**PROJETO E CONSTRUÇÃO DE UM ROBÔ PIANISTA**

*Aluno: Luiz Gustavo Pfitscher e Feldmann*

*Professor: Sérgio Boscato Garcia*

***Curso: Engenharia Mecânica***

**Resumo**

Robôs pianistas para entretenimento vêm sendo desenvolvidos desde 1863, com soluções mecânicas variando desde acionamentos individuais das teclas até mãos robóticas. O objetivo deste trabalho é o projeto e fabricação de um robô pianista composto de duas mãos robóticas móveis, com cinco mecanismos de dedos cada, capazes de rotação e atuação vertical. O projeto inclui modelagem mecânica, síntese de mecanismos, dimensionamento de atuadores, dos seus controladores eletrônicos e desenvolvimento do *software* de controle. O robô é constituído de um chassi de perfis sobre o qual é montado um sistema de movimentação horizontal por meio de correias acionadas por motores de passo. As mãos são dotadas de servomotores que rotacionam os dedos e cilindros pneumáticos que os pressionam contra as teclas. Um gabinete anexo abriga os componentes eletrônicos como fontes, *drivers*, além de um Arduino Mega e um microcomputador Raspberry Pi que realizam o controle lógico. Nesse computador interno é executado um aplicativo desenvolvido para permitir a programação do repertório do robô, realizada desenhando-se as trajetórias de movimentação sobre uma linha do tempo com notas carregadas a partir de uma partitura em MIDI.

**Palavras-chave**: robô pianista; mão robótica; mecanismos.

**DESIGN AND MANUFACTURING OF A PIANIST ROBOT**

*Student: Luiz Gustavo Pfitscher e Feldmann*

*Professor: Sérgio Boscato Garcia*

**Abstract**

Pianist robots for entertainment have been developed since 1861, with mechanical designs ranging from individual key pressing to robotic hands. The goal of this work is to design and manufacture a pianist robot comprised of two moving robotic hands, with five finger mechanisms each, all capable of rotation and vertical action. The project includes mechanical modelling, mechanism synthesis, specification of actuators, electronic controllers, and the developement of a controller software. The robot is based on a beam structure over which is mounted a horizontal motion system of belts tractioned by stepper motors. The hands contain servomotors that rotate the fingers, and pneumatic cylinders which press them against the keys. An attached cabinet houses the electronic components, such as power supplies and drivers, besides an Arduino Mega and a Raspberry Pi microcomputer, which perform the logical control. This computer runs an application developed to allow the programming of the robot, which is done by drawing movement paths over a timeline of notes loaded from a MIDI music sheet.

**Key-words**: pianist robot; robot hand; mechanisms.

**1. INTRODUÇÃO**

Um robô é uma máquina programável capaz de executar tarefas de maneira autônoma e repetitiva. Existem inúmeros tipos de robôs destinados às mais variadas aplicações, dentro e fora da indústria. São empregados, geralmente, em substituição a um ser humano por poderem apresentar muito mais destreza, precisão, força, repetibilidade e poderem atuar em ambientes perigosos ou impróprios às pessoas e por longos períodos de tempo sem interrupção. A Figura 1.1 mostra algumas aplicações da robótica por área. Em linhas de montagem de veículos, por exemplo, os robôs articulados permitem rápido e preciso posicionamento de peças da carroceria e subsequente soldagem com grande velocidade e repetibilidade. Já na medicina, robôs cirurgiões realizam operações de precisão com cortes muito menores e menos invasivos. O uso de robôs veiculares (*rovers*) possibilita a exploração de ambientes inóspitos ou inacessíveis como as profundezas do oceano ou mesmo outros planetas. Também há aplicações domésticas para a robótica, como a limpeza e entretenimento.

|  |  |  |
| --- | --- | --- |
| (a) | (b) | (c) |
| (d) | (e) | (f) |

**Figura 1.1**. Aplicações da robótica: (a) soldagem da carroceria na indústria automotiva; (b) cirurgia de precisão; (c) *rover* de exploração de Marte; (d) desarme de bombas a distância; (e) limpeza doméstica; (f) entretenimento. Adaptado de IEEE Robots (2019).

O uso de robôs para aplicações industriais já é comum desde a metade do século XX (Ray, 2008). Porém, nota-se uma tendência de disseminação da robótica para áreas mais próximas do cotidiano, onde se incluem carros autônomos, robôs entregadores de encomendas, robôs recepcionistas e cuidadores de idosos. Atualmente, podem ser encontrados robôs domésticos voltados para limpeza e entretenimento.

De acordo com Kapur (2005), casos de robótica aplicada à música existem pelo menos desde 1863, sendo o *Pianista* o primeiro conhecido. Desenvolvido pelo francês Jean Fourneaux, não se trata propriamente de um robô e sim de um piano automatizado, onde o pressionar das teclas se dava por um rolo de cames com acionamento por pedal. Outro piano autônomo é o *Pianola*, de 1896, também totalmente mecânico mas, nesse caso, programável por um rolo de papel perfurado. Uma versão mais moderna do aparato é o *Contraption Instant Prepared Piano* desenvolvido por Gerhard Trimpin na década de 80, com acionamento elétrico e comandado por MIDI.

Desde então, surgiram casos de robôs, de fato, destinados a tocar vários instrumentos, não apenas pianos. Kajitani (1989) desenvolveu um robô violinista articulado. Williamson (1999) apresentou o *Cog,* um robô percussionista humanóide desenvolvido no *Massachusetts Institute of Technology* (MIT). Solis (2009) cita o robô saxofonista WAS-1 e o flautista WF4-RIV, ambos desenvolvidos na universidade japonesa de Waseda, que já conta com uma longa tradição de desenvolvimento de robôs instrumentistas. Os robôs citados são exibidos na Figura 1.2.

|  |  |  |
| --- | --- | --- |
| (a) | | (b) |
| (c) | (d) | (e) |

**Figura 1.2**. Robôs músicos para diversos instrumentos: (a) piano automático *Contraption Instant Prepared Piano* (Kapur, 2005); (b)robô percussionista *Cog* (Williamson, 1999)*;* (c) robô flautista WF4-RIV (Solis, 2004); (d) robô saxofonista *WAS-1* (Solis, 2009); (e) robô violinista Mubot (Kajitani, 1989).

O robô flautista WF4-RIV, diferente dos demais, apresenta um objetivo educacional. Além de demonstrar a técnica para tocar seu instrumento, ele grava e analisa estudantes de música iniciantes, identificando seus erros e fazendo, por voz, críticas e sugestões pré-programadas.

Com um viés educativo ou de entretenimento, um robô instrumentista contribui para a aproximação entre robótica, engenharia e o cotidiano das pessoas. Nesse contexto, toma-se por objetivo projetar e construir um robô pianista capaz de reproduzir com fidelidade uma grande variedade de músicas.

**2. TEORIA MUSICAL E REPRESENTAÇÃO DA MÚSICA**

De acordo com Med (1996), o som é a sensação produzida pela vibração do tímpano, causada por ondas de diferença de pressão que se propagam no ar. Essas ondas, para serem percebidas, devem estar na faixa de frequências audíveis que é entre 20 Hz e 20 kHz, podendo variar ligeiramente entre os indivíduos e ao longo do tempo. Ondas sonoras irregulares são meros ruídos, já as regulares (que apresentam altura definida) são notas musicais. Uma nota musical é definida pela sua altura (que corresponde à sua frequência), duração, intensidade (a sua amplitude ou volume) e timbre. O timbre está relacionado ao formato da onda e varia entre os instrumentos. A diferença entre a nota “lá” de um piano e de um violino, por exemplo, está justamente no timbre. O instrumento gera a nota transmitindo ao ar as vibrações que produz, podendo fazer isso de diferentes maneiras para obter diferentes timbres, como pelo impacto (percussão), atrito em uma corda ou eletronicamente.

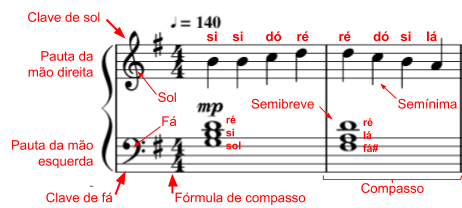
Existem apenas sete notas musicais, que são representadas por sílabas em línguas latinas e por letras em línguas germânicas: “dó” (C), “ré” (D), “mi” (E), “fá” (F), “sol” (G), “lá” (A) e “si” (B). Notas intermediárias são obtidas por meio sustenido (♯) e do bemol (♭) que representam, respectivamente, um aumento e um decremento de um semitom na altura da nota. No piano essas notas estão associadas às teclas pretas. O intervalo de frequências audíveis está subdividido em oitavas, cada uma composta destas mesmas sete notas. É convencionado que a nota “lá” da 4º oitava tem a frequência de 440 Hz, e a partir dessa referência todas as outras notas são definidas.

A música é formada pela combinação de notas musicais e outros sons de maneira simultânea e sucessiva. A componente simultânea é chamada harmonia e a sucessiva, melodia (Med, 1996). O ritmo da música é o ordenamento das notas em padrões que se repetem ao longo do tempo. A música costuma ser representada por meio de uma linguagem própria chamada partitura, escrita com um conjunto de símbolos padronizado chamado notação musical. Pauta é o conjunto de cinco linhas sobre o qual as figuras musicais são escritas. Fazendo analogia com um plano cartesiano, o eixo horizontal corresponde ao tempo e o vertical à altura da nota. No início da pauta é colocada a clave, um símbolo para referenciar o eixo vertical. A “clave de sol”, por exemplo, marca em qual linha fica nota “sol”, e todas as outras notas são então definidas relativamente a esse ponto. Barras verticais dividem a pauta em intervalos de mesma duração chamados compassos. Após a clave é colocada uma fração, a fórmula de compasso, com a função de referenciar a escala do eixo tempo. O numerador indica quantas unidades de tempo cabem em um compasso e o denominador informa qual figura musical tem a duração de uma unidade de tempo, conforme Tabela 2.1.

**Tabela 2.1.** Figuras musicais e seus símbolos.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 8 | 16 |
| Semibreve | Mínima | Semínima | Colcheia | Semicolcheia |
|  |  |  |  |  |

A figura musical é o símbolo que dá a duração daquela nota. As durações das figuras estão em uma progressão geométrica de razão ½ , ou seja, cada figura tem metade da duração da figura anterior. Uma semibreve tem a duração de duas mínimas e uma colcheia tem metade da duração de uma semínima. A Figura 2.1 apresenta um trecho de uma partitura e indica os principais elementos. A relação de compasso 4/4 representa um compasso de duração de quatro unidades de tempo, cada unidade com a duração de uma semínima. No compasso cabe, portanto, quatro semínimas ou uma semibreve. Há duas pautas, a primeira traz as notas a serem tocadas com a mão direita, onde a “clave de sol” indica que a nota “sol” fica na segunda linha. A segunda pauta, para a mão esquerda, possui uma “clave de fá” indicando que a nota “fá” fica na quarta linha.



**Figura 2.1.** Trecho da partitura de “Ode a Alegria” para piano com indicação dos principais elementos da notação musical.

A partitura escrita em notação musical é muito conveniente para músicos e compositores, porém imprópria para representar a música em computadores ou instrumentos eletrônicos. Para suprir essa necessidade, fabricantes de equipamentos musicais eletrônicos formaram na década de 80 o consórcio *The International MIDI Association*. A organização define o *Musical Instrument Digital Interface* (MIDI) como um padrão destinado a unificar a representação musical no meio digital. A norma associa a cada nota (Anexo A1) e a cada instrumento um código numérico e um série de eventos padronizados que descrevem ações relacionadas essas notas e instrumentos. O arquivo “.midi” contém uma sequência de eventos de maneira a descrever em ordem cronológica cada ação que deve ser tomada por um instrumento eletrônico ou sintetizador musical de modo a reproduzir a música. O Anexo A2 traz uma representação esquemática da estrutura desse formato de arquivo.

**3. ROBÔS PIANISTAS E PARÂMETROS CONSTRUTIVOS**

Robôs pianistas já foram apresentados nos mais diversos modelos, dos mais simplórios aos mais complexos. Um dos mais simples é o *Mini Humanoid Pianist* (Batula, 2010), construído pela modificação do brinquedo *RoboNova* e capaz de tocar apenas duas teclas. Na mesma linha há o robô de LEGO (Zhang, 2009), que pode tocar uma tecla por vez, mas construído sobre um mecanismo de movimentação linear de modo a se posicionar sobre o teclado conforme necessário.

