica, o que proporciona ao Arduino, se programado, a capacidade de ativar e desativar a bomba d'água entre outros comandos.

Este capítulo é baseado no trabalho de STRAUB (2016).

13.2 Componentes usados

1. **Sensor de nível de água com boia horizontal:** dispositivo eletrônico capaz de detectar o nível de líquido em um determinado reservatório, pode ser combinado com microcontroladores Arduino e funciona como uma chave magnética que se fecha e conduz corrente elétrica quando a boia horizontal é elevada;

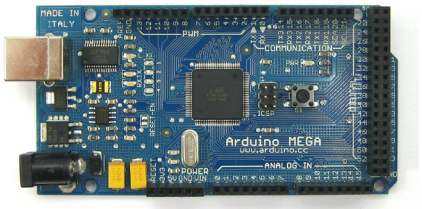
Figura 73 – Sensor de nível de água com boia horizontal



Fonte: Acervo pessoal de Josiney de Souza

2. **Arduino Mega:** seu desempenho é parecido com o do Arduino UNO, porém possuindo mais qualidade e recursos – exemplos: entradas analógicas e saídas PWM. É aplicado em projetos eletrônicos que necessitam de muitos pinos digitais ou analógicos;

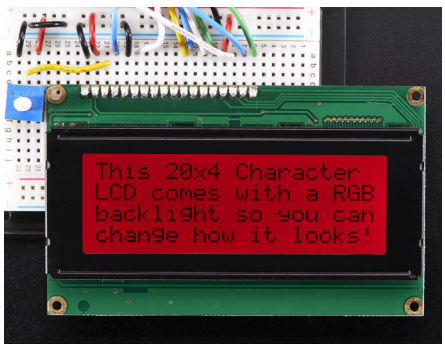
Figura 74 – Arduino Mega



Fonte: ⟨https://www.flickr.com/photos/mellis/4784333051/in/photolist-8hLXwT-6agNNc-7JgAkv-6EHb7k-9iqy2W-nwdfge-4qnzqY-6EqNm5-6EmCdK-6EmCWD-6EmCCR-6EqMYL-6EqMNW-95PtTT-95Sx21-95Sx3G-dvsjkS-adQqbf-adQqbh-9FkDWW-9FkEzS-6ahK43-6ahK5u-6ahHFu-6adxQp-8VFv4r-6bo6Kf-dvS7eQ-6ahHqW-f2WZjL-6ady1k-6ady6v-6BnKg7-73yC9Z-6dHVgi-fgyU53-btG9Rk-73CUi9-8Y4p7Q-9GhEwA-GkWGme-GcMyfY-9iqpQ7-9iniqe-9iqn5j-9iqmbL-9inhtV-e3oTuU-6g1HgQ-8H5n8M⟩

3. ***Display LCD*:** usado em projetos que se tenha a necessidade de visualizar a leitura de um sensor ou transmitir informações para o usuário;

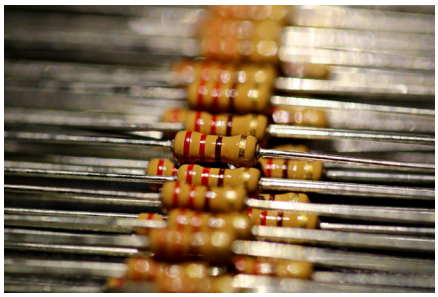
Figura 75 – *Display LCD*



Fonte: ⟨https://www.flickr.com/photos/adafruit/17572014992/in/photolist-sLMaRJ-rQaMVe-sLMphG-suoCZu-pqdMbd-5bPtJj-kaLZX-5uL15p-28W72tR-rPYb5j-wFKKcj-suorFs-sLZ9MD-sLZ324-w2mcyE-suwagD-suw1nr-46TgVD-9fRJhz-fJpNvP-pYAajq-butLFK-sLZ73v-4m4igU-suopyw-ydrRfY-sJDZBL-suw8Mr-5uQn5s-5KV3Kk-pG4NaJ-sLZ8hV-rQaFu8-suw8YZ-sLYHMz-2kxPHVK-2nm4WD6-2j1SfCA-2mZYuwP-KJjLBv-4mGuxe-562gNk-2gm61nq-5xcWHj-suoxDj-sLYQWZ-suvMV2-sLZ5Ln-suor69-46TgJv⟩

4. **Resistor 10KΩ:** dispositivo muito utilizado em equipamentos elétricos e circuitos eletrônicos. Apresenta funções importantes e variadas - exemplos: gerar calor, limitar a corrente elétrica e produzir queda de tensão;

Figura 76 – Resistor

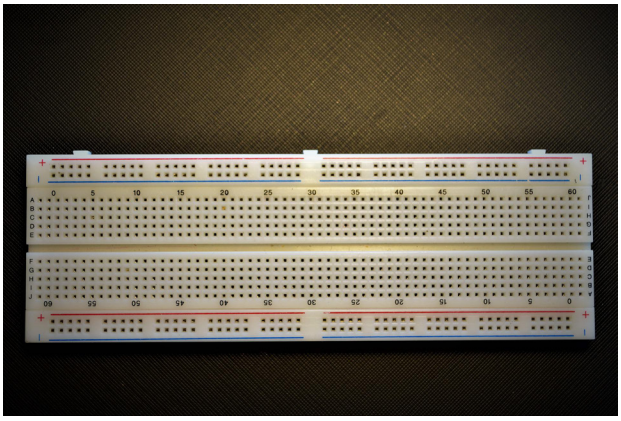


Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

5. **Módulo Relé:** são programados para abrir ou fechar uma corrente elétrica, permitindo ou bloqueando a passagem do fluxo de eletricidade. Seu acionamento ocorre quando uma bobina é estimulada por meio elétrico;

6. ***Protoboard*:** placa que possui furos e conexões internas, servindo para a montagem de circuitos, principalmente quando desejamos testá-los;

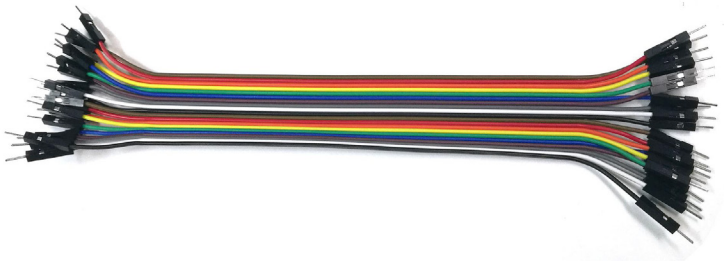
Figura 77 – *Protoboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

7. **Cabos *jumper*:** fios elétricos com pontas preparadas para fazer as conexões entre os componentes de um circuito, possibilitando a condução elétrica ao longo dele.

Figura 78 – Fios *jumper* macho-macho

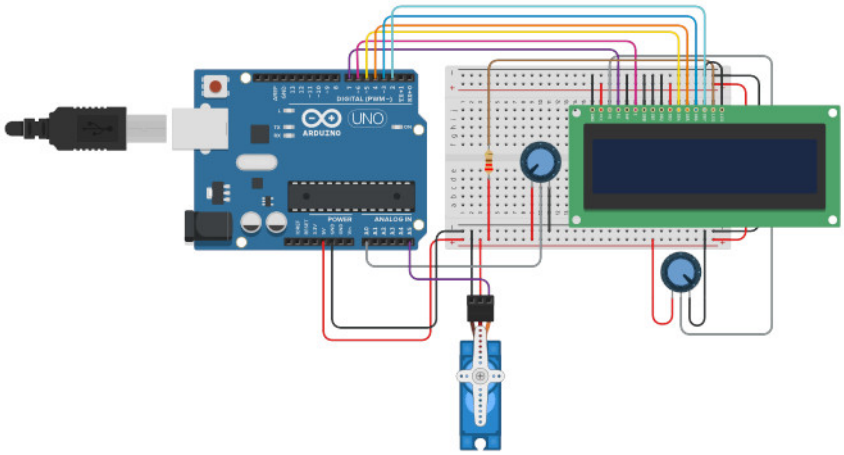


Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

13.3 Esquema de ligações

Para conseguirmos utilizar o sensor de nível de água é necessário utilizar um resistor. O resistor serve para deixar o pino em que ele está ligado em nível lógico baixo, servindo para que o Arduino evite fazer detecções erradas, conseguindo assim, fazer uma detecção precisa. É importante ressaltar que os fios do sensor não têm posição correta, por isso, não há o risco de troca de ordem dos fios.

Figura 79 – Esquema de ligações



Fonte: ⟨https://www.tinkercad.com/things/41eOGr7CCHc-boia-de-caixa-dagua⟩

13.4 Código-fonte

Abaixo encontram-se os dois códigos utilizados para programar funções do projeto.

Parte 1:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

int PinSensor = 8;
int VarSensor = 0;

void setup()
{
lcd.begin(16, 2);
lcd.setCursor( 0, 0);
lcd.println(" Nivel Agua ");
lcd.setCursor( 0, 1);
lcd.println("Em caixa D'Agua ");
delay(3000);
lcd.clear();
pinMode(13, OUTPUT);
}
```

Parte 2:

```
 2. void loop()
{
VarSensor = digitalRead(PinSensor);
lcd.setCursor( 0, 0);
lcd.println("Sensor de Nivel ");
lcd.setCursor( 0, 1);
lcd.println("---> Cheio! <---");

if (VarSensor == 1)
{
lcd.clear();
lcd.setCursor(0, 0);
lcd.println(" Caixa Vazia ");
lcd.setCursor(0, 1);
lcd.println("--> Enchendo <--");
```

```
digitalWrite(13, HIGH);

