

TRABALHO PRÁTICO – LAB. ELETRÔNICA DIGITAL

Grupo: Alex Felicio Costa, Rafael Costa Bitencourt.

Data: 22 / 06 / 2023

OBJETIVOS

Estudar o funcionamento do Microcontrolador Arduino através de sua aplicação prática na construção de um Controle Remoto Infravermelho.

METODOLOGIA DE DESENVOLVIMENTO

O projeto foi desenvolvido em 3 principais etapas:

- 1) Pesquisa;
- 2) Programação e Simulação;
- 3) Montagem e Correções.

1. Pesquisa:

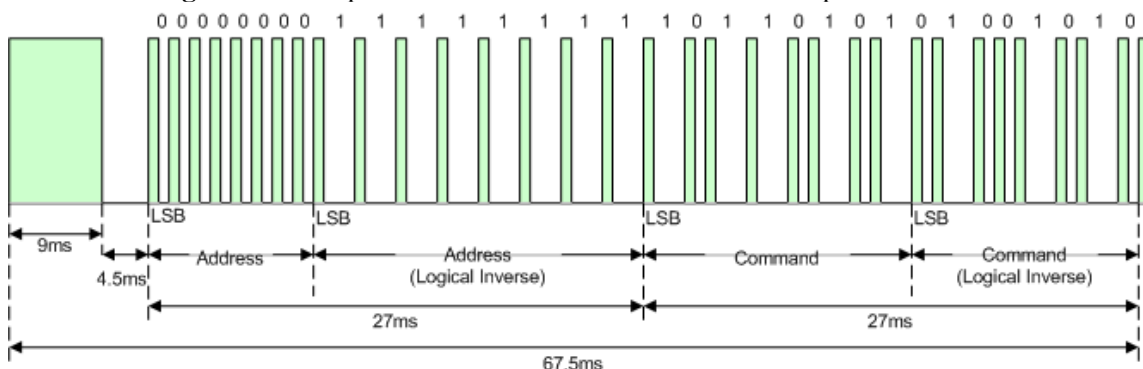
A comunicação infravermelho presente em controles remotos utiliza a radiação eletromagnética na faixa do infravermelho para emitir ou receber sinais, o infravermelho é conveniente porque pode ser emitido por um simples diodo emissor de luz (“Light emitting diode” - LED) e porque o ser humano não é capaz de enxergá-lo.

Como qualquer sistema de comunicação digital é necessário seguir um protocolo, no caso de controles remotos infravermelhos o protocolo mais comum é o NEC. O protocolo NEC é um sistema de comunicação serial com frequência de aproximadamente 38 kHz.

Ele consiste na transmissão de 4 bytes de dados (32 bits), como ilustrado na figura 1. Para isso, primeiramente é emitido um pulso de 9ms seguido de um espaço “vazio” de 4.5ms e então os 4 bytes: O endereço (para o dispositivo receptor), o endereço invertido, o comando (instrução a ser executada) e o comando invertido.

O nível lógico alto (1) é determinado por um pulso de 562.5µs seguido por um espaço “vazio” de 1.6875ms e o nível lógico baixo (0) por um pulso de 562.5µs seguido por um espaço “vazio” de 562.5µs.

Figura 1: Exemplo do formato de transmissão usando o protocolo NEC.



Fonte: <https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol>, 2023

Visto isso, para implementar a comunicação com protocolo NEC ao Arduino, é necessário traduzir o padrão de transmissão para a linguagem de programação (no caso C++), processo já feito pela comunidade. Portanto utilizou-se a biblioteca [Arduino IRremote](#), capaz converter o sinal de entrada digital do Arduino em um número seguindo o protocolo NEC (além de outros protocolos menos utilizados) e também converter um número em sinal para transmissão e emití-lo em uma saída digital. Nesse caso, o número em questão representa a sequência binária de 32 bits utilizada para transmitir o sinal.