O conceito de movimentação linear é incorporado por Zhang (2009) em um projeto dotado de uma mão robótica com cinco dedos acionados por servomotores e que se movimenta sobre uma cremalheira. Um modelo semelhante é o de Lin (2010), com duas mãos robóticas e um sistema de movimentação linear, porém acionando as teclas utilizando cilindros pneumáticos. Além disso, motores de passo são utilizados para pivotar os dedos no plano do teclado, dando a cada dedo mais um grau de liberdade que permite atingir teclas adjacentes. Lin (2014) aprimora o projeto substituindo os motores de passo por servomotores e tornando o conjunto da mão mais compacto.

Como os instrumentos musicais foram criados para serem tocados por pessoas, é esperado que um formato humanóide seja a escolha natural para um robô instrumentista, porém outra possível abordagem é a automatização de cada tecla individualmente com atuadores solenóides (Zhang, 2009). Essa abordagem torna o *design* mais eficiente, pois elimina o problema de criar trajetórias de movimentação. Contudo, pode ser considerado menos elegante do ponto de vista de entretenimento por parecer menos natural. Alguns robôs pianistas são apresentados na Figura 3.1.

|  |  |  |
| --- | --- | --- |
| (a) | (b) | (c) |

|  |  |  |
| --- | --- | --- |
| (d) | (e) | |

**Figura 3.1**. Robôs pianistas: (a) *Mini Humanoid Pianist* (Batula, 2010); (b) feito de LEGO (Zhang, 2009); (c) *Piano Playing Robot* (Zhang, 2009); (d) *Electronic Piano Playing Robot* (Lin, 2010) e (e) novo *Electronic Piano Playing Robot* (Lin, 2014).

**3.1. Mecanismos e acionamentos**

Um mecanismo é um sistema de elementos unidos e organizados para transmitir movimento de maneira predeterminada (Norton, 2011). São compostos de corpos rígidos chamados elos que se unem por meio de juntas. As juntas são conexões que permitem algum tipo de movimento relativo entre os elos, seja de translação ou rotação. O processo de criação de um mecanismo que cumpra um movimento desejado é chamado síntese. Há vários tipos de síntese que podem ser utilizadas de acordo com o objetivo do projetista. A geração de trajetória é um tipo de síntese que permite criar um mecanismo onde um determinado ponto de um dos elos percorra um caminho prescrito. Há métodos sistemáticos para realizar esse processo para mecanismos de três, quatro ou mais barras.

Mecanismos apresentam uma entrada que recebe um movimento por meio de um dispositivo de acionamento e provocam outro movimento na saída. Os acionamentos podem ser contínuos (rotativos), proporcionados por motores, ou intermitentes produzidos por atuadores lineares. Alguns tipos comuns de acionamentos são motores elétricos, atuadores solenóides, cilindros hidráulicos e pneumáticos.

Servomotores são motores elétricos que atuam em malha fechada por meio de sensores de posição (*encoders*) montados no seu eixo (Norton, 2013). São dotados de um circuito de controle que reage aos sinais dos sensores para modular a tensão ou a corrente elétricas do motor, permitindo controle de posição, velocidade e aceleração de acordo com funções programadas. Tais motores têm resposta rápida e são muito precisos, porém de custo elevado. Já os motores de passo, pelo contrário, atuam em malha aberta. Segundo Norton (2013), esses motores são voltados para aplicações de posicionamento. A cada pulso recebido na bobina do motor, o eixo se move um ângulo fixo chamado passo. O posicionamento da saída do motor é feito justamente fornecendo um número contado de pulsos. Esses motores são compactos, de menor custo que os servomotores, e apresentam alto torque de retenção. Entretanto o comando por pulsos elétricos significa que o controlador eletrônico (*driver*) deverá ser capaz de superar a indutância do motor a cada passo (*All About Circuits*, 2019) e a sua velocidade de rotação máxima fica limitada conforme Equação (1). Se o motor acionar uma correia, a sua velocidade linear estará relacionada com a de rotação por meio da relação de transmissão da polia conforme Equação (2).

|  |  |  |
| --- | --- | --- |
|  | *N =* | (1) |
|  |  | (2) |

Onde:

N - Rotação máxima do motor [];

U - Tensão elétrica da fonte de alimentação [V];

L - Indutância das bobinas do motor [H];

Im - Máxima corrente elétrica fornecida pelo *driver* [A];

ppr - Passos por revolução do motor [s.u.];

v - Velocidade linear [mm/s];

P - Passo da polia [mm];

Z - Número de dentes da polia.

Solenóides são atuadores lineares que obtém sua força de um campo magnético produzido por um eletroímã. Segundo Norton (2013), eles são ineficientes, apresentam pequenos cursos de movimentação, desenvolvem uma força variada ao longo do deslocamento, exigem altas correntes elétricas e aceitam pouca potência. Contudo, são de menor custo e tem resposta rápida. Costumam ser utilizados para acionamento indireto, ou seja, chaveando interruptores e válvulas.

Atuadores pneumáticos são os mais empregados em sistemas automatizados na indústria. Mais conhecidos como cilindros pneumáticos, esses elementos convertem a energia potencial pneumática em energia cinética pela expansão do ar comprimido (Fialho, 2011). A força exercida no avanço do cilindro é constante ao longo da trajetória, conforme a Equação (3), e é função do diâmetro do êmbolo (Prudente, 2015).

|  |  |  |
| --- | --- | --- |
|  | *F = P.* | (3) |

Onde:

F - Força exercida no avanço do cilindro pneumático [N];

P - Pressão do ar fornecido ao cilindro [Pa];

d - Diâmetro do êmbolo [m].

**3.2. Sistemas de controle**

Segundo Prudente (2015), um sistema automatizado é aquele capaz de agir sem intervenção humana, cumprindo tarefas e tomando decisões de acordo com uma determinada lógica. Essa lógica pode ser do tipo cabeado, determinada rigidamente pelas ligações elétricas entre atuadores e sensores e que não apresenta flexibilidade alguma, ou do tipo programável. A lógica programável é uma sequência de instruções (algoritmo) executada por elementos genéricos como relés, registradores, contadores, temporizadores e operadores matemáticos dentro de um controlador eletrônico, podendo atingir altos níveis de complexidade. É dita programável pois pode ser definida e modificada via *software*, sem a necessidade de um sistema eletrônico dedicado especificamente à tarefa. O tipo de equipamento aplicado nesses sistemas é chamado na indústria de controlador lógico programável, ou CLP. Trata-se de um computador voltado ao gerenciamento de processos em ambiente industrial, apresentando grande resistência à temperatura, umidade, vibrações, sujidades e demais adversidades comumente encontradas em ambiente fabril. Os CLPs costumam apresentar entradas e saídas analógicas e digitais em padrões industriais como 0-24 V e 4-20 mA, podendo ser prontamente integrados a instrumentos e atuadores dos mais variados tipos. Além disso, são programáveis por meio de diagramas de relés ou diagramas de blocos, incluindo blocos nativos para funções comuns como controle de velocidade, de temperatura e PID (proporcional - integral - derivativo). São, portanto, adequados ao uso sobre aspectos mecânicos e elétricos de processos, mas sem requerer profundo conhecimento de computação. Entretanto, nota-se que a robustez, confiabilidade e integrabilidade de um CLP acarretam um alto custo, assim como um *tradeoff* de poder de processamento, que costuma ser muito inferior ao de um computador convencional.

Existe uma abordagem alternativa e de baixo custo para implantação do controle programável: os microcontroladores. A principal diferença é que o CLP é um dispositivo do tipo “caixa preta”, onde basta que sejam identificadas as entradas e saídas do processo, conectadas aos bornes oferecidos no controlador e ele estará pronto para operar. Já um sistema microcontrolado baseia-se em um microprocessador no qual se constrói um sistema eletrônico dedicado ao condicionamento de sinais de acordo com a aplicação específica (Silva, 2015). Em outras palavras, trata-se de uma solução *ad hoc* onde se adota um microprocessador capaz de atender a demanda de processamento e se projeta os componentes intermediários. Visando reduzir justamente o tempo e a complexidade desse tipo projeto, tem-se buscado o desenvolvimento de plataformas padronizadas, como é o caso do Arduino.

O Arduino é uma plataforma de desenvolvimento de sistemas microcontrolados de baixo custo visando alta flexibilidade, que se apresenta na forma de uma série de modelos de placas de circuito. Cada modelo de placa Arduino é dotada de um microprocessador e um arranjo variado de circuitos periféricos como regulador de tensão de alimentação, entradas e saídas digitais em nível lógico 0-5 V, saídas PWM (*pulse width modulation*) e entradas analógicas, além de interface USB para conectividade com o computador e suporte a diversos protocolos de comunicação como serial padrão RS232, I2C e SPI. A programação é feita em linguagem C/C++ em um ambiente de desenvolvimento de fonte aberta (*open source*). Isso levou ao surgimento de uma comunidade de “hobbistas” (movimento *maker*) dedicada a essa plataforma, com *websites* e fóruns, onde se pode obter ou compartilhar bibliotecas de código e esquemáticos de circuitos elétricos gratuitamente.

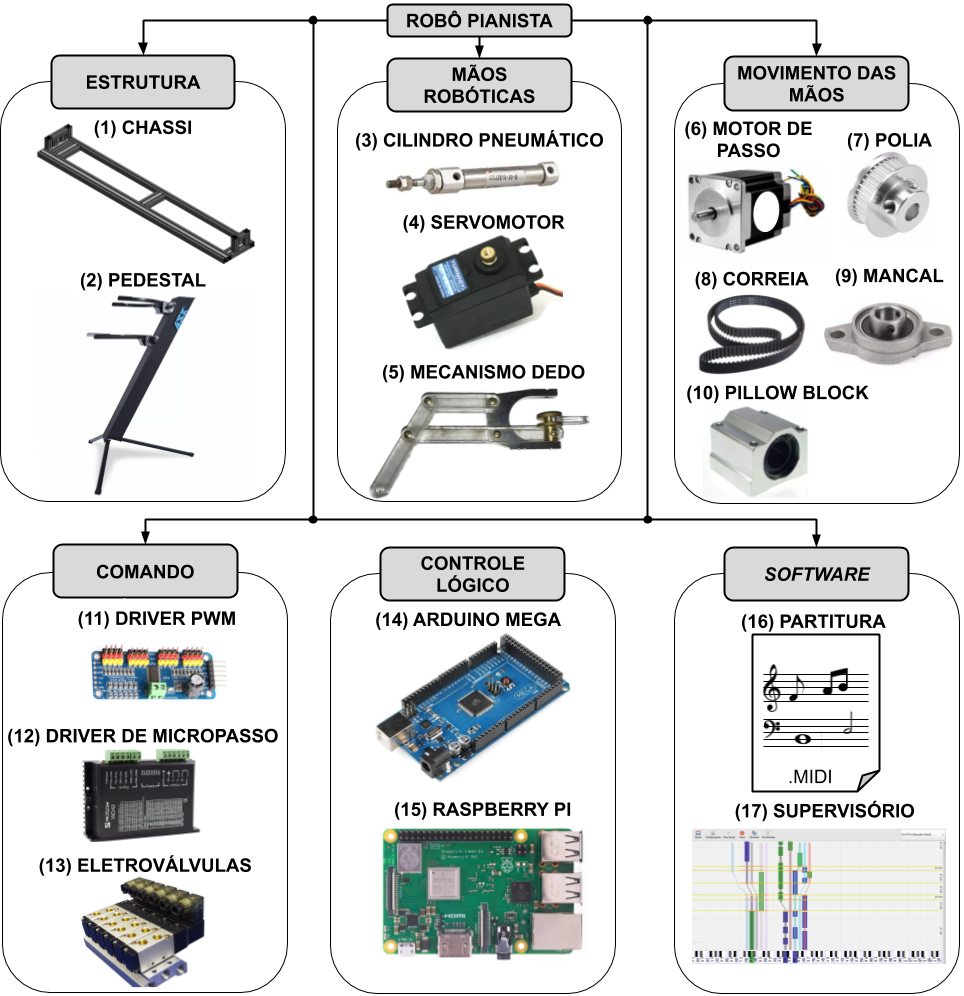
Conforme Silva (2015), o outro aspecto fundamental da plataforma é a modularidade. Há uma variedade de placas de circuito periféricas, chamadas *shields*, que podem ser expressamente acopladas ao Arduino para agregar funcionalidades. Alguns exemplos são: *drivers* de motores de passo, *drivers* de servomotores, telas, gerenciadores de bateria, transdutores de pressão e umidade, sensores capacitivos, indutivos e de temperatura, módulos GPS e de comunicação por radiofrequência, por *Bluetooth*, giroscópios, acelerômetros, entre outros.