}

else
{
digitalWrite(13, LOW);
}
delay(500);
}
}
```

13.5 Sugestões de uso

Após pesquisas, sugerimos a instalação de um sensor de nível de água na caixa d'água da nossa instituição de ensino - Instituto Federal Catarinense *campus* Brusque. Como também em todas as residências e empresas que desejam facilitar o controle da quantidade de água que possuem em suas caixas.

13.6 Apresentação no canal de *YouTube* do *campus*

Para mais informações, assista a apresentação oral gravada do nosso projeto que ocorreu no dia 05/08 e está disponível no canal oficial do Instituto - IFC Brusque. Ou acesse o link: ⟨https://youtu.be/HF8c3yjI0D0?t=4879⟩.

14 *DIMMER* DE UMA LÂMPADA

Grupo 13:
Bruno Ricardo Piaz - ⟨brunor.piaz@gmail.com⟩
Luiz Gustavo Dionísio Martinho - ⟨luizgustavodm14@gmail.com⟩
Lael Pereira - ⟨noglayol@gmail.com⟩
Maiara de Souza Mazzini - ⟨maimazzinii25@gmail.com⟩
Luana Heloisa Stedile - ⟨stedileluanaheloisa@gmail.com⟩
Thiago Prudencio Correia - ⟨thiagoprudencio91@gmail.com⟩

14.1 Este projeto: O que é

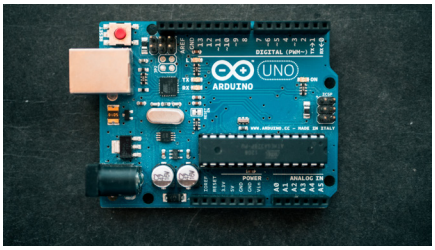
O nosso projeto teve como intuito principal apresentar o que é e como funciona um Arduino. Primeiramente o que é o Arduíno? É uma placa de prototipagem eletrônica propícia a servir a diversos projetos. Mas no nosso projeto o utilizamos para fazer a ligação de um simples *LED*, e como fizemos isso? Realizando devidas ligações entre um Arduino ligado a 4 cabos jumpers conectados a uma placa de ensaio pequena, um resistor, um potenciômetro e um cabo USB. E que devido ao código de execução, a *LED* é programada para acender e apagar uma vez a cada 3 segundos.

Este capítulo é baseado no trabalho de SOUZA (2021).

14.2 Componentes usados

1. **Arduino e cabo de alimentação/dados:** Arduino é uma placa de prototipagem eletrônica de código aberto. O cabo USB está sendo utilizado para transmitir o código-fonte para o Arduino e também sendo utilizado como fonte de alimentação;

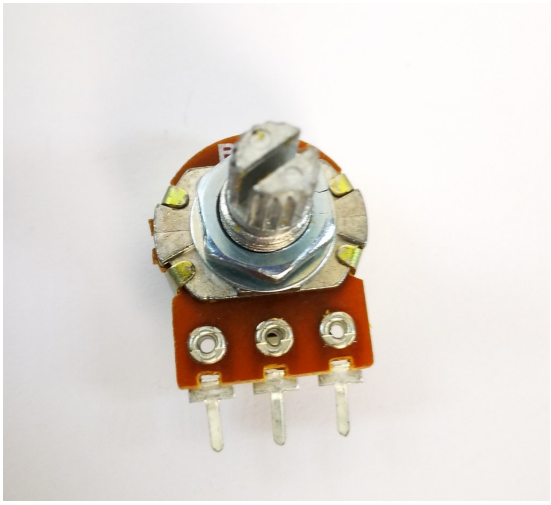
Figura 80 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

2. **Potenciômetro:** é o controlador da resistência que é enviada para o *LED*;

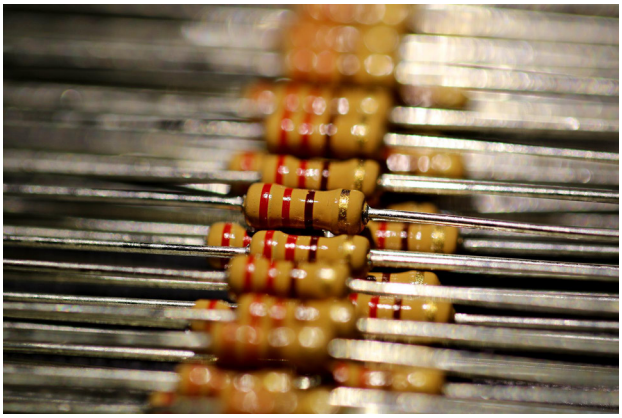
Figura 81 – Potenciômetro



Fonte: ⟨https://www.flickr.com/photos/161851377@N08/42193851721/in/photolist-FUsduJ-Fp7uL9-5NwodA-gWfSHA-CsNbQ-28WEbu-kmYyju-kmWc5R-aNLWzM-kmWa9B-29symT-kmYz5C-kmX5Zk-BBVpkz-kmX5v4-dpSUth-kmW9kc-dvziTU-kmWbdR-7h4fK2-27hww76-7bnFsu-4WeyN9-6t4Q9o-kmYEnw-4RVkax-7NCgto-2tdRFu-9t8yuz-5BeEv2-pwQCxC-2c9NnWt-VjtLEE-pwToUw-HJmon1-kntyRV-Yj8RJH-2aveXcZ-x2gpRE-uAJD2S-Rw1ygG-eimuiF-ai1Rp5-a83EzQ-4RZwFS-knvQfA-kmWfVM-4RVkVa-7qUWQj-knujDc⟩

3. **Resistor:** componente elétrico com a função de evitar uma sobrecarga de energia;

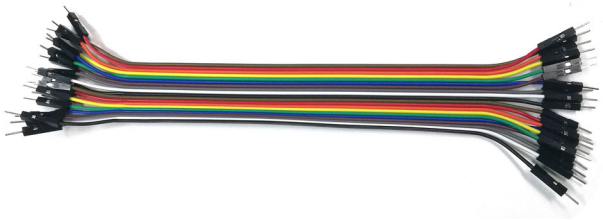
Figura 82 – Resistor



Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

4. **Cabos *jumper*/fio:** utilizados para realizar a transmissão de dados aos sensores e principais componentes;

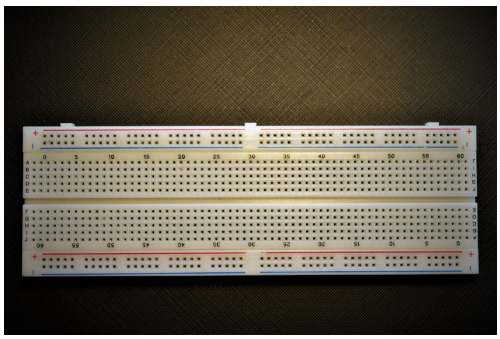
Figura 83 – Fios *jumper* macho-macho



Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

5. **Placa de ensaio pequena (*protoboard*):** suporte para realizar ligações entre componentes;

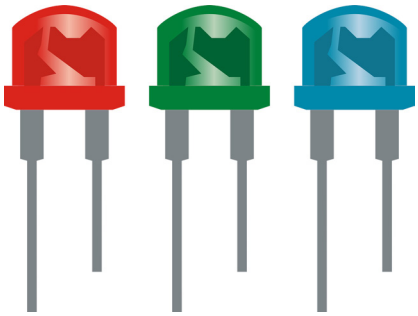
Figura 84 – *Protoboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

6. ***LED*:** é um dispositivo capaz de emitir luz de forma eficiente e econômica;

Figura 85 – *LED*

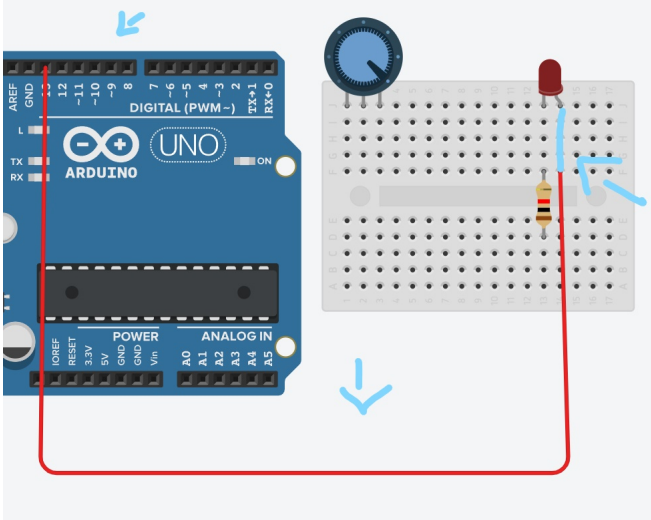


Fonte: ⟨https://pixabay.com/pt/vectors/rgb-conduziu-8mm-l%c3%a2mpadas-luz-2270087/⟩

14.3 Esquema de ligações

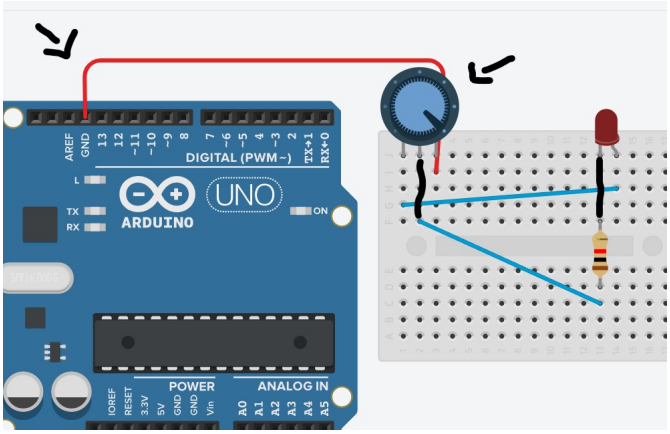
O D13 do arduino está ligado e passando energia para a parte positiva do *LED* (Figura 86). Já na parte negativa do *LED*, passa pelo resistor e chega ao potenciômetro (Figura 87). Ao chegar no potenciômetro, essa energia se fecha, e a ligação que está no GND, faz com que o *LED* se acenda (Figura 88).

Figura 86 – Esquema de ligação - D13 ligado à parte positiva do *LED*



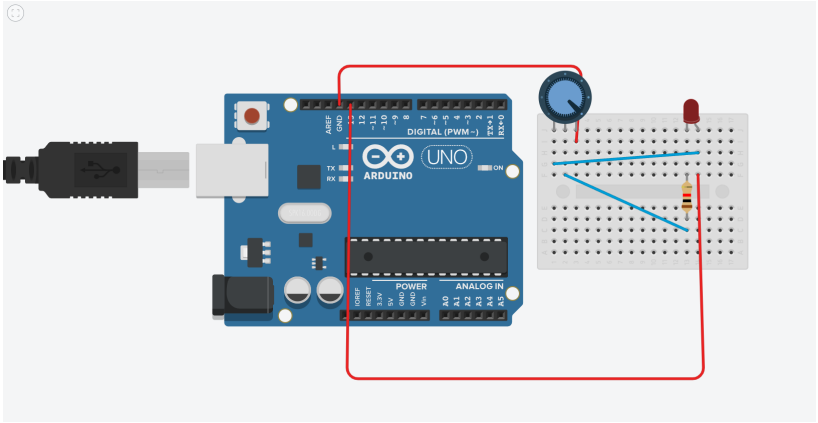
Fonte: próprio autor/Adaptado de Tinkercad.

Figura 87 – Esquema de ligação - GND ligado à parte negativa do *LED*, passando pelo resistor e chegando ao potenciômetro



Fonte: próprio autor/Adaptado de Tinkercad.

Figura 88 – Esquema completo de ligações



Fonte: próprio autor/Tinkercad.

14.4 Código-fonte

Abaixo temos um código-fonte simples, para fazer o pino 13 (pinMode(13, OUTPUT)) apagar e acender uma vez a cada três segundos (delay(3000); // Wait for 3000 millisecond(s)). Também utilizando o código loop (void loop) para repetir todo o processo.

```
// C++ code
//
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);
  delay(3000); // Wait for 3000 millisecond(s)
  digitalWrite(13, LOW);
  delay(3000); // Wait for 3000 millisecond(s)
}
```

14.5 Sugestões de uso

O nosso trabalho (Dimmer de uma lâmpada) tem diversas utilidades e entre elas, poderíamos utilizar algumas em nosso cotidiano como: utilizado em salas de estar, dormitórios,

escritórios, sensores (para o uso em janelas e portas automáticas), alarmes, lâmpadas. Além de serem instalados como interruptor controlável na parede, os dimmers mais atuais também dispõem de controles remoto.

14.6 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 12/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/DzjiyOzZOTs?t=5443⟩.

15 COMO CONTROLAR UM MOTOR DE PASSO 5V COM ARDUINO

Grupo 15:
André Jair Heckert Junior - ⟨heckertjuninho@gmail.com⟩
George Valtrich - ⟨franckvaltrich@gmail.com⟩
Guilherme Henrique Macena - ⟨macenagui1@gmail.com⟩

15.1 Este projeto: O que é

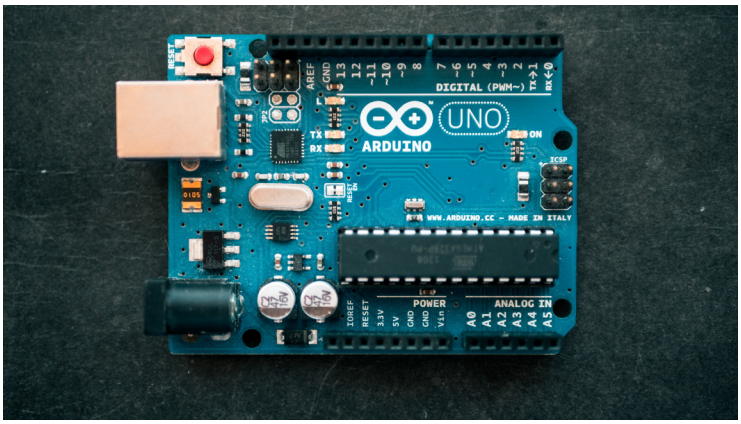
Este projeto visa ensinar a controlar da maneira mais simples, a controlar a rotação do eixo de um motor de passo 5V utilizando a placa Arduino Uno. Controlando esse eixo, nós podemos utilizar o tal motor para algumas funções não tão conhecidas, mas bastante úteis. Ele permite girar seu eixo em angulações específicas, como 45°, 90°, 120° e 180°, por exemplo.

Este capítulo é baseado no trabalho de THOMSEN (2013).

15.2 Componentes usados

1. **Arduino Uno:** placa microcontroladora usada para se trabalhar com protótipos;

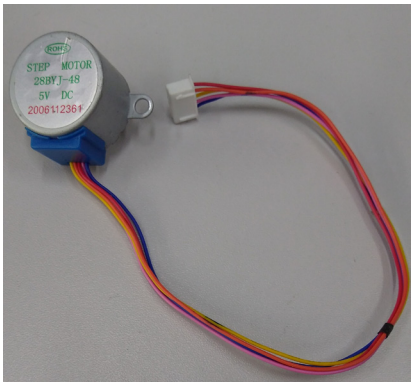
Figura 89 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

2. **Motor de Passo 5V:** dispositivo eletromecânico que converte os impulsos elétricos em movimentos discretos mecânicos;

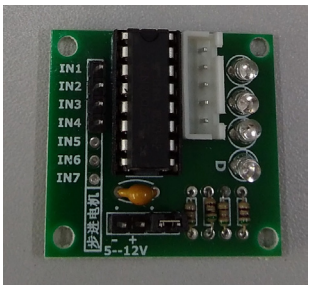
Figura 90 – Motor de passo



Fonte: Acervo pessoal de Josiney de Souza

3. ***Driver* Uln2003:** *driver* de corrente que permite o Arduino controlar motores com correntes superiores a 50 mA, neste caso até 500mA;

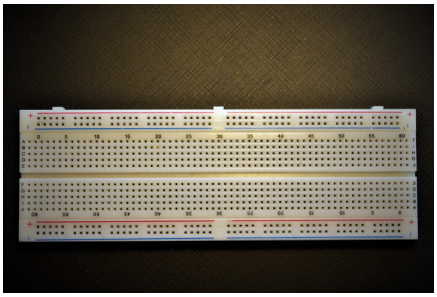
Figura 91 – *Driver* para motor de passo



Fonte: Acervo pessoal de Josiney de Souza

4. ***PROTOBOARD*:** placa com furos e conexões condutoras utilizada para a montagem de protótipos e projetos em estado inicial;

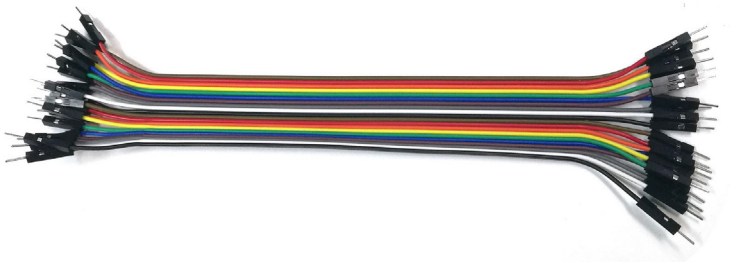
Figura 92 – *Protoboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

5. **CABO *JUMPER* MACHO-MACHO:** um *jumper* é uma peça plástica que contém um pequeno filamento de metal responsável pela condução de eletricidade;

Figura 93 – Fios *jumper* macho-macho



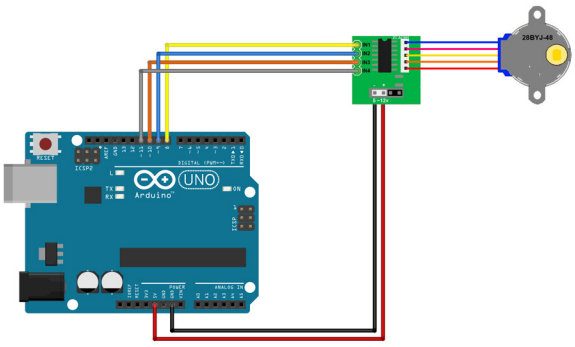
Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

15.3 Esquema de ligações

A Figura 94 mostra um esquema de ligações feito utilizando as ferramentas do simulador Fritzing. Iniciamos ligando o motor de passo 5V na placa Arduino, usando como intermediário, o módulo driver Uln2003. Nele ligamos os cabos aos respectivos pinos: 11 ao IN4, 10 ao IN3, 9 ao IN2 e o pino 8 ao IN1. Em seguida, para fornecer energia ao módulo e ao motor, conectamos os cabos *jumper* macho-macho no Arduino (provedor de energia) nos pinos inferiores mostrados na imagem, GND (fio preto) e o 5V (fio vermelho), nas entradas do módulo, 5 12V. Com isso, teremos apenas que juntar o nosso motor de passo do modelo 28BYJ-48, utilizando os próprios fios agregados nele, conectando os pinos A, B, C, D e E.

Obs: A fonte de alimentação fornecida pelo Arduino, virá do USB conectado a ele que o liga na nossa máquina programadora.

Figura 94 – Esquema de ligações



Fonte: Próprio autor/Fritzing

15.4 Código-fonte

Abaixo está o código-fonte criado para a IDE do Arduino que faz a coleta e exibição de dados. Este código não é de autoria própria, foi pesquisado unicamente para mostrar como conseguimos controlar e girar com precisão o eixo do motor de passo. Ele irá fazer com que o motor gire seu eixo em angulações prédeterminadas.