A estrutura para utilização da biblioteca consiste em:

```
IrReceiver.decodedIRData.decodedRawData
```

Variável que retorna o valor numérico de 32 bits do sinal “Bruto” (contém todos 4 bytes) recebido

```
IrSender.sendNECRaw(VALOR, NUMERO_DE_VEZES);
```

Função que transmite um valor de 32 bits X vezes para a saída digital especificada.

```
if (IrReceiver.decode()){CODIGO}
```

Checa se algum sinal foi recebido.

```
IrReceiver.resume();
```

Habilita a variável “IrReceiver.decodedIRData.decodedRawData” a receber um novo valor.

Também foi necessário implementar no código um meio de salvar valores na memória EEPROM (“Electrically Erasable Programmable Read Only Memory” - Memória Somente de Leitura Programável Apagável Eletricamente). Para tal utilizou-se da biblioteca [EEPROM](#) e duas funções baseadas no artigo [Arduino Store int into EEPROM](#), já que a leitura e armazenamento na memória do arduino não é trivial, visto que cada espaço de memória armazena um byte, sendo necessário “quebrar” o valor a ser armazenado (que tem 4 bytes) em 4 partes para armazená-lo e depois “juntar” as partes para lê-lo:

```
writeLongIntoEEPROM(ENDERECO, VALOR);
```

“Escreve” valor no endereço de memória especificado (o valor ocupa o endereço especificado e os próximos 3, já que ocupa 4 bytes). Importante saber que o Arduino Uno utilizado tem 1024 bytes de memória disponíveis para uso (além do espaço dedicado ao código).

```
readLongFromEEPROM(int address)
```

Variável que retorna o valor que foi armazenado no endereço de memória especificado (primeiro endereço de memória utilizado).

2. Programação e Simulação

Após o estudo e compreensão dos conceitos e recursos necessários, partiu-se para elaboração do código, utilizando a linguagem C++, a qual o Arduino é compatível.

A parte lógica do código consiste em um conjunto de instruções de IF e ELSE IF, estruturas condicionais que servem para checar se alguma condição é verdadeira ou não e assim executar um comando:

```
if( digitalRead(botao_gravar) == 1 ){CODIGO}
```

Verifica se o pino digital 'gravar' está no estado lógico alto e se sim, executa o código.

No caso quando o botão responsável pelo modo de gravação for acionado, se algum botão de função for acionado ao mesmo tempo, o sinal recebido será gravado na memória associada à aquele botão:

```
if( digitalRead(botao_gravar) == 1 ){  
    if( digitalRead(botao1) == 1 && IrReceiver.decode() ){  
        writeLongIntoEEPROM(0, IrReceiver.decodedIRData.decodedRawData);  
        IrReceiver.resume();  
    }  
}
```

Esse conjunto de condições verifica primeiro se o botão de gravação está ativado e se sim, verifica se o botão 1 está ativado e (simbologia &&) se o receptor está recebendo algum sinal, em caso positivo, grava o valor lido pelo receptor na memória 0 (e nas próximas 3) e habilita a recepção de um novo sinal.

Para emissão do sinal em infravermelho é necessário verificar se o botão de gravação está desativado e se sim verificar se algum botão de função foi acionado, emitindo o valor na memória associada à esse botão:

```
if( digitalRead(botao_gravar) == 1 ){  
    if( digitalRead(botao1) == 1 && IrReceiver.decode() ){  
        writeLongIntoEEPROM(0, IrReceiver.decodedIRData.decodedRawData);  
        IrReceiver.resume();  
    }  
}  
else{  
    if (digitalRead(botao1) == 1) {  
        IrSender.sendNECRaw(readLongFromEEPROM(0), 1);  
    }  
}
```

As condições acima representam a seguinte lógica: Se o botão de gravação estiver ativo, execute o código para gravação (já explicado anteriormente), senão, verifica se o botão 1 está ativado, se estiver ativo, emita uma vez o sinal utilizando o valor gravado na memória 0.

