Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Henrique Manuel Lages Pereira Ribeiro Bernardino nº17295
João Filipe Coelho da Costa Felício nº16567

# Utilização de Microcontrolador na redução de consumos de elevadores

**Projeto de Licenciatura**

Engenharia Eletrotécnica

Prof. António Ferreira e Mestre Paulo Correia

Setembro de 2023

**“O sucesso é a soma de pequenos esforços repetidos dia após dia”**

**Robert Colli**

# Agradecimentos

O projeto é o resultado de um percurso académico que não teria sido possível realizar sem o precioso apoio e colaboração de várias pessoas, às quais queremos expressar os nossos sinceros agradecimentos por nos terem ajudado.

Agradecemos à Instituição que tornou possível a realização deste projeto, a Escola Superior de Tecnologia e Gestão de Viseu.

Dirigimos também um agradecimento aos Professor António Ferreira e Paulo Correia, nossos orientadores, não só pelos riquíssimos conhecimentos que nos transmitiram e que, em parte, contribuíram para o sucesso deste projeto, como também pela oportunidade, disponibilidade e atenção que nos proporcionaram.

Agradecemos a todos os docentes por todos os conhecimentos, dedicação e contributo transmitidos ao longo do nosso curso de Licenciatura em Engenharia Eletrotécnica.

Queremos agradecer também de forma especial a todos os familiares e amigos que, direta ou indiretamente, influenciaram este percurso académico, através do seu companheirismo e amizade.

# Resumo

O presente documento apresenta a descrição do trabalho desenvolvido no âmbito da unidade curricular “Projeto”, da Licenciatura em Engenharia Eletrotécnica e que versa o desenvolvimento de um protótipo de dois elevadores de seis andares conjugados. Ao longo deste é feita uma pequena revisão dos conceitos e das tecnologias associadas.

Este projeto tem como objetivo, com a ajuda de um microcontrolador, otimizar a eficiência energética, em dois elevadores de 6 andares.

O sistema é composto por uma unidade microcontrolador, implementado um sistema onde os elevadores se movem de acordo com a necessidade do utilizador, quando chamado o elevador vai o elevador mais próximo, logo chega mais rápido e tem maior eficiência energética.

# Siglas

PWM - Pulse Width Modulation

IDE- Integrated Development Environment

ESTGV- Escola Superior de Tecnologia e Gestão de Viseu

# Índice Geral

# Índice de Figuras

# 1 Introdução

## 1.1 Âmbito e motivação

A Inovação não se baseia apenas no desenvolvimento de novos produtos, mas também no aperfeiçoamento e melhoramento dos já existentes, oferecendo novos serviços e melhorando os processos, de modo a tornar mais fácil a vida das pessoas e, acima de tudo, fazer com que esses progressos cheguem a quem necessita.

Para que isso aconteça, é importante que a inovação seja valorizada e devidamente reconhecida e, em vez de ser vista como um gasto, seja entendida como um investimento. Para tal, a sociedade deve estar aberta a novas ideias e evoluções e aproveitar as oportunidades que são oferecidas para melhorar a qualidade de vida das populações. Também é importante incentivar novas competências científicas e a aceitação de novas tecnologias, algo que requer o compromisso de todos os governantes, indústrias, escolas e universidades.

Pretendemos com o desenvolvimento deste protótipo, um equilíbrio entre o consumo de energia e o uso dos serviços básicos necessários ao nosso dia a dia. Os avanços tecnológicos relacionados com a produção de energia, transporte e o desenvolvimento de dispositivos de economia, desempenham um papel fundamental na busca por essa otimização energética.

Embora o conceito de eficiência energética não seja novo, é uma responsabilidade compartilhada por todos. Por esse motivo, entendemos que deve ser um compromisso de governos, empresas de energia e consumidores, a promoção dessa realidade.

## 1.2 Descrição do problema

Para a realização deste trabalho foi-nos proposta a construção de um sistema de dois elevadores e, com a ajuda de um microcontrolador programar o sistema de modo que este comunique entre si e identifique qual o elevador que está mais próximo do andar pretendido e em seguida este desloca-se até ao andar para o qual foi chamado.

Idealizámos um sistema versátil, de baixo custo, que faça e possibilite o controlo dos elevadores de um edifício e em que a sua utilização seja automatizada de forma a minimizar tanto quanto possível os seus gastos energéticos.

## 1.3 Objetivos

Pretende-se desenvolver um protótipo para a implementação de um sistema baseado em microcontroladores que faça a otimização energética e mecânica, simulando o contexto real e tendo por base os seguintes requisitos:

• Criar uma solução que melhore a eficiência energética e mecânica utilizando componentes de baixo custo;

• Construção de um sistema composto por uma unidade microcontrolador que comunique com os dois elevadores, com recurso a linguagem de programação.

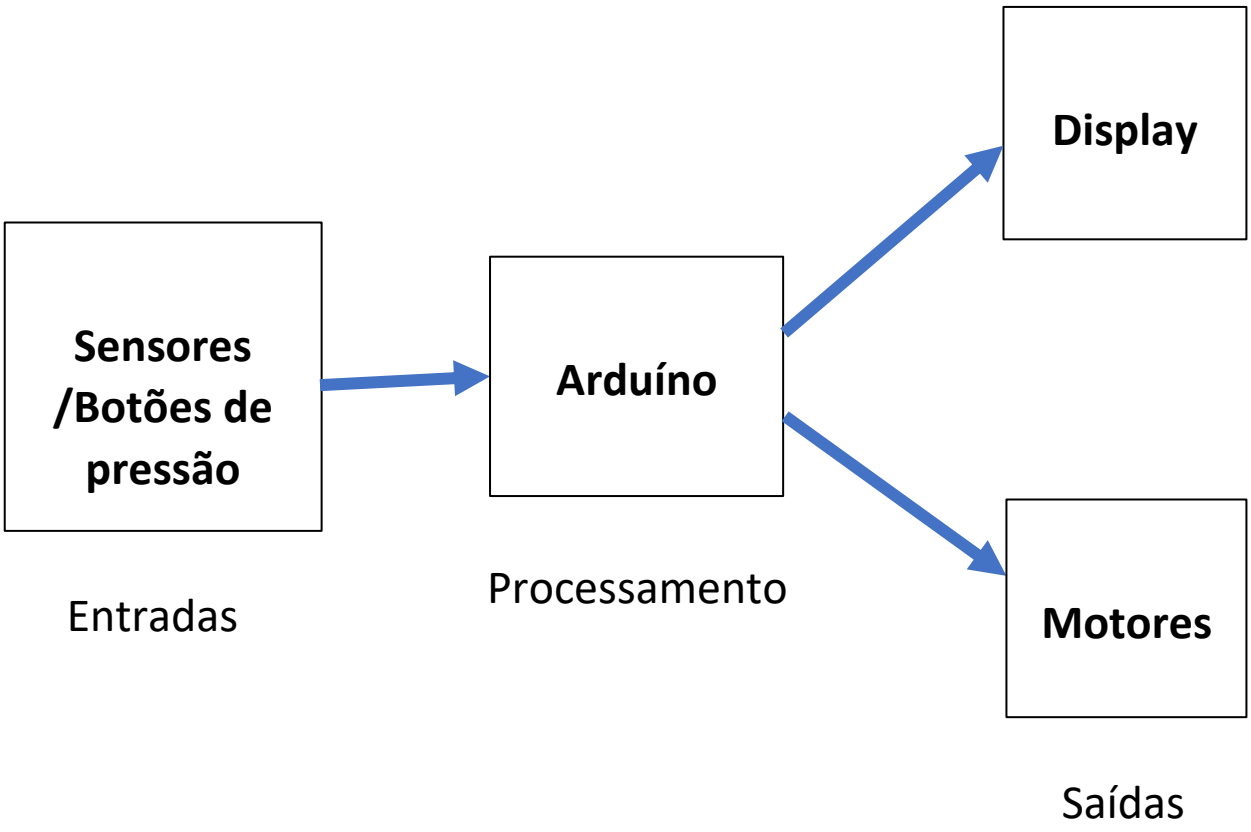
## 1.4 Contribuição

Este trabalho contribui com o desenvolvimento de um protótipo de sistema baseado em microcontroladores que faça a otimização dos consumos energéticos de um sistema de dois elevadores.
O protótipo é capaz de identificar o andar em que é requisitado o elevador e de forma a poupar tempo e energia, designar o elevador mais que está próximo para fazer a deslocação para esse mesmo andar.
.