```
#include <Stepper.h>

const int stepsPerRevolution = 500;

//Inicializa a biblioteca utilizando as portas de 8 a 11 para
//ligação ao motor
Stepper myStepper(stepsPerRevolution, 8,10,9,11);

void setup()
{
 //Determina a velocidade inicial do motor
 myStepper.setSpeed(60);
}

void loop()
{
 //Gira o motor no sentido horário a 90 graus
 for (int i = 0; i<=3; i++)
 {
 myStepper.step(-512);
 delay(2000);
 }

 //Gira o motor no sentido anti-horário a 120 graus
 for (int i = 0; i<=2; i++)
 {
 myStepper.step(682);
 delay(2000);
 }

 //Gira o motor no sentido horário, aumentando a
```

```
 //velocidade gradativamente
 for (int i = 10; i<=60; i=i+10)
 {
 myStepper.setSpeed(i);
 myStepper.step(40*i);
 }
 delay(2000);
}
```

15.5 Sugestões de uso

Como é um motor, podemos esperar que suas funções não sejam muito caseiras, ou seja, já que ele nos possibilita controlar com maestria e precisão o ângulo de seu eixo, a maior parte de suas utilidades são industriais.

Na indústria, essa tecnologia possui uma infinidade de aplicações, possibilitando a automação de processos e proporcionando inovação tecnológica, economia, qualidade e segurança na produção. Dentre as muitas aplicações industriais, temos: Routers CNC, Máquinas de Corte a Plasma ou a Laser, Controle de Válvulas, Rotuladoras, Mesas de Posicionamento, Braços Manipuladores, Atuadores Lineares, etc.

15.6 Apresentação no canal de *YouTube* do *campus*

O nosso projeto foi apresentado na disciplina de Hardware e Sistemas Operacionais do professor Josiney no dia 19/08/2021. A apresentação está no canal do *YouTube* do IFC *campus* Brusque e está localizada no link: ⟨https://youtu.be/iqQwer30uSY?t=795⟩.

16 SEMÁFORO PARA CARROS E PEDESTRES

Grupo 16:
Guilherme Ricci - ⟨guilhermericci1445@gmail.com⟩
Jhonatan Mateus - ⟨bia.dkj@gmail.com⟩
Kallyandra Gorisch - ⟨Kallyandra.gorisch@gmail.com⟩
João Cervi - ⟨joao.erbrecht23@gmail.com⟩

16.1 Este projeto: O que é

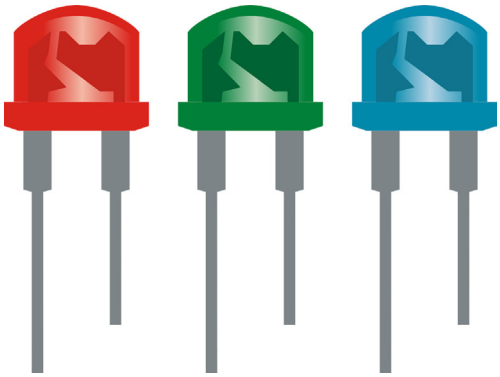
Semáforo é um sinal de trânsito que funciona como um instrumento de controle do tráfego de automóveis e pedestres nas estradas. O semáforo serve para auxiliar os motoristas e pedestres a se locomoverem com cautela nas vias de circulação das cidades.

Este capítulo é baseado no trabalho de FILIPEFLOP (2022b).

16.2 Componentes usados

1. **2x *LED* Vermelho 5mm:**
2. **2x *LED* Verde 5mm:**
3. **1x *LED* Amarelo 5mm:**

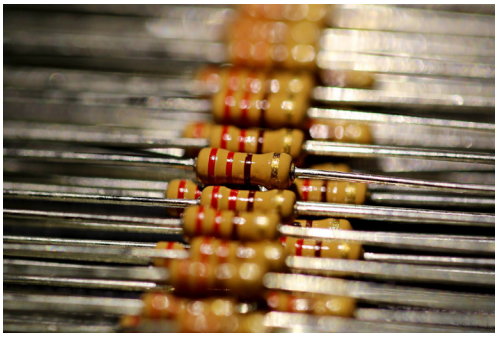
Figura 95 – *LED*



Fonte: ⟨https://pixabay.com/pt/vectors/rgb-conduziu-8mm-l%c3%a2mpadas-luz-2270087/⟩

4. **5x Resistor 220 ohms:**

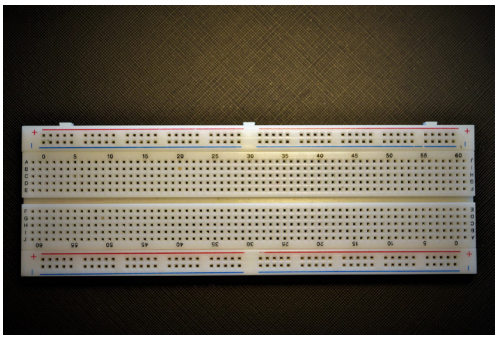
Figura 96 – Resistor



Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

5. **1x *Protoboard*:**

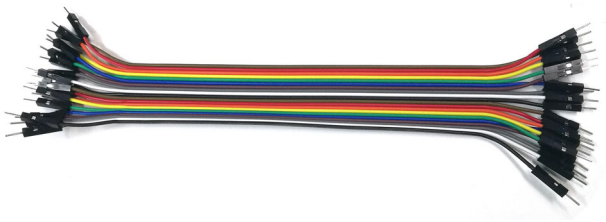
Figura 97 – *Protoboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

6. **11x *Jumper* macho-macho:**

Figura 98 – Fios *jumper* macho-macho



Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

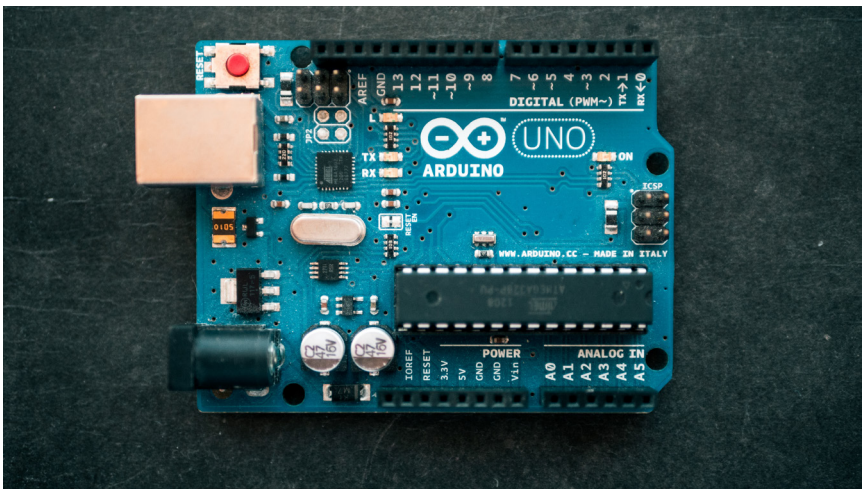
7. **1x Cabo USB:**

Figura 99 – Cabo USB tipo B

Fonte: ⟨https://unsplash.com/photos/oXOZAx_MJM4?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

8. **1x Placa Uno:**

Figura 100 – Placa Arduino Uno

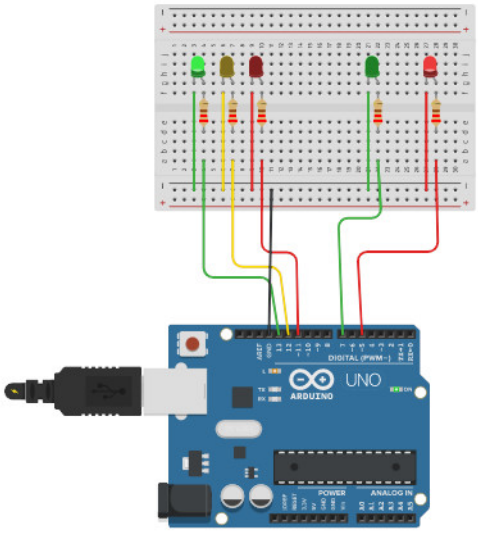


Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

16.3 Esquema de ligações

O esquema de ligação dos componentes está representado na Figura 101.

Figura 101 – Esquema de ligações



Fonte: ⟨https://www.tinkercad.com/things/bOuEJoiQkPp-ale-semaforo-⟩

16.4 Benefícios

Durabilidade a utilização da tecnologia *LEDs* diminui a necessidade de troca das lâmpadas.

Economia reduzido custo e necessidade de manutenção, o equipamento também se destaca por possuir baixo consumo de energia elétrica.

Segurança o sistema fornece uma clara noção de tempo aos condutores e pedestres. É também uma solução para pedestres daltônicos.

Visibilidade o uso de *LEDs* proporciona visibilidade superior aos equipamentos tradicionais pela intensidade de seu brilho. Pode-se escolher a quantidade de *LEDs* definindo o diâmetro dos módulos entre 200 ou 250m.

Flexibilidade a quantidade e a intensidade dos *LEDs* podem ser customizadas. O equipamento possui um sistema articulado e pode ser direcionado para a posição desejada.

16.5 Funcionamento

Pressione o botão para iniciar o nosso semáforo.

O programa irá iniciar com o semáforo no verde, para permitir que os carros passem, e a luz para pedestres no vermelho.

O semáforo para carros vai do verde para o amarelo e depois para o vermelho, e então o semáforo para pedestres vai para o verde.

Depois de um intervalo de tempo, a luz verde para pedestre deverá piscar para avisar aos pedestres que devem atravessar rapidamente antes que o semáforo feche.

Então, a luz vermelha do semáforo do pedestre ascende e a luz dos carros vai do vermelho para o verde, permitindo novamente o fluxo do tráfego.

16.6 Código-fonte

```
// Projeto 4 - Semaforo

int pedVerde = 9; // Define os pinos que serao utilizados
int pedVermelho = 8;
int carroVerde = 12;
int carroAmarelo = 11;
int carroVermelho = 10;
```

int, como acima, é para indicar uma variável inteira, que no caso vai definir os pinos que vao ser utilizados.

```
void setup() // Define os pinos como saidas
{
  pinMode(pedVerde, OUTPUT);
  pinMode(pedVermelho, OUTPUT);
  pinMode(carroVerde, OUTPUT);
  pinMode(carroAmarelo, OUTPUT);
  pinMode(carroVermelho, OUTPUT);

  digitalWrite(carroVerde, HIGH); // Coloca na posicao inicial. Somente o verde
      dos carros e o vermelho dos pedestres acesos
  digitalWrite(carroAmarelo, LOW);
  digitalWrite(carroVermelho, LOW);
  digitalWrite(pedVerde, LOW);
  digitalWrite(pedVermelho, HIGH);
}
```

void setup, como acima, serve para configurar os pinos da placa, e é executada apenas uma vez.