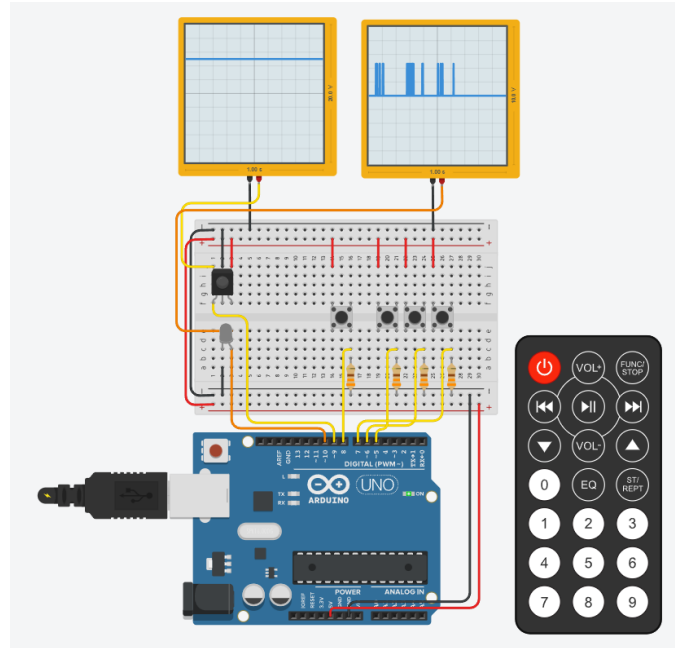
Em seguida aplicou-se a lógica para implementar mais botões, para isso basta utilizar ‘else if’ após o condição do primeiro botão, assim, caso o primeiro botão não esteja ativo, o programa verifica se o segundo está. É possível utilizar somente estruturas de ‘if’, porém, assim seria possível acionar mais de um botão ao mesmo tempo, causando erros:

```
if( digitalRead(botao_gravar) == 1 ){
    if( digitalRead(botao1) == 1 && IrReceiver.decode() ){
        writeLongIntoEEPROM(0, IrReceiver.decodedIRData.decodedRawData);
        IrReceiver.resume();
    }
    else if( digitalRead(botao2) == 1 && IrReceiver.decode() {
        writeLongIntoEEPROM(3, IrReceiver.decodedIRData.decodedRawData);
        IrReceiver.resume(); }
else{
    if (digitalRead(botao1) == 1) {
        IrSender.sendNECRaw(readLongFromEEPROM(0), 1);
    }
    else if (digitalRead(botao2) == 1) {
        IrSender.sendNECRaw(readLongFromEEPROM(3), 1);
    }
}
```

Nessa estrutura de condições foi adicionado um segundo botão, que está associado ao endereço de memória 4 (já que o primeiro valor ocupa os espaços de memória de 0 à 3)

Com código parcialmente escrito, utilizou-se a plataforma [Tinkercad](#) para simulação do projeto (figura 2), principalmente para verificar seu funcionamento e encontrar ‘bugs’. Porém, não é possível testar sua aplicação, já que o Tinkercad não oferece dispositivos com receptores infravermelho.

Figura 2: Circuito para testes montado no Tinkercad.



Fonte: Arquivo próprio, 2023

3. Montagem e Correções

Com o código completo e teoricamente funcional, se deu início à montagem seguindo o esquema elétrico simulado. Por se tratar de um projeto com fim educacional, o circuito foi montado utilizando componentes THT (“Through Hole Technology”, Tecnologia de pinos inseridos em furos) e protoboard, geralmente usada para prototipagem.

Após a realização de testes com aparelhos com comunicação via infravermelho, foi possível reconhecer alguns erros, sendo eles:

- **Gravação de sinais nulos:** Eventualmente, ao acionar o modo de gravação, o receptor retornava um sinal com valor zero, então adicionou-se condição:

```
if (IrReceiver.decodedIRData.decodedRawData != 0)
```

Verifica se ‘IrReceiver.decodedIRData.decodedRawData’ é diferente de 0.

