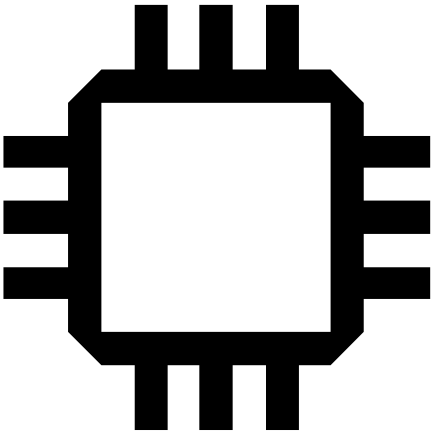
Microcontroladores:

Introdução para autodidatas



por Jhonata Flôres Sande

Microcontroladores: Introdução para autodidatas

Trabalho de Extenção

Jhonata Flores Sande
Tutor Marco C.

Faculdade Estácio
Programação de Microcontroladores

Vitória da Conquista — BA, 2024

Conteúdo

Introdução

O avanço dos computadores, desde as primeiras máquinas de cálculos mecânicas até os atuais microcontroladores sofisticados, revolucionou nossa forma de interagir com o mundo e de entender a própria informação. Em poucas décadas, a computação se tornou parte essencial do cotidiano, movendo desde dispositivos domésticos até complexos sistemas de automação industrial. Esta evolução impressionante, que começou com pioneiros como Charles Babbage e Alan Turing, agora nos permite moldar a energia e a informação com precisão em uma infinidade de aplicações.

O campo dos microcontroladores, que este livro aborda aborda profundidade, uma das mais fascinantes de comostáticas com exemplos a se manifesta em uso de diário. Os microcontroladores não são 'pequenos', computadores; eles são vitais peças em sistemas embarcados, capazes de transformar elétricos em sistemas definidas em ações, que se dão se os sistemas dos variados mais. Como bem observação Harold Abelson e Gerald Jay Sussman emEstrutura e Interpretação de Programas de Computador (SICP), "a ciência da computação não é tanto a ciência do que os computadores fazem, mas daquilo que podemos instruí-los a fazer". Esta perspectiva fundamenta o entendimento dos microcontroladores, que executam comandos complexos e precisos a partir de abstrações bem planejadas.

A jornada deste livro começa com uma exploração da história e das bases dos microcontroladores, avançando para tópicos como estruturas internas, arquiteturas e linguagens de programação específicas. Em cada capítulo, abordamos as tecnologias que permitiram a miniaturização dos sistemas e a combinação de eficiência energética e potência de processamento, como a arquitetura Von Neumann e a Harvard, além dos modelos RISC e CISC.

Este livro pretende ser uma ponte entre teoria e prática, oferecendo tanto fundamentos teóricos quanto guias práticos, como o desenvolvimento de projetos no Arduino. Aqui, você não só compreenderá o que faz um microcontrolador funcionar, mas também como utilizá-lo para realizar suas próprias criações. A proposta é revelar os segredos da programação de baixo nível e apresentar a integração entre software e hardware como uma verdadeira arte, desmistificando o processo de transformar impulsos elétricos em soluções funcionais.

Seja você um iniciante ou um entusiasta da área, este livro oferece um mergulho completo nas possibilidades dos microcontroladores. Assim como a energia é transformada em informação e, posteriormente, em ações, espero que as ideias aqui apresentadas inspirem você a inovar e a contribuir com novos desenvolvimentos na era digital.

Da energia à programação

Os computadores, como os conhecemos hoje, surgiram para atender às necessidades de matemáticos e cientistas de lidar com cálculos complexos, resolver equações elaboradas e processar grandes volumes de dados de maneira eficiente. No início, as ferramentas de cálculo eram limitadas, baseadas em ábacos ou calculadoras mecânicas, que, embora fossem inovações significativas, ainda exigiam esforço manual e tempo considerável. A ideia de uma máquina automática para realizar cálculos complexos começou a ganhar forma no século XIX, com figuras notáveis como Charles Babbage, conhecido como o "pai do computador". Ele projetou a Máquina Analítica, uma máquina que, embora nunca finalizada, possuía muitos dos conceitos que formariam a base dos computadores modernos, como memória, unidade de processamento e controle sequencial.