```
void loop()
{
```

```
  digitalWrite(carroVerde, HIGH); // Acende o verde dos carros e o vermelho dos
      pedestres
  digitalWrite(pedVermelho, HIGH);
  delay(5000); // Aguarda 5 segundos
  digitalWrite(carroVerde, LOW);
  digitalWrite(carroAmarelo, HIGH); // apaga o verde dos carros e acende o amarelo
  delay(3000); // aguarda mais 3 segundos
  digitalWrite(carroAmarelo, LOW); // apaga o amarelo dos carros e acende o
      vermelho
  digitalWrite(carroVermelho, HIGH);
  digitalWrite(pedVermelho, LOW); // apaga o vermelho dos pedestres e acende o
      verde
  digitalWrite(pedVerde, HIGH);
  delay(5000); // aguarda mais 5 segundos
  digitalWrite(pedVerde, LOW);
  for(int x = 0; x<5; x++) // Pisca o vermelho dos pedestres
  {
    digitalWrite(pedVermelho, HIGH);
    delay(250);
    digitalWrite(pedVermelho, LOW);
    delay(250);
  }
  digitalWrite(carroVermelho, LOW);
}
```

void loop, como acima, é uma função que executa os comandos que são colocados nela infinitamente. E serve para configurar um pino como sendo saída ou entrada.

while é um loop de repetição que vai continuar repetindo IF, serve para selecionar a parte do código que será executada e também serve para alterar o fluxo.

for também é um loop de repetição.

16.7 Sugestões de uso

O lugar que esse projeto poderia funcionar na escola seria algum lugar com fila, por exemplo na cantina ou na própria entrada do IFC.

16.8 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 19/08/2021. A apresentação foi transmitida pelo canal

de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://www.youtube.com/watch?v=iqQwer30uSY&t=1306s⟩.

17 TECLADO NUMÉRICO ARDUINO

Grupo 17:
Cauã de Vargas Santos - ⟨extragg0@gmail.com⟩

17.1 Este projeto: O que é

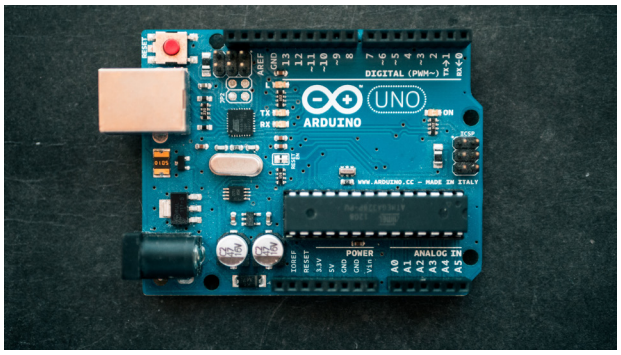
Este projeto do Teclado Matricial Arduino, é mostrar o que é, como funciona, quais materiais utilizar, e em que usar. Meu objetivo é tentar ajudar o máximo que conseguir, com as informações que estavam disponíveis, não só ajudar os outros a compreender este conteúdo, mas eu também aprendi mais sobre este tipo de componente.

Este capítulo é baseado no trabalho de PORTA (2016).

17.2 Componentes usados

1. **Arduino Uno R3:** o Arduino é uma plataforma de prototipagem eletrônica open-source que se baseia em hardware e software flexíveis e fáceis de usar. É destinado a artistas, designers, hobbistas e qualquer pessoa interessada em criar objetos ou ambientes interativos;

Figura 102 – Placa Arduino Uno

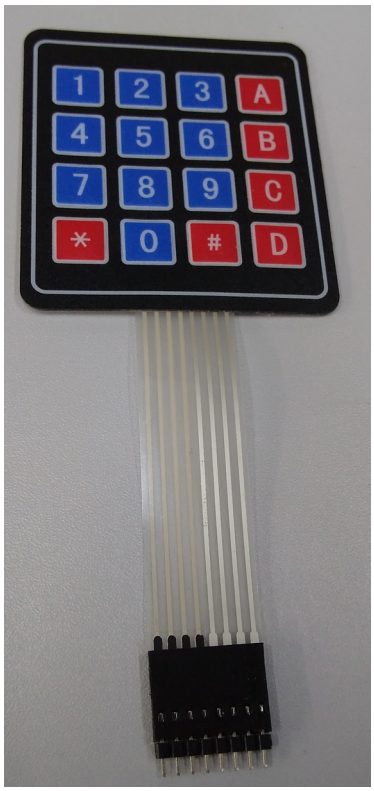


Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

2. **Teclado Matricial de Membrana:** o Teclado Matricial de Membrana 4X4 com 16 teclas foi desenvolvido com a finalidade de facilitar a entrada de dados em projetos com plataformas microcontroladas. Este teclado possui 16 teclas, onde 10 teclas são

numerais, 4 literais e 2 de caracteres. As 16 teclas estão dispostas em 4 linhas por 4 colunas e o teclado possui um conector de 8 pinos para ligação;

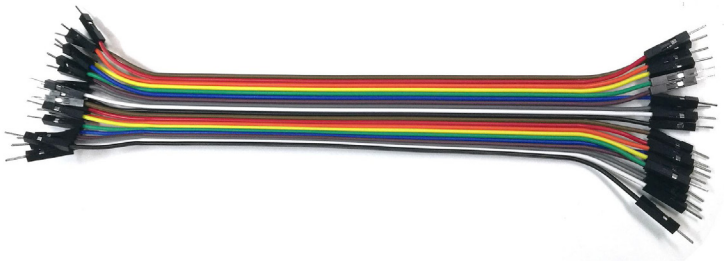
Figura 103 – Teclado matricial 16 teclas



Fonte: Acervo pessoal de Josiney de Souza

3. **Kit de *Jumpers*:** os *Jumpers* são pequenos fios condutores que podem ser conectados a uma protoboard para interligar dois pontos do circuito em projetos eletrônicos, geralmente utilizados em conexões com Arduino, Raspberry Pi, entre outros.

Figura 104 – Fios *jumper* macho-macho



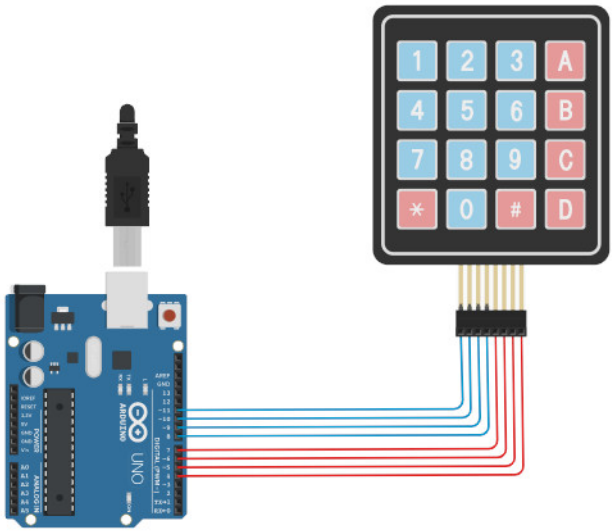
Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

17.3 Esquema de ligações

O sistema de multiplexação é um sistema básico que funciona da seguinte forma: o Arduino define que todas as colunas fiquem como saída, e as linhas como entrada, assim aguardando o pressionamento de algum botão, se algum botão for pressionado, instantaneamente o microcontrolador troca as entradas pelas saídas e novamente verifica qual botão foi pressionado, assim ao juntar as duas informações teremos a localização do botão apertado pelas coordenadas das linhas pelas colunas.

Para perfeito funcionamento do projeto é necessário que as ligações sejam feitas de forma correta, obedecendo as ligações conforme a imagem (Figura 105). Para conexão do Arduino com o teclado podem ser utilizados *jumpers* do tipo macho-macho, especialmente coloridos para evitar a ligação incorreta.

Figura 105 – Esquema de ligações



Fonte: ⟨https://www.tinkercad.com/things/1dwrB00rZNS-teclado-matricial-4x4⟩

17.4 Código-fonte

```
/**********************************
 ***Autor: Leonardo Dalla Porta****
 ***********25/05/2014************
 ************Atenção*************
 ***O Codigo esta livre para uso,**
 *desde que seja mantida sua fonte*
```

```
 ********e seu autor.**************
 ********Faça um bom uso************
 ******Att. Equipe UsinaInfo*******
 *********************************/

#include <Keypad.h>

const byte Linhas = 4; //4 Linhas
const byte Colunas = 4; //4 Colunas

//Define o caractere para cada tecla
char hexaKeys[Linhas][Colunas] = {
  {
    '1','2','3','A'  }
  ,
  {
    '4','5','6','B'  }
  ,
  {
    '7','8','9','C'  }
  ,
  {
    '*','0','#','D'  }
};

byte PinoLinha[Linhas] = {
  9, 8, 7, 6}; //Pinos que serao conectados as linhas do teclado
byte PinoColuna[Colunas] = {
  5, 4, 3, 2}; //Pinos que serao conectados as colunas do teclado
```

Acima foi incluído a biblioteca do teclado matricial Keypad.h. Nele foram definidos o número de linhas e colunas do teclado.

```
//Cria um Objeto para o teclado
Keypad Teclado = Keypad( makeKeymap(hexaKeys), PinoLinha, PinoColuna, Linhas,
    Colunas);

void setup()
```

```
{
  Serial.begin(9600);
  Serial.println("www.usinainfo.com.br");
}

void loop(){
  char Valor = Teclado.getKey();