- **Velocidade de emissão dos sinais:** O programa emite os sinais mais rápido do que a maioria dos aparelhos consegue receber, assim, se pressionado constantemente, o botão só emite o sinal uma vez. Para resolver adicionou-se um atraso de 100 ms entre a emissão de um sinal e outro:

```
delay(100);
```

Espera 100 ms.

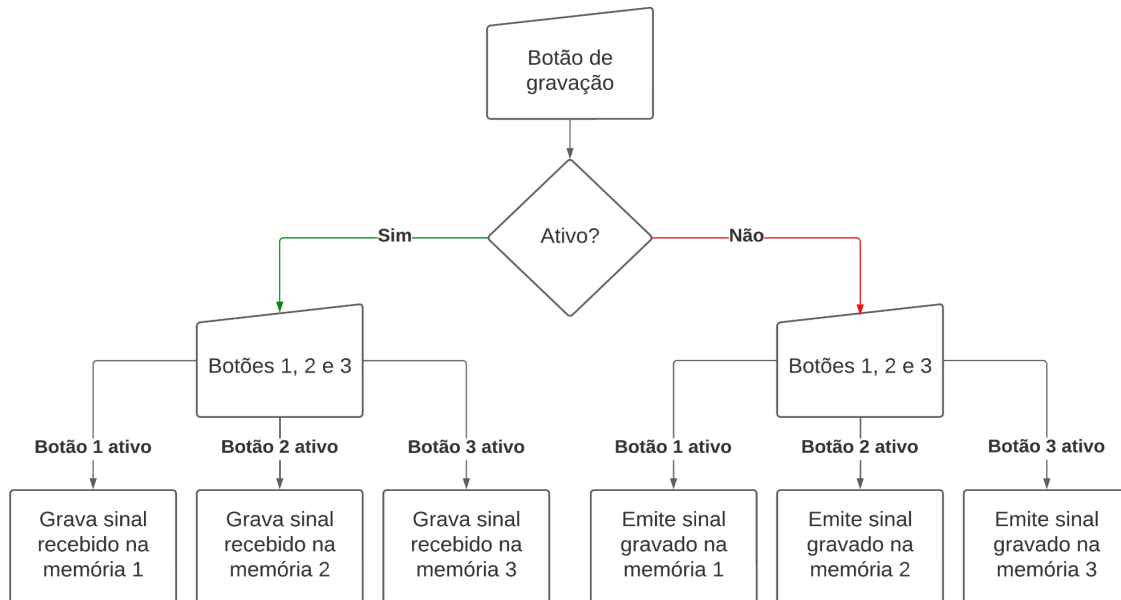
Estrutura lógica do código finalizada após correções:

```
if (digitalRead(botao_gravar) == 1) {
    if (IrReceiver.decode() && digitalRead(botao1) == 1) {
        if (IrReceiver.decodedIRData.decodedRawData != 0)
            writeLongIntoEEPROM(0, IrReceiver.decodedIRData.decodedRawData);
        IrReceiver.resume();
    }
    else if (IrReceiver.decode() && digitalRead(botao2) == 1) {
        if (IrReceiver.decodedIRData.decodedRawData != 0)
            writeLongIntoEEPROM(4, IrReceiver.decodedIRData.decodedRawData);
        IrReceiver.resume();
    }
    else if (IrReceiver.decode() && digitalRead(botao3) == 1) {
        if (IrReceiver.decodedIRData.decodedRawData != 0)
            writeLongIntoEEPROM(8, IrReceiver.decodedIRData.decodedRawData);
        IrReceiver.resume();
    }
}
else {
    if (digitalRead(botao1) == 1) {
        IrSender.sendNECRaw(readLongFromEEPROM(0), 1);
        delay(100);
    }
    else if (digitalRead(botao2) == 1) {
        IrSender.sendNECRaw(readLongFromEEPROM(4), 1);
        delay(100);
    }
    else if (digitalRead(botao3) == 1) {
        IrSender.sendNECRaw(readLongFromEEPROM(8), 1);
        delay(100);
    }
}
```

DESCRITIVO TÉCNICO DE FUNCIONAMENTO

O circuito desempenha a função de um Controle Remoto Infravermelho Programável, seguindo a lógica da figura 3:

Figura 3: Fluxograma do funcionamento lógico do circuito.