O Arduino, contudo, pode ser insuficiente em termos de poder de processamento. Para aplicações onde tal aspecto é fundamental, é possível dividir a tarefa com um computador, deixando este com a função de ler arquivos, estabelecer e interpolar trajetórias, apresentar uma interface ao usuário e enviar comandos ao Arduino que ficará encarregado apenas de converter os comandos em sinais elétricos. Nesse contexto é relevante o *Raspberry Pi*. Trata-se de um computador completo na forma de uma única placa de circuito de pequenas dimensões (85,6 mm x 56,5 mm), o que o torna especialmente atraente para sistemas embarcados.

**4. MATERIAIS E MÉTODOS**

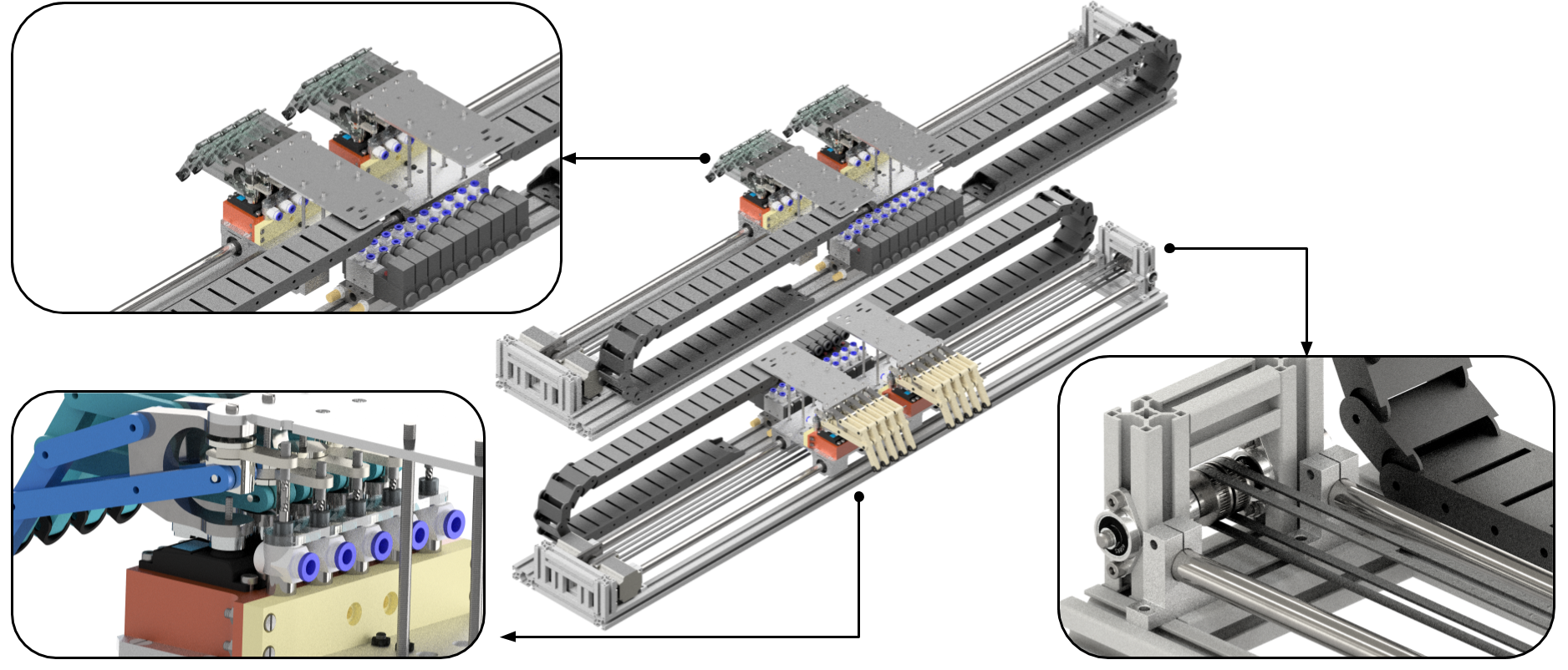
Dentre os modelos de robôs pianistas apresentados, o *design* com acionamento individual das teclas é descartado por não parecer natural em relação ao uso do instrumento. Optando-se por um modelo com mãos robóticas, o mais completo foi o de Lin (2015), dotado de 22 graus de liberdade, dos quais dois correspondem a movimentação linear de duas mãos com cinco dedos. Os dedos possuem outros dois graus de liberdade cada: um de pivotamento no plano do teclado para alcançar as teclas adjacentes e outro de movimento vertical para acionar a tecla.

O presente projeto, esquematizado na Figura 4.1, segue o mesmo conceito e possui, igualmente, 22 graus de liberdade. Desenvolvido para atender a diversos modelos de pianos e teclados, robô é instalado sobre um pedestal ajustável (item 2) que permite que o posicionamento vertical e horizontal sobre o teclado seja adaptado conforme o instrumento. O comprimento e a área útil de movimentação são dimensionados com base no teclado Yamaha PSR-E333. O robô é construído sobre um chassi de perfis estruturais de alumínio (item 1), escolhidos em razão da sua versatilidade, facilidade de montagem, leveza e alto momento de inércia da seção que proporciona rigidez contra flexão. Sobre o chassi é construído um sistema de movimentação linear horizontal pelo qual se deslocam as mãos robóticas. Esse sistema consiste de duas barras retificadas paralelas, que servem de guias lineares para os patins (item 10) no lado inferior das mãos, e dois motores de passo (item 6) que movimentam as mãos por correias (item 8).



**Figura 4.1.** Diagrama relacionando os subsistemas que compõe o robô pianista e os seus principais componentes.

Cada mão possui cinco dedos robóticos (item 5) que podem atuar verticalmente para pressionar a tecla abaixo deles e também rotacionar horizontalmente para alcançar teclas vizinhas. Esses dois movimentos dos dedos são acionados, respectivamente, por cilindros pneumáticos (item 3) e servomotores (item 4). O controle lógico do sistema é feito por um *software* (item 17) desenvolvido especialmente para o projeto, que é executado em um computador *Raspberry Pi 3 B+*, como o ilustrado na Figura 4.1 (item 15). Esse *software* pode carregar arquivos de partitura (item 16) em formato MIDI e exibir as notas em uma linha do tempo, sobre a qual o usuário prescreve pontos de passagem que definem trajetórias de movimentação harmônicas para as mãos e os dedos. Os pontos de passagem são enviados em tempo real para um Arduino Mega 2556 (item 14) que interpola entre eles, gerando pulsos para o *driver* de micropasso (item 12) que gerencia os motores de passo do sistema de movimentação linear. O Arduino também interpola os ângulos de rotação dos dedos e encaminha os valores para um *driver* de PWM (item 11) que comanda os servomotores. Os sinais de acionamento dos dedos são encaminhados do computador, via Arduino, para um bloco *manifold* de eletroválvulas pneumáticas (item 13) que pilotam os cilindros dos dedos. A Figura 4.2 apresenta uma visão global do projeto.



**Figura 4.2**. Modelagem em CAD do robô com destaque para alguns componentes.

**4.1. Projeto da mão robótica**

As mãos do robô são os conjuntos de cinco dedos e seus respectivos atuadores que se movem longitudinalmente ao longo do teclado conforme a posição mais adequada para se alcançar as teclas pertinentes à música naquele instante. Sua estrutura básica, conforme mostrado na Figura 4.3, é uma chapa de aço inoxidável 304 de espessura 0,8 mm (item 1), cortada a plasma, sob a qual são fixados os mancais lineares (item 2). Sobre a chapa há um bloco de plástico PLA (item 3), impresso em 3D, que sustenta os subconjuntos dos dedos apresentados na Figura 4.4. Cada subconjunto é constituído de: cilindro pneumático (item 4), servomotor (item 5), mecanismo de barras (item 6), par rotativo (item 7), garfo seguidor (item 8) e suporte dos dedos (item 9). Uma segunda chapa (item 10), do mesmo material, é montada acima dos dedos para aumentar a rigidez e permitir a fixação da esteira porta-cabos que comporta as linhas de ar comprimido. A Figura 4.3 apresenta como esses elementos são montados em um conjunto de mão robótica. A Figura 4.4 mostra em mais detalhes os componentes que compõem as submontagens dos dedos.

Cada dedo é capaz de se movimentar verticalmente sobre a tecla de modo a acioná-la, movimento produzido por meio de um cilindro pneumático de dupla-ação. Esse tipo de atuador apresenta alta confiabilidade no avanço e no retorno, exercendo uma mesma força conhecida em ambos os casos, e fácil controle por apresentar apenas duas posições. Para pilotar os cilindros um bloco *manifold* com dez eletroválvulas de retorno por mola de cinco vias e duas posições (5/2) é utilizado. A velocidade pode ser ajustada conforme necessário pelo fechamento parcial das válvulas de escape (silenciadores) e a força é determinada pela pressão, conforme Equação (2), que pode ser ajustada no regulador de pressão do compressor de ar. Cada cilindro deve ser capaz de fornecer ao menos 0,6 N, que é a força aproximada de atuação sobre uma tecla de piano (Zheng, 2009). Assim, o modelo especificado é o CDJ2B10-30-B, da marca SMC, com curso de 30 mm e êmbolo de diâmetro 10 mm. Considerando a pressão de trabalho de 0,8 MPa, o mesmo pode exercer na haste 6,4 kgf, conforme Equação (3).

|  |  |
| --- | --- |
|  |  |
| (a) |
|  |
| (b) | (c) |

**Figura 4.3**. Vistas do modelo desenvolvido em CAD do conjunto de mão robótica com indicação dos principais componentes: (a) lateral; (b) superior e (c) explodida.

|  |  |
| --- | --- |
|  | |
| (a) | (b) |

**Figura 4.4**. Mecanismo combinado de dois graus de liberdade dos dedos: (a) pressionar da tecla e (b) pivotamento lateral (direita).

Os dedos são espaçados 22 mm entre si, de maneira que cada um sobrepõe uma tecla, mas também precisam rotacionar no plano do teclado de modo a alcançar teclas adjacentes e aumentar o volume de trabalho de cada mão. Dentre os mecanismos considerados para essa finalidade estão o servomotor e o motor de passo. Ambos são muito precisos e confiáveis, porém o motor de passo apresenta a desvantagem de exigir que seja referenciado, já que atua em malha aberta. Por isso, a escolha ideal de atuador para o giro do dedo é o servomotor, que possui controle por posição absoluta e caixa de redução integrada. O servomotor adotado é o de modelo TGY-901D, da marca Turnigy. O modelo é adequado por ser o de mais baixo custo dentro da linha de servomotores pequenos e com engrenagens de metal, oferecendo maior durabilidade. O modelo especificado possui torque de 13 Kgf.cm, capaz de produzir elevada aceleração angular, e velocidade máxima de operação 0,14 s / 60º (rotaciona 60º em 0,14 segundo), podendo assim reposicionar os dedos rapidamente.

