Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

LaserHarp

Construção de um instrumento eletrônico do zero

Raphael dos Reis Gusmão

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisor: Prof. Roberto Hirata Junior

São Paulo
2019

Resumo

Raphael dos Reis Gusmão. **LaserHarp:** ***Construção de um instrumento eletrônico do zero***. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2019.

Harpa Laser é o nome dado a um instrumento eletrônico que consiste em vários feixes de laser que, ao serem interrompidos, tocam uma nota musical. A essência do funcionamento de uma harpa laser baseia-se no fenômeno da Persistência da Visão. Trata-se de uma espécie de interpolação natural realizada pelo cérebro para transformar uma sequência discreta de imagens em um fluxo contínuo, de modo a se ter percepção de movimento. O objetivo deste trabalho é construir um instrumento como este utilizando materiais de baixo custo. Bem como desenvolver um software capaz de controlá-lo via comunicação serial e de enviar sinais MIDI para outros programas.

Palavras-chave: Harpa Laser. Arduino. Hardware. Software. Modelagem 3D.

Abstract

Raphael dos Reis Gusmão. **LaserHarp:** ***Building an electronic instrument from scratch***. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2019.

Laser Harp is the name given to an electronic instrument that consists of several laser beams that, when interrupted, play a musical note. The essence of the operation of a laser harp is based on the phenomenon of Persistence of Vision. It is a kind of natural interpolation performed by the brain to transform a discrete sequence of images into a continuous flow, in order to have a perception of movement. The goal of this work is to build an instrument like this using low-cost materials. As well as developing a software capable of controlling it via serial communication and sending MIDI signals to other programs.

Keywords: Laser Harp. Arduino. Hardware. Software. 3D Modeling.

Lista de Figuras

Lista de Programas

Sumário

Capítulo 1

Introdução

Harpa Laser é o nome dado a um instrumento eletrônico que consiste em vários feixes de laser que, ao serem interrompidos, tocam uma nota musical. Os feixes de laser, ou cordas[1], são produzidos com a ajuda de um espelho fixo a um motor girando em alta velocidade. Um feixe é emitido de cada vez, mas, como isso acontece muito rápido, tem-se a impressão de que todos são formados ao mesmo tempo. É atribuído a esse fenômeno o nome de Persistência da Visão.

1.1 Mercado

Não se tem notícia de harpas lasers sendo vendidas no Brasil, apenas no exterior. A única harpa laser profissional à venda se chama Prolight Laser Harp[28] (Figura 1.1), fabricada por um empresa localizada na Croácia chamada Prolight. Seu custo pode chegar a R$4.000,00, sem contar com impostos de importação.

Figura 1.1: *Prolight Laser Harp*
Fonte: https://laser-harp.com/

A harpa produzida neste trabalho custou em torno de R$300,00 contando com todos os seus componentes eletrônicos e com o material usado para fabricar sua estrutura.

[1] Ao longo deste trabalho, a palavra “corda” será utilizada como sinônimo de “laser” no momento em que ele é refletido.

1.2 Motivação

A harpa laser se tornou popular pelo músico francês Jean Michel Jarre, que a usa constantemente em seus shows desde os anos 80. Na internet existem diversos vídeos[19] em que o músico toca sua harpa laser ao vivo, com inúmeros efeitos visuais pirotécnicos pelo palco. Tais vídeos motivaram muitas pessoas a criarem suas próprias harpas lasers e, consequentemente, inspiraram este trabalho.

1.3 Objetivo

Este trabalho tem como objetivo construir uma harpa laser de baixo custo utilizando conhecimentos da área da computação, bem como conhecimentos de eletrônica. As etapas de desenvolvimentos foram divididas em capítulos e seções da seguinte forma:

- Modelagem em 3D da estrutura do instrumento (Capítulo 2).
- Montagem do hardware da harpa (Capítulo 3).
- Desenvolvimento do código do hardware (Seção 3.3)
- Desenvolvimento do software para permitir o controle da harpa pelo computador (Capítulo 4).

Foi dado o nome de **LaserHarp** para a harpa laser desenvolvida.

Capítulo 2

Estrutura física

É de fundamental importância que os componentes como o motor, o espelho e os lasers estejam em uma posição específica e bem presos para funcionarem corretamente. Sem isso, fica difícil de se ter uma formação de cordas com qualidade. Para deixar todos os componentes bem fixos e protegidos, foi modelada uma estrutura física para a LaserHarp.

O fator visual também foi levado em consideração durante a modelagem. A estrutura, que possui um formato triangular, foi toda desenhada para ser o mais simétrica possível. Uma estética bonita chama mais a atenção de quem vê e qualifica o instrumento como uma arte em si mesmo.

2.1 Modelagem 3D

A seguir são mostradas todas as peças produzidas para montar a LaserHarp. Todas foram modeladas em 3D com ajuda do software Fusion 360[6] da Autodesk sob licença gratuita para estudantes. As imagens presentes nesta Seção foram renderizadas neste mesmo software.

Algumas peças, como a base, paredes e tampa foram salvas em um formato de imagem chamado DXF[9] (*Drawing Exchange Format*), que é o formato aceito pelas máquinas cortadoras a laser. Todas estão na pasta "**`laser_cut/`**". As demais peças foram salvas no formato STL (abreviação de "*stereolithography*"), o mais utilizado pelas impressoras 3D. Todos sob a pasta "**`3d_print/`**".

Todos os componentes eletrônicos citados a seguir estão detalhados no Capítulo 3.

Os modelos 3D dos componentes eletrônicos (motor, sensor de velocidade, conector P4, módulo micro USB e placa fenolite) foram baixados gratuitamente do site GrabCAD[29].

- **Base** (Figura 2.1)
  - Essa é a "peça mãe" da LaserHarp. Todas as demais peças são encaixadas e parafusadas nela. Para isso, a base possui 20 buracos para parafusos. Na LaserHarp ela foi feita com madeira MDF (ver Seção 2.2) cortada a laser devido a resistência desse material.
  - Arquivo: "**`laser_cut/base.dxf`**".

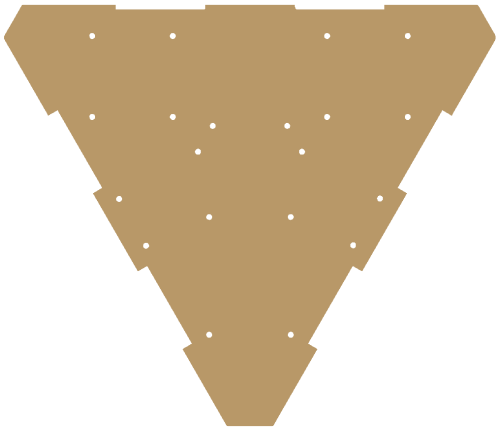


Figura 2.1: *Desenho da base da estrutura da LaserHarp.*

- **Paredes** (Figura 2.2)
  - A estrutura possui três lados, cada um com uma parede, todas encaixáveis na base (Figura 2.3). As paredes laterais são idênticas e possuem um buraco no meio para posicionar os suportes do conector P4 e do módulo micro USB, um em cada parede. Essas paredes também foram produzidas com MDF, com exceção da parede frontal, que foi feita com acrílico (ver Seção 2.2). O acrílico escolhido, preto fumê, permite que o interior da estrutura possa ser visto mesmo com ela fechada.
  - Arquivos:
    - "**`laser_cut/wall_front.dxf`**".
    - "**`laser_cut/wall_left.dxf`**".
    - "**`laser_cut/wall_right.dxf`**".

(a) *Parede lateral.* **(b)** *Parede frontal.*

Figura 2.2: *Desenhos das paredes da estrutura da LaserHarp.*

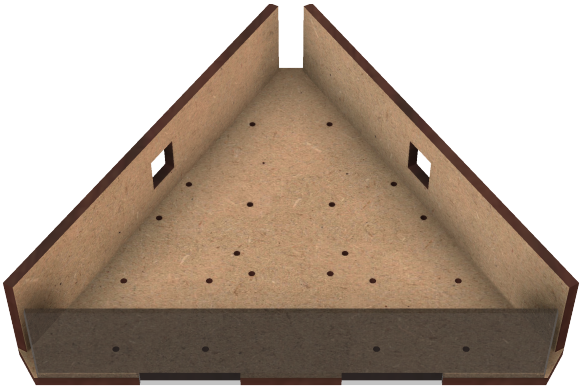
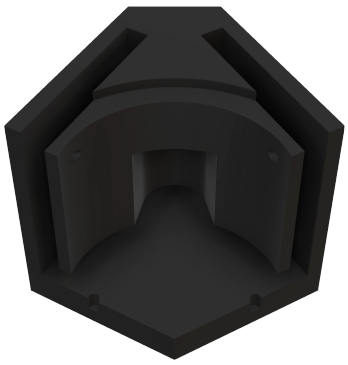
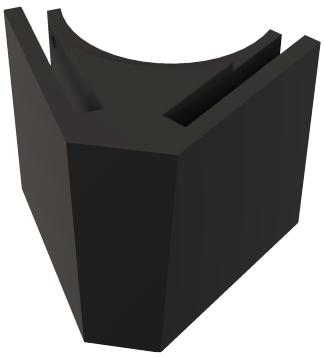


Figura 2.3: *Paredes encaixadas na base da estrutura da LaserHarp.*

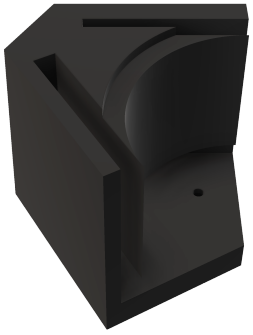
- **Pés da estrutura** (Figura 2.4)
  - Para sustentar toda a estrutura sem que os parafusos toquem o chão, foram modelados estes pés, que são encaixáveis na base (Figura 2.5) e também servem para fixar as paredes na base depois que estas foram encaixadas. O pé traseiro possui uma abertura no centro para acomodar o cabo USB que é ligado ao Arduino e dois buracos na parte superior para encaixar o eixo da tampa.
  - Arquivos:
    * "`3d_print/foot_back.stl`".
    * "`3d_print/foot_right.stl`".
    * "`3d_print/foot_left.stl`".



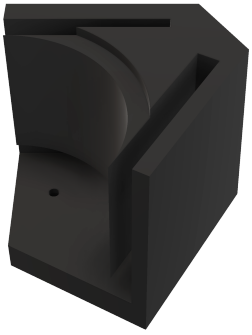
(a) *Parte da frente do pé traseiro.*



(b) *Parte de trás do pé traseiro.*



(c) *Parte de trás do pé direito.*

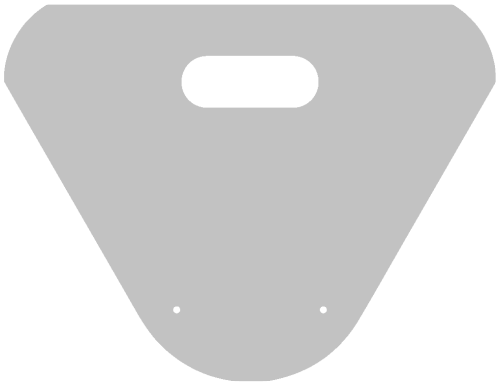


(d) *Parte de trás do pé esquerdo.*

Figura 2.4: *Pés da estrutura da LaserHarp.*

Figura 2.5: *Os 3 pés encaixados na base da estrutura com o eixo da tampa encaixado no pé traseiro.*

- **Tampa** (Figura 2.6)
  - Para proteger todos os componentes (e evitar que o espelho seja arremessado longe caso ocorra algum imprevisto) foi modelada uma tampa para a estrutura da LaserHarp. A tampa (Figura 2.6), assim como a parede frontal, também foi produzida com acrílico semitransparente, para que o interior seja visto sem a necessidade de se abri-la. Repare que há uma abertura no meio da tampa, para que todas as cordas da harpa possam passar mesmo quando ela estiver fechada. Foi modelada também uma peça (Figura 2.6b) capaz de rotacionar sobre um eixo. Esta peça possui dois buracos para que possa ser parafusada na tampa e, assim, fixá-la na estrutura da LaserHarp. A Figura 2.7 mostra o resultado de se juntar a tampa, a peça e o eixo.
  - Arquivos:
    - “`laser_cut/lid.dxf`”.
    - “`3d_print/lid_hinge.stl`”.



(a) *Desenho da tampa da estrutura da LaserHarp.*

(b) *Peça para juntar a tampa e o eixo.*

Figura 2.6: *Tampa da estrutura da LaserHarp.*

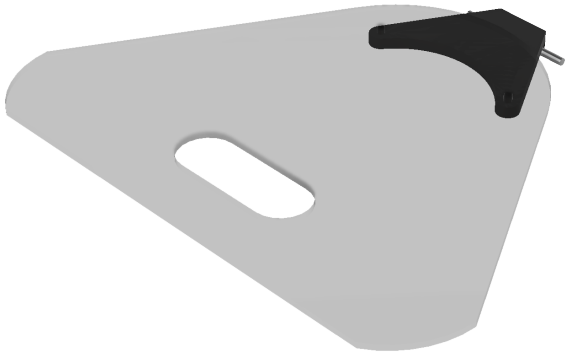
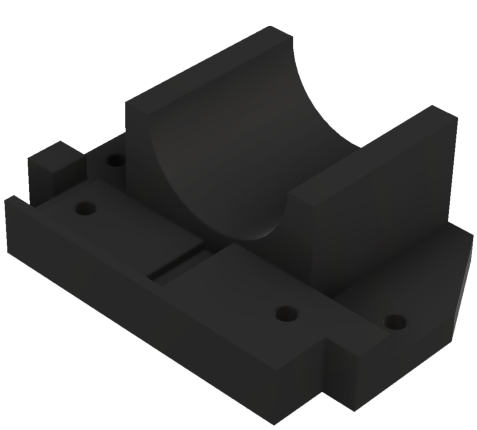


Figura 2.7: *Paredes encaixadas na base da estrutura da LaserHarp.*

- Após juntar todas as partes da estrutura o resultado é o visto na Figura 2.8.