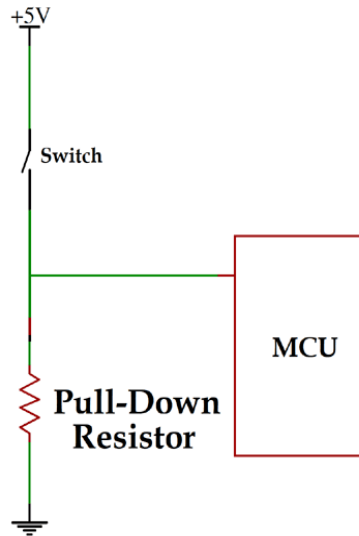


Fonte: Arquivo próprio, 2023

Desse modo, o sistema tem 5 entradas (Receptor infravermelho, botão do modo gravação, botão 1, botão 2, botão 3 e botão 4) e 1 saída (LED infravermelho). Todas as entradas e saídas operam com sinal digital com estado lógico alto em $\approx 5\text{v}$.

A interação com o circuito é realizada a partir de botões que utilizam resistores de Pull-Down para comunicação com o Microcontrolador, assim, quando desativados os botões conectam as entradas digitais ao “ground” (nível lógico baixo) e quando ativos ao Vcc (nível lógico alto), como ilustrado na figura 4:

Figura 4: Exemplo de diagrama elétrico para resistor de Pull-Down.





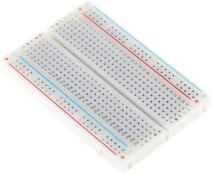



Fonte: Arquivo próprio, 2023

A recepção dos comandos é feita utilizando o receptor infravermelho, que já tem internamente um transdutor e conversor analógico digital, transformando então a luz infravermelha captada em sinal digital. Os dados fornecidos pelo receptor digital (comandos de um controle remoto) são interpretados pelo microcontrolador utilizando o protocolo NEC e armazenados na memória.

A comunicação entre o circuito e aparelhos controlados por infravermelho é feita através do LED infravermelho, que é ligado e desligado de maneira a transmitir a sequência binária que corresponde ao código armazenado na memória segundo o protocolo NEC. O LED infravermelho foi conectado diretamente ao “ground” e à saída digital do Arduino, isso porque a corrente máxima das saídas digitais é 40mA, não necessitando um resistor em série com o LED para limitar a corrente.

COMPONENTES

Tabela 1: Lista de componentes usados na montagem do projeto.

Componente	Quantidade
	Resistor 4
	Botão tátil 4
	Protoboard 400 pontos 1
	Arduino Uno 1
	Receptor Universal Infravermelho 1
	LED Infravermelho 1