## 1.5 Estrutura do documento

Este relatório encontra-se organizado em quatro capítulos. No primeiro capítulo é feita a introdução do trabalho onde são apresentadas as motivações, a descrição do problema, os objetivos e a estrutura do documento. No segundo capítulo são apresentados os componentes do projeto. No terceiro capítulo são apresentadas todas as ligações do projeto desenvolvido e é dada a explicação do seu funcionamento. Finalmente, no quarto capítulo, faz-se uma revisão do trabalho desenvolvido, síntese das nossas contribuições e sugeridas algumas direções para trabalho futuro.

# 2 Componentes do Projeto



## 2.1 Arduíno

A placa Arduíno Mega 2560 é uma placa da plataforma Arduíno que possui recursos bastante interessantes para protótipos e projetos mais elaborados. Possui 54 pinos de entradas e saídas digitais, onde 15 destes podem ser utilizados como saídas PWM. Possui 16 entradas analógicas e 4 portas de comunicação serial. Esta será uma ótima opção para projetos que necessitem de muitos pinos de entrada e saída, pois, para além da quantidade de pinos também conta com uma grande quantidade de memória.

A escolha deste Arduíno para o nosso projeto, baseou-se na excelente capacidade de processamento, o que é importante, já que há necessidade de lidar com várias entradas e saídas, controle de motores, sensores, botões e lógica de programação e, também, pelo seu número elevado de entradas e saídas, uma vez que existem bastantes sensores e botões, bem como, ainda, pela sua capacidade de memoria, que nos é útil para armazenar informações sobre o estado dos elevadores durante a sua utilização, se o Arduíno ficar sem alimentação, quando ligado novamente ele começa o ciclo de novo.

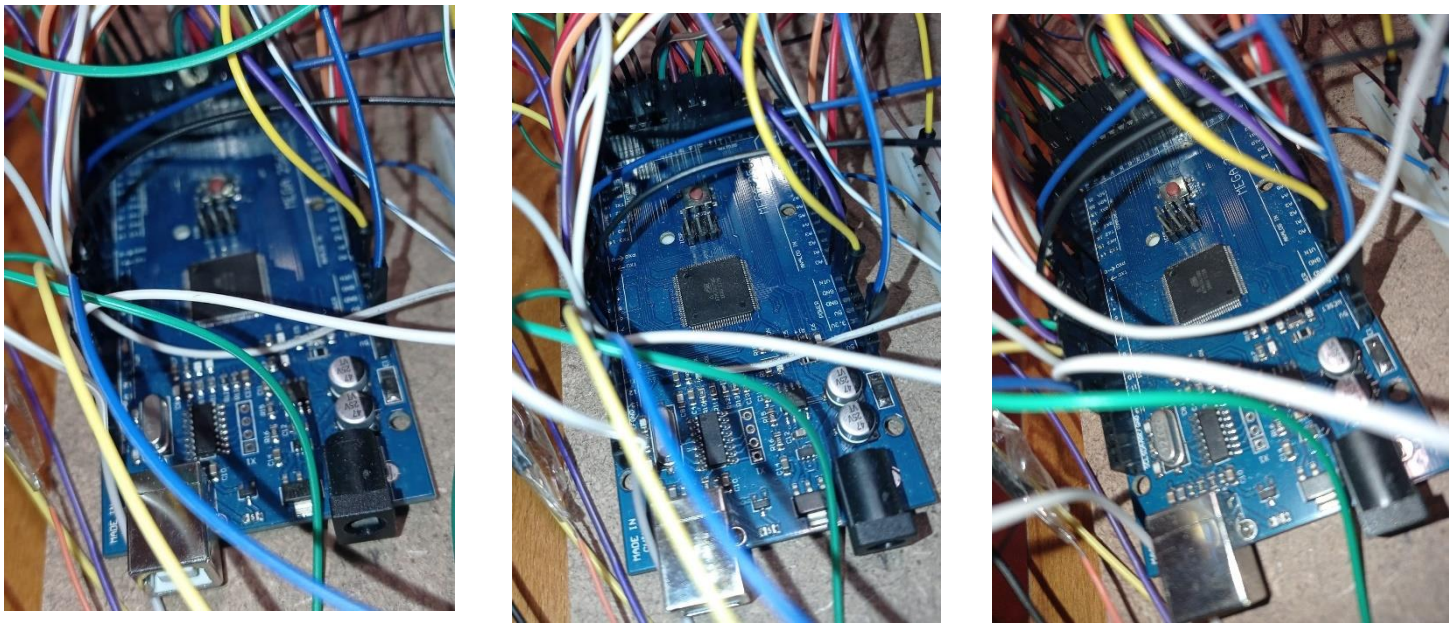


Figura 1- Arduíno

## 2.2 Motor redutor DC 12V

Optamos por usar um motor redutor DC 12V reversível para alto torque de 50 RPM.

Este é um dispositivo que combina um motor elétrico de corrente contínua (DC) de 12 volts com um sistema de redução de velocidade para produzir um alto torque de saída a uma velocidade de rotação de 50 rotações por minuto (RPM).

Assim asseguramos que eleva os nossos 2 elevadores bem como assegura um movimento lento e suave pois esta roda a 5 RPM.


Figura 2 - Motor redutor DC 12V

## 2.3 Displays Led

Os Kingbright SC52-11SRWA são um tipo de display de LED, mais especificamente, um display de sete segmentos vermelhos de 0,52 polegadas, produzidos pela Kingbright, que é um fabricante conhecido de componentes eletrónicos, incluindo displays de LED.

O display SC52-11SRWA é composto por sete segmentos individuais, além de um ponto decimal, acessível, de forma a permitir a exibição de dígitos de 0 a 9 e caracteres alfanuméricos básicos. A cor do LED é vermelha e sua altura é de 0,52 polegadas, o que o torna relativamente pequeno, em termos de displays de sete segmentos.

Esse tipo de display é amplamente utilizado em aplicações eletrónicas e industriais e aonde é necessário exibir números, tais como relógios digitais, medidores, equipamentos de teste, painéis de controle, entre outros dispositivos.