Para conciliar o movimento de giro e translação do dedo, a solução mecânica é utilizar um eixo único para ambos, ou seja, o eixo do servomotor deve coincidir com o eixo de translação do acionamento dos dedos. Os cilindros pneumáticos, por meio de um seguidor em acrílico, transmitem o movimento de translação a um par rotativo especialmente projetado e torneado em latão. Um par de parafusos fixados nesta junta provocam o movimento do mecanismo ao percorrerem um oblongo em uma das barras. O apertar da tecla por um dedo do tipo alavanca associado ao seu movimento lateral implicaria que a ponta do mesmo estivesse restrita a uma casca esférica. Essa geometria poderia permitir colisão do dedo com teclas próximas caso o ângulo de flexão fosse muito grande. Para evitar isso, cabe a metodologia de síntese para gerar um mecanismo de três barras onde uma exerce o papel do dedo e sua ponta percorre uma trajetória prescrita vertical, com comprimento 20 mm, para oprimir a tecla. A especificação desse curso baseia-se uma depressão mínima de 18 mm da tecla do teclado de referência. Utilizando-se o algoritmo de Norton (2011) é sintetizado o mecanismo da Figura 4.5. Para concretizar o mecanismo descrito, são utilizadas barras fabricadas em acrílico de espessura 4 mm pelo processo de corte laser. Parafusos M3x20 exercem a função de pino para união das juntas. Um suporte fabricado por corte plasma em aço inoxidável 304 tem a função de mancal e de realizar a união do mecanismo ao servomotor.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

**Figura 4.5**. (a) Síntese posições de um mecanismo de três barras com trajetória prescrita para a ponta do dedo do robô. (b) Diagrama do mecanismo com a fase de repouso representada em azul e fase de aperto de tecla em vermelho.

**4.2. Sistema de movimentação linear**

Alternativas para movimentação linear são a correia e o fuso. O fuso apresenta a melhor precisão e também o custo mais elevado. Já a movimentação por correia é de baixo custo e permite maiores velocidades. Além disso, ela é muito empregada em impressoras 3D, havendo assim uma ampla variedade de conjuntos de polia, correia, mancais, guias e rolamentos disponíveis a um custo acessível, o que justifica a escolha para o projeto. O modelo especificado é o GT-2, com passo de 2 mm e largura 6 mm. As polias, que têm 40 dentes, são fixadas diretamente aos motores de passo em uma extremidade da estrutura. Na outra extremidade, são suportadas por mancais com rolamentos modelo Kfl08, com eixo de 8 mm. Os motores de passo são da marca Wotiom, modelo WS23-0150-30-4, com torque de 15 kgf.cm, corrente máxima de 3 A, indutância 2,2 mH e resistência elétrica de 1,3 Ω. Alimentados por uma fonte de 24 V, podem atingir 545 rotações por minuto. Considerando a relação de transmissão, a velocidade máxima das mãos é de 0,7 m/s, ou 33 teclas/s.

As guias lineares são capazes de suportar o pesos das mãos de modo que a flecha da flexão seja suficientemente pequena para considerar que as mãos estejam a uma distância constante do teclado. Por meio de uma modelagem no *software* “Ftool”, supondo as duas mãos próximas do centro do vão (pior caso), foi dimensionado um par de barras de aço SAE 1055, temperado e revenido (dureza 64 HRC), comprimento 1456 mm, diâmetro 16 mm e acabamento retificado H6. A flecha resultante, de 1,5 mm, atende ao requisito. A Figura 4.6. exibe uma visão do sistema e seus principais componentes.

|  |  |  |
| --- | --- | --- |
| (a) | | |
| (b) | (c) | (d) |

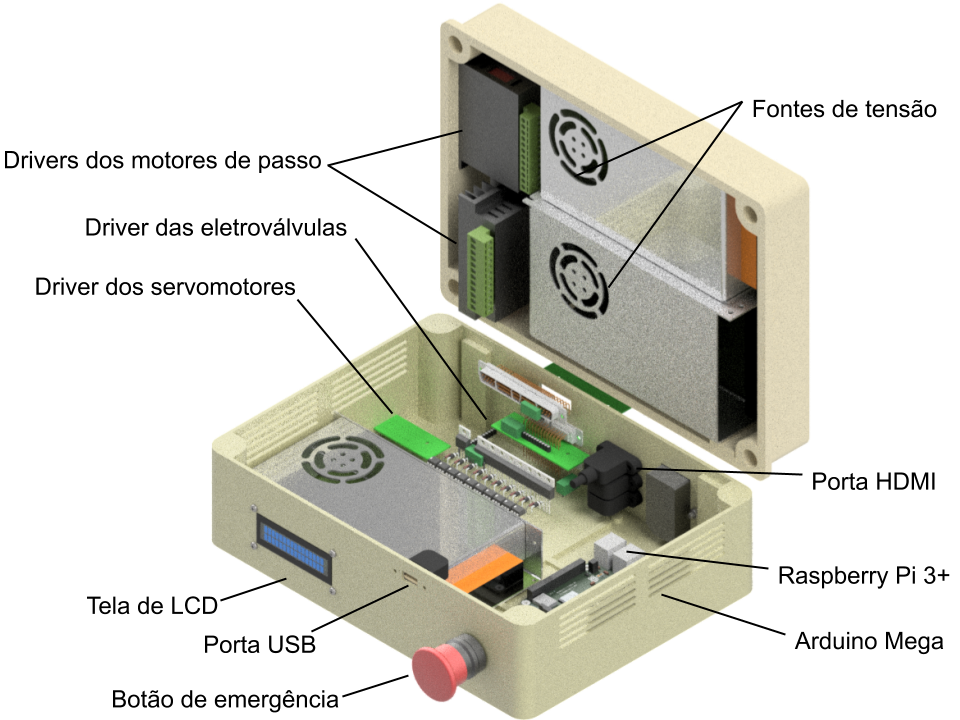
**Figura 4.6**. (a) Vista superior do modelo em CAD do robô com indicação dos elementos do sistema de movimentação linear das mãos; (b) polia GT-2 de 40 dentes; (c) mancal com rolamento KFL08 e (d) motor de passo WS23-0150-30-4.

**4.3. Sistema eletrônico**

A automação do robô demanda o emprego de diversos dispositivos eletrônicos. O programa de supervisório é executado em um computador Raspberry Pi 3 modelo B+, que envia instruções para um Arduino Mega 2560, responsável pelo controle efetivo dos demais componentes. Ambos os dispositivos são alimentados por uma fonte de tensão regulada de 5 V, demandando até 500 mA cada.

Os servomotores utilizados para rotacionar os dedos recebem um sinal modulado em largura de pulso (PWM), ou seja, uma onda quadrada cujo percentual do período em nível lógico alto é proporcional ao ângulo desejado no eixo. Apesar de o Arduino ser capaz de gerar esse tipo de sinal, essa tarefa é muito onerosa em termos de tempo de processamento. De modo a não sobrecarregá-lo, utiliza-se um *driver* de PWM dedicado. Com interface de comunicação I2C, o PCA9625 recebe do microcontrolador os ângulos interpolados dos dedos, em tempo real, e gera o sinal apropriado. Esse modelo apresenta um gerador de funções com resolução de 12 bits, podendo assim comandar os servomotores com uma resolução de 0,09º. Os dez servomotores também são alimentados com a mesma tensão elétrica de 5 V, porém a corrente é altamente dependente do torque exercido. Considerando-se o torque e a velocidade máximos, pode-se calcular a potência e isolar a corrente de pico: 1,6 A cada.

Semelhantemente, os motores de passo também requerem um *driver* dedicado. Esse tipo de motor possui um rotor dentado e duas bobinas. Quando o Arduino envia um pulso ao *driver*, este se encarrega de energizar e desenergizar as bobinas em uma sequência específica para produzir no rotor um giro de um dente (um passo) ou uma fração desse ângulo (micropasso). O *driver* também gerencia a tensão fornecida para que, apesar da indutância das bobinas, a corrente seja sempre máxima e altos torque e velocidades sejam assim alcançados. O *driver* que faz conjunto com o motor especificado é o Wotiom WD-2404, que opera com tensão elétrica de 24 V, possui corrente máxima ajustável até 4 A e resolução de micropasso de até 1/32.



**Figura 4.7.** Modelo em CAD do gabinete elétrico com indicação dos principais componentes.

As saídas digitais do Arduino trabalham em nível lógico 5 V e corrente máxima de 50 mA. As eletroválvulas funcionam com alimentação de 12 V e drenam 400 mA cada. Portanto, é necessário uma mudança de nível lógico entre ambos. Nesse caso, utiliza-se um um *driver* com optoacopladores e chaveamento com transistores Darlington.

São necessárias três tensões elétricas diferentes para alimentar todos componentes do robô. Pela compacticidade, confiabilidade, baixo custo e facilidade de obtenção, optou-se por utilizar fontes de tensão chaveadas. Para os dois motores de passo, utiliza-se uma fonte de 24 V e 12,5 A. Para as eletroválvulas, uma fonte de 12 V e 20 A. Para os servomotores, Arduino e Raspberry Pi, uma fonte de 5 V e 20 A é aplicada.

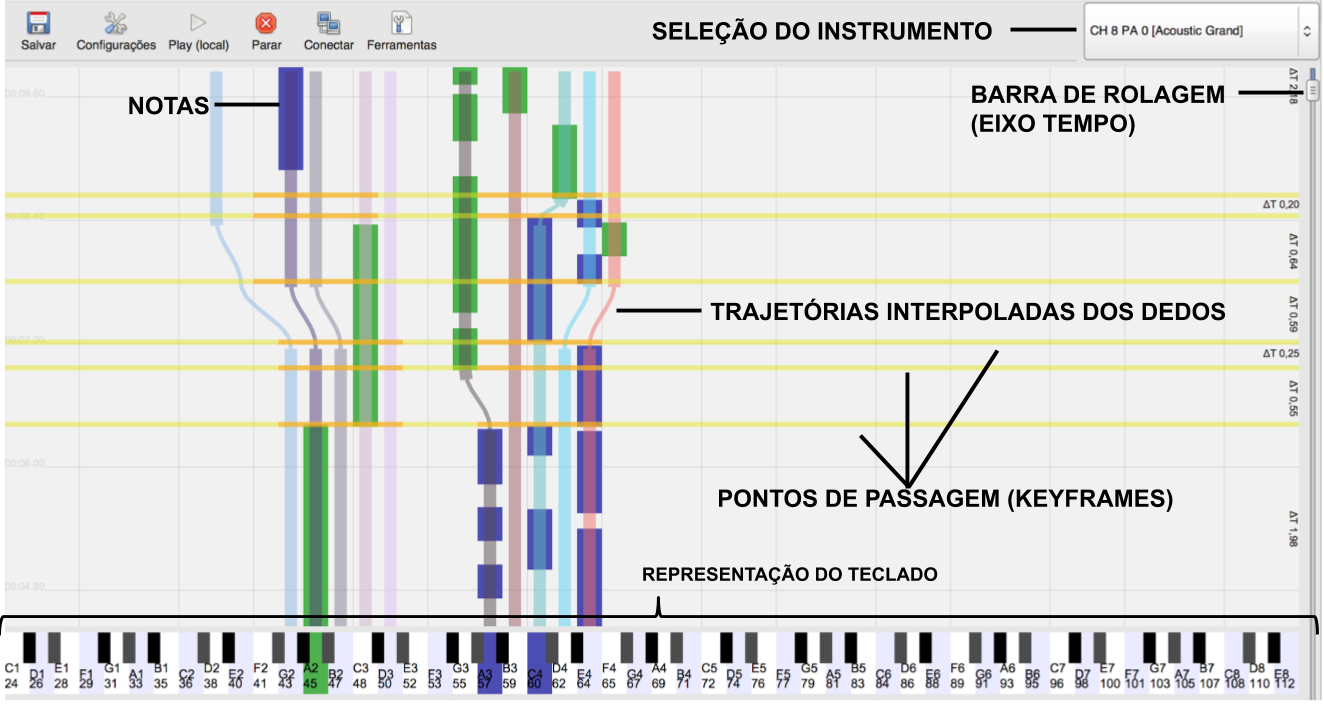
Além disso, o sistema conta com um botão de emergência para possibilitar parada imediata caso haja uma colisão e uma tela de cristal líquido para exibir mensagens pertinentes ou o nome da música em execução. Para acomodar todos esses componentes, é utilizado um gabinete fabricado a partir de uma caixa de montagem em PVC, marca Steck, tamanho 310x230x125 mm. A caixa possui aberturas, fabricadas por corte a laser, para passagem de ar de ventilação e para fixação de componentes como o visor, botões, portas USB, HDMI, conector de tomada e dos cabos que se ligam ao robô. O diagrama elétrico completo pode ser consultado no Anexo B, e o modelo CAD do gabinete na Figura 4.7.

