ROBÓTICA

AULA 27

Primeiros Passos Módulo 3

Elevador Elétrico II

Diretoria de Tecnologia e Inovação

PARANÁ
GOVERNO DO ESTADO
SECRETARIA DA EDUCAÇÃO

GOVERNADOR DO ESTADO DO PARANÁ

Carlos Massa Ratinho Júnior

SECRETÁRIO DE ESTADO DA EDUCAÇÃO

Roni Miranda Vieira

DIRETOR DE TECNOLOGIA E INOVAÇÃO

Claudio Aparecido de Oliveira

COORDENADOR DE TECNOLOGIAS EDUCACIONAIS

Marcelo Gasparin

Produção de Conteúdo

Adilson Carlos Batista

Validação de Conteúdo

Cleiton Rosa
Darice Alessandra Deckmann Zanardini
Viviane Dziubate Pittner

Revisão Textual

Kellen Pricila dos Santos Cochinski

Projeto Gráfico e Diagramação

Edna do Rocio Becker

2024

SUMÁRIO

AULA 27

Elevador elétrico II

Introdução

Na aula 26, você trabalhou com a primeira parte desta aula e, além de aprender um pouco sobre Werner Von Siemens o criador do elevador elétrico, você também prototipou a estrutura, a cabine e algumas peças como a polia do elevador. Agora, vamos para a segunda parte – a montagem da parte eletrônica e a programação do protótipo do elevador elétrico. Mãos à obra!

Objetivos desta aula

- Montar a parte eletrônica do elevador;
- Programar o protótipo do elevador elétrico.

Lista de materiais

- Microcontrolador Arduino;
- Protoboard;
- 15 Jumpers machos;
- 5 Jumpers fêmeas;
- 1 Resistor de 1 kΩ;
- 1 Bateria de 9V;
- 1 Ponte H;
- 1 LED;
- 3 Botões;
- 1 Motor DC;
- Matriz de LED;
- Estrutura montada anteriormente.

AULA 27

Elevador elétrico II

ROTEIRO DA AULA

1. Contextualização

Na aula 26, destacamos que o elevador elétrico foi inventado por Werner Von Siemens em 1880 e rapidamente se tornou um meio de transporte vertical essencial, e que antes dos elevadores elétricos, os modelos eram rudimentares, como plataformas de madeira içadas por cordas. Vimos também que os elevadores modernos são muito mais seguros e confortáveis do que seus predecessores. Eles incorporam sensores para impedir o fechamento das portas se houver obstrução, sistemas de frenagem automática e alarmes para comunicar falhas.

Em resumo, o elevador se transformou de uma máquina simples para uma peça fundamental na infraestrutura urbana, impactando a acessibilidade, a mobilidade e a própria construção de arranha-céus.

Elevador elétrico II

Saiba mais!

O vídeo explica o funcionamento de elevadores, destacando a importância do contrapeso e dos sistemas de freio para segurança. Mostra como a energia elétrica e os componentes mecânicos interagem para mover a cabine, além de comparar tecnologias antigas e modernas. A visita a uma fabricante em São Paulo oferece uma visão detalhada dos mecanismos ocultos que garantem a operação segura dos elevadores.

YouTube

Como funciona um elevador#boravê_manual do mundo

AULA 27

Elevador elétrico II

O elevador é composto por diversos elementos que trabalham em conjunto para garantir o transporte seguro e eficiente de pessoas e cargas:

Casa de máquinas: abriga os equipamentos e componentes responsáveis pela movimentação e operação do elevador. Geralmente, ela está localizada na parte superior do edifício, mas em alguns casos pode estar em andares inferiores. Por razões de segurança, o acesso ao local é restrito a técnicos de manutenção, ao síndico e ao zelador. Nos elevadores mais modernos, a casa de máquinas foi eliminada, pois as novas tecnologias permitiram compactar a maioria dos componentes, que agora são instalados no topo da caixa de corrida.

Máquina de tração: é o principal componente que garante o movimento do elevador a cada comando de subida ou descida. Ela sustenta o peso da cabina — incluindo passageiros e carga —, bem como o contrapeso, cabos de aço e correntes de compensação.

Limitador de velocidade: esse dispositivo é essencial para a segurança dos passageiros, pois monitora a velocidade do elevador. Se o limite de velocidade for excedido, o limitador aciona automaticamente o freio de segurança da cabina, imobilizando-a.

Passadiço (caixa de corrida): é o espaço físico que contém as paredes verticais, o fundo do poço e o teto da instalação do elevador. É uma área técnica destinada à montagem de componentes como guias, cabos, cabina e contrapeso. O acesso a essa área é restrito a profissionais responsáveis pela manutenção.

Cabina: é o componente que transporta passageiros e cargas. É fundamental que sua capacidade máxima, tanto em número de passageiros quanto em peso, esteja claramente sinalizada e visível, evitando sobrecargas que possam comprometer a operação do elevador.

Pavimento: é o local de embarque e desembarque de passageiros dentro da estrutura do edifício. Nele ficam instalados a porta externa (porta de andar), fecho eletromecânico, botoeira de pavimento, indicador de posição digital, botão(ões) de chamado identificados por braile, placa de sinalização.

Poço: é a parte inferior da caixa de corrida, onde são instalados dispositivos que garantem o funcionamento e a segurança do elevador. O elevador utiliza da energia elétrica para puxar o carro através de um motor elétrico, conectado ao elevador por um sistema de cabos de aço e polias, possuindo um contrapeso aplicando tração na direção contrária do cabo ao elevador, buscando diminuir o esforço do motor e a frenagem necessária para seu funcionamento.

Contrapeso: é parte de um sistema de contrabalanço que equilibra o peso da cabina, reduzindo o esforço necessário da máquina de tração e economizando energia. Para que o contrapeso funcione adequadamente, seu peso deve equivaler a 40% a 50% do peso total da cabina e sua capacidade de carga máxima.

Elevador elétrico II

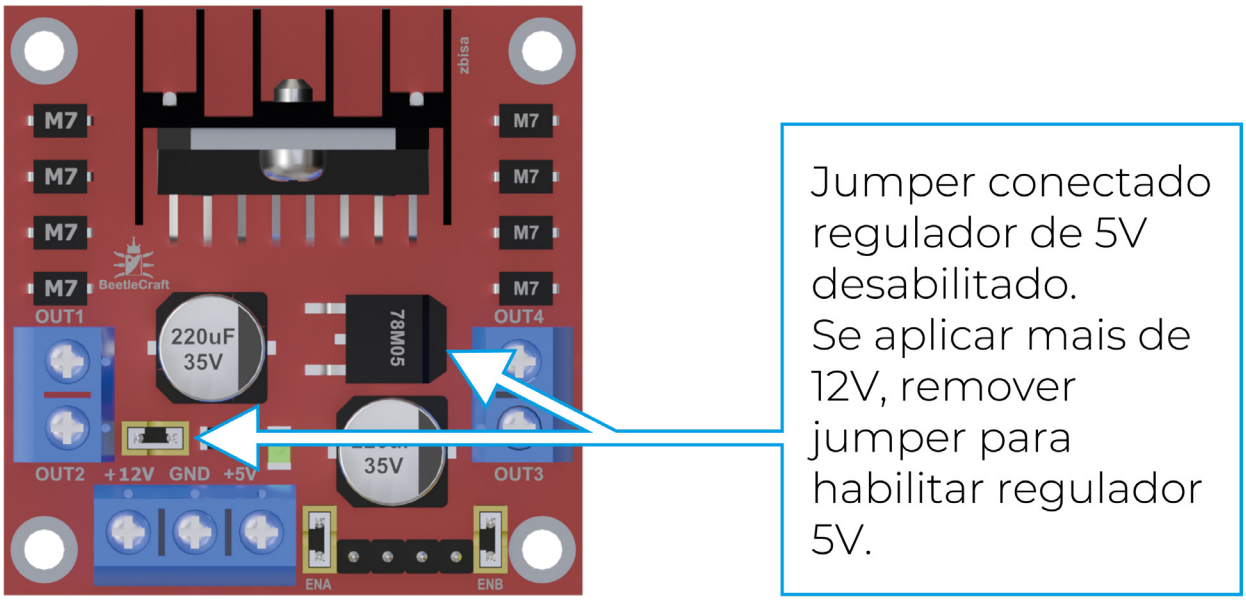
Agora, vamos para a montagem do nosso protótipo! Entretanto, como estamos inserindo um componente novo, vamos explicá-lo aqui.

Você sabe o que é uma ponte H. e qual a função desse componente?

Imagine um interruptor muito especial que permite controlar o fluxo de corrente em duas direções. Essa é a função básica de uma ponte H. Ela é composta por quatro transistores conectados para criar dois pares complementares, conforme figura. Ao acionar os transistores de forma adequada, você pode inverter a polaridade da tensão aplicada a uma carga, como um motor DC. Permitindo controlar o sentido de rotação e a velocidade de motores DC. Ela também pode ser utilizada para controlar relés e solenoides para ativar e desativar cargas indutivas de forma eficiente, bem como ajustar brilho de LEDs ou criar efeito de iluminação.

Existem diversos modelos de ponte H, no nosso caso, iremos utilizar a modelo L298N, conforme Figura 1.

Figura 1 - Ponte H L298N



Fonte: Roberto Carlos Rodrigues, 2024.

Ela é composta por quatro transistores conectados em uma configuração específica, formando um "H" quando visualizados em um diagrama esquemático.

AULA 27

Elevador elétrico II

A ponte H L298N é um circuito integrado muito popular utilizado para controlar motores DC e motores de passo. Ela permite inverter o sentido de rotação do motor e controlar a velocidade, sendo uma peça fundamental em diversos projetos de robótica e automação.

Características da L298N:

1. **Controle de dois motores:** a L298N permite controlar dois motores DC independentemente.
2. **Alta corrente:** cada saída da L298N pode fornecer até 2A de corrente, o que a torna adequada para controlar motores de médio porte.
3. **Tensão de operação:** funciona com uma ampla faixa de tensão, geralmente entre 5V e 35V.
4. **Proteção térmica:** possui proteção interna contra sobreaquecimento.
5. **Facilidade de uso:** é relativamente fácil de usar e integrar em projetos.

Como funciona a L298N:

Ao aplicar sinais lógicos (0 ou 1) nos pinos de entrada da L298N, você determina o sentido de rotação e a velocidade do motor. A modulação por largura de pulso (PWM) é utilizada para controlar a velocidade do motor, variando a largura dos pulsos enviados aos pinos de entrada.

Elevador elétrico II

Por que precisamos utilizar uma ponte H com o Arduino?

A ponte H L298N é um componente essencial em muitos projetos com Arduino, especialmente quando envolvem o controle de motores DC. Mas por que ela é tão importante? Vamos entender as principais razões:

1. Controle bidirecional:

- **Inversão de polaridade:** a ponte H permite inverter a polaridade da tensão aplicada ao motor, permitindo que ele gire em ambos os sentidos. Isso é crucial para diversas aplicações, como robôs móveis que precisam mudar de direção.
- **Precisão:** o controle preciso da direção é fundamental para realizar movimentos precisos e coordenados.