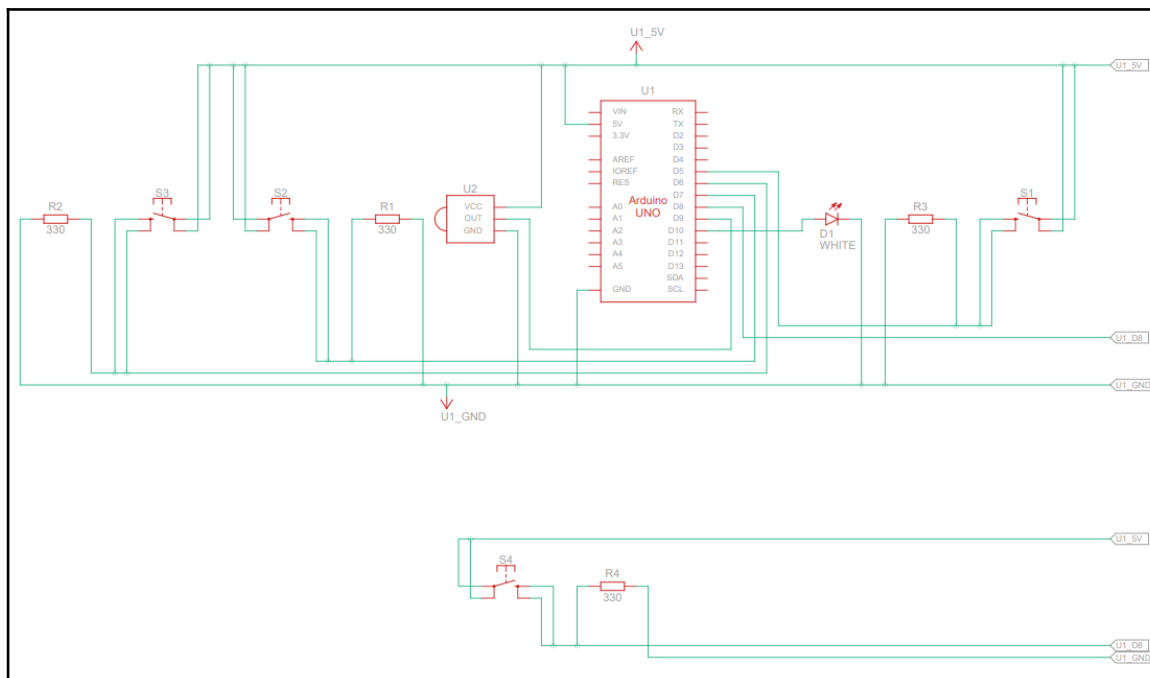
Fonte: Elaborado pelos autores, 2023.

Características relevantes para o projeto:

- Resistor:
 - Resistência: 330Ω ;
 - Potência: $1/4\text{ W}$;
 - 5% de tolerância.
- Botão tátil:
 - 4 pinos, 2 interligados;
- Protoboard 400 pontos:
 - Dimensões: $8.3 \times 5.5 \times 1.0\text{ cm}$;
 - 10 colunas;
 - 30 linhas interligadas;
 - 4 colunas para alimentação.
- Arduino Uno:
 - Baseado no microchip ATmega328P;
 - Tensão de operação: $6\text{-}20\text{V DC}$;
 - 14 entradas/saídas digitais;
 - Saída de alimentação 5v DC .
- Receptor Universal Infravermelho:
 - Tensão de operação: $2.7\text{-}5.5\text{ DC}$;
 - Frequência de operação: 38kHz ;
 - Tensão em nível lógico baixo: 0.4V ;
 - Tensão em nível lógico baixo: até 4.5V .
- LED infravermelho:
 - Tensão de operação $1.2\text{-}1.4\text{V DC}$