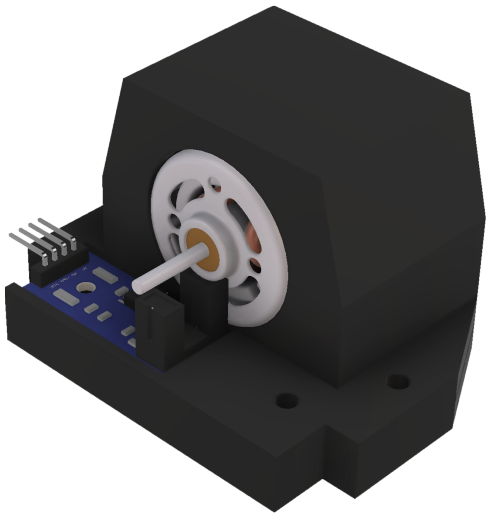
Figura 2.8: *Estrutura completa da LaserHarp.*

- **Suporte para o motor e para o sensor de velocidade** (Figura 2.9)
  - É de vital importância que o motor fique muito bem fixo para evitar oscilações na formação dos lasers. Para isso foi modelado este suporte, que consiste na peça inferior (Figura 2.9a) e na peça superior (Figura 2.9b), ambas encaixáveis na base da estrutura harpa com parafusos. Além disso, a parte inferior do suporte possui um espaço para encaixar o sensor de velocidade na exata posição em que estará o disco *encoder* do suporte do espelho. O resultado pode ser visto na Figura 2.9c, em que estão presentes versões em 3D do motor e do sensor.
  - Arquivos:
    * "`3d_print/motor_support_bottom.stl`".
    * "`3d_print/motor_support_top.stl`".



(a) *Peça inferior do suporte.*

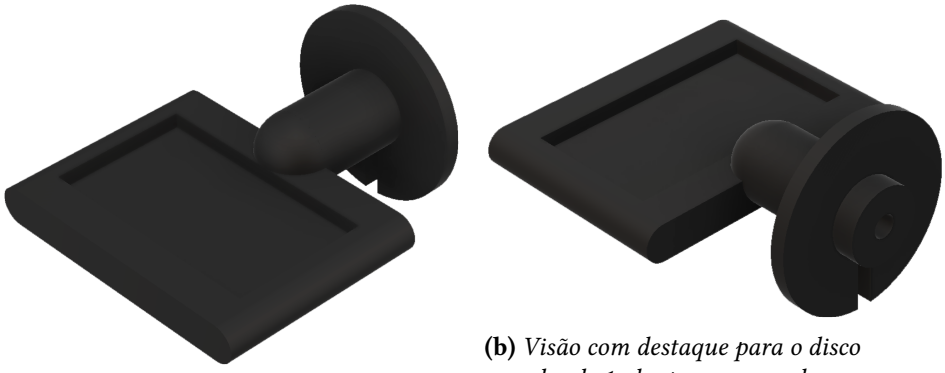
(b) *Peça superior do suporte.*



(c) *Resultado das peças encaixadas junto com os componentes.*

Figura 2.9: *Suporte para o motor e para o sensor de velocidade.*

- **Suporte para o espelho** (Figura 2.10)
  - É a peça responsável por fixar o espelho no eixo do motor. Em sua base há um disco *encoder* de 1 abertura, usado para se calcular a velocidade do motor (Seção 3.4).
  - Arquivos:
    * "`3d_print/mirror_support_bottom.stl`".
    * "`3d_print/mirror_support_middle.stl`".
    * "`3d_print/mirror_support_top.stl`".

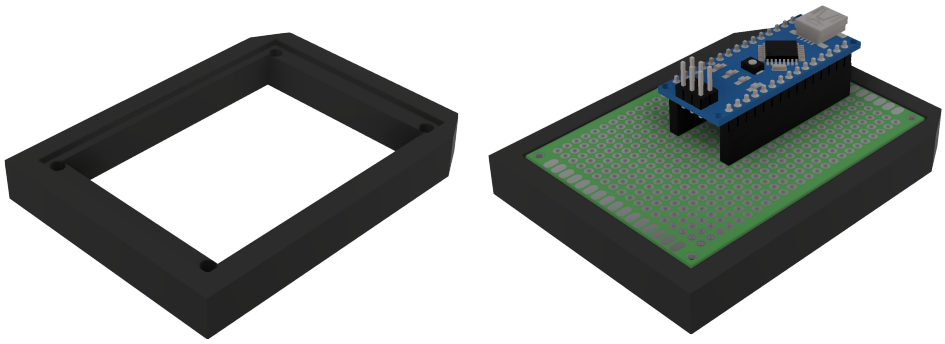


(a) *Visão frontal do suporte.*

(b) *Visão com destaque para o disco encoder de 1 abertura em sua base.*

Figura 2.10: *Suporte para o espelho.*

- **Suporte para a placa fenolite** (Figura 2.11)
  - Suporte usado para fixar a placa fenolite na base da estrutura da LaserHarp.
  - Arquivo: "`3d_print/board_support.stl`".

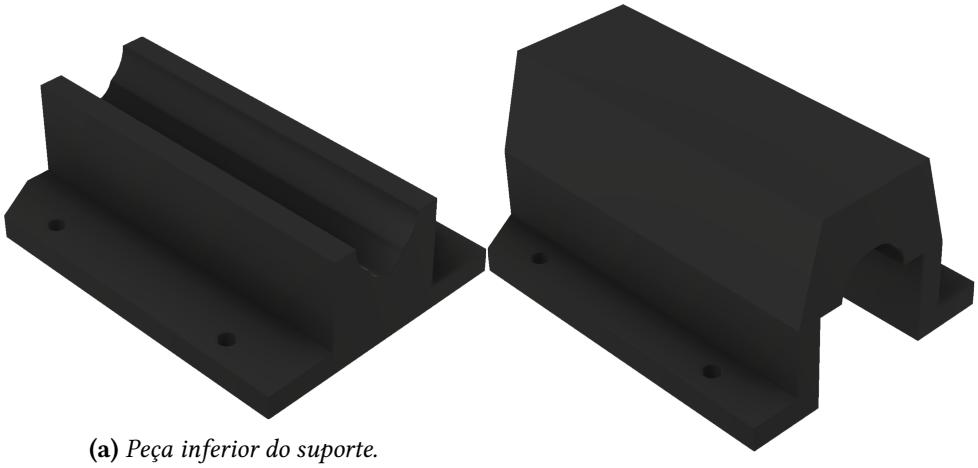


(a) *Suporte sem a placa.*

(b) *Suporte com a placa encaixada junto com um Arduino Nano.*

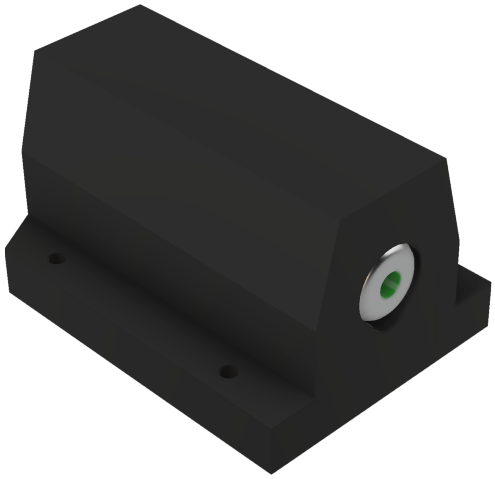
Figura 2.11: *Suporte para a placa fenolite.*

- **Suportes para as canetas laser** (Figura 2.12)
  - Para segurar cada caneta laser na posição exata em que ela deve ficar, apontada para o centro do espelho, foi modelado este suporte. Ambas as canetas utilizam este suporte, então devem ser produzidos dois. O suporte consiste em duas partes: A inferior, que segura a caneta na posição correta, e a superior, que a prende. Há buracos no suporte para que ele possa ser parafusado na base da estrutura da LaserHarp.
  - Arquivos:
    - "`3d_print/laser_support_bottom.stl`".
    - "`3d_print/laser_support_top.stl`".



(a) *Peça inferior do suporte.*

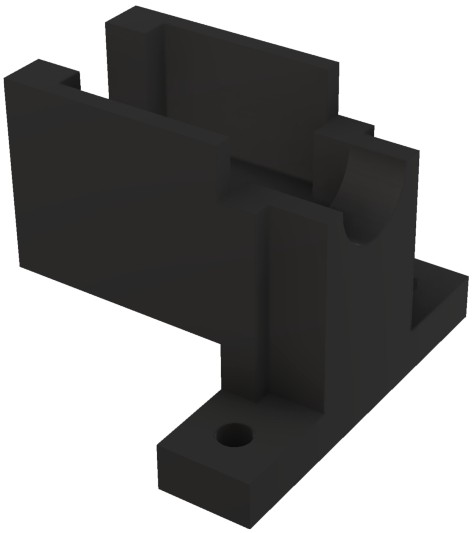
(b) *Peça superior do suporte.*



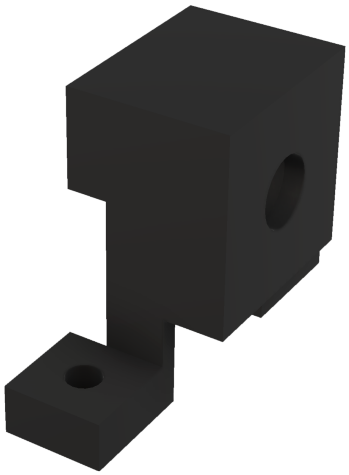
(c) *Resultado das peças encaixadas junto com a caneta laser verde.*

Figura 2.12: *Suporte para caneta laser.*

- **Suporte para o conector P4** (Figura 2.13)
  - Este suporte foi modelado para fixar o conector P4 na parede lateral direita da estrutura da LaserHarp.
  - Arquivos:
    - “`3d_print/power_jack_support_bottom.stl`”.
    - “`3d_print/power_jack_support_top.stl`”.



(a) *Peça inferior do suporte.*

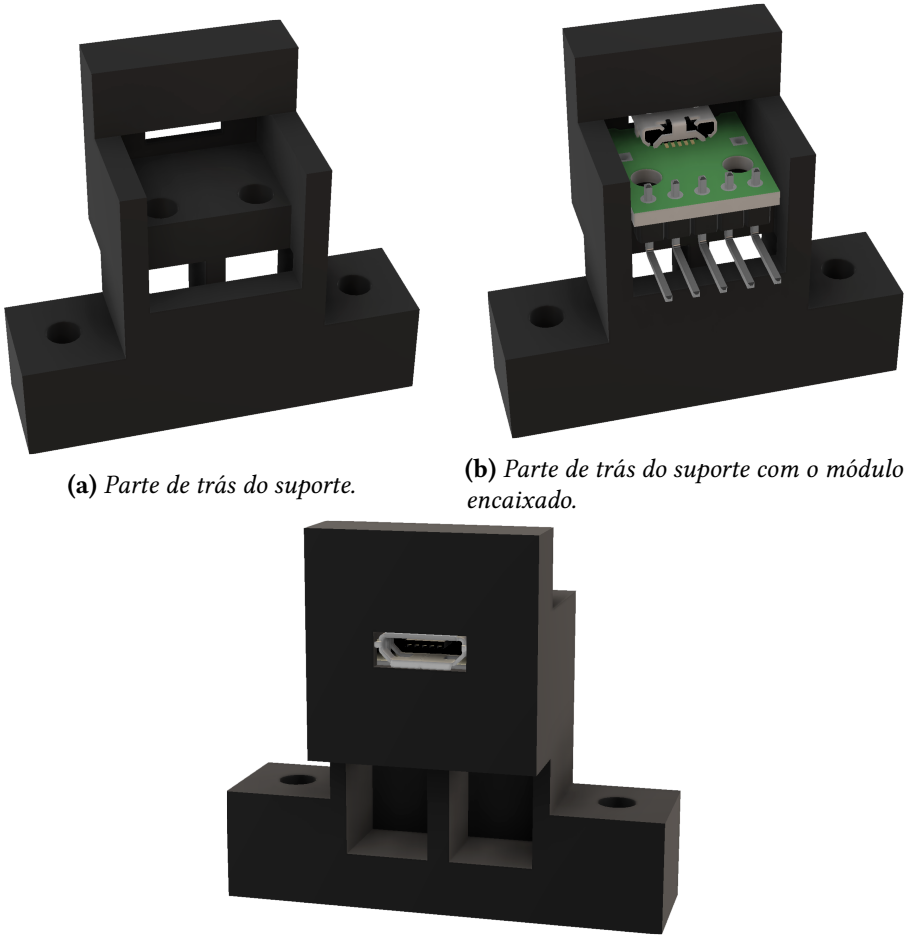


(b) *Peça superior do suporte.*

(c) *Resultado das peças encaixadas junto com conector P4.*

Figura 2.13: *Suporte para o conector P4.*

- **Suporte para o módulo micro USB** (Figura 2.14)
  - Este suporte foi modelado para fixar o módulo micro USB na parede lateral esquerda da estrutura da LaserHarp.
  - Arquivo: "**`3d_print/usb_support.stl`**".



(a) *Parte de trás do suporte.*

(b) *Parte de trás do suporte com o módulo encaixado.*

(c) *Parte da frente do suporte com o módulo encaixado.*

Figura 2.14: *Suporte para o módulo micro USB.*

- A LaserHarp completa com todos os componentes pode ser vista nas Figuras 2.15, 2.16 e 2.17.