2. Controle de velocidade:

- **Modulação por Largura de Pulso (PWM):** combinada com a ponte H, a técnica de PWM permite variar a velocidade do motor de forma suave e precisa. Ao ajustar a largura dos pulsos, você controla a quantidade de energia entregue ao motor.
- **Flexibilidade:** essa flexibilidade é essencial para diversas aplicações, como controlar a velocidade de um braço robótico ou ajustar a velocidade de um carro autônomo.

3. Proteção:

- **Isolamento:** a ponte H atua como um isolante entre a fonte de alimentação e o microcontrolador (Arduino), protegendo-o de possíveis sobrecargas e curtos-circuitos.
- **Dissipação de calor:** muitos modelos de ponte H, como a L298N, possuem proteção térmica interna, evitando danos ao componente em caso de sobreaquecimento.

4. Alta corrente:

- **Capacidade de acionamento:** a L298N pode fornecer correntes relativamente altas para os motores, permitindo o controle de motores mais potentes.
- **Versatilidade:** isso amplia o leque de aplicações, desde pequenos projetos até sistemas mais complexos.

5. Facilidade de uso:

- **Integração com Arduino:** a L298N é fácil de integrar com o Arduino, graças à sua interface simples e à disponibilidade de bibliotecas.
- **Configuração:** a configuração dos pinos de controle é direta, facilitando a implementação em seus projetos.

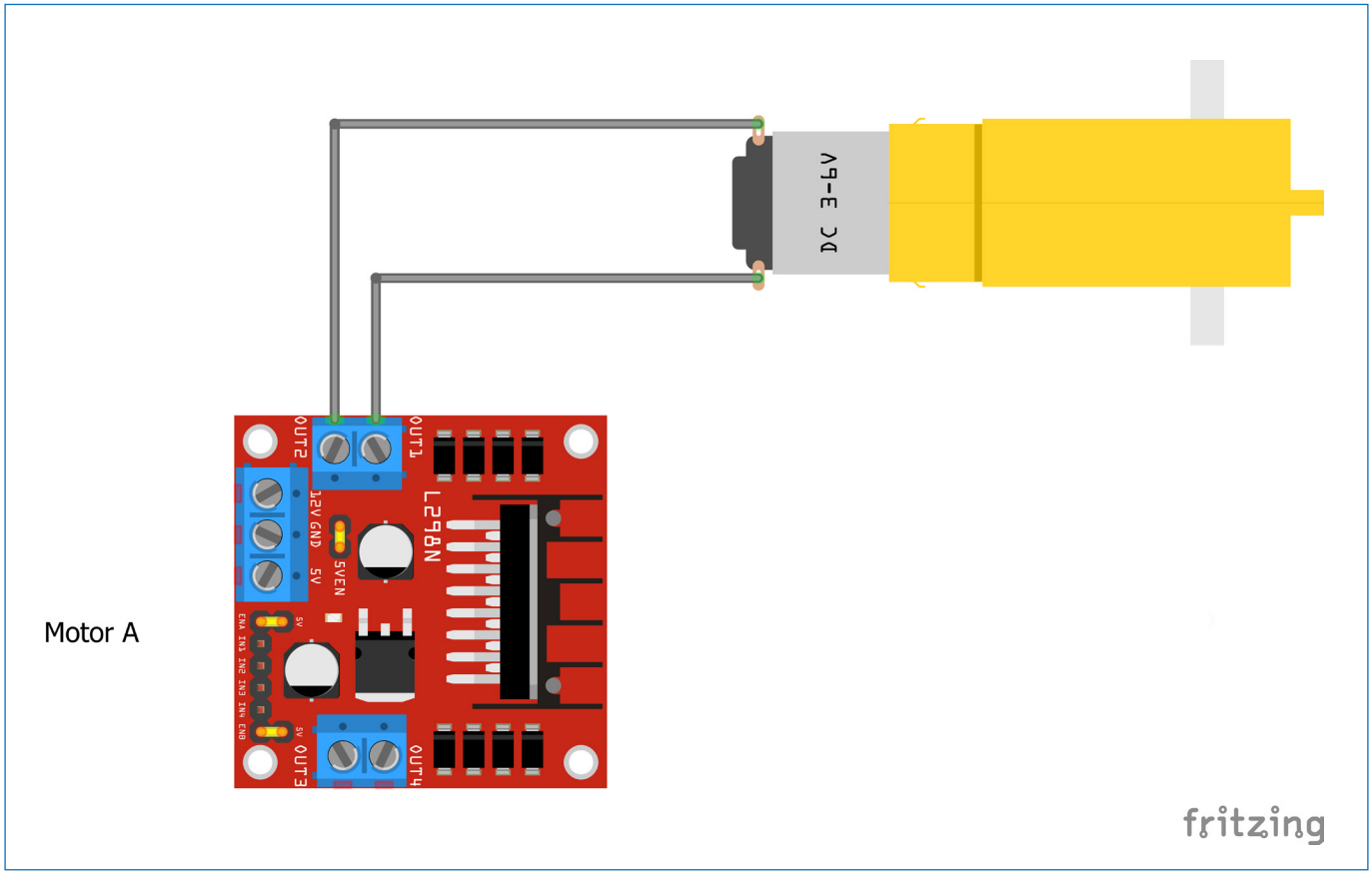
Agora, vamos para a montagem!

2. Montagem:

Conectando o motor DC à ponte H:

- **Identifique os terminais do motor:** geralmente, os motores DC possuem dois terminais: positivo (+) e negativo (-).
- **Prepare os fios:** se os fios do motor não estiverem soldados, solde-os aos terminais correspondentes. Certifique-se de usar fios com a bitola adequada para suportar a corrente do motor.
- **Conecte à ponte H:**
  a. **OUT1 e OUT2:** conecte os fios do motor aos terminais OUT1 e OUT2 da ponte H. A polaridade não é crítica nesse ponto, pois a direção do motor será controlada pelos sinais digitais.

Figura 2 – Conexão do motor à ponte H

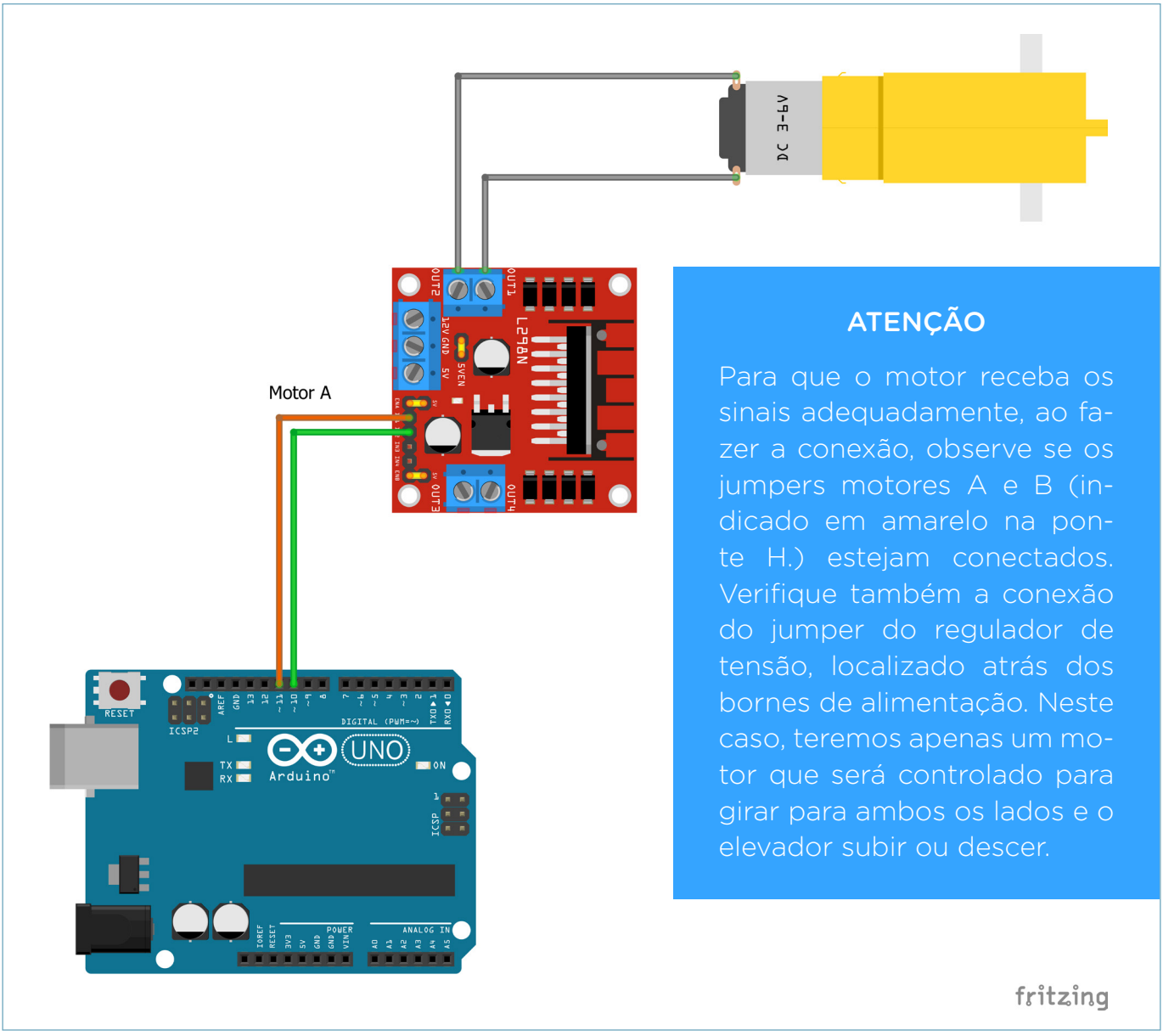


Fonte: SEED/DTI/CTE.

Elevador elétrico II

Utilize um jumper macho-macho para conectar o pino IN1 da ponte H ao pino digital 11 do Arduino. Este pino (IN1) controla a direção de rotação do motor DC. **É fundamental observar a polaridade dos jumpers e a configuração dos outros pinos da ponte H**, pois isso determinará o sentido de giro do motor para subir e descer o elevador. Se o motor girar no sentido contrário ao desejado, basta inverter as conexões dos pinos IN1 e IN2 da ponte H. Utilize um jumper macho-macho para conectar o pino IN2 da ponte H ao pino digital 10 do Arduino.

Figura 3 - Conexão do controle para a ponte H



ATENÇÃO

Para que o motor receba os sinais adequadamente, ao fazer a conexão, observe se os jumpers motores A e B (indicado em amarelo na ponte H.) estejam conectados. Verifique também a conexão do jumper do regulador de tensão, localizado atrás dos bornes de alimentação. Neste caso, teremos apenas um motor que será controlado para girar para ambos os lados e o elevador subir ou descer.

Elevador elétrico II

Conecte o terminal GND da ponte H diretamente ao terminal GND do Arduino. Conecte o 12V da ponte H a porta VIN do Arduino.

ATENÇÃO

Antes de conectar a ponte H à porta VIN do Arduino, é crucial considerar a corrente, a tensão elétrica e o tempo de funcionamento do motor utilizado. **A porta VIN possui limitações de corrente e tensão**, e o uso inadequado pode danificar o Arduino. Esta aula foi desenvolvida com a utilização dos componentes disponibilizados no kit. **Recomendamos, caso use um motor mais potente, diferente dos motores do kit de Robótica (motor DC 3-6V, corrente sem carga ≤200mA (6V) e ≤150mA (3V)), e fonte (fonte DC chaveada 9V 1A plug P4), utilizar uma fonte externa exclusiva para a ponte H**, para garantir um funcionamento seguro e eficiente do seu projeto e a não utilização da porta VIN.