**4.4. Projeto do software de controle**

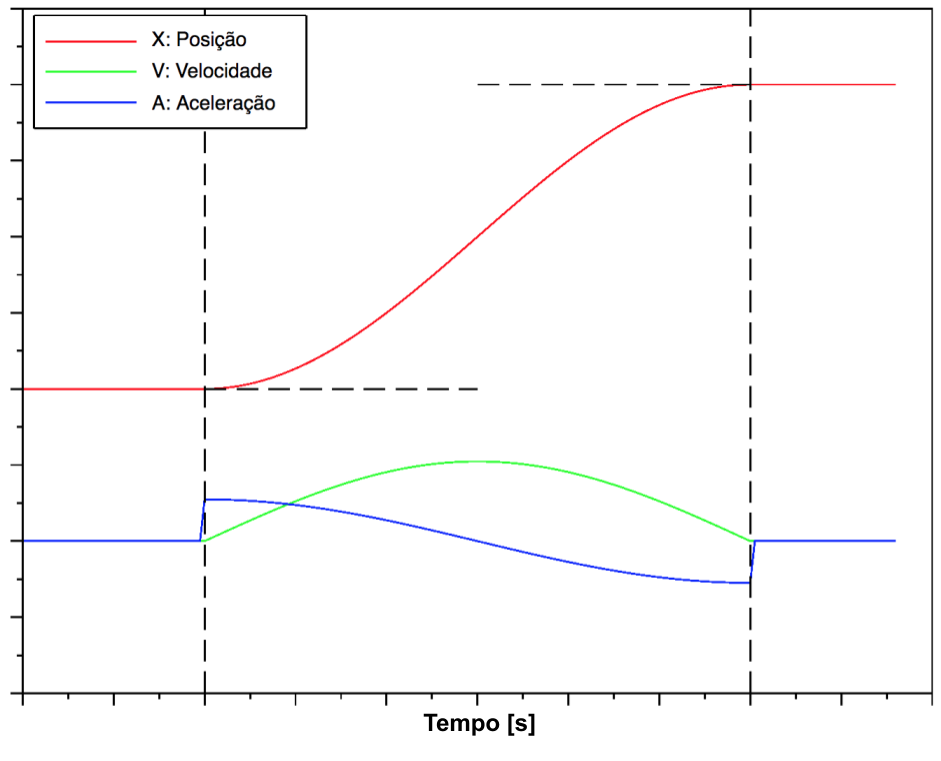
O *software* do robô tem duas funções distintas. O primeira delas é, dada uma partitura em MIDI, gerar um programa que especifica como cada membro deve se posicionar, a cada instante de tempo, para que todas as notas possam ser tocadas no momento e com a duração certos. O posicionamento deve ser tal que o robô possa se realocar para as notas seguintes em tempo hábil, executando o menor número possível de movimentos. Portanto, as limitações geométricas e dinâmicas do sistema, assim como a sequencialidade das notas, devem ser consideradas préviamente. É um problema discreto e de múltiplas soluções, já que há diversas estratégias possíveis para cumprir esse posicionamento. Lin (2014) demonstra um algoritmo capaz de automatizar a geração desse programa, porém nesse projeto é adotada uma solução mais simples: a programação manual.

Para realizar a programação do robô utiliza-se um aplicativo desenvolvido em C#, na plataforma Mono, que é executado no Raspberry Pi dentro do gabinete da unidade de controle. O mesmo aplicativo, cuja interface é explicada na Figura 4.8, também atua como supervisório, exibindo status, coordenadas e gerenciando o *hardware*. A programação manual consiste de carregar uma partitura em MIDI, que é então exibida em uma linha do tempo, sobre a qual o usuário marca os pontos de passagem correspondentes a estratégia que ele definir. As trajetórias das mãos e dos dedos são geradas por interpolação entre os pontos informados e apresentadas sobrepostas às notas da mesma linha do tempo.

A segunda função é o controle propriamente dito dos atuadores a partir da programação realizada. Durante a performance da música, esses pontos de passagem são enviados sequencialmente ao Arduino por comunicação serial, com os pacotes enquadrados e formatados no padrão RFC 1055. O microcontrolador avalia uma função de interpolação harmônica continuamente e produz os movimentos correspondentes nos motores de passo nos servomotores. Assim, a ponta do dedo percorre uma trajetória que, sendo a soma dos dois movimentos, também é descrita por um perfil harmônico. Além disso, a velocidade e aceleração também são funções harmônicas e, portanto, contínuas e suaves. No caso da aceleração, isso é especialmente desejável, pois o diagrama de espectro da transformada de Fourier passa a consistir de bandas discretas, cuja probabilidade de coincidir com a frequência de ressonância do robô é muito baixa. A Figura 4.9 mostra os perfis de posição, velocidade e aceleração descritos pelas Equações (4), (5) e (6), respectivamente.



**Figura 4.8**. Tela do aplicativo supervisório e de programação de estratégias do robô.



**Figura 4.9**. Perfil harmônico de posição, velocidade e aceleração em função do tempo.

|  |  |  |
| --- | --- | --- |
|  | *X = K1 + K2.sin(ωt + φ)* | (4) |
|  | *V = ωK2.cos(ωt + φ)* | (5) |
|  | *A = -ω²K2.sin(ωt + φ)* | (6) |

Onde:

X - Posição instantânea [teclas, pulsos ou graus];

V - Velocidade instantânea [teclas/s, pulsos/s ou graus/s];

A - Aceleração instantânea [teclas/s², pulsos/s², graus/s²];

t - Variável livre tempo [s];

ω - Velocidade angular [rad/s];

φ -ângulo de fase [rad];

K1 - coeficiente linear [teclas, pulsos ou graus];

K2 - amplitude do movimento [teclas, pulsos ou graus].

O movimento fica completamente determinado por meio de quatro constantes (ω, φ, K1,K2) que são computadas pelo Arduino para cada interpolação entre dois pontos de passagem, utilizando as equações (7), (8), (9) e (10):

|  |  |  |
| --- | --- | --- |
|  | *K1 = 0,5.(XF + X0)* | (7) |
|  | *K2 = 0,5.(X0-XF)* | (8) |
|  | *ω =* | (9) |
|  | *φ =* | (10) |

Onde:

X0 - Posição inicial [teclas, pulsos ou graus];

XF - Posição final [teclas, pulsos ou graus];

T0 - Tempo inicial [s];

TF - Tempo final [s].

Isso é possível conhecendo-se os dois pontos (posição e tempo iniciais e finais) além de duas condições de contorno: velocidade nula na partida e na chegada. Os pontos de passagem são definidos na programação das trajetórias pelo usuário do *software* supervisório em unidade de teclas. Convenientemente, os dados já são enviados convertidos nas unidades pertinentes a partir do dimensional do robô, conforme equações (11) e (12).

|  |  |  |
| --- | --- | --- |
|  | P = | (11) |

Onde:

P - Deslocamento em passos do motor;

ppr - Passos por revolução do *driver* de micropasso;

L - Largura da tecla [mm];

∆X - Deslocamento da mão [teclas];

Z - Número de dentes da polia motora;

S - Passo da correia [mm].

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

Onde:

θ - Ângulo de giro do dedo;

X - Deslocamento da ponta do dedo sobre o teclado em relação à referência [teclas];

L - Largura da tecla [mm];

R - Raio de giro dado pelo comprimento do dedo.

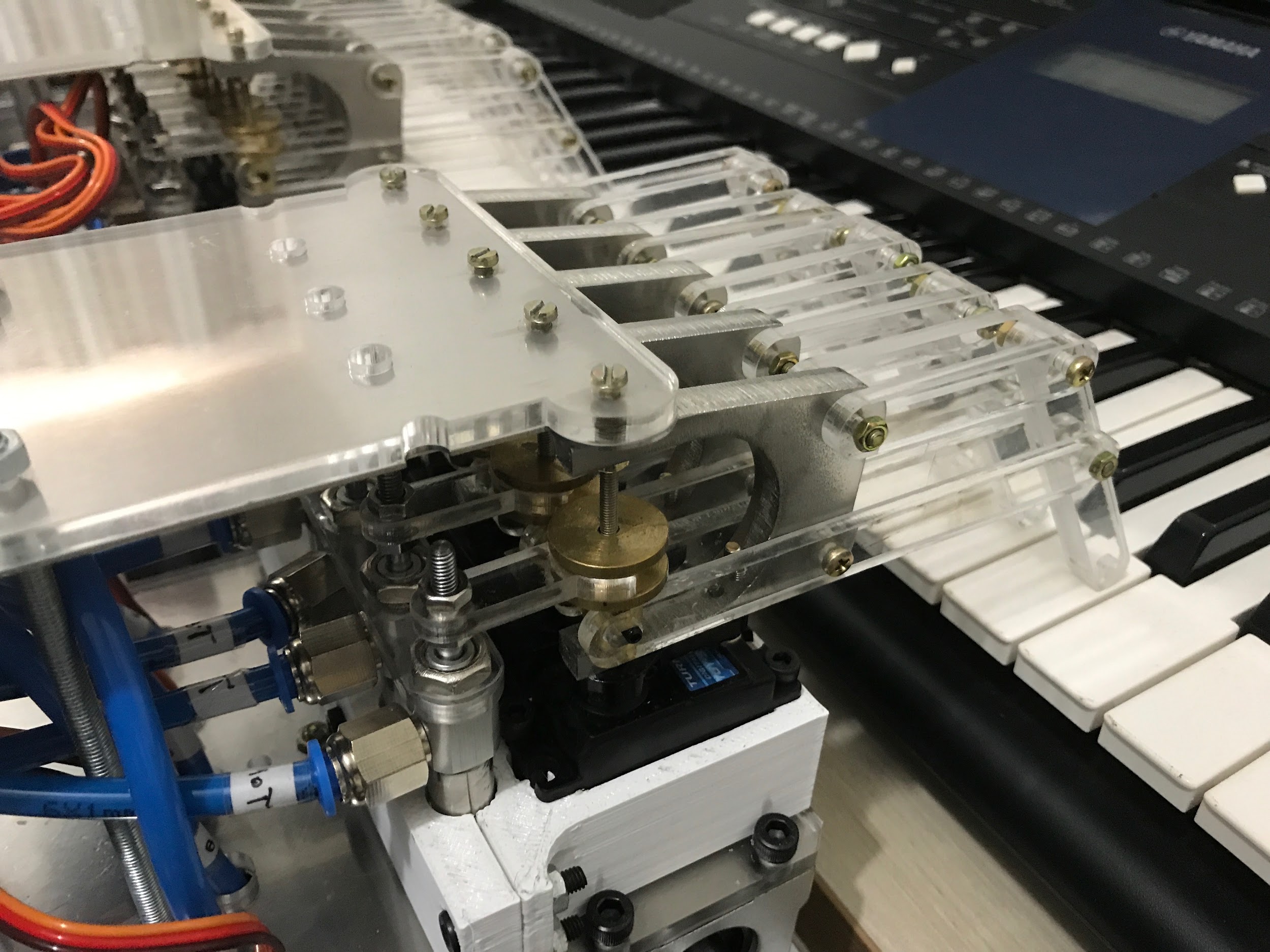
**5. RESULTADOS E DISCUSSÕES**

O mecanismo de dedo, cuja montagem é mostrada na Figura 5.1, funcionou conforme idealizado no projeto. A força gerada pelo cilindro pneumático se mostrou suficiente e pôde ser transmitida pelo garfo seguidor através da cruzeta e pelo pino, percorrendo o oblongo na barra, acionando o mecanismo e pressionando a tecla subjacente. Foi necessário o uso de cola trava roscas para manter atarraxadas as porcas nos parafusos das juntas, uma vez que um torque excessivo limitaria a mobilidade. Não houveram folgas que prejudicassem a precisão ou a rigidez do conjunto.



**Figura 5.1**. Submontagem de um conjunto de dedo, contendo mecanismo de barras, suporte (mancal), par rotativo (cruzeta) e garfo seguidor.

Foi necessário desbastar as faces internas das cavidades do suporte dos motores para eliminar interferências razão das imprecisões inerentes ao processo de fabricação por impressão em 3D. O alinhamento longitudinal dos dedos com as teclas do teclado musical foi adequado, entretanto o alinhamento angular, que só é possível em intervalos de 14,4° devido aos 25 rasgos no eixo do servomotor, precisou ser corrigido pelo *software*. A rotação dos dedos alcançou a amplitude esperada de duas teclas para cada lado, sem colisão com os dedos vizinhos ou com as teclas adjacentes, quando no ponto inferior, comprovando o conceito do mecanismo proposto. Os testes mostraram que o curso vertical especificado para a ponta do dedo foi adequado para pressionar as teclas, mesmo com um espaço de 5 mm entre a posição de repouso do dedo e o instrumento. A força aplicada pelos atuadores pneumáticos também foi suficiente. A Figura 5.2. mostra o conjunto de mão montado no robô, sobreposto a um teclado musical.