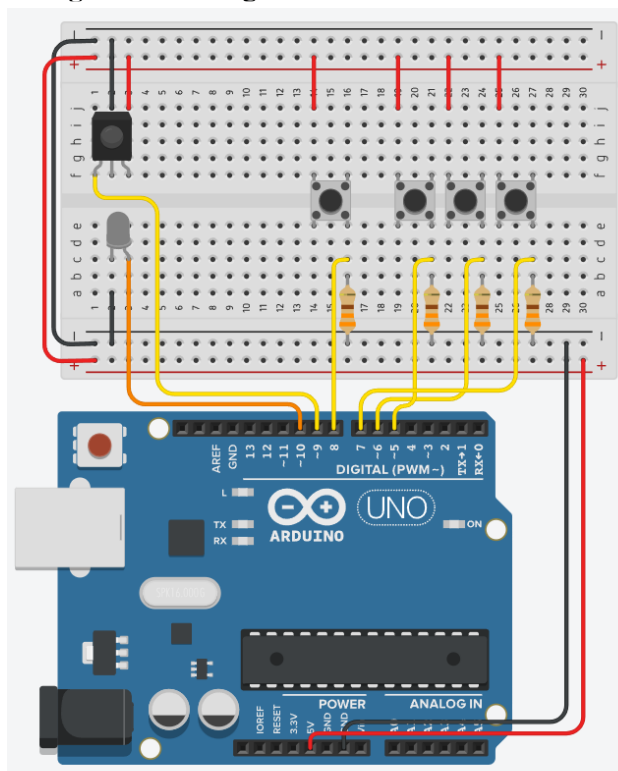
CIRCUITOS

Figura 5: Diagrama elétrico do circuito.



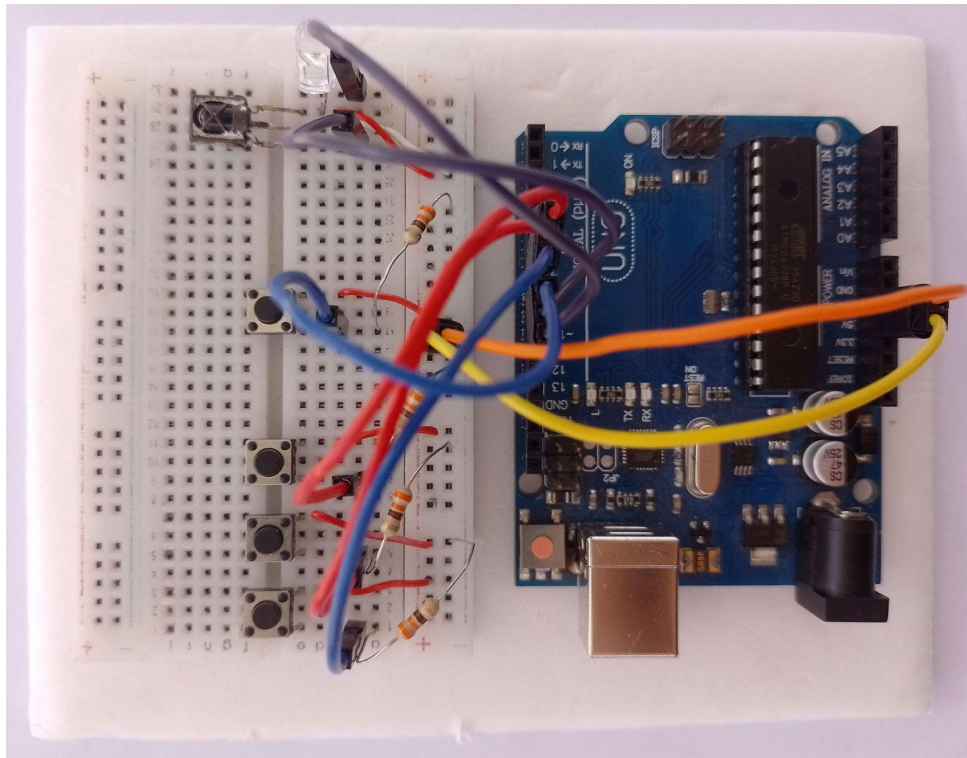
Fonte: Arquivo próprio, 2023

Figura 6: Montagem do circuito no Tinkercad.