Para isso, basta conectar o positivo da fonte externa ao VCC 12 V e o negativo ao GND da ponte H juntamente com o GND do Arduino, as demais conexões permanecem as mesmas. A fonte continuará sendo usada para alimentar o Arduino.

AULA 27

Elevador elétrico II

O uso da porta VIN do Arduino em conjunto com uma ponte H para alimentar um motor DC pode ser uma opção, porém exige cuidados especiais:

1. **Simplificação do circuito:** ao conectar o terminal de 12V da ponte H à porta VIN do Arduino, você aproveita uma única fonte de alimentação para ambos os componentes (ponte H e Arduino), simplificando o circuito e evitando a necessidade de fontes separadas para o Arduino e o motor, desde que tome os cuidados necessários, conforme comentados acima.
2. **Fornecimento de energia:** a porta VIN permite que o Arduino seja alimentado por uma fonte externa, mas é **importante verificar as especificações do seu modelo de Arduino** quanto à tensão máxima suportada na porta VIN. **Exceder essa tensão pode danificar o regulador de tensão interno.**
3. **Corrente:** além da tensão, é crucial considerar a **corrente máxima que a porta VIN pode fornecer**. Se o motor DC exigir uma corrente elevada, a porta VIN pode não ser capaz de suprir essa demanda adequadamente, **afetando o desempenho do Arduino e da ponte H.**
4. **Dissipação de potência:** a conexão direta da fonte de 12V à porta VIN pode gerar **calor excessivo no regulador de tensão do Arduino. É recomendado utilizar um dissipador** de calor para evitar danos ao componente.
5. **Estabilidade:** embora a conexão dos terminais GND garanta um ponto de referência comum, **flutuações na tensão da fonte externa podem afetar a estabilidade do circuito. É aconselhável utilizar um regulador de tensão externo** para garantir uma alimentação mais limpa e estável para o Arduino.

Elevador elétrico II

Quando utilizar uma fonte externa separada?

- **Motores de alta corrente:** motores que exigem uma corrente superior à capacidade da porta VIN.
- **Funcionamento contínuo:** aplicações que exigem que o motor funcione por longos períodos.
- **Tensões elevadas:** quando a tensão necessária para o motor ultrapassa a tensão máxima suportada pela porta VIN.

Vantagens da fonte externa separada:

- **Maior segurança:** protege o Arduino e outros componentes.
- **Maior eficiência:** permite otimizar a alimentação para cada componente.
- **Maior flexibilidade:** permite o uso de diferentes tipos de motores e fontes de alimentação.

Em resumo:

Embora seja possível conectar a ponte H à porta VIN do Arduino, **a utilização de uma fonte externa separada é a prática mais recomendada** para garantir a segurança, a estabilidade e a durabilidade do seu projeto.

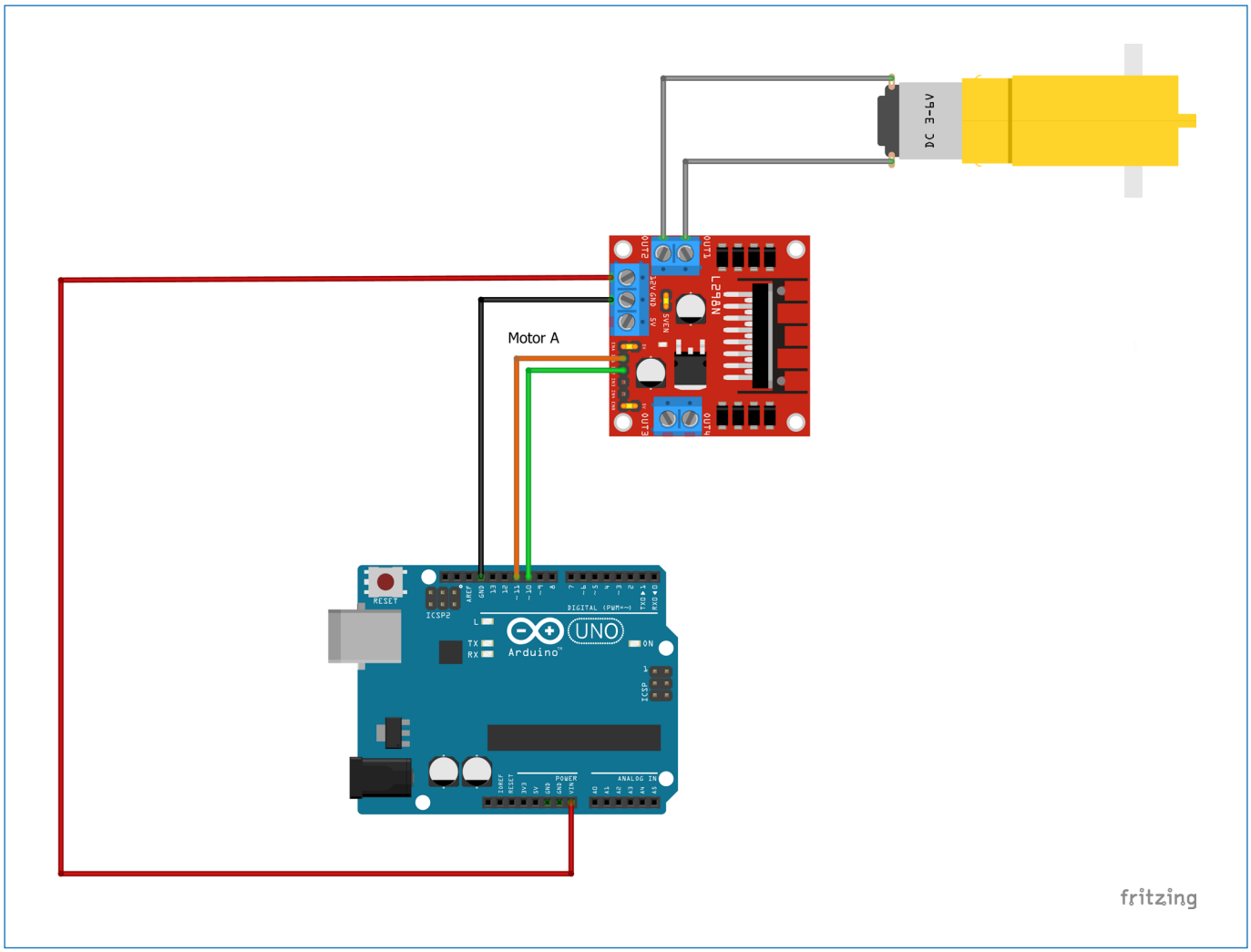
Ao escolher a fonte externa, considere:

- **Tensão:** deve ser igual ou superior à tensão nominal do motor.
- **Corrente:** deve ser capaz de fornecer a corrente máxima exigida pelo motor.
- **Polaridade:** verifique a polaridade da fonte para evitar danos aos componentes.

Lembre-se! A segurança deve ser sempre a prioridade em seus projetos eletrônicos. Ao seguir essas recomendações, você garantirá um funcionamento mais confiável e evitará problemas futuros.

Elevador elétrico II

Figura 4 - Conexão da ponte ao Arduino

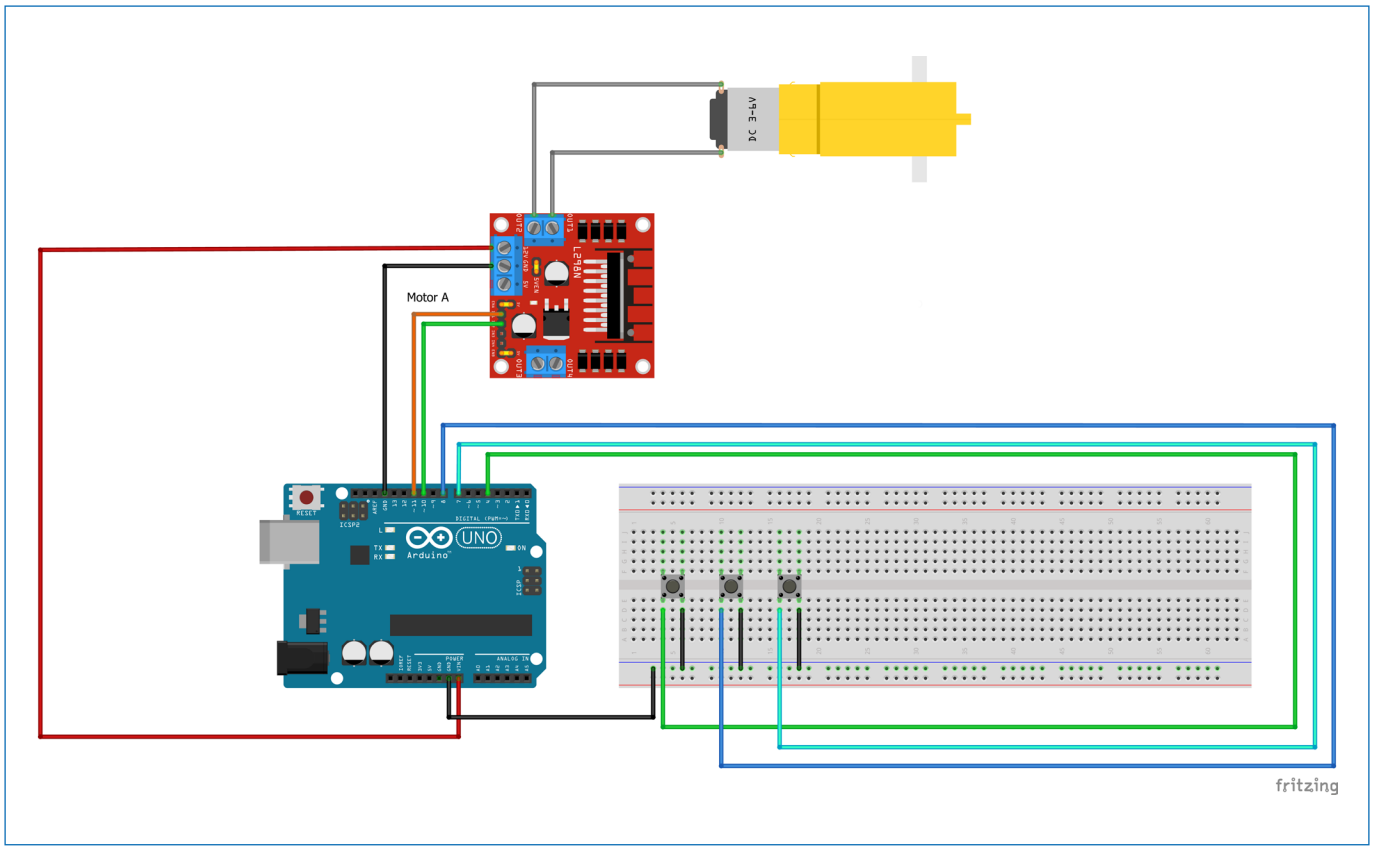


No eixo do motor DC, fixe a roldana conforme descrito na aula 26. Em seguida, amarre uma das extremidades do barbante na roldana. Certifique-se de que a roldana esteja bem fixada para evitar que o barbante escorregue.