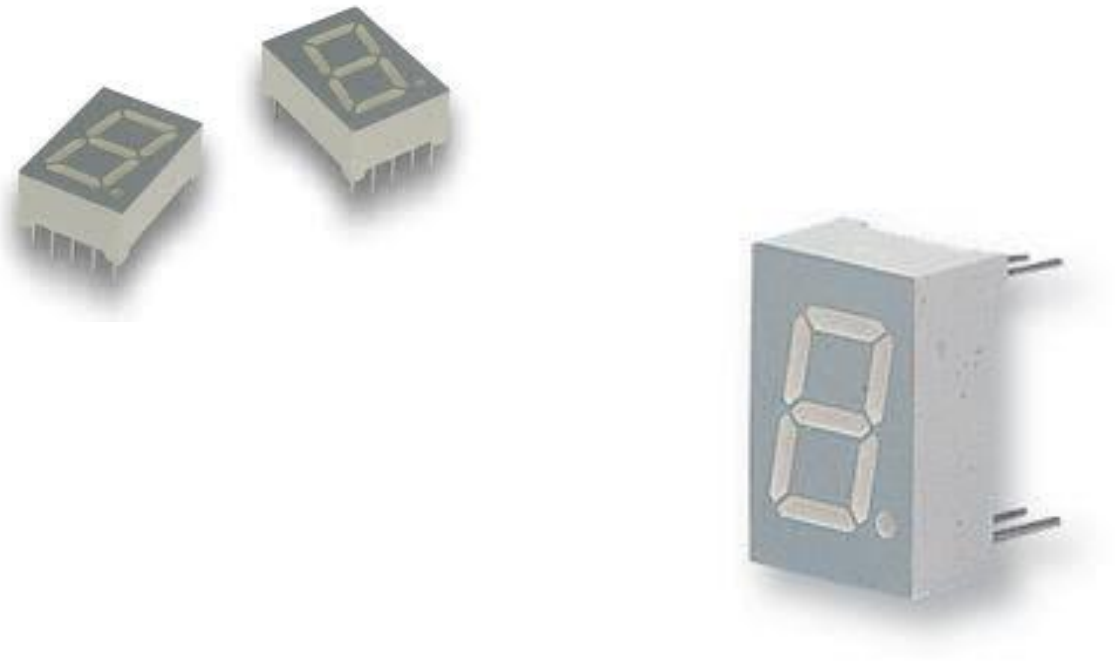


Figura 3 - Display Led

## 2.4 Sensores fim de curso

Sensores de fim de curso são dispositivos eletrónicos usados para detetar a posição de um objeto num sistema mecânico. Eles são geralmente usados em máquinas industriais, equipamentos robóticos, sistemas de automação e outros dispositivos que exigem controle preciso de movimento. Esses sensores são projetados para fornecer um sinal de saída quando um objeto ou componente mecânico atinge uma posição específica, geralmente indicando o fim de um percurso ou um limite predefinido. O sinal de saída pode ser através de um interruptor, com sinal digital ou analógico, dependendo do tipo de sensor. Existem diferentes tipos de sensores de fim de curso, sendo os mais comuns os interruptores de fim de curso mecânicos, que são dispositivos com contatos físicos, que são ativados ou desativados quando uma peça mecânica atinge a posição desejada.

Nos elevadores não são só utilizados sensores fins de curso, pois também podem ser utilizados sensores magnéticos, sensores de proximidade, sensores de velocidade e sensores de peso.

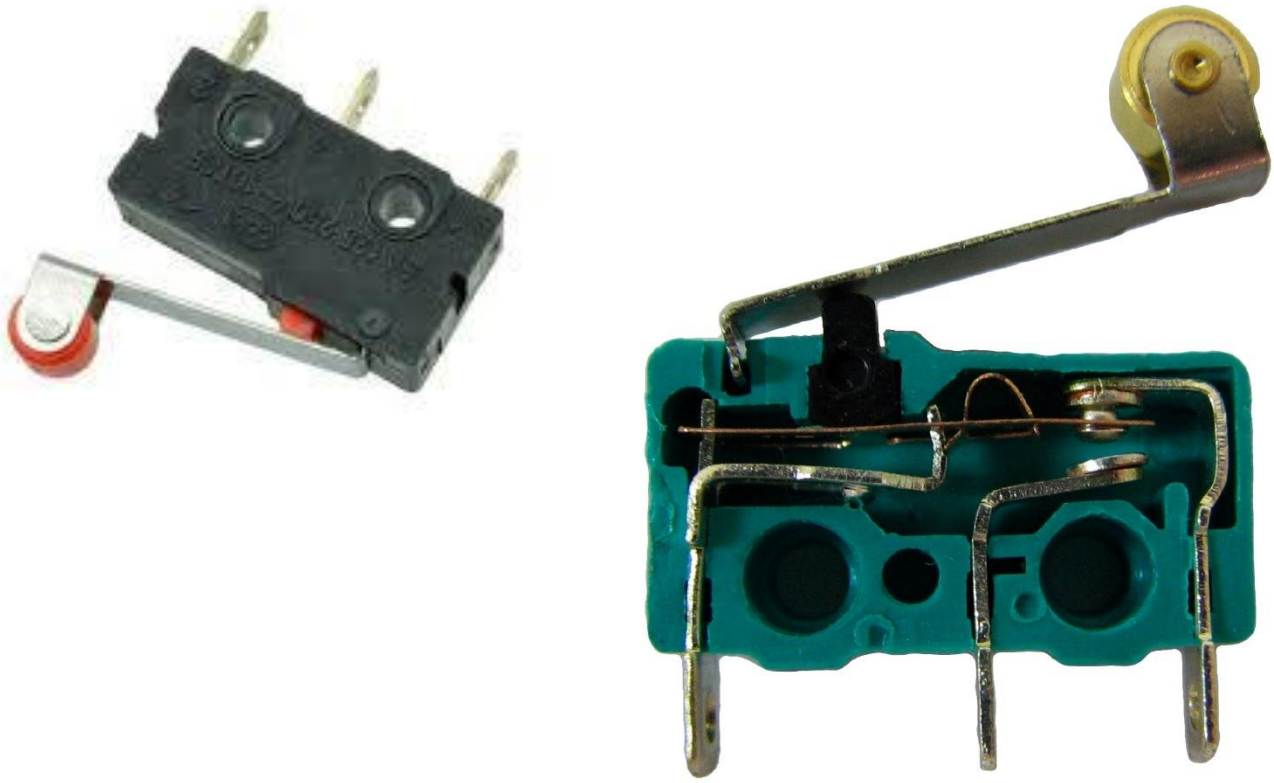


Figura 4 - Sensor fim de curso

## 2.5 Botões de pressão

Botões de pressão, também conhecidos como botões momentâneos, são dispositivos usados para controlar circuitos elétricos temporariamente, com a operação ocorrendo apenas quando e enquanto o botão é acionado. Eles são muito utilizados em dispositivos eletrónicos, como aparelhos domésticos e equipamentos industriais. Estes botões possuem um corpo que tem um mecanismo interno, com mola e conexões elétricas. Quando o botão é acionado, os contatos são fechados, estabelecendo uma ligação elétrica temporária. Ao ser libertada, a mola retorna os contatos à posição original, interrompendo a ligação elétrica. Esses botões fornecem um sinal momentâneo de pulso ou iniciam uma ação num circuito. Podem ter diferentes formatos, tamanhos e configurações de contatos.

Figura 5 - Botões de pressão

## 2.6 Breadboard

Uma breadboard (também conhecida como placa de ensaio) é uma plataforma de teste utilizada para montar circuitos eletrónicos temporários

A breadboard é composta por uma base isolante feita de plástico com furos e uma matriz de contatos. Esses furos são conectados internamente por circuitos condutores de metal. Na maioria das breadboards, a matriz de contatos é organizada em várias linhas e colunas, facilitando a ligação dos componentes eletrónicos.

