dem
universidade de aveiro
departamento de engenharia mecânica

Curso Mestrado em Engenharia e Automação Industrial

Disciplina Projeto em Engenharia de Automação

Ano letivo 2020/2021

RELATÓRIO

Projeto e controlo de uma máquina de estampagem incremental compacta

Jorge Fernandes, nº 104580

Data 16/07/2021

Orientador Professor Doutor Daniel Gil Afonso

Resumo Projeto do quadro elétrico, seleção de motores e programação da cinemática de uma máquina compacta para estampagem incremental baseada num braço RR plano.

Índice remissivo

Índice de figuras

1. Introdução

A utilização de máquinas controlados por comandos numéricos tem vindo a aumentar nas últimas décadas. São exemplos disso, o uso de CNCs, máquinas de corte a laser, impressoras 3D, braços robotizados, etc. Todos estes equipamentos trazem benefícios para a indústria em geral, reduzindo tempos de fabrico de componentes e aumentando a repetibilidade.
Por norma, o seu acionamento é feito por meio de um código, interpretado pela máquina de forma sequencial, contendo um conjunto de posições e/ou ações a ser tomadas. Este código, designado de código máquina ou g-code, é interpretado por um controlador responsável por acionar os componentes da máquina, como por exemplo motores, termístores, fusos, entre outros.

Neste projeto pretende-se selecionar um conjunto de componentes para controlo de uma máquina de estampagem incremental compacta baseada num braço robótico RR plano. Esta máquina tem um total de 3 eixos, X, Y, Z e será controlada por um controlador a selecionar. O controlador será responsável pela cinemática da máquina, para que se consiga o correto posicionamento da ferramenta de estampagem, um punção, nas múltiplas posições desejados de modo a conformar a chapa com a forma pretendida.
No decorrer deste projeto são selecionados: motores, drivers e todos os componentes elétricos de funcionamento e segurança necessário para o quadro elétrico. É determinada a cinemática da máquina e implementado o código para interpretação de comandos numéricos vindos do computador por porta de comunicação. Por último é desenvolvido uma interface para computador, de modo que o utilizador possa controlar a máquina de forma prática.

1.1 Estampagem incremental

O processo de estampagem incremental consiste numa tecnologia utilizada para conformar chapas de metal utilizando deformações incrementais sequenciais. É utilizada uma ferramenta de ponta arredondada, acoplada a uma CNC ou braço robótico, que deforma a chapa com o contorno da peça pretendida.
O percurso de uma peça fabricada por estampagem incremental começa pelo desenhado num software de CAD que é depois transferido para uma CNC, com o correto prato e sistema de grampos ou gabarito. Posiciona-se a chapa de metal na posição desejada, alinhada com a ferramenta de deformação e dá-se início ao processo. Finalizada a deformação, a peça está pronta a utilizar, não necessitando de nenhum processo adicional.
As principais aplicações de peças conformadas por estampagem incremental são protótipos, aplicações na medicina, aplicações aeronáuticas e automotivas sendo as suas vantagens:

- Facilidade na prototipagem rápida;
- Forças de conformação baixas, devido à natureza do processo incremental;
- É facilmente adaptado a qualquer tipo de CNC;
- Peças finais são produzidas diretamente dos ficheiros CAD;
- Não necessita de matrizes adicionais para a conformação;
- As dimensões das peças só são limitadas pelas dimensões da máquina;
- Excelente acabamento superficial.

Por outro lado, as desvantagens deste processo são:

- Elevados tempos de conformação quando comparado com outros processos de estampagem;
- Apresenta menor precisão dimensional especialmente em zonas com ângulos acentuados;
- Dificuldades na obtenção de ângulos retos;
- Pode ocorrer retorno do material (*springback*) após a conformação.

A Figura 1 ilustra alguns exemplos de estampagem incremental:

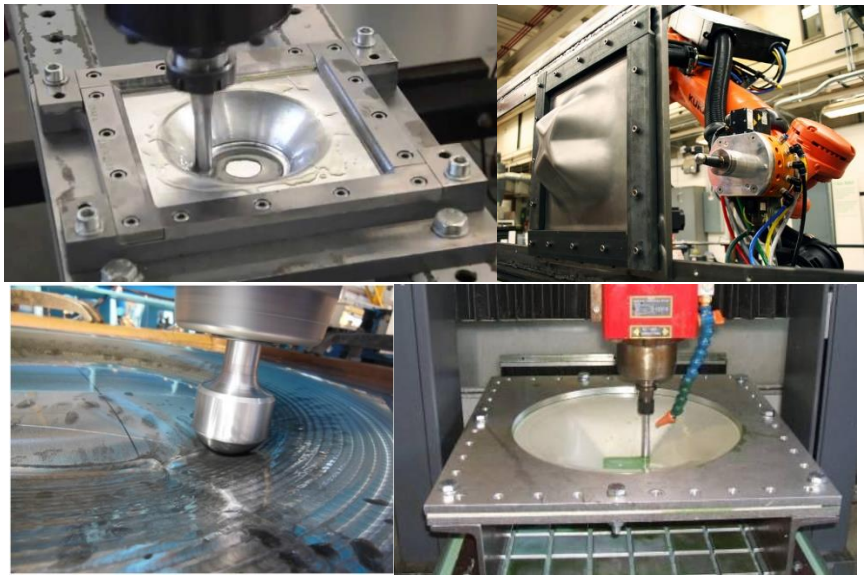


Figura 1 - Estampagem incremental

1.2 Máquina de estampagem incremental compacta

O projeto de construção de uma máquina de estampagem incremental compacta parte da necessidade de utilização de peças estampadas para protótipos. O acesso a uma máquina deste tipo, em conjunto com o uso de fresadoras ou impressoras 3D facilita a prototipagem rápida para testes ou ensaios.
Neste projeto é abordado somente as componentes elétricas e de comando da máquina, contudo para a programação da cinemática e para a seleção dos motores são tidos em conta a forma, as forças e os momentos da máquina durante o processo de estampagem. Será utilizado para controlo da ferramenta, nas coordenadas Y e Z, um braço robótico RR plano e para controlo do eixo X, onde fica o prato de suporte da chapa, será utilizado dois fusos.

As dimensões definidas para a máquina são:

- Braço de 120 mm posicionado a uma altura de 200 mm da base;
- Antebraço de 200 mm;
- Prato quadrado de lado 200 mm.

O eixo de coordenados encontra-se no centro da máquina. [Figura 2]

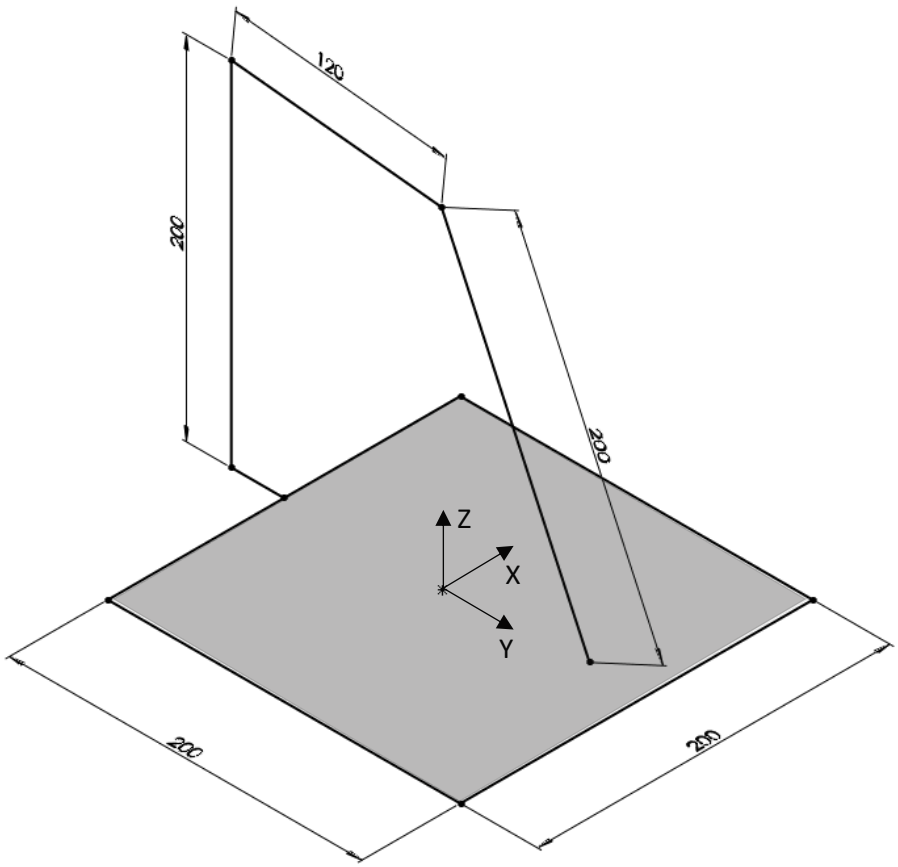


Figura 2 - Máquina de estampagem compacta (Unifilar - vista isométrica)

Os 4 motores a selecionar MZ, MY, MX1 e MX2 serão posicionadas como na Figura 3. As forças máximas estimadas para a estampagem de chapa de alumínio são de: 3 kN no eixo vertical Z e 1 kN nos eixos horizontais Y e X. Os binários TZ, TY e TX esperados são de:

$$TZ = 660\ Nm$$
$$TY = 400\ Nm$$
$$TX = 32\ Nm$$

Nas ligações aos 4 motores está previsto o uso de uma caixa redutora planetária com uma relação de transmissão de 100:1. Deste modo o binário máximo esperado no motor será 100 vezes menor do que o sentido nas ligações.
Apesar dos binários não serem iguais nos 3 eixos, serão selecionados motores semelhantes que satisfaçam os requisitos para o eixo Z, com um binário máximo de 660 Nm.
Posto isto, é esperado que os motores selecionados tenham um binário nominal de 6.6 Nm à velocidade máxima de funcionamento.

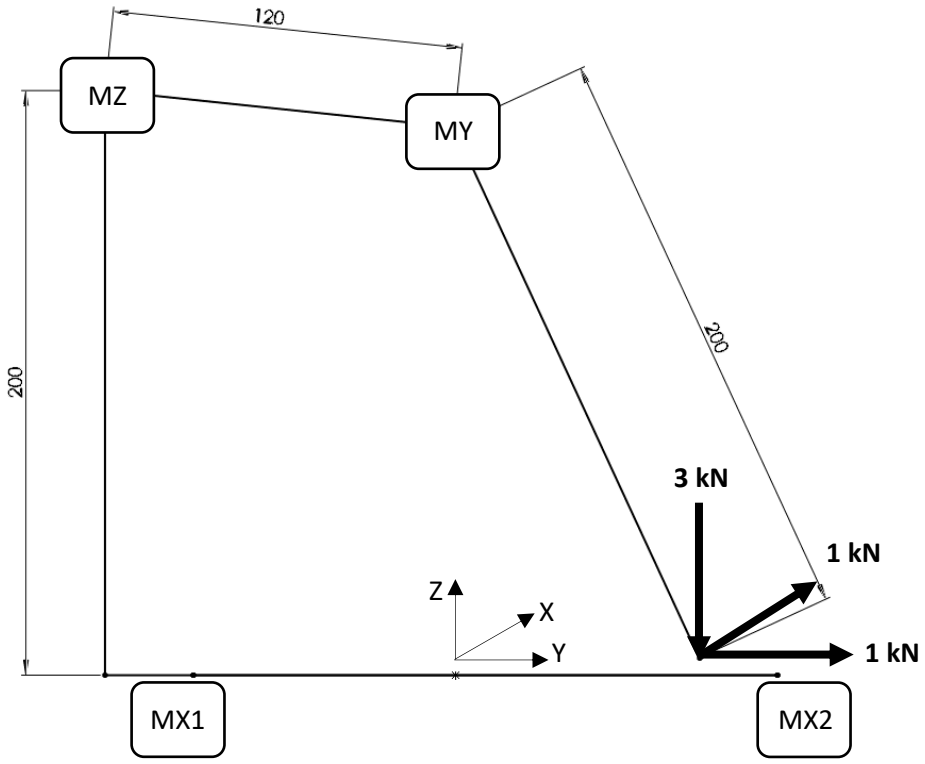


Figura 3 - Forças nos eixos Z, Y e X (Unifilar - vista lateral)

2. Seleção de motores

Os motores pretendidos para a aplicação devem garantir a completa conformação da peça sem que ocorrência de perca de passos. Ou seja, na presença de um aumento de binário, o motor deve garantir que finaliza todas as instruções enviadas pelo controlador. A maneira de conseguir este pressuposto é utilizando motores em malha fechada. Este tipo de motores possuem acoplado ao seu veio um componentes que vai enviar um sinal de resposta para o controlador, correspondente ao deslocamento real efetuado pelo veio, denominado de encoder. Assim sempre que o controlador enviar um determinado deslocamento para o motor, o encoder vai responder ao controlador com a quantidade de deslocamento real efetuado pelo veio do motor e o controlador vai saber sempre se o motor se encontra na posição correta. No caso de falha, o controlador deve tomar uma ação. A Figura 4 ilustra o funcionamento de um motor em malha fechada.

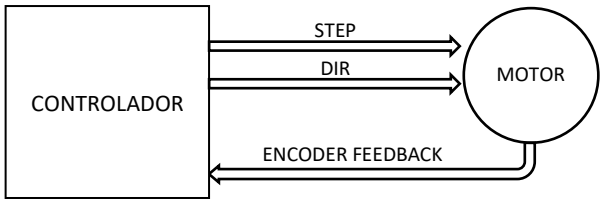


Figura 4 - Motor em malha fechada

Dentro do mercado existem diferentes ofertas no que toca a motores de malha fechada sendo os ponderados para a aplicação: servo motores, motores sem escovas de corrente contínua (BLDC – *brushless direct current*) e motores de passo.

2.1 Motores sem escova de corrente contínua (BLDC)

Foi ponderado para os motores sem escovas de corrente contínua (*BLDC*) o uso do motor ODrive. Este motor possui o seu próprio controlador, que é programado em linguagem python. O controlador ODrive possui todas as conexões para ligar os 3 fios de dois motores BLDC (o controlador contém o *ESC* integrado – *Electronic Speed Control*), ligações para um encoder de cada motor, ligações para alimentação, ligações para a resistência de travagem e ligação USB [Figura 5].

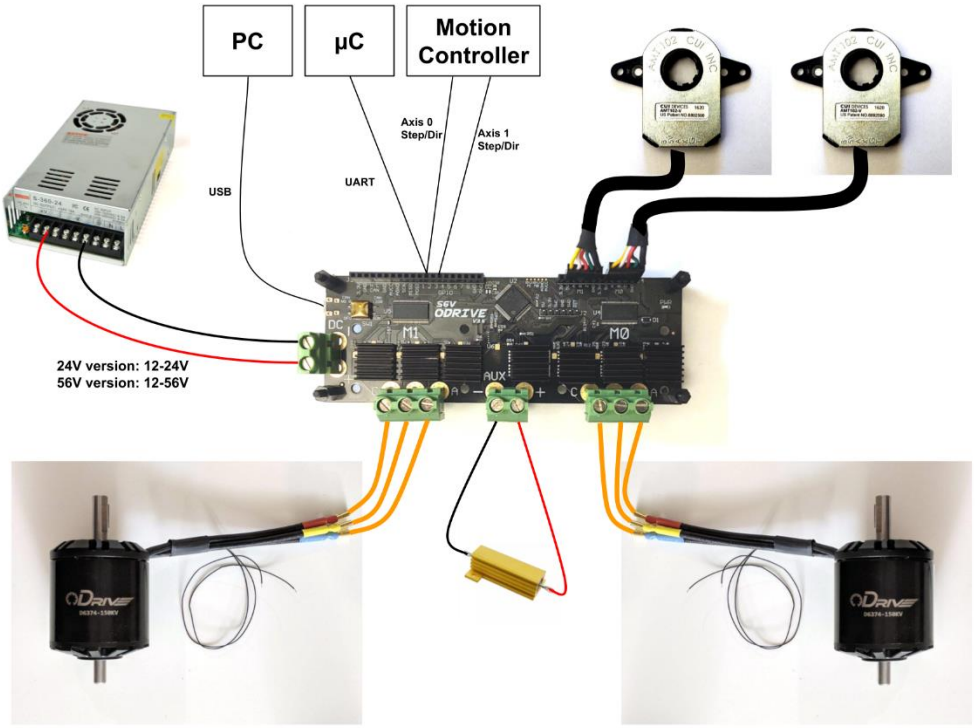


Figura 5 - Ligações ODrive

Este controlador não obriga a utilizar motores da sua marca, mas só trabalha com motores BLDC. Os motores compatíveis com o ODrive têm binários de trabalho na média dos 4 Nm e a característica de apresentarem um binário maior para elevadas rotações.
Nesta aplicação, as velocidades são reduzidas, mesmo com a caixa redutora utilizada, e os binários desejados para os motores são da ordem dos 7 Nm, assim concluiu-se que este tipo de motores não seria o mais apropriado.
Contudo é importante referir que o controlador ODrive apresenta grandes potencialidades a um preço reduzido quando comparado com outros produtos do género. É um controlador ideal para aplicações como CNCs de gravação (*engraving machines, cartesian plotter*), braços robóticos de bancada, e todo o tipo de aplicações que trabalhem com baixos/médios binários e velocidades elevadas que necessitem de um elevado controlo de precisão.

2.2 Servo motores

Os servo motores são os mais utilizados a nível industrial. Estes apresentam uma elevada gama de binários e velocidades, compatíveis com todo o tipo de aplicações.
Cada motor possui o seu driver associado, sendo o driver o componente responsável por controlar a tensão e corrente que chegam ao motor. Este driver pode ser comandado por um controlador separado, ou em alguns casos pode funcionar sozinho, necessitando de uma programação dedicada.
No que toca a servo motores, a oferta no mercado é enorme, mas os preços são bastante mais elevados quando comparados com o ODrive. Foram consultados os modelos:

- Yaskawa Sigma 5;
- Mitsubishi MELSERVO;
- Oriental Motors BMU Series;
- Delta Servo System ECMA.

Qualquer uma destas marcas apresentava soluções com binário suficiente para a aplicação, contudo nenhuma se encontrava dentro do orçamento estipulado inicialmente. Concluiu-se que os servo motores, apesar de satisfazerem perfeitamente as exigências da aplicação, os preços praticados não se enquadram neste projeto.

Figura 6 - Exemplos de servo motores

2.3 Motores de passo (solução proposta)

Como referido anteriormente, a máquina vai trabalhar a baixas velocidades e com elevados binários. Os motores de passo, em semelhança com os servo motores, são ligados a um driver, responsável pela gestão da tensão e da corrente para o motor. Estes motores apresentam a característica de um maior binário para baixa rotação, contudo a grande maioria trabalha em malha aberta, ou seja, não existe comunicação vinda do motor. Este funcionamento pode ser contornado com o uso de motores de eixo duplo, acoplando um encoder numa das extremidades do eixo.

No mercado já existem soluções deste género, onde os motores de passo são acoplados a um encoder, e a resposta do encoder é enviada para o controlador ou para o driver. Na grande maioria das aplicações profissionais, este sinal é na verdade enviado para o driver. Este tipo de drivers possui um circuito integrado que processa os sinais vindos do encoder. Deste modo reduz-se a quantidade de informação que o controlador necessita de processar, aumentado a sua performance.

Com isto, a procura dos motores indicados para a aplicação assenta em encontrar motores de passo, que funcionem em malha fechada e apresentem binário suficiente para satisfazer as forças e momentos estimados.
Foram contatados vários fornecedores para que se procedesse a uma comparação preço/qualidade onde se selecionaram 3 propostas:
Damen CNC [Figura 7]:

- 4 x Kit de motor + driver e cabos;
    - Motor de passo NEMA 34 de 8 Nm com encoder;
    - Driver HBS86 30-80 VDC, 8.2 A de corrente de pico;
    - Cabo de ligação do encoder com 3 metros;
    - Cada por 289.99€;
- 4 x fonte de alimentação de 48 VDC 400 W cada por 74.99€.

O fornecedor é sediado na Holanda, assim incluindo o valor de transporte e IVA o equipamento terá um custo total de 1808.85€.

Figura 7 - Encomanda Damen CNC

StepperOnline [Figura 8]:

- 3 x Kit de motor + driver e cabos;
    - Motor de passo NEMA 34 de 12 Nm com encoder;
    - Driver CL86T 24-80 VDC, 8.2 A de corrente de pico;
    - Cada por 170.89€;
- 1 x Motor de passo NEMA 34 de 12 Nm com encoder e travão para o eixo Z por 174.56€;
- 1 x Driver CL86T 24-80 VDC, 8.2 A de corrente de pico por 52.30€;
- 4 x fonte de alimentação de 48 VDC 400 W por 25.38€;
- 1 x fonte de alimentação de 24 VDC 100 W por 10.15€.

A sede do fornecedor é na China, assim incluindo o valor de transporte e IVA o equipamento terá um custo total de 1509.97€. Nesta encomenda foi incluída uma fonte de alimentação de 24 VDC destinada ao controlador.

Figura 8 - Encomenda StepperOnline

Igus [Figura 9]:

- 3 x Motor de passo NEMA 34 de 5.9 Nm com encoder, cada por 635€;
- 1 x Motor de passo NEMA 34 de 5.9 Nm com encoder e travão para o eixo Z por 946.42€;
- 4 x Driver Drylin D1 12-48 VDC, 21 A de corrente de pico cada por 386.58€. Este driver funciona como controlador para 1 eixo com:

- 10 entradas digitais;
- 5 saídas digitais;
- 2 entradas analógicas de sinal ±10 VDC;
- Controlo por computador, tablet ou PLC;
- Comunicação por CANopen e/ou ModbusTCP;

Este fornecedor faz o envio de Portugal, assim incluindo o valor de transporte e IVA o equipamento terá um custo total de 5481.40€. Nesta encomenda não são contempladas as fontes de alimentação.

Figura 9 - Encomenda Igus

Interessa referir que todos os fornecedores referidos têm certificação CE e respeitam a diretiva RoHS.

A Tabela 1 apresenta a comparação entre os fornecedores referidos:

Damen CNC		**StepperOnline**		**Igus**	
Vantagens	*Desvantagens*	*Vantagens*	*Desvantagens*	*Vantagens*	*Desvantagens*
✓ Dentro da União Europeia;	× Não comercializa motor com travão; × Binário máximo de 8 Nm; × Preço total 1808.85€;	✓ Binário de 12 Nm; ✓ Motor com travão; ✓ Preço total 1509.97€;	× Encomenda vem da China;	✓ Motor com travão; ✓ Comercializado em Portugal; ✓ Drivers de qualidade superior com funções de controlador;	× Preço total 5481.40€; × Binário máximo de 5.9 Nm; × Encomenda não contempla as fontes de alimentação;

Tabela 1 - Comparação global entre fornecedores

As características comuns a todos não foram contempladas na tabela anterior.
Com isto, a solução mais vantajosa é a proposta da StepperOnline, isto porque cumpre todos os requisitos e apresenta o custo mais baixo bem como o maior binário (*holding torque*).
Algumas características do motor selecionado encontram-se na Tabela 2:

Binário (*holding torque)*	12 N.m
Bipolar	2 enrolamentos, 4 fios
Corrente por fase	6 A
Passo/pulso	1.8°

Tabela 2 - NEMA 34

É esperado que os motores trabalhem a uma velocidade (*feed rate*) entre 500 e 1500 pulsos por segundo, o que dá um tempo entre pulsos de 2 milissegundos a 650 microssegundos, respetivamente. Uma velocidade aceitável para este tipo de motores.
O motor vai ser configurado para trabalhar a 400 pulsos por revolução, sendo este o menor valor aceite pelo driver selecionado. Este valor garante que se extrai o máximo de binário do driver.
Com esta configuração e com o *feed rate* mencionado é esperado que o motor trabalhe de 75 a 225 rpm. O binário mínimo esperado é de aproximadamente 7 Nm.
A Figura 10 apresenta a curva de binário do motor, disponibilizada pelo fornecedor.