Conecte cada botão a uma saída digital do Arduino, configurando-as como entradas com resistor de pull-down. Cada botão representará um andar: o térreo (conectado à saída digital 4), o primeiro andar (à saída digital 7) e o segundo andar (à saída digital 8), conforme a Figura 4. A configuração de pull-down garante que os pinos digitais estejam em nível lógico baixo quando os botões não forem pressionados.

Elevador elétrico II

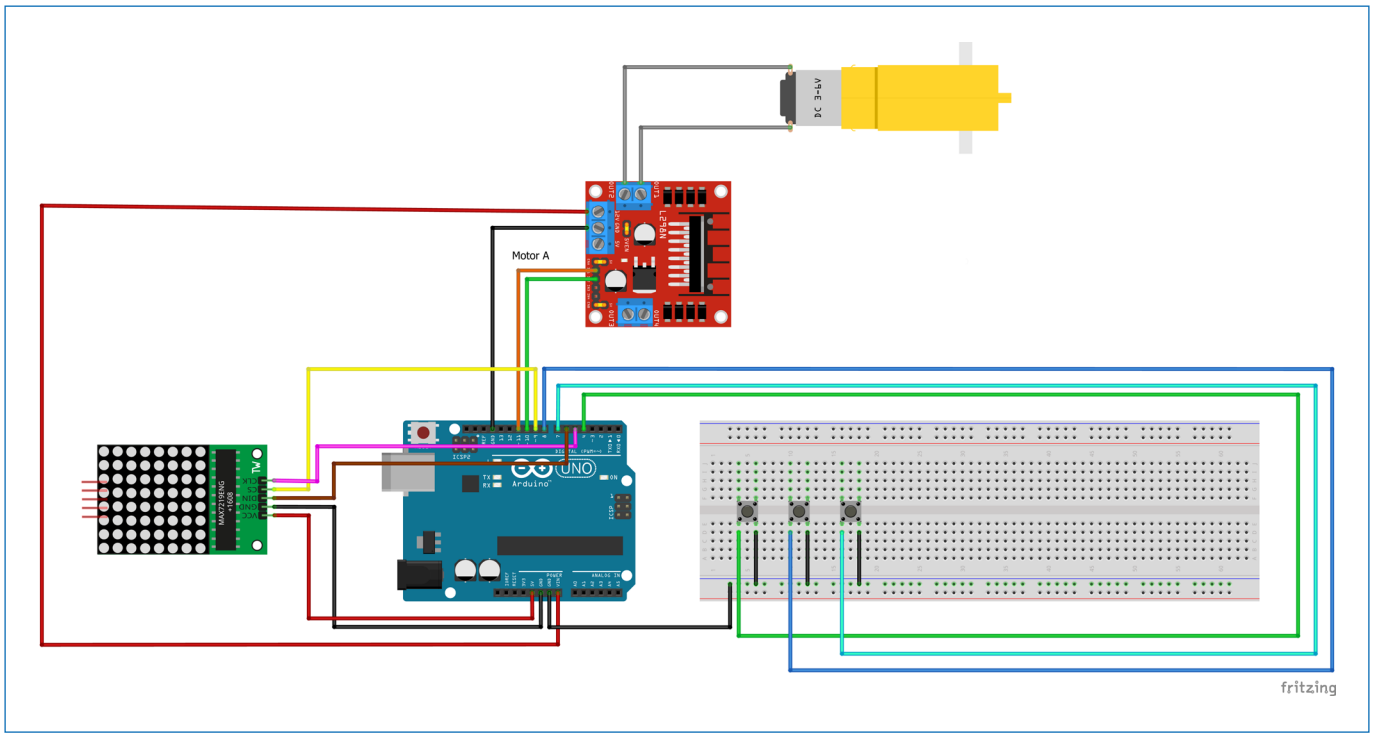
Figura 5 - Adição de botões à protoboard



Elevador elétrico II

Conecte o LED à porta digital 4, conforme a Figura 6. Utilize um resistor limitador de corrente em série com o LED para evitar danos. Esse LED servirá como indicador visual do estado do elevador: quando o elevador estiver em movimento (subindo ou descendo), o LED acenderá. Quando o elevador estiver parado, o LED apagará.

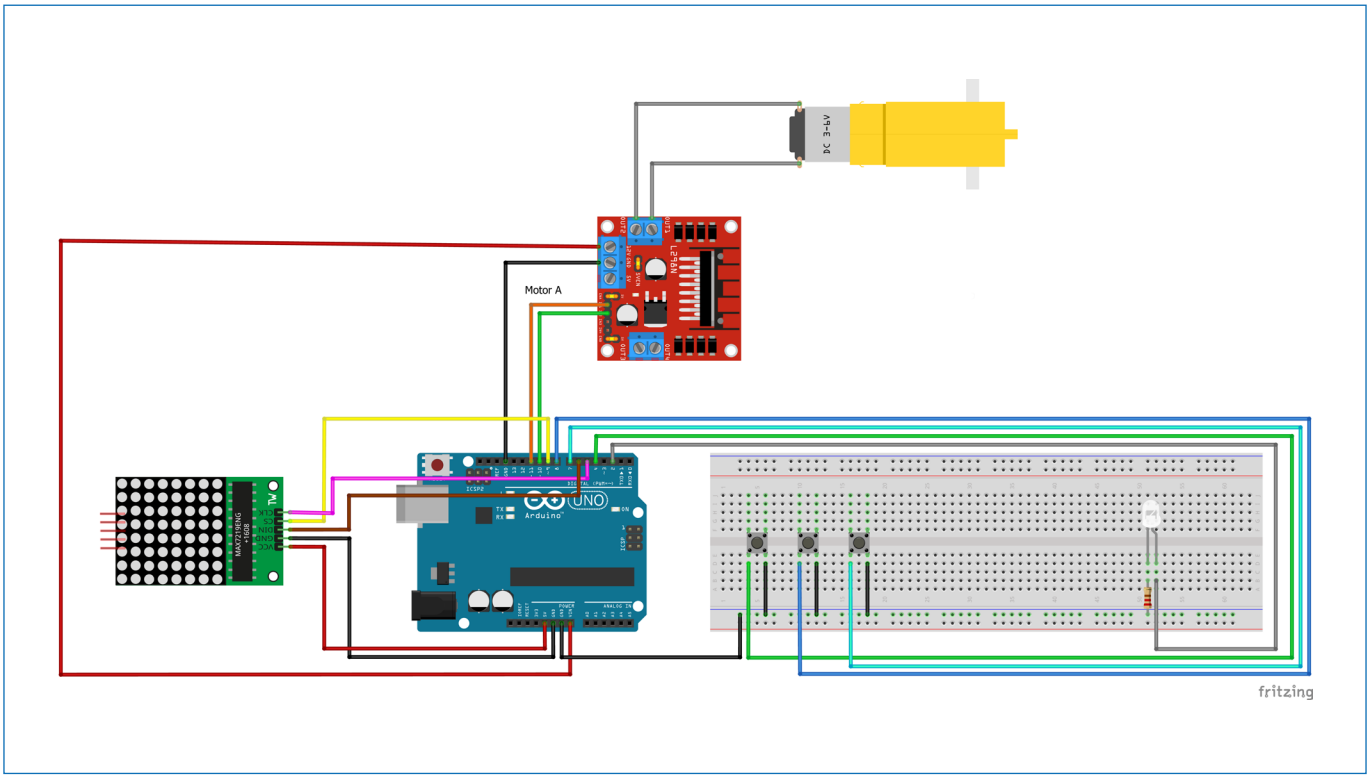
Figura 6 - Adição do LED de sinalização do elevador



Elevador elétrico II

Para visualizar de forma mais intuitiva a posição e o movimento do elevador, adicionaremos uma matriz de LEDs. Conecte o GND e o VCC da matriz diretamente aos pinos de alimentação do Arduino. Em seguida, conecte os pinos de controle da matriz (CS, CLK e DIN) aos pinos digitais 5, 9 e 6 do Arduino, respectivamente. Essa configuração permitirá que a matriz exiba diferentes padrões de iluminação - ou acionamento - de seus LEDs, os quais formarão símbolos e números para indicar o andar atual e a direção do movimento do elevador.

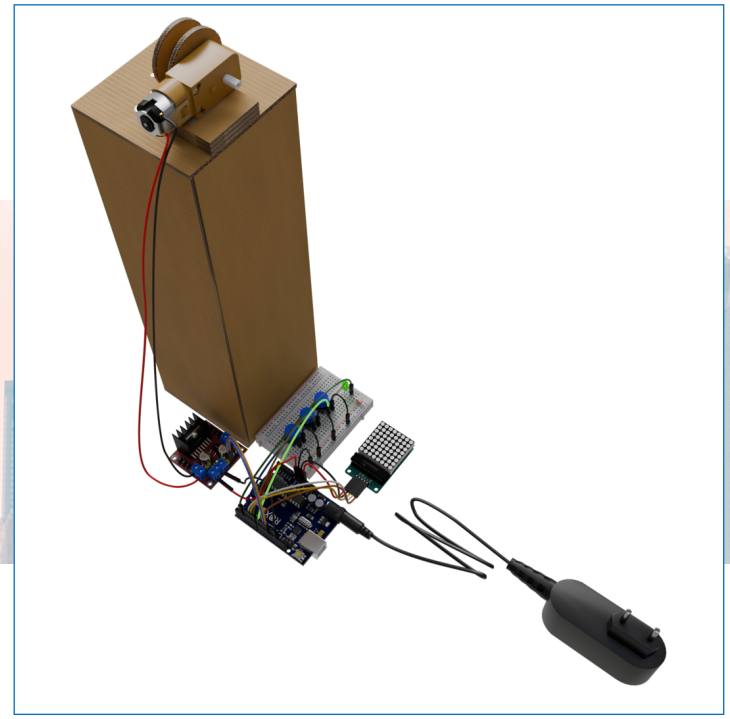
Figura 7 - Adição da matriz de LEDs



Para finalizar a montagem do protótipo do elevador, encaixe cuidadosamente a caixa, construída na aula 26, na base. Verifique se a caixa está alinhada e nivelada em relação à base. Em seguida, conecte todos os componentes, como o motor, a polia e os cabos, conforme indicado nas Figuras 8 e 9. Preste atenção à polaridade das conexões elétricas e utilize conectores adequados para garantir uma conexão segura. Certifique-se de que todas as peças estejam firmemente fixadas e alinhadas para evitar problemas durante a operação do elevador.