Uma breadboard geralmente possui duas seções principais: a seção central, onde a maioria dos componentes é inserida, e duas faixas de contatos na parte superior e inferior, que são conectadas eletricamente em linhas horizontais para a ligação de componentes comuns de um circuito.

As breadboards são uma ótima ferramenta para testar ideias rapidamente, projetar montar e fazer experiências com circuitos eletrónicos, sem a necessidade de desenvolvimento.

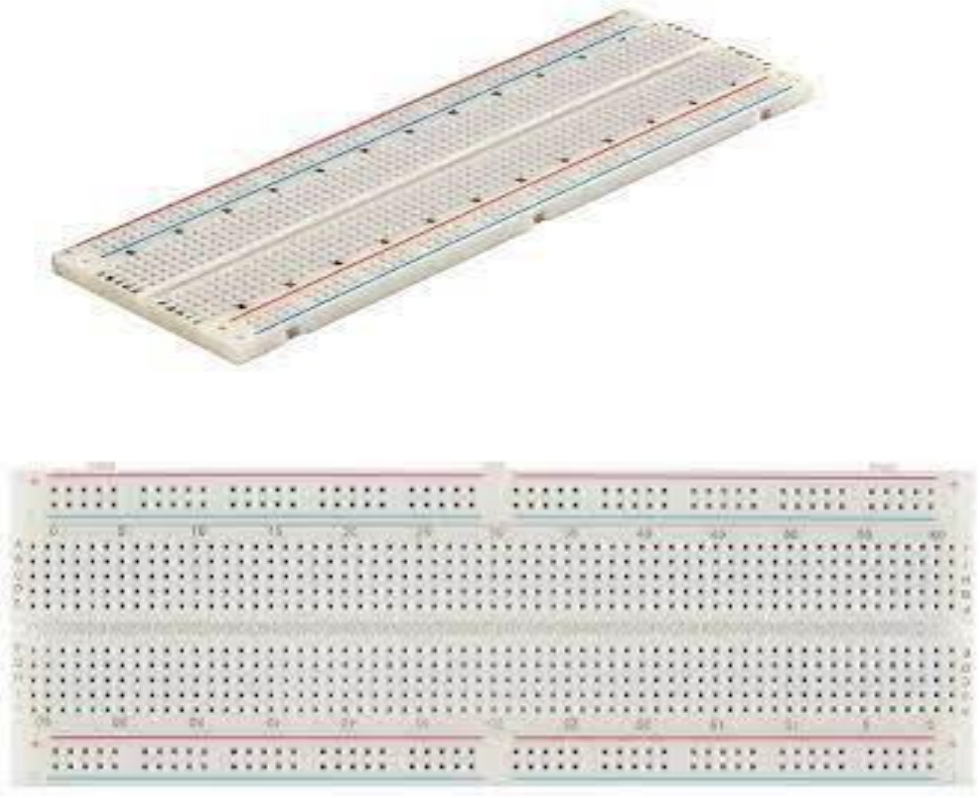


Figura 6 - Breadbord

## 2.7 L9110s

O L9110S é um “chip” controlador de motor duplo usado em projetos de robótica. Ele permite controlar a direção e a velocidade de dois motores DC separados. É amplamente utilizado em projetos de robôs e veículos autónomos. É fácil de usar e requer poucos componentes externos. É geralmente usado em conjunto com microcontroladores, como o Arduíno, para controlar os motores de forma precisa.

Suporta uma tensão de 5V a 28V e fornece uma corrente de saída contínua que vai até 1,5A. Possui alta eficiência energética e proteção contra sobrecarga térmica e curto-circuito.

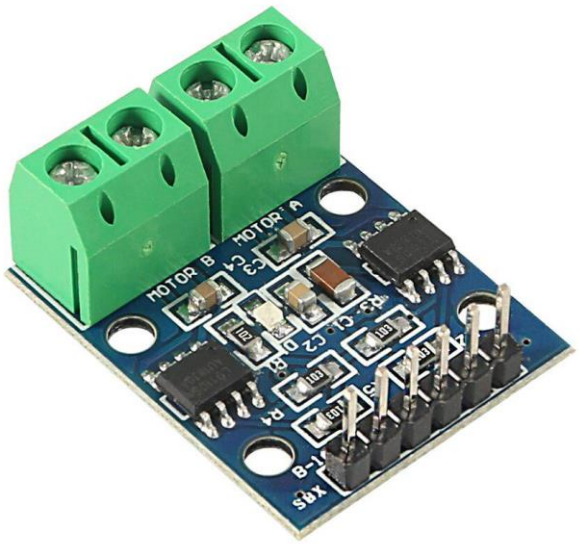



Figura 7 - L9110s

## 2.8 Software

A ferramenta de desenvolvimento utilizada para programar o microcontrolador, foi o IDE do Arduíno. Este permite-nos programar de forma prática e rápida o Arduíno do protótipo. Arduíno Integrated Development Environment é uma aplicação de plataforma cruzada, onde podemos escrever funções. É usado para escrever e fazer upload de programas em placas compatíveis com Arduíno, mas também, com a ajuda de núcleos de terceiros, outras placas de desenvolvimento de fornecedores.

## 2.9 Síntese

Após o desenvolvimento do projeto elétrico e da estrutura, iniciámos as fases de montagem da estrutura do protótipo e, de seguida, a programação do software.
A montagem deste protótipo foi feita por fases:

- Inicialmente montámos a estrutura, com material adquirido na escola e, seguidamente, fizemos o esquema elétrico;
- Montámos de seguida os componentes elétricos, fazendo as ligações entre o Arduíno e a restante estrutura.

Este protótipo foi preparado para simular dois elevadores controlados pelo microcontrolador, no qual o utilizador tem de pressionar os botões para fazer com que o elevador se movimente. O facto de os motores inicialmente escolhidos não terem potência suficiente, levou a termos de optar por dois motores mais robustos.
Para a montagem da estrutura de controlo foi utilizado;

| | |
|---|---|
| 1 | Fonte de alimentação |
| 1 | Arduíno Mega + cabo USB |
| 1 | Driver duplo ponte H L9110s |
| 12 | Sensores fins de curso |
| 18 | Botões para acionamento do elevador |

# 3 Montagem do circuito e o seu funcionamento

Neste capítulo apresentamos todas as ligações para a montagem do projeto e descrevemos o seu funcionamento.

## 3.1 Montagem do circuito

A estrutura do elevador foi feita de modo a conciliar o maior peso possível com custos e gastos de energia menores.

Fizemos um esboço em papel e requisitámos o material junto dos docentes.

Como já foi referido o facto de a estrutura não ser tão leve como inicialmente equacionávamos, levou a termos de optar por dois motores de maior potência.

A estrutura é feita de madeira, medindo esta 1,10 metros de atura, e com uma distância de 20 centímetros em cada andar.