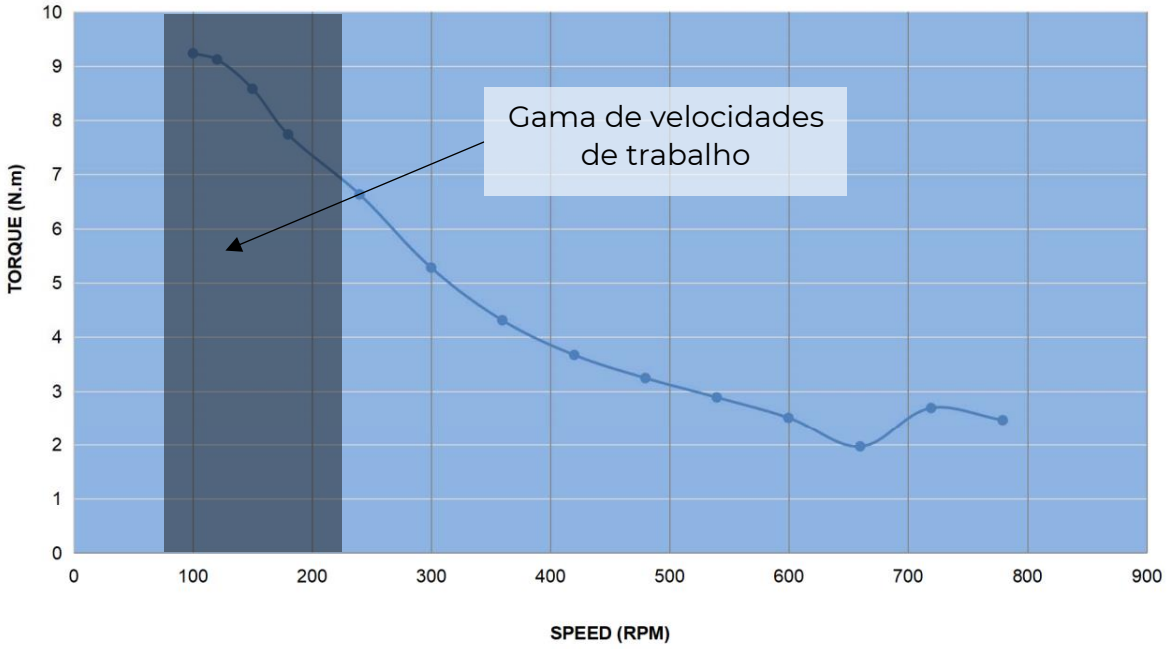
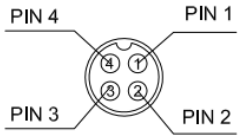


Figura 10 - Curva de binário do motor

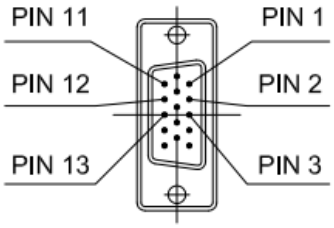
A Figura 11 ilustra as ligações dos enrolamentos do motor ao driver com ficha GX16 fêmea:



PIN 1	A+
PIN2	A-
PIN 3	B+
PIN 4	B-

Figura 11 - Conexões dos enrolamentos do motor

A Figura 12 ilustra as ligações do encoder ao driver da ficha DB15 macho:



PIN 1	EA+
PIN 2	VCC
PIN 3	GND
PIN 11	EB+
PIN 12	EB-
PIN 13	EA-

Figura 12 - Conexões do encoder ao driver

A Figura 13 representa o desenho técnico do motor:

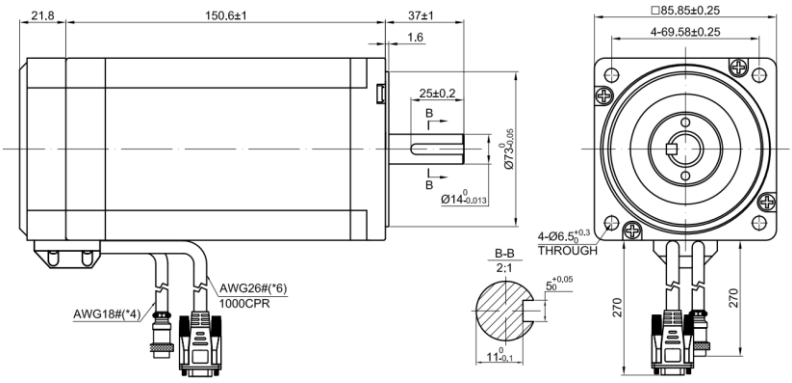


Figura 13 - Desenho técnico do motor

As características do driver são descritas na secção 3.3.

3. Esquema elétrico

Na elaboração do quadro elétrico são incluídos todos os componentes elétricos e eletrónicos responsáveis pelo funcionamento e controlo da máquina. São incluídos componentes de segurança de pessoas, bem como de proteção contra sobrecarga e curto-circuito.
O quadro é projetado para uma alimentação monofásica, de tensão alternada ~230V, 50 Hz, sendo esta a disponível nas tomadas em Portugal. A Figura 14 ilustra as interações entre os componentes.

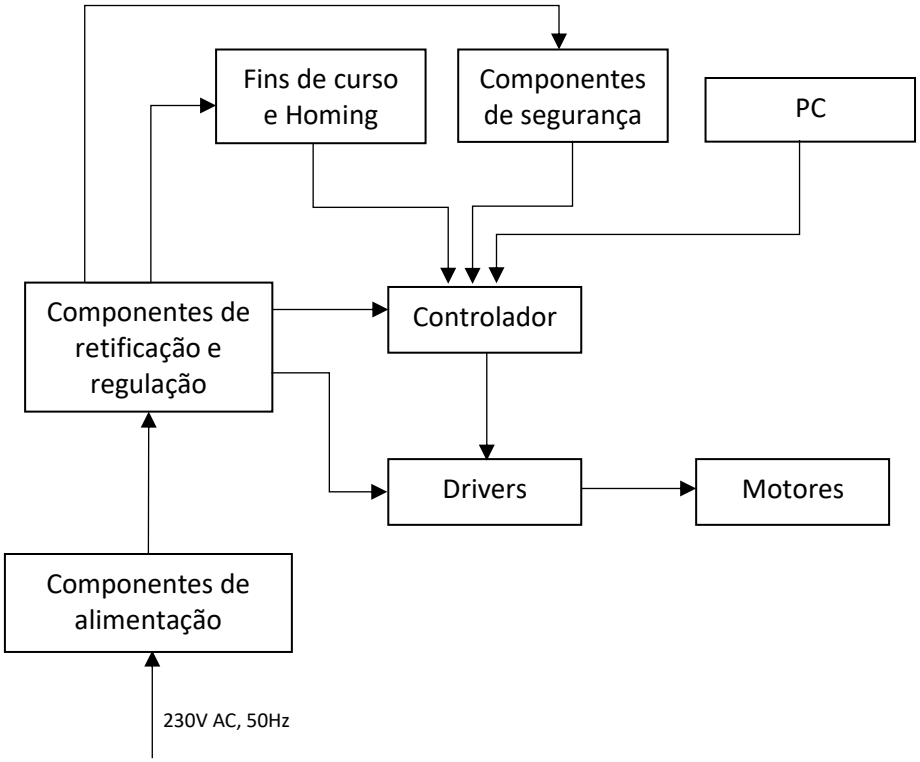


Figura 14 - Esquema de ligações entre componentes

Os componentes de potência, bem como os componentes de segurança, são selecionados após uma pesquisa de mercado, tendo em conta as suas características elétricas e a relação preço/qualidade. A seleção das fontes de alimentação é feita tendo em conta os requisitos dos motores. O controlador é escolhido com base no tipo de programação a implementar e no número e tipo de entradas e saídas necessárias.
O esquema elétrico completo com a designação de todos os componentes selecionados pode ser consultado no Anexo 1.

3.1 Circuito de potência

A energia é fornecida ligando uma ficha de tomada monofásica a uma ficha do tipo IEC C14 presente na lateral do quadro. Por sua vez esta passa por um seccionador geral de 3 polos 20A, onde só são utilizados dois polos. Um para a linha e outro para o neutro. Em série encontra-se um disjuntor diferencial de 25A, com proteção de pessoas de 30mA.
É incluída uma tomada auxiliar e respetivo disjuntor dentro do quadro caso seja necessário ligar algum equipamento externo que necessite de alimentação monofásica AC.
São também alimentados a 230V um sinal luminoso verde que acederá indicando que o circuito se encontra com tenção e uma ventoinha para resfriamento dos componentes no interior do quadro. O sinal luminoso é montado na porta frontal e a ventoinha na lateral. Estes últimos componentes são ligados em série com um fusível de 250V, 2A para proteção contra curto-circuito.
A alimentação segue para um conjunto de fontes de alimentação. Cada fonte de alimentação encontra-se ligada em séria com um disjuntor de 1 polo ligado na linha. Cada fonte de alimentação foi escolhida para o componente que vai alimentar. Optou-se por utilizar diferentes fontes de alimentação, uma para cada componente, em vez de utilizar uma de maior dimensão que alimentasse todo o circuito DC. Esta opção foi tida em conta após diálogo com o fornecedor dos motores e com a análise dos preços praticados para fontes com tensões e correntes de grande dimensão. Ao utilizar diferentes fontes de alimentação é também esperado uma maior estabilidade da corrente para os drivers e um menor aquecimento no interior do quadro. Assim tem-se um total de 5 fontes de alimentação, sendo quatro delas de 400W com saída de 48V DC e uma de 60W com saída de 24V DC. As fontes de 48V alimentam os diferentes drivers e a fonte de 24V alimenta o controlador, os componentes de segurança, fins de curso e sensores de homing. Para as fontes de alimentação de 48V é incluindo um componente de sinalização luminoso ligado em paralelo com a saída da fonte, que permite uma rápida análise do funcionamento da mesma. Para a fonte de 24V este componente não é necessário pois a fonte já traz um sinalizador luminoso incorporado. A Figura 15 e Figura 16 ilustram o esquema elétrico descrito.

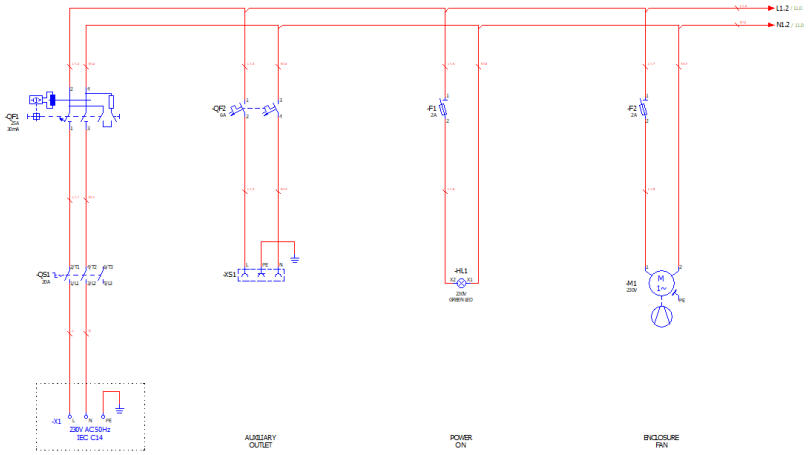


Figura 15 - Circuito de potência (parte 1)

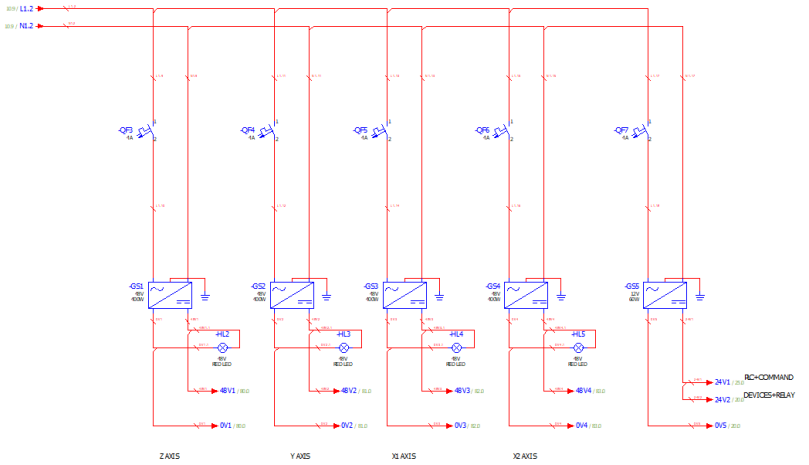


Figura 16 - Circuito de potência (parte 2)

3.2 Controlador

Na seleção do controlador a selecionar foi feito um levantamento das entradas e saídas necessárias. Foi também escolhido uma linguagem de programação onde se consiga implementar as equações da cinemática da máquina.
A analise do mercado levou a considerar diferentes soluções, desde soluções mais industriais, utilizando o software Mach 3 ou Mach 4, até soluções mais baratas e *open source* como o GRBL e o Universal Gcode Sender.
Das soluções encontradas, as mais interessantes são:

- CPU5A4E 4-axis Ethernet Software, utilizando como software o Eding CNC, comercializado pela DemonCNC;
- Acorn Centroid, utilizando como software o Acorn CNC, comercializado pela Acorn;
- Ethernet SmoothStepper, utilizando como software o Mach 3;
- Masso G3, utilizando como software o Masso CNC, comercializado pela Masso;
- Machifit NVCM3, utilizando como software o Mach 4;
- Raspberry Pi 4, utilizando como software o LinuxCNC;
- BlackBox Motion Control System, utilizando o firmware o GRBL, comercializado pela OpenBluids.

Com a detalhada análise destas opções, percebeu-se que a escolha devia assentar naquela onde a implementação das equações da cinemática fosse possível. Assim, e devido ao teor académico deste projeto, dissidiu-se utilizar uma opção baseada em programação C/C++, e utilizar o Arduino IDE como software de programação. Neste ambiente de programação é então possível implementar o sistema de equações que satisfazem o funcionamento da máquina, bem como todas as entradas e saídas necessárias para o seu funcionamento.
Existem várias opções de microcontroladores no mercado compatíveis com o ambiente de programação Arduino. Dentro deles foi ponderado o microcontrolador da Espressif Systems ESP 32, o microcontrolador Teensy 4.1 da PJRC, mas optou-se por uma solução mais robusta, apta a trabalhar em ambientes mais agressivos, com semelhanças a um PLC.
Optou-se então por analisar soluções de PLCs compatíveis com linguagem Arduino, como é o caso do Controllino. Esta marca apresenta PLCs de diferentes dimensões, aptos para trabalhar em ambientes industriais e compatível com diferentes softwares de programação como Arduino IDE, Atmel Studio, PlatformIO entre outros. Algumas das suas características são:

- Entradas digitais compatíveis com 5V e 24V;
- Saídas digitais de 5V ou 24V;
- Saídas a relé que suportam até 250V, 16A;
- Modulo de relógio RTC integrado;
- Comunicação Serial UART RS232;
- Comunicação RS 485, I^2C e SPI;
- Comunicação por USB e Ethernet.

Basta então determinar o número de entradas e saídas necessárias para que se possa escolher o modelo a selecionar da Controllino. A Tabela 3 contem o levantamento de entradas e saídas:

Sinal	Tipo	Quantidade
Botoeira de emergência	Entrada digital	1
Sinal da barreira	Entrada digital	1
Botoeira de rearm	Entrada digital	1
Home switch	Entrada digital	3
Alarme drivers	Entrada digital	4
Fins de curso	Entrada digital	3
Pulso drivers	Saída digital	4
Direção drivers	Saída digital	4
Enable drivers	Saída digital	1

Tabela 3 - Lista de entradas e saídas

Verifica-se a necessidade de um total de 13 entradas digitais e 9 saídas digitais.
Dos modelos apresentados pela Controllino, constata-se que o Controllino MAXI satisfaz os requisitos, contudo é selecionado o Controllino Mega para esta aplicação. O Controllino MEGA apresenta um maior número de entradas e saídas e a relação de preço quando comparado com o Controllino MAXI é vantajosa. As principais características do Controllino MEGA são:

- Microchip ATmega 2560;
- 256 kB de memória flash;
- 8 kB de SRAM;
- 4 kB de EEPROM
- 16 MHz de *clock rate*;
- 21 entradas digitais;
- 24 saídas digitais;
- 16 saídas a relé;
- Alimentação de 12V ou 24V.

Este PLC é então incluído no quadro elétrico, montado em trilha DIN, lado a lado com os restantes componentes. A sua alimentação é proveniente da fonte de 24V mencionada anteriormente e as suas entradas e saídas são conectadas diretamente aos componentes ou passando por bornes se necessário. Da porta USB do controllino mega é ligada uma extensão USB macho-fêmea de modo a tornar esta ligação acessível do exterior, sem que haja a necessidade da abertura da porta do quadro. A ligação USB fêmea é montada na porta frontal do quadro elétrico. A Figura 17 é uma representação real do controlador proposto para a aplicação. O esquema completo das ligações pode ser encontrado no Anexo 1.

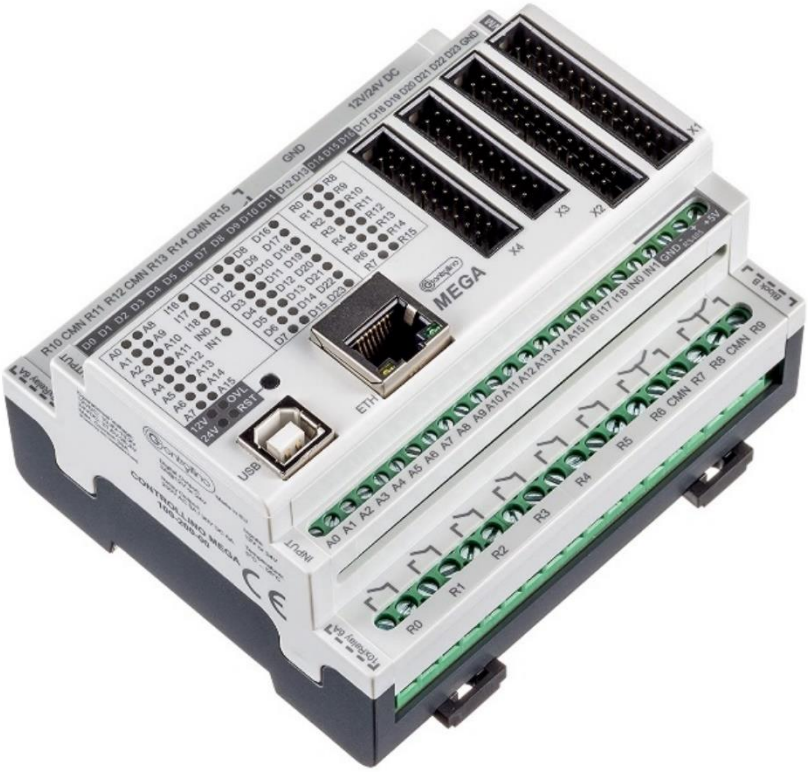


Figura 17 - Controllino Mega

3.3 Drivers

O driver é o componente responsável por movimentar o motor. Este recebe os sinais vindos do controlador, nomeadamente pulso e direção, e converte esses sinais para o motor. Cada pulso recebido pelo driver corresponde a um dado deslocamento angular por parte do motor e controlando a frequência de pulsos consegue-se controlar a velocidade do motor. O sinal da direção permite definir o sentido de rotação, consoante este sinal está ligado ou desligado.
Assim pode dizer-se que o driver faz a ligação entre o circuito de baixa tensão, vindo do controlador, e o circuito de alta tensão, vindo da fonte de alimentação e indo para o motor. É no driver que se define a quantidade de corrente que deve ir para o motor, de modo que este tenho binário suficiente e não sobreaqueça. É também o driver que controla o número de pulsos necessários para que o motor complete uma rotação completa. Este valor pode ir desde os 400 pulsos por revolução até aos 51200 pulsos por revolução. Com o aumento deste valor consegue-se aumentar consideravelmente a precisão no posicionamento, à custa de uma perca de binário por parte do motor.
Como referido anteriormente, os motores vão funcionar em malha fechada, e neste caso em concreto, o controlo dos sinais do encoder e feito pelo driver. Existem opções de motores em que os sinais do encoder vão direto para o controlador, mas aqui é o driver que é encarregue por processar estes sinais, libertando espaço de processamento no controlador. Esta capacidade do driver de gerenciar os sinais enviados pelo encoder, é possível devido à existência de um microcontrolador incorporado no driver com um firmware dedicado para este propósito. Este firmware tem em si um conjunto de parâmetros configuráveis, que podem ser acedidos conectando o driver ao computador. É necessário instalar um software dedicado no computador e ligar a porta RJ-45 do driver, utilizando um cabo para comunicação RS 232 específico. Neste software é possível configurar as características de controlo de malha fechada. Isto é feito utilizando um controlo PID tanto para o controlo da corrente como para o posicionamento exato do motor. Aqui pode-se ajustar os valores proporcional, integral e derivativo característicos deste tipo de controlo (PID). É ainda possível definir um conjunto de características como:

- Corrente consumida quando o motor está parado;
- O tipo de sinal das saídas do driver (*high(1)* ou *low(0)*);
- Ação a tomar para o sinal da entrada enable(ENA) do drive.

Com o driver ligado ao computador pode testar-se as configurações diretamente no motor, sendo possível simular velocidade e posicionamento com rampas de aceleração e desaceleração.
O driver possui duas saídas que podem ser ligadas a entradas do controlador. Uma é responsável pelo sinal de alarme em caso de falha de sincronismo entre o driver e o motor e a outra é responsável por avisar o controlador que o motor está em movimento.
Para o motor selecionado é fornecido uma tensão de 48V DC, sendo a corrente consumida pelo motor de 6A. O driver que melhor se enquadra nestes parâmetros é o driver modelo CL86T do mesmo fornecedor dos motores. Este driver permite o funcionamento com uma tensão de 20V a 80V DC e fornece uma corrente de pico de 8,2A. Possui entrada compatível com o encoder do motor e possui todas as entradas e saídas mencionadas anteriormente. A

Figura 18 ilustra as ligações entre os componentes, disponibilizada pelo fornecedor. Percebe-se que as entradas e saídas são ativadas pelo uso de fotodíodos e fototransístores.

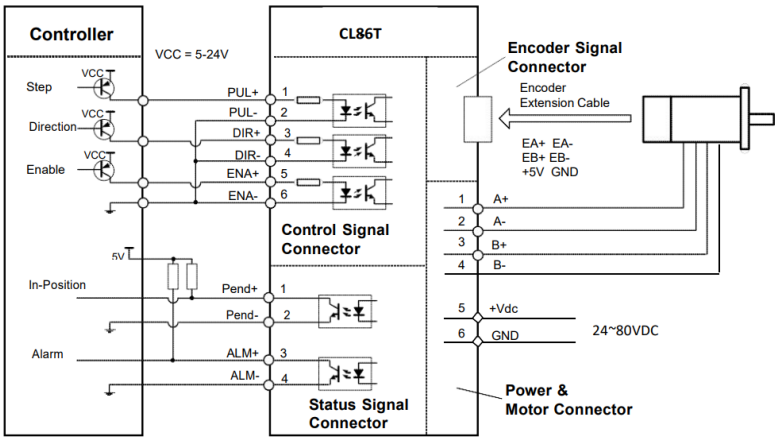


Figura 18 - Conexões entre controlador, driver e motor

Outras características do driver selecionado são:

- Frequência de pulse de entrada: 200kHz;
- Corrente média do sinal lógico: 7mA;
- Resistência do isolamento: 500MΩ;
- Peso: 580g.

No datasheet do driver é possível encontrar outras características como as suas dimensões e a configuração dos pinos da porta de comunicação RS 232.
São assim incluídos no quadro elétrico quatro drivers CL86T, cada um responsável pela movimentação de um motor. A Figura 19 é a representação real do driver proposto.