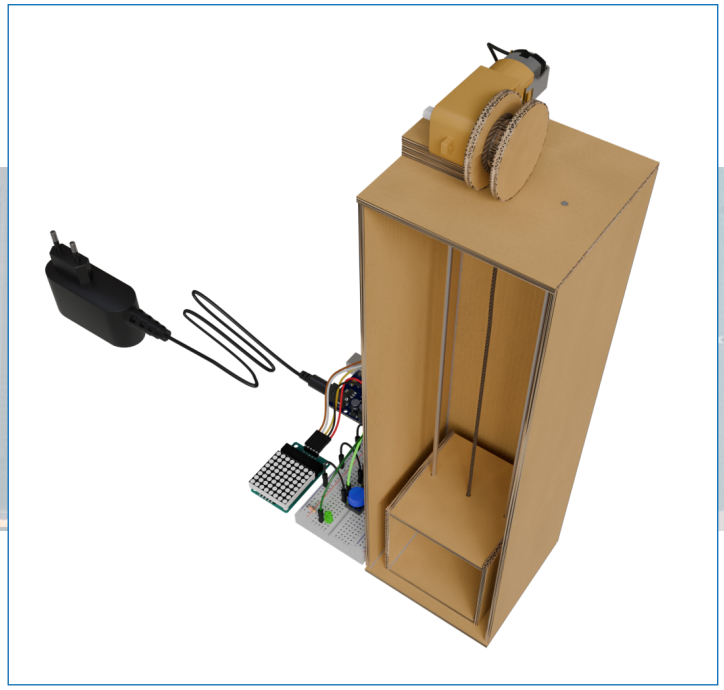
Elevador elétrico II

Figura 8 - Protótipo montado – parte de trás



Fonte: Roberto Carlos Rodrigues, 2024.

Figura 9 - Protótipo montado - frente



Fonte: Roberto Carlos Rodrigues, 2024.

AULA 27

Elevador elétrico II

Este projeto tem como objetivo construir um modelo funcional de elevador utilizando a plataforma Arduino. Através da montagem e programação, exploraremos conceitos de eletrônica, mecânica e programação, simulando o funcionamento de um elevador em um edifício.

O elevador será controlado por três botões, cada um correspondente a um andar. Um display mostrará o andar atual e uma seta indicará a direção do movimento. Um LED sinalizará quando o elevador estiver em funcionamento.

Para otimizar o código e facilitar a sua manutenção, dividiremos as tarefas em etapas de programação:

- **Desenhos():** configura o display para exibir informações como o andar atual e a direção do movimento.
- **Sobe():** ativa o motor para mover o elevador para cima.
- **Desce():** ativa o motor para mover o elevador para baixo.
- **Pare():** para o motor. Ao invés de repetir os mesmos comandos em diferentes partes do código, podemos chamar essas funções quando necessário, tornando o programa mais organizado e legível. Por exemplo, para fazer o elevador subir, basta chamar a função.

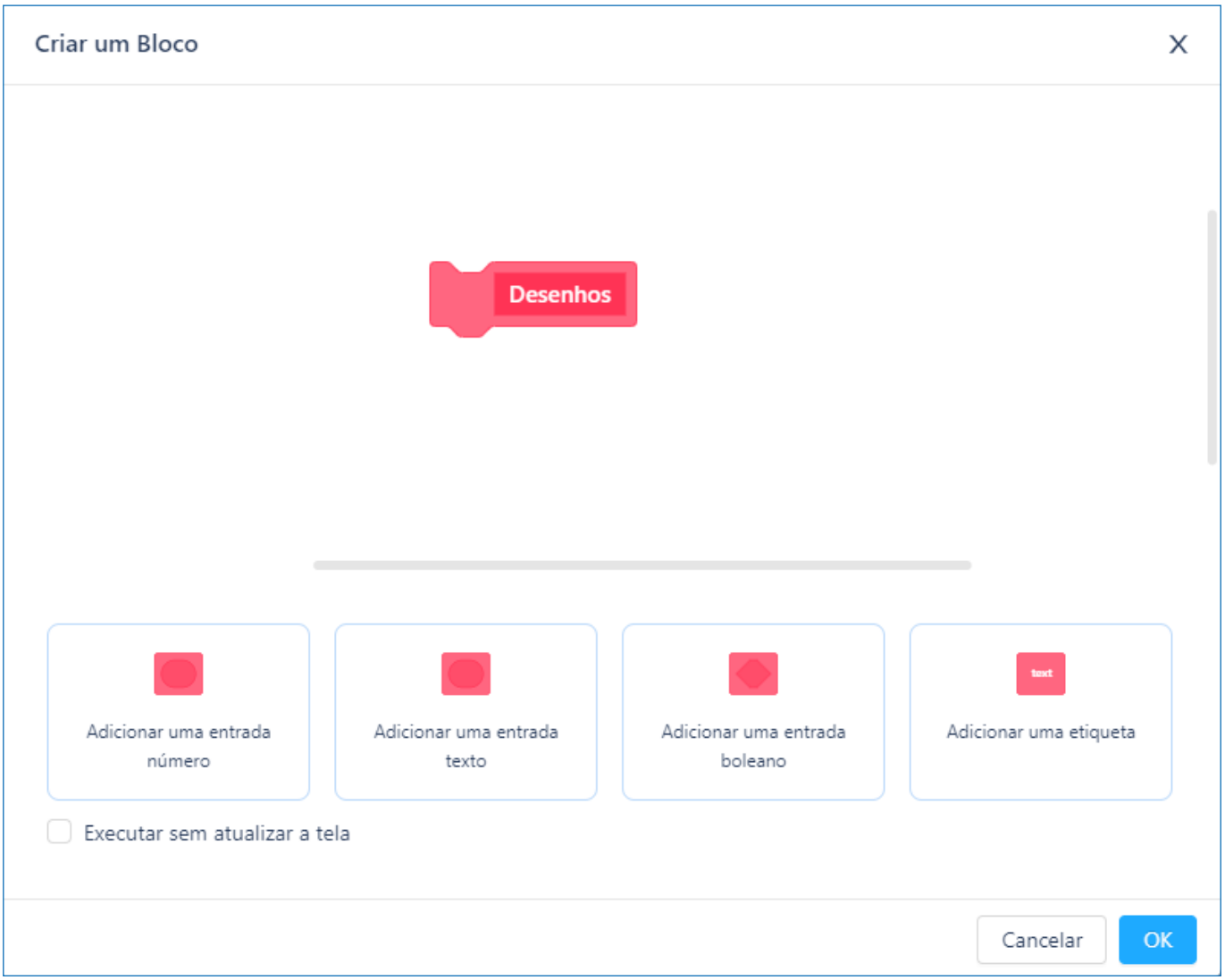
Cada função agrupa um conjunto de comandos relacionados a uma determinada tarefa. A função "desenhos", por exemplo, contém todas as instruções para criar e atualizar a interface gráfica do elevador no display. Os pilares do pensamento computacional ajudam a resolver problemas de maneira mais fácil e organizada. Ao usar funções no código, aplicamos esses pilares de forma prática. Primeiro, dividimos o problema em partes menores (decomposição), o que facilita o entendimento e a solução. Em seguida, reconhecemos padrões que podem ser repetidos no código, o que evita retrabalho. Além disso, as funções permitem simplificar as ideias (abstração), focando apenas no que é importante para resolver o problema. Por fim, cada função segue um conjunto de passos lógicos (algoritmos) que ajudam a alcançar a solução final.

Ao criarmos funções, encapsulamos os blocos em tarefas específicas a serem utilizadas. Para organizar o código, criaremos blocos que terão que executar determinadas tarefas, por exemplo, o bloco que irá desenhar as figuras que serão exibidas na matriz.

Para criar uma nova função, acesse a seção "**Os Meus Blocos**" e clique em "**Criar um bloco**". Nomeie o bloco como <**Desenhos**>.

Elevador elétrico II

Figura 10 - Criar o bloco desenho



Fonte: SEED/DTI/CTE.

Em seguida, adicione os comandos necessários para configurar o display e desenhar as figuras desejadas dentro desse bloco. Ao chamar a função "**desenhos**" em outra parte do programa, todos esses comandos serão executados.

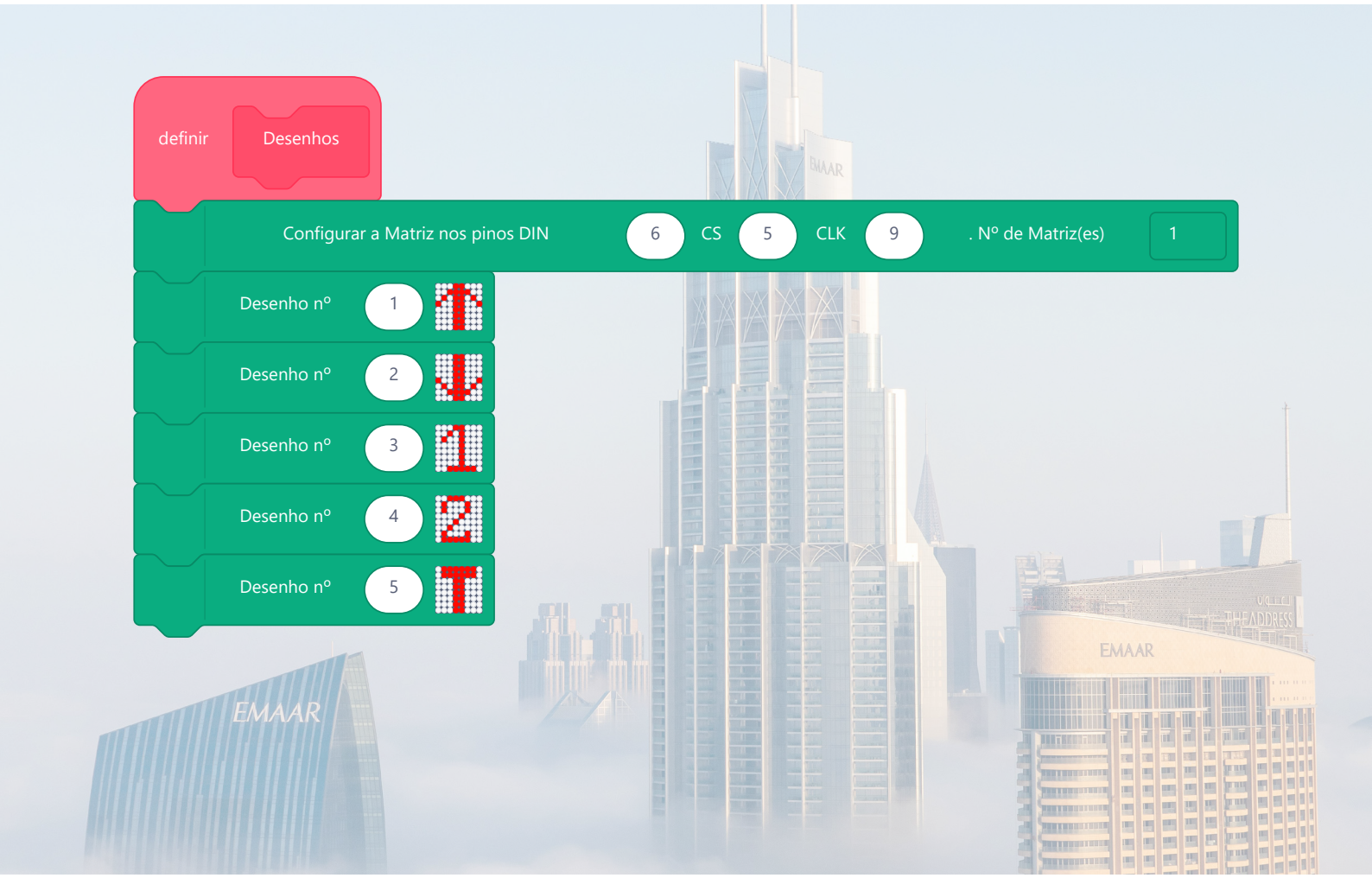
A função <**Desenhos**> é responsável por gerenciar a exibição de imagens na matriz de LEDs. Para configurá-la, inicie conectando o bloco <**Configurar a Matriz...**>. Nesse bloco, você deverá inserir os valores correspondentes às conexões digitais DIN, CS e CLK, que ligam o microcontrolador à matriz. Cada imagem que você deseja exibir na matriz receberá um número de identificação, como mostrado na Figura 11.