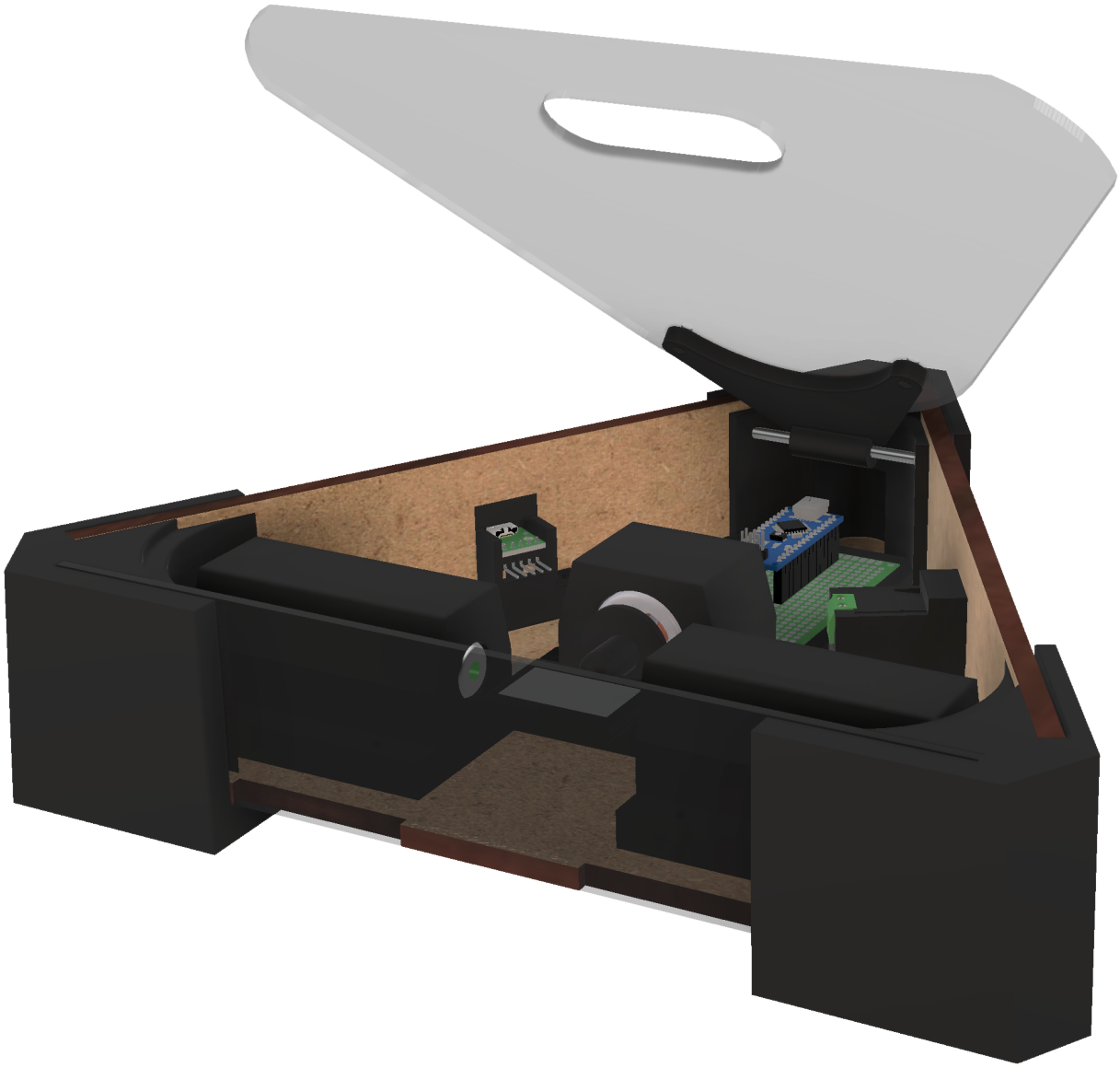


Figura 2.15: *LaserHarp vista de lado com a tampa semiaberta.*

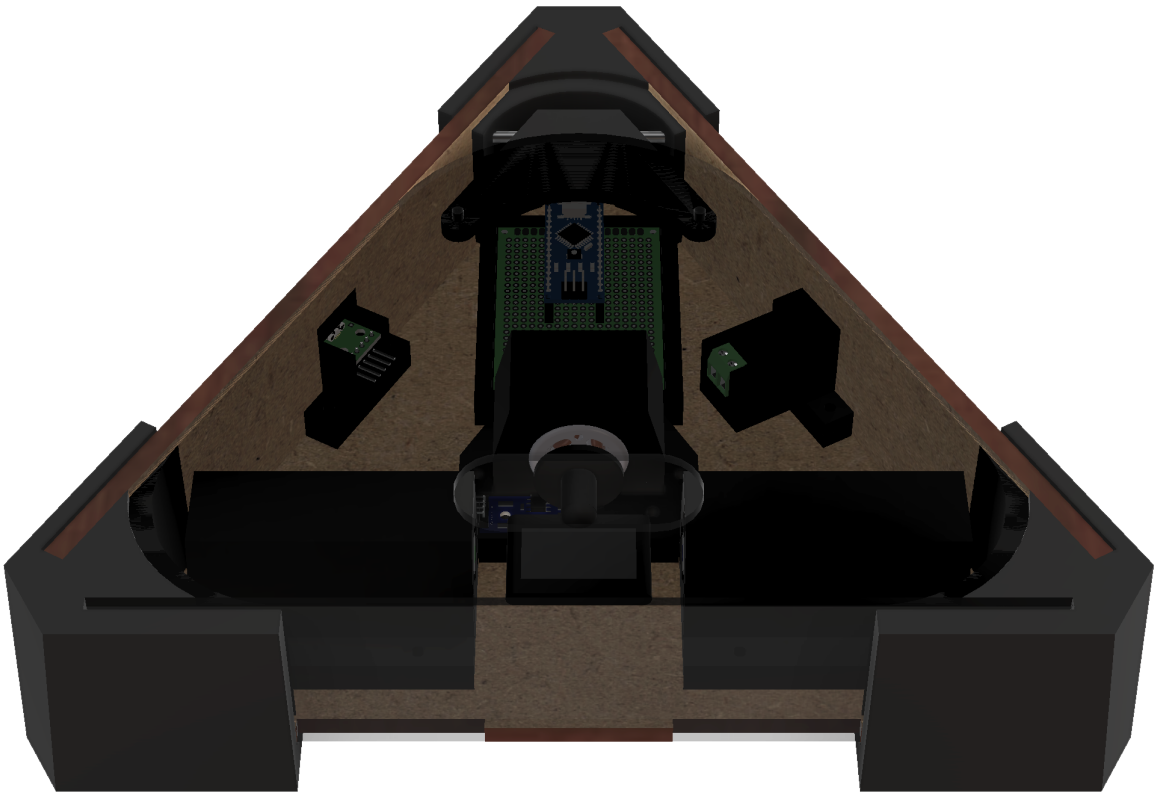


Figura 2.16: *LaserHarp vista de frente com a tampa fechada.*

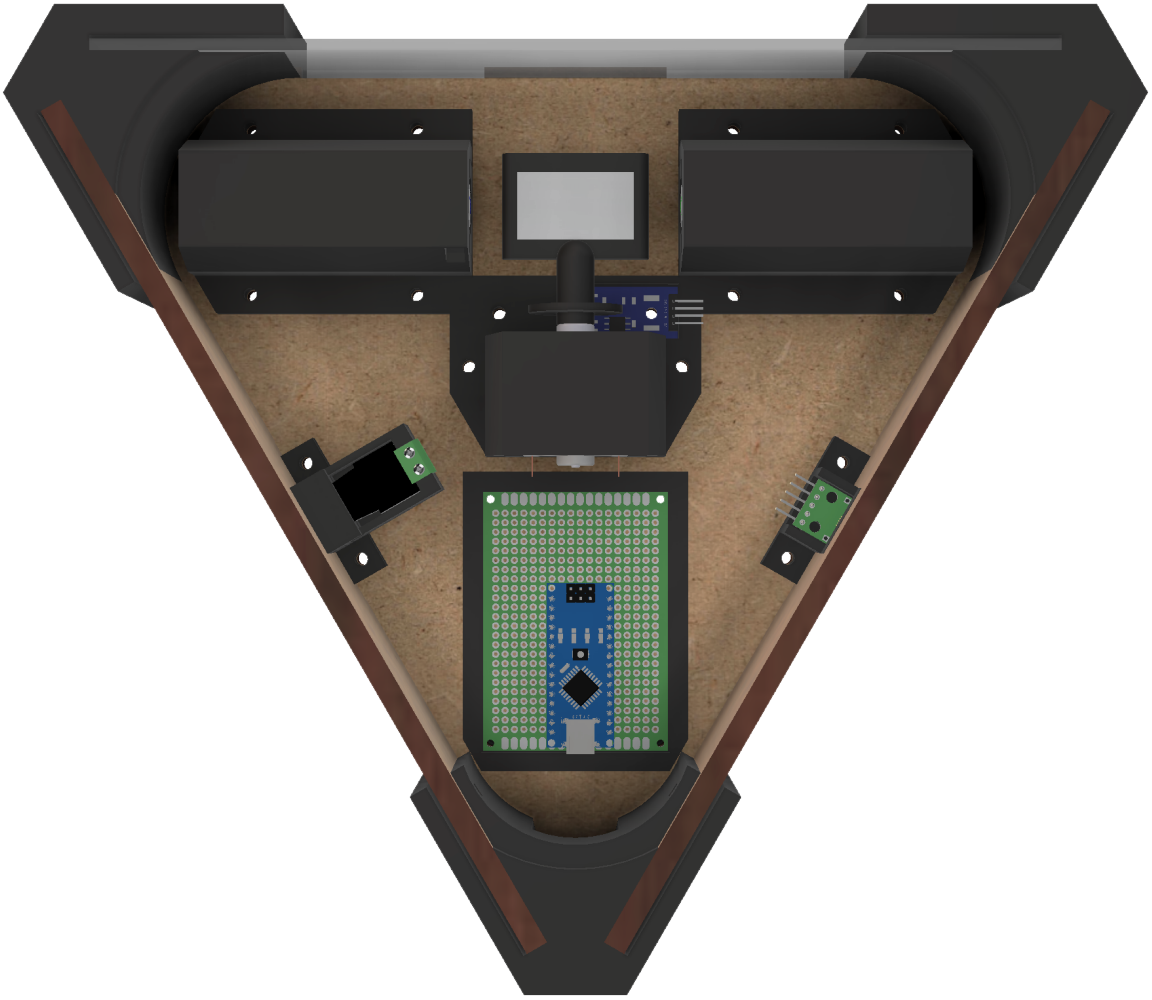


Figura 2.17: *LaserHarp vista de cima sem a tampa.*

2.2 Materiais

Para se produzir e montar a estrutura da LaserHarp e todas as suas peças que foram modeladas, são necessários os seguintes materiais:

- **Chapa de MDF**
  Para se fazer estruturas rígidas de baixo custo, o material recomendado é o MDF (*Medium Density Fiberboard*), que é derivado da madeira. Seu custo varia bastante, dependendo da qualidade do material, podendo chegar a R$100,00 por metro quadrado. Para a estrutura da LaserHarp se usa bem menos que isso, então o custo sai mais baixo.

- **Chapa de acrílico**
  É tão resistente quanto o MDF, além disso possui uma aparência mais atraente. Na LaserHarp, a parede frontal e a tampa da estrutura foram feitas com acrílico preto fumê, ou seja, semitransparente. Seu custo pode chegar a R$150,00 por metro quadrado. Novamente, a estrutura possui uma medida bem menor que $1m^2$, então o valor a ser pago é menor.

- **Parafusos, porcas e arruela**
  Para fixar todos os componentes e peças na estrutura da harpa foram usados:

  - 16 parafusos M3 de 25mm de comprimento.
  - 4 parafusos M3 de 16mm de comprimento.
  - 4 parafusos M2 de 20mm de comprimento.
  - 20 porcas M3.
  - 4 porcas M2.
  - 1 arruela M2 de nylon[1]

  Tudo custou cerca de R$10,00.

- **Eixo de aço**
  Para fazer o eixo da tampa foi necessário utilizar um pedaço cortado de um eixo de aço. Seu custo foi de aproximadamente R$5,00.

- **Espelho**
  O espelho da LaserHarp foi reaproveitado de uma impressora a laser descartada. Custo zero. Suas dimensões são: 295mm x 170mm x 15mm.

- **Filamento ABS preto**
  As peças 3D da LaserHarp foram impressas com filamento ABS preto. O ABS (Acrilonitrila Butadieno Estireno) é um plástico bastante usado para impressões em 3D. Este material foi escolhido devido à sua alta durabilidade e resistência. Seu preço depende de sua qualidade, podendo chegar a R$80,00 por quilo. As peças da LaserHarp pesam poucos gramas, então o valor de produção é bem mais baixo que isso.

[1] Foi utilizada na fixação do sensor de velocidade. A arruela de nylon é recomendada para evitar o contato da cabeça do parafuso com o componente eletrônico.

Capítulo 3

Hardware

A essência do funcionamento de uma harpa laser baseia-se no fenômeno da Persistência da Visão. Trata-se de uma espécie de interpolação natural realizada pelo cérebro para transformar uma sequência discreta de imagens em um fluxo contínuo, de modo a se ter percepção de movimento.

A LaserHarp se aproveita deste fenômeno para que seus lasers pareçam ativos a todo momento. O que de fato acontece é que a harpa aciona seus lasers (1 verde e 1 azul) várias vezes durante períodos curtíssimos de tempo direcionando-os para um motor girando em alta velocidade. O motor possui um espelho preso ao seu eixo e, por isso, acaba refletindo todos os acionamentos dos lasers apontados para ele. Esses reflexos formam as cordas da harpa. O cérebro humano identifica todas as cordas como se estivessem presentes ao mesmo tempo, quando, na verdade, aparecem apenas uma de cada vez. Para criar este efeito, a LaserHarp gira seu motor a 55 rotações por segundo.

Apesar de parecer complexo no primeiro momento, o fenômeno é relativamente simples de ser reproduzido. O presente capítulo demonstra, de forma detalhada, como desenvolver a LaserHarp, inteiramente construida com materiais de baixo custo e fáceis de serem encontrados no Brasil.

3.1 Componentes eletrônicos

No desenvolvimento do hardware da LaserHarp, foram usados os seguintes componentes:

- 1x **`Arduino Nano`**[2] (Figura 3.1)
  - Pequena placa com um microcontrolador programável. É o "cérebro" da LaserHarp.
  - Preço estimado: **R$ 30,00**.

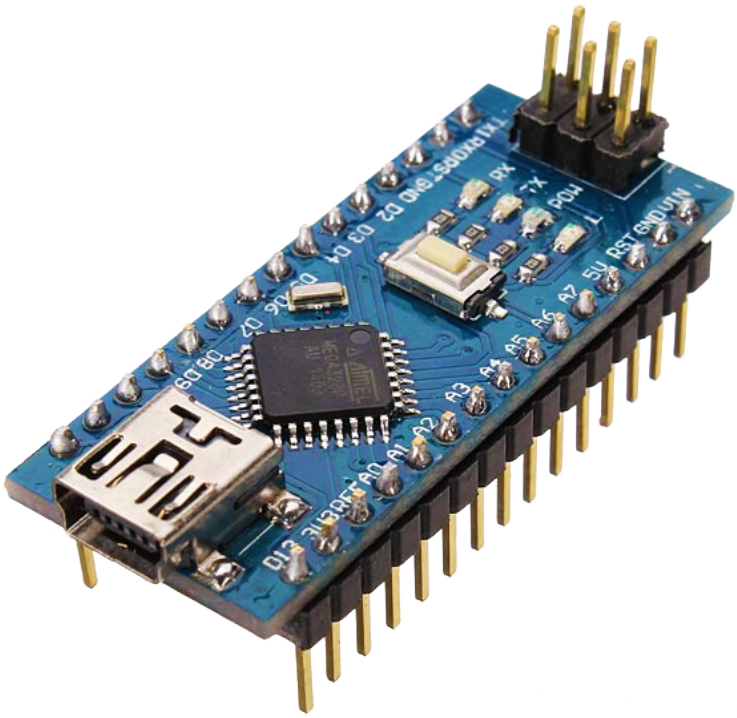


Figura 3.1: *Arduino Nano.*
Fonte: https://www.filipeflop.com/

- 1x **`Caneta Laser Verde`** (Figura 3.2)
  - Na prática só é usada a metade da caneta que contém o diodo emissor do laser.
  - Comprimento de onda: 532nm ± 10
  - Potência máxima de saída: <5mW
  - Preço estimado: **R$ 30,00**.