  if (Valor){
    Serial.println(Valor);
  }
}
```

Acima foi criado um objeto Keypad (teclado) com o mapa do teclado mostrado anteriormente, os pinos que serão utilizados no Arduino para a ligação das vias de linhas e colunas do Teclado Matricial. Em void setup, foram feitas as configurações de taxa de comunicação serial e a mensagem inicial no monitor serial. No programa principal a função getKey verifica qual tecla foi pressionada do objeto Keypad e armazenada na variável Valor. A tecla pressionada aparecerá no monitor serial ao lado da mensagem "Tecla Pressionada:".

17.5 Sugestões de uso

Exemplo retirado diretamente do trabalho apresentado via *YouTube*:

Pergunta: A partir dessa base, que tipo de outros projetos derivados daria para fazer e aplicar no *campus* Brusque?

Resposta: Bom, em minha opinião eu usaria ele para a segurança, por exemplo se tiver uma sala só para funcionários de alta patente, poderiam usar para colocar a senha para abrir a porta, evitando o acesso de gente de fora, alunos ou até invasores, porque hoje em dia, e fácil destrancar uma porta qualquer.

17.6 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 05/08/2021. A apresentação foi transmitida pelo canal do *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/HF8c3yjI0D0?t=5547⟩.

17.7 Conclusão

Aprendemos o funcionamento básico que o Teclado Matricial 4×4 com Arduino apresenta, além de ser um dispositivo de alta qualidade, ter um design diferenciado e muito fino, facilitando o uso em travas elétricas, controle de acesso, cofres, balanças eletrônicas, entre outros.

Relembrando: nós devemos sempre conectar as trilhas corretamente para evitar curtos ou erros na montagem.

18 SISTEMA DE PISCA-PISCA COM ARDUINO

Grupo 20:
Leandro Kohler Fagundes - ⟨cisplatina0@gmail.com⟩
Matheus Klann - ⟨matheusklann10@gmail.com⟩
Saimon Joás dos Santos Pollheim - ⟨blockstrikejogo021@gmail.com⟩

18.1 Este projeto: O que é

Ele é um Arduino que possui diversos esquemas juntos a ele, para que ele consiga, completar essa ação de fazer um *LED* piscar, onde tem a parte dos componentes onde o Arduino é ligado há uma fonte de energia e código, onde esses códigos mostram o que deveria acontecer e o que deve ligar, de acordo com a pessoa que programou. Logo após isso temos os esquemas de ligação que conectam o Arduino e os *LEDs*, em que são usados diversos cabos para essa ligação e após fazer isso de pôr, positivo com uma entrada negativa no *LED*, junto com o código, temos nosso grandioso trabalho que o Arduino realiza, como a ação que ele foi programada e construída, de piscar 9 *LEDs*.

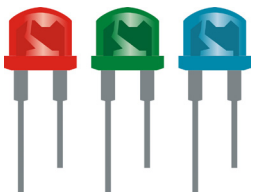
Este capítulo é baseado no trabalho de SHIGUE (2014).

18.2 Componentes usados

Para que esse sistema funcione será necessário vários componentes como por exemplo:

1. **Nove *LEDs*:** para conseguir cumprir o principal objetivo do sistema, será necessário 9 *LEDs*, os quais cumprem a sua função de desligar e ligar;

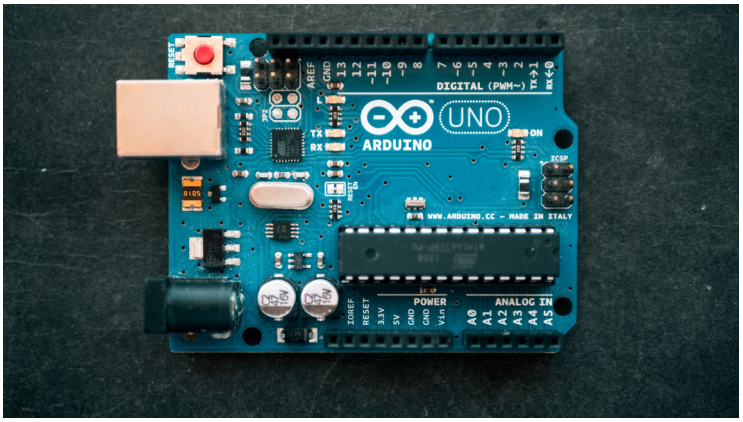
Figura 106 – *LED*



Fonte: ⟨https://pixabay.com/pt/vectors/rgb-conduziu-8mm-l%c3%a2mpadas-luz-2270087/⟩

2. **Um Arduino Uno R3:** Arduíno é como se fosse um micro-computador, ele é uma plataforma eletrônica de prototipagem, onde pode ter muitas utilidades na robótica e no trabalho *maker*, ela possui entrada e saída, uma linguagem de programação padrão e plataforma C/C++, que é capaz de receber informação através de outro dispositivo conectado a ele, como se fosse um painel de controle;

Figura 107 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

3. **Um Cabo USB:** para que o arduíno tenha uma fonte de energia e um código-fonte, basta apenas um cabo USB para poder fazer essa transmissão, de um notebook ou outro objeto, para o Arduino;

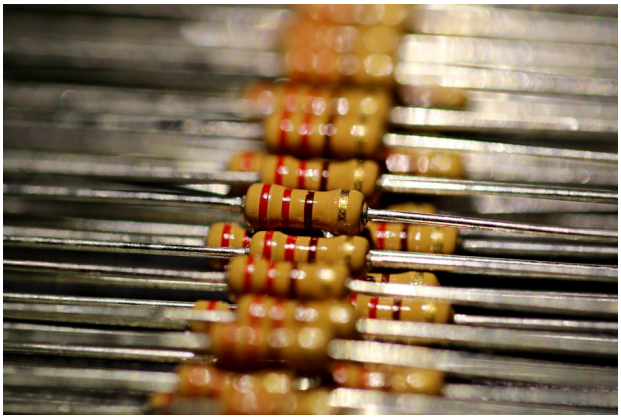
Figura 108 – Cabo USB tipo B

Fonte: ⟨https://unsplash.com/photos/oXOZAx_MJM4?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

4. **Um laptop ou notebook:** para o Arduino receber esses códigos e também essa fonte de energia, um laptop ou notebook é necessário, onde ele exerce esse papel, da criação do código-fonte e da principal fonte de energia;

5. **Nove Resistores de 150 Ohms:** apesar de ter uma fonte de energia, não foi especificado o quanto de energia será passado, o resistor serve para fazer esse limite de passagem de energia, onde ele transforma a energia, e passa somente o necessário para que não corra o risco de queimar ou danificar algo;

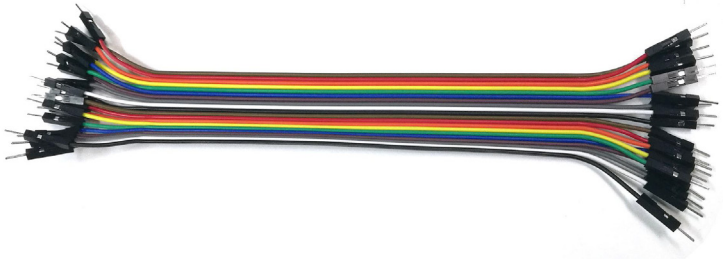
Figura 109 – Resistor



Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

6. **19 Cabos *Jumpers*:** tem como sua principal função na parte das ligações, onde ela irá distribuir a quantidade de energia dado a ele, para as demais partes que ele está conectado, fazendo assim a sua transmissão;

Figura 110 – Fios *jumper* macho-macho

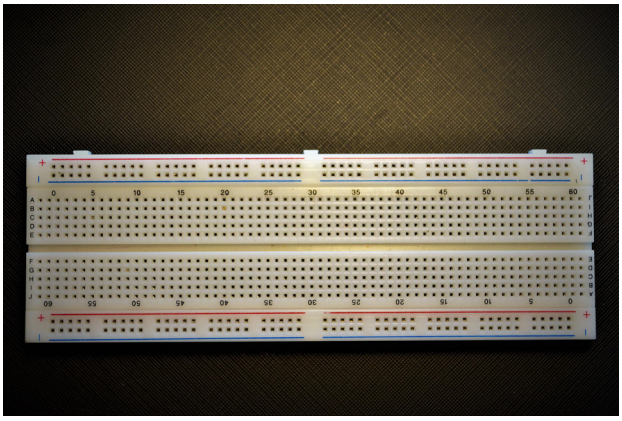


Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

7. **Uma *Protoboard*:** ela é responsável pelas ligações, onde ela será ligado a diversos jumpers e *LEDs*, para que assim tenha a transmissão e conexão das demais partes,

sendo que a *Protoboard* pertence também ao código-fonte, onde se diz quais partes devem estar em uso;

Figura 111 – *Protoboard*

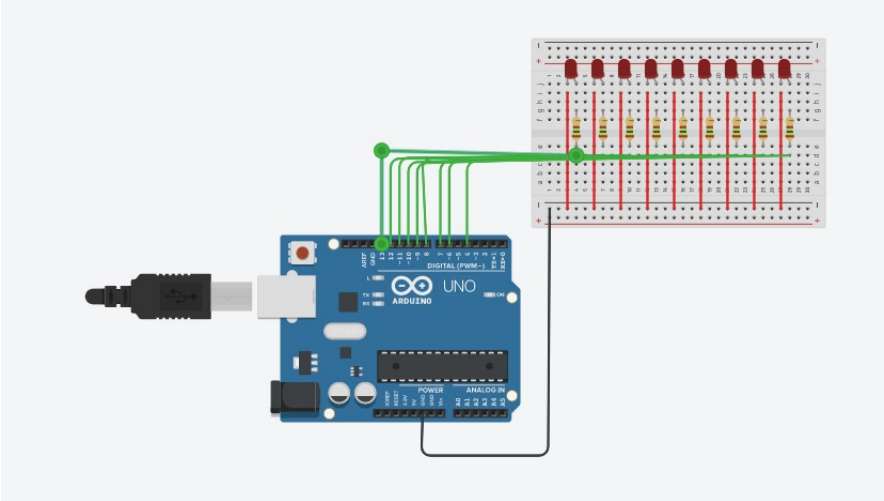


Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

18.3 Esquema de ligações

Como podemos observar na Figura 112, temos o notebook conectado ao um cabo USB no Arduino, que está fazendo a alimentação e passando o código-fonte; no Arduino podemos ver vários cabos conectados a eles e ali podemos ver as ligações.

Figura 112 – Esquema de ligações



Fonte: Tinkercad/reprodução/próprio autor

Abaixo, está um quadro de ligação dos pinos na *protoboard*:

Arduino	*Protoboard*	*LED*
D13	D4	i4
D12	D7	i7
D11	D10	i10
D10	D13	i13
D9	D16	16
D8	D19	i19
D7	D22	i22
D7	D25	i25
D4	D28	i28
Positivos		

Protoboard	*LED*
-	i
3	i3
6	i6
9	i9
12	i12
15	i15
18	i18
21	i21
24	i24
27	i27

Negativos (GND no Arduino para a entrada negativa da *protobard*)
Resistores acima da entrada positiva e *LEDs* 2 casas acima do resistor.

18.4 Código-fonte

Nas primeiras linhas do código-fonte abaixo foi definido quais pinos serão liberados para as ligações; na segunda linha em void setup definimos todos os pinos para que nem todos liguem de uma vez só; na terceira vemos *void loop* que é a repetição enquanto houver energia ou enquanto estiver conectado; em *void change* está sendo definido que os *LEDs* irão ligar 1 em 1 ou seja passando a imagem de movimento em que a conexão do *LED* 1 esse desliga e troca pro 2 e vai indo, e também está sendo definido que após chegar no último *LED* ele irá mudar de direção.

```
// Projeto Pisca-Pisca com Arduino
byte ledPin[] = {4, 6, 7, 8, 9, 10, 11, 12, 13}; //pinos do arduino positivas
int ledDelay(65); // Mudança do delay
int direction = 1;
int currentLED = 0;
unsigned long changeTime;

void setup() {
for (int x=0; x<10; x++) {
// definir todos os pinos para saída
pinMode(ledPin[x], OUTPUT); }
changeTime = millis();
```

```
}

void loop() {
if ((millis() - changeTime) > ledDelay) { // Repetia a ação
changeLED();
changeTime = millis();
}
}

void changeLED() {
for (int x=0; x<10; x++) {
// Desligar todos os leds
digitalWrite(ledPin[x], LOW);
}
digitalWrite(ledPin[currentLED], HIGH); // Ligar todos os leds
currentLED += direction;
// Trocando de leds 1 a 1
// Mudar de direção quando chegar no último led
if (currentLED == 9) {direction = -1;}
if (currentLED == 0) {direction = 1;}
}
```

18.5 Sugestões de uso

Ele pode até parecer algo muito simples, inútil entre outros diminutivos, porém ele pode ter um bom uso se for mais praticado como um começo para novos *makers*, para entender como essa lógica funciona, ele também pode ser acoplado em outros projetos para poder dar um destaque, como por exemplo num alarme, que ao ele ser acionado possa acender esse sistema de pisca-pisca, ou talvez para algo mais para decoração em objetos ou até mesmo em seu próprio quarto.

18.6 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 19/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/iqQwer30uSY?t=3330⟩.

19 VERIFICAÇÃO DE UMIDADE DO SOLO COM ARDUINO

Grupo 23:
Gabriel Lemos - ⟨gabriellemos1601@gmail.com.br⟩
Gustavo Iunceck - ⟨iunceckgustavo@gmail.com⟩

19.1 Este projeto: O que é

O objetivo do projeto é fazer com que cada luz acenda de acordo com o nível de umidade do solo. Como funciona:

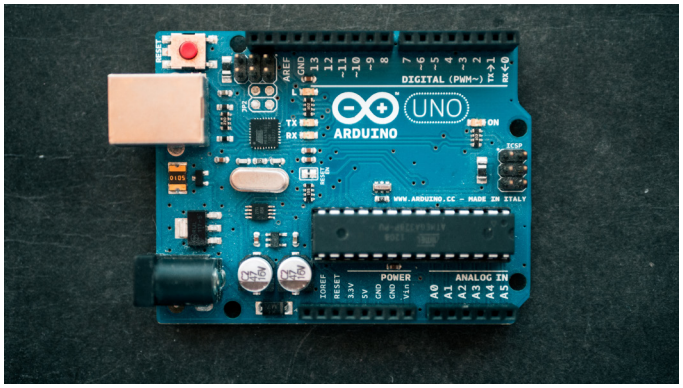
- Verde: Acende quando o solo está úmido;
- Amarela: Acende quando o solo tem uma umidade moderada;
- Vermelha: Acende quando o solo está seco.

Este capítulo é baseado no trabalho de THOMSEN (2016).

19.2 Componentes usados

1. **Arduino:** é uma placa composta por um microcontrolador, com entrada e saída, sendo usada em vários lugares;

Figura 113 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

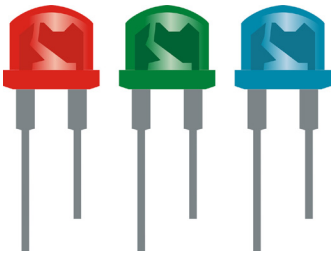
2. **Cabo USB:** transmite energia e o código-fonte para a placa;

Figura 114 – Cabo USB tipo B

Fonte: ⟨https://unsplash.com/photos/oXOZAx_MJM4?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

3. ***LEDs*:** servem para identificar a umidade do solo;

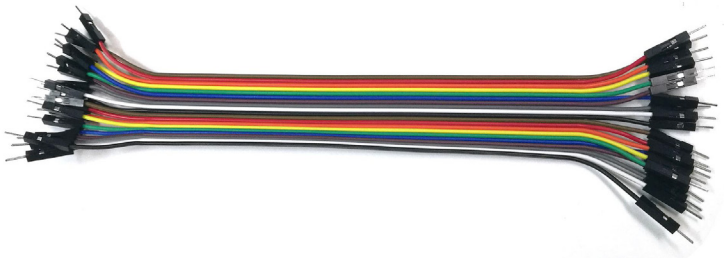
Figura 115 – *LED*



Fonte: ⟨https://pixabay.com/pt/vectors/rgb-conduziu-8mm-l%c3%a2mpadas-luz-2270087/⟩

4. **Cabos *jumper*:** fios que transmitem os dados;

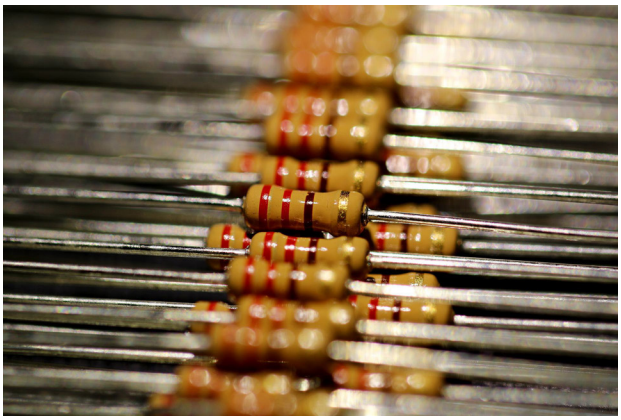
Figura 116 – Fios *jumper* macho-macho



Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

5. **Resistores 10 kΩ:** converte a energia elétrica em energia térmica;

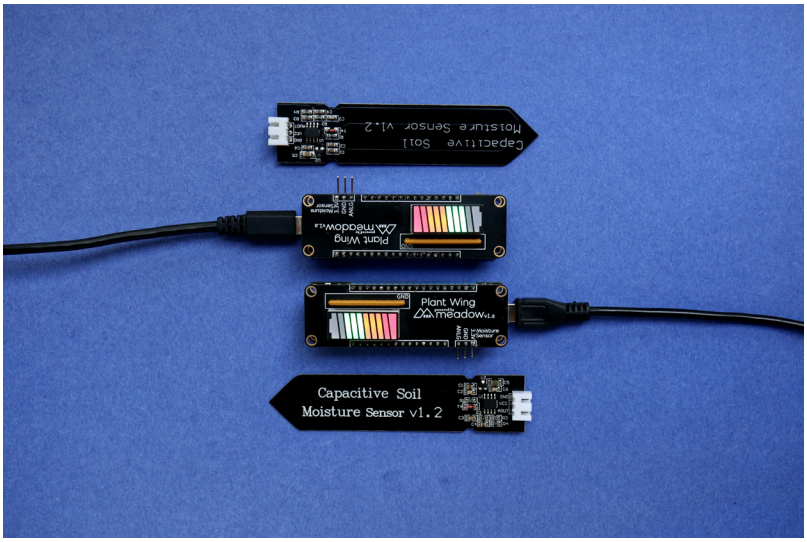
Figura 117 – Resistor



Fonte: ⟨https://pixabay.com/pt/photos/resist%c3%aancia-resistor-registro-5722984/⟩

6. **Sensor de umidade de solo:** possui dois eletrodos para conduzir corrente elétrica pelo solo;

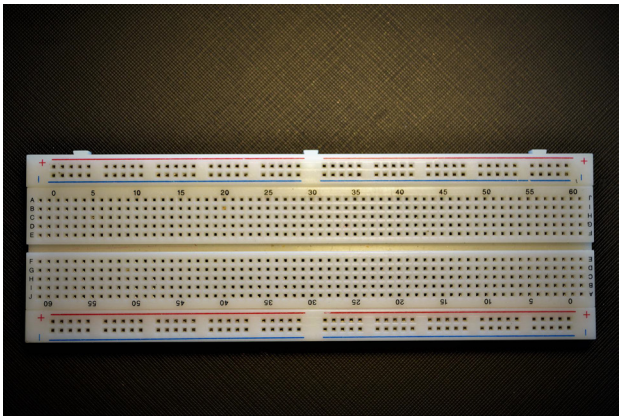
Figura 118 – Sensor de umidade do solo



Fonte: ⟨https://unsplash.com/photos/VNfiAyRoKv4?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

7. ***Protoboard*:** placa para montar os circuitos;

Figura 119 – *Protoboard*

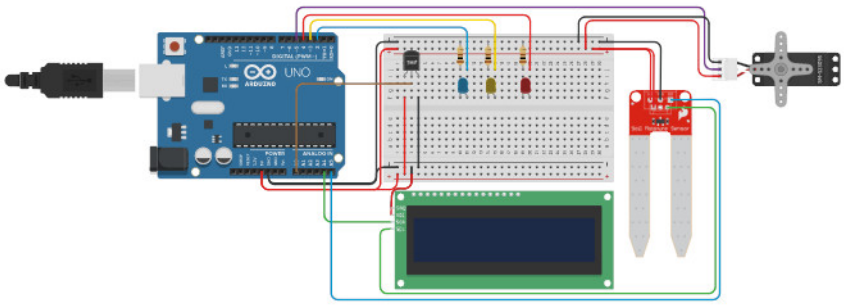


Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

19.3 Esquema de ligações

Os cabos pretos (GND) servem para controlar a tensão, o cabo vermelho conectado ao pino 5V serve para enviar energia ao sensor de umidade do solo, o cabo azul conectado ao pino A0 server para ler as informações do sensor de umidade do solo, e os cabos coloridos conectados aos pinos digitais 5, 6 e 7, serve para enviar a mensagem de que certo *LED* deve acender.

Figura 120 – Esquema de ligações



Fonte: ⟨https://www.tinkercad.com/things/78yBwd4Z3VQ-temperatura-e-umidade-do-solo⟩

19.4 Código-fonte

Trazer um exemplo de código-fonte.