*Figura 8- Maquete do projeto*

## 3.2 Ligações dos diversos componentes ao Arduíno

```
botaoAndarL1P1 = 2;
botaoAndarL1P2 = 3;
botaoAndarL1P3 = 4;
botaoAndarL1P4 = 5;
botaoAndarL1P5 = 6;
botaoAndarL1P6 = 7;

botaoAndarL2P1 = 41;
botaoAndarL2P2 = 43;
botaoAndarL2P3 = 45;
botaoAndarL2P4 = 47;
botaoAndarL2P5 = 49;
botaoAndarL2P6 = 51;

botaoEstrutura1 = 38;
botaoEstrutura2 = 40;
botaoEstrutura3 = 42;
botaoEstrutura4 = 44;
botaoEstrutura5 = 46;
botaoEstrutura6 = 48;

botaomanutençao = 53;

sensorAndar1Elevador1Pin = 8;
sensorAndar2Elevador1Pin = 9;
sensorAndar3Elevador1Pin = 10;
sensorAndar4Elevador1Pin = 11;
sensorAndar5Elevador1Pin = 12;
sensorAndar6Elevador1Pin = 13;

sensorAndar1Elevador2Pin = A0;
sensorAndar2Elevador2Pin = A1;
sensorAndar3Elevador2Pin = A2;
sensorAndar4Elevador2Pin = A3;
sensorAndar5Elevador2Pin = A4;
sensorAndar6Elevador2Pin = A5;

elevador1MotorPin1 = 22;
elevador1MotorPin2 = 23;
elevador2MotorPin1 = 24;
elevador2MotorPin2 = 25;

pinledA1 = 14;
pinledB1 = 15;
pinledC1= 16;
pinledD1= 17;
pinledE1 = 18;
pinledF1 = 19;
pinledG1 = 20;

pinledA2 = 27;
pinledB2 = 29;
pinledC2 = 31;
pinledD2 = 33;
pinledE2 = 35;
pinledF2 = 37;
pinledG2 = 39;
```

## 3.3 Funcionamento

Em primeiro lugar, são definidos os pinos necessários para os botões de chamada de cada andar, sensores de posição de cada andar para cada elevador, pinos para os motores dos elevadores, pinos para os LEDs dos displays de 7 segmentos.

Na função setup(), os pinos são configurados como entrada ou saída, dependendo da sua função no sistema. Por exemplo, os pinos dos botões de chamada e dos sensores de posição são configurados como entrada para ler os estados desses dispositivos, enquanto os pinos dos motores e dos LEDs são configurados como saída para controlar o movimento do elevador e exibir informações nos displays, respetivamente.

A função principal loop() é executada continuamente num loop infinito, e é onde ocorre o fluxo geral do funcionamento do sistema.

Primeiro, o sistema verifica se algum botão de chamada de andar foi pressionado. Se algum botão for pressionado, a função elevador_escolhido() vai calcular com base nos andares atuais dos elevadores, qual dos dois está mais perto do andar chamado e retorna qual dos elevadores se encontra mais perto. No entanto, se a posição de ambos for a mesma, o elevador que retorna da função será aquele que até ao momento foi chamado menos vezes. Seguidamente, através da função chamarElevador() é chamado o elevador correto e este inicia o movimento na direção correta. Esta função também determina a direção do elevador com base no andar atual e no andar chamado. Seguidamente, a função moverElevador() irá repetir-se até o elevador chegar ao andar desejado.

Assim, o sistema verifica todos os sensores de posição dos andares, e quando o sensor do andar desejado for pressionado, a função pararElevador() é chamada para parar o movimento do elevador e atualizar o andar atual.

O funcionamento dos botões referentes a cada elevador é o mesmo que o dos botões da estrutura, ou seja, o programa fica a aguardar na função loop() que estes botões (tanto do elevador 1 como do elevador 2) sejam acionados.

Quase a terminar a lista de funções, o sistema através da função display() atualiza os LEDs dos displays de 7 segmentos para mostrar o andar atual de cada elevador, fornecendo essa informação visual sobre a posição dos elevadores.

Por fim, e não menos importante, conseguimos através de um dos botões presentes junto às torres dos elevadores, acionar o modo “manutenção” dos elevadores. Este modo consiste em (independentemente do andar atual de cada elevador), colocar ambos os elevadores no 1ºandar. É possível verificar também um segundo botão, porém este associado à função reset do Arduino já predefinida. Este apenas reinicia todo o sistema, bloqueando assim os elevadores mesmo que estes estejam em movimento.

A função moverElevador() é responsável por controlar os motores dos elevadores. Ela recebe o número do elevador e a direção em que o elevador se deve mover. Com base nesses valores, a função aciona os pinos corretos para controlar o movimento do motor do elevador correspondente.

A função pararElevador() é chamada quando o elevador alcança o andar chamado ou é interrompido por algum sensor de posição de andar. Essa função para o movimento do elevador desativando os pinos do motor correspondentes.

Em resumo, o código permite que os elevadores se movam entre os andares quando são chamados por botões e parem nos andares corretos quando atingem o destino, com base na informação dos sensores de posição de cada andar. Os LEDs dos displays de 7 segmentos mostram o andar atual de cada elevador, fornecendo informações visuais aos usuários do sistema. Este projeto é apenas um exemplo simplificado, um sistema de controle de elevador real seria mais complexo e envolveria mais considerações de segurança e lógica para lidar com situações específicas.

Figura 9 -Fotos do projeto

## 3.4 Síntese

Durante a implementação do projeto, dedicámo-nos ao estudo minucioso da ligação entre os componentes e dispositivos envolvidos, bem como a toda a programação do Arduíno para que todos os componentes funcionassem em sintonia.

# 4 Conclusão

## 4.1 Revisão do trabalho

O protótipo do elevador colocou-nos perante vários desafios e logo, inicialmente, na montagem da estrutura. O maior desafio passou pela instalação dos sensores de fim de curso sendo que, na questão do software, também nos deparamos com alguns desafios. Inicialmente foi contextualizado o âmbito do nosso trabalho e realizado um trabalho de pesquisa sobre projetos semelhantes à nossa proposta.

De seguida, foram identificados o tipo de microcontrolador bem como os tipos de sensores/atuadores necessários.

Ao longo da realização do projeto foram surgindo dúvidas e desafios. Contudo, a vontade de alcançar os objetivos inicialmente propostos, o trabalho de pesquisa e a ajuda e disponibilidade dos nossos orientadores, levou a que conseguíssemos superar todas as dificuldades encontradas.

Este trabalho foi uma mais-valia para o nosso futuro, enquanto profissionais, já que nos deu a possibilidade de aprender, não só sobre o sistema microcontrolador utilizado, mas também sobre toda a programação e componentes que tivemos de interligar.

Durante a realização do projeto desenvolvemos um vasto conjunto de competências, das quais salientamos: sentido de responsabilidade, autonomia na realização de tarefas, capacidade de adaptação a situações imprevistas, comunicação e espírito de grupo.

No geral, foi um estudo realmente interessante e definitivamente instrutivo, elevando o grau do nosso conhecimento das matérias aplicadas.

## 4.2 Síntese da contribuição

O sistema é capaz de, conforme o andar onde o utilizador se encontre, gerir de forma mais eficiente da deslocação de cada um dos elevadores.
Este protótipo pode tornar-se num projeto viável e com grande potencial de expansão já que tem por base conceitos de funcionamento simples e acessíveis conseguido a partir de ferramentas e componentes de baixo custo.

## 4.3 Trabalhos futuros

Os objetivos inicialmente propostos foram atingidos. No entanto, é possível melhorar e aperfeiçoar alguns aspetos no sentido de aumentar o desempenho do sistema. Neste âmbito, apresentamos de seguida uma possível perspetiva de funcionalidades a desenvolver:

• Criar um sistema que faça ligação entre o microcontrolador e o telemóvel de modo que a sua monitorização possa ser feita á distancia;
• Expandir o sistema para mais unidades, com o objetivo de controlar vários elevadores;
• Monitorizar os padrões de uso do elevador e ajustar o tempo de espera em cada andar;
• Controlar o motor do elevador de maneira inteligente, ajustando a velocidade e a força de acordo com a carga e a necessidade.