Figura 3.2: *Caneta Laser Verde.*
Fonte: https://www.dx.com/

- 1x **`Caneta Laser Azul`**[1] (Figura 3.3)
  - Na prática só é usada a metade da caneta que contém o diodo emissor do laser.
  - Comprimento de onda: 405nm ± 10
  - Potência máxima de saída: <5mW
  - Preço estimado: **R$ 30,00**.

Figura 3.3: *Caneta Laser Azul.*
Fonte: https://www.dx.com/

[1] O laser possui uma cor meio violeta, mas, quando apontado para algum tecido branco, seu reflexo fica azul.

- 1x **`Motor DC`** (Figura 3.4)
  - Motor para girar o espelho da harpa. No caso foi usado um motor Motor DC de 12V e 12500 RPM, mas, na prática, não será preciso mais que 6000 RPM.
  - Preço estimado: **R$ 15,00**.

Figura 3.4: *Motor DC.*
Fonte: https://www.dx.com/

- 1x **`Módulo Sensor de Velocidade com Comparador LM393`**[13] (Figura 3.5)
  - Usado para medir a velocidade de rotação do motor.
  - Preço estimado: **R$ 20,00**.

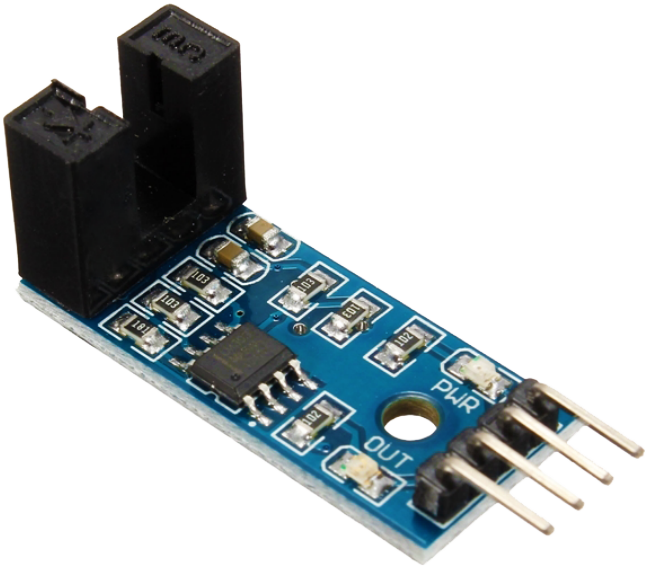


Figura 3.5: *Módulo Sensor de Velocidade com Comparador LM393.*
Fonte: https://www.filipeflop.com/

- 3x **`Transistores NPN TIP120`**[14] (Figura 3.6)
  - Transistores necessários para que o Arduino consiga controlar o motor e os lasers (ver Subseção 3.2.1).
  - Preço estimado por unidade: **R$ 1,50**.

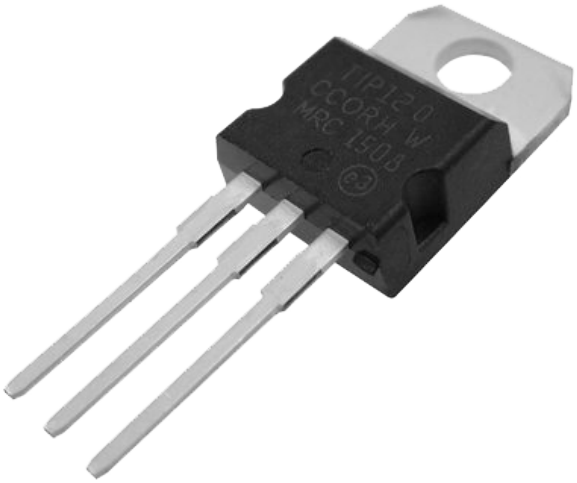


Figura 3.6: *Transistor NPN TIP120.*
Fonte: https://www.filipeflop.com/

- 3x **`Dissipadores de Calor`** (Figura 3.7)
  - Por segurança, é recomendado colocar um dissipador de calor em cada transistor, pois eles costumam esquentar.
  - Preço estimado por unidade: **R$ 2,00**.

Figura 3.7: *Dissipador de Calor.*
Fonte: https://pt.aliexpress.com/

- 1x **`Módulo Micro USB Fêmea`** (Figura 3.8)
  - Módulo para adaptar a porta Mini USB do Arduino Nano para Micro USB. Usado em conjunto com o cabo fabricado visto na Subseção 3.4.1.
  - Preço estimado: **R$ 15,00**.

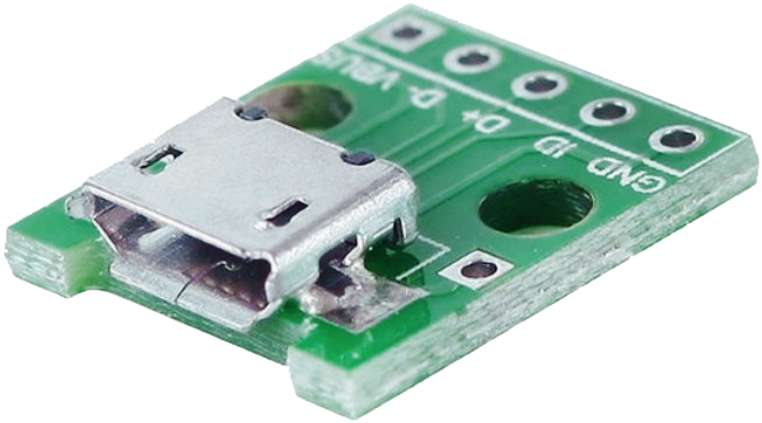


Figura 3.8: *Módulo Micro USB Fêmea.*
Fonte: https://www.baudaeletronica.com.br/

- 1x **`Cabo Mini USB`** (Figura 3.9)
  - Cabo base para a fabricação mostrada na Subseção 3.4.1.
  - Preço estimado: **R$ 10,00**.

Figura 3.9: *Cabo Mini USB.*
Fonte: https://pt.aliexpress.com/

- 1x **`Fonte de 9V e 1A`** (Figura 3.10)
  - Fonte para alimentar o motor e os lasers.
  - Preço estimado: **R$ 20,00**.

Figura 3.10: *Fonte de 9V e 1A.*
Fonte: https://www.filipeflop.com/

- 1x **`Conector P4 Fêmea com Borne`** (Figura 3.11)
  - Conector para fazer a ponte entre a fonte e a LaserHarp.
  - Preço estimado: **R$ 3,00**.

Figura 3.11: *Conector P4 Fêmea com Borne.*
Fonte: https://www.filipeflop.com/

- 1x **`Placa Fenolite Perfurada 5x7cm`** (Figura 3.12)
  - Placa para soldagem de componentes.
  - Preço estimado: **R$ 15,00**.

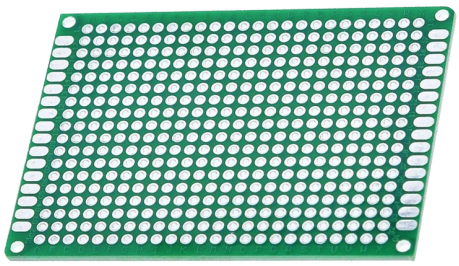


Figura 3.12: *Placa Fenolite Perfurada 5x7cm.*
Fonte: https://pt.aliexpress.com/

- 1x **`Protoboard 400 pontos`** (Figura 3.13)
  - Placa para prototipagem rápida.
  - Preço estimado: **R$ 10,00**.

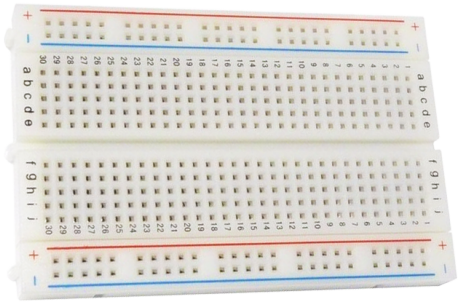


Figura 3.13: *Protoboard 400 pontos.*
Fonte: https://www.filipeflop.com/

- **`Jumpers`** (Figura 3.14)
  - Fios para ligar os componentes e criar o circuito.
  - Preço estimado: **R$ 10,00**.

Figura 3.14: *Jumpers.*
Fonte: https://www.filipeflop.com/

- 2x **`Capacitores Cerâmicos 104`** (Figura 3.15)
  - Capacitores de 100nF para o motor e para o sensor de velocidade.
  - Preço estimado por unidade: **R$ 0,50**.

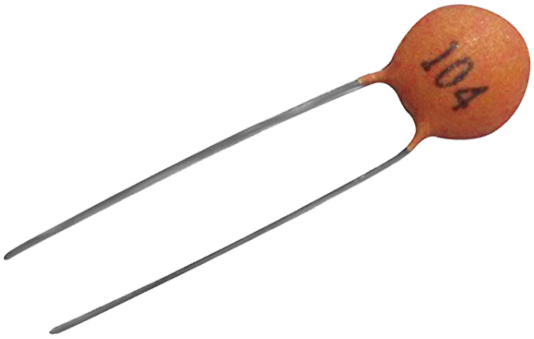


Figura 3.15: *Capacitor Cerâmico 104.*
Fonte: https://pt.aliexpress.com/

- 1x **`LED Verde 5mm`** (Figura 3.16)
  - LED para indicar quando a LaserHarp estiver ligada na tomada.
  - Preço estimado: **R$ 0,30**.

Figura 3.16: *LED Verde 5mm.*
Fonte: https://pt.aliexpress.com/

- 4x **`Resistores de 1k`**Ω (Figura 3.17)
  - Resistores para os transistores e para o LED.
  - Preço estimado por unidade: **R$ 0,05**.

Figura 3.17: *Resistor 1k*Ω.
Fonte: https://www.baudaeletronica.com.br/

Preço total estimado: **R$ 220,00**.

3.2 Circuito

A versão final do circuito da LaserHarp pode ser vista no esquema da Figura 3.18, feito com ajuda do software de código aberto chamado Fritzing[15].

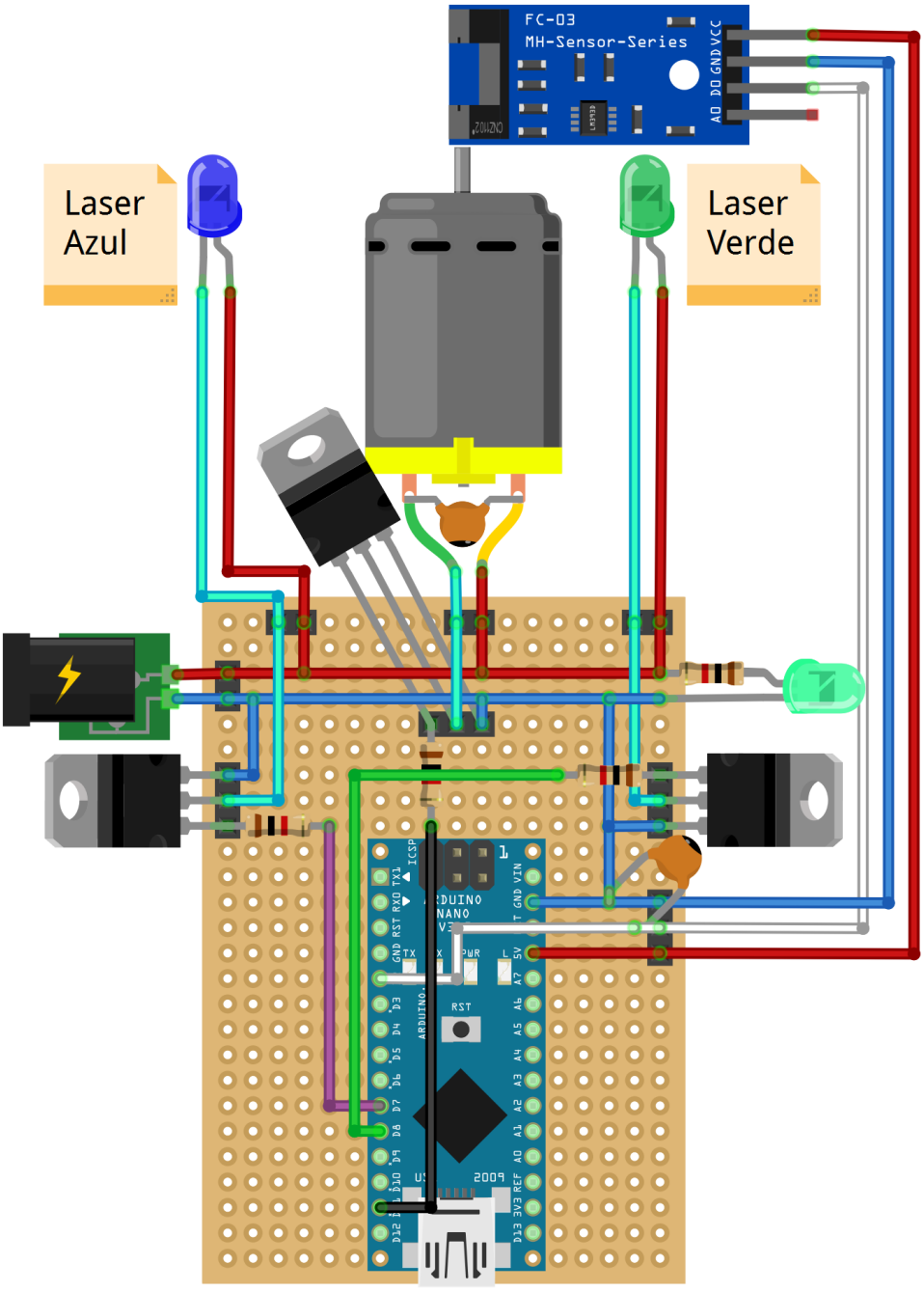


Figura 3.18: *Esquema do circuito da LaserHarp.*

3.2.1 Transistor

O Arduino é capaz de lidar com no máximo 40mA de corrente em cada pino de entrada e saída, o que é insuficiente para controlar componentes como o motor e os lasers. Para contornar esta limitação, foi usado o Transistor `TIP120`, visto na Seção 3.1. Está fora do escopo deste trabalho entender em detalhes o seu funcionamento, basta saber que cada transistor possui 3 pinos (Figura 3.19):

- **`Base`**
  Usado para ligar e desligar o transistor e, consequentemente, o componente que estiver ligado a ele. Deve ser conectado a um pino de saída do Arduino.
- **`Coletor`**
  Pino que deve ser conectado ao terminal negativo do componente em questão.
- **`Emissor`**
  Pino que deve ser conectado ao terra (`GND`) do circuito.