**Figura 5.2.** Conjunto montado de uma mão do robô sobre o teclado musical.

Os primeiros testes do sistema de movimentação linear, realizados com perfil de velocidade constante, apresentaram intensa vibração das mãos e sobretudo das correias. Com a implementação do controle harmônico do movimento, notou-se significativa redução das vibrações, como esperado, porém ainda acima do aceitável quando em baixas velocidades instantâneas. Nessa condição, o intervalo de tempo entre os passos do motor aumenta o suficiente para manifestar descontinuidades na tração aplicada e provocar acelerações e desacelerações no movimento das mãos, induzindo vibrações. Visando absorver a intermitência do acionamento do motor, o *driver* de micropasso foi reconfigurado de 200 para 800 passos/revolução. Com esse ajuste o movimento tornou-se contínuo mesmo em baixas velocidades. Entretanto, nos casos com deslocamento assimétrico das mãos prevaleceu uma leve oscilação do pedestal que sustenta o robô. Dois modelos diferentes de pedestais destinados a teclados musicais foram testados, apresentando resultado semelhante. A Figura 5.3 exibe as duas extremidades do sistema de movimentação horizontal.

|  |  |
| --- | --- |
| (a) | (b) |

**Figura 5.3.** Conjunto de movimentação linear das mãos: (a) extremidade motriz, ênfase nos motores de passo e polias motoras; (b) extremidade oposta, ênfase nas polias movidas e mancais.

A unidade de controle foi montada conforme o projeto realizado em CAD e todos os eletrônicos puderam ser dispostos dentro do gabinete, como mostra a Figura 5.4. Nos diversos testes de reprodução de músicas realizados com o robô, nenhum componente dotado de medidas de proteção contra superaquecimento (fontes e *drivers*) acionou os respectivos alarmes. Todos os atuadores puderam ser acionados simultaneamente, indicando que não houve subdimensionamento na potência das fontes. Não houveram problemas de compatibilidade entre os diversos *drivers* e o Arduino.

|  |  |
| --- | --- |
| (a) | (b) |
| (c) | (d) |

**Figura 5.4.** Unidade de controle do robô: (a) vista frontal e (b) vista posterior da parte externa; (c) componentes montados na parte interna da base e (d) da tampa.

O *software* de controle desenvolvido atendeu as funcionalidades exigidas pelo projeto. O carregamento de músicas no formato MIDI e exibição das notas na linha do tempo resultou em um método rápido e prático de introduzir dados musicais no ambiente de programação. A interface gráfica foi elaborada com diversos recursos para facilitar a programação manual pelo operador, conforme passo-a-passo exemplificado na Figura 5.5. As barras horizontais indicam os pontos de passagem e as curvas na vertical representam a trajetória dos dedos entre eles, assumindo espessura fina quando em movimento ou larga quando em repouso, para indicar quais dedos estão instantâneamente aptos a acionar uma tecla. Contudo, notou-se que programar o robô desta maneira é uma tarefa que consome tempo. Elaborar a estratégia de movimentação não é trivial, necessitando várias iterações e ajustes nos inúmeros pontos de passagem para se obter uma sequência satisfatória. A implementação de um algoritmo de programação automática é, nesse contexto, uma oportunidade de melhoria. Outro recurso que foi implementado no *software* é a transposição de escala musical. Essa técnica permite deslocar a altura de todas as notas seguindo um padrão específico de modo a transformar a partitura em outra equivalente. No processo são eliminados os sustenidos e bemóis (teclas pretas), que não são acessíveis pelo robô, contornando essa limitação mecânica. O resultado final da construção do protótipo é mostrado na Figura 5.6.

|  |  |
| --- | --- |
| (a) | (b) |
| (c) | (d) |

**Figura 5.5.** Programação de uma música: (a) condição inicial; (b) inserção do primeiro ponto de passagem; (c) e (d) trajetórias obtidas com vários pontos.



**Figura 5.6**. Robô pianista posicionado no pedestal sobre um teclado musical.

**6. CONCLUSÃO**

Foi projetado e construído um robô com duas mãos móveis e dez dedos rotativos, capaz de tocar o piano ou o teclado musical de maneira semelhante a uma pessoa. O projeto abrangeu o modelagem mecânica em CAD de um chassi, de um sistema de movimentação horizontal, das mãos robóticas e a síntese de um mecanismo para os dedos. Foram abordadas a especificação e escolha dos atuadores e demais componentes, inclusive dos eletrônicos responsáveis pela automação e controle lógico, bem como o desenvolvimento de um *software* supervisório para controle e programação das músicas tocadas.

A metodologia de programação manual foi suficiente para os testes de funcionamento do robô. Entretanto, para expandir seu repertório de maneira a possibilitar apresentações a um público, convém o desenvolvimento de um algoritmo de programação automática. O pedestal que sustenta o robô sobre o instrumento teve sua importância subestimada. O desenvolvimento de um suporte mais adequado visando melhorar a estabilidade do robô e facilitar sua instalação é uma outra possibilidade de projeto futuro.

O protótipo como um todo apresentou funcionamento satisfatório, sendo capaz de reproduzir os movimentos programados com a velocidade e destreza necessárias para tocar o instrumento, indicando que a metodologia de projeto proposta foi bem sucedida e os dimensionamentos foram adequados. Apesar de estar restrito às teclas brancas do teclado e não possuir controle de intensidade sobre o pressionar das teclas, os seus 22 graus de liberdade provém versatilidade suficiente para atender razoavelmente um amplo espectro de músicas, conferindo assim ao robô o potencial de entretenimento desejado.

**Agradecimentos**

Agradeço ao meu orientador pelo apoio e acompanhamento. Estendo também o agradecimento aos demais professores da Escola Politécnica que de alguma maneira contribuíram à realização do projeto, à Universidade pelos materiais e espaço cedidos para construção do protótipo e aos técnicos do Labfab que se dispuseram a auxiliar na fabricação de peças.

**7. REFERÊNCIAS BIBLIOGRÁFICAS**

BATULA, Alyssa M.; KIM, Youngmoo E. Development of a mini-humanoid pianist. In: **2010 10TH IEEE-RAS INTERNATIONAL CONFERENCE ON HUMANOID ROBOTS** (HUMANOIDS 2010), dez. 2010, [S.l.]: IEEE, dez. 2010. Disponível em: <http://dx.doi.org/10.1109/ICHR.2010.5686330>.

FIALHO, Arivelto Bustamante. **Automação Pneumática: Projetos, Dimensionamento e Análise de Circuitos**. São Paulo: Érica, 2011.

International Midi Association. **Standard Midi File Format Spec**. Disponível em: <https://www.midi.org/specifications/item/the-midi-1-0-specification>. Acesso em: 20 abr. 2019.

JEN-CHANG, Lin et al. Design of Piano -playing Robotic Hand. **IAES International Journal of Robotics and Automation (IJRA)**, v. 3, n. 2, 1 jun. 2014. Disponível em: <http://dx.doi.org/10.11591/ijra.v3i2.4269>.

KAJITANI, Makoto. Development of Musician Robots. **Journal of Robotics and Mechatronics**, v. 1, n. 3, p. 254–255, 20 out. 1989. Disponível em: <http://dx.doi.org/10.20965/jrm.1989.p0254>.

KAPUR, A; A History of Robotic Musical Instruments. In: **International Computer Music Conference**: 2005, Barcelona.

LI, Yen-Fang; CHI-YI LAI. Intelligent algorithm for music playing robot: Applied to the anthropomorphic piano robot control. In: **2014 IEEE 23RD INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS (ISIE)**, jun. 2014, [S.l.]: IEEE, jun. 2014. Disponível em: <http://dx.doi.org/10.1109/ISIE.2014.6864843>.

LIN, Jen-Chang et al. Electronic piano playing robot. In: **2010 INTERNATIONAL SYMPOSIUM ON COMPUTER, COMMUNICATION, CONTROL AND AUTOMATION** (3CA), maio 2010, [S.l.]: IEEE, maio 2010. Disponível em: <http://dx.doi.org/10.1109/3CA.2010.5533457>.

MED, B. **Teoria da Música**. Brasília: Musimed, 1996.

NORTON, Robert L. **Cinemática e dinâmica dos mecanismos**. Porto Alegre: AMGH, 2011.

NORTON, Robert L. **Projeto de máquinas: uma abordagem integrada**. Porto Alegre: Bookman, 2013.

PRUDENTE, Francesco. **Automação industrial PLC: teoria e aplicações**. Rio de Janeiro: LTC, 2015.

RAY, C.; MONDADA, F.; SIEGWART, R. What do people expect from robots? In: **2008 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS**, set. 2008, [S.l.]: IEEE, set. 2008. Disponível em: <http://dx.doi.org/10.1109/IROS.2008.4650714>.

SILVA, Rodrigo Adamshuk; STEVAN Jr.,Sergio Luiz. **Automação e instrumentação industrial com arduino: teoria e projetos**. São Paulo: Érica, 2015.

SOLIS, Jorge et al. Development of anthropomorphic musical performance robots: From understanding the nature of music performance to its application to entertainment robotics. In: **2009 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS** (IROS 2009), out. 2009, [S.l.]: IEEE, out. 2009. Disponível em: <http://dx.doi.org/10.1109/IROS.2009.5354547>.

SOLIS, Jorge et al. Learning to play the flute with an anthropomorphic flutist robot. In: Proceedings of International Computer Music Conference (ICMC 2004). Disponível em: <http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-21019>

WILLIAMSON, Matthew. Robot Arm Control Exploiting Natural Dynamics. 1999. Dissertação (PhD) - Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1999.

ZHANG, Dan et al. Design and analysis of a piano playing robot. In: **2009 INTERNATIONAL CONFERENCE ON INFORMATION AND AUTOMATION** (ICIA), jun. 2009, [S.l.]: IEEE, jun. 2009. Disponível em: <http://dx.doi.org/10.1109/ICINFA.2009.5205022>.