Elevador elétrico II

É fundamental que a extensão "**RP - Matriz de LEDs**" esteja instalada no mBlock para que você possa utilizar esses blocos. Para instalá-la, acesse a aba "Extensão", digite <**RP - Matriz de LEDs**> na barra de pesquisa e clique em "**Adicionar**".

Para criar ou modificar os desenhos que serão exibidos na matriz de LEDs, clique no ícone do desenho dentro do bloco. Isso abrirá um editor visual intuitivo, onde você poderá desenhar à mão livre ou escolher um dos desenhos pré-definidos (Figuras 10 e 11). Após configurar os pinos DIN, CS e CLK da matriz, arraste o bloco 'Desenho nº1' e selecione a imagem da flecha para cima para indicar o movimento ascendente do elevador. Repita esse processo para os demais desenhos: flecha para baixo para o movimento descendente, números 1 e 2 para indicar os andares e a letra T para representar o térreo (Figura 11). A Figura 13 apresenta um exemplo de como utilizar o editor de desenhos.

Figura 11 - Criação do bloco "Desenhos"



Elevador elétrico II

Figura 12 - Edição da figura na função "Desenhos"

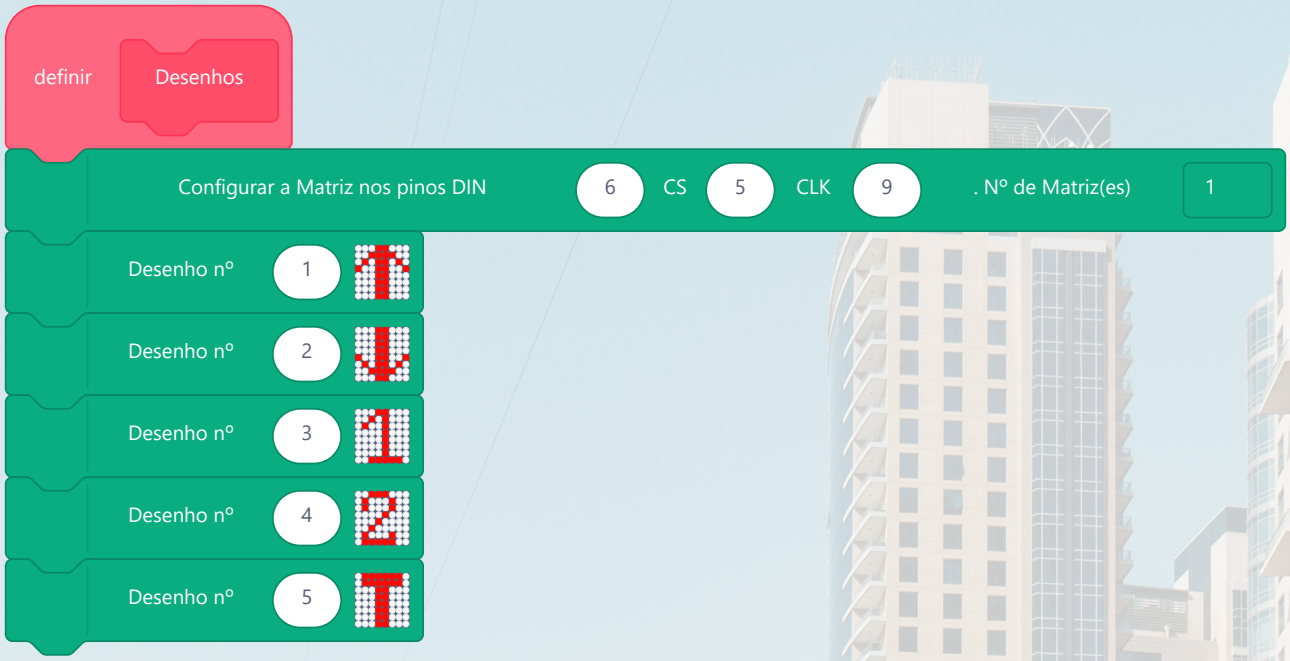
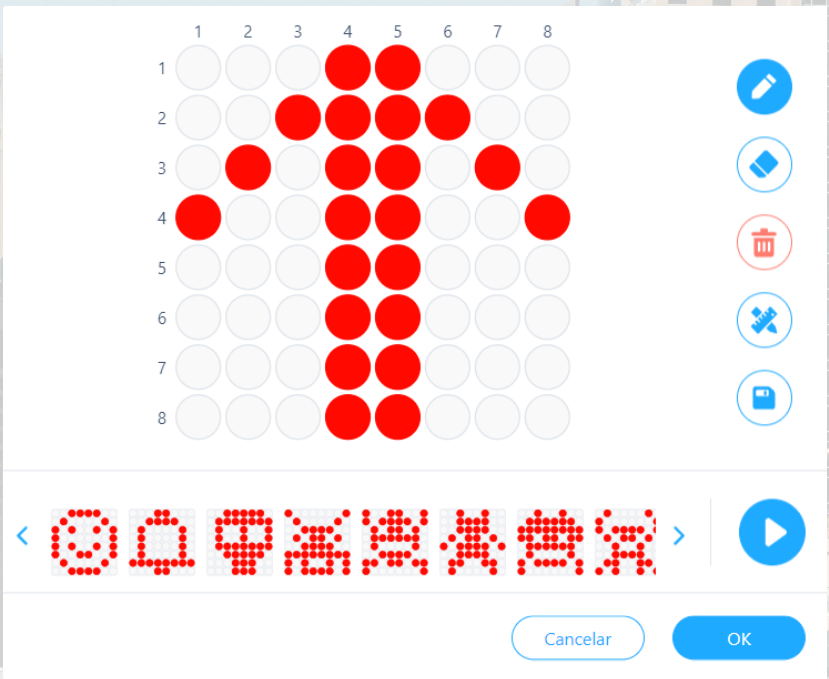


Figura 13 - Edição da imagem do display



Elevador elétrico II

Os cinco desenhos criados (térreo, primeiro e segundo andares, seta para cima e seta para baixo) serão utilizados para indicar visualmente o estado do elevador na matriz de LEDs. Para simular o movimento do elevador, precisamos criar funções que controlem o motor e atualizem a interface.

Para simular o movimento de descida do elevador, criaremos uma função chamada "Desce". Novamente, devemos ir à opção <**Os Meus Blocos**> e <**criar um bloco**>, agora basta nomeá-lo como <**Desce**>.

Essa função será responsável por:

- **Mostrar a seta para baixo:** na matriz de LEDs, mostraremos o desenho da seta apontando para baixo para indicar que o elevador está descendo.
- **Acender o LED:** ligaremos o LED para sinalizar que o elevador está em movimento.
- **Mover o motor:** enviaremos um sinal elétrico ao motor através da porta PWM 11, fazendo-o girar e mover o elevador para baixo. A velocidade do motor pode ser ajustada alterando o valor do sinal PWM.
- **Esperar:** pausaremos a execução do programa por 2,5 segundos para simular o tempo que o elevador leva para descer de um andar para outro.

O bloco <**Desce**> quando for executado, mostrará o desenho de uma seta para cima, nesse caso, o desenho 1. Além disso, a porta digital 2 (do LED) será ativada (colocada em nível HIGH) e a porta PWM 11 que fará o motor girar no sentido horário ou anti-horário (dependendo da ordem que os terminais do motor foram conectados ao borne da ponte H) recebe valor PWM 60 (para uma rotação mais lenta), conforme a Figura 14. O último bloco, <**esperar 2,5 segundos**>, tem por objetivo esperar o elevador chegar ao próximo andar.

Elevador elétrico II

Figura 14 - Bloco "Desce"

Elevador elétrico II

Para simular a subida do elevador, precisamos fazer algumas alterações na função. Primeiramente, trocamos o desenho exibido na matriz de LEDs para a seta que indica a direção para cima (bloco 3, conforme Figura 11). Em seguida, mantemos o LED ligado para indicar que o elevador está em movimento. Para fazer o motor girar no sentido oposto e elevar o elevador, ajustamos o valor PWM da porta 10 para 70. Esse valor define a velocidade do motor. Por fim, inserimos um bloco de espera de 4 segundos para simular o tempo que o elevador leva para subir entre os andares.

Figura 15 - Bloco "Sobe"

Elevador elétrico II

Para finalizar a programação do elevador, criaremos o bloco "Parar". Esse bloco será responsável por fazer com que o elevador pare no andar selecionado pelo usuário e permaneça parado até que seja solicitado um novo movimento. Ao executar esse bloco, o motor do elevador será desligado e a indicação visual no display será atualizada para mostrar que o elevador está parado no andar desejado. Para garantir que o elevador permaneça parado no andar desejado, o bloco <**Parar**> configura os valores de PWM como zero, parando o motor, e desativa o LED, indicando que o elevador não está em movimento.

Figura 16 - Bloco "Parar"

Uma vez que definimos as principais funções que usaremos, devemos pensar na lógica do elevador:

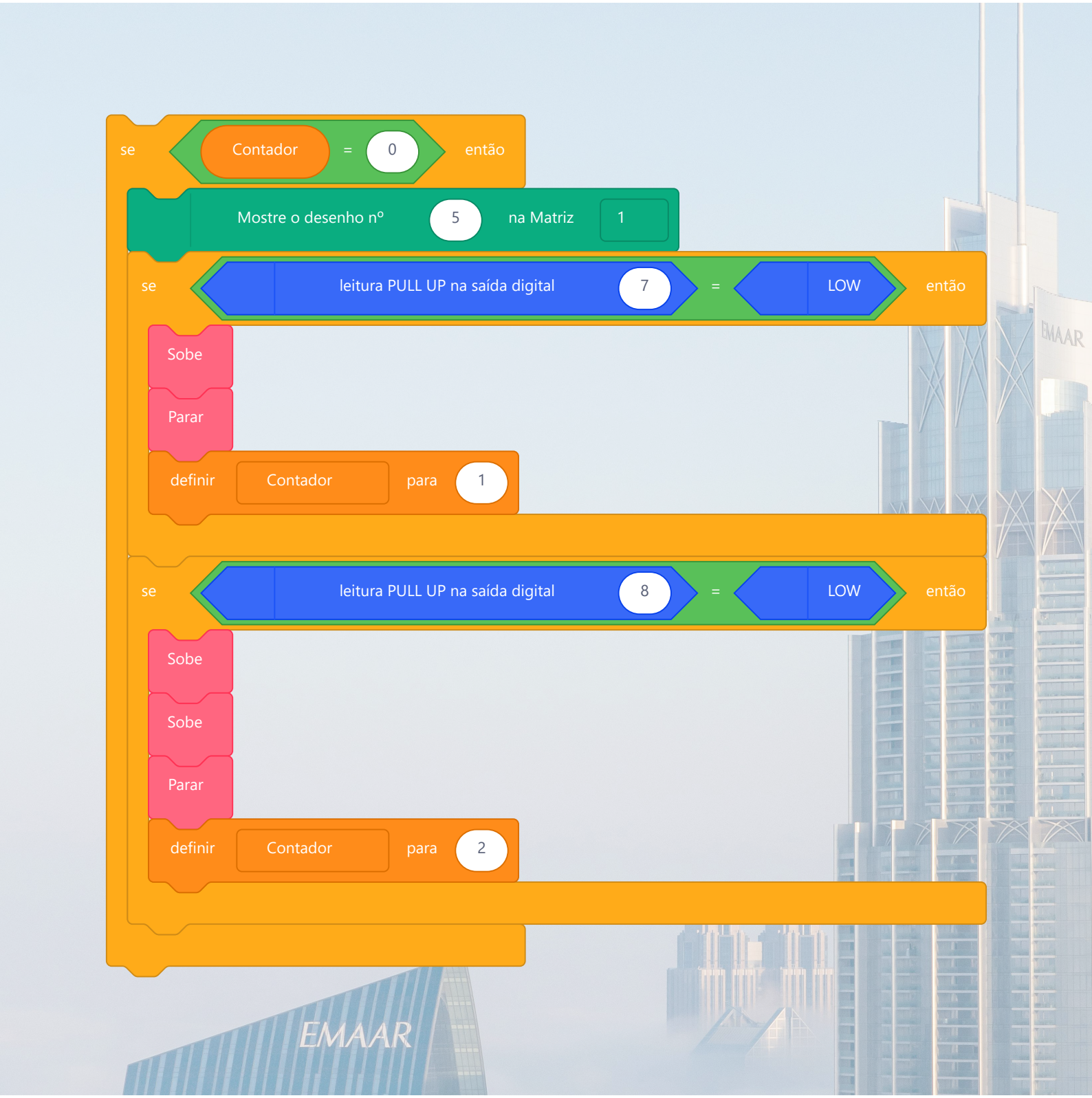
A lógica é da seguinte forma - o sistema precisa ter a referência de qual andar o elevador se encontra, para ser possível definir o quanto deve descer ou subir para chegar ao andar cujo botão foi pressionado. Para isso, criamos uma variável chamada "**Contador**", que começa em zero e tem seu valor alterado à medida que subimos ou descemos o elevador. 0 (zero) corresponde ao térreo, 1 ao primeiro andar e 2 ao segundo.

Conforme a Figura 16, se o elevador estiver no térreo (identificado para o contador como o número zero) e o botão do primeiro andar for apertado, ele deve subir uma única vez. Já se o toque for no segundo, ele deve subir duas vezes.

Pode-se identificar que um botão foi apertado por sua saída digital estar igual a LOW.

Elevador elétrico II

Figura 17 - Elevador no térreo



Elevador elétrico II

Quando o <**Contador**> estiver em 0, o display deve mostrar a imagem referente ao térreo, desenho 5. Após chegar ao andar desejado, seja ele o primeiro (identificado pela porta digital 7) ou o segundo (identificado pela porta digital 8), o contador deve ser atualizado para a nova posição (1 ou 2). Vale ressaltar que a quantidade de vezes que as funções "sobe" e "desce" são chamadas depende do número de andares a serem subidos ou descidos.

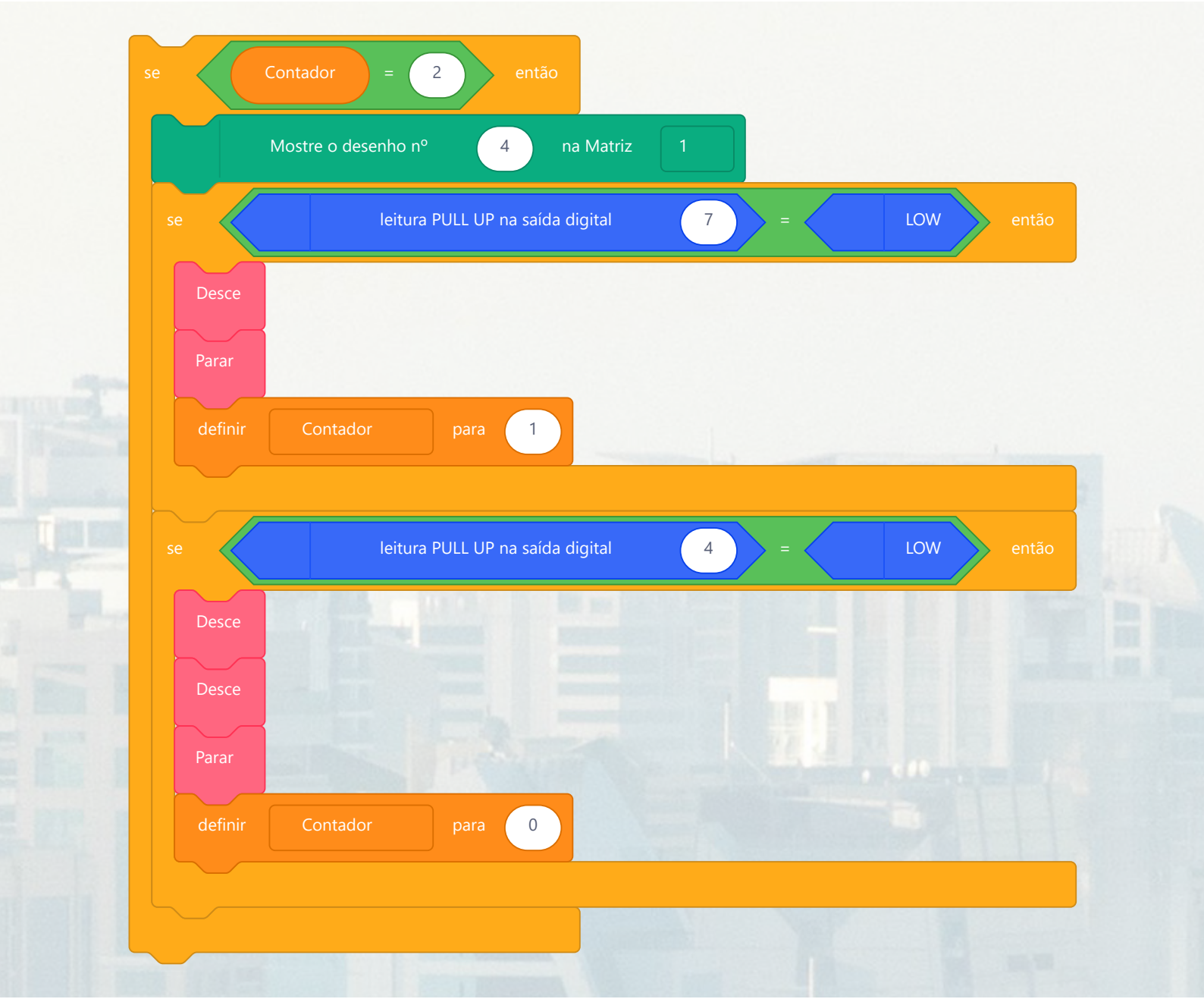
Essa mesma lógica se aplica quando o contador está em 1 (primeiro andar) ou em 2 (segundo andar), mudando apenas as informações referentes às portas e saídas. As Figuras 18 e 19 a seguir ilustram esses trechos, respectivamente.

Figura 18- Elevador no primeiro andar

Elevador elétrico II

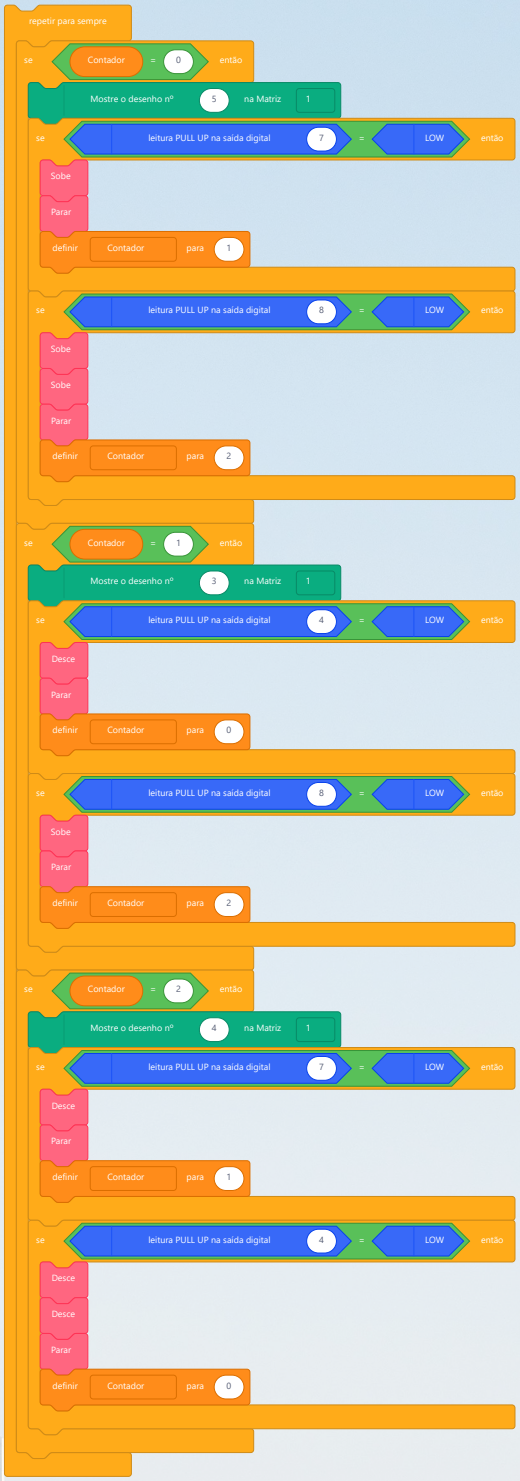
Nesse trecho, quando o contador for igual a 1, o desenho referente ao primeiro andar será exibido, nesse caso o nº 3, e se o botão do térreo (porta digital 4) for pressionado, o elevador deve descer uma vez e parar, e em seguida definir o contador para zero. Já se o botão do segundo andar for pressionado (porta digital 8) o elevador subirá uma vez e parar e, em seguida, modificar o contador para 2.

Figura 19 - Elevador no segundo andar



Elevador elétrico II

Figura 20 - Laço repetir para sempre / possibilidade de movimentação do elevador



Quando o contador for igual a 2, o desenho referente ao primeiro andar será exibido, nesse caso o nº 4.

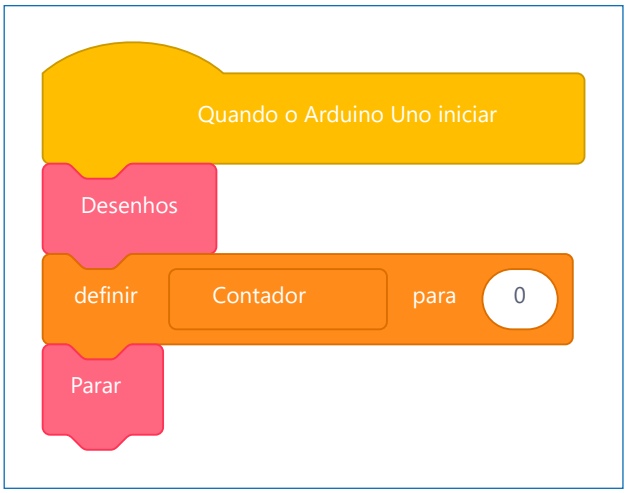
Nesse momento, o elevador só pode descer, caso o botão do primeiro andar (porta digital 7) for pressionado, o elevador deve descer uma vez e parar, e em seguida definir o contador para 1. Já se o botão do térreo for pressionado (porta digital 4), o elevador deve descer duas vezes e parar e, em seguida, modificar o contador para 0.