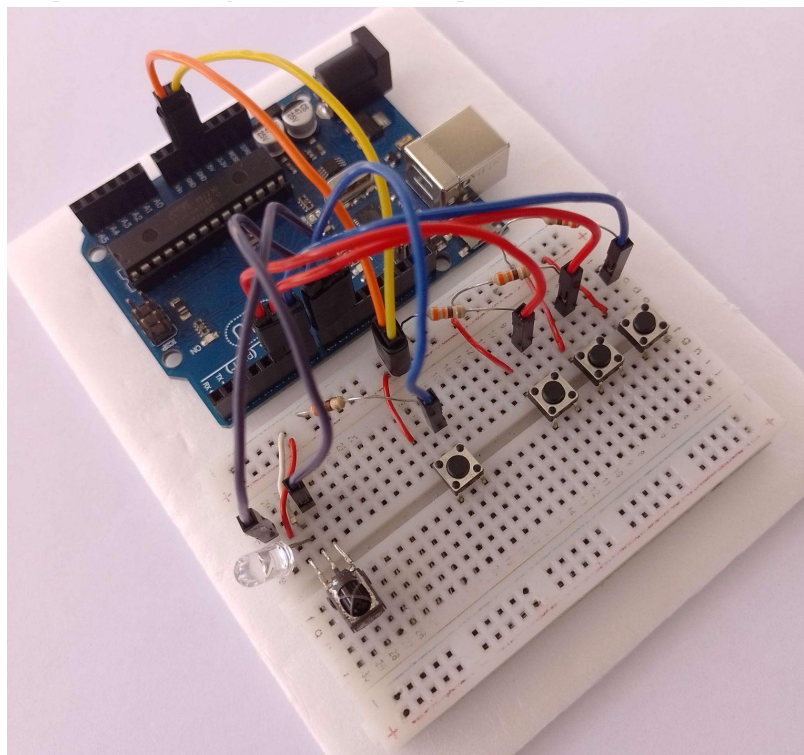
Fonte: Arquivo próprio, 2023

Figura 7: Montagem do circuito em protoboard, visão superior.



Fonte: Arquivo próprio, 2023

Figura 8: Montagem do circuito em protoboard, visão isométrica.



Fonte: Arquivo próprio, 2023

RESULTADOS

Para verificar seu funcionamento, realizou-se testes com diversos aparelhos, averiguando se os comandos eram salvos e emitidos corretamente, além de testar sua capacidade de armazenar os comandos mesmo sem alimentação.

Após testes constatou-se que o protótipo funciona corretamente e consegue se comunicar com a maioria dos aparelhos de controle remoto. Utilizando-o é possível salvar 3 funções do controle remoto original de algum aparelho. Para demonstração durante a aula, salvou-se as funções Ligar/Desligar, Volume + e Volume - de um Projetor EPSON presente no laboratório.

CONCLUSÕES

Com o intuito de compreender o funcionamento do microcontrolador Arduino através de sua aplicação em um Controle Remoto Infravermelho Programável, realizou-se a pesquisa para a implementação de um protocolo de comunicação via infravermelho, o protocolo NEC, além da utilização da memória EEPROM do Arduino para armazenar comandos.

Assim, utilizando o Tinkercad para simulação e experimentação, desenvolveu-se o software necessário para a operação do Controle Remoto. Após correções, partiu-se para a montagem do circuito em protoboard.

Com o circuito montado e operante, realizou-se testes com diversos aparelhos para provar seu funcionamento, testes estes bem sucedidos, comprovando seu desempenho.

Portanto, conclui-se a grande utilidade e efetividade do Arduino como plataforma de prototipagem, ensino e até mesmo com potencial para implementação comercial, devido sua acessibilidade, grande acervo online e comunidade ativa.

Por fim o código completo e arquivos do projeto foram disponibilizados na plataforma [GitHub](https://github.com/RafaelCostaBiten/Lab.Eletronica.Digital.Projeto.Final): <<https://github.com/RafaelCostaBiten/Lab.Eletronica.Digital.Projeto.Final>>

REFERÊNCIAS BIBLIOGRÁFICAS

1. GitHub - Arduino-IRremote/Arduino-IRremote: Infrared remote library for Arduino: send and receive infrared signals with multiple protocols. Disponível em: <<https://github.com/Arduino-IRremote/Arduino-IRremote>>.
2. NEC Infrared Transmission Protocol | Online Documentation for Altium Products. Disponível em: <<https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol>>.
3. ED. Arduino Store int into EEPROM. Disponível em: <<https://roboticsbackend.com/arduino-store-int-into-eprom/>>.
4. EEPROM. Disponível em: <<https://pt.wikipedia.org/wiki/EEPROM>>.