O desenvolvimento dos computadores também foi impulsionado por momentos de extrema necessidade histórica, como a Segunda Guerra Mundial, onde houve uma demanda urgente por máquinas que pudessem decifrar códigos inimigos e realizar cálculos balísticos. Combinando avanços em eletrônica e teoria da computação, cientistas e engenheiros criaram as primeiras máquinas de computação modernas, usando componentes como válvulas e relés. Cada avanço exigia colaboração interdisciplinar e contou com a contribuição de

várias figuras visionárias, como Alan Turing, que estabeleceu fundamentos teóricos sobre máquinas de computação, e John von Neumann, – grave bem esse nome – que propôs a arquitetura que até hoje é usada na maioria dos computadores.

Assim, ao longo de décadas, conceitos de lógica, matemática e engenharia foram integrados em uma única estrutura que, com o tempo e o aperfeiçoamento das técnicas, resultou no que chamamos de computadores.

Evolução das estruturas de computadores

Os primeiros computadores eram dispositivos mecânicos, onde cada cálculo dependia de operações físicas, como a rotação de engrenagens. Essas máquinas foram substituídas por computadores eletrônicos, que utilizavam cartões perfurados para representar dados. A entrada e a saída de dados eram feitas fisicamente, com cartões que eram "lidos" pela máquina para realizar operações. Os cartões perfurados foram, então, sucedidos por sistemas de válvulas e relés, dispositivos que permitiam o fluxo de corrente elétrica e podiam representar valores binários (0 e 1), o início do que mais tarde se tornaria a linguagem dos computadores.

O próximo avanço veio com a criação dos transistores, que substituíram as válvulas e permitiram uma miniaturização significativa dos circuitos. Com os transistores, surgiram os circuitos integrados, que consistem em milhares de transistores em um único chip. Esses circuitos integrados marcaram uma

nova era, possibilitando o desenvolvimento dos microchips – pequenos dispositivos capazes de integrar múltiplos componentes eletrônicos, como resistores e capacitores, em uma estrutura compacta.

O avanço na miniaturização dos circuitos e no poder de processamento culminou nos microcontroladores, que são chips programáveis capazes de realizar várias funções de um computador em uma estrutura pequena e econômica. Esses microcontroladores são essenciais para o funcionamento de inúmeros dispositivos que usamos no dia a dia, desde eletrodomésticos até sistemas de controle de veículos, pois são circuitos integrados que podem processar informações e executar comandos predefinidos.

Conversão de energia em informação

Para entender o funcionamento de um microcontrolador e como ele transforma energia em instruções úteis, é essencial compreender o conceito de abstração. Abstração é uma forma de simplificar a realidade, tornando mais fácil representar e manipular conceitos complexos. Por exemplo, ao chamarmos um circuito integrado de "microcontrolador", estamos usando uma abstração: essa palavra engloba todos os componentes internos e as funções que ele realiza em um único termo, permitindo que possamos falar dele sem entrar em detalhes técnicos cada vez que o mencionamos.

No núcleo do funcionamento de um microcontrolador está a energia elétrica. Para operar, o microcontrolador precisa de um fluxo constante de eletricidade, que é transmitido por meio de pulsos elétricos. Esses pulsos são interpretados em uma forma que o microcontrolador consegue "entender" através de um sistema binário de representação, onde um pulso é identificado como 1 e a ausência de um pulso é identificada como 0. Esses 1s e 0s são chamados de bits, as menores unidades de informação em computação.

Os bits são organizados e interpretados pelo **kernel** – o "cérebro" que controla a operação do microcontrolador. O kernel converte esses bits em bytes (conjuntos de oito bits), que, por sua vez, podem representar instruções mais complexas, como números, caracteres e até mesmo comandos específicos. Esse processo de conversão é uma verdadeira tradução da energia elétrica em linguagem binária, permitindo que as instruções do usuário sejam entendidas pelo microcontrolador.

Quando o usuário interage com um sistema controlado por um microcontrolador, ele está, essencialmente, criando uma sequência de comandos que serão convertidos em linguagem binária, interpretados e executados pelo microcontrolador. Em outras palavras, o usuário cria regras e define comportamentos por meio de um processo de programação, que, no final, se traduz em uma sequência de pulsos elétricos, permitindo que o dispositivo funcione conforme o desejado.