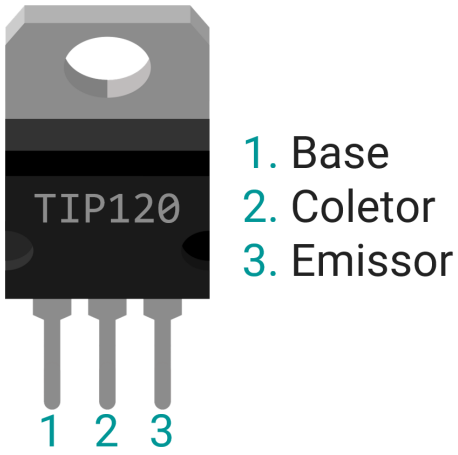
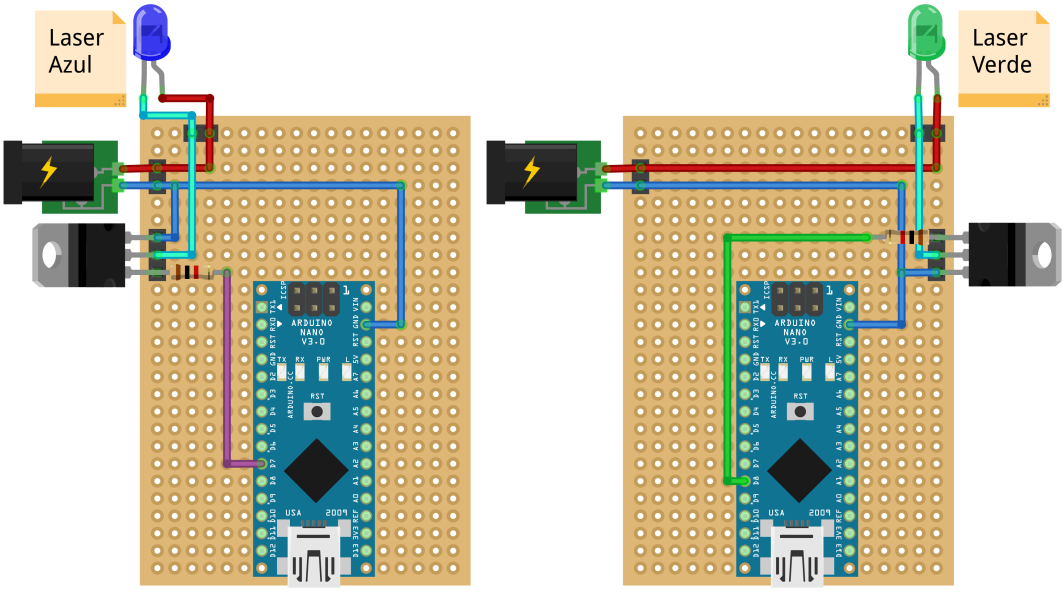


Figura 3.19: *Pinos do `Transistor TIP120`.*

3.2.2 Detalhes

A seguir são mostradas, em detalhes, cada parte separadamente do circuito:

- **`Laser Azul`** (Figura 3.20a)
  Como explicado anteriormente, para que fosse possível o controle dos lasers pelo Arduino, foi utilizado o Transistor `TIP120`. Seu uso pode ser visto nos esquemas da Figura 3.20.
  - O terminal positivo do laser é ligado ao fio positivo da fonte de 9V.
  - O terminal negativo do laser é ligado ao pino Coletor do `TIP120`.
  - O pino Base do `TIP120` é ligado ao fio negativo da fonte de 9V.
  - O pino Emissor do `TIP120` é ligado a um resistor de 1kΩ que é ligado ao pino `7` do `Arduino`.
  - O pino `GND` do Arduino é ligado ao fio negativo da fonte de 9V.
- **`Laser Verde`** (Figura 3.20b)
  O circuito do `Laser Verde` é essencialmente o mesmo do `Laser Azul`, a diferença está em sua posição na placa e no pino do Arduino que é usado.
  - O pino Emissor do `TIP120` é ligado a um resistor de 1kΩ que é ligado ao pino `8` do `Arduino`.



(a) *`Laser Azul`.* **(b)** *`Laser Verde`.*

Figura 3.20: *Esquema dos circuitos dos `Lasers`.*

- **`Motor`** (Figura 3.21)
  O circuito que controla o Motor possui o mesmo padrão do circuito dos lasers, também com a presença de um transistor. A novidade é que deve ser colocado um capacitor cerâmico de 100nF (Figura 3.15) nos terminais do motor para amenizar eventuais interferências eletromagnéticas causadas por ele.
  - O pino Emissor do `TIP120` é ligado a um resistor de 1kΩ que é ligado ao pino `11` do `Arduino`.

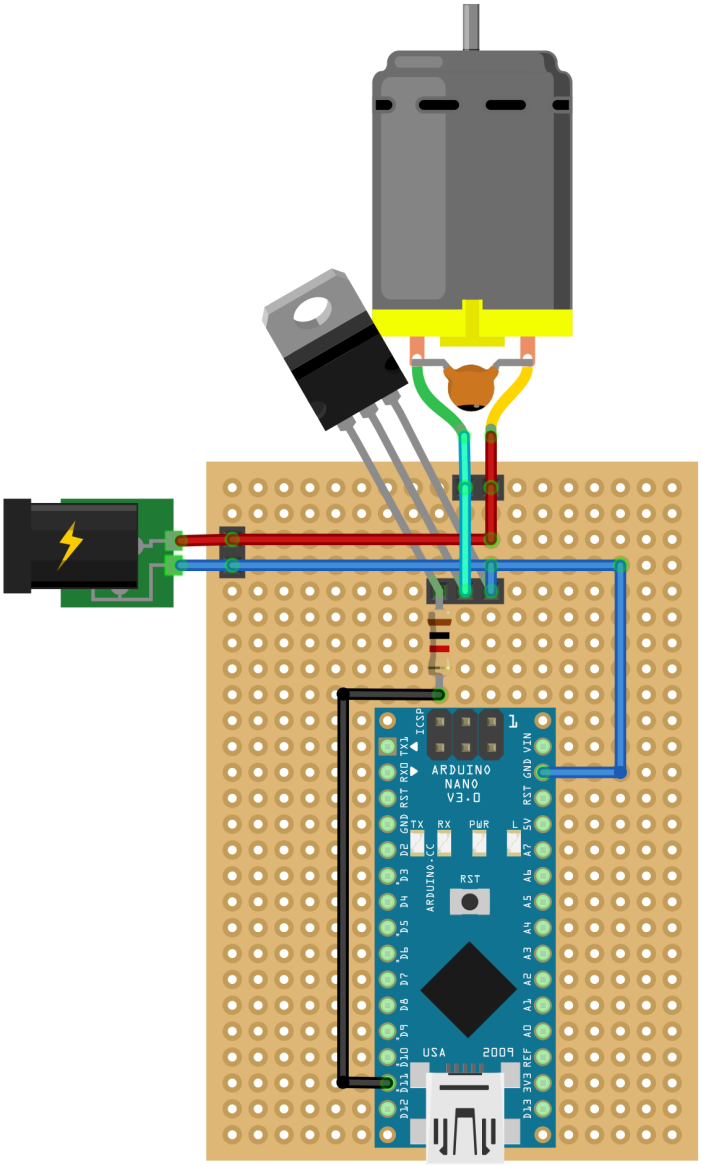


Figura 3.21: *Esquema do circuito do `Motor`.*

- **`Sensor de Velocidade`** (Figura 3.22)
  O circuito do sensor de velocidade é o único que não depende da fonte de 9V, recebendo energia diretamente do pino de 5V do Arduino. O objetivo do sensor é calcular quantas rotações o motor realiza por segundo, ou seja, calcular seu `RPS` (*Revolution Per Second*). Para isso, deve ser acoplado ao motor um disco *encoder*, como o mostrado na Figura 3.26b. Com ele, cada volta completa do motor produzirá um pulso no sensor, que será repassado para o Arduino. Cada pulso é gerado quando a abertura no disco *encoder* passa entre o emissor e o receptor infravermelho presentes na extremidade do sensor. Eventualmente pode acontecer de o Arduino ler mais pulsos do que deveria, isso acontece pois o sensor é bastante sensivel a interferências elétricas externas. Para resolver este problema, deve ser adicionado um capacitor cerâmico de 100nF (Figura 3.15) entre os pinos `GND` e `D0` do sensor.

  - O pino `VCC` do sensor é ligado ao pino `5V` do `Arduino`.
  - O pino `GND` do sensor é ligado ao pino `GND` do `Arduino`.
  - O pino `D0` do sensor é ligado ao pino `2` do `Arduino`.

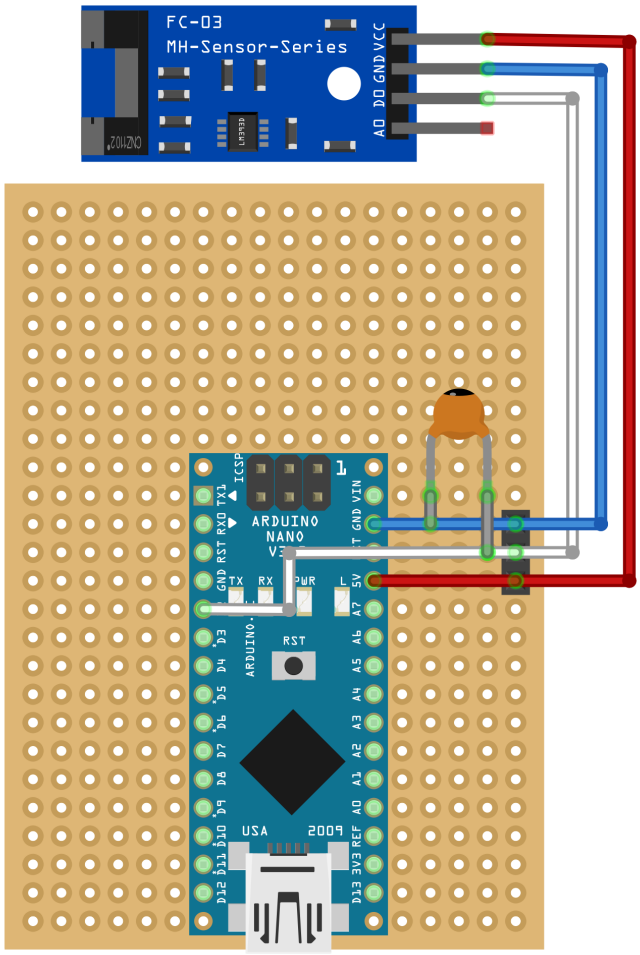


Figura 3.22: *Esquema do circuito do `Sensor de Velocidade`.*

3.3 Códigos

O código que controla todos os componentes da LaserHarp é composto de um único arquivo (“**`hardware/laserharp/laserharp.ino`**”) e foi desenvolvido na linguagem C++ adaptada para Arduino. O funcionamento geral do código está esquematizado na Figura 3.23.

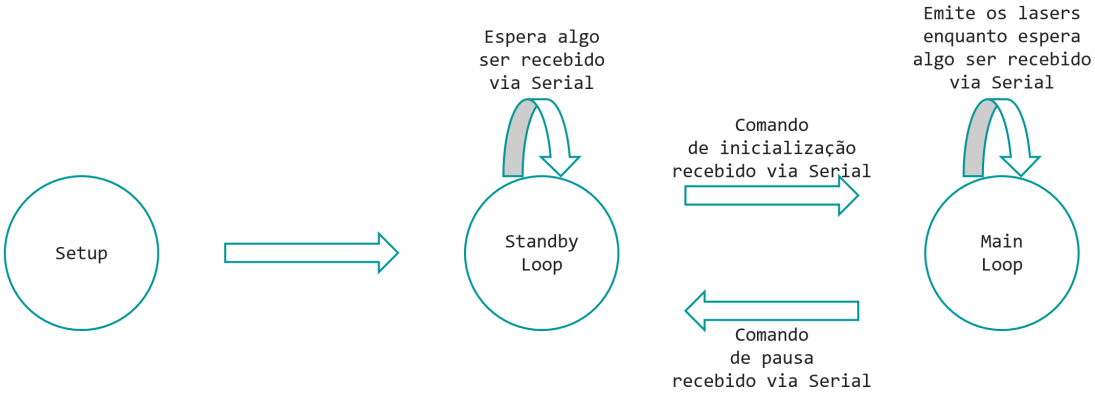


Figura 3.23: *Esquema do funcionamento geral do código do hardware.*

O ciclo de vida do código do hardware é dividido em 3 estados, cada um deles é representado por uma função de mesmo nome:

- **`Setup`**
  É o estado inicial, executado quando a LaserHarp é ligada no computador via USB. Nele são definidos quais são os pinos do Arduino que serão utilizados, assim como é designado qual função deve ser executada quando o Arduino receber um pulso do sensor de velocidade. Após isso, o Arduino envia uma *string* “`LASERHARP`” para o computador, para que este possa estabelecer uma conexão entre os dois no software da LaserHarp. Então o Arduino passa para o estado `Standby Loop`.

- **`Standby Loop`**
  Neste estado o motor e os lasers ficam parados e, a cada 0.1 segundo, o Arduino verifica se alguma mensagem foi recebida pela porta Serial[3]. Quando um comando de inicialização é recebido, o Arduino guarda os dados contidos nele (mais informações na Seção 4.8), aciona o motor e ativa um timer que fica ajustando a velocidade do motor a cada 1 segundo. Então o Arduino passa para o estado `Main Loop`.