Figura 19 - Driver CL86T

3.4 Elementos de segurança

O processo de estampagem incremental é por norma efetuado a baixas velocidades de conformação, contudo, as forças utilizadas são de alguma grandeza. É assim importante ter em consideração o fator segurança, pois a utilização inadequada da máquina pode levar a lesões de alguma gravidade. O HRN (*Hazard Rating Number*) calcula o nível de risco que um equipamento ou máquina representa. É calculado tendo em conta 4 parâmetros tabelados que combinam a gravidade da lesão e a probabilidade de ocorrência. Tem-se:

- Possibilidade de ocorrência do perigo, designado por LO, aqui com o valor de 5 – *Casual, pode ocorrer*;
- Frequência de exposição ao perigo, designado de FE, aqui com o valor 5 – *Constante*;
- Perca máxima possível, designado de DPH, aqui com o valor 1 – *Quebra ou lesão dos membros menores*;
- Número de pessoas expostas, designado de NP, aqui com o valor 1 – *Uma a duas pessoas;*

$$HRN = LO * FE * DPH * NP = 25$$

O valor de 25 determinado para o HRN indica um nível de baixo, mas relevante de risco de lesão, por essa razão é estipulada o uso de uma barreira imaterial na zona de acesso lateral da máquina. Esta barreira será acionada sempre que a máquina esteja ligada e algo aceda á zona de conformação. O seu restante perímetro pode ser envolto de uma parede de acrílico ou vedação de metal. Outra solução podia ser o uso de uma vedação no perímetro total, onde a zona lateral seria acedida por uma porta com *interlock*. Em ambos os casos, é gerado um sinal elétrico caso haja acesso à zona de conformação da máquina. Este sinal é usado para acionar o sistema de segurança para que tome uma ação imediata.
É aqui proposto o uso de uma barreira imaterial, com um comprimento de 150 mm, montada verticalmente e com uma tensão de alimentação de 24V. Esta barreira é composta por dois elementos, um recetor e um emissor e possui um sinal OSSD (*Output Signal Switching Device*) a ser ligado a um relé de segurança.
O relé de segurança deve conter um total de 2 entradas de acionamento vindas por parte da barreira imaterial e da botoeira de emergência e uma entrada de rearm feita pelo acionamento de uma botoeira. O relé vai então acionar a bobina auxiliar de um conjunto de 4 contatores e um e de um relé de 24V. Os contatores vão permitir a passagem de corrente dos drivers para os motores, ou seja, com a energia ligada os contatores encontram-se atracados e só vão abrir no caso de o relé de segurança ser acionado. O relé de 24V encontra-se da mesma forma acionado, e só abre no caso de falta de energia na sua bobina. Este relé tem como função acionar o travão do motor do eixo Z, que se encontra travado sempre que não está energizado.
Por outro lado, são utilizados dois sinais A1 e A2, levadas para entradas do controlador, acionadas pela botoeira de emergência e pela botoeira de rearm,

respetivamente. Estas botoeiras podem estar fixas à porta do quadro elétrico, ou podem estar incorporados num comando móvel.
O sinal A0 pertence igualmente ao circuito de emergência, mas este é gerado pelo relé de segurança. São aqui usados sinais diferentes para que o controlador possa distinguir quando é pressionada a botoeira de emergência ou quando a barreira é atravessada. O esquema de ligações do relé de segurança escolhido encontram-se na Figura 20.

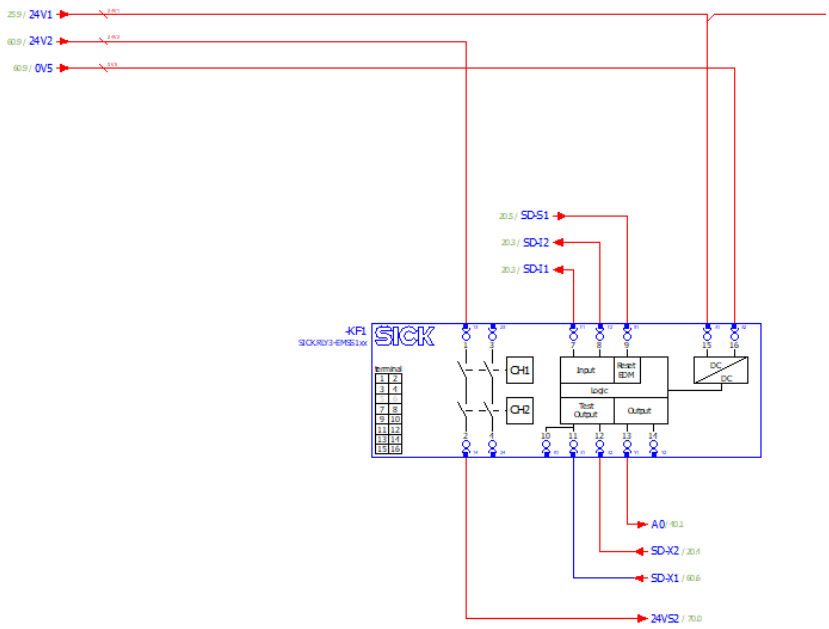


Figura 20 - Relé de segurança

A Figura 21 é uma representação real do relé de segurança proposto.

Figura 21 - Relé de segurança proposto

A Figura 22 representa o esquema de ligações da barreira imaterial.

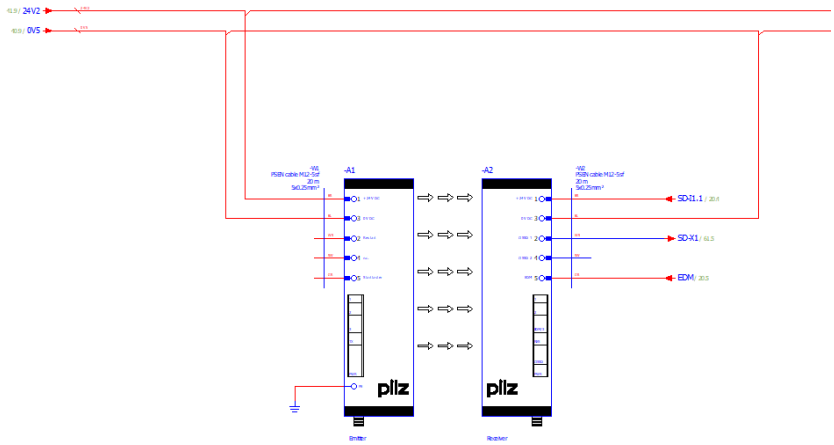


Figura 22 - Barreira imaterial

A Figura 23 é uma representação real da barreira imaterial proposta.

Figura 23 - Barreira imaterial proposta

3.5 Modelo 3D do quadro elétrico

A elaboração do modelo 3D do quadro elétrico teve em conta as dimensões reais de todos os componentes. Grande parte dos fornecedores disponibilizam os ficheiros CAD dos componentes selecionados e os restantes foram dimensionados com base nos desenhos técnicos dos mesmos. Utilizou-se o software SolidWorks para a montagem e o software KeyShot para a renderização.
A disposição dos componentes foi escolhida admitindo a entrada da energia no canto superior esquerdo do quadro e juntando os componentes do mesmo tipo por zonas.

- Zona 1: Entrada de potência e componentes de proteção;
- Zona 2: Drivers e contatores de acionamento dos motores;
- Zona 3: Bornes, relé de segurança, sinais luminosos e controlador;
- Zona 4: Fontes de alimentação.

O quadro está envolto por calha de plástico de modo a facilitar a passagem dos condutores de ligação dos componentes. Os componentes estão montados em calha DIN ou aparafusados diretamente na chapa de montagem.

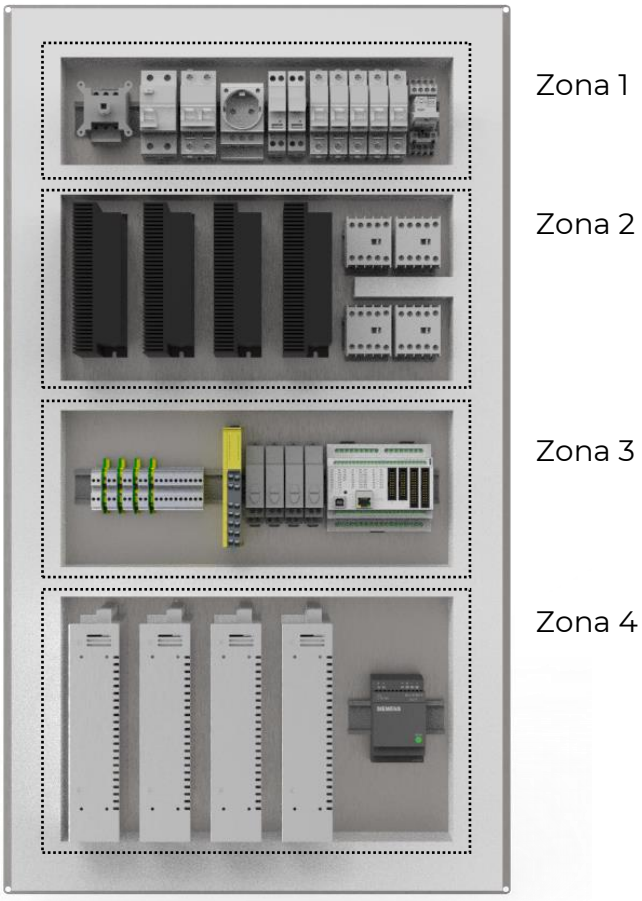


Figura 24 - Quadro elétrico (vista frontal)

Figura 25 - Quadro elétrico

Na lateral do quadro são incorporadas 4 fichas GX16 de 4 pinos para ligação entre os drivers e os motores, 4 fichas DB15 de 3 linhas e uma ficha IEC C14.

Figura 26 - Quadro elétrico completo

4. Implementação da cinemática

Na implementação da cinemática para o controlo da máquina de estampagem incremental, é preconizado o controlo da extremidade da ferramenta de conformação. O objetivo é movimentar esta ferramenta para uma coordenada espacial (*x, y, z*) definida, controlando a aceleração e desaceleração do movimento. É admitido que a ferramenta pode deslocar-se 80 milímetros em Z, 220 milímetros em Y e 200 milímetros em X. O ponto 0,0,0 é definido no centro da máquina e admite-se um deslocamento negativo. Assim tem-se:

- Eixo X pode variar de [-100; +100];
- Eixo Y pode variar de [-110; +110];
- Eixo Z pode variar de [-40; +40].

Pretende-se que este deslocamento seja feito de forma síncrona, ou seja, todos os motores devem acabar o seu movimento no mesmo instante, independentemente do seu deslocamento. Pretende-se ainda que o movimento descreva uma linha reta no deslocamento entre pontos.

4.1 Determinação do número de pulsos

O controlo do eixo Z e Y é baseada num braço RR plano cujas equações da cinemática inversa para o cálculo dos ângulos para o caso específico são:

$$\theta_2 = -arccos\frac{(120-y)^2 + (200-z)^2 - L_1^2 - L_2^2}{2L_1L_2};$$

$$\theta_1 = arctan\left[\frac{(200-z)(L_1 + L_2\cos\theta_2) - xL_2 sin\theta_2}{(120-y)(L_1 + L_2\cos\theta_2) - yL_2 sin\theta_2}\right];$$

As unidades θ_1 e θ_2 são calculadas em radianos, mas são facilmente convertidas para graus. Para os comprimentos do braço, L_1 e L_2, as dimensões adotadas são de 120 mm e 200 mm, respetivamente. A base do eixo Z é colocada a uma distância de 200 mm de altura, conforme referido na secção 1.2. Uma representação gráfica das dimensões do RR plano encontra-se na Figura 27.

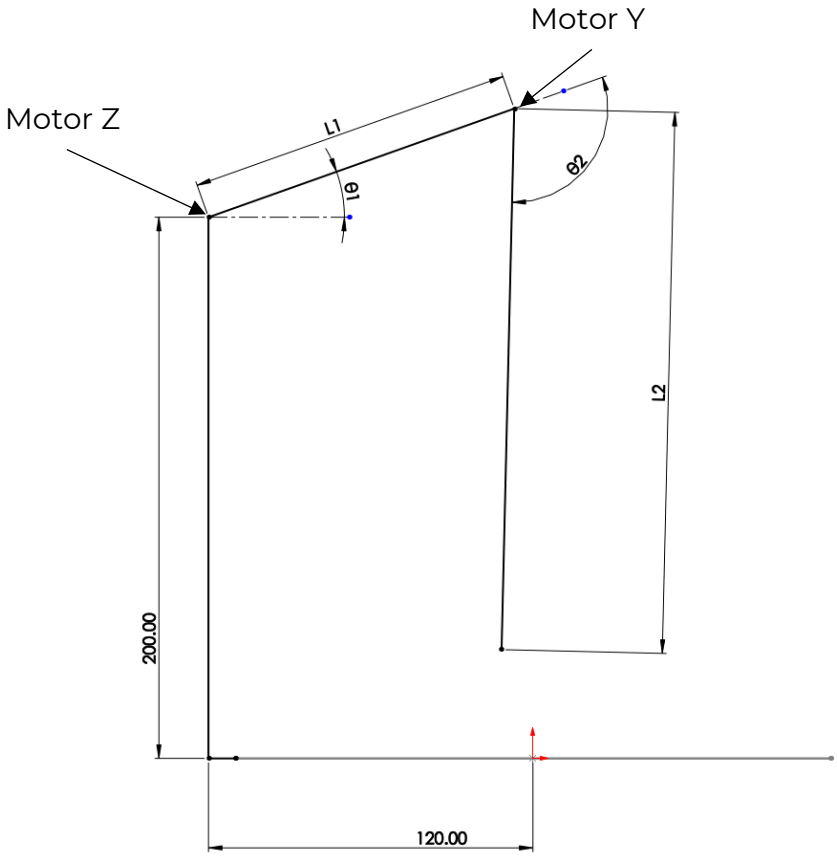


Figura 27 - Representação RR plano

Sabendo o valor dos ângulos é possível determinar o deslocamento angular que deve ser feito pelo motor para este se deslocar da posição atual para a posição desejada. Sabendo o número de pulsos para uma revolução pode calcular-se o número de pulsos necessários. Admite-se a utilização de uma caixa redutora entre a saída do motor e a ligação ao braço de 100:1, assim percebe-se que a rotação do motor deve ser 100 vezes maior que o deslocamento do braço. O número de pulsos para o motor Z e para o motor Y é calculado por:

$$nº\ de\ pulsos\ motor\ Z = \theta_1\ [deg] * redução * \frac{pulsos/rev}{360º};$$

$$nº\ de\ pulsos\ motor\ Y = \theta_2\ [deg] * redução * \frac{pulsos/rev}{360º};$$

O eixo X é controlado pela movimentação de dois fusos coordenados. É admito o uso de um fuso de passo 4 milímetros e é utilizado o mesmo sinal do controlador para ambus os drivers, ou seja, trata-se o eixo X como se fosse controlado por um

único motor. A equação utilizada no cálculo do número de pulsos para o motor, admitindo também uma redução de 100:1 é:

$$nº\ de\ pulsos\ motor\ X = (x_f - x_i) * \frac{pulsos/rev}{passo};$$

Com estas equações é possível determinar o número de pulsos que o controlador deve enviar para cada motor.

4.2 Controlo da aceleração e desaceleração

Para implementar uma aceleração no movimento dos motores de passo deve-se variar o intervalo de tempo entre cada pulso, partindo de um valor inicial pré-determinado, reduzindo este valor até um intervalo de tempo entre pulsos mínimo, sendo este a velocidade máxima do movimento. A partir deste momento, pretende-se que a velocidade seja constante até que entre no período de desaceleração parando na posição final. Neste período o intervalo de tempo entre pulsos aumenta até o valor inicial.

O cálculo do valor inicial, ou seja, o primeiro intervalo entre pulsos, pode ser estimado. Sabe-se que a velocidade, v, pode ser calculada em função da aceleração, α, e do tempo, t, por:

$$v = \int \alpha \, dt = \alpha t;$$

O deslocamento, s, é calculado por:

$$s = \int vt \, dt = \frac{1}{2}\alpha t^2;$$

Por outro lado, o deslocamento de um motor de passo também é calculado em função do número de pulsos, n, e do ângulo/pulso, β, por:

$$s = n\beta;$$

Assim igualando as duas expressões anteriores, pode determinar-se um tempo para uma determinada posição tem-se:

$$\frac{1}{2}\alpha t^2 = n\beta <=> \mathrm{t} = \sqrt{\frac{2n\beta}{\alpha}}$$

A partir da expressão anterior é possível determinar uma expressão para o cálculo do intervalo entre dois pulsos consecutivos. Tem-se:

$$c_n t_n = t_{n+1} - t_n = \sqrt{\frac{2\beta}{\alpha}}\left(\sqrt{n+1} - \sqrt{n}\right);$$

A partir daqui pode-se determinar uma expressão que permite calcular um valor para os intervalos entre pulsos. Estes valores começam num valor inicial e os restantes valores dependem do anterior. Tem-se:

$$c_n = \frac{1}{t_n}\sqrt{\frac{2\beta}{\alpha}}\left(\sqrt{n+1}-\sqrt{n}\right);$$

Por fim o valor inicial, c_0, é calculado por:

$$c_0 = \frac{1}{t_n}\sqrt{\frac{2\beta}{\alpha}};$$

E o valor de c_n:

$$c_n = c_0\left(\sqrt{n+1}-\sqrt{n}\right);$$

O cálculo dos diferentes valores de c_n, são feitos por parte do controlador, e de forma a reduzir espaço de computação no cálculo de duas raízes quadradas, é usado uma aproximação para este valor com menor complexidade matemática. Esta aproximação foi obtida usando uma serie de Taylor, aqui é somente apresentada a expressão final:

$$c_n = c_{n-1} - \frac{2c_{n-1}}{4n+1};$$

Esta expressão introduz um erro de 0,44 para n=1, mas por outro lado permite um cálculo mais rápido por parte do controlador. Uma maneira de compensar este erro consiste em multiplicar c_0 por 0,676 (Atmel, 2006).

Com base na expressão anterior, é possível criar um código cíclico, onde é calculado um novo valor para c_n partindo de um valor inicial c_0. O limite definido para este valor consiste no intervalo de tempo mínimo entre pulsos. Estes valores são afixados num *array* que é inserido sequencialmente nos intervalos entre pulsos. Em forma crescente quando o motor está em aceleração, e em forma decrescente em desaceleração.

4.3 Posicionamento incremental

Recorrendo às equações da cinemática é possível determinar o movimento angular, e por sua vez o número de pulsos que o motor deve executar para posicionar a ferramenta nas coordenadas desejadas. Ainda assim, o movimento conjunto dos motores, leva a que a ferramenta não descreve um movimento retilíneo, mas sim descreva um arco.
A maneira de contornar este comportamento, consiste em movimentar a ferramenta entre posições sucessíveis, recalculando a nova posição com base na posição anterior, utilizando um valor incremental para o deslocamento, até que a ferramenta atinja a posição final. Para uma dada posição final X_f, Y_f e Z_f, partindo de uma posição inicial X_0, Y_0 e Z_0, é possível calcular um vetor deslocamento no plano YZ:

$$vetor = \sqrt{(\Delta Y)^2 + (\Delta Z)^2} = \sqrt{(Y_f - Y_0)^2 + (Z_f - Z_0)^2};$$

Pode também calcular-se o ângulo do vetor deslocamento com a horizontal, δ por:

$$\delta = \begin{cases} arctan\dfrac{\Delta Z}{\Delta Y} = arctan\left(\dfrac{Z_f - Z_0}{Y_f - Y_0}\right), & Y_f > Y_0 \\ arctan\dfrac{\Delta Z}{\Delta Y} = arctan\left(\dfrac{Z_f - Z_0}{Y_f - Y_0}\right) + \pi, & Y_f \leq Y_0 \end{cases}$$

Este valor depende se o braço se encontra em avanço ou recuo, assim depende da posição final Y_f.
Encontra-se uma representação esquemática na Figura 28:

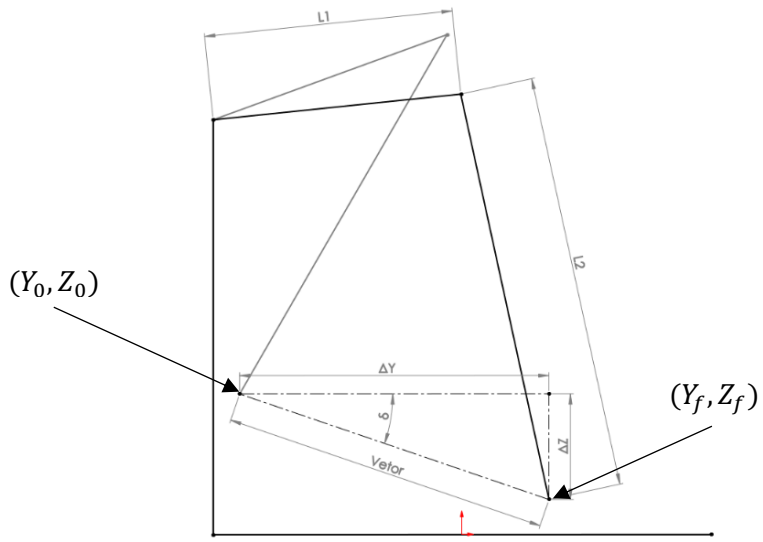


Figura 28 - Posicionamento incremental

Admitindo um valor para o incremento mínimo entre posições, Δs, pode escrever-se o seguinte sistema de equações para as posições dos eixos Y e Z:

$$\begin{cases} \Delta Y_i \\ \Delta Z_i \end{cases} = \begin{cases} Y_0 + i * \Delta s * \cos\delta \\ Z_0 + i * \Delta s * \operatorname{sen}\delta \end{cases};$$

Os valores das sucessivas posições de ΔY_i e ΔZ_i são afixados num *array* com um comprimento igual ao número total de incrementos, N_i, dado em função do tamanho do vetor deslocamento:

$$N_i = \frac{vetor}{\Delta s};$$

O posicionamento do eixo X é independente dos restantes movimentos, contudo pretende-se que os movimentos sejam coordenados entre os 3 eixos. A abordagem para este eixo consiste em pegar no número de incrementos que os eixos Y e Z vão executar e achar um valor para o incremento de ΔX em função da posição final, X_f e da posição inicial, X_0.

$$\Delta s_x = \frac{\Delta X}{N_i} = \frac{X_f - X_0}{N_i};$$

O cálculo das sucessivas posições de ΔX_i é:

$$\Delta X_i = X_0 + i * \Delta s_x;$$

Do mesmo modo, o valor das sucessivas posições de ΔX_i é afixado num *array* com o comprimento de N_i.
O fluxograma da Figura 29 representa as etapas de cálculo efetuadas pelo controlador no posicionamento da ferramenta:

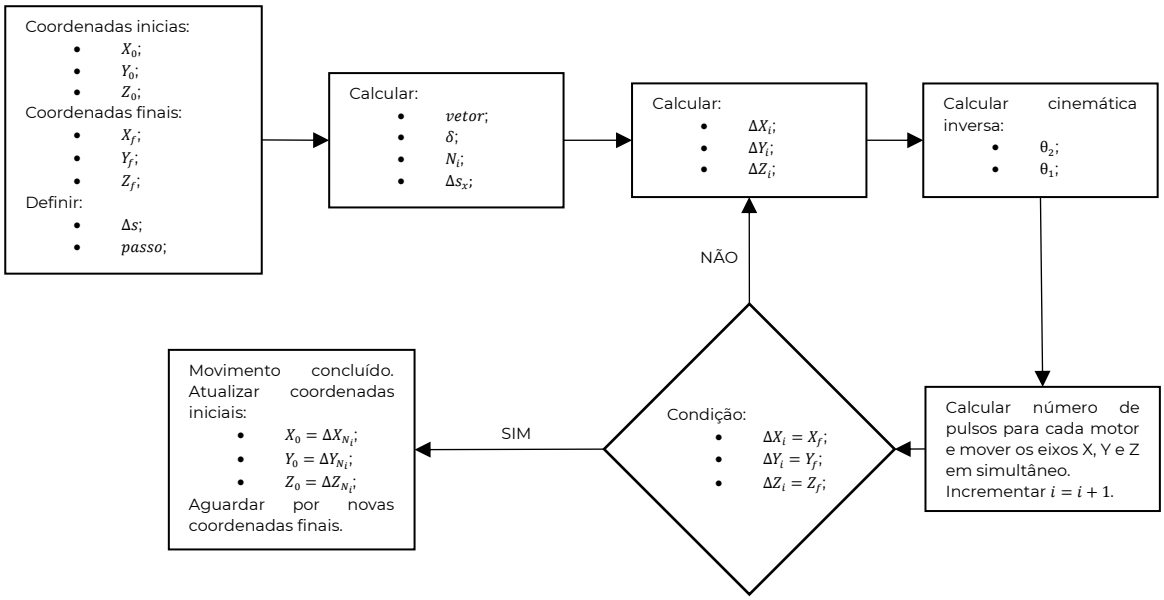


Figura 29 - Fluxograma de posionamento incremental

A implementação da aceleração e desaceleração dos motores é feita em simultâneo com o incrementar de i. Contudo o comprimento do *array* contendo os valores dos intervalos entre pulsos referidos anteriormente é de $N_i/2$. Deste modo consegue-se garantir que o motor entra em desaceleração mesmo quando os movimentos são curtos, ou seja, mesmo que o motor não chegue a atingir a velocidade máxima i.e. mínimo intervalo entre pulsos.