```
//Programa: Monitoracao de planta usando Arduino
//Autor: FILIPEFLOP

#define pino_sinal_analogico A0
#define pino_led_vermelho 5
#define pino_led_amarelo 6
#define pino_led_verde 7

int valor_analogico;

void setup()
{
  Serial.begin(9600);
  pinMode(pino_sinal_analogico, INPUT);
  pinMode(pino_led_vermelho, OUTPUT);
  pinMode(pino_led_amarelo, OUTPUT);
  pinMode(pino_led_verde, OUTPUT);
}

void loop()
{
  //Le o valor do pino A0 do sensor
  valor_analogico = analogRead(pino_sinal_analogico);

  //Mostra o valor da porta analogica no serial monitor
  Serial.print("Porta analogica: ");
  Serial.print(valor_analogico);

  //Solo umido, acende o led verde
  if (valor_analogico > 0 && valor_analogico < 400)
  {
    Serial.println(" Status: Solo umido");
    apagaleds();
    digitalWrite(pino_led_verde, HIGH);
```

```
  }

  //Solo com umidade moderada, acende led amarelo
  if (valor_analogico > 400 && valor_analogico < 800)
  {
    Serial.println(" Status: Umidade moderada");
    apagaleds();
    digitalWrite(pino_led_amarelo, HIGH);
  }

  //Solo seco, acende led vermelho
  if (valor_analogico > 800 && valor_analogico < 1024)
  {
    Serial.println(" Status: Solo seco");
    apagaleds();
    digitalWrite(pino_led_vermelho, HIGH);
  }
  delay(100);
}

void apagaleds()
{
  digitalWrite(pino_led_vermelho, LOW);
  digitalWrite(pino_led_amarelo, LOW);
  digitalWrite(pino_led_verde, LOW);
}
```

19.5 Sugestões de uso

O projeto serve para ser usado em plantas ou plantações, para verificar se elas precisam de água ou se já tem o suficiente, assim ajudando o dono na hora de regar elas.

19.6 Apresentação no canal de *YouTube* do *campus*

Este projeto foi um trabalho avaliativo da disciplina de Hardware e Sistemas Operacionais do Professor Josiney, no dia 19 do 08 de 2021. Ocorreu uma apresentação que foi transmitida

pelo canal do *YouTube* do IFC *campus* Brusque e está gravada em: ⟨https://www.youtube.com/watch?v=iqQwer30uSY&t=4116s⟩.

20 SEMÁFORO PARA CARROS

Grupo 25:
Timóteo Vargas Follmann - ⟨timoteovargas151@gmail.com⟩
Jonatan Mendes - ⟨scoby.mendes@gmail.com⟩

20.1 Este projeto: O que é

Esse projeto simula um semáforo de carros e pedestre, a sequência inicia no *LED* verde para os carros passarem e vermelho para os pedestres. O semáforo depois de um tempo fica amarelo e vai indo para a cor vermelha, assim que estiver vermelho o semáforo dos pedestres fica verde e nessa sequência volta para o início e repete o mesma sequência.

Este capítulo é baseado no trabalho de SILVA (2020).

20.2 Componentes usados

1. **2x *LED* vermelho 5mm, 2x *LED* verde 5mm, 2x *LED* amarelo 5mm:** um componente que usa eletricidade para emitir luz de uma cor. É um componente eletrônico semicondutor, L.E.D = Light Emitter Diode, mesma tecnologia utilizada nos chips dos computadores, que tem a propriedade de transformar energia elétrica em luz. É também um componente do tipo bipolar, tem um terminal chamado ânodo e outro chamado cátodo, que permite ou não a passagem de corrente elétrica e, consequentemente, a geração ou não de luz. Fonte: ⟨https://totallight.com.br/led-o-que-e-e-como-funciona⟩;

2. **5x Resistor 220 ohm:** o resistor é um componente elétrico passivo que tem a função primária de limitar o fluxo da corrente elétrica em um circuito. Para facilitar o entendimento, observe o exemplo da água passando por um cano representado na imagem abaixo. Neste caso, o fluxo de água é uma analogia à corrente elétrica que flui em um circuito elétrico. Portanto, quando criamos uma resistência ao fluxo da água, a corrente irá se reduzir. Numa versão mais simplificada, um resistor é o responsável por diminuir a corrente elétrica, um exemplo é a lâmpada, ela impede a corrente elétrica para acender, pois ocorre um desvio que coleta a quantidade necessária e a outra parte segue a diante no circuito elétrico. Fonte: ⟨https://www.mundodaeletrica.com.br/o-que-e-um-resistor/⟩;

3. **1x *Protoboard*:** é uma placa usada para fazer conexões elétrica, geralmente utilizado no Arduino Uno. Também conhecida como matriz de contatos ou placa de prototipagem, a protoboard é uma placa que possui furos e conexões internas para montagem de circuitos, utilizada para testes com componentes eletrônicos. Sua maior vantagem de uso é que ele dispensa a necessidade de solda para conectar tais circuitos. Fonte: ⟨https://blog.multcomercial.com.br/saiba-o-que-e-protoboard-e-qual-sua-utilidade⟩;

4. **11x *Jumper* macho-macho:** são cabos que fazem as conexões na protoboard e o componente usado. Os cabos *Jumpers* macho-macho são utilizados na prototipagem de componentes eletrônicos. Cabos *Jumpers* macho-macho são peças indispensáveis na sua bancada de projetos. O uso destes cabos é ideal para efetuar as conexões entre componentes eletrônicos nos seus projetos pessoais ou projetos de TCC da faculdade ou curso técnico seja com Arduino, NodeMCU ESP8266 ou outros microcontroladores. São 40 cabos no total, e as duas extremidades dos cabos possuem conexão macho. Fonte: ⟨https://www.masterwalkershop.com.br/cabo-jumper-macho-macho-20cm-kit-com-40pcs⟩;

5. **1x Cabo USB:** um cabo usado para passar os dados do computador para o Arduino. O USB permite instalar periféricos ao no computador (sua principal utilidade) de maneira muito mais fácil do que antigamente, quando essa função era direcionada para pessoas experientes. Fonte: ⟨https://canaltech.com.br/hardware/o-que-e-usb-e-por-que-o-cabo-e-necessario/⟩;

6. **Placa Uno:** uma placa que executa códigos programados do computador. Em termos gerais, o Arduino é um placa composta por um microcontrolador Atmel de hardware livre. A placa conta com suporte de entrada/saída embutido (o que nos permite facilmente conectá-lo no computador). Já sua linguagem de programação padrão se baseia em C e C++. Ele pode ser usado de maneira independente para controlar diversos equipamentos ou até mesmo criar outros. Fonte: ⟨https://empeltecjr.com/automacao-controle-e-arduino/⟩;

20.3 Esquema de ligações

Primeiro passo: colocar os *LEDs* na *breadboard*, assim como as respetivas resistências (todas de 330 Ω). Assegurem-se que ligam os *LEDs* na posição correta, e fazem a ligação da perna do negativo à resistência, e das resistências ao negativo/ground. Nas pernas positivas do *LED* iremos ligar os fios às portas digitais no Arduino (mais tarde).

Segundo passo: se tiverem um *buzzer*, liguem-no após o semáforo dos carros. Liguem a resistência de 300 Ω na perna negativa do *buzzer*, e liguem essa resistência ao negativo, tal como nos *LEDs*.

Terceiro passo: coloquem agora os semáforos dos peões (vermelho e verde), tal como foi feito para o semáforo dos carros (colocação dos *LEDs* e resistências).

Quarto passo: temos agora, da esquerda para a direita: 3 *LEDs*, vermelho, amarelo e verde para os carros, um *buzzer* (opcional) e 2 *LEDs* vermelho e verde, para os peões. Depois do semáforo dos peões coloquem o botão e, numa das pernas, liguem a resistência de 1 kΩ, que por sua vez liga ao negativo. Liguem a outra perna diagonalmente oposta ao positivo (5V) da *breadboard*.

Vamos agora proceder à ligação dos fios que controlam as luzes dos semáforos às portas digitais do Arduino. Vamos fazer a ligação da esquerda para a direita.

1. No positivo do *LED* vermelho do semáforo dos peões, liguem um fio à porta digital 10.
2. No positivo do *LED* amarelo, liguem um fio à porta digital 9.
3. No positivo do *LED* verde, liguem um fio à porta digital 8.
4. Se usarem o *buzzer*, liguem um fio ao positivo do buzzer à porta digital 6.
5. Liguem um fio do numa das pernas do botão (entre a resistência), e liguem a outra extremidade à porta digital 5.
6. Agora, para o *LED* vermelho dos peões, no positivo, liguem um fio à porta digital 3.
7. Por fim, para o *LED* verde, liguem um fio da perna do lado positivo à porta digital 2.

Para saber mais acesse: ⟨https://andresilva.me/archive/notes/arduino/semaforos⟩.

20.4 Código-fonte

```
// Projeto 4 - Semáforo

int pedVerde = 9; // Define os pinos que serao utilizados
int pedVermelho = 8;
int carroVerde = 12;
int carroAmarelo = 11;
int carroVermelho = 10;
```

```
void setup() // Define os pinos como saidas
{
pinMode(pedVerde, OUTPUT);
pinMode(pedVermelho, OUTPUT);
pinMode(carroVerde, OUTPUT);
pinMode(carroAmarelo, OUTPUT);
pinMode(carroVermelho, OUTPUT);
digitalWrite(carroVerde, HIGH); // Coloca na posição inicial. Somente o
verde dos carros e o vermelho dos pedestres acesos
digitalWrite(carroAmarelo, LOW);
digitalWrite(carroVermelho, LOW);
digitalWrite(pedVerde, LOW);
digitalWrite(pedVermelho, HIGH);
}

void loop()
{
digitalWrite(carroVerde, HIGH); // Acende o verde dos carros e o vermelho
dos pedestres
digitalWrite(pedVermelho, HIGH);
delay(5000); // Aguarda 5 segundos
digitalWrite(carroVerde, LOW);
digitalWrite(carroAmarelo, HIGH); // apaga o verde dos carros e acende o
amarelo
delay(3000); // aguarda mais 3 segundos
digitalWrite(carroAmarelo, LOW); // apaga o amarelo dos carros e acende o
vermelho
digitalWrite(carroVermelho, HIGH);
digitalWrite(pedVermelho, LOW); // apaga o vermelho dos pedestres e acende
o verde
digitalWrite(pedVerde, HIGH);
delay(5000); // aguarda mais 5 segundos
digitalWrite(pedVerde, LOW);
for(int x = 0; x<5; x++) // Pisca o vermelho dos pedestres
{
digitalWrite(pedVermelho, HIGH);
delay(250);
```