# 5 Anexo

```
#include <stdio.h>

// Definição dos pinos para os botões de chamada de cada andar
const int botaoAndarL1P1 = 2;
const int botaoAndarL1P2 = 3;
const int botaoAndarL1P3 = 4;
const int botaoAndarL1P4 = 5;
const int botaoAndarL1P5 = 6;
const int botaoAndarL1P6 = 7;

const int botaoAndarL2P1 = 41;
const int botaoAndarL2P2 = 43;
const int botaoAndarL2P3 = 45;
const int botaoAndarL2P4 = 47;
const int botaoAndarL2P5 = 49;
const int botaoAndarL2P6 = 51;

const int botaoEstrutura1 = 38;
const int botaoEstrutura2 = 40;
const int botaoEstrutura3 = 42;
const int botaoEstrutura4 = 44;
const int botaoEstrutura5 = 46;
const int botaoEstrutura6 = 48;

const int botaomanutencao = 53;

// Definição dos pinos para os sensores de posição de cada andar para cada elevador
const int sensorAndar1Elevador1Pin = 8;
const int sensorAndar2Elevador1Pin = 9;
const int sensorAndar3Elevador1Pin = 10;
const int sensorAndar4Elevador1Pin = 11;
const int sensorAndar5Elevador1Pin = 12;
const int sensorAndar6Elevador1Pin = 13;
//########################################
const int sensorAndar1Elevador2Pin = A0;
const int sensorAndar2Elevador2Pin = A1;
const int sensorAndar3Elevador2Pin = A2;
const int sensorAndar4Elevador2Pin = A3;
const int sensorAndar5Elevador2Pin = A4;
const int sensorAndar6Elevador2Pin = A5;

// Definição dos pinos para os motores dos elevadores usando o módulo L9110S
const int elevador1MotorPin1 = 22;
const int elevador1MotorPin2 = 23;
const int elevador2MotorPin1 = 24;
const int elevador2MotorPin2 = 25;

// Variáveis para controlar o estado dos elevadores e o andar atual
int andarAtualElevador1 = 1;
int andarAtualElevador2 = 1;
bool elevador1EmMovimento = false;
bool elevador2EmMovimento = false;

// Definição dos pinos para os LEDs dos displays de 7 segmentos
const int pinledA = 14;
const int pinledB = 15;
const int pinledC = 16;
const int pinledD = 17;
const int pinledE = 18;
const int pinledF = 19;
const int pinledG = 20;
//########################################
const int pinledA2 = 27;
const int pinledB2 = 29;
const int pinledC2 = 31;
const int pinledD2 = 33;
const int pinledE2 = 35;
```

```
const int pinledF2 = 37;
const int pinledG2 = 39;

int elevador = 0;
int andarAtual = 0;
int andarPedido = 0;

int Dif1 = 0;
int Dif2 = 0;
int Dif1_mod = 0;
int Dif2_mod = 0;

int novo_elevador = 0;
bool read1;
bool read2;
int elevador1count = 0;
int elevador2count = 0;

void setup() {
  // Configuração dos pinos dos botões de chamada de cada andar como entrada
  pinMode(botaoAndarL1P1, INPUT_PULLUP);
  pinMode(botaoAndarL1P2, INPUT_PULLUP);
  pinMode(botaoAndarL1P3, INPUT_PULLUP);
  pinMode(botaoAndarL1P4, INPUT_PULLUP);
  pinMode(botaoAndarL1P5, INPUT_PULLUP);
  pinMode(botaoAndarL1P6, INPUT_PULLUP);

  // Configuração dos pinos dos botões de chamada de cada andar como entrada
  pinMode(botaoAndarL2P1, INPUT_PULLUP);
  pinMode(botaoAndarL2P2, INPUT_PULLUP);
  pinMode(botaoAndarL2P3, INPUT_PULLUP);
  pinMode(botaoAndarL2P4, INPUT_PULLUP);
  pinMode(botaoAndarL2P5, INPUT_PULLUP);
  pinMode(botaoAndarL2P6, INPUT_PULLUP);

  // Configuração dos pinos dos sensores de posição de cada andar para cada elevador como entrada
  pinMode(sensorAndar1Elevador1Pin, INPUT_PULLUP);
  pinMode(sensorAndar2Elevador1Pin, INPUT_PULLUP);
  pinMode(sensorAndar3Elevador1Pin, INPUT_PULLUP);
  pinMode(sensorAndar4Elevador1Pin, INPUT_PULLUP);
  pinMode(sensorAndar5Elevador1Pin, INPUT_PULLUP);
  pinMode(sensorAndar6Elevador1Pin, INPUT_PULLUP);
  //########################################
  pinMode(sensorAndar1Elevador2Pin, INPUT_PULLUP);
  pinMode(sensorAndar2Elevador2Pin, INPUT_PULLUP);
  pinMode(sensorAndar3Elevador2Pin, INPUT_PULLUP);
  pinMode(sensorAndar4Elevador2Pin, INPUT_PULLUP);
  pinMode(sensorAndar5Elevador2Pin, INPUT_PULLUP);
  pinMode(sensorAndar6Elevador2Pin, INPUT_PULLUP);

  // Configuração dos pinos dos motores dos elevadores usando o módulo L9110S como saída
  pinMode(elevador1MotorPin1, OUTPUT);
  pinMode(elevador1MotorPin2, OUTPUT);
  pinMode(elevador2MotorPin1, OUTPUT);
  pinMode(elevador2MotorPin2, OUTPUT);

  pinMode(pinledA, OUTPUT);
  pinMode(pinledB, OUTPUT);
  pinMode(pinledC, OUTPUT);
  pinMode(pinledD, OUTPUT);
  pinMode(pinledE, OUTPUT);
  pinMode(pinledF, OUTPUT);
  pinMode(pinledG, OUTPUT);

  pinMode(pinledA2, OUTPUT);
  pinMode(pinledB2, OUTPUT);
```

```
  pinMode(pinledC2, OUTPUT);
  pinMode(pinledD2, OUTPUT);
  pinMode(pinledE2, OUTPUT);
  pinMode(pinledF2, OUTPUT);
  pinMode(pinledG2, OUTPUT);

  pinMode(botaoEstrutura1, INPUT_PULLUP);
  pinMode(botaoEstrutura2, INPUT_PULLUP);
  pinMode(botaoEstrutura3, INPUT_PULLUP);
  pinMode(botaoEstrutura4, INPUT_PULLUP);
  pinMode(botaoEstrutura5, INPUT_PULLUP);
  pinMode(botaoEstrutura6, INPUT_PULLUP);

  pinMode(botaomanutencao, INPUT_PULLUP);

  Serial.begin(9600);
}

void loop() {

  if (digitalRead(botaoEstrutura1) == LOW) {

    andarPedido = 1;
    elevador = elevador_escolhido(andarAtualElevador1, andarAtualElevador2, andarPedido, elevador1count, elevador2count);
    novo_elevador = chamarElevador(andarAtualElevador1, andarAtualElevador2, 1, elevador);

  } else if (digitalRead(botaoEstrutura2) == LOW) {

    andarPedido = 2;
    elevador = elevador_escolhido(andarAtualElevador1, andarAtualElevador2, andarPedido, elevador1count, elevador2count);
    novo_elevador = chamarElevador(andarAtualElevador1, andarAtualElevador2, 2, elevador);

  } else if (digitalRead(botaoEstrutura3) == LOW) {

    andarPedido = 3;
    elevador = elevador_escolhido(andarAtualElevador1, andarAtualElevador2, andarPedido, elevador1count, elevador2count);
    novo_elevador = chamarElevador(andarAtualElevador1, andarAtualElevador2, 3, elevador);

  } else if (digitalRead(botaoEstrutura4) == LOW) {

    andarPedido = 4;
    elevador = elevador_escolhido(andarAtualElevador1, andarAtualElevador2, andarPedido, elevador1count, elevador2count);
    novo_elevador = chamarElevador(andarAtualElevador1, andarAtualElevador2, 4, elevador);

  } else if (digitalRead(botaoEstrutura5) == LOW) {

    andarPedido = 5;
    elevador = elevador_escolhido(andarAtualElevador1, andarAtualElevador2, andarPedido, elevador1count, elevador2count);
    novo_elevador = chamarElevador(andarAtualElevador1, andarAtualElevador2, 5, elevador);

  } else if (digitalRead(botaoEstrutura6) == LOW) {

    andarPedido = 6;