4.4 Projeto experimental

Como referido na seleção do controlador, a programação utilizada é baseada em C++ e o software utilizado é o Arduino IDE. De modo a efetuar uma simulação do funcionamento da máquina de forma aproximada, é utilizado um Arduino MEGA para implementação e teste do código. Apesar de existirem bastantes diferenças no uso do Controllino MEGA em relação ao uso do Arduino Mega, nomeadamente na atribuição das saídas e entradas analógicas e digitais, a implementação das rotinas de posicionamento são iguais. Em conjunto, são também utilizados 3 motores de passo bipolares e 3 drivers, cada um representando um dos eixos X, Y e Z. Tanto os motores como os drivers são diferentes dos selecionados na secção 3, contudo o funcionamento é semelhante, ambus utilizam pinos de pulso e direção. A alimentação dos drivers é feita por parte de uma fonte de alimentação de 12V DC, já o Arduino é alimentado pela sua porta USB.
São utilizados dois NEMA 23 para os eixos Y e Z, e um NEMA 11 para o eixo X. A documentação dos componentes apresenta as seguintes características:

NEMA 23	NEMA 11
Eixo Y e Z	Eixo X
1,85 Nm	0,095 Nm
2,80 A/fase	0,67 A/fase
1.8 °/pulso	1.8 °/pulso

Tabela 4 - Características dos motores de passo

Em conjunto utilizam-se os drivers TB6600:

Tensão de alimentação	9-42 V DC
Resolução	200 a 6400 Pulsos/Revolução
Corrente de pico	4 A
Frequência de pulso máxima	20 kHz

Tabela 5 - Características driver TB6600

Em semelhança aos drivers propostos na secção 3.3, no corpo deles existem um conjunto de seletores que permitem fazer o ajuste de corrente e de pulsos por revolução. A sua parametrização é feita com a consulta das tabelas gravadas na lateral do driver, Figura 30, aqui para o motor NEMA 11. Apesar da corrente por fase do motor ser de 0.67 A, o driver é parametrizado para uma corrente de 0.5 A. Deste modo evita-se o sobreaquecimento do motor, contudo à custa de perda de binário. Para os restantes drivers é utilizado a mesma abordagem, ou seja, ambus os drivers são parametrizas para uma corrente de 2 A, apesar dos motores aceitarem 2.8 A. Para todos os drivers é utilizado uma resolução de 200 pulsos por revolução.

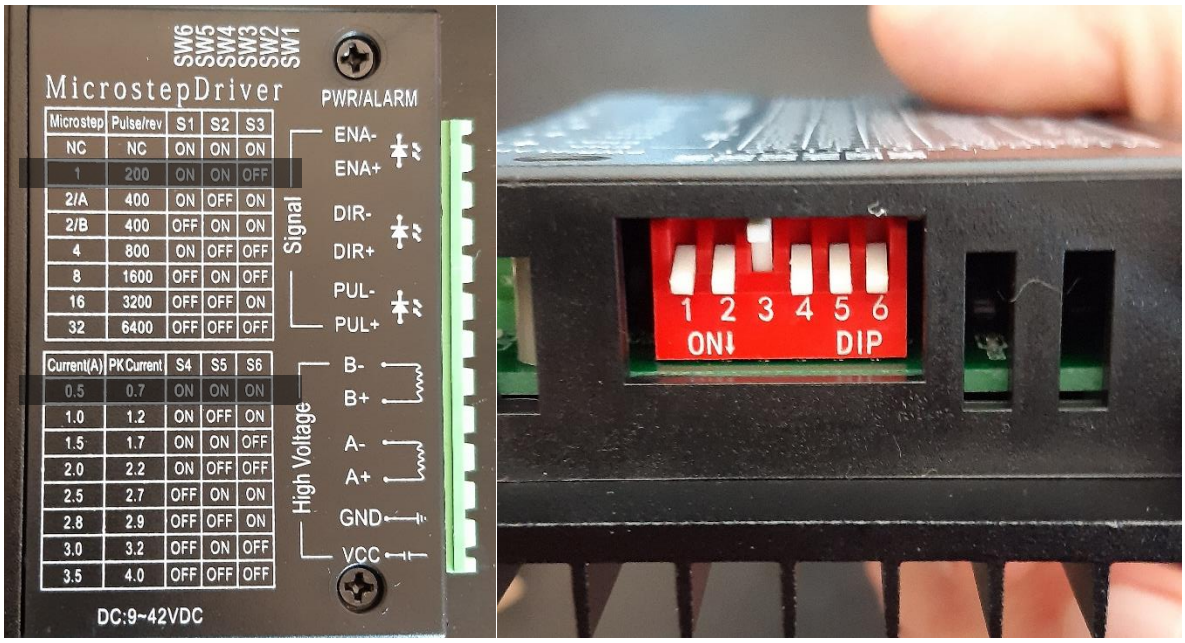


Figura 30 - Driver TB6600

A fonte de alimentação dos drivers e motores tem as seguintes características:

Modelo	S-201-12
Tensão de saída	12 V DC
Corrente de saída	16,5 A
Tensão de entrada	115 V / 230 V AC com seletor
Corrente de entrada	3.8 A / 2.0 A AC

Tabela 6 - Características da fonte de alimentação

São utilizados 3 fins de curso mecânicos, ligados ao Arduino Mega, que são usados como sensores limitadores de movimento, possibilitando a implementação de uma rotina de *homing*.
A montagem de todos os componentes numa banca de ensaio encontra-se na Figura 31 e 32. Todas as ligações são feitas com base na documentação dos componentes. As ligações ao Arduino dos sinais de pulso e direção utilizam os pinos digitais 2 a 7, o sinal *enable* para ativação dos drivers utiliza o pino 8 e os fins de curso utilizam os pinos digitais 18 a 20. Estes pinos têm a característica de funcionar como *hardware interrupt,* o que possibilita a criação de uma rotina de segurança de maior fiabilidade em caso de falha na movimentação dos motores. Estes sinais são do tipo normalmente fechado e utilizam uma resistência de pull-down de 3.3kΩ, assim o sinal no pino é sempre de aproximadamente 5V e vai a 0V quando o sensor (fim de curso) é ativado. Esta configuração permite detetar caso existam condutores desligados nos fins de curso.

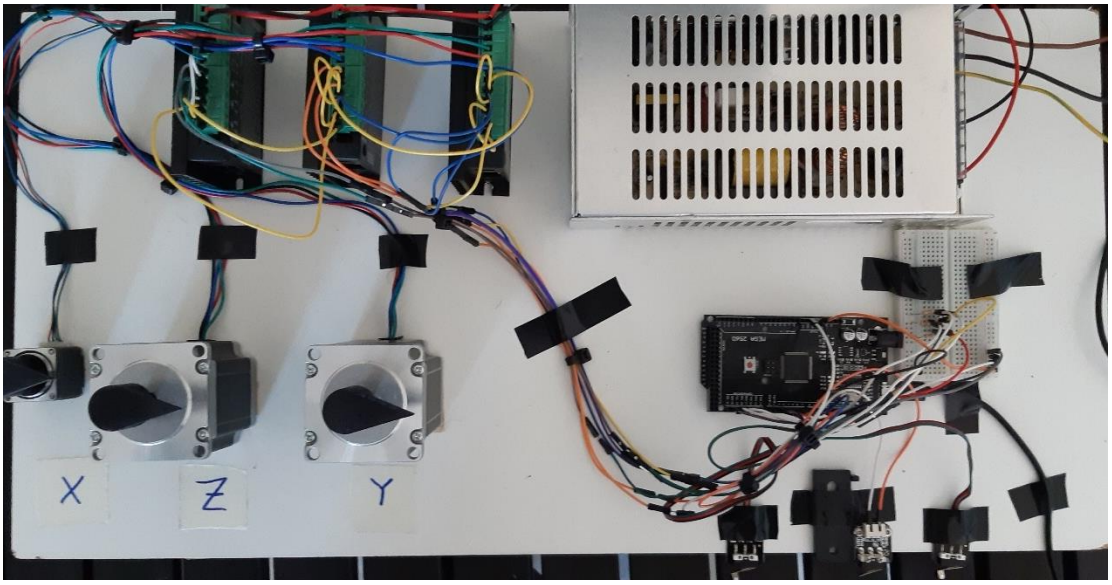


Figura 31 - Montagem em bancada 1

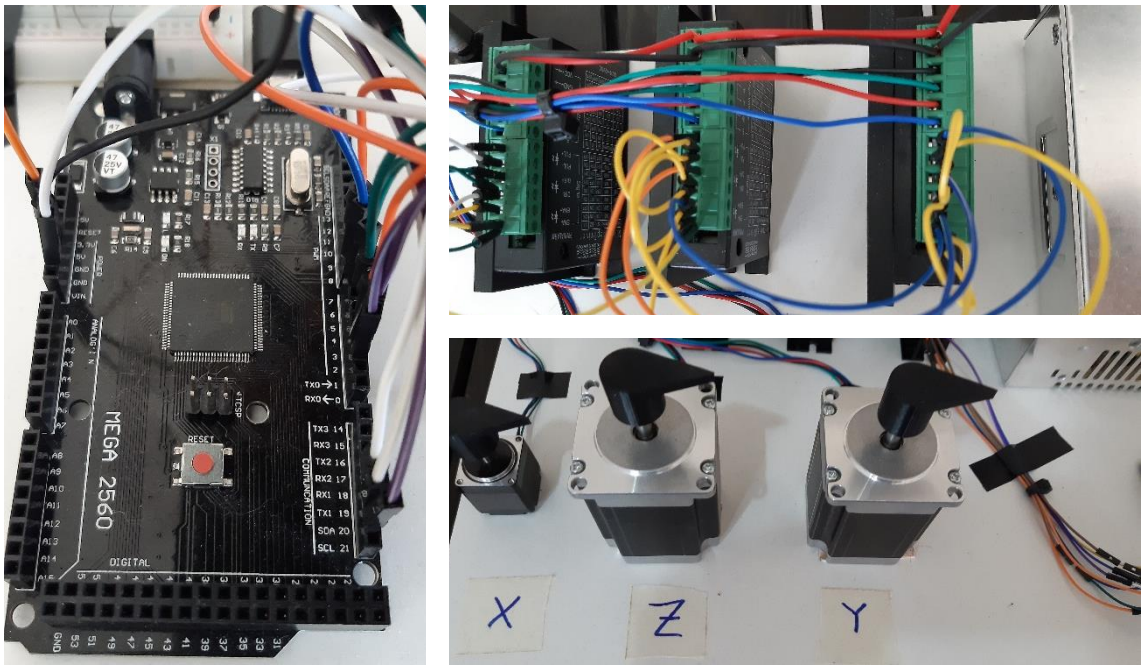


Figura 32 - Montagem em bancada 2

4.5 Implementação do controlo numérico

Numa primeira fase, foi feito o controlo dos motores de passo de uma forma simples, gerando sinais de 0 V ou 5 V com um intervalo de tempo definido em microssegundos nos pinos do controlador ligados à entrada PUL+ do driver.
De seguida introduziu-se uma rotina de leitura de dados recebidos pela porta de comunicação RS 232 do Arduino. Esta porta vai ficar sempre à espera de mensagens, e vai executar um comando consoante a mensagem que receber.
Começou-se por enviar pela porta de comunicação o número de pulsos pretendidos, e foram feitos alguns testes para determinar as velocidades máximas dos motores, ou seja, o mínimo tempo entre pulsos em microssegundos.
Por esta altura verificou-se que a abordagem que se estava a implementar, não garantia o funcionamento dos motores em simultâneo, nem a paragem de ambus numa determinada coordenada x, y, z. Verificou-se que o motor com menor distância atingia o ponto final mais rápido que os restantes motores, deste modo, devia definir-se as acelerações e velocidades dos motores para sincronizar este movimento.
Depois de investigação sobre o tópico, decidiu-se utilizar uma biblioteca para o Arduino IDE, a biblioteca *AccelStepper*. Esta biblioteca é bastante conhecida no que toca ao controlo de motores de passo e apresenta um vasto leque de classes para diferentes aplicações. Sobretudo, permite definir uma posição para os motores atribuindo um determinado número de pulsos. Com esta característica pode implementar-se uma rotina de *homing* para calibrar a posição dos motores e movimentar para as coordenadas desejadas a partir desta última. Tendo isto, todas as posições seguintes serão conhecidas de forma exata. Caso haja perda de passos, o encoder do motor vai detetar e avisar o driver que por sua vez informa o controlador.
Nesta biblioteca os diferentes motores são enumerados como objetos do tipo *AccelStepper*, atribuindo o tipo de motor (unipolar, bipolar, etc.), e os pinos de pulso e direção [Figura 33].

```
//bibliotecas
#include <AccelStepper.h>
#include <MultiStepper.h>
#include <math.h>

//pinos motores
#define PulY 2 //pino pulso Y
#define DirY 3 //pino direção Y
#define PulZ 4 //pino pulso Z
#define DirZ 5 //pino direção Z
#define PulX 6 //pino pulso X
#define DirX 7 //pino direção X
#define Ena 8 //pino enable para X, Y, Z

//pinos fim de curso - pinos com hardware interrupt
#define LSwitchY 18 //fim de curso ligado em NC com pull down
#define LSwitchZ 19 //fim de curso ligado em NC com pull down
#define LSwitchX 20 //fim de curso ligado em NC com pull down

//objetos dos motores
AccelStepper Yaxis(1, PulY, DirY); //motor Y
AccelStepper Zaxis(1, PulZ, DirZ); //motor Z
AccelStepper Xaxis(1, PulX, DirX); //motor X

//objeto para controlo conjunto
MultiStepper SYNCMovXYZ;
```

Figura 33 - Iniciação de objetos

Para o movimento conjunto dos motores recorre-se a classe *MultiStepper*. Esta classe aceita objetos do tipo *MultiStepper*, onde permite agrupar objetos do tipo *AccelStepper* com a função *addStepper*.
Na Figura 33 estão também definidos os pinos dos fins de curso que têm a característica de funcionarem como *hardware interrupt*. No Arduino mega os pinos com esta característica são os pinos 2, 3, 18, 19, 20 e 21. Aqui utilizam-se os pinos 18, 19 e 20. A implementação do *hardware interrupt* é feita na função *setup()*. Aqui é adicionada a função *attachInterrupt()* que tem como argumentos o pino, a função que deve ser chamada e o modo como a função é chamada, seja na descida (*FALLING*), subida (RISING) ou alteração (CHANGE) da extremidade de um sinal. Neste caso a função é chamada na descida.

```
void setup() {
  //iniciar o monitor serial
  Serial.begin(9600);

  //iniciar os pinos dos fins de curso como entradas
  pinMode(LSwitchY, INPUT);
  pinMode(LSwitchZ, INPUT);
  pinMode(LSwitchX, INPUT);

  //hardware interrupts (pino, função, modo)
  attachInterrupt(digitalPinToInterrupt(LSwitchY), HardLimitsFault, FALLING);
  attachInterrupt(digitalPinToInterrupt(LSwitchZ), HardLimitsFault, FALLING);
  attachInterrupt(digitalPinToInterrupt(LSwitchX), HardLimitsFault, FALLING);

  //iniciar o pino enable dos drivers como saída
  pinMode(Ena, OUTPUT);
  digitalWrite(Ena, HIGH); //começar com os motores desligados

  //agrupar os motores para posicionamento conjunto
  SYNCMovXYZ.addStepper(Yaxis);
  SYNCMovXYZ.addStepper(Zaxis);
  SYNCMovXYZ.addStepper(Xaxis);
}
```

Figura 34 - Função setup

Após a função *setup()*, na função *loop()* apenas são chamadas a função de leitura de dados pela porta de comunicação, a função de posicionamento, função de calibração e função de sequência de movimentos. Estas função são dependentes de diferentes *flags*.

```
void loop() {
  msgIn();
  IncPos();
  Homing();
  Move();
}
```

Figura 35 - Função loop

A função *msgIn()* tem como *flag* de entrada a chegada de bits à porta de comunicação. Os valores recebidos são concatenados numa *string* chamada msn_IN que é depois avaliada. Consoante o seu conteúdo o programa vai efetuar diferentes ações [Figura 36]:

- H – Executar rotina de calibração, *homing*;

- ENA – ligar os motores;
- DIS – desligar os motores;
- MOV – movimentos em sequência;
- SAV – guardar posições de sequência;
- ACC+ - ativar aceleração;
- ACC- - desativar aceleração.

```
void msgIn() {
  while (Serial.available()) {
    delay(3);  //delay para preencher a string
    if (Serial.available() > 0) {
      char c = Serial.read();
      msg_IN += c;
    }
  }
  if (msg_IN.length() > 0) {
    if (msg_IN == "H") {
      HomingRotine = true;
      msg_IN = "";
    } else if (msg_IN == "ENA") {
      digitalWrite(Ena, LOW); // ligar motores
      msg_IN = "";
    } else if (msg_IN == "DIS") {
      digitalWrite(Ena, HIGH); // desligar motores
      msg_IN = "";
    } else if (msg_IN == "MOV") {
      MoveArm = true;
      msg_IN = "";
    } else if (msg_IN == "ACC+") {
      WithACCDCC = true;
      msg_IN = "";
    } else if (msg_IN == "ACC-") {
      WithACCDCC = false;
      msg_IN = "";
```

Figura 36 - Leitura de mensagem recebida (parte 1)

Se a mensagem recebida for SET, os seus valores seguintes vão ser convertidos para variáveis do tipo *integer* e atribuídos aos valores da velocidade (vel), intervalo máximo entre pulsos (acc) e intervalo mínimo entre pulsos (MINdelay). Após a atribuição dos valores às variáveis, o programa responde de volta com os valores atuais das variáveis. Deste modo é possível garantir que os valores foram corretamente transmitidos. [Figura 37]

```
} else if (msg_IN.startsWith("SET")) {
  int index = msg_IN.indexOf(" "); //localizar o espaço (" ")
  vel = atol(msg_IN.substring(3, index).c_str()); //atribuir velocidade
  msg_IN = msg_IN.substring(index + 1); //eliminar texto lido
  index = msg_IN.indexOf(" "); //localizar o espaço (" ")
  acc = atol(msg_IN.substring(0, index).c_str()); //atribuir aceleração
  msg_IN = msg_IN.substring(index + 1); //eliminar texto lido
  index = msg_IN.indexOf(" "); //localizar o espaço (" ")
  MINdelay = atol(msg_IN.substring(0, index).c_str()); //atribuir min delay
  msg_IN = "";
  //resposta de confirmação
  Serial.print("VEL");
  Serial.println(vel);
  delay(100);
  Serial.print("ACC");
  Serial.println(acc);
  delay(100);
  Serial.print("MDL");
  Serial.println(MINdelay);
```

Figura 37 - Leitura de mensagem recebida (parte 2)

Por último, se não se cumprir nenhuma das condições anteriores, significa que os dados recebidos representam coordenadas de posicionamento. Estas valores são igualmente concatenados na *string* msn_IN, convertidos para *integer* e atribuídos a variáveis do tipo *integer* nomeadamente coordY, coordZ e coordX. É chamada a função *CalcPos()* e a *flag* DataReceived ganha o valor *true*.

```
    } else {
      //atribuir coordenadas lidas
      int index = msg_IN.indexOf(" "); //localizar o espaço
      coordY = atol(msg_IN.substring(0, index).c_str());
      msg_IN = msg_IN.substring(index + 1); //eliminar texto lido
      index = msg_IN.indexOf(" "); //localizar o espaço
      coordZ = atol(msg_IN.substring(0, index).c_str());
      msg_IN = msg_IN.substring(index + 1); //eliminar texto lido
      index = msg_IN.indexOf(" "); //localizar o espaço
      coordX = atol(msg_IN.substring(0, index).c_str());
      msg_IN = msg_IN.substring(index + 1); //eliminar texto lido
      //chamar função de calculo de posições
      CalcPos();
      //apagar dados de entrada
      msg_IN = "";
      DataReceived = true;
    }
  }
}
```

Figura 38 - Leitura de mensagem recebida (parte 3)

A função *CalcPos()* vai calcular o *array* de intervalos de tempo entre pulsos, responsável pela aceleração e desaceleração dos motores referido na secção 4.2., bem como o *array* de posições incrementais referido na secção 4.3. Ambus os *arrays* são calculados com um ciclo *for* de comprimento igual ao número total de incrementos. Os valores do vetor deslocamento YZ, do ângulo e do número de incrementos do eixo X são também calculados aqui.
Dentro do ciclo *for* responsável pelo cálculo do *array* de posições incrementais é chamada a função *CinInv()*. Esta função é responsável por calcular os valores dos ângulos referidos na secção 4.1 e respetivo número de pulsos para os eixos X, Y e Z.

Quando a *flag* DataReceived ganha o valor *true*, o código dentro da função *IncPos()* é executado e aqui começa-se por atribuir uma velocidade aos motores, depois é definida uma posição atual com base nos *arrays* calculados anteriormente.
A movimentação dos motores é feita neste momento, pelo recurso a um ciclo *for*, onde todos os valores de pulsos calculados são aqui enviados para os drivers. É também aqui que se implementa a aceleração e desaceleração. As funções que realmente movem os motores são as funções *moveTo()* e *runSpeedToPosition()* [Figura 39].

```
void IncPos() {
  if (DataReceived == true) {

    //definir uma velocidade para o movimento
    Yaxis.setMaxSpeed(vel);
    Zaxis.setMaxSpeed(vel);
    Xaxis.setMaxSpeed(vel);

    //atribuir posição atual ao primeiro valor do array
    Yaxis.setCurrentPosition(FINALPOSY[0]);
    Zaxis.setCurrentPosition(FINALPOSZ[0]);
    Xaxis.setCurrentPosition(FINALPOSX[0]);

    //mover os motores pelo array
    for (int i = 1; i <= (inc); i++) {

      //verificar se o movimento é com ou sem aceleração
      if (i <= (inc / 2) && WithACCDCC == true) {
        delayMicroseconds(contPul[i - 1]);
      } else if (i > (inc / 2) && WithACCDCC == true) {
        delayMicroseconds(contPul[inc - i - 1]);
      }

      //atribuir valores dos arrays ao array comum (necessário devido à classe MultiStepper)
      finalposXYZ[0] = FINALPOSY[i];
      finalposXYZ[1] = FINALPOSZ[i];
      finalposXYZ[2] = FINALPOSX[i];
      SYNCMovXYZ.moveTo(finalposXYZ);
      SYNCMovXYZ.runSpeedToPosition();
    }
```

Figura 39 - Posicionamento incremental (parte 1)

O restante desta função consiste em atualizar as posições finais, desativar a *flag* e enviar os valores atuais das posições dos motores e o vetor pela porta de comunicação [Figura 40].

```
      //atualizar posições
      FINALPOSY[0] = FINALPOSY[inc];
      FINALPOSZ[0] = FINALPOSZ[inc];
      FINALPOSX[0] = FINALPOSX[inc];
      realPosY = coordY;
      realPosZ = coordZ;
      realPosX = coordX;

      //desativar flag (movimento concluido)
      DataReceived = false;

      //resposta de atualização de posições e vetor
      delay(100);
      Serial.print("PosY");
      Serial.println(realPosY);
      delay(100);
      Serial.print("PosZ");
      Serial.println(realPosZ);
      delay(100);
      Serial.print("PosX");
      Serial.println(realPosX);
      delay(100);
      Serial.print("VET");
      Serial.println(vector);
    }
  }
```

Figura 40 - Posicionamento incremental (parte 2)

Concluído estes passos os motores vão encontrar-se na posição final desejada.
Para que o controlador saiba a posição dos motores quando a energia é ligada é necessário definir uma posição inicial. Isto é conseguido à custa dos fins de curso e de uma rotina de *calibração*. Esta rotina é executada quando a *flag* HomingRotine ganha o valor *true*, ou seja, quando se recebe a letra H na porta de comunicação.
A função de calibração, denominada *Homing()*, vai então movimentar os motores numa direção pré determinada até que os fins de curso sejam atingidos. Após isto, os motores deslocam-se na direção contrária até que deixem de tocar nos fins de curso. Cada motor faz este procedimento individualmente.
A Figura 41 demonstra como é implementado este processo para o eixo Z:

```
void Homing() {

  if (HomingRotine == true) {
    //definir posição atual como 0
    Yaxis.setCurrentPosition(0);
    Zaxis.setCurrentPosition(0);
    Xaxis.setCurrentPosition(0);
    //variaveis de homing
    int homeY = 0;
    int homeZ = 0;
    int homeX = 0;
    //aproximar do fim de cuso
    while (digitalRead(LSwitchZ) == 1) {
      Zaxis.setMaxSpeed(200.0); //definir velocidade
      Zaxis.moveTo(homeZ);
      homeZ--;
      Zaxis.run();
    }
    delay(500); //aguardar meio segundo
    //afastar do fim de curso
    while (digitalRead(LSwitchZ) == 0) {
      Zaxis.setMaxSpeed(20.0);
      homeZ = 1;
      Zaxis.moveTo(homeZ);
      homeZ++;
      Zaxis.run();
    }
    Serial.println("Zhome"); //eixo Z calibrado
```

Figura 41 - Homing

Após a calibração de cada eixo o controlador envia uma mensagem pela porta de comunicação para confirmar o procedimento.
Quando todos os eixos estão calibrados o controlador atribui as coordenadas e respetivos pulsos às devidas variáveis:

- $Y_{HOME} = -107;$
- $Z_{HOME} = 68;$
- $X_{HOME} = -100;$

Por último o controlador indica que a calibração está concluída e atualiza as posições atuais pela porta de comunicação.