Estrutura de um microcontrolador

O microcontrolador é, essencialmente, um tipo de circuito integrado. Mas, afinal, o que é um circuito integrado e como ele funciona? Podemos compará-lo a uma bicicleta, onde cada parte desempenha uma função específica que contribui para o movimento como um todo. Assim como a bicicleta tem rodas, um guidão e uma estrutura que conecta tudo, o microcontrolador tem seus próprios componentes fundamentais que, juntos, formam uma unidade de processamento compacta e autossuficiente.

Imagine a estrutura de uma bicicleta. As rodas, por exemplo, têm um formato cilíndrico que permite rotação fácil, tornando a locomoção mais eficiente. No microcontrolador, o "equivalente" das rodas são os transistores. Eles funcionam como pequenos interruptores que controlam o fluxo de corrente, permitindo a passagem de informações e comandos. Assim como as rodas movem a bicicleta para frente, os transistores impulsionam os dados dentro do microcontrolador.

Além das rodas, a bicicleta possui um guidão, que permite ao usuário controlar a direção. No microcontrolador, esse papel é realizado pela unidade de controle, que direciona o fluxo das instruções e define a ordem de execução das operações. O guidão (unidade de controle) é essencial para garantir que todos os comandos sigam o caminho correto e atinjam o destino certo.

Há também a estrutura da bicicleta em si, que mantém tudo conectado. No microcontrolador, essa estrutura é formada por circuitos que interligam todos os componentes, como memória, unidade de processamento e portas de entrada e saída. Esses circuitos conectam cada "peça" para que todas trabalhem em harmonia, permitindo que o microcontrolador execute suas tarefas de forma eficiente.

Por que essa estrutura é necessária?

A estrutura de um microcontrolador é cuidadosamente planejada com base em princípios de engenharia elétrica. Cada componente tem um propósito específico e é posicionado de maneira a otimizar a comunicação entre as partes. Por exemplo, a unidade de processamento central (ou CPU) precisa de acesso rápido à memória, já que esta armazena instruções e dados que a CPU deve processar constantemente.

Outro ponto importante é a presença de portas de entrada e saída, que permitem a comunicação do microcontrolador com o mundo externo. Essas portas recebem sinais de sensores e enviam comandos para atuadores, controlando, por exemplo, motores ou displays. Essas conexões externas precisam estar localizadas em áreas específicas do microcontrolador para facilitar o fluxo de dados e a eficiência energética.

Essa organização interna foi desenvolvida para atender às necessidades de desempenho, custo e tamanho reduzido, fatores essenciais para o uso de microcontroladores em dispositivos

pequenos, como eletrodomésticos, automóveis e dispositivos de IoT. Cada parte da estrutura interage de forma a minimizar os gastos de energia e maximizar a velocidade de processamento.

Arquiteturas de Microcontroladores

A forma como os componentes de um microcontrolador são organizados e interagem entre si é conhecida como arquitetura. Existem várias arquiteturas, cada uma com características próprias que impactam a performance, a eficiência e o custo do microcontrolador. Aqui estão algumas das arquiteturas mais relevantes:

Arquitetura Harvard

Na arquitetura Harvard, a memória de programa e a memória de dados são separadas fisicamente, ou seja, cada tipo de dado (instruções e informações de dados) possui seu próprio barramento. Essa separação permite que o microcontrolador acesse dados e instruções simultaneamente, aumentando a velocidade de processamento.

- Vantagens:
    - Maior eficiência e velocidade, já que permite acessar dados e instruções ao mesmo tempo.
    - Ideal para aplicações em que a performance é prioridade, como sistemas de controle em tempo real.

- Desvantagens:
    - Estrutura mais complexa e, portanto, mais cara.
    - Limitação de flexibilidade, pois é mais difícil modificar as instruções e os dados dinamicamente.

Arquitetura Von Neumann

Na arquitetura Von Neumann, também conhecida como Arquitetura de Armazenamento Comum, tanto dados quanto instruções compartilham a mesma memória e o mesmo barramento. Nesse caso, o microcontrolador deve alternar entre acessar dados e instruções, pois utiliza o mesmo caminho para ambos.

- Vantagens:
    - Estrutura mais simples e custo reduzido.
    - Mais fácil de programar e modificar, pois dados e instruções estão na mesma área de memória.
- Desvantagens:
    - Velocidade limitada, pois não é possível acessar dados e instruções simultaneamente.
    - Possível ocorrência de "gargalos", já que o barramento único pode ficar sobrecarregado.