**ANEXO A**

**Anexo A1.** Códigos MIDI para notas musicais (*International Midi Association*, 1996).

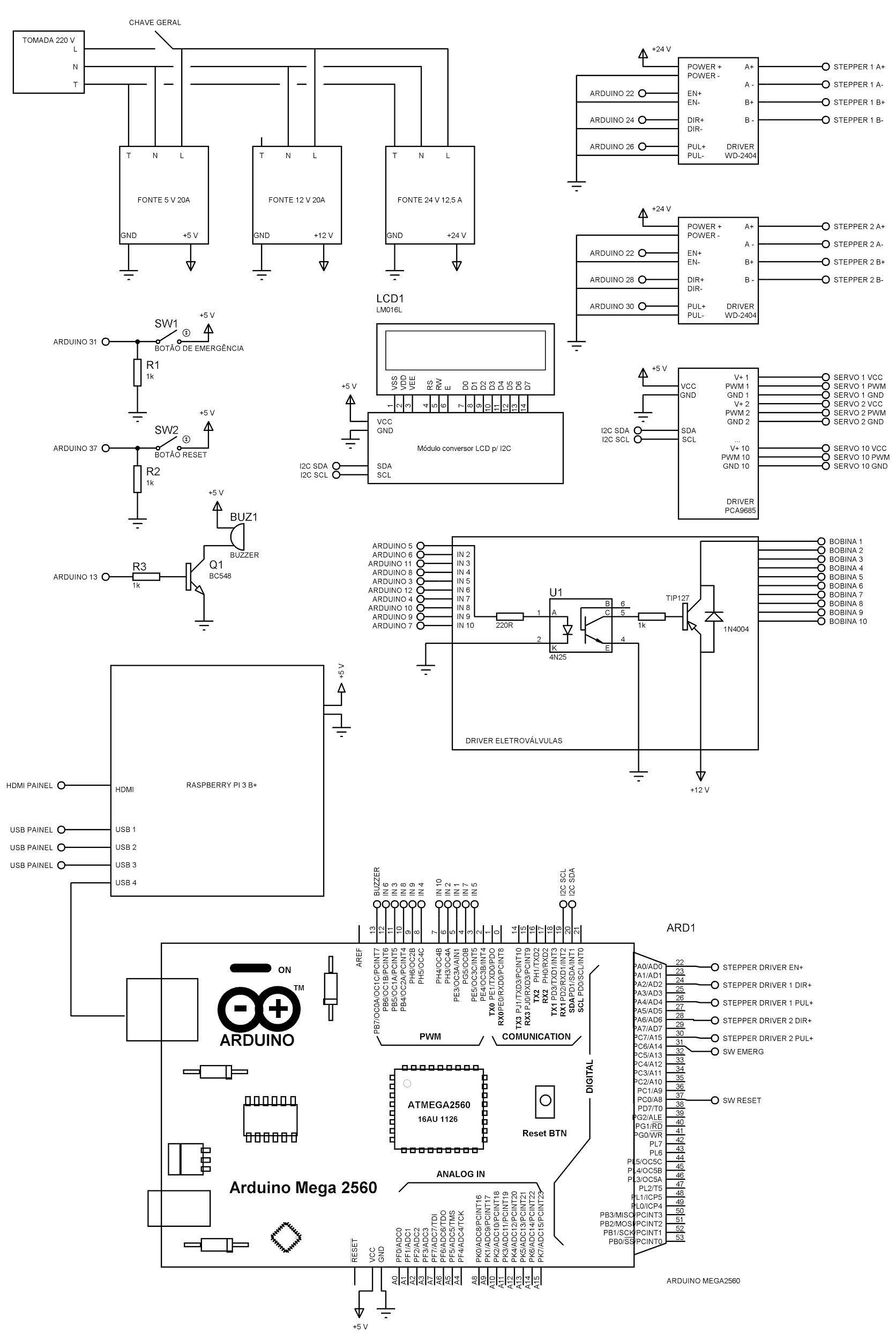
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Oitava** | **Código MIDI da nota** | | | | | | | | | | | |
|  | **C** | **C#** | **D** | **D#** | **E** | **F** | **F#** | **G** | **G#** | **A** | **A#** | **B** |
| **-1** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| **0** | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| **1** | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |
| **2** | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| **3** | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
| **4** | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| **5** | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 |
| **6** | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| **7** | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 |
| **8** | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| **9** | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |  |  |  |  |

**Anexo A2.** Diagrama de blocos exemplificando a estrutura de um arquivo MIDI.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Arquivo “9º Sinfonia.mid”**   |  | | --- | | **Cabeçalho** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Faixa 1**   |  |  |  | | --- | --- | --- | | Meta evento | Nome da música | 9º Sinfonia | | Meta evento | Velocidade (metrônomo) | 120 Batidas/segundo (BPS) | | Meta evento: Seleciona instrumento | Piano (1) | Canal 0 | | Meta evento: Seleciona instrumento | Violino (41) | Canal 0 |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Tempo (B)** | **Evento** | **Canal** | **Nota** | **Intensidade (%)** | **Descrição** | | 120 | Liga nota | 0 | 60 | 100 | Após 1 s começa nota Dó4 no piano com intensidade 100% | | 0 | Liga nota | 1 | 75 | 50 | Simultaneamente começa nota Ré5 no violino com intensidade 50% | | 240 | Desl. nota | 0 | 60 | - | Após 2 s o piano para de tocar a nota Dó4 | | |

**ANEXO B**

Diagrama elétrico do robô



**ANEXO C**

Circuito pneumático responsável pelo acionamento dos dedos

|  |  |
| --- | --- |
| DEDO 1 | DEDO 6 |
| DEDO 2 | DEDO 7 |
| DEDO 3 | DEDO 8 |
| DEDO 4 | DEDO 9 |
| DEDO 5 | DEDO 10 |

**ANEXO D**

Trechos de código do *software* de controle do robô

**ANEXO D1.** Software executado no Arduino.

MyComms.cpp

Comunicação entre Arduino e Raspberry Pi utilizando para enquadramento dos pacotes o protocolo SLIP (Serial Line Internet Protocol), descrito na norma RFC 1055.

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | const byte SLIP\_END = 0xC0;  const byte SLIP\_ESC = 0xDB;  const byte SLIP\_ESC\_END = 0xDC;  const byte SLIP\_ESC\_ESC = 0xDE;  byte serialBuffer[128];  int bufferPosition = 0;  int frame\_size;  int frame\_read\_position;  void SetupConnection() {  **Serial**.begin(57600, SERIAL\_8N2);  }  bool SerialTalk() {  if (**Serial**.available() <= 0)  return false;  byte incoming = **Serial**.read();  switch (incoming) {  case SLIP\_END:  frame\_read\_position = 0;  frame\_size = bufferPosition;  bufferPosition = 0;    return (frame\_size > 0);  break;    case SLIP\_ESC:  incoming = **Serial**.read();  if (incoming == SLIP\_ESC\_END)  incoming = SLIP\_END;  else if (incoming == SLIP\_ESC\_ESC)  incoming = SLIP\_ESC;  else {  **Serial**.println("Protocol violation");  }    default:  serialBuffer[bufferPosition] = incoming;  bufferPosition++;  break;  }  return false;  } | 50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | bool CommReadBool() {  return (serialBuffer[frame\_read\_position++] == 0xFF);  }  int CommReadInt() {  int i = 256 \* serialBuffer[frame\_read\_position+1] + serialBuffer[frame\_read\_position];  frame\_read\_position += 2;  return i;  }  int CommFrameLength() {  return frame\_size;  }  byte CommReadByte() {  frame\_read\_position++;  return serialBuffer[frame\_read\_position-1];  } |

MyStepper.cpp

Comando dos motores de passo seguindo perfil de velocidades harmônico. São geradas interrupções usando Timer1 e, em algumas dessas interrupções, um pulso é emitido. O intervalo entre os pulsos é calculado em termos da posição instantânea.

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74 | #include "TimerOne.h"  StructSTEPPER STEPPERS[2];  Output<STEPPER\_ENABLE> EN = LOW;  float MotionDuration;  void SetupSteppers()  {  EN = true; // Disable Outputs  STEPPERS[0].PUL = new OutputPin(STEPPER1\_PUL);  STEPPERS[0].DIR = new OutputPin(STEPPER1\_DIR);  STEPPERS[0].bInverted = false;  STEPPERS[0].TargetPosition = 0;  STEPPERS[0].CurrentPosition = 0;  STEPPERS[0].StartPosition = 0;  STEPPERS[0].Counter = 0;  STEPPERS[1].PUL = new OutputPin(STEPPER2\_PUL);  STEPPERS[1].DIR = new OutputPin(STEPPER2\_DIR);  STEPPERS[1].bInverted = true;  STEPPERS[1].TargetPosition = 0;  STEPPERS[1].CurrentPosition = 0;  STEPPERS[1].StartPosition = 0;  STEPPERS[1].Counter = 0;  MotionDuration = 0;    Timer1.initialize(Timer1DelayUs);  Timer1.attachInterrupt(RunSteppers);  }  int GetCounter(float x, float delta)  {  float omega = PI/MotionDuration;  float X = constrain(x, 10, delta-10);  float speed = omega\*sqrt(delta\*X - X\*X);  float del = 1/speed;    return round(del/Timer1DelayUs);    }  void SetMotion(int Target[], long DurationMicro)  {  Timer1.stop();    MotionDuration = DurationMicro;  for (int i = 0; i < numSteppers; i++)  {  STEPPERS[i].TargetPosition = Target[i];  STEPPERS[i].StartPosition = STEPPERS[i].CurrentPosition;  STEPPERS[i].Counter = 0;    if (STEPPERS[i].TargetPosition > STEPPERS[i].CurrentPosition)  {  STEPPERS[i].Increment = 1;  STEPPERS[i].DIR -> write(STEPPERS[i].bInverted);  }  else  {  STEPPERS[i].Increment = -1;  STEPPERS[i].DIR -> write(!STEPPERS[i].bInverted);  }  }  EN = false; // enable low = activate drivers  Timer1.restart();  } | 75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110 | void RunSteppers()  {  int ready = 0;  for (int i = 0; i < numSteppers; i++)  {  if (STEPPERS[i].CurrentPosition == STEPPERS[i].TargetPosition)  {  ready++;  continue;  }  if (STEPPERS[i].Counter != 0)  {  STEPPERS[i].Counter--;  continue;  }  float delta = abs(STEPPERS[i].StartPosition - STEPPERS[i].TargetPosition);  float x = abs(STEPPERS[i].StartPosition - STEPPERS[i].CurrentPosition);  STEPPERS[i].Counter = GetCounter(x, delta);    STEPPERS[i].PUL -> pulse();  STEPPERS[i].CurrentPosition += STEPPERS[i].Increment;  }  if (ready >= numSteppers && !EN)  {  EN = true;  Timer1.stop();  }  } |

MyServos.cpp

Comando dos servomotores seguindo perfil de velocidades harmônico. O ângulo instantâneo é enviado para o driver, via I2C, a cada execução da rotina principal do programa.

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | const int channelList[] = {6, 7, 8, 9, 10, 11, 12, 13, 14, 15};  float OffsetDeflection[] = {1, -9, -1, -2, 6, -8, 0, 7, 2, -2};  int restPosition[] = {900, 900, 900, 900, 900, 900, 900, 900, 900, 900};  #define CONSTRAIN\_ANGLE\_MIN 65  #define CONSTRAIN\_ANGLE\_MAX 115  Adafruit\_PWMServoDriver pwm = Adafruit\_PWMServoDriver();  const int NumServos = 10;  MyServo SERVOS[NumServos];  long MotionDurationMs;  long TimeStart = 0;  long LastIteration = 0;  volatile bool bHasArrived = true;  void SetupServos() {  pwm.begin();  pwm.setPWMFreq(50);  for (int i = 0; i < NumServos;i++) {  SERVOS[i].CurrentAngle = 90;  SERVOS[i].Channel = channelList[i];  }  SetThetas(restPosition, 500);  }  void SetThetas(int Target[], long Duration) {  MotionDurationMs = Duration;  bool bChange;    for (int i = 0; i < NumServos;i++) {  SERVOS[i].StartAngle = SERVOS[i].CurrentAngle;  SERVOS[i].TargetAngle = constrain((float)(Target[i]) / 10.0, CONSTRAIN\_ANGLE\_MIN, CONSTRAIN\_ANGLE\_MAX);  if (i != 0 && i != 9)  SERVOS[i].TargetAngle = constrain(SERVOS[i].TargetAngle, SERVOS[i+1].TargetAngle, SERVOS[i-1].TargetAngle);  }  TimeStart = millis();  bHasArrived = false;  } | 48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80 | void StopServos() {  bHasArrived = true;  }  void RunServos() {  if (bHasArrived)  return;    long Now = millis();  if (Now > TimeStart + MotionDurationMs) {  bHasArrived = true;  return;  }  if (Now < LastIteration + ServoPeriodMs)  return;  for (int i = 0; i < NumServos; i++) {  float theta = 0.5\*(SERVOS[i].StartAngle + SERVOS[i].TargetAngle) + 0.5\*(SERVOS[i].TargetAngle-SERVOS[i].StartAngle)\*sin((float)(Now - TimeStart)\*PI/(float)MotionDurationMs - PI/2);  SERVOS[i].CurrentAngle = constrain(theta, CONSTRAIN\_ANGLE\_MIN, CONSTRAIN\_ANGLE\_MAX);  pwm.setPWM(SERVOS[i].Channel, 0, map(SERVOS[i].CurrentAngle + OffsetDeflection[i], SERVO\_MIN\_DEG, SERVO\_MAX\_DEG, SERVO\_MIN\_PWM, SERVO\_MAX\_PWM));  }    LastIteration = Now;  } |

**Anexo D2**. Trechos do código executado no Raspberry Pi.

Comunicação: Formatação de pacotes e envio de comandos para o Arduino.