- **`Main Loop`**
  Estado em que os lasers são gerados, ou seja, as cordas da harpa aparecem. Enquanto isso, o Arduino fica verificando constantemente se um comando de pausa (*string* “`PAUSE`”) é recebido pela porta Serial. Caso positivo, o timer é desativado e o motor é parado. Então o Arduino passa para o estado `Standby Loop`.

3.3.1 Funções

No código da LaserHarp temos as seguintes funções:

- **`serial_read()`**
  Verifica se há alguma mensagem na porta serial. Caso positivo, guarda a mensagem na variável "`serial_buffer`".
- **`serial_flush()`**
  Limpa a porta serial.
- **`is_equal(char *s1, char *s2)`**
  Verifica se as *strings* passadas como parâmetro são iguais.
- **`parse_values()`**
  Extrai as informações do comando contido na variável "`serial_buffer`". Na prática, este comando será sempre o de inicialização (ver Seção 4.8).
- **`calculateSpeed()`**
  Ela recalcula a velocidade que deve ser passada para o motor de acordo com o `RPS` calculado com base no *feedback* dado pelo sensor de velocidade. O objetivo disso é manter sempre uma velocidade constante, mesmo se houver interferência externa. Também calcula o tempo que demora para o motor girar 1°.
- **`delay_angle(float angle)`**
  Espera a quantidade de tempo suficiente para o motor girar o ângulo passado como parâmetro.
- **`timerIsr()`**
  Função executada a cada 1 segundo enquanto o Arduino estiver no estado `Main Loop`. Nela é executada a função `calculateSpeed()` e a variável "*pulses*" é zerada.
- **`pulse()`**
  Incrementa o valor da variável "*pulses*". Executada a cada pulso recebido do sensor de velocidade.
- **`motor_run(float speed)`**
  Aciona o motor com a velocidade passada como parâmetro (um número real de 0 a 255).
- **`motor_stop()`**
  Para o motor.
- **`emit_laser(int color)`**
  Emite o laser passado como parâmetro por uma pequena fração de tempo.
- **`make_lasers()`**
  É a função mais importante do código. Ela gera todos os lasers, criando, assim, as cordas da harpa. O função pode ser vista no Programa 3.1.
- **`setup()`**
  Representa o estado `Setup` visto na Figura 3.23.
- **`loop()`**
  Alterna entre os estados `Standby Loop` e `Main Loop`.
- **`standby_loop()`**
  Representa o estado `Standby Loop` visto na Figura 3.23.
- **`main_loop()`**
  Representa o estado `Main Loop` visto na Figura 3.23.

Programa 3.1 Função que gera os lasers da harpa.[a]

```
1   // n_strings: Número de cordas.
2   // angle: Ângulo entre as cordas.
3   // notes: Vetor binário em que "1" representa uma nota verde e "0" uma azul.
4
5   // A inclinação que o espelho deve estar para formar o primeiro laser.
6   float first_angle = 45-(n_strings-1)*angle/4;
7
8   // O ângulo que o motor deve girar entre duas emissões de laser.
9   float step_angle = angle/2;
10
11  // Obs: O motor gira no sentido horário, com o emissor de laser verde do seu
        lado direito e o azul do esquerdo.
12
13  /*********************************************************************/
14  void make_lasers() {
15
16    // Espera o espelho chegar no ângulo da primeira nota,
17    // apontando para o emissor de laser verde.
18    delay_angle(first_angle);
19
20    // Lasers verdes
21    for (i = 0; i < n_strings; i++) {
22      // Verifica se o elemento i do vetor corresponde a uma nota verde.
23      if (notes[i]-41 == LASER_GREEN) {
24        emit_laser(LASER_GREEN);
25        delay_angle(step_angle - laser_angle);
26      } else {
27        delay_angle(step_angle);
28      }
29      // Espera o espelho chegar na posição da próxima nota.
30    }
31
32    // Espera o espelho chegar no ângulo da primeira nota,
33    // apontando para o emissor de laser azul.
34    delay_angle(2*first_angle - step_angle + 180);
35
36    // Lasers azuis
37    for (i = 0; i < n_strings; i++) {
38      // Verifica se o elemento i do vetor corresponde a uma nota azul.
39      if (notes[i]-41 == LASER_BLUE) {
40        emit_laser(LASER_BLUE);
41        delay_angle(step_angle - laser_angle);
42      } else {
43        delay_angle(step_angle);
44      }
45      // Espera o espelho chegar na posição da próxima nota.
46    }
47  }
```

[a] A função foi ligeiramente modificada para facilitar seu entendimento.

3.4 Desenvolvimento

No início do projeto, foi planejado o uso de um Motor de Passo do tipo NEMA 17[10] (Figura 3.24), junto com um Driver DRV8825[30] (Figura 3.25).

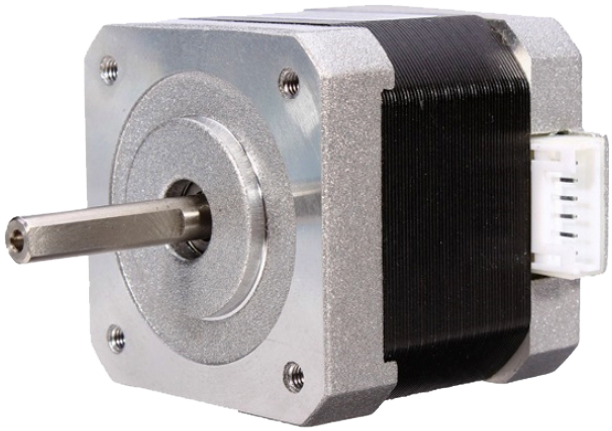


Figura 3.24: *Motor de Passo NEMA 17.*
Fonte: https://www.filipeflop.com/

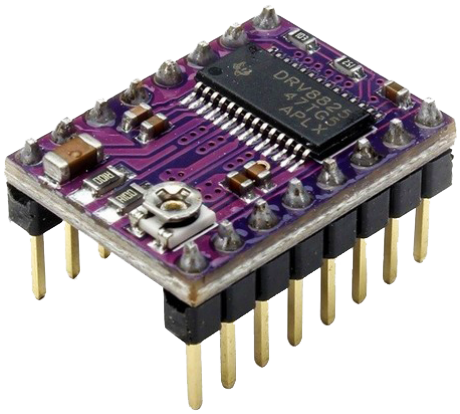


Figura 3.25: *Driver DRV8825 para motores de passo.*
Fonte: https://www.filipeflop.com/

A maioria dos projetos de harpa laser existentes na internet utilizam motores de passo devido a facilidade e precisão com que eles lidam com ângulos. Cada passo completo desse motor tem 1.8°. Como cada volta possui 360°, então, para completar 1 volta, o motor precisa dar 200 passos. Há também a possibilidade de aumentar a resolução de cada passo subdividindo-o em até 32 partes, tecnica conhecida como *microstepping*.

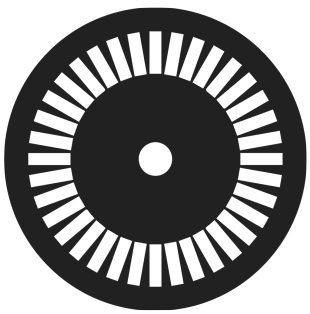
No caso da LaserHarp, seria necessário subdividir cada passo em 4 partes de 0.45°cada, para que assim fosse possível gerar lasers com um intervalo mínimo de 0.9°. Desse modo, o motor precisaria dar 800 passos para completar uma volta. Cada passo do motor é gerado por 1 pulso recebido do driver. De acordo com informações do fabricante[10], o motor é capaz de receber até 10000 pulsos por segundo (PPS). Isto nos leva à conclusão de que o motor pode realizar até 12,5 rotações por segundo (RPS). Esta velocidade de rotação é baixa para a LaserHarp, pois assim o fenômeno de persistência da visão não teria efeito.

Diante da limitação do motor de passo, optou-se pela utilização de um Motor DC. Este tipo de motor é bastante usado em carrinhos, drones, furadeiras, entre outros equipamentos.

Além de atingir velocidades muito altas, ele é mais leve e mais barato. A troca realizada resultou em uma economia de cerca de R$60,00. A desvantagem foi a perda da precisão que se tinha com a utilização do motor de passo.

O Motor DC sozinho não é capaz de saber o quanto ele já girou. Para contornar esta limitação, é necessário utilizar sensores que identifiquem a velocidade do motor e o momento em que ele passa por uma determinada posição. Com essas informações, é possível saber com exatidão em que posição angular o motor está a qualquer momento.

Eis então que foi cogitado o uso do Sensor de Velocidade visto na Seção 3.1. Para seu funcionamento, é necessário que um disco *encoder* seja acoplado no eixo do motor. Inicialmente foi tentada a utilização de um disco com 36 aberturas (Figura 3.26a), 1 a cada 10°. A vantagem de se utilizar um disco com várias aberturas é que assim a quantidade de voltas por segundo dadas pelo motor é contada com maior precisão. Por exemplo, pode-se contabilizar que o motor realizou `50,25` voltas em 1 segundo, enquanto um *encoder* com 1 abertura contabilizaria apenas `50` voltas. Isto influencia diretamente na criação do lasers, momento em que é necessário saber com exatidão o quanto se tem que esperar para o espelho preso ao motor chegar em determinadas posições. Porém, por algum motivo inexplicável, o sensor foi incapaz de trabalhar em conjunto com o *encoder* de 36 aberturas. Ele simplesmente não gerava nenhum pulso para o Arduino, impedindo, assim, de se calcular a velocidade do motor. Então, para contornar este problema misterioso, foi modelado um novo disco com apenas 1 abertura (Figura 3.26b). Consequentemente, perdeu-se a precisão no cálculo da velocidade, mas, para compensar, surgiu uma grande vantagem. Como o disco possui apenas 1 abertura, ela pode ser usada para determinar a posição inicial do espelho preso ao motor (horizontal apontado para cima). Assim torna-se possível realizar duas tarefas com um único sensor: calcular a velocidade do motor e identificar sua posição inicial. Com isso, o trabalho com o hardware e seu código foi simplificado.



(a) *Disco Encoder com 36 aberturas.*

(b) *Disco Encoder com 1 abertura.*

Figura 3.26: *Discos Encoder produzidos.*

Geralmente os motores DC são utilizados em conjunto com um Driver Ponte H, como o L9110s[4] (Figura 3.27). Com este driver é possível controlar 2 motores de uma vez, além de ser possível escolher o sentido que cada um gira. Como na LaserHarp só é necessário 1 motor e ele sempre girará no mesmo sentido, o uso do driver foi descartado, dando lugar a um simples transistor TIP120 (Subseção 3.2.1) para controlar a velocidade do motor pelo Arduino, causando mais uma economia de cerca de R$15,00.

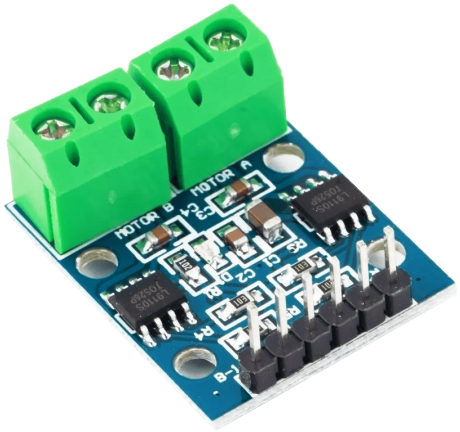


Figura 3.27: *Driver Ponte H Duplo L9110s.*
Fonte: https://www.dx.com/

3.4.1 Cabo USB modificado

Como mostrado na Seção 3.1, a LaserHarp utiliza um Arduino Nano para controlar seus componentes eletrônicos. Ele possui uma entrada Mini USB. Isto significa, obviamente, que a entrada é compatível apenas com cabos Mini USB, como o visto na Figura 3.9. Acontece que este tipo de cabo já não é tão utilizado. Os aparelhos eletrônicos mais novos costumam utilizar um cabo Micro USB, que possui um conector menor. Pensando nisso, foi desenvolvida uma maneira de conectar um cabo Micro USB no Arduino Nano. Essa conexão se dá por meio de um cabo Mini USB (Figura 3.9) em que a ponta USB convencional foi substituida por um *plug* capaz de ser conectado a um módulo com uma entrada Micro USB (Figura 3.8). Com isso, é possível conectar a LaserHarp ao computador utilizando um cabo Micro USB de celular, por exemplo. O resultado pode ser visto na Figura 3.28.

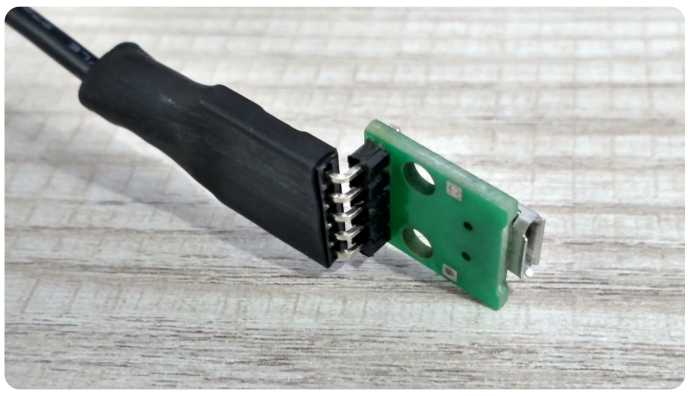


Figura 3.28: *Ponta do cabo USB modificado encaixada no módulo Micro USB fêmea.*

3.4.2 Problemas

Os emissores de laser utilizados, como visto na Seção 3.1, são canetas laser. Porém, o propósito para o qual elas foram construídas difere da necessidade da LaserHarp. O esperado é que o laser seja acionado manualmente. Por isso o hardware interno da caneta não foi desenvolvido pensando-se em piscar o laser rapidamente.