Arquitetura RISC (Computador com Conjunto Reduzido de Instruções)

A arquitetura RISC é projetada para processar um conjunto limitado de instruções simples, mas de maneira extremamente eficiente. Cada instrução leva apenas um ciclo de clock para ser executada, tornando essa arquitetura muito rápida e eficiente para operações repetitivas.

Vantagens:

Altamente eficiente e com menor consumo de energia.

Ideal para microcontroladores em sistemas embarcados, como smartphones e dispositivos portáteis.

Desvantagens:

Limitado em termos de complexidade de instruções.

Exige mais instruções para realizar operações complexas, o que pode aumentar o tempo de execução em algumas aplicações.

Arquitetura CISC (Computador com Conjunto Complexo de Instruções)

A arquitetura CISC, por outro lado, é projetada para realizar operações complexas com poucas instruções. Cada instrução pode realizar várias operações internas, o que diminui a quantidade de instruções necessárias para uma tarefa.

- Vantagens:

    - Capaz de realizar operações complexas com poucas instruções.
    - Ideal para tarefas que exigem manipulação complexa de dados.
- Desvantagens:
    - Estrutura mais complexa, exigindo mais energia para funcionar.
    - Menor eficiência em comparação à arquitetura RISC para operações simples e repetitivas.

Estrutura Interna e a Comunicação com a Camada Superior

No núcleo de um microcontrolador, encontramos componentes fundamentais que realizam as operações lógicas e matemáticas, como o kernel e a unidade aritmética e lógica (ALU). Esses dois elementos são responsáveis por transformar as instruções do código em ações que o microcontrolador executará. Vamos explorar cada um desses componentes e entender como eles se comunicam entre si e com as camadas de abstração superior.

O Kernel e a ALU: A Base do Processamento

O kernel é a unidade de gerenciamento central do microcontrolador, uma espécie de "gerente de operações" que organiza e direciona o fluxo de dados entre os componentes. Ele é responsável por coordenar a comunicação entre as diferentes partes do microcontrolador, garantindo que cada operação aconteça no momento certo e com o recurso necessário.

A ALU (Unidade Aritmética e Lógica) é a seção encarregada de realizar operações matemáticas básicas, como adição e subtração, além de operações lógicas, como comparações entre valores. Sempre que o microcontrolador precisa realizar uma operação matemática ou lógica, o kernel envia a instrução para a ALU, que processa o cálculo e devolve o resultado.

Essa interação entre kernel e ALU é constante e ocorre em alta velocidade, já que a maioria das tarefas do microcontrolador envolve cálculos e decisões baseadas nesses cálculos. A comunicação é facilitada pelo uso de barramentos – linhas de conexão que permitem a transferência de dados e instruções entre o kernel, a ALU e outros componentes, como a memória e as portas de entrada e saída.

Comunicação com a Camada de Abstração Maior

Para entender como o kernel e a ALU interagem com camadas de abstração superiores, é importante lembrar que os microcontroladores funcionam em níveis organizados. As instruções que o usuário vê – o código em uma linguagem de programação, como C ou Python – são, na verdade, abstrações que precisam ser "traduzidas" para que o microcontrolador as entenda.

Essa tradução é feita pelo compilador, que converte o código de uma linguagem de alto nível em uma linguagem de máquina que o kernel e a ALU conseguem interpretar e executar.

O Papel do Compilador

O compilador é uma ferramenta essencial no desenvolvimento de programas para microcontroladores. Sua função é transformar o código-fonte, escrito em uma linguagem de programação de alto nível (fácil de entender para os

programadores), em código de máquina (composto por 0s e 1s) que o microcontrolador pode processar diretamente.

Existem diferentes tipos de compiladores, e eles variam em funcionalidade e forma de operação:

Tipos de Compiladores

1. Compiladores Diretos: Esses compiladores traduzem o código de alto nível diretamente para o código de máquina, sem etapas intermediárias. Esse processo é eficiente em termos de tempo de compilação e resulta em um código executável otimizado para o hardware específico do microcontrolador. No entanto, esse tipo de compilador pode ser menos flexível, pois é frequentemente desenvolvido para um hardware específico.
2. Compiladores Cruzados (Cross Compilers): Os compiladores cruzados permitem que o programador desenvolva o código em um sistema (por exemplo, um computador) e o compile para outro sistema, que é o microcontrolador. Isso é muito útil, pois permite o desenvolvimento em um ambiente mais poderoso, que facilita a escrita e o teste do código.
3. Compiladores Interpretados: Esses compiladores funcionam por meio de uma "tradução" em tempo real, onde cada linha do código é convertida para código de

máquina conforme a execução progride. Embora sejam mais lentos do que os compiladores diretos, os interpretadores permitem maior flexibilidade e facilidade de teste, sendo úteis para a depuração e verificação de erros.