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | public class SerialFrame : IDisposable  {  private MemoryStream memory;  private BinaryWriter bw;  public SerialFrame(byte cmdCode) {  memory = new MemoryStream();  bw = new BinaryWriter(memory);  memory.WriteByte(cmdCode);  }  public void AddArduinoInt(short number) {  bw.Write((Int16)number);  }  public byte[] GetData() {  return memory.ToArray();  }  }  private void SendFrame(SerialFrame frame) {  SendFrame(frame.GetData());  }  private void SendFrame(byte[] frame) {  const byte SLIP\_END = 0xC0;  const byte SLIP\_ESC = 0xDB;  const byte SLIP\_ESC\_END = 0xDC;  const byte SLIP\_ESC\_ESC = 0xDE; | 31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | MemoryStream frame\_escaped = new MemoryStream();  frame\_escaped.WriteByte(SLIP\_END);  for (int i = 0; i < frame.Length; i++) {  switch (frame[i]) {  case SLIP\_END:  frame\_escaped.WriteByte(SLIP\_ESC);  frame\_escaped.WriteByte(SLIP\_ESC\_END);  Debug.WriteLine("ESCAPE\_END");  break;  case SLIP\_ESC:  frame\_escaped.WriteByte(SLIP\_ESC);  frame\_escaped.WriteByte(SLIP\_ESC\_ESC);  Debug.WriteLine("ESCAPE\_ESC");  break;  default:  frame\_escaped.WriteByte(frame[i]);  break;  }  }  frame\_escaped.WriteByte(SLIP\_END);  Port.Write(frame\_escaped.ToArray(), 0, (int)frame\_escaped.Length);  frame\_escaped.Close();  } |

Música: Leitura dos eventos em um arquivo MIDI, utilizando funções da biblioteca NAudio.NET, e sequenciamento das notas correspondentes.

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | public class SheetMusic  {  public string SequenceName = "";  public InstrumentTrack[] Instruments;  public float DurationMs { get; set; }  public SheetMusic() {  DurationMs = 0;  }  public void LoadFromMidi(string filename) {  MidiFile mfile = new MidiFile(filename);  List<KeyValuePair<int, int>> channel\_x\_patch = new List<KeyValuePair<int, int>>();  List<MusicNote> notes = new List<MusicNote>();  float tickDuration = 60000f / (float)(120 \* mfile.DeltaTicksPerQuarterNote);  for (int track = 0; track < mfile.Tracks; track++)  foreach (MidiEvent me in mfile.Events[track]) {  if (MidiEvent.IsNoteOn(me)) {  NoteOnEvent noe = me as NoteOnEvent;  MusicNote mn = new MusicNote() {  StartTime = noe.AbsoluteTime \* tickDuration,  EndTime = (noe.AbsoluteTime + noe.NoteLength) \* tickDuration,  Channel = noe.Channel,  NoteNumber = noe.NoteNumber,  Velocity = noe.Velocity,  };  DurationMs = Math.Max(DurationMs, mn.EndTime);  notes.Add(mn);  }  if (me is MetaEvent) {  switch ((me as MetaEvent).MetaEventType) {  case MetaEventType.ProgramName:  case MetaEventType.SequenceTrackName:  SequenceName = (me as TextEvent).Text;  break; | 43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76 | case MetaEventType.SetTempo:  TempoEvent te = me as TempoEvent;  tickDuration = 60000f / (float)(te.Tempo \* mfile.DeltaTicksPerQuarterNote);  break;  case MetaEventType.TimeSignature:  TimeSignatureEvent ts = me as TimeSignatureEvent;  // not suported yet  break;    }  }  if (me is PatchChangeEvent) {  PatchChangeEvent pe = (me as PatchChangeEvent);  channel\_x\_patch.Add(new KeyValuePair<int, int>(pe.Channel, pe.Patch));  }  }  Instruments = new InstrumentTrack[channel\_x\_patch.Count];  for (int i = 0; i < Instruments.Length; i++) {  Instruments[i] = new InstrumentTrack() {  Patch = channel\_x\_patch[i].Value,  Channel = channel\_x\_patch[i].Key,  PatchName= PatchChangeEvent.GetPatchName(channel\_x\_patch[i].Value,  Notes = notes.FindAll(x => x.Channel == channel\_x\_patch[i].Key).ToArray()  };  }  }  } |

Interpolação: Trecho de código que calcula um ponto do perfil de posição para um dado tempo.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  2829  30  31  32  33  34  35 | public struct InterpolationPoint {  public double t;  public double x;  }  public static double Sigmoid(double t, InterpolationPoint p1, InterpolationPoint p2, out double velocity) {  double K1 = (p2.x + p1.x) / 2f;  double K2 = (p1.x - p2.x) / 2f;  double W = Math.PI / (p2.t - p1.t);  double F = 0.5f \* Math.PI \* (3 \* p2.t - p1.t) / (p1.t - p2.t);  velocity = W\*K2 \*Math.Cos(W \* t + F);  return K1 + K2\*Math.Sin( W \* t + F);  }  public void InterpolateToState(int DurationMs, int[] AbsSteps, float[] FingerAnglesDeg) {  var frame1 = new SerialFrame((byte)'M');  frame1.AddArduinoInt((short)DurationMs);  frame1.AddArduinoInt((short)AbsSteps[0]);  frame1.AddArduinoInt((short)AbsSteps[1]);  SendFrame(frame1);  var frame2 = new SerialFrame((byte)'R');  frame2.AddArduinoInt((short)DurationMs);  for (int i = 0; i < 10; i++) {  short angle = (short)(FingerAnglesDeg[i] \* 10.0);  frame2.AddArduinoInt(angle);  }    SendFrame(frame2);  } |

Sistema de coordenadas: Conversão de coordenadas em unidades de teclas para unidades de passos do motor ou de ângulo dos dedos.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public static float DeflectionToThetaDeg(float x) {  return (float)(90.0 - Math.Asin(x\*KeyWidthMM/FingerRadiusMM)\*180.0/Math.PI);  }  public static int CoordinateToStep(float x) {  return (int)Math.Round((float)StepsPerRevolution \* KeyWidthMM \* x / (PulleyZ \* BeltPitch));  } |

**ANEXO E**

Lista de materiais e preços orçados

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Item** | | | | **Quant** | **Preço un.** | **Preço** | | Raspberry Pi 3 Model B+ | | | | 1 | 188,49 | 188,49 | | Mangueira Pneumática De Poliuretano Tubo 6 mm [metro] | | | | 40 | 2,00 | 80,00 | | Silenciador Sinterizado Com Regulagem De Fluxo Rosca 1/4 | | | | 5 | 6,70 | 33,50 | | Conexões Pneumáticas Reta Tubo Ø06 para Rosca M5 macho com O'ring | | | | 20 | 2,15 | 43,00 | | Bloco Manifold Com trilho e 10 Válvulas Solenóides 5/2 Vias 1/4" 12 V | | | | 1 | 599,90 | 599,90 | | Conexões Pneumáticas Reta Tubo Ø06 Rosca 1/4 Bsp O'ring | | | | 20 | 2,57 | 51,40 | | Fixador Emenda para Correia Dentada Gt2 em Liga de Alumínio | | | | 2 | 16,90 | 33,80 | | Botão Emergência 22 mm 1nf Cogumelo | | | | 1 | 17,90 | 17,90 | | Suporte Sk16 p/ Eixo Linear 16 mm em Alumínio | | | | 4 | 17,90 | 71,60 | | Modulo Serial I2C P/ LCD 16x2 | | | | 1 | 12,90 | 12,90 | | Cabo Extensão USB 30 Cm Para Painel | | | | 3 | 21,90 | 65,70 | | Display LCD 16x2 Azul | | | | 1 | 16,90 | 16,90 | | Caixa Protetora Em Acrílico Para Arduino Mega 2560 | | | | 1 | 17,90 | 17,90 | | Tomada Painel Interruptor Iluminado 10A+cabo De Força | | | | 1 | 27,90 | 27,90 | | Polia Sincronizada GT2 40 Dentes Alumínio Eixo 8 mm | | | | 4 | 27,90 | 111,60 | | Correia dentada GT2 Passo 2mm, 6mm De Largura [metro] | | | | 6 | 7,25 | 43,50 | | Pillow Block Fechado Scs16uu 16mm | | | | 8 | 27,70 | 221,60 | | Mancal P/ Eixo 8 mm Kfl08 C/ Rolamento | | | | 4 | 7,50 | 30,00 | | Fonte Chaveada  12 V 20 A 250 W | | | | 1 | 49,90 | 49,90 | | Extensão Cabo Hdmi 30cm P/ Embutir Em Painel | | | | 1 | 23,90 | 23,90 | | Case Acrilico Para Raspberry Pi3 com Cooler e dissipadores | | | | 1 | 31,90 | 31,90 | | Fonte Chaveada  5 V 20 A 100 W | | | | 1 | 49,70 | 49,70 | | Fonte Chaveada Andeli Bivolt 24 V 12,5 A 300 W | | | | 1 | 109,90 | 109,90 | | Suporte P/ Motor De Passo Nema 23 - 90º em aço 4,76 mm | | | | 2 | 22,90 | 45,80 | | Servo Driver I2C 16 Canais Pca9685 | | | | 1 | 19,90 | 19,90 | | Compressor De Ar 2 HP 25 L Pratic Air Schulz | | | | 1 | 497,00 | 497,00 | | Estante Suporte Teclado Z Mellody Q1b Ajustavel | | | | 1 | 149,00 | 149,00 | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Servomotor TGY-901D 13 kgf.cm | | | | 10 | 42,16 | 421,60 | | Motor de passo NEMA 23 Wotiom  3 A 15 kgf.cm | | | | 2 | 152,82 | 305,64 | | Driver p/ motor de passo Wotiom WD-2404 - 4 A | | | | 2 | 79,90 | 159,80 | | Arduino Mega 2560 | | | | 1 | 79,88 | 79,88 | | Braço metálico Futaba 25T M3 | | | | 10 | 3,96 | 39,60 | | Filtro de ar com regulador de pressão AFP-2000 rosca 1/4 BSP | | | | 1 | 37,53 | 37,53 | | Esteira porta cabos 18x50 mm [metro] | | | | 3 | 49,90 | 149,70 | | Cilindro pneumático CDJ2B10-30-B - curso 30 mm furo 10 mm | | | | 10 | 14,08 | 140,80 | | Transdutor de pressão | | | | 1 | 30,80 | 30,80 | | Suporte p/ dedos em aço inoxidável 304 - 8 mm | | | | 10 | 16,90 | 169,00 | | Placa de apoio inferior em aço inoxidável 304 - 0,8 mm | | | | 2 | 24,26 | 48,52 | | Placa de apoio superior em aço inoxidável 304 - 0,8 mm | | | | 2 | 15,56 | 31,12 | | Placa de apoio inferior em acrílico - 4,5 mm | | | | 2 | 0,00 | 0,00 | | Placa de apoio superior em acrílico - 4,5 mm | | | | 2 | 0,00 | 0,00 | | Dedos em acrílico | | | | 10 | 0,00 | 0,00 | | Eixo guia linear 16 mm H6 retificado polido em aço 1055 [cm] | | | | 2,63 | 62,79 | 165,14 | | Caixa de montagem Steck 310x230x125 mm | | | | 1 | 84,90 | 84,90 | | Barra roscada M5 [metro] | | | | 2 | 5,90 | 11,80 | | Conector DB9 p/ placa de circuito | | | | 1 | 5,00 | 5,00 | | Conector DB15 p/ placa de circuito | | | | 2 | 5,00 | 10,00 | | Conector DB15 p/ cabo | | | | 6 | 5,00 | 30,00 | | Conector DB9 p/ cabo | | | | 3 | 5,00 | 15,00 | | Cabo manga 16 vias AWG24 [metro] | | | | 12 | 0,00 | 0,00 | | Placa de circuito de acionamento de potência personalizada | | | | 1 | 90,00 | 90,00 | | Perfil estrutural em alumínio  30x30 mm [metro] | | | | 2,04 | 35,50 | 72,42 | | Perfil estrutural em alumínio  30x60 mm [metro] | | | | 2,06 | 58,50 | 120,51 | | Capa fechamento frontal 30x30 mm | | | | 6 | 1,50 | 9,00 | | Capa fechamento frontal 30x60 mm | | | | 2 | 3,50 | 7,00 | | Porca martelo canal 8 mm rosca M4 | | | | 10 | 1,50 | 15,00 | | Tira de fechamento lateral [metro] | | | | 6 | 2,20 | 13,20 | | Bucha de redução M14 p/ M8 | | | | 18 | 4,00 | 72,00 | | Porca martelo canal 8 mm rosca M5 | | | | 12 | 1,70 | 20,40 |  |  | | --- | | TOTAL: **R$ 4.999,95** | |