A caneta laser, quando acionada, demora uma curta fração de tempo para atingir seu brilho máximo. Isto ocorre tão rápido que é imperceptível a olho nu. Porém, no contexto da LaserHarp, em que a caneta tem que ligar e desligar num período bastante especifico de tempo para o laser ser refletido por um espelho girando a 55 rotações por segundo, a demora para o brilho máximo ser atingido se torna um grande problema. Acontece que o tempo que o laser deve ficar ligado para gerar uma corda é inferior a essa demora. Como consequência, as cordas geradas não possuem o alto brilho esperado de uma caneta laser. Com isso fica difícil de se utilizar a LaserHarp para o propósito para o qual ela foi construída, já que um brilho forte é essencial para ser captado pela câmera do computador para, assim, realizar um processamento das imagens com a finalidade de se descobrir quais cordas estão sendo tocadas. Portanto, como objetivo futuro, para resolver este problema, o hardware interno da caneta precisa ser estudado em busca de modificações que possam aumentar a velocidade com que o brilho máximo do laser é atingido.

Capítulo 4

Software

Para controlar a LaserHarp, foi criado um software (Figura 4.1) para computador inteiramente na linguagem de programação **JavaScript**[24]. Não é tão comum que esta linguagem seja usada na criação de aplicações *desktop*, mas isso foi possível graças a um *Framework* chamado **`Electron`**[16], que permite criar programas utilizando todas as ferramentas disponíveis para desenvolvimento *Web*. Ou seja, todo o *Front-end* é feito com HTML e CSS e todo o *Back-end* é feito com Javascript. Isto significa que todas as bibliotecas existentes para aplicações *Web* podem ser aproveitadas no desenvolvimento *desktop*, tornando-o mais simples do que se fosse feito do modo tradicional.

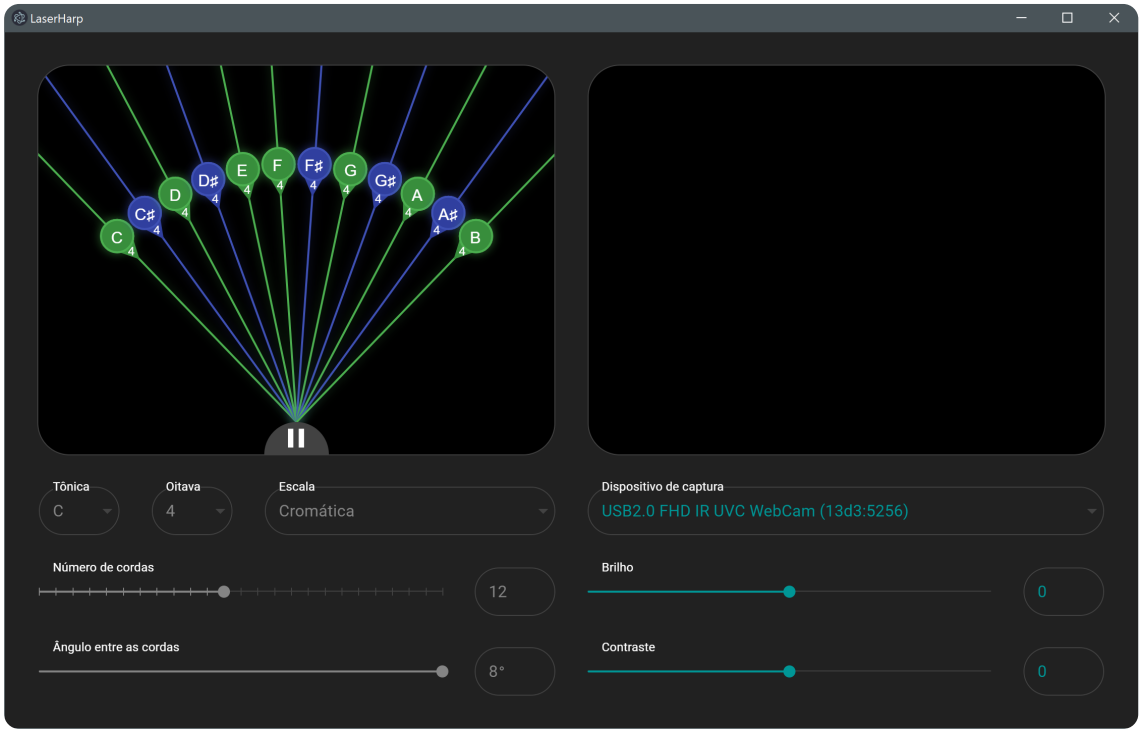


Figura 4.1: *Janela do software da LaserHarp.*

A principal biblioteca utilizada foi o **`React`**[12], que permite a criação de interfaces de usuário de maneira simples e eficiente. Dentre as vantagens obtidas ao utilizá-la, destacam-se:

- Facilidade em se criar componentes personalizados, como botões, *sliders* e caixas de texto.
- Definição de variáveis globais que definem o estado do programa (`state`).
- *Front-end* responsivo, ou seja, que consegue se adaptar a qualquer tamanho de janela.

A terceira vantagem é especialmente útil no caso da LaserHarp pois, caso o usuário queira usar o software em um telão, durante alguma apresentação musical, por exemplo, o desenho da harpa e a imagem da câmera ficarão maiores de acordo com o espaço disponível (Figura 4.2). Da mesma forma, o usuário pode optar por reduzir o tamanho da janela do programa de modo a liberar espaço na tela para outros programas (Figura 4.3).

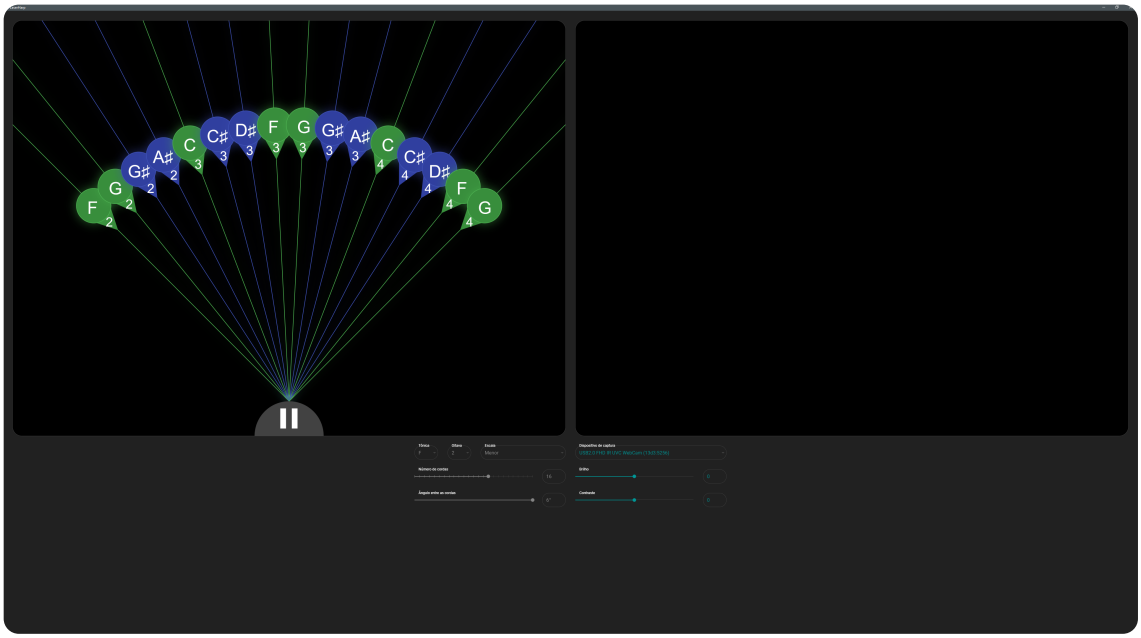


Figura 4.2: *Janela do software em uma tela de 40 polegadas.*

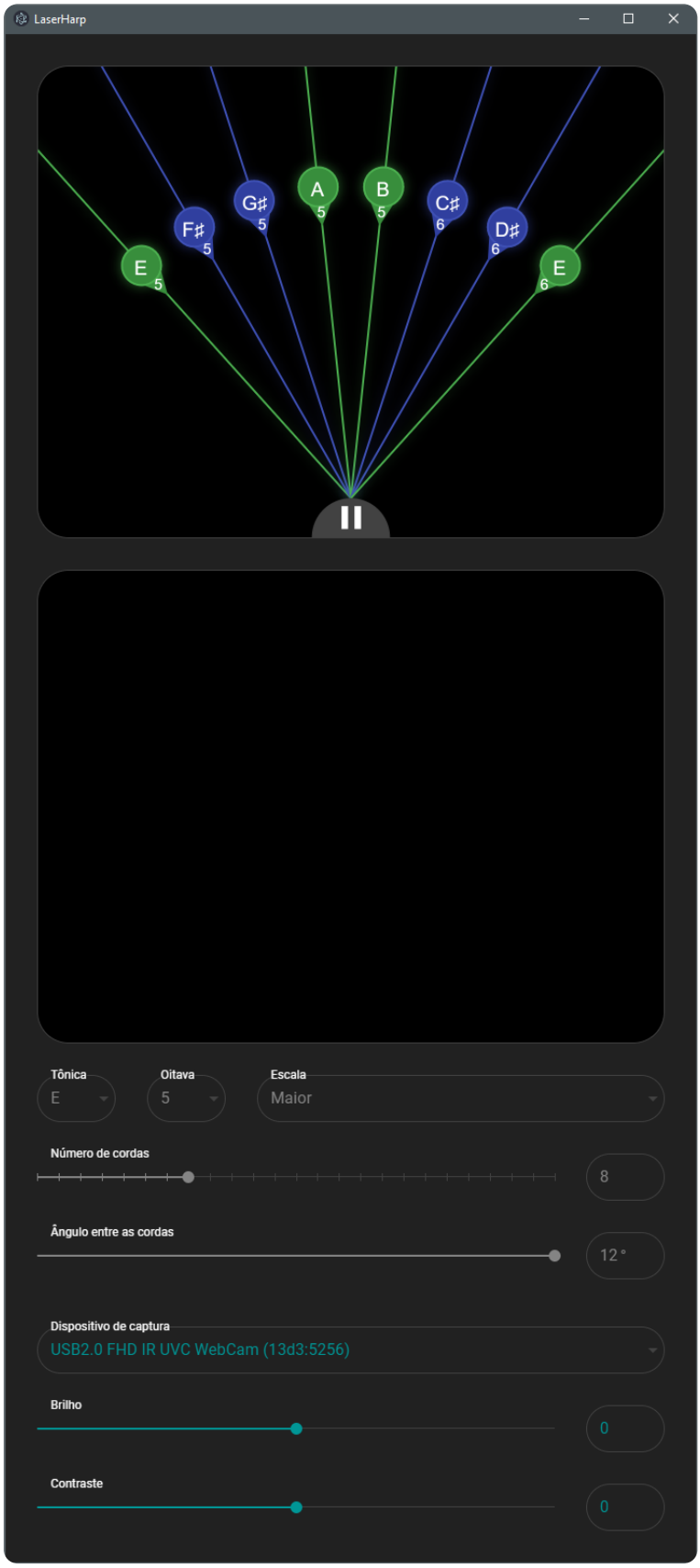


Figura 4.3: *Janela do software em tamanho reduzido.*

4.1 Códigos

O software é composto por 10 arquivos de códigos e 2 arquivos de configurações, todos sob a pasta "**`software/laserharp`**". A estrutura em árvore do código pode ser vista na Figura 4.4.

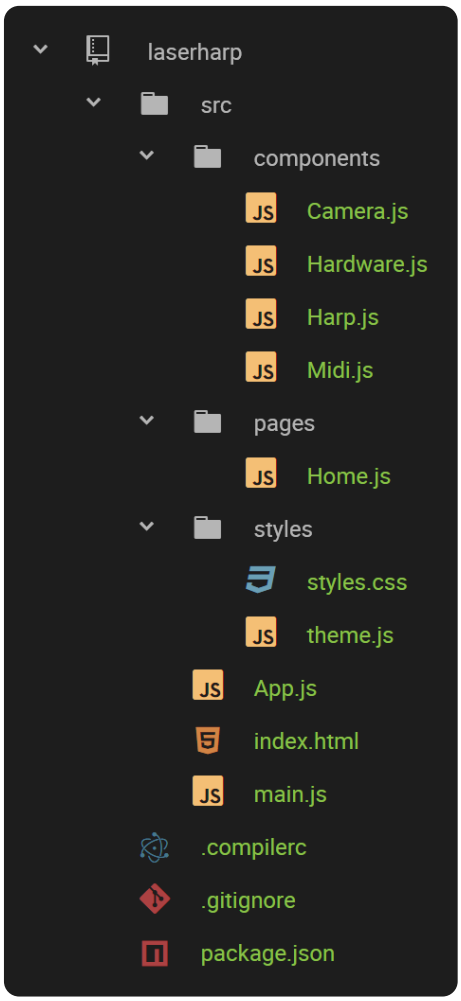


Figura 4.4: *Estrutura em árvore do código.*

- **`Camera.js`**
  Possui a classe `Camera`, cujo objetivo é tratar as imagens obtidas pela câmera e exibi-las no programa (ver Seção 4.7).

- **`Hardware.js`**
  Contém a classe `Hardware` com as funções responsáveis pela comunicação serial[3] com a LaserHarp (ver Seção 4.8).

- **`Harp.js`**
  Possui a classe `Harp`, responsável pelo desenho da harpa mostrado no programa. Aqui são feitos todos os cálculos para se saber quais notas musicais devem ser usadas com base nas informações passadas pelo usuário. Além disso, esta classe possui a função que gera a *string* do comando de inicialização que é enviada para a LaserHarp (ver Seção 4.8).

- **`Midi.js`**
  Contém a classe `Midi` com as funções referentes ao protocolo MIDI (ver Seção 4.6).
- **`Home.js`**
  Possui a classe `Home`, que é a principal de todo o programa. Nela são definidas todas as variáveis globais e toda a interface gráfica, assim como todos os `inputs` e as funções que definem seus comportamentos.
- **`styles.css`**
  Arquivo CSS com algumas configurações gerais para o HTML.
- **`theme.js`**
  Possui configurações de aparência do programa.
- **`App.js`**
  Renderiza o arquivo `Home.js` junto com o arquivo `theme.js`.
- **`index.html`**
  Página HTML responsável por renderizar a aplicação em React (arquivo `App.js`) juntamente com o arquivo `styles.css`.
- **`main.js`**
  Contém os códigos que são executados assim que o programa é inicializado. Possui a responsabilidade de criar uma janela para o programa e carregar o conteúdo da página `index.html`.
- **`.compilerc`**
  Arquivo de configuração do `electron`[16].
- **`package.json`**
  Arquivo JSON[31] com todas as configurações do software. Aqui estão definidas todas as dependências do programa, ou seja, o nome e a versão de todos os pacotes utilizados (ver Seção 4.2).