```
digitalWrite(pedVermelho, LOW);
delay(250);
}
digitalWrite(carroVermelho, LOW);
}
```

20.5 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 12/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/DzjiyOzZOTs?t=1741⟩.

21 SENSOR DE UMIDADE E TEMPERATURA COM *SHIELD* ETHERNET

Grupo 99:
Josiney de Souza - ⟨josiney.souza@ifc.edu.br⟩

21.1 Este projeto: O que é

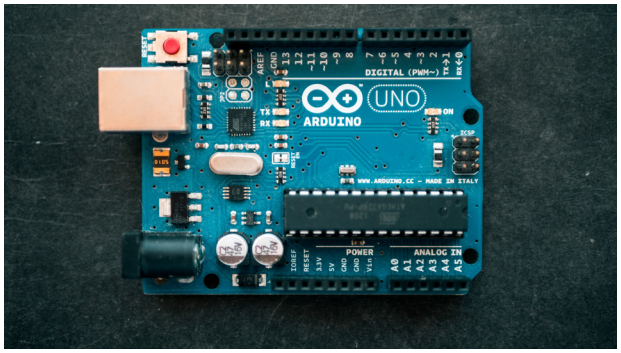
Este projeto visa coletar a umidade e a temperatura de um sensor DHT-11 e montar uma página web para exibir os dados coletados usando um *shield* Ethernet e a plataforma Arduino. A coleta de informações ocorre a cada 5 minutos e é armazenada em uma matriz para depois ser exibida em uma tabela.

Este capítulo é baseado no trabalho de THOMSEN (2014).

21.2 Componentes usados

1. **Arduino e cabo de alimentação/dados:** placa microcontroladora usada para se trabalhar com protótipos. O cabo USB é usado para transmitir o código-fonte para a placa e também funciona como fonte de alimentação. Esta última pode ser substituída por uma fonte chaveada 9V e 1A.

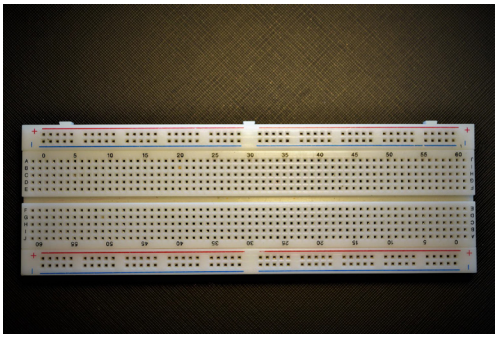
Figura 121 – Placa Arduino Uno



Fonte: ⟨https://unsplash.com/photos/fZB51omnY_Y?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink⟩

2. ***Protoboard* 170 pontos:** placa extensora usada para prototipação;

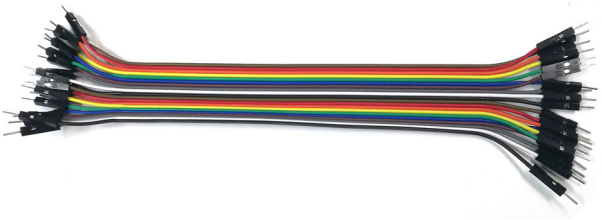
Figura 122 – *Protoboard*



Fonte: ⟨https://pixabay.com/pt/photos/protoboard-eletr%c3%b4nico-circuito-5210635/⟩

3. **Fios/cabos *jumper*:** cabos usados para transmitir dados de e para os sensores e atuadores;

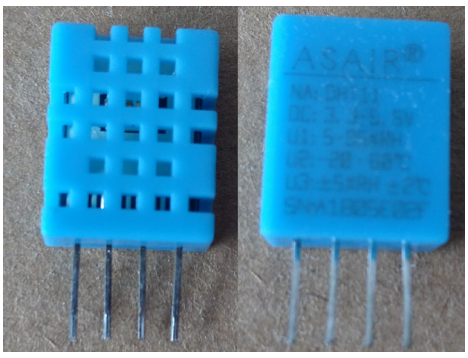
Figura 123 – Fios *jumper* macho-macho



Fonte: ⟨https://pixabay.com/pt/photos/cabo-idc-cabo-idc-fio-el%c3%a9trico-1991608/⟩

4. **Sensor DHT-11:** sensor que mede a umidade relativa do ar e a temperatura de um ambiente. Funciona com valores de 20% a 90% para a umidade relativa do ar e com valores de 0ºC a 50ºC para a temperatura;

Figura 124 – Sensor DHT-11



Fonte: próprio autor/Arquivo pessoal

5. ***Shield* Ethernet:** placa que permite criar um servidor de páginas web.

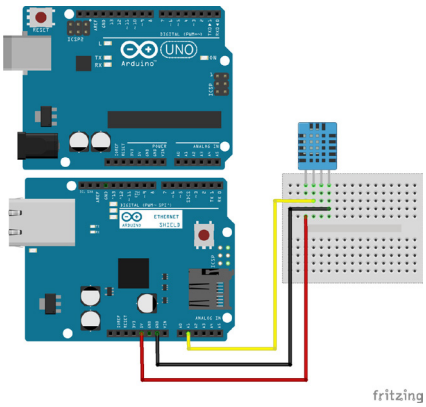
Figura 125 – *Shield* Ethernet

Fonte: ⟨https://www.flickr.com/photos/snazzyguy/3632890354/in/photolist-6x2vJA-cgfBtJ-bycTsQ-bkQyfm-bkQy7E-d1fuHq-bkQycu-jtqunm-dTm6R2-9Nc7na-6cTGyb-4SzUMx-76mQqx-kyBt3X-9C1UvU-9iLoh9-76qJv7-68BP3J-bBhZEd-e4dQq6-CDkVNZ-juq1uU-82FRu5-2jaRjBT-akeCuj-8mWMKv-8tZGw1-9nRpNC-9sR3Ni-6XW48D-4RREUw-6cTCGm-6cPG4D-aGdjiH-8n4wao-bBhZR5-c6VGXA-adCgAv-8swku9-7wSkVD-5NGorH-bmAZRx-6Yk8sK-8vEY5L-245Bzbj-7usJxn-bhWSmX-bQcDrH-9Z1myD-Kbtuat⟩

21.3 Esquema de ligações

A Figura 126 mostra o esquema de ligações para o projeto usando a ferramenta Fritzing. Dos 4 pinos do sensor DHT-11, apenas 3 são utilizados. Ao primeiro pino, está ligado a alimentação do sensor, ou seja, 5V (fio vermelho). Ao segundo pino, está a ligação para obtenção de dados do sensor atrelada à porta A1 do Arduino (fio amarelo). O terceiro pino não é usado. Ao quarto pino, está associado o terra para fechar o circuito (fio preto). Nessa figura, o *shield* Ethernet está abaixo da placa Arduino, devido às ligações, mas ele deve ser montado sobre o Arduino, assim como todos os *shields*.

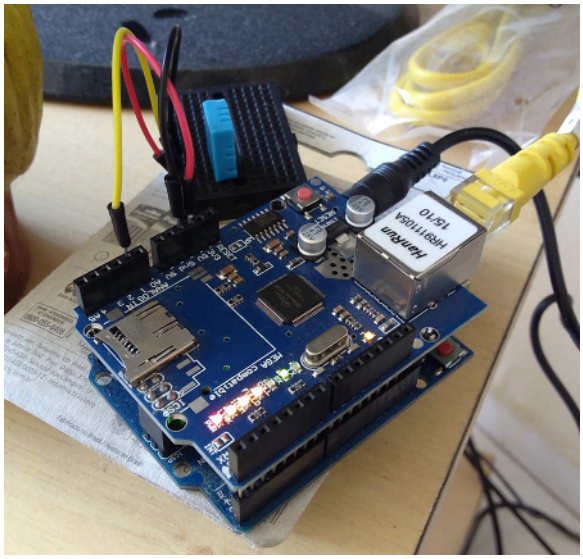
Figura 126 – Esquema de ligação feito com a ferramenta Fritzing



Fonte: próprio autor/Fritzing

A Figura 127 demonstra o esquema de ligações feito em ambiente real. Nela, é possível ver as ligações realizadas no protótipo e que o *shield* Ethernet se liga à placa Arduino ficando no topo do mesmo.

Figura 127 – Foto do esquema de ligações realizada pelo autor



Fonte: próprio autor/Acervo pessoal

21.4 Código-fonte

Abaixo está o código-fonte criado para a IDE do Arduino que faz a coleta e exibição de dados. O código foi baseado no exemplo de uso do *shield* Ethernet disponibilizado junto à IDE e modificado para atender ao objetivo deste projeto.

```
/*
  Web Server - Modificado

 A simple web server that shows the value of the analog input pins.
 using an Arduino Wiznet Ethernet shield.

 Circuit:
 * Ethernet shield attached to pins 10, 11, 12, 13
 * Analog inputs attached to pins A0 through A5 (optional)

 created 18 Dec 2009
 by David A. Mellis
```

```
 modified 9 Apr 2012
 by Tom Igoe

 */

#include <SPI.h>
#include <Ethernet.h>
// Incorporado por Josiney
#include <dht.h>

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0x00, 0x1F, 0xC0, 0x13, 0x00, 0x01 };
// 1FC1FC - Comecar com 1F da erro ao iniciar o servidor
//IPAddress ip(10,5,10,251);
IPAddress ip(192,168,1,250);

/* ============================
======== Minhas adicoes ========
============================ */
// Tempo entre as coletas, em minutos
#define PERIODICIDADE 5
// Tempo de espera antes da proxima coleta, em milissegundos
#define TEMPO_ESPERA (unsigned long) PERIODICIDADE*60*1000
// Quantidade de colunas da tabela
#define TOTAL_COLUNAS 12
// Quantidade de linhas da tabela
#define HORAS_MAX 24
// Numero total de amostras_no_dia antes de reutilizar a tabela
#define AMOSTRAS_MAX HORAS_MAX*TOTAL_COLUNAS

// Amostras coletadas no dia, para controle da tabela
unsigned int amostras_no_dia = 0;
// Controle de dia que o programa estah executando
unsigned int dia = 0;
// Tabela de dados de temperatura coletadas
byte dados_temp[HORAS_MAX][TOTAL_COLUNAS];
// Tabela de dados de umidade coletadas
```

```
//byte dados_umid[HORAS_MAX][TOTAL_COLUNAS];
// Inicializacao de indice de linha da matriz de dados
byte linha = 0;
byte linha_destaque;
// Inicializacao de indice de linha da matriz de dados
byte coluna = 0;
byte coluna_destaque;
// Objeto de dados para leitura dos valores do sensor
dht DHT;

byte i, j;
byte minima = 255;
byte maxima = 0;

// Inicializacao de variavel que faz com que o contador de tempo
// decorrido nao trave quando der overflow, continuando a executar
// o codigo enviado ao microcontrolador. Mais detalhes em:
// https://www.baldengineer.com/arduino-how-do-you-reset-millis.html
unsigned long contador_anterior = 0;
unsigned long contador_atual;

// Caso seja necessario fazer mais configuracoes sobre a rede na
// qual o shield Ethernet esta executando, pode-se usar as linhas
// abaixo (comentadas por padrao). Mais informacoes em:
// http://blogmasterwalkershop.com.br/arduino/arduino-utilizando-o-ethernet-shield
    -w5100-via-web-server/
//byte gateway[] = {10.5.10.1};
//byte subnet[] = {255, 255, 254, 0};

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

void setup() {
 // Open serial communications and wait for port to open:
  /*Serial.begin(9600);
   while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
```

```
  }*/

  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  //Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();
  //Serial.print("server is at ");
  //Serial.println(Ethernet.localIP());

  realiza_coleta_dados();
}


void loop() {
  contador_atual = millis();

  if ((unsigned long) contador_atual - contador_anterior >= (unsigned long)
      TEMPO_ESPERA)
  {
    realiza_coleta_dados();
    contador_anterior = contador_atual;
  }

  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    //Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        //Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          //client.println("HTTP/1.1 200 OK");
```

```
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close"); // the connection will be closed
              after completion of the response
                //client.println("Refresh: 10"); // refresh the page automatically
                    every 5 sec
          client.println();
          mostra_cabecalho_if(client);
          // TODO: Mostrar a tabela de umidade tambem
          cria_cabecalho_tabela_dados(client);

          for (i=0; i < HORAS_MAX; i++)
          {
            client.println("<tr>");
            client.print("<td>");
            client.print(i+1);
            client.println("</td>");
            for (j=0; j < TOTAL_COLUNAS; j++)
            {
              if (i == linha_destaque && j == coluna_destaque)
                client.print("<td style=color:#FFF;background-color:#080;>");
              else
                client.print("<td>");
              client.print(dados_temp[i][j]);
              client.println("</td>");
            }
            client.println("</tr>");
          }
          client.println("</table>");
          client.println("</body>");
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          // you've gotten a character on the current line
```