Como referido anteriormente os fins de curso encontram-se ligados a pinos com *interrupt* e a função que é chamada consiste numa função que imobiliza instantaneamente todos os motores enviando uma mensagem de falha [Figura 42]. Esta função é dependente de duas *flags* já referidas: HomingRotine e DataReceived. Assim é possível utilizar os fins de curso na rotina de calibração e estar à espera de mau funcionamento somente quando o código se encontra na rotina de posicionamento.

```
void HardLimitsFault() {

  if (DataReceived == true && HomingRotine == false) {
    MoveArm = false;
    DataReceived = false;
    digitalWrite(Ena, HIGH); //começar com o motor desligado
    Yaxis.stop();
    Zaxis.stop();
    Xaxis.stop();
    Serial.println("FAULT");
    //atualizar posições
    FINALPOSY[0] = FINALPOSY[inc];
    FINALPOSZ[0] = FINALPOSZ[inc];
    FINALPOSX[0] = FINALPOSX[inc];
    realPosY = coordY;
    realPosZ = coordZ;
    realPosX = coordX;
  }
}
```

Figura 42 - Função de interrupt dos fins de curso

Esta função reinicia as *flags* e desativa os motores utilizando a função *stop* da biblioteca utilizada. De seguida atualiza as posições e obriga a uma rotina de calibração por parte do utilizador.

A última função a referir é a função *Move()*. Esta função é igualmente dependente de uma *flag* (MoveArm) e consiste numa função que executa um conjunto de movimentos predefinidos de forma sequencial, com ou sem aceleração. Os movimentos a executar são definidos pelo utilizador na planilha Sequential e atribuídos a variáveis do microcontrolador quando o utilizador pressiona o botão Save.

Todo este controlo de comunicação com o Arduino é feito por uma interface desenvolvida em *VisualStudio* para o propósito. Nesta interface o utilizador consegue facilmente controlar toda a máquina como:

- Configurar os parâmetros de comunicação [Figura 45];
- Configurar os valores de aceleração e velocidade máxima [Figura 45];
- Ligar ou desligar os motores [Figura 43];
- Efetuar a rotina de calibração [Figura 43];
- Posicionar a ferramenta numa coordenada X, Y e Z [Figura 43];
- Movimentação sequencial [Figura 44].

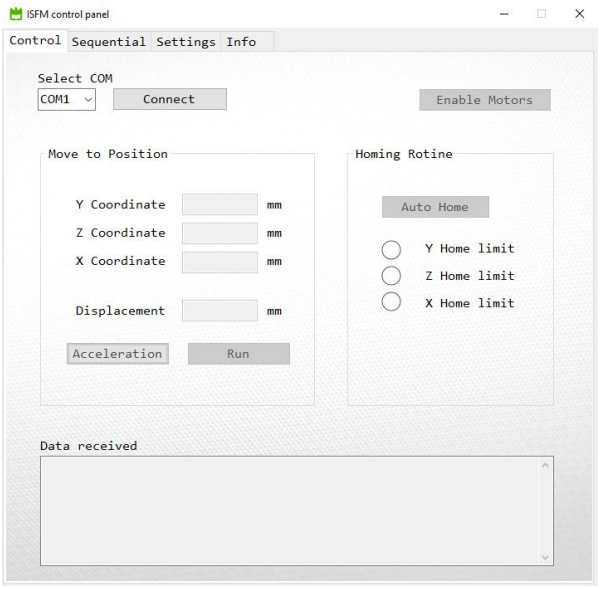


Figura 43 - Interface de controlo (1)

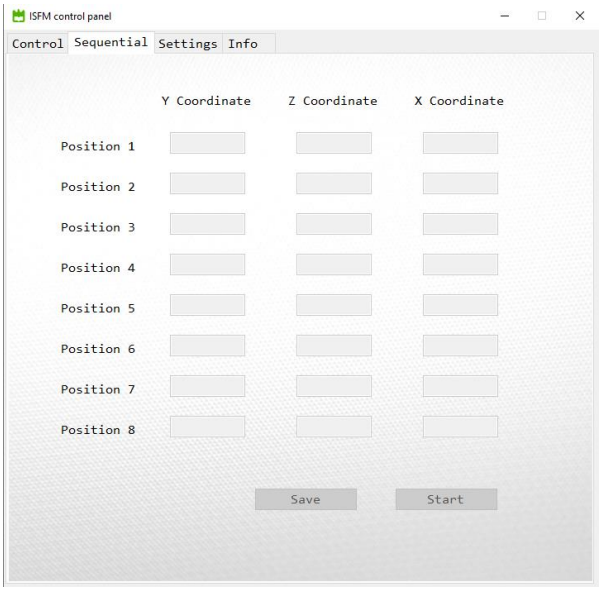


Figura 44 - Interface de controlo (2)

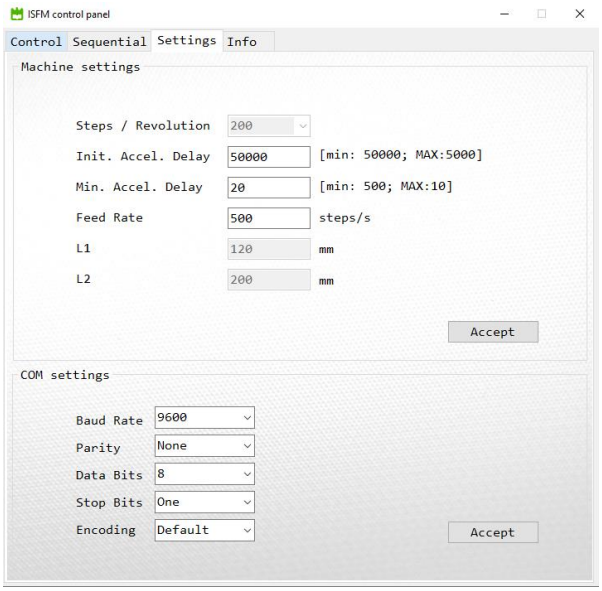


Figura 45 - Interface de controlo (3)

Com esta interface é possível configurar todos os parâmetros necessários para o controlo da máquina. Com o posicionamento sequencial pode efetuar-te até um total de 8 segmentos de reta utilizando o posicionamento incremental.
No botão *acceleration* ativa-se ou desativa-se a aceleração e desaceleração configurada na planilha *settings*.

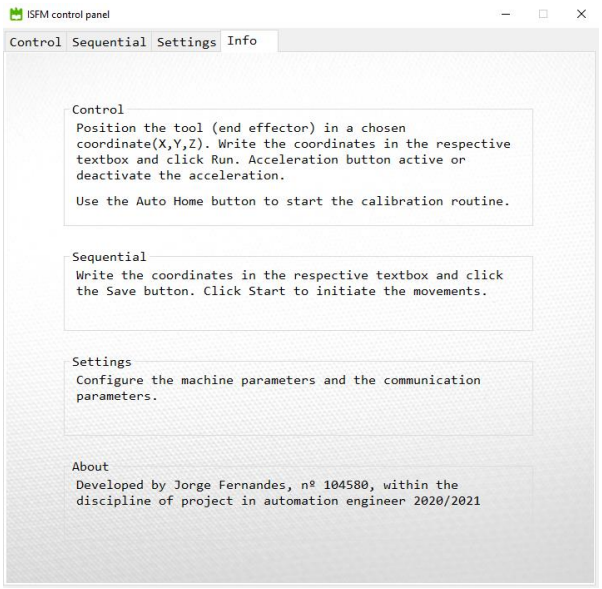


Figura 46 - Interface de controlo (4)

A última planilha contém informações sobre como utilizar a interface. Todo o desenvolvimento e várias iterações deste projeto pode ser consultado em https://github.com/JorgeFernandes-Git/ISFM.

5. Conclusões

No decorrer do projeto foram cumpridos os vários objetivos propostos. Conseguiu-se o dimensionamento dos componentes elétricos de controlo e segurança, bem como dos motores. Devido ao aspeto académico deste projeto, deu-se enfase nos equipamentos elétricos de proteção e segurança como disjuntores, contatores e relé de segurança, contudo numa primeira fase de prototipagem, alguns componentes podem ser minimizados ou omitidos.

O controlo numérico implementado, apesar de conter muitas possibilidades de melhoramento, é capaz de posicionar a ferramenta da máquina numa coordenada definida, com controlo de velocidade, aceleração e desaceleração. Numa análise de comparação entre os valores matemáticos obtidos nas rotinas do controlador com valores obtidos numa folha de Excel, verificou pequenas diferenças nos valores dos cálculos dos ângulos da cinemática inversa e consequentemente nos valores dos pulsos a enviar para o motor. Inicialmente, foi verificado um erro médio de 12 pulsos para o motor Z e 19 pulsos para o motor Y, o que se traduzia num erro nos ângulos dos motores de 20º e 40º, respetivamente (motores de 1.8º / pulso). Após algumas melhorias, nomeadamente nas equações e no tipo de variáveis utilizadas (inteiros para flutuantes) conseguiu-se reduzir este erro para 2 pulsos em Z (4º) e 10 pulsos em Y (18º). É de notar que estes são os graus de desvio no motor; os graus na máquina serão 100 vezes menores devido ao uso da redução de 100:1 acoplado à saída dos motores. Assim, estes valores dos erros podem ser desprezados.
A remodelação do programa desenvolvido, no que toca às alterações nos tipos de variáveis e no aumento da resolução no número de incrementos entre posições, levou a um aumento drástico do uso da memoria dinâmica do controlador, o que pode afetar a sua performance, assim é importante melhorar o código com este aspeto em mente. Uma solução será utilizar equações de cinemática diferencial.

Relativamente aos motores, foi pedida uma pro-forma à StepperOnline do material referido na secção 2.3 para que se desse início à encomenda.

Por último interessa referir todo o conhecimento que este projeto trouce ao aluno no que toca ao controlo de máquinas por comandos numéricos. Todo o desenvolvimento da cinemática e do controlo de velocidade e aceleração dos motores introduziu um desafio na procura de diferentes soluções, pretendendo aplicar em conjunto os conhecimentos adquiridos na disciplina.

6. A implementar

Para projetos futuros era indispensável que o controlador executasse as múltiplas posições necessária para a conformação da chapa de forma sequencial. Uma maneira de isso ser possível seria o uso de um leitor de cartões SSD ligado ao controlador. Deste modo todas as posições seriam gravadas no cartão e lidas sequencialmente pelo controlador. Deste modo evita-se que o controlador esteja sempre ligado ao computador.

Outros pontos que são necessários implementar são a interpretação por parte do controlador dos sinais de alarme vindos dos drivers em caso da perca de passos no motor, bem como implementação do acionamento de rearme em caso de aperto da botoeira de emergência ou de quebra da barreira imaterial. Para este tipo de paragens e outras paragens não programadas, o controlador devia ter um espaço na sua memoria EEPROM definida a guardar a posição do código onde ficou, para que pudesse continuar da mesma posição caso fosse possível. A EEPROM também devia ficar responsável por guardar certas configurações na máquina quando esta fosse desligada, como por exemplo, a última posição dos eixos, as velocidades e acelerações.

Está também previsto o uso de fins de curso mecânicos como *hard limits* da máquina. Aqui foi implementado uma rotina de *homing*, que utilizam sensores de proximidade indutivos, mas era importante existirem sensores mecânicos logo a seguir para garantir uma maior segurança.

Os comandos numéricos enviados para o controlador foram desenvolvidos especificamente para esta aplicação e seria interessante serem melhorados de forma a aproximarem-se de comandos numéricos utilizados por outros equipamentos. Isto porque, a utilização de softwares como gerados de código máquina (g-code) permite passar peças modeladas em softwares de CAD para código máquina de forma rápida e simples. Para isso o controlador tem de estar apto a interpretar estes comandos.

7. Bibliografia

Single Point Incremental Forming and Multi-Stage Incremental Forming on Aluminium Alloy 1050, Premika Suriyaprakan, 2013

Atmel. (2006). *AVR446: Linear speed control of stepper motor*.
http://www.atmel.com/images/doc8017.pdf

AccelStepper: AccelStepper Class Reference, from
https://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html#details

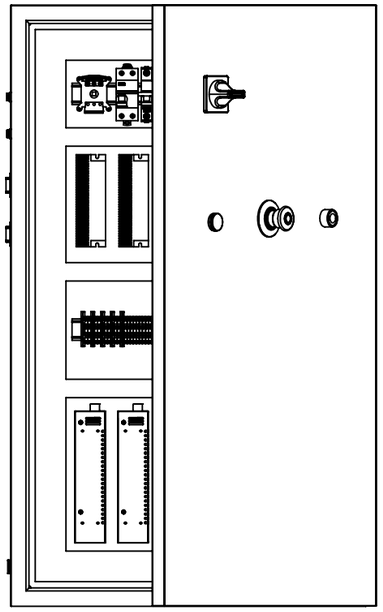
AccelStepper: MultiStepper Class Reference, from
https://www.airspayce.com/mikem/arduino/AccelStepper/classMultiStepper.html#details

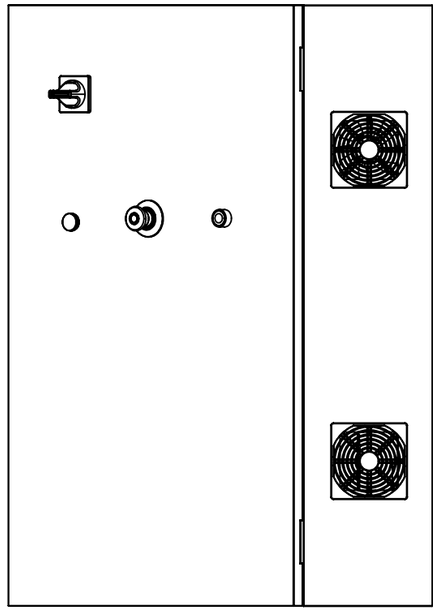
Apontamentos da disciplina de Projeto em Engenharia de Automação, 2021

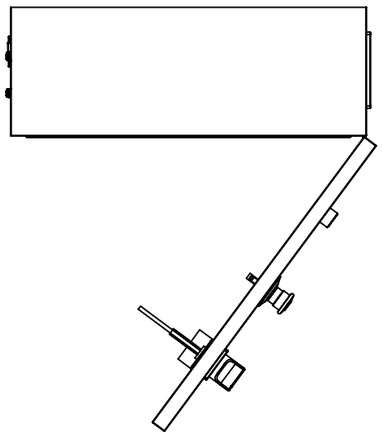
8. Anexos

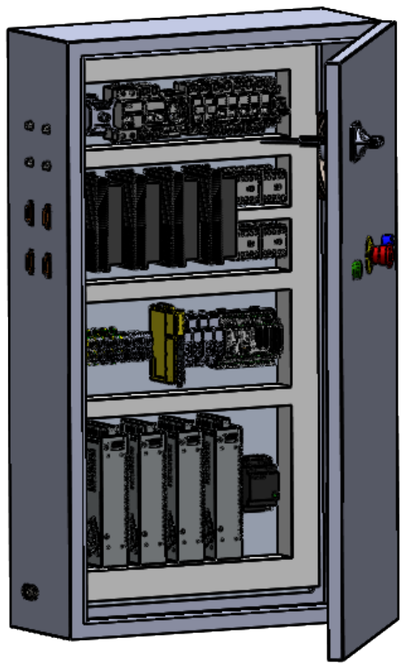
Anexo 1 – Quadro elétrico ISFM

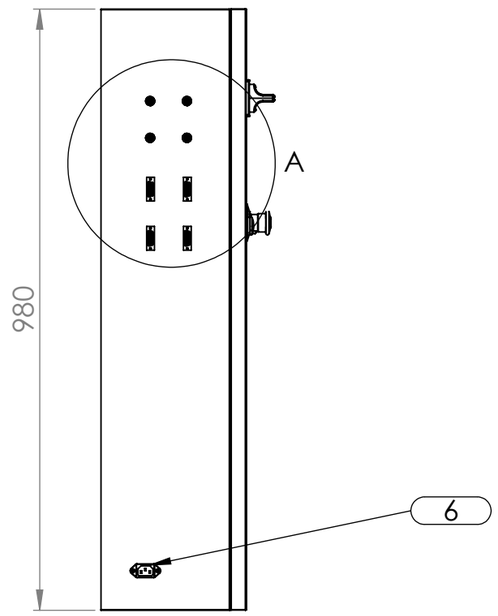
Esquema eletrico e dimensões do quadro

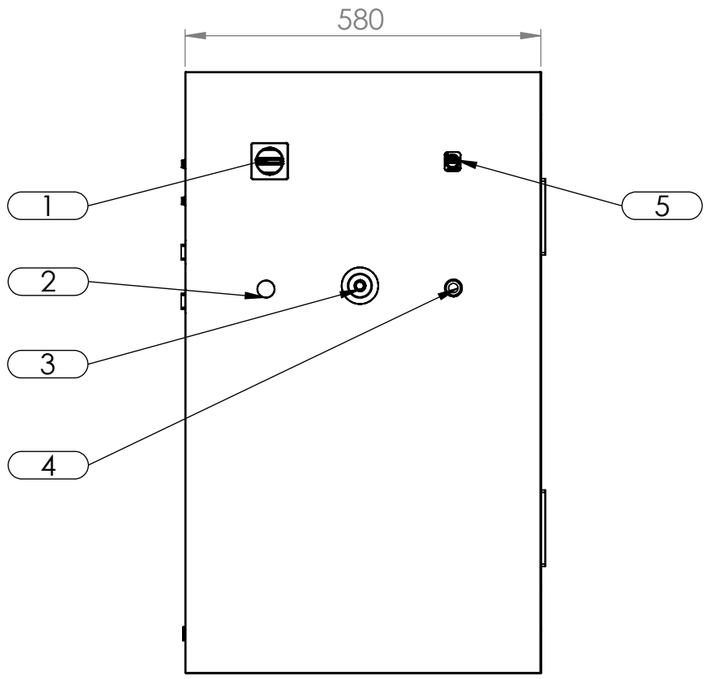


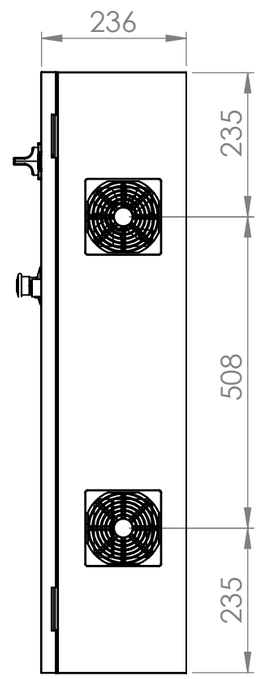


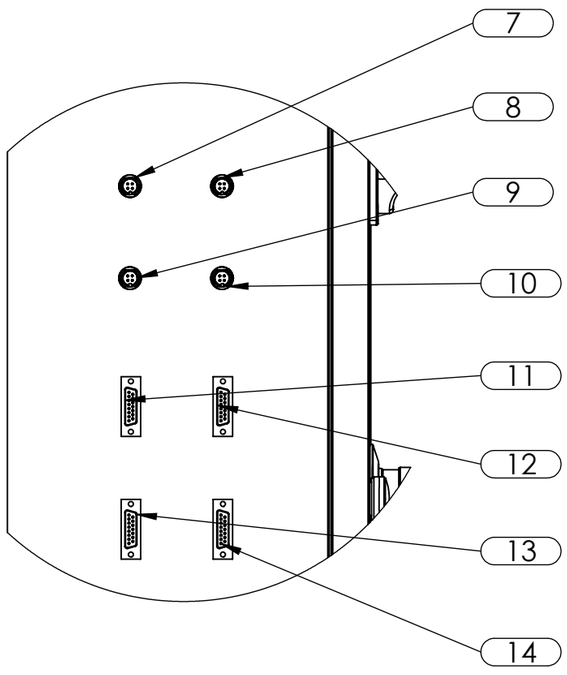








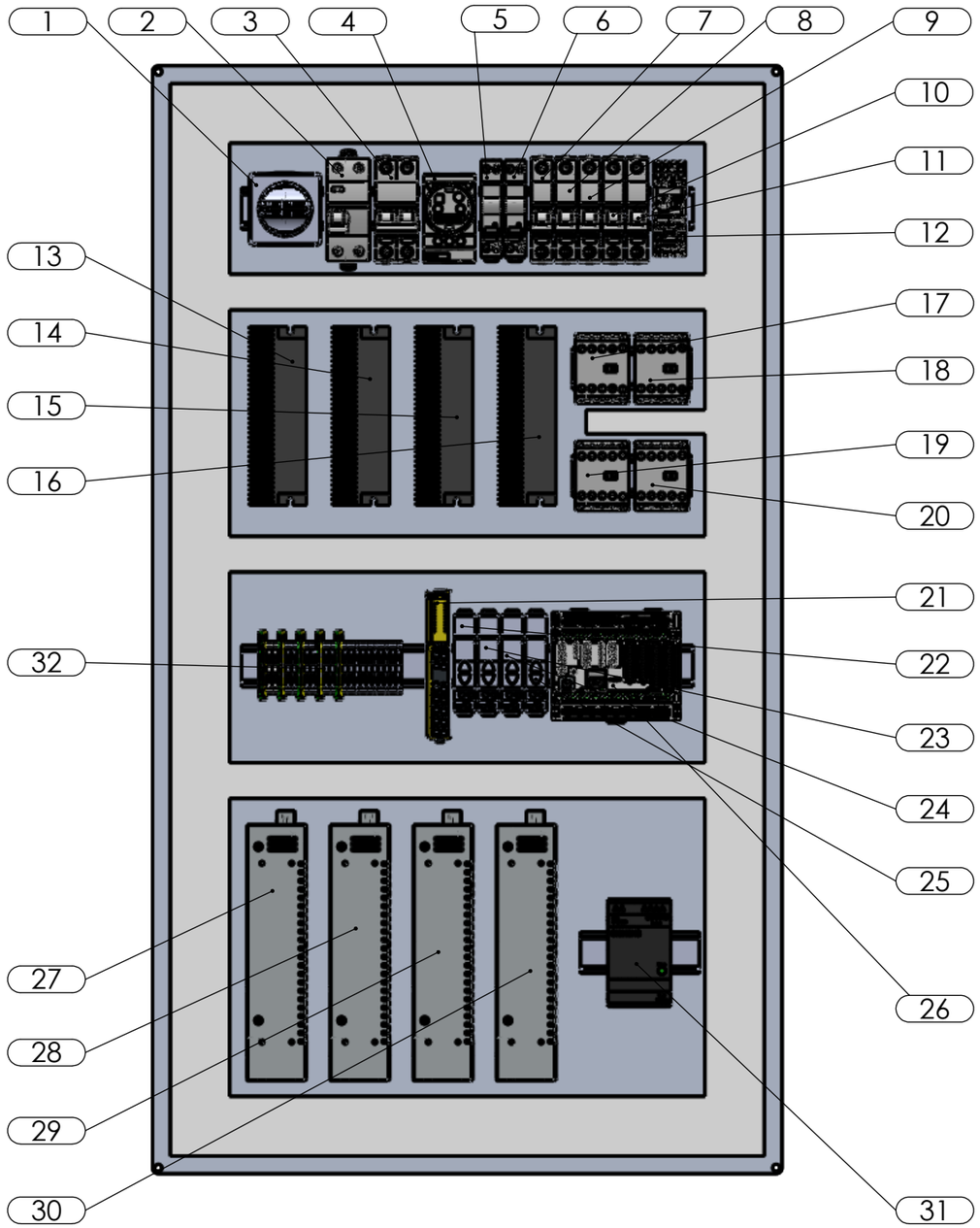




DETALHE A
ESCALA 1 : 4

Nº	COMPONENTE
1	SECCIONADOR GERAL
2	LED INDICADOR POWER VERDE
3	BOTÃO DE EMERGÊNCIA
4	BOTÃO REARM/RESET
5	FICHA USB PARA COMUNICAÇÃO SERIAL
6	FICHA IEC C14 DE ALIMENTAÇÃO 220V
7	GX 16 4 PINOS Z AXIS
8	GX 16 4 PINOS Y AXIS
9	GX 16 4 PINOS X1 AXIS
10	GX 16 4 PINOS X2 AXIS
11	DB 15 LIGAÇÃO ENCODER Z AXIS
12	DB 15 LIGAÇÃO ENCODER Y AXIS
13	DB 15 LIGAÇÃO ENCODER X1 AXIS
14	DB 15 LIGAÇÃO ENCODER X2 AXIS