Linguagens de Programação para Microcontroladores

Após o código ser compilado, ele precisa ser executado pelo microcontrolador, e as linguagens de programação desempenham um papel fundamental nesse processo. Existem várias linguagens usadas para programar microcontroladores, cada uma com características que a tornam mais adequada para certos tipos de projetos. Abaixo estão algumas das linguagens mais comuns:

Linguagens de Programação Mais Utilizadas

1. C: A linguagem C é, de longe, a mais popular para programação de microcontroladores. Ela é extremamente eficiente e permite controle detalhado sobre os recursos de hardware, tornando-a ideal para sistemas embarcados. Em C, o programador pode manipular diretamente registradores e portas de entrada e saída, o que é essencial em projetos de baixo nível.
    - Vantagens: Eficiência, controle de hardware, performance elevada.

- Desvantagens: Exige conhecimento técnico, menor abstração.

2. Assembly: A linguagem Assembly fornece controle direto e específico do hardware, permitindo que o programador otimize ao máximo o desempenho do microcontrolador. Embora seja complexa e difícil de aprender, é ideal para programadores que precisam de um controle absoluto sobre o sistema.
    - Vantagens: Controle completo e precisão, máxima eficiência.
    - Desvantagens: Complexidade, código difícil de ler e manter.
3. Python: Embora Python seja uma linguagem de alto nível e geralmente mais lenta, ela está sendo usada em microcontroladores com mais frequência, especialmente com plataformas como o MicroPython. Essa linguagem é ideal para projetos mais simples e para prototipagem rápida.
    - Vantagens: Facilidade de uso, legibilidade, ideal para prototipagem.
    - Desvantagens: Desempenho inferior em comparação com C ou Assembly.

4. Arduino (derivado de C/C++): Para microcontroladores Arduino, existe uma linguagem simplificada derivada de C/C++. Essa linguagem é ideal para iniciantes e permite uma programação mais amigável, com muitas funções pré-definidas que simplificam o desenvolvimento.
    - Vantagens: Simplicidade, facilidade de uso, vasta documentação.
    - Desvantagens: Menor controle do hardware em comparação ao C puro.

Por Que Essas Linguagens São Usadas?

Essas linguagens foram escolhidas para a programação de microcontroladores por uma combinação de fatores: eficiência, controle e facilidade de uso. A linguagem C, por exemplo, é utilizada principalmente devido ao seu excelente balanço entre controle direto do hardware e facilidade de leitura e manutenção, tornando-a a escolha preferida para a maioria dos sistemas embarcados.

Por outro lado, Assembly é usada em casos específicos onde cada ciclo de clock é essencial, permitindo uma otimização máxima, enquanto Python, apesar de mais lento, é amplamente adotado para projetos de prototipagem e para usuários que valorizam simplicidade e rapidez.

Cada linguagem tem suas próprias características e é adequada para tipos específicos de aplicações. A escolha depende das

necessidades do projeto e da experiência do programador. Em sistemas onde o controle do hardware e a velocidade são críticos, C ou Assembly são preferidos. Para projetos que exigem prototipagem rápida e desenvolvimento de código legível, Python e a linguagem Arduino se destacam.

Projeto com Arduino Uno

O Arduino Uno é uma plataforma de prototipagem eletrônica amplamente utilizada devido à sua facilidade de uso e flexibilidade. Com ele, é possível criar uma vasta gama de projetos, desde automação residencial até sistemas de monitoramento ambiental. Neste capítulo, exploraremos um projeto básico que demonstra os conceitos fundamentais da programação de microcontroladores: o controle de uma lâmpada. Embora simples, esse projeto serve como base para projetos mais complexos e pode ser adaptado para diversas aplicações práticas.