```
        currentLineIsBlank = false;
      }
     }
   }
   // give the web browser time to receive the data
   delay(1);
   // close the connection:
   client.stop();
   //Serial.println("client disonnected");
  }
}

void mostra_cabecalho_if (EthernetClient client)
{
  client.println("<!DOCTYPE html>");
  client.println("<html lang=pt-BR>");
  client.println("<head>");
  //client.println("<title>IFC Brusque - Lab Info 1</title>");
  client.println("<title>Sala de TV</title>");
  client.println("<meta charset=utf-8>");
  client.println("<style>");
  client.println("p,table,th,td{border:1px solid;border-color:red green green red;
      text-align:center}");
  client.println("</style>");
  client.println("</head>");
  client.println("<body>");
  //client.println("<p>Instituto Federal Catarinense - <i>campus</i> Brusque <br>
      Temperatura do Laboratorio de Informatica 1:</p>");
  client.println("<p>Temperatura da Sala de TV");
  DHT.read11(A1);
  client.print("<br>Temp. atual: ");
  client.print(DHT.temperature);
  client.print("<br>Umid. atual: ");
  client.print(DHT.humidity);
  client.print("<br>Minima: ");
  client.print(minima);
  client.print("<br>Maxima: ");
  client.print(maxima);
  client.println("</p>");
```

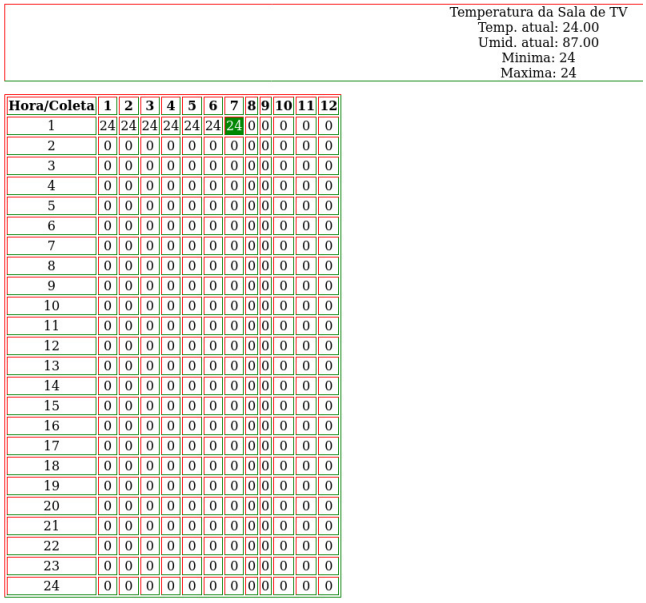
```
}

void cria_cabecalho_tabela_dados (EthernetClient client)
{
  // TODO: AUTOMATIZAR PARA QUANDO TIVER OUTRA PERIODICIDADE //
  client.println("<table>");
  client.println("<tr>");
  client.println("<th>Hora/Coleta</th>");
  client.println("<th>1</th>");
  client.println("<th>2</th>");
  client.println("<th>3</th>");
  client.println("<th>4</th>");
  client.println("<th>5</th>");
  client.println("<th>6</th>");
  client.println("<th>7</th>");
  client.println("<th>8</th>");
  client.println("<th>9</th>");
  client.println("<th>10</th>");
  client.println("<th>11</th>");
  client.println("<th>12</th>");
  client.println("</tr>");
}

void realiza_coleta_dados ()
{
  amostras_no_dia++;

  DHT.read11(A1);
  dados_temp[linha][coluna] = DHT.temperature;
  if (DHT.temperature < minima)
    minima = DHT.temperature;
  if (DHT.temperature > maxima)
    maxima = DHT.temperature;
  //dados_umid[linha][coluna] = DHT.humidity;

  linha_destaque = linha;
  coluna_destaque = coluna;

  coluna++;
```

```
  if (coluna >= TOTAL_COLUNAS)
  {
    coluna = 0;
    linha++;
    if (linha >= HORAS_MAX)
    {
      linha = 0;
      amostras_no_dia = 0;
      dia++;
    }
  }
}
```

21.5 Exemplo

A Figura 128 mostra um exemplo do resultado final visto em navegador web após a execução do programa por 35 minutos. Nota-se que a célula com a leitura mais atual tem destaque frente às demais e que, se uma leitura ainda não foi realizada, a célula é preenchida com valor 0.

Figura 128 – Exemplo de visualização do resultado final do programa em navegador web



Temperatura da Sala de TV
Temp. atual: 24.00
Umid. atual: 87.00
Minima: 24
Maxima: 24

Hora/Coleta	1	2	3	4	5	6	7	8	9	10	11	12
1	24	24	24	24	24	24	24	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0

Fonte: próprio autor/Reprodução

Caso se tenham passado mais de 24 horas, os dados mais antigos são descartados para que

novos dados de coleta possam aparecer na tabela. Além disso, como o momento da coleta pode ser diferente do momento da visualização no navegador web, então mostra-se a leitura atual de temperatura e de umidade. Também é mostrada a maior e a menor temperatura registrada desde que o Arduino foi ligado.

21.6 Sugestões de uso

Este trabalho por ser usado tanto para se monitorar quanto também para se exibir valores de temperatura e de umidade relativa do ar em qualquer ambiente fechado. Seu uso pode ser parte de um projeto de monitoramento de umidade e de temperatura (como uma estação meteorológica).

21.7 Apresentação no canal de *YouTube* do *campus*

Este projeto foi apresentado nas atividades avaliativas da disciplina Hardware e Sistemas Operacionais do professor Josiney em 19/08/2021. A apresentação foi transmitida pelo canal de *YouTube* do IFC *campus* Brusque e está gravada no seguinte endereço: ⟨https://youtu.be/iqQwer30uSY?t=5296⟩.

Referências

CASTRO, G. de; CASSIOLI, M. **Usando o Sensor de Som**. 2020. Robocore. On-line. Disponível em: ⟨https://www.robocore.net/tutoriais/usando-sensor-de-som⟩. Acesso em 26 de jan. de 2022.

FILIPEFLOP. **Projeto 10 - Sensor de luz ambiente - FilipeFlop**. 2022. FilipeFlop. On-line. Disponível em: ⟨https://www.filipeflop.com/universidade/kit-maker-arduino/projeto-10-sensor-de-luz-ambiente/⟩. Acesso em 26 de jan. de 2022.

FILIPEFLOP. **Projeto 4 – Semáforo**. 2022. FilipeFlop. On-line. Disponível em: ⟨https://www.filipeflop.com/universidade/kit-maker-arduino/projeto-4-semaforo/⟩. Acesso em 26 de jan. de 2022.

FROMAGET, P. **15 Best Operating Systems for Raspberry Pi (with pictures)**. 2022. Raspberry Tips. Disponível em: ⟨https://raspberrytips.com/best-os-for-raspberry-pi/⟩. Acesso em 22 de dez. de 2021.

KENSHIMA, G. **Makey Makey – Tornando o mundo maker mais divertido**. 2016. Embarcados. On-line. Disponível em: ⟨https://www.embarcados.com.br/makey-makey/⟩. Acesso em 26 de jan. de 2022.

MURTA, J. G. A. **Guia completo do Display LCD - Arduino**. 2018. Eletrogate. On-line. Disponível em: ⟨https://blog.eletrogate.com/guia-completo-do-display-lcd-arduino/⟩. Acesso em 26 de jan. de 2022.

MURTA, J. G. A. **Guia completo do Controle Remoto IR + Receptor IR para Arduino**. 2021. Eletrogate. On-line. Disponível em: ⟨https://blog.eletrogate.com/guia-completo-do-controle-remoto-ir-receptor-ir-para-arduino/⟩. Acesso em 26 de jan. de 2022.

NATIVA, A. **Tutorial: como medir o pH da água e do solo com Arduino?** 2018. Acqua Nativa. On-line. Disponível em: ⟨https://www.acquanativa.com.br/aplicacoes/kit-sensor-ph-com-arduino-5-passos.html⟩. Acesso em 26 de jan. de 2022.

PORTA, L. D. **Teclado Matricial 4×4 com Arduino**. 2016. Matheus Gebert Straub em UsinaInfo. On-line. Disponível em: ⟨https://www.usinainfo.com.br/blog/teclado-matricial-4x4-com-arduino/⟩. Acesso em 26 de jan. de 2022.

RASPBIAN. **RaspbianInstaller - Raspbian**. 2012. Raspbian. On-line. Disponível em: ⟨https://www.raspbian.org/RaspbianInstaller⟩. Acesso em 26 de jan. de 2022.

REIS, F. dos. **Como usar um Sensor de Movimento PIR com Arduino**. 2018. Bóson Treinamentos em Ciência e Tecnologia. On-line. Disponível em: ⟨http://www.bosontreinamentos.com.br/eletronica/arduino/como-usar-um-sensor-de-movimento-pir-com-arduino/⟩. Acesso em 26 de jan. de 2022.

SHIGUE, C. Y. **TUTORIAL: Experimentos e conceitos de programação computacional usando o microcontrolador ARDUINO**. 2014. UNIVERSIDADE

DE SÃO PAULO - ESCOLA DE ENGENHARIA DE LORENA - Ambiente eDisciplinas. On-line. Disponível em: ⟨https://edisciplinas.usp.br/mod/resource/view.php?id=1539153&usg=AOvVaw3blo6iqU2fu1JhaN6S-otd⟩. Acesso em 26 de jan. de 2022.

SILVA, A. **Simulação de semáforos para tráfego e peões**. 2020. Wiki do André. On-line. Disponível em: ⟨https://andresilva.me/archive/notes/arduino/semaforos⟩. Acesso em 26 de jan. de 2022.

SOUZA, J. de. **[IFC Brusque] Dimmer**. 2021. Tinkercad. On-line. Disponível em: ⟨https://www.tinkercad.com/things/68Coy0hxK0S⟩. Acesso em 26 de jan. de 2022.

STRAUB, M. G. **Sensor de Nível de Água Arduino para Automação com Relé e Display**. 2016. UsinaInfo. On-line. Disponível em: ⟨https://www.usinainfo.com.br/blog/sensor-de-nivel-de-agua-arduino-para-automacao-com-rele-e-display/⟩. Acesso em 26 de jan. de 2022.

TERRA, P. R. R. **Piano com Makey Makey e Scratch**. 2019. Makerzine. On-line. Disponível em: ⟨https://www.makerzine.com.br/educacao/piano-com-makey-makey-e-scratch/⟩. Acesso em 26 de jan. de 2022.

THOMSEN, A. **Controlando um Motor de Passo 5v com Arduino**. 2013. FilipeFlop. On-line. Disponível em: ⟨https://www.filipeflop.com/blog/controlando-um-motor-de-passo-5v-com-arduino/⟩. Acesso em 26 de jan. de 2022.

THOMSEN, A. **Como comunicar com o Arduino Ethernet Shield W5100**. 2014. FilipeFlop. On-line. Disponível em: ⟨https://www.filipeflop.com/blog/tutorial-ethernet-shield-w5100/⟩. Acesso em 26 de jan. de 2022.

THOMSEN, A. **Tutorial Módulo Bluetooth com Arduino**. 2015. FilipeFlop. On-line. Disponível em: ⟨https://www.filipeflop.com/blog/tutorial-modulo-bluetooth-com-arduino/⟩. Acesso em 26 de jan. de 2022.

THOMSEN, A. **Monitore sua planta usando Arduino**. 2016. FilipeFlop. On-line. Disponível em: ⟨https://www.filipeflop.com/blog/monitore-sua-planta-usando-arduino/⟩. Acesso em 26 de jan. de 2022.

Lista de ilustrações

Os Organizadores

Bernardo Victoria Escobar Silva

Aluno egresso do IFC campus Brusque do curso Técnico em Informática Integrado ao Ensino Médio. Foi voluntário no projeto IFmaker IFC campus Brusque que originou este livro e bolsista do projeto Conexões da Física com a Cultura Maker derivado do primeiro projeto. Atuou como estagiário de TI no IFC campus Brusque.

Josiney de Souza

Professor EBTT do IFC campus Brusque da área de Informática que trabalha com hardware, redes e sistemas operacionais. Já foi coordenador de TI, do curso Técnico em Informática e do CST em Redes de Computadores no campus. Trabalhou com projetos de monitoria, de pesquisa, de extensão e integrados no campus. Foi o coordenador do projeto IFmaker IFC campus Brusque que originou este livro. Possui Bacharelado em Ciência da Computação e Mestrado em Informática ambos pela UFPR. Já foi professor PRONATEC pelo IFPR, professor substituto pela UFPR e bolsista do C3SL na UFPR.

Leonardo Felipe de Avila Calbusch

Professor EBTT do IFC campus Brusque da área de Informática que trabalha com programação e desenvolvimento de sistemas. Já foi coordenador curso Técnico em Informática no campus. Trabalhou com projetos de monitoria e de pesquisa no campus. Foi colaborador no projeto IFmaker IFC campus Brusque que originou este livro. Possui graduação em Sistemas de Informação pela UNIPLAC e Mestrado em Computação pela UNIVALI.

Otavio Henrique da Silva

Aluno do IFC campus Brusque do curso Técnico em Informática Integrado ao Ensino Médio. Foi bolsista do projeto IFmaker IFC campus Brusque que originou este livro. Atuou em apresentações do projeto na comunidade de inovação catarinense. De Poços de

Caldas-MG é uma pessoa muito fascinada com a tecnologia e como ela pode ser usada para automatizar as coisas no seu dia a dia, também curte animes e mangás, gosta também de computadores e videogames sendo os principais gêneros de corrida, metroidvania e jogos em primeira pessoa. É uma pessoa que gosta de ficar mais na sua mas, quando está com os seus amigos, fica mais solto e interage mais. Curte bastante escutar música; escuta a maioria dos gêneros musicais, curte também programação e web design. Também adora viajar. É uma pessoa com grandes sonhos e com vontade de trabalhar em alguma área que mexa com tecnologia, com vontade de morar fora do Brasil, de viajar e conhecer alguns países e de aprender algumas línguas novas e culturas novas.

Tiago Rafael de Almeida Alves

Professor EBTT do IFC campus Brusque da área de Física. Trabalhou com projetos de monitoria, de pesquisa e de extensão no campus. Foi colaborador no projeto IFmaker IFC campus Brusque que originou este livro e coordenador do projeto Conexões da Física com a Cultura Maker derivado do primeiro projeto. Possui Licenciatura e Bacharelado em Física pela FURG e Mestrado em Ensino
de Física pela UFSC.

Este livro retrata um trabalho integrado entre ensino, pesquisa e extensão que o Instituto Federal Catarinense *campus* Brusque desenvolveu em 2021 dentro da disciplina Hardware e Sistemas Operacionais do Técnico em Informática Integrado ao Ensino Médio. A partir do projeto de pesquisa de mesmo nome do e-book, foi possível chegar nesse resultado mesmo em meio à pandemia global de COVID-19, bem como envolver diversos servidores do *campus* e participar ativamente da comunidade de inovação catarinense ao entregar apresentações de temáticas relevantes e deixar um legado para uso dos diferentes laboratórios *maker* dos Institutos Federais pelo Brasil.

editora IFC

Associação Brasileira
das Editoras Universitárias