Distribuição dos componentes



1	SECCIONADOR GERAL
2	DISJUNTOR DIFERENCIAL
3	DISJUNTOR 2P TOMADA
4	TOMADA
5	FUSÍVEL LED INDICADOR
6	FUSÍVEL VENTOINHA DO QUADRO
7	DISJUNTOR 1P FONTE DE ALIMENTAÇÃO Z AXIS
8	DISJUNTOR 1P FONTE DE ALIMENTAÇÃO Y AXIS
9	DISJUNTOR 1P FONTE DE ALIMENTAÇÃO X1 AXIS
10	DISJUNTOR 1P FONTE DE ALIMENTAÇÃO X2 AXIS
11	DISJUNTOR 1P FONTE DE ALIMENTAÇÃO 24V
12	RELÉ 24V
13	DRIVER Z AXIS
14	DRIVER Y AXIS
15	DRIVER X1 AXIS
16	DRIVER X2 AXIS
17	CONTATOR Z AXIS
18	CONTATOR Y AXIS
19	CONTATOR X1 AXIS
20	CONTATOR X2 AXIS
21	RELÉ DE SEGURANÇA
22	LED INDICADOR 48V FONTE DE ALIMENTAÇÃO Z AXIS
23	LED INDICADOR 48V FONTE DE ALIMENTAÇÃO Y AXIS
24	LED INDICADOR 48V FONTE DE ALIMENTAÇÃO X1 AXIS
25	LED INDICADOR 48V FONTE DE ALIMENTAÇÃO X2 AXIS
26	PLC CONTROLLINO MEGA
27	FONTE DE ALIMENTAÇÃO 48V Z AXIS
28	FONTE DE ALIMENTAÇÃO 48V Y AXIS
29	FONTE DE ALIMENTAÇÃO 48V X1 AXIS
30	FONTE DE ALIMENTAÇÃO 48V X2 AXIS
31	FONTE DE ALIMENTAÇÃO 24V
32	BORNES

Dimensões (vista frontal)

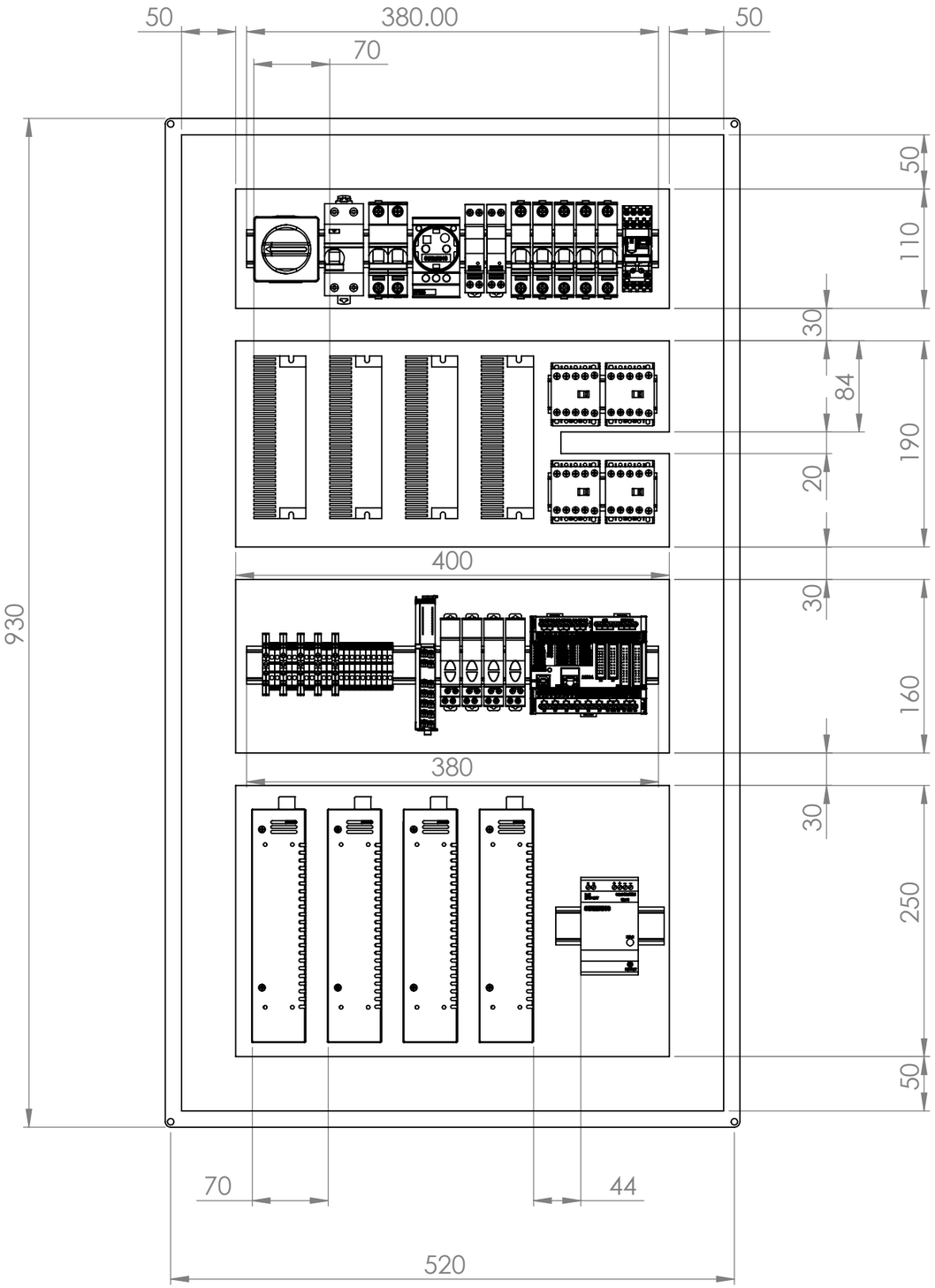
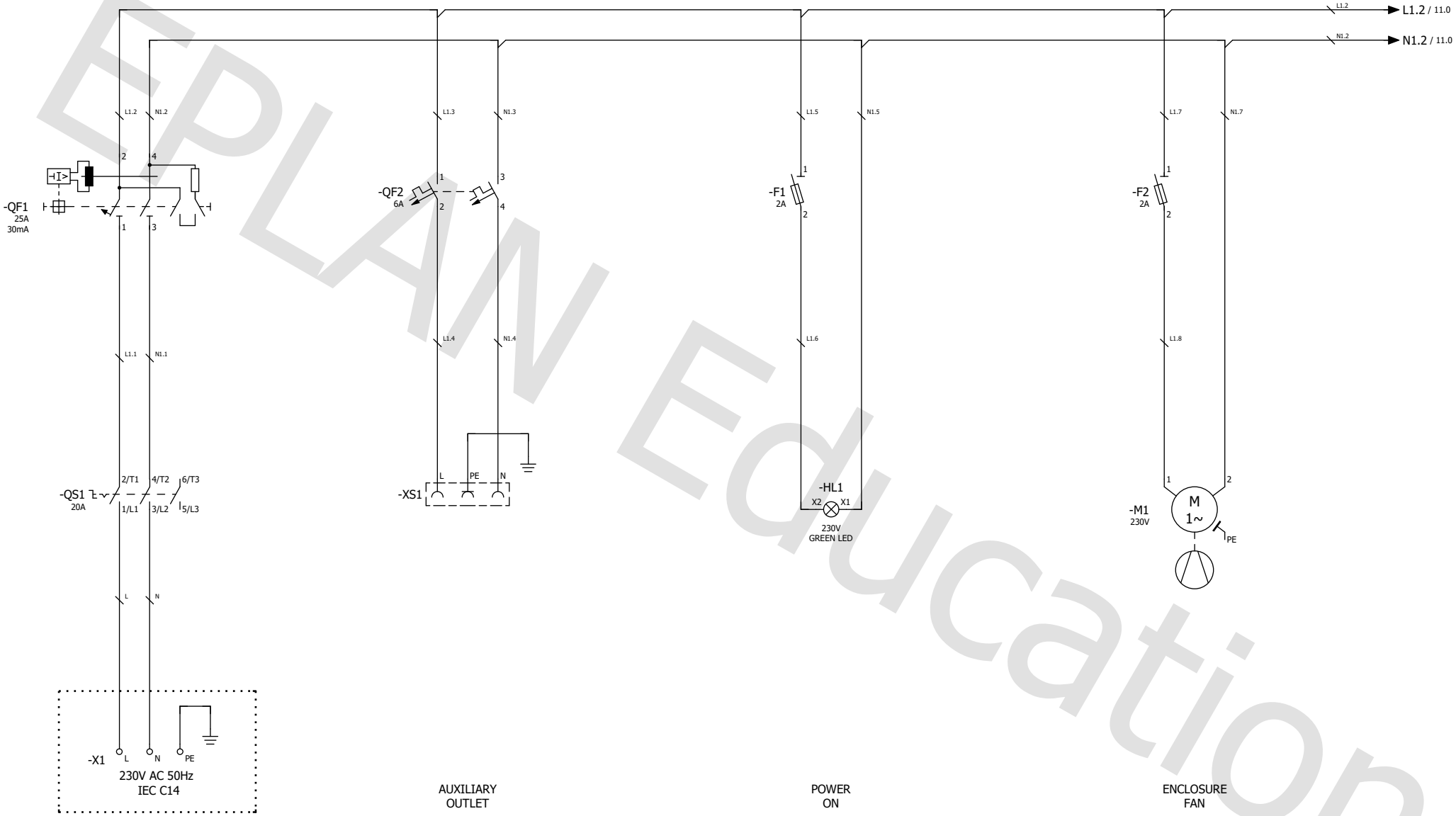


Tabela de componentes

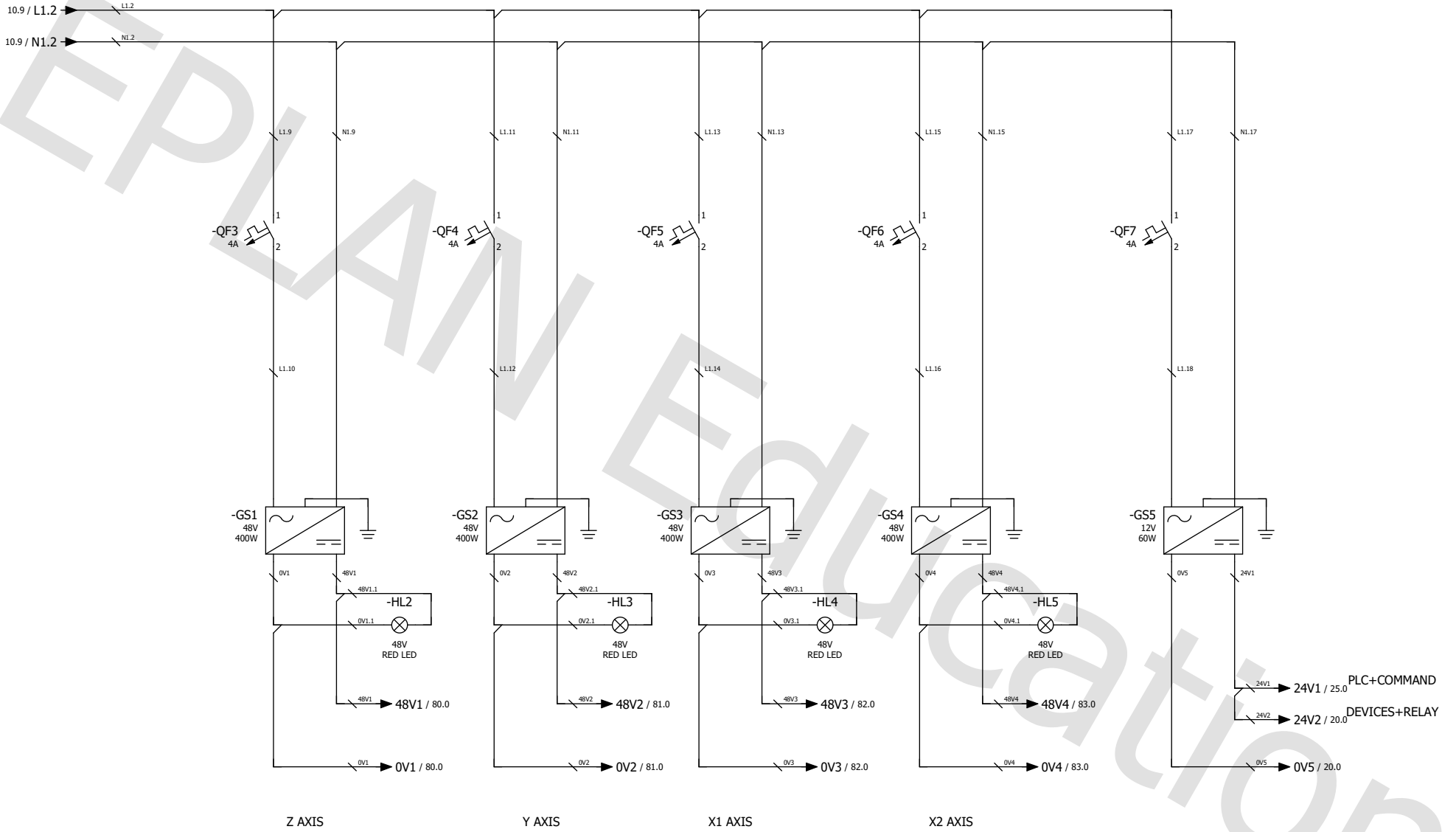
1	SECCIONADOR GERAL	-QS1	Schneider	SE.VCCDN20
2	DISJUNTOR DIFERENCIAL	-QF1	Schneider	SE.A9R60225
3	DISJUNTOR 2P TOMADA	-QF2	Schneider	SE.A9F06206
4	TOMADA	-XS1	Siemens	SEI.5TE6800
5	FUSÍVEL LED INDICADOR	-F1	Schneider	SE.A9N15646
6	FUSÍVEL VENTOINHA DO QUADRO	-F2	Schneider	SE.A9N15646
7	DISJUNTOR 1P FONTE DE ALIMENTAÇÃO Z AXIS	-QF3	Schneider	SE.A9F03104
8	DISJUNTOR 1P FONTE DE ALIMENTAÇÃO Y AXIS	-QF4	Schneider	SE.A9F03104
9	DISJUNTOR 1P FONTE DE ALIMENTAÇÃO X1 AXIS	-QF5	Schneider	SE.A9F03104
10	DISJUNTOR 1P FONTE DE ALIMENTAÇÃO X2 AXIS	-QF6	Schneider	SE.A9F03104
11	DISJUNTOR 1P FONTE DE ALIMENTAÇÃO 24V	-QF7	Schneider	SE.A9F03104
12	RELÉ 24V	-RA1	Schneider	SE.RXM4AB2BDPVM
13	DRIVER Z AXIS	-Z_CL86T	StepperOnline	CL86T
14	DRIVER Y AXIS	-Y_CL86T	StepperOnline	CL86T
15	DRIVER X1 AXIS	-X1_CL86T	StepperOnline	CL86T
16	DRIVER X2 AXIS	-X2_CL86T	StepperOnline	CL86T
17	CONTATOR Z AXIS	-KM1	Schneider	SE.CA4KN40BW3
18	CONTATOR Y AXIS	-KM1	Schneider	SE.CA4KN40BW3
19	CONTATOR X1 AXIS	-KM1	Schneider	SE.CA4KN40BW3
20	CONTATOR X2 AXIS	-KM1	Schneider	SE.CA4KN40BW3
21	RELÉ DE SEGURANÇA	-KF1	SICK	SICK.RLY3-EMSS1xx
22	LED INDICADOR 48V FONTE DE ALIMENTAÇÃO Z AXIS	-HL2	Schneider	SE.A9E18330
23	LED INDICADOR 48V FONTE DE ALIMENTAÇÃO Y AXIS	-HL3	Schneider	SE.A9E18330
24	LED INDICADOR 48V FONTE DE ALIMENTAÇÃO X1 AXIS	-HL4	Schneider	SE.A9E18330
25	LED INDICADOR 48V FONTE DE ALIMENTAÇÃO X2 AXIS	-HL5	Schneider	SE.A9E18330
26	PLC CONTROLLINO MEGA	-PLC	Controllino	CTR.100-200-00
27	FONTE DE ALIMENTAÇÃO 48V Z AXIS	-GS1	StepperOnline	S-400-48
28	FONTE DE ALIMENTAÇÃO 48V Y AXIS	-GS2	StepperOnline	S-400-48
29	FONTE DE ALIMENTAÇÃO 48V X1 AXIS	-GS3	StepperOnline	S-400-48
30	FONTE DE ALIMENTAÇÃO 48V X2 AXIS	-GS4	StepperOnline	S-400-48
31	FONTE DE ALIMENTAÇÃO 24V	-GS5	Siemens	SEI.6EP1332-1SH71
32	BORNES	-X1	Phoenix	PXC.3210356

Esquema de Potência 1



			Date	05/06/2021			GENERAL POWER		
			Ed.	Jorge Fernandes	Estampagem Incremental				
			Appr.					IEC_tpl003	Page 10
Modification	Date	Name	Original		Replacement of	Replaced by			Page 1 / 16

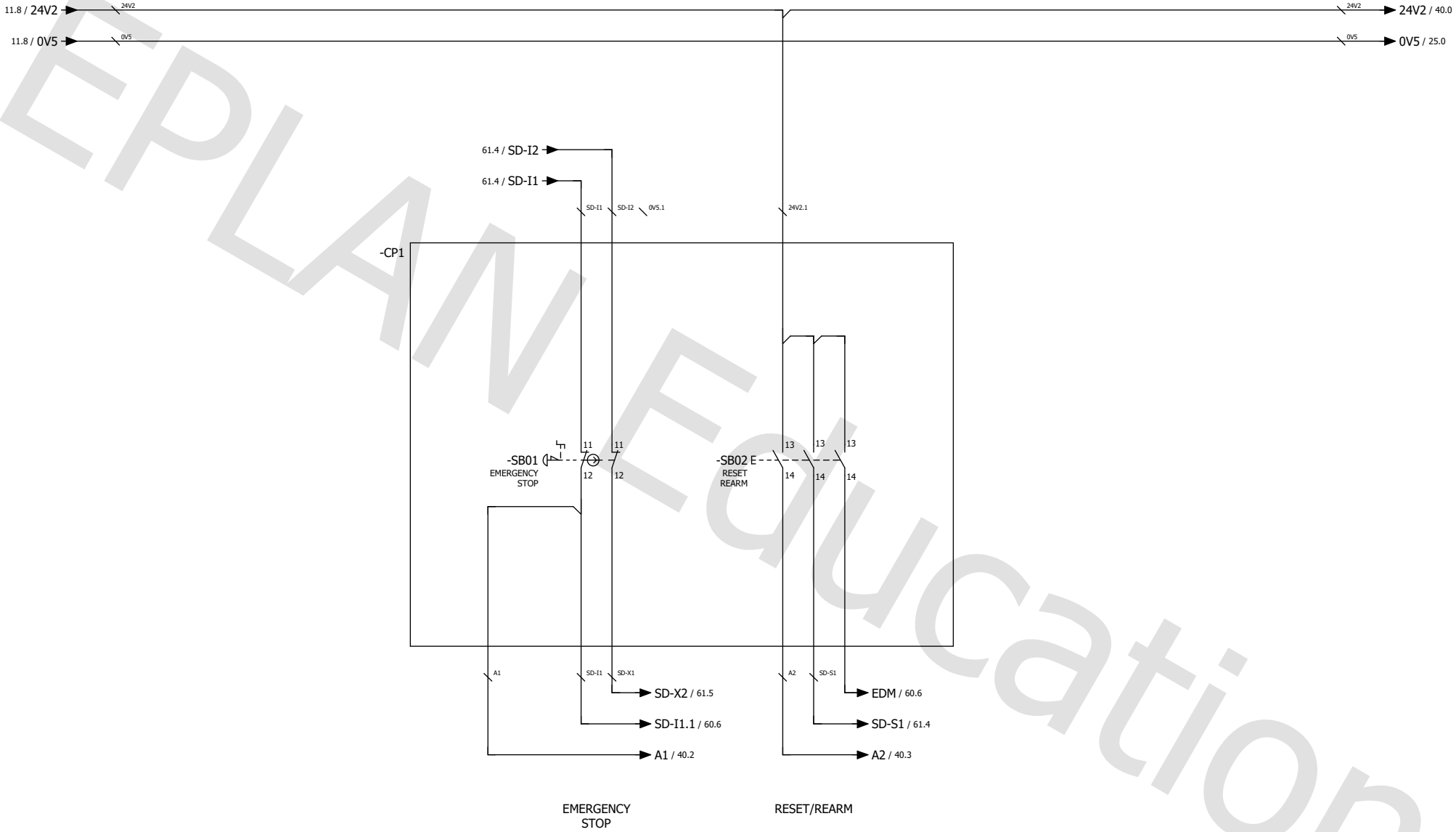
Esquema de Potência 2



10 20

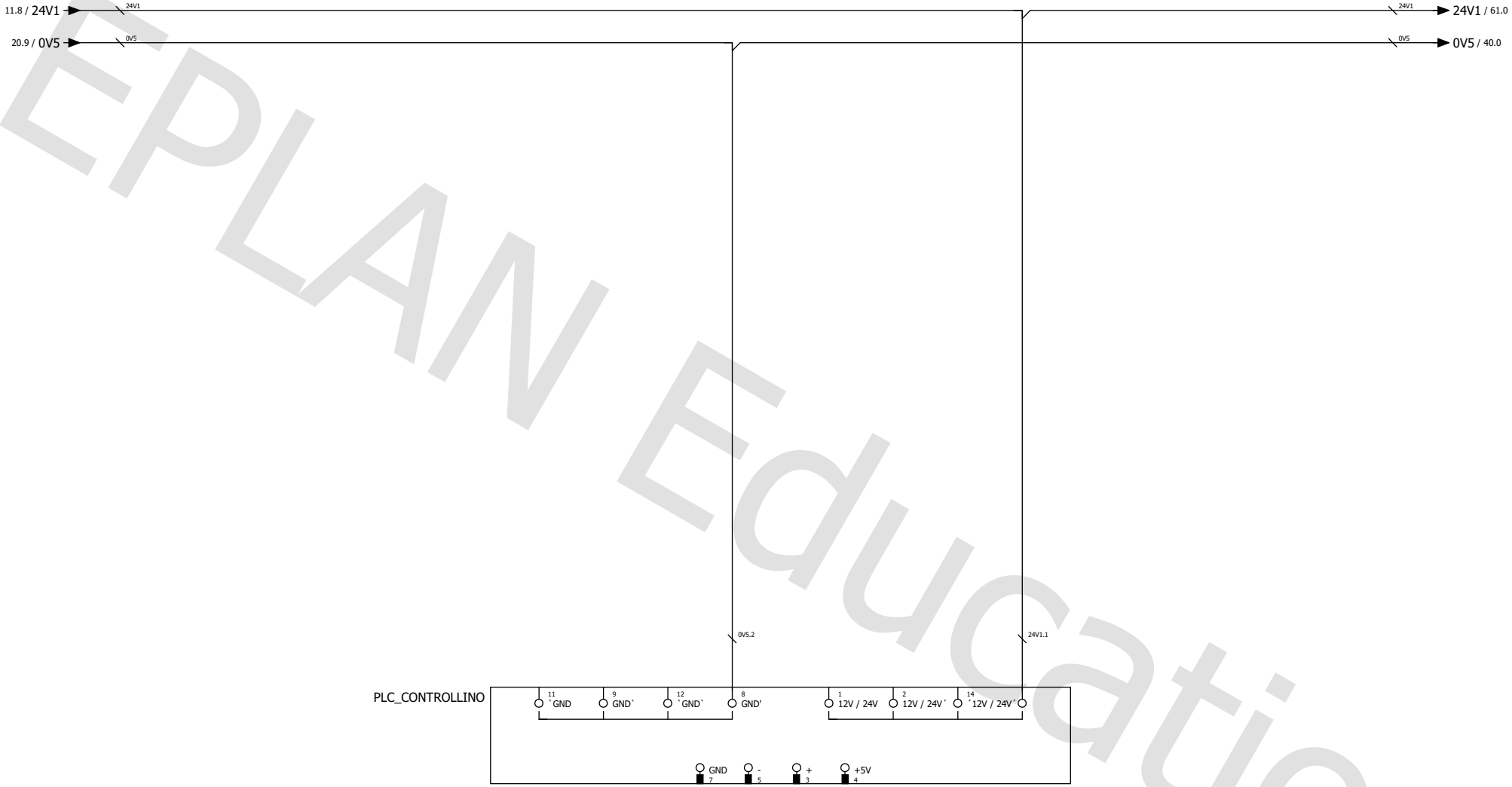
			Date	05/06/2021			GENERAL POWER		
			Ed.	Jorge Fernandes	Estampagem Incremental				
			Appr.					IEC_tpl003	Page 11
Modification	Date	Name	Original		Replacement of	Replaced by			Page 2 / 16