4.2 Pacotes utilizados

Para o desenvolvimento do software, alguns pacotes *open-source* tiveram que ser usados:

- **`material-ui`**[23]
  Disponibiliza diversos componentes baseados no padrão Material Design[17].
- **`easymidi`**[21]
  Pacote que contém funções que permitem o envio de comandos MIDI para outras aplicações (Ver Seção 4.6).
- **`electron`**[16]
  Framework para criação de aplicações *desktop* utilizando ferramentas de desenvolvimento *Web*: HTML, CSS e JavaScript.
- **`konva`**[22]
  Permite a criação de desenhos complexos (Ver Seção 4.5).
- **`opencv4nodejs`**[20]
  Biblioteca OpenCV[26] para Node.JS com funções para realizar processamento de imagens (Ver Seção 4.7).
- **`react`**[12]
  Biblioteca para construir interfaces de usuário.
- **`serialport`**[25]
  Possibilita a comunicação serial[3] do software com a LaserHarp (Ver Seção 4.8).
- **`babel`**[8]
  Transpilador que converte o código da nova versão do JavaScript, chamada EcmaScript 6 (ou ES6), em um código compatível com versões anteriores do JavaScript.

4.3 Como executar

Para executar o programa, primeiramente é necessário instalar os seguintes softwares:

- **`Node.js`**[27]
  Interpretador de JavaScript.
- **`Yarn`**[33]
  Gerenciador de pacotes para JavaScript.

Em seguida, deve-se abrir um terminal, ir para a pasta onde estão os códigos (“`software/laserharp`”) e executar os seguintes comandos:

- **`yarn install`**
  Para instalar todos os pacotes necessários (ver Seção 4.2).
- **`yarn run opencv-rebuild`**
  Para instalar o pacote OpenCV[26] para JavaScript (ver Seção 4.7).
- **`yarn run serialport-rebuild`**
  Para instalar o pacote `serialport` (ver Seção 4.8).
- **`yarn start`**
  Para inicializar o programa.

O software é compatível com Linux, Mac e Windows. Porém, como o Window não fornece portas MIDI virtuais nativamente (ver Seção 4.6), se faz necessário o uso de mais um programa externo chamado `loopMIDI`[11] (Figura 4.5). Para utilizá-lo junto com o programa da harpa, basta digitar “`LASERHARP`” no campo “`New port-name`” e clicar no botão “`+`”. O `loopMIDI` deve ficar aberto durante todo o tempo em que o programa da harpa for utilizado.

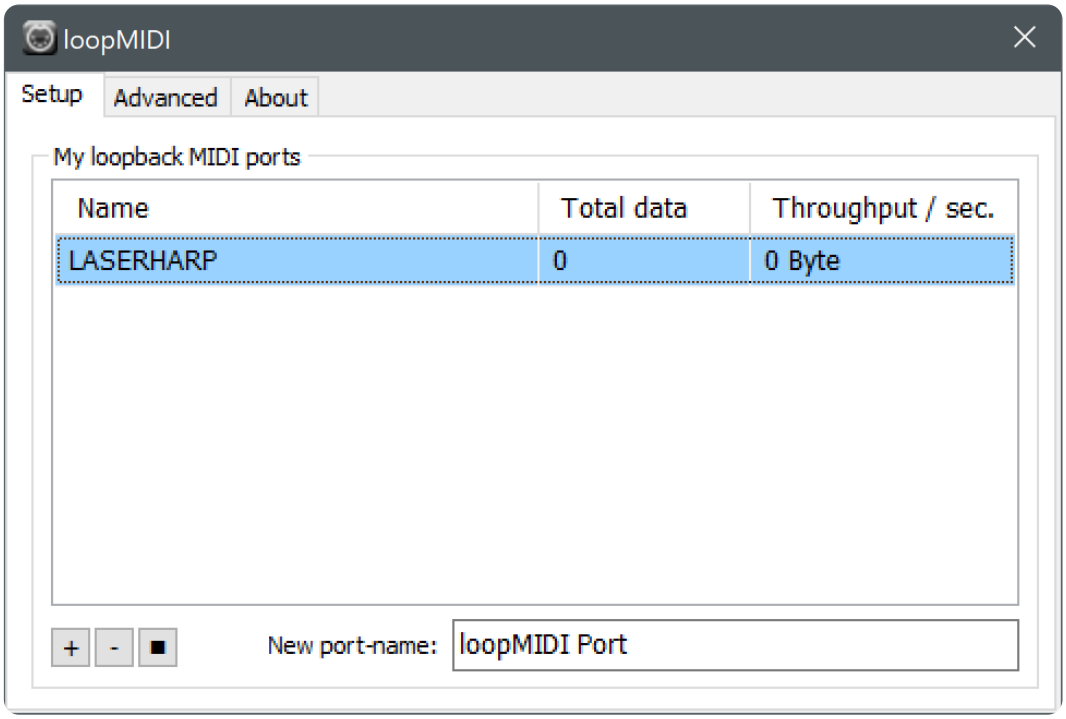


Figura 4.5: *Programa `loopMIDI`.*

4.4 Como usar

O programa foi desenvolvido de modo a deixar seu uso bastante intuitivo. Primeiramente, a LaserHarp deve ser conectada ao computador por meio de um cabo USB. Em seguida, deve-se pressionar o botão "`Procurar LaserHarp`" para que o programa a reconheça, caso isso não seja feito automaticamente. A partir desse ponto o usuário é livre para escolher os valores que serão transmitidos para a LaserHarp para que os lasers sejam gerados. Mais informações sobre como se dá esta transmissão estão presentes na Seção 4.8. Todas as alterações feitas nos valores são refletidas imediatamente no desenho da harpa.

Os valores (juntamente com as variáveis que os representam) são os seguintes:

- **Tônica** (`tonic`)
  - Primeira nota da escala musical.
  - Valores possíveis (veja a Figura 4.6):
    - "**C**": Nota Dó.
    - "**C♯**": Nota Dó Sustenido, que é equivalente a Ré Bemol "**D♭**".
    - "**D**": Nota Ré.
    - "**D♯**": Nota Ré Sustenido, que é equivalente a Mi Bemol "**E♭**".
    - "**E**": Nota Mi.
    - "**F**": Nota Fá.
    - "**F♯**": Nota Fá Sustenido, que é equivalente a Sol Bemol "**G♭**".
    - "**G**": Nota Sol.
    - "**G♯**": Nota Sol Sustenido, que é equivalente a Lá Bemol "**A♭**".
    - "**A**": Nota Lá.
    - "**A♯**": Nota Lá Sustenido, que é equivalente a Si Bemol "**B♭**".
    - "**B**": Nota Si.

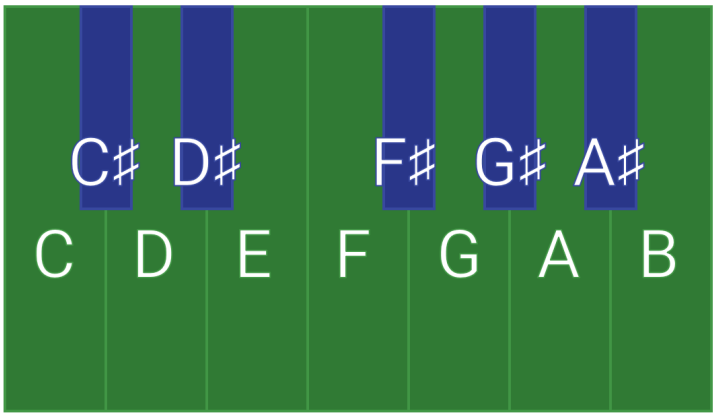


Figura 4.6: *Notas de um teclado com as cores dos lasers da LaserHarp.*

- **Oitava** (`octave`)
  - Um número de 0 a 8 especificando qual oitava a tônica escolhida se encontra. Cada oitava é um conjunto de 12 notas, como visto na Figura 4.7.

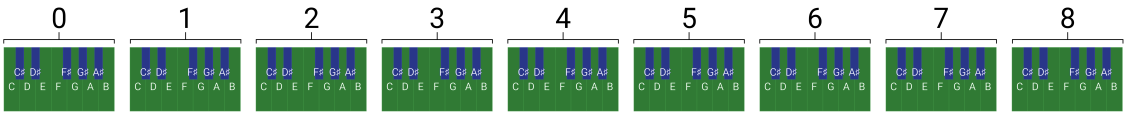


Figura 4.7: *Todas as 9 oitavas que podem ser escolhidas dispostas lado a lado, como em um teclado musical.*

- **Escala** (`scale_id`)
  - Padrão da sequência de notas que serão representadas pelos lasers.
  - No código há um vetor com algumas escalas predefinidas. A variável `scale_id` indica o índice da escala escolhida.
  - Escalas disponíveis: "**`Cromática`**", "**`Maior`**", "**`Menor`**", "**`Pentatônica Maior`**" e "**`Pentatônica Menor`**".
  - Um exemplo de escala pode ser visto no Programa 4.1.
  - A variável `pattern` (padrão) é uma codificação binária indicando quais notas estão presentes na escala.
  - A aplicação do padrão da escala Pentatônica Menor está exemplificada na Figura 4.8.
    - 1: A nota está presente na escala.
    - 0: A nota não está presente na escala.

Programa 4.1 Exemplo de escala.

```
1  {
2    name: "Pentatônica Menor",
3    pattern: [1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0],
4  }
```

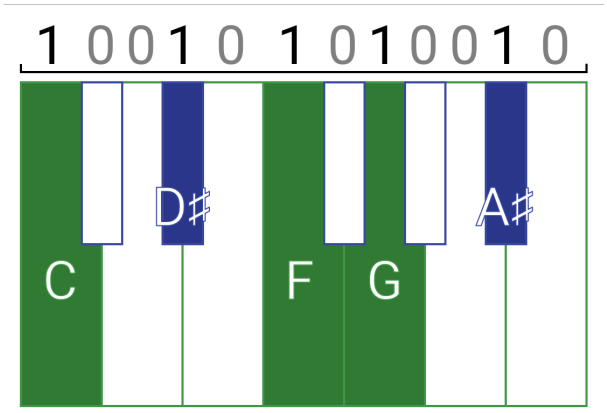


Figura 4.8: *Aplicação do padrão da escala Pentatônica Menor.*

- **Número de cordas** (`n_strings`)
  - Número inteiro de 1 a 25 que indica a quantidade de lasers que devem ser gerados pela LaserHarp.
- **Ângulo entre as cordas** (`angle_strings`)
  - Número real positivo que indica o ângulo entre cada par de lasers adjacentes.
  - O ângulo é limitado superiormente por `90/(n_strings - 1)`, ou seja, o ângulo máximo é tal para que a soma de todos os ângulos seja no máximo 90°.

Quando os valores escolhidos são os mostrados na Figura 4.10, as notas musicais são selecionadas de acordo com o esquema da Figura 4.9.

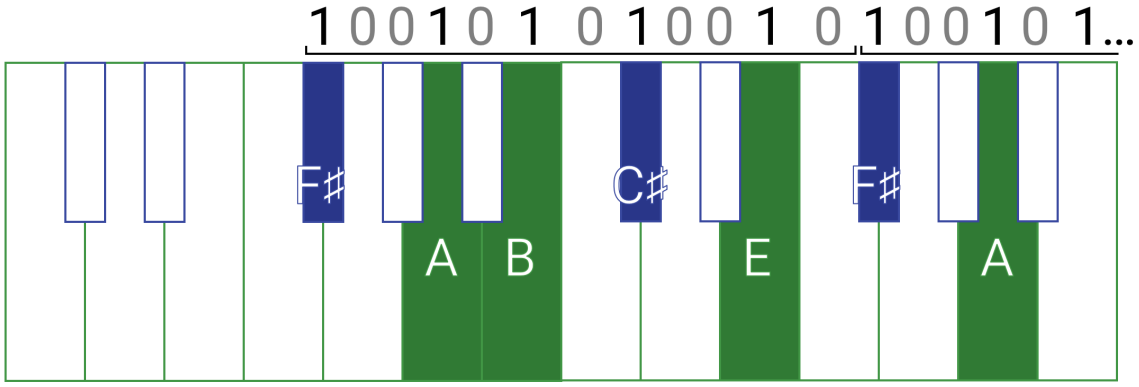


Figura 4.9: *Exemplo da aplicação de valores.*

Figura 4.10: *Exemplo de valores escolhidos.*

4.5 Interface gráfica

O software da LaserHarp foi desenvolvido de modo a ser o mais intuitivo possível. Para cumprir tal objetivo, a interface gráfica do programa foi toda desenhada seguindo um padrão de design chamado **Material Design**[17]. Tal padrão foi anunciado pela Google em uma conferência em 2014 e, desde então, vem sendo largamente usado em diversos aplicativos para as mais variadas plataformas.

O padrão tem como característica principal o uso de elementos simples, porém bonitos e intuitivos. Ele possui também uma preocupação especial com as medidas dos componentes e as distâncias entre eles fazendo com que todos os valores tenham relação um com o outro, de modo que tudo tenha uma razão de ser. Como resultado disso, pode-se ver (Figura 4.1) que os componentes ficaram todos harmonicamente dispostos na janela do programa. Este objetivo foi atingido com o auxílio do pacote `material-ui`[23], que disponibiliza vários componentes baseados no Material Design para uso.

Outro fator importante na criação de uma boa interface é a escolha da paleta de cores. No caso, ela foi escolhida pensando-se no contexto geral da LaserHarp: As cores principais são justamente as mesmas cores dos lasers e a cor de fundo é escura assim como deve ser o ambiente em que a harpa precisa estar para ser vista funcionando com qualidade (Veja a Figura 4.11).

Figura 4.11: *Paleta de cores do software da LaserHarp com os respectivos valores em hexadecimal.*

Para fazer o desenho dos lasers na interface, foi utilizado o pacote **konva**[22]. Cada vez que um valor é alterado nos *inputs*, o desenho é atualizado para refletir tal alteração. Assim o usuário pode ter uma noção de como a harpa se comportará antes de iniciá-la. Além disso, é possível testar facilmente a interação do software com outros programas de áudio: o desenho da harpa é interativo[1], basta clicar sobre cada laser para gerar um comando MIDI com a nota musical atribuída a ele (Veja a Figura 4.12).

[1] Bônus: Ao se passar o mouse por cima de um laser, é mostrada a posição que o espelho tem que estar para refleti-lo.