O Projeto

O **objetivo** é desenvolver um sistema que permita ligar e desligar uma luz de casa usando um componente físico ou um sensor para isso é preciso:

- 1 x Placa Arduino Uno
- 1 x Relé de 5V
- 1 x Lâmpada (pode ser uma lâmpada LED de baixa potência)
- 1 x Interruptor ou botão ou eté mesmo um sensor
- Jumpers e protoboard (se necessário)

Detalhando

Arduino UNO (figura 1) "é uma placa de microcontrolador ele contém tudo o que é necessário para suportar o microcontrolador; basta conectá-lo a um computador com um cabo USB ou ligá-lo com um adaptador AC-DC ou bateria para começar. Você pode mexer com sua UNO sem se preocupar muito em fazer algo errado, no pior cenário, você pode substituir o chip por alguns dólares e começar de novo. – Documentação do Arduino".

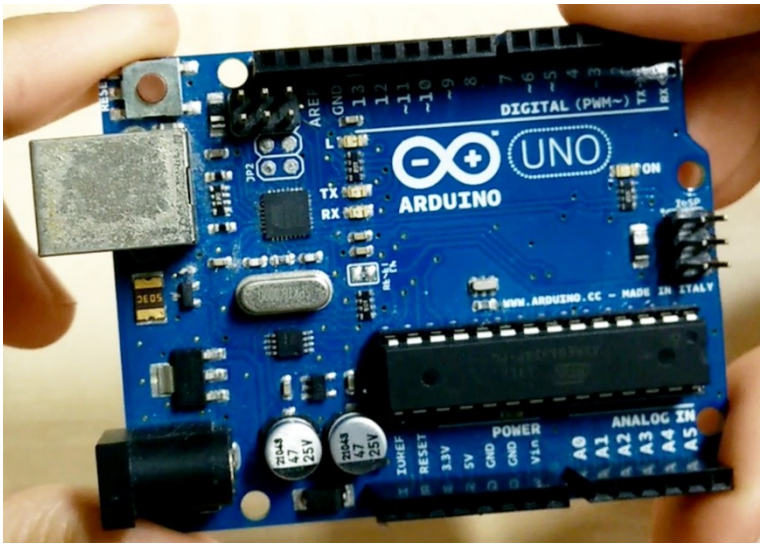


Figura 1. Placa Arduino UNO

Relé 5V é um componente eletrônico que funciona como um interruptor controlado eletricamente. Ele permite que você ligue ou desligue circuitos de maior potência usando um sinal elétrico de baixa potência, como o fornecido por um microcontrolador como o Arduino.

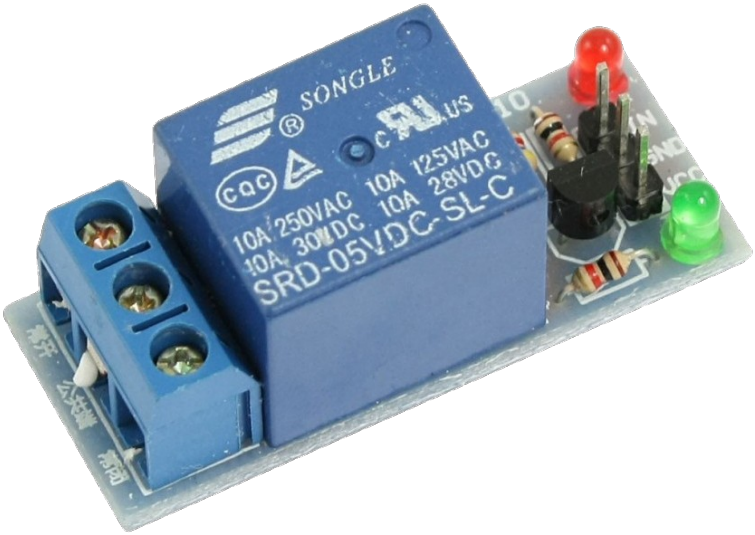


Figura 2. Relé 5V

Mão na masssa

Se estiver usando um protoboard, insira os componentes e conecte-os usando os jumpers, agora vamos as conexões:.

- VCC do relé: Conecte ao pino 5V do Arduino.
- GND do relé: Conecte ao GND do Arduino.
- IN do relé: Conecte a um pino digital do Arduino (por exemplo, pino 2).
- NO do relé: Conecte a um terminal da lâmpada.
- COM do relé: Conecte ao outro terminal da lâmpada.
- Lâmpada: Conecte a lâmpada a uma fonte de alimentação adequada.

Ps. Nas placas a uma identificação para cada conexão.