    elevador = elevador_escolhido(andarAtualElevador1, andarAtualElevador2, andarPedido, elevador1count, elevador2count);
    novo_elevador = chamarElevador(andarAtualElevador1, andarAtualElevador2, 6, elevador);

  }
if (digitalRead(botaoAndarL1P1) == LOW && !elevador1EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 1, 1);

  } else if (digitalRead(botaoAndarL1P2) == LOW && !elevador1EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 2, 1);

  } else if (digitalRead(botaoAndarL1P3) == LOW && !elevador1EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 3, 1);

  } else if (digitalRead(botaoAndarL1P4) == LOW && !elevador1EmMovimento) {
```

```
  } else if (digitalRead(botaoAndarL1P4) == LOW && !elevador1EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 4, 1);

  } else if (digitalRead(botaoAndarL1P5) == LOW && !elevador1EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 5, 1);

  } else if (digitalRead(botaoAndarL1P6) == LOW && !elevador1EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 6, 1);

  }
 if (digitalRead(botaoAndarL2P1) == LOW && !elevador2EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 1, 2);

  } else if (digitalRead(botaoAndarL2P2) == LOW && !elevador2EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 2, 2);

  } else if (digitalRead(botaoAndarL2P3) == LOW && !elevador2EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 3, 2);

  } else if (digitalRead(botaoAndarL2P4) == LOW && !elevador2EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 4, 2);

  } else if (digitalRead(botaoAndarL2P5) == LOW && !elevador2EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 5, 2);

  } else if (digitalRead(botaoAndarL2P6) == LOW && !elevador2EmMovimento) {
    chamarElevador(andarAtualElevador1, andarAtualElevador2, 6, 2);
  }


 if (digitalRead(botaomanutencao) == LOW ){
    manutencao();
    andarAtualElevador1 = 1;
    andarAtualElevador2 = 1;
}
display(andarAtualElevador1, andarAtualElevador2);
}

//////////////////////////////////////////////////////.  FUNÇÕES DO PROGRAMA  ////////////////////////////////

int elevador_escolhido (int piso_atual_elev1, int piso_atual_elev2, int piso_pretendido, int count1, int count2){

  Dif1 = (piso_atual_elev1 - piso_pretendido);
  Dif2 = (piso_atual_elev2 - piso_pretendido);
  Dif1_mod = abs(Dif1);
  Dif2_mod = abs(Dif2);


  if(Dif1_mod<Dif2_mod){
    return 1;
  }else if(Dif1_mod>Dif2_mod){
    return 2;
  }else if(Dif1_mod=Dif2_mod){
      if (count1 > count2){
        return 2;
       }else return 1;
    }
}

//////////////////////////////////////////////////////.  PARAR ELEVADOR  ////////////////////////////////////////

void pararElevador(int elevador) {
  if (elevador == 1) {
    digitalWrite(elevador1MotorPin1, HIGH);
```

```
    digitalWrite(elevador1MotorPin2, HIGH);
    elevador1EmMovimento = false;

  } else if (elevador == 2) {
    digitalWrite(elevador2MotorPin1, HIGH);
    digitalWrite(elevador2MotorPin2, HIGH);
    elevador2EmMovimento = false;
  }
}

////////////////////////////////////////////////////.  M O V E R  E L E V A D O R  ////////////////////////////

void moverElevador(int elevador, int direcao, int andarChamado) {

  if (elevador == 1) {
    if (direcao == 1) {
      digitalWrite(elevador1MotorPin1, HIGH);
      digitalWrite(elevador1MotorPin2, LOW);
      digitalWrite(elevador2MotorPin1, HIGH);
      digitalWrite(elevador2MotorPin2, HIGH);
    } else {
      digitalWrite(elevador1MotorPin1, LOW);
      digitalWrite(elevador1MotorPin2, HIGH);
      digitalWrite(elevador2MotorPin1, HIGH);
      digitalWrite(elevador2MotorPin2, HIGH);
    }

    elevador1EmMovimento = true;
    elevador1count ++;

  } else if (elevador == 2) {
    if (direcao == 1) {
      digitalWrite(elevador1MotorPin1, HIGH);
      digitalWrite(elevador1MotorPin2, HIGH);
      digitalWrite(elevador2MotorPin1, HIGH);
      digitalWrite(elevador2MotorPin2, LOW);
    } else {
      digitalWrite(elevador1MotorPin1, HIGH);
      digitalWrite(elevador1MotorPin2, HIGH);
      digitalWrite(elevador2MotorPin1, LOW);
      digitalWrite(elevador2MotorPin2, HIGH);
    }
    elevador2EmMovimento = true;
    elevador2count ++;
  }

}

////////////////////////////////////////////////////.  C H A M A R  E L E V A D O R  ////////////////////

int chamarElevador(int andarAtual_elev1, int andarAtual_elev2, int andarChamado, int elevador) {

int direcao;

switch(elevador){
  case 1:
    direcao = (andarChamado > andarAtual_elev1) ? 1 : -1;
    break;
  case 2:
    direcao = (andarChamado > andarAtual_elev2) ? 1 : -1;
}

  // Move o elevador até o andar chamado
  if (elevador == 1) {

    while (andarAtualElevador1 != andarChamado) {
```

```
331         moverElevador(1, direcao, andarChamado);
332         // Move o elevador 1 na direção correta
333         // Atualiza o andar atual com base na direção do movimento
334
335
336         // Verifica se algum sensor de posição de andar foi acionado
337         if (digitalRead(sensorAndar1Elevador1Pin) == LOW && andarChamado == 1) {
338           andarAtualElevador1 = 1;
339           pararElevador(1);
340           break;
341
342         } else if (digitalRead(sensorAndar2Elevador1Pin) == LOW && andarChamado == 2) {
343           andarAtualElevador1 = 2;
344           pararElevador(1);
345           break;
346
347         } else if (digitalRead(sensorAndar3Elevador1Pin) == LOW && andarChamado == 3) {
348           andarAtualElevador1 = 3;
349           pararElevador(1);
350           break;
351
352         } else if (digitalRead(sensorAndar4Elevador1Pin) == LOW && andarChamado == 4) {
353           andarAtualElevador1 = 4;
354           pararElevador(1);
355           break;
356
357         } else if (digitalRead(sensorAndar5Elevador1Pin) == LOW && andarChamado == 5) {
358           andarAtualElevador1 = 5;
359           pararElevador(1);
360           break;
361
362         } else if (digitalRead(sensorAndar6Elevador1Pin) == LOW && andarChamado == 6) {
363           andarAtualElevador1 = 6;
364           pararElevador(1);
365           break;
366         }
367       }
368     } else if (elevador == 2) {
369       while (andarAtual_elev2 != andarChamado) {
370         moverElevador(2, direcao, andarChamado);
371         // Move o elevador 2 na direção correta
372         // Atualiza o andar atual com base na direção do movimento