Caixa de comando



			Date	05/06/2021			COMMAND BOX		
			Ed.	Jorge Fernandes	Estampagem Incremental				
			Appr.					IEC_tpl003	Page 20
Modification	Date	Name	Original		Replacement of	Replaced by			Page 3 / 16

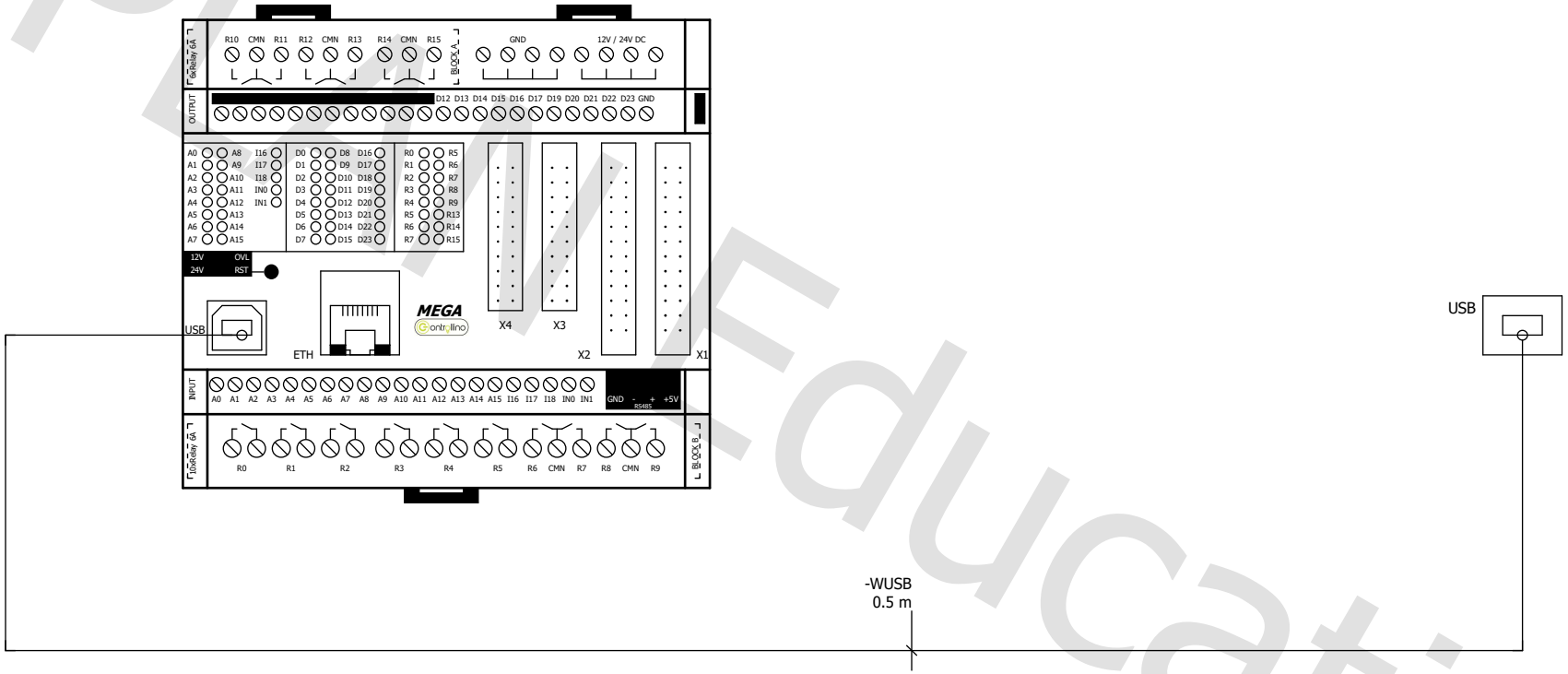
Alimentação controlador



20 30

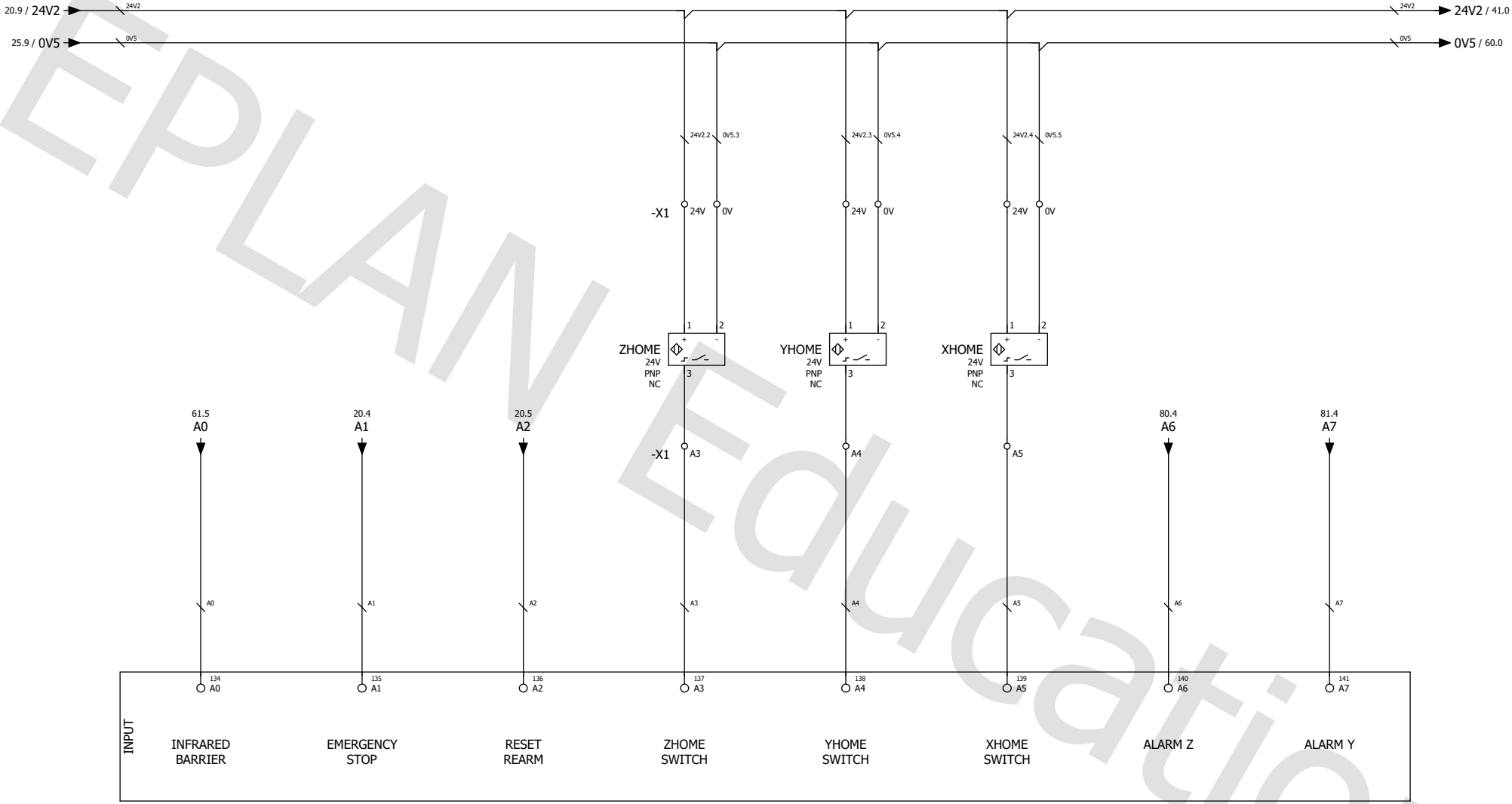
			Date	05/06/2021			POWER SUPPLY PLC		
			Ed.	Jorge Fernandes					
			Appr.		Estampagem Incremental			IEC_tpl003	Page 25
Modification	Date	Name	Original		Replacement of	Replaced by			Page 4 / 16

Controlador



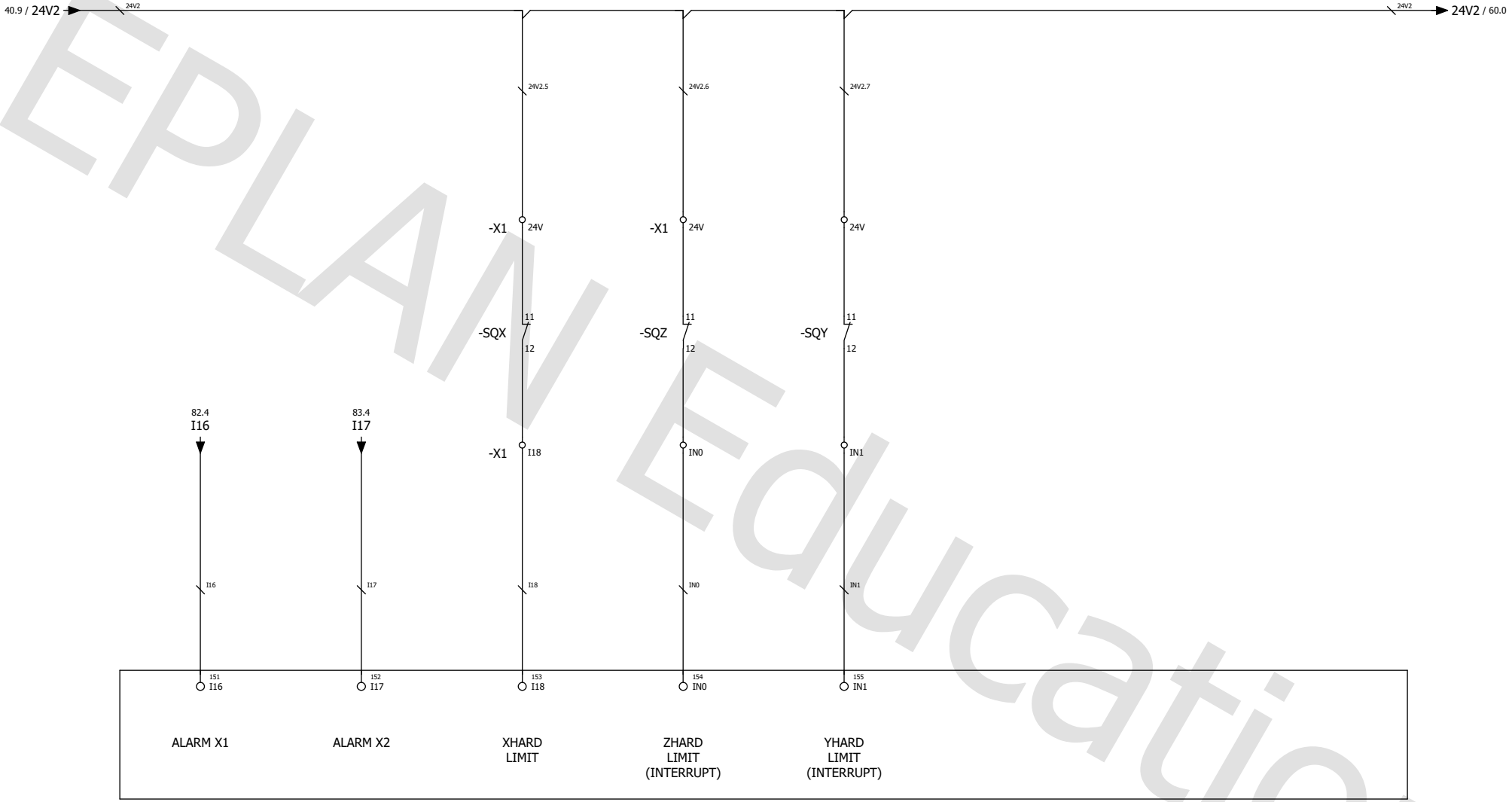
			Date	10/05/2021			CONTROLLER		
			Ed.	Jorge Fernandes					
			Appr.		Estampagem Incremental			IEC_tpl003	Page 30
Modification	Date	Name	Original		Replacement of	Replaced by			Page 5 / 16

Entradas 1



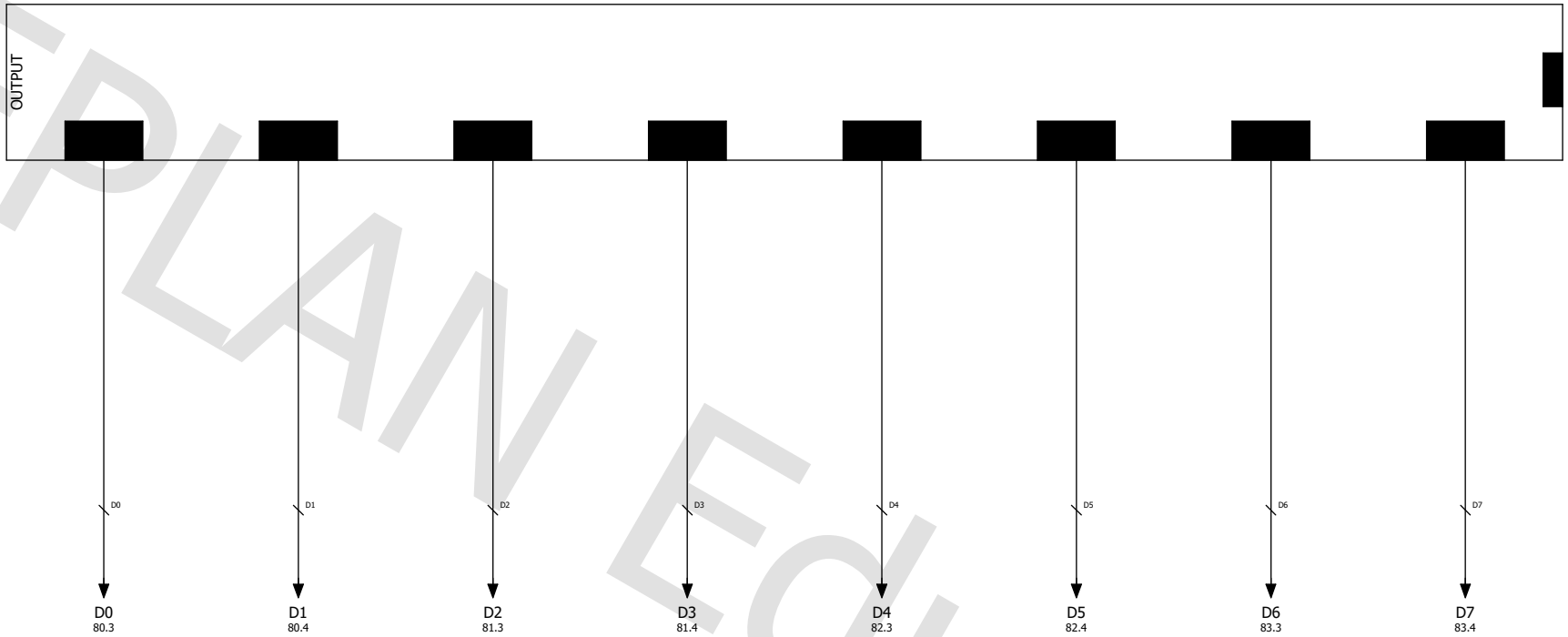
			Date	05/06/2021			INPUT		
			Ed.	Jorge Fernandes	Estampagem Incremental				
			Appr.					IEC_tpl003	Page 40
Modification	Date	Name	Original		Replacement of	Replaced by			Page 6 / 16

Entradas 2



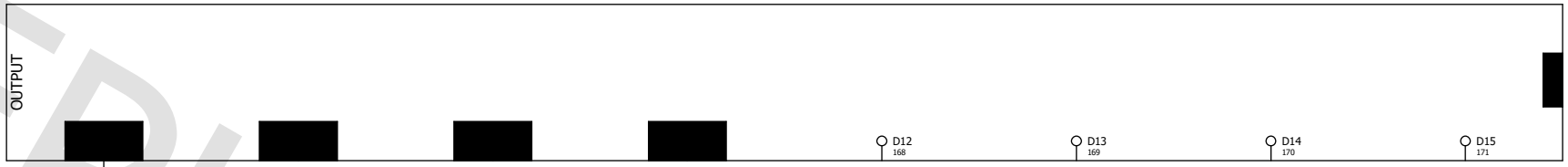
			Date	05/06/2021			INPUT	
			Ed.	Jorge Fernandes				
			Appr.		Estampagem Incremental			IEC_tpl003
Modification	Date	Name	Original		Replacement of	Replaced by		

Saídas 1



			Date	10/05/2021			OUTPUT		
			Ed.	Jorge Fernandes					
			Appr.		Estampagem Incremental			IEC_tpl003	Page 50
Modification	Date	Name	Original		Replacement of	Replaced by			Page 8 / 16

Saídas 2

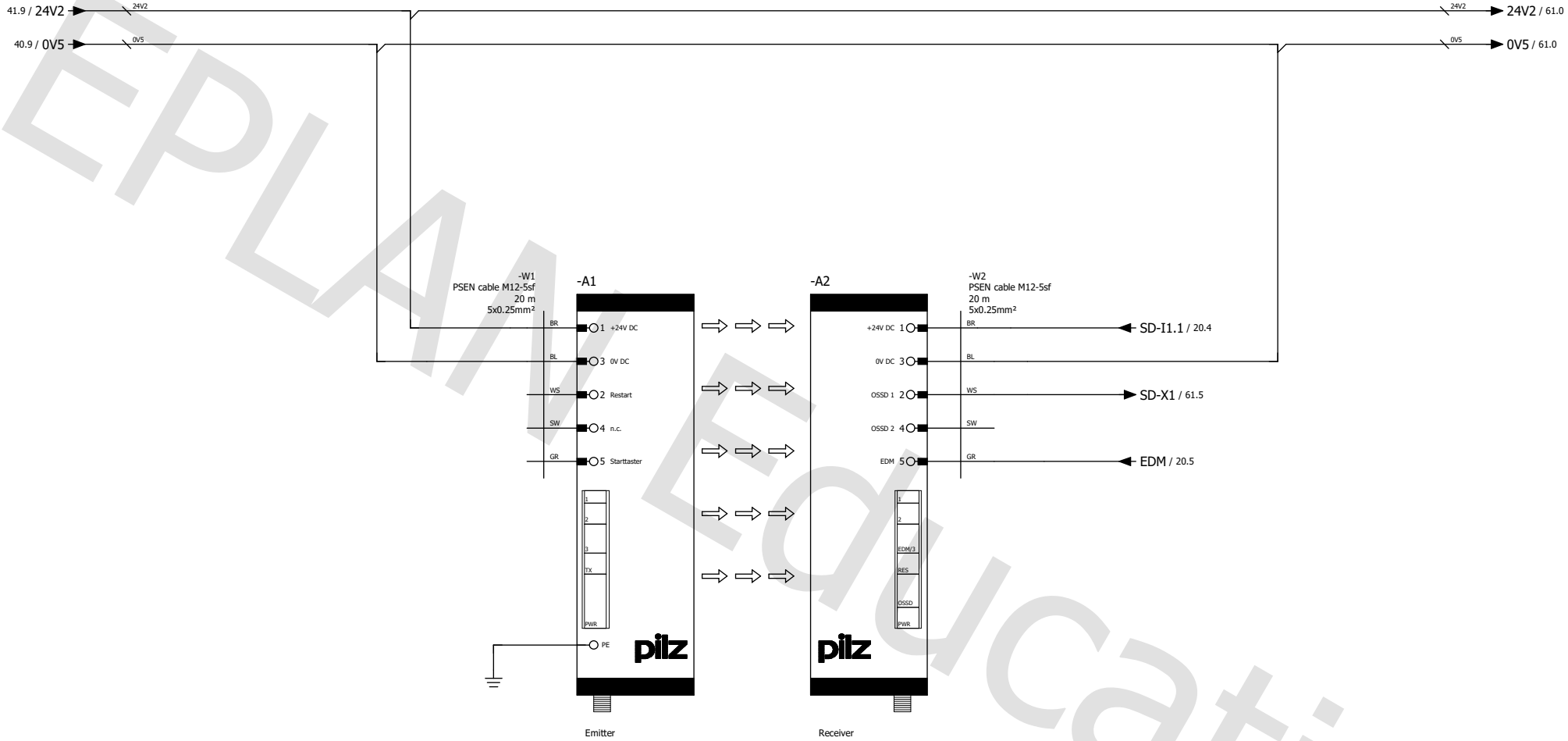


ENA
MOTORS

50 60

			Date	10/05/2021			OUTPUT		
			Ed.	Jorge Fernandes	Estampagem Incremental				
			Appr.					IEC_tpl003	Page 51
Modification	Date	Name	Original		Replacement of	Replaced by			Page 9 / 16