Como queremos que o elevador fique funcional o tempo todo, devemos colocar esses 3 trechos de código em sequência em um laço <**repetir para sempre**>, conforme mostra a Figura 20.

Elevador elétrico II

Para finalizar, basta ajustar os valores iniciais do programa e juntar com o trecho da Figura 19, como mostram as Figuras 20 e 21.

Figura 21 - Início do programa

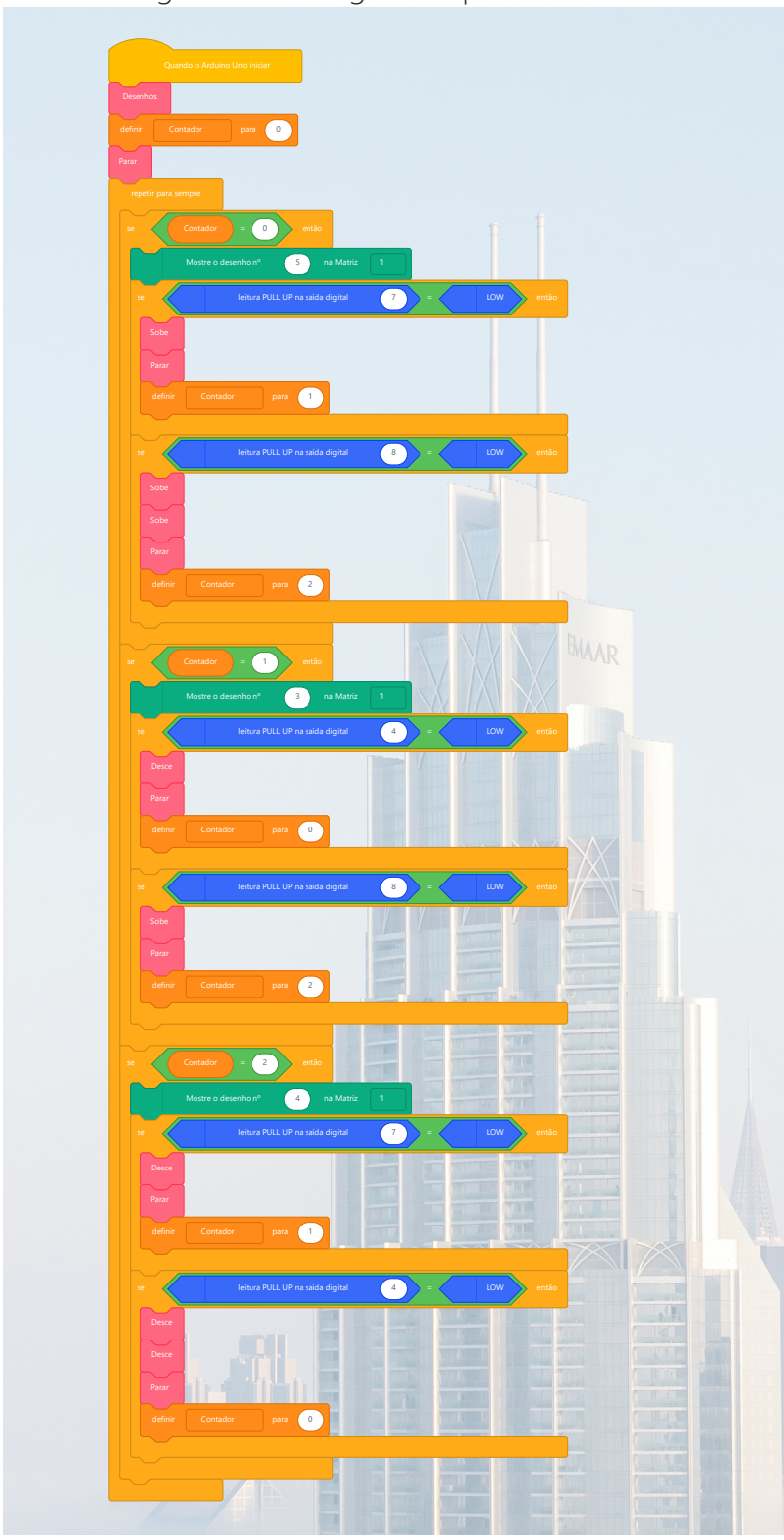


O primeiro bloco que colocamos ao inicializar o Arduino é chamar a função <**Desenhos**> e definir o contador para 0 <**definir "Contador" para___**> (inicializando o elevador no térreo), em seguida, utilizamos o bloco que criamos <**Parar**> (para que o motor inicie parado).

Em sequência, basta colocar o laço <**repetir para sempre**> (visível na Figura 18) abaixo do código da Figura 22.

Assim, finalizamos nossa programação, agora podemos fazer os testes e verificar se tudo irá funcionar conforme o planejado.

Figura 22 - Códigos completos



Fonte: SEED/DTI/CTE.

Elevador elétrico II

Desafios:

Que tal...

1. Acrescentar portas ao elevador elétrico com servomotores e sensores para abrir e fechar a porta?
2. Fazer a modificação da lógica de programação para responder a emergências? Essa nova funcionalidade tornará o projeto ainda mais desafiador e realista.

E se...

O elevador não estiver funcionando, verifique os seguintes pontos:

- **Conexões:** certifique-se de que todos os jumpers estão conectados corretamente aos pinos dos componentes e do Arduino, e que a bateria está bem conectada.
- **Programação:** verifique se os pinos digitais estão configurados corretamente e se a lógica de programação está de acordo com o diagrama esquemático.

Dicas adicionais:

- **Teste os componentes individualmente:** se o elevador não estiver funcionando, teste cada parte separadamente. Por exemplo, ligue o motor para ver se ele gira. Isso ajuda a identificar se algum componente específico está com problema.
- **Peça ajuda e colabore:** se você ainda não conseguir descobrir o que está errado, não hesite em pedir ajuda. Converse com seus colegas ou pergunte ao seu professor. Trabalhar em equipe pode trazer novas ideias e soluções que você não havia considerado.
- **Consulte a documentação:** se você estiver utilizando alguma biblioteca ou componente específico, consulte a documentação para obter mais informações sobre sua configuração e utilização.

3. Feedback e finalização

Compartilhe seu projeto de elevador com seus colegas e discutam os resultados obtidos. Avaliem se o elevador funciona como esperado, considerando os seguintes aspectos:

- **Funcionalidade:** o elevador realiza todas as tarefas programadas, como subir, descer e parar nos andares corretos?
- **Precisão:** o elevador para nos andares com a precisão desejada?
- **Confiabilidade:** o elevador funciona de forma consistente, sem falhas frequentes?

Além disso, reflitam sobre o processo de desenvolvimento:

- **Trabalho em equipe:** como foi a colaboração entre os membros da equipe? As ideias foram compartilhadas e discutidas de forma construtiva?
- **Resolução de problemas:** quais foram os principais desafios enfrentados durante o projeto? Como vocês os superaram?
- **Aprendizado:** quais novos conhecimentos foram adquiridos durante a realização do projeto? Como esses conhecimentos podem ser aplicados em futuros projetos?

Reúna todos os componentes utilizados nesta aula e os organize novamente, junto aos demais, no kit de Robótica.

3. REFERÊNCIAS

ARDUINO. **Documentação de Referência da Linguagem Arduino**. Disponível em: https://www.arduino.cc/reference/pt/. Acesso em: 27 mai. 2024.

ARDUINO. **Powering Alternatives for Arduino Boards**. Disponível em: https://docs.arduino.cc/learn/electronics/power-pins/. Acesso em: 07 out. 2024.

WERNER VON SIEMENS. **Wikipédia, a enciclopédia livre**: https://pt.wikipedia.org/wiki/Werner_von_Siemens. Acesso em 23 mai.2024.

Tutor Brasil. **Sistema mecânico do elevador**. Disponível em: www.tutorbrasil.com.br/forum/viewtopic.php?t=37173. Acesso em:07 out. 2024.

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL (UFMS)
FACULDADE DE COMPUTAÇÃO (FACOM)

PROFESSORES

- Amaury Antônio de Castro Junior
- Anderson Corrêa de Lima
- Glauder Guimarães Ghinozzi
- Graziela Santos de Araújo
- Said Sadique Adi

ESTUDANTES (elaboração prévia)

- Filipe de Andrade Machado - Ciência da Computação
- Gabriel Alves Massuda Duarte - Engenharia de Computação
- José Augusto Lajo Vieira Vital - Ciência da Computação
- Lorena Valente Cavalheiro - Engenharia de Computação
- Matheus Kazumi Silva Miyashiro - Engenharia de Computação
- Nathalia dos Santos Melo - Engenharia de Software
- Yan Arruda Cunha - Engenharia de Computação
- Thiago Ferronatto - Ciência da Computação
- Vitor Hugo dos Santos Duarte - Engenharia de Computação
- Wilker Sebastian Afonso Pereira - Ciência da Computação

ESTUDANTES (revisão)

- Arthur Henrique Andrade Farias - Ciência da Computação
- Bruno Pereira Wesner da Silva - Engenharia de Computação
- Fernanda das Neves Merqueades Santos - Ciência da Computação
- Gabriel Pereira Falcão - Ciência da Computação
- Jenniffer Oliveira Checchia - Ciência da Computação
- Leonardo Vargas de Paula - Sistemas de Informação
- Marcos Gabriel da Silva Rocha - Engenharia de Computação
- Maria Paula do Nascimento Santos - Engenharia de Computação
- Nathanael Martins Wink - Ciência da Computação
- Victor Luiz Marques Saldanha Rodrigues - Ciência da Computação

DIRETORIA DE TECNOLOGIAS E INOVAÇÃO (DTI)

COORDENAÇÃO DE TECNOLOGIAS EDUCACIONAIS (CTE)

EQUIPE ROBÓTICA PARANÁ

- Adilson Carlos Batista
- Ailton Lopes
- Andrea da Silva Castagini Padilha
- Cleiton Rosa
- Darice Alessandra Deckmann Zanardini
- Edna do Rocio Becker
- Kellen Pricila dos Santos Cochinski
- Marcelo Gasparin
- Michele Serpe Fernandes
- Michelle dos Santos
- Roberto Carlos Rodrigues
- Sandra Aguera Alcova Silva
- Viviane Dziubate Pittner

Os materiais, aulas e projetos da "Robótica Paraná", foram produzidos pela Coordenação de Tecnologias Educacionais (CTE), da Diretoria de Tecnologia e Inovação (DTI), da Secretaria de Estado da Educação do Paraná (SEED), com o objetivo de subsidiar as práticas docentes com os estudantes por meio da Robótica.
Este material foi produzido para uso didático-pedagógico exclusivo em sala de aula.