Código Arduino

O código para controlar o relé e, consequentemente, a lâmpada, é simples. O interruptor irá acionar a lâmpada para ligá-la ou desligá-la. Então para começar, definimos quais pinos do Arduino serão utilizados para controlar o botão e o relé. Declaramos as variáveis *interruptor* e *rele* como do tipo int (inteiro), pois os pinos digitais do Arduino são representados por números inteiros. Utilizamos o tipo int para garantir que o Arduino reserve a quantidade correta de memória (4 bytes) para armazenar esses valores. E dexaremos o interrruptor pré configurado como 0 (zero) o que por conveção representa desligado.

```
const int interruptor = 2;
const int rele = 7;

int estadoInterruptor = 0;
```

Devemos ser muito precisos ao escrever o código, pois o microcontrolador não 'entende' o que queremos fazer se não declararmos explicitamente cada variável e cada ação e por isso precisamos definir o estado inicial do interruptor como 0 (zero) antes de começarmos a controlar o relé.

```
int estadoInterruptor = 0;
```

Uma analogia: a função matemática, como $f(x) = x^2 + 2ab - c$, é uma relação entre uma entrada x e uma saída (o resultado do

cálculo). Na programação, as funções têm essa e outras funções, pois como é mencionado na obra *Structure and Interpretation of Computer Programs* de Harold Abelson, Gerald Jay Sussman e Julie Sussman: "Em matemática, geralmente nos preocupamos com descrições declarativas (o que é), enquanto na ciência da computação, geralmente nos preocupamos com descrições imperativas (como fazer)". Nem sempre precisamos que a função nos retorne um resultado, e as variáveis podem ser também parâmetros.

Entendido o que é função, vamos usar da abstração função para dizer a máquina que queremos que o pino interruptor esteja preparado para qualquer INPUT (entrada) e o pino rele tenha a função de OUTPUT (saída), o que irá permitir que o Arduíno envie um sinal para a relé, ligando ou desligando a carga conectada a ele.

```
void setup() {
     pinMode(interruptor, INPUT);
     pinMode(rele, OUTPUT);
}
```

A função loop() será executada repetidamente em quanto o sistema estiver ligado, verificando constantemente o estado do interruptor. Em cada iteração, ela armazena o estado atual do interruptor em uma variável temporária. Em seguida, compara esse novo estado com o estado armazenado anteriormente. Se houver divergência, a função atualiza o estado armazenado e aciona o relé para ajustar o estado da lâmpada. Dessa forma, a

lâmpada é ligada ou desligada automaticamente sempre que o interruptor é acionado.

```
void loop() {
     int novoEstado =
digitalRead(interruptor);

     if (novoEstado ≠ estadoInterruptor) {
          estadoInterruptor = novoEstado;
          digitalWrite(rele,
estadoInterruptor);
     }
}
```

Ao controlar uma lâmpada de forma automatizada, este projeto demonstra o potencial do Arduino em transformar nossas casas em ambientes mais inteligentes e eficientes. Essa mesma lógica pode ser aplicada em diversos outros projetos, como sistemas de irrigação automática, controle de temperatura e abertura de portões, abrindo um leque de possibilidades para a automação residencial.

Referências

"Arquitetura de von Neumann." Wikipedia, 6 May 2021, pt.wikipedia.org/wiki/Arquitetura_de_von_Neumann. Accessed 2 Nov. 2024.

"Arquitetura Harvard." Wikipedia, 20 Oct. 2021, pt.wikipedia.org/wiki/Arquitetura_Harvard. Accessed 2 Nov. 2024.

"CISC." Wikipedia, 20 Oct. 2020, pt.wikipedia.org/wiki/CISC. Accessed 2 Nov. 2024.

Penido, Édilus de Carvalho Castro, and Ronaldo Silva Trindade. Ouro Preto -MG Microcontroladores. 2013.

"RISC." Wikipedia, 20 Oct. 2020, pt.wikipedia.org/wiki/RISC. Accessed 2 Nov. 2024.

Sousa, Fábio Guimarães de . "Arquitetura de Um Microcontrolador - Aula 3 - MC - Mundo Projetado." Mundoprojetado.com.br, 30 Jan. 2020, mundoprojetado.com.br/arquitetura-de-um-microcontrolador/. Accessed 2 Nov. 2024.

Stallings, William. Arquitetura E Organização de Computadores. 8th ed., São Paulo (Sp), Pearson, 2010.