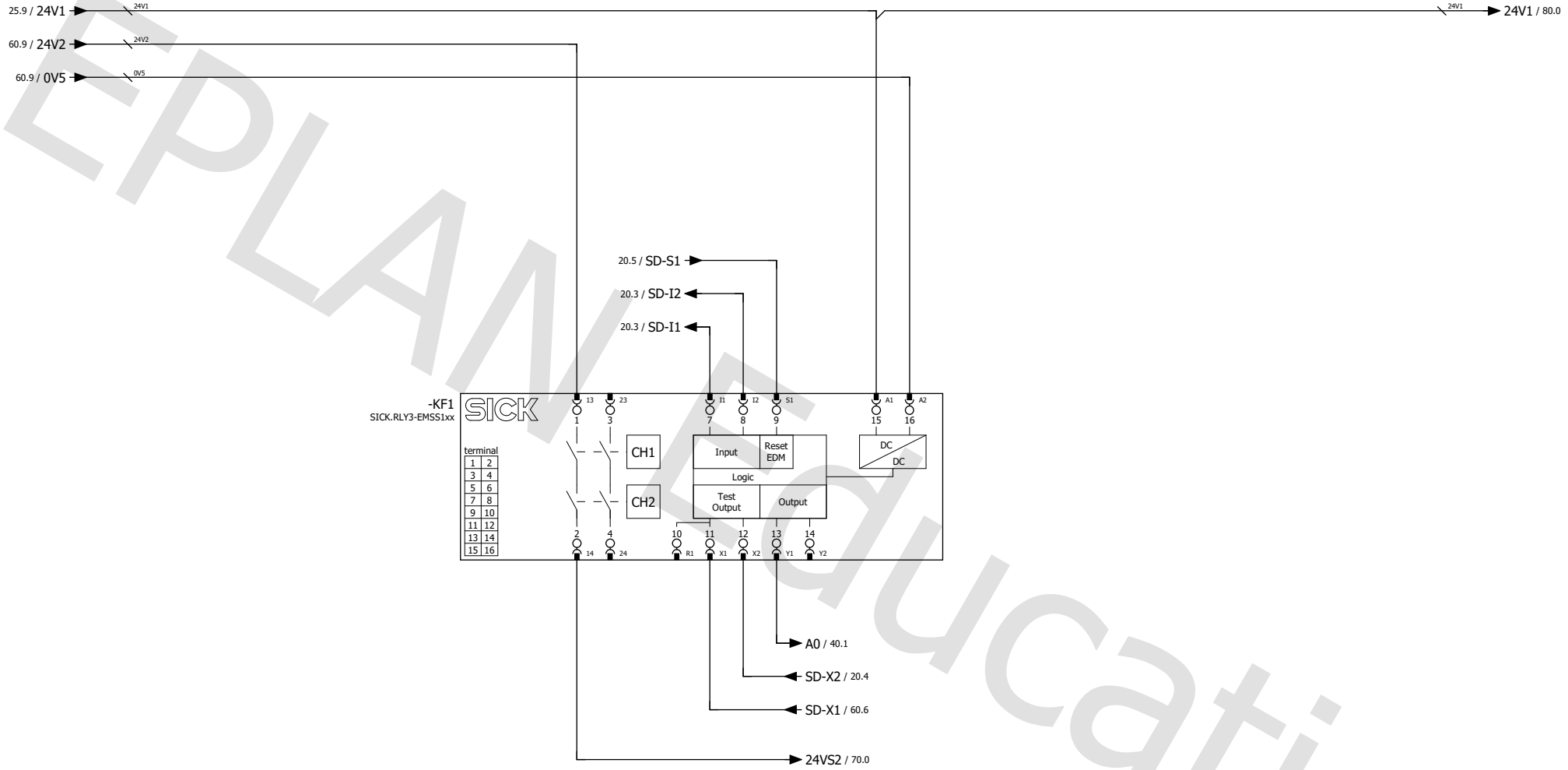
Barreira imaterial



51 61

			Date	05/06/2021			LIGHT BARRIER		
			Ed.	Jorge Fernandes					
			Appr.		Estampagem Incremental			IEC_tpl003	Page 60
Modification	Date	Name	Original		Replacement of	Replaced by			Page 10 / 16

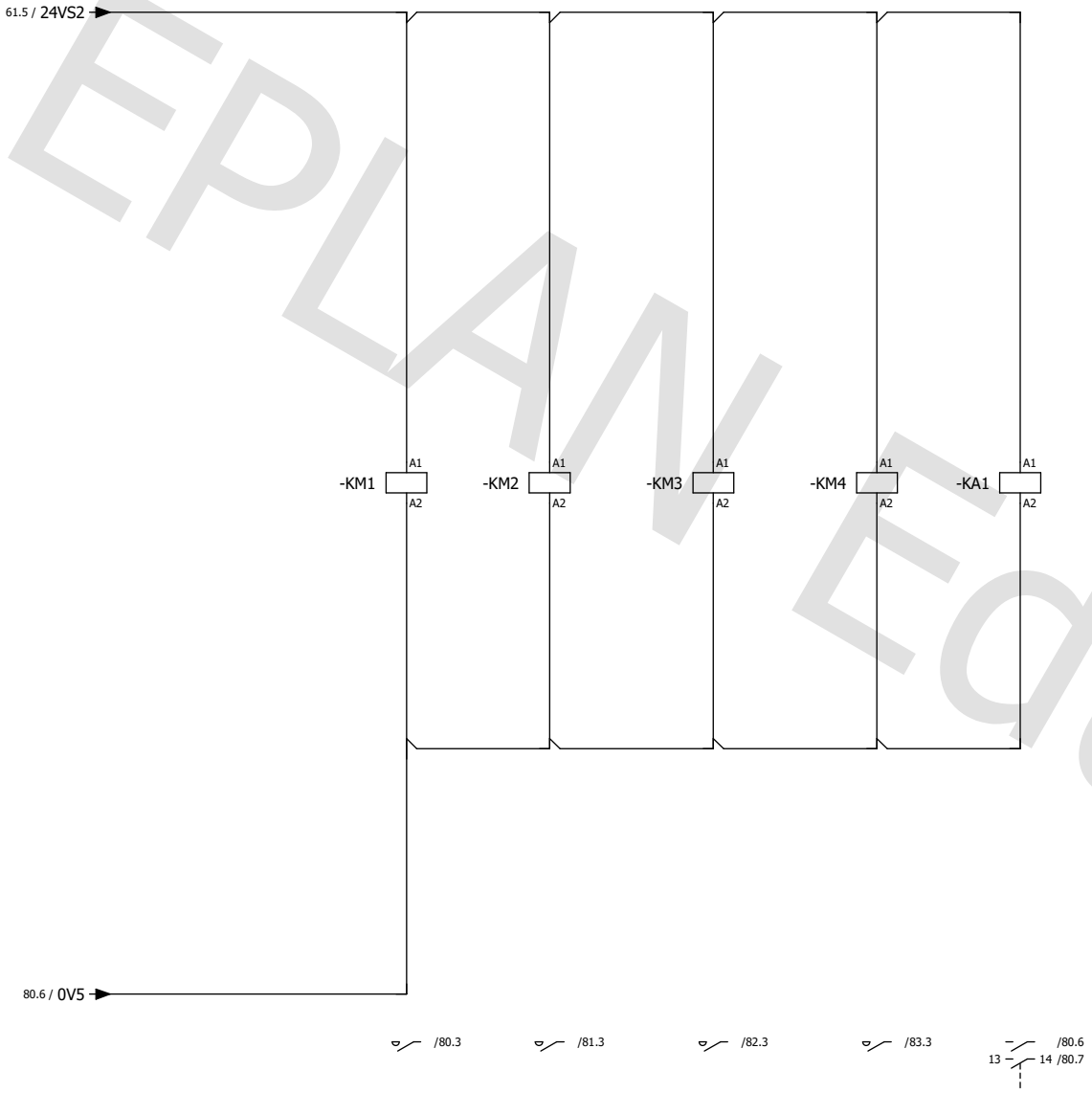
Relé de segurança



60 70

			Date	05/06/2021			SAFETY RELAY		
			Ed.	Jorge Fernandes					
			Appr.		Estampagem Incremental			IEC_tpl003	Page 61
Modification	Date	Name	Original		Replacement of	Replaced by			Page 11 / 16

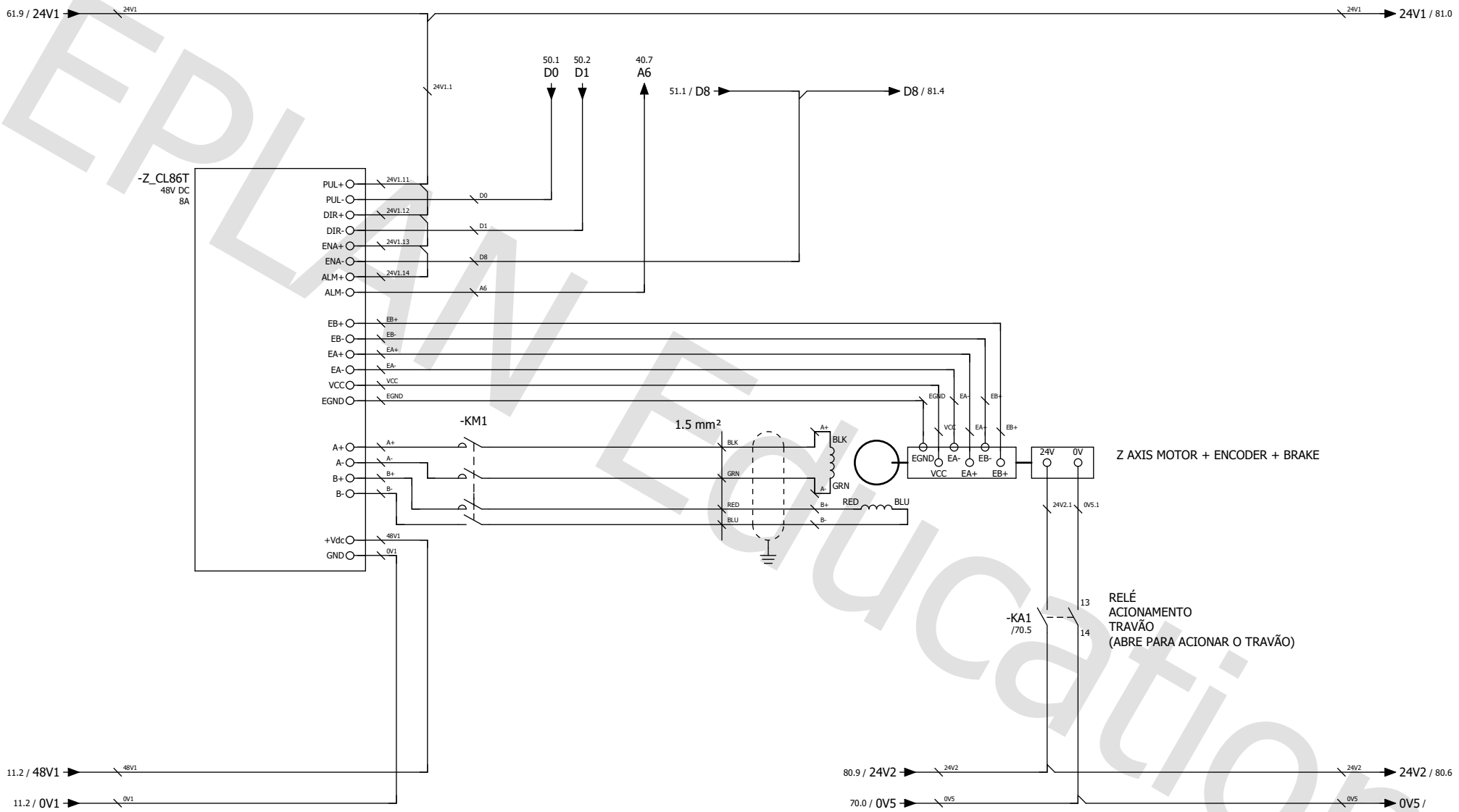
Ativação e controlo



61 80

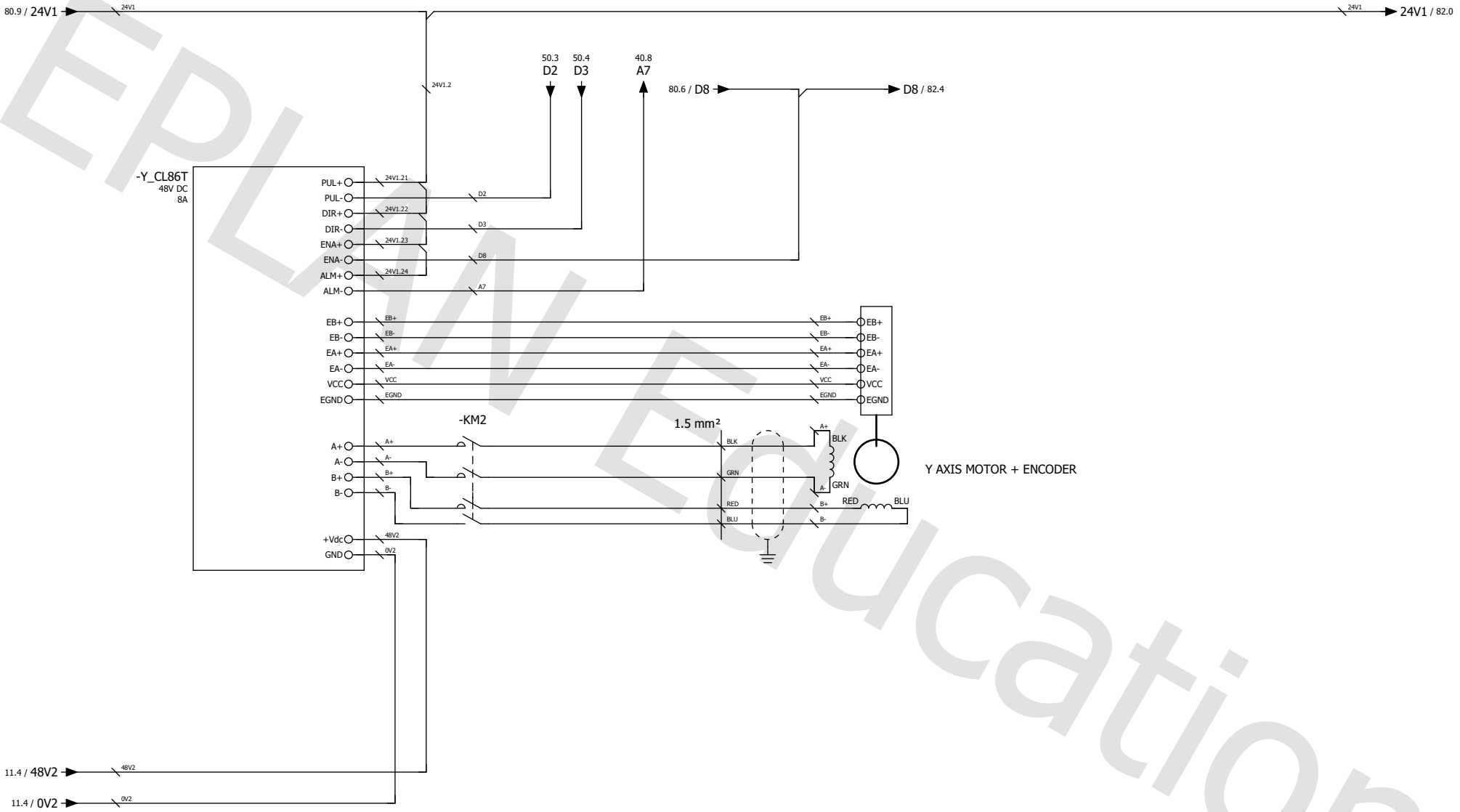
			Date	10/05/2021			ACTIVATION AND CONTROL		
			Ed.	Jorge Fernandes					
			Appr.		Estampagem Incremental			IEC_tpl003	Page 70
Modification	Date	Name	Original		Replacement of	Replaced by			Page 12 / 16

Driver eixo Z



			Date	10/05/2021			Z AXIS DRIVER		
			Ed.	Jorge Fernandes					
			Appr.		Estampagem Incremental			IEC_tpl003	Page 80
Modification	Date	Name	Original		Replacement of	Replaced by			Page 13 / 16

Driver eixo Y

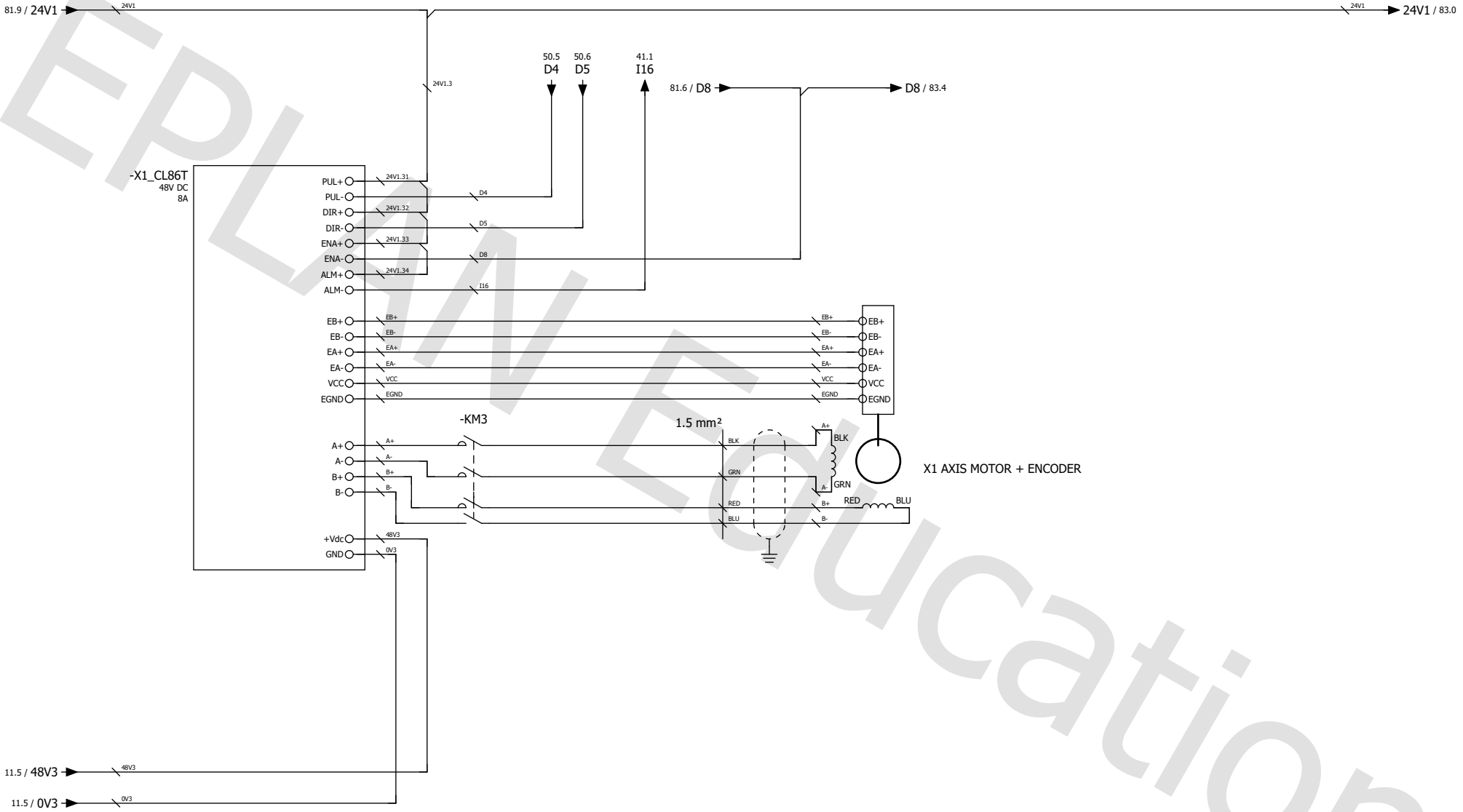


80

82

			Date	10/05/2021			Y AXIS DRIVER		
			Ed.	Jorge Fernandes					
			Appr.		Estampagem Incremental			IEC_tpl003	Page 81
Modification	Date	Name	Original		Replacement of	Replaced by			Page 14 / 16

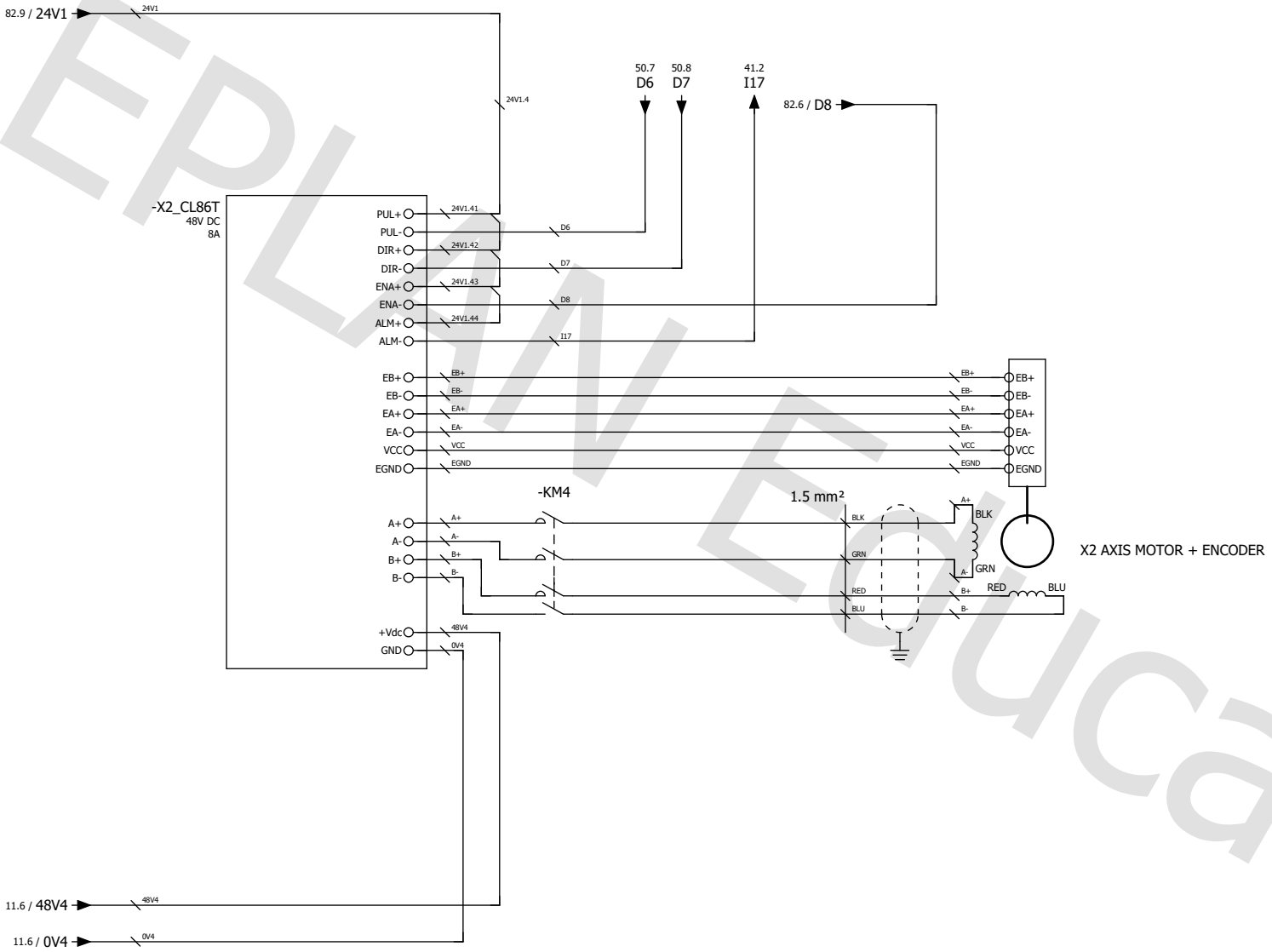
Driver eixo X1



81 83

			Date	10/05/2021			X1 AXIS DRIVER		
			Ed.	Jorge Fernandes					
			Appr.		Estampagem Incremental			IEC_tpl003	Page 82
Modification	Date	Name	Original		Replacement of	Replaced by			Page 15 / 16

Driver eixo X2



			Date	10/05/2021			X2 AXIS DRIVER		
			Ed.	Jorge Fernandes	Estampagem Incremental				
			Appr.					IEC_tpl003	Page 83
Modification	Date	Name	Original		Replacement of	Replaced by			Page 16 / 16