373
374         // Verifica se algum sensor de posição de andar foi acionado
375         if (digitalRead(sensorAndar1Elevador2Pin) == LOW && andarChamado == 1) {
376           andarAtualElevador2 = 1;
377           pararElevador(2);
378           break;
379         } else if (digitalRead(sensorAndar2Elevador2Pin) == LOW && andarChamado == 2) {
380           andarAtualElevador2 = 2;
381           pararElevador(2);
382           break;
383         } else if (digitalRead(sensorAndar3Elevador2Pin) == LOW && andarChamado == 3) {
384           andarAtualElevador2 = 3;
385           pararElevador(2);
386           break;
387         } else if (digitalRead(sensorAndar4Elevador2Pin) == LOW && andarChamado == 4) {
388           andarAtualElevador2 = 4;
389           pararElevador(2);
390           break;
391         } else if (digitalRead(sensorAndar5Elevador2Pin) == LOW && andarChamado == 5) {
392           andarAtualElevador2 = 5;
393           pararElevador(2);
394           break;
395         } else if (digitalRead(sensorAndar6Elevador2Pin) == LOW && andarChamado == 6) {
396           andarAtualElevador2 = 6;
```

```
397             pararElevador(2);
398             break;
399           }
400         }
401
402       }
403     }
404
405     ////////////////////////////////////////////////////.  D I S P L A Y ' S  ////////////////////////
406
407     void display(int pisosA, int pisosB){
408
409       switch(pisosB){
410         case 1:
411           digitalWrite(pinledB2, HIGH);
412           digitalWrite(pinledC2, HIGH);
413           digitalWrite(pinledA2, LOW);
414           digitalWrite(pinledD2, LOW);
415           digitalWrite(pinledE2, LOW);
416           digitalWrite(pinledF2, LOW);
417           digitalWrite(pinledG2, LOW);
418         break;
419         case 2:
420           digitalWrite(pinledA2, HIGH);
421           digitalWrite(pinledB2, HIGH);
422           digitalWrite(pinledG2, HIGH);
423           digitalWrite(pinledE2, HIGH);
424           digitalWrite(pinledD2, HIGH);
425           digitalWrite(pinledC2, LOW);
426           digitalWrite(pinledF2, LOW);
427         break;
428         case 3:
429           digitalWrite(pinledA2, HIGH);
430           digitalWrite(pinledB2, HIGH);
431           digitalWrite(pinledG2, HIGH);
432           digitalWrite(pinledC2, HIGH);
433           digitalWrite(pinledD2, HIGH);
434           digitalWrite(pinledF2, LOW);
435           digitalWrite(pinledE2, LOW);
436         break;
437         case 4:
438           digitalWrite(pinledF2, HIGH);
439           digitalWrite(pinledG2, HIGH);
440           digitalWrite(pinledB2, HIGH);
441           digitalWrite(pinledC2, HIGH);
442           digitalWrite(pinledA2, LOW);
443           digitalWrite(pinledE2, LOW);
444           digitalWrite(pinledD2, LOW);
445         break;
446         case 5:
447           digitalWrite(pinledA2, HIGH);
448           digitalWrite(pinledF2, HIGH);
449           digitalWrite(pinledG2, HIGH);
450           digitalWrite(pinledC2, HIGH);
451           digitalWrite(pinledD2, HIGH);
452           digitalWrite(pinledB2, LOW);
453           digitalWrite(pinledE2, LOW);
454         break;
455         case 6:
456           digitalWrite(pinledA2, HIGH);
457           digitalWrite(pinledF2, HIGH);
458           digitalWrite(pinledG2, HIGH);
459           digitalWrite(pinledE2, HIGH);
460           digitalWrite(pinledD2, HIGH);
461           digitalWrite(pinledC2, HIGH);
462           digitalWrite(pinledB2, LOW);
```

```
463       break;
464     }
465
466     switch(pisosA){
467       case 1:
468         digitalWrite(pinledB, HIGH);
469         digitalWrite(pinledC, HIGH);
470         digitalWrite(pinledA, LOW);
471         digitalWrite(pinledD, LOW);
472         digitalWrite(pinledE, LOW);
473         digitalWrite(pinledF, LOW);
474         digitalWrite(pinledG, LOW);
475       break;
476       case 2:
477         digitalWrite(pinledA, HIGH);
478         digitalWrite(pinledB, HIGH);
479         digitalWrite(pinledG, HIGH);
480         digitalWrite(pinledE, HIGH);
481         digitalWrite(pinledD, HIGH);
482         digitalWrite(pinledC, LOW);
483         digitalWrite(pinledF, LOW);
484       break;
485       case 3:
486         digitalWrite(pinledA, HIGH);
487         digitalWrite(pinledB, HIGH);
488         digitalWrite(pinledG, HIGH);
489         digitalWrite(pinledC, HIGH);
490         digitalWrite(pinledD, HIGH);
491         digitalWrite(pinledF, LOW);
492         digitalWrite(pinledE, LOW);
493       break;
494       case 4:
495         digitalWrite(pinledF, HIGH);
```

```
      digitalWrite(pinledG, HIGH);
      digitalWrite(pinledB, HIGH);
      digitalWrite(pinledC, HIGH);
      digitalWrite(pinledA, LOW);
      digitalWrite(pinledE, LOW);
      digitalWrite(pinledD, LOW);
    break;
    case 5:
      digitalWrite(pinledA, HIGH);
      digitalWrite(pinledF, HIGH);
      digitalWrite(pinledG, HIGH);
      digitalWrite(pinledC, HIGH);
      digitalWrite(pinledD, HIGH);
      digitalWrite(pinledB, LOW);
      digitalWrite(pinledE, LOW);
    break;
    case 6:
      digitalWrite(pinledA, HIGH);
      digitalWrite(pinledF, HIGH);
      digitalWrite(pinledG, HIGH);
      digitalWrite(pinledE, HIGH);
      digitalWrite(pinledD, HIGH);
      digitalWrite(pinledC, HIGH);
      digitalWrite(pinledB, LOW);
    break;
  }
}

void manutencao(){

    while(digitalRead(sensorAndar1Elevador1Pin) == HIGH){
      moverElevador(1,-1,1);
    }
    pararElevador(1);

    delay (1000);

    while(digitalRead(sensorAndar1Elevador2Pin) == HIGH){
      moverElevador(2,-1,1);
    }
    pararElevador(2);
}
```

# Referências

Joaquim Delgado, Apontamentos da Unidade Curricular Instrumentação Industrial, Viseu: DEE-ESTGVIPV.

Rui Alves, Apontamentos da Unidade Curricular Programação de Computadores, Viseu: DEE-ESTGVIPV

Rui Alves, Apontamentos da Unidade Curricular Programação Avançada, Viseu: DEE-ESTGVIPV

António Ferreira, Apontamentos da Unidade Curricular Microssistemas, Viseu: DEE-ESTGVIPV

José Eduardo Paiva Apontamentos da Unidade Curricular Eletrónica de potência, Viseu: DEE-ESTGVIPV

https://www.arduino.cc/en/software

http://eletronicaparaartistas.com.br/experimento-35-extra-usando-displays-de-7-segmentos/

https://www.laskakit.cz/user/related_files/l9110_2_channel_motor_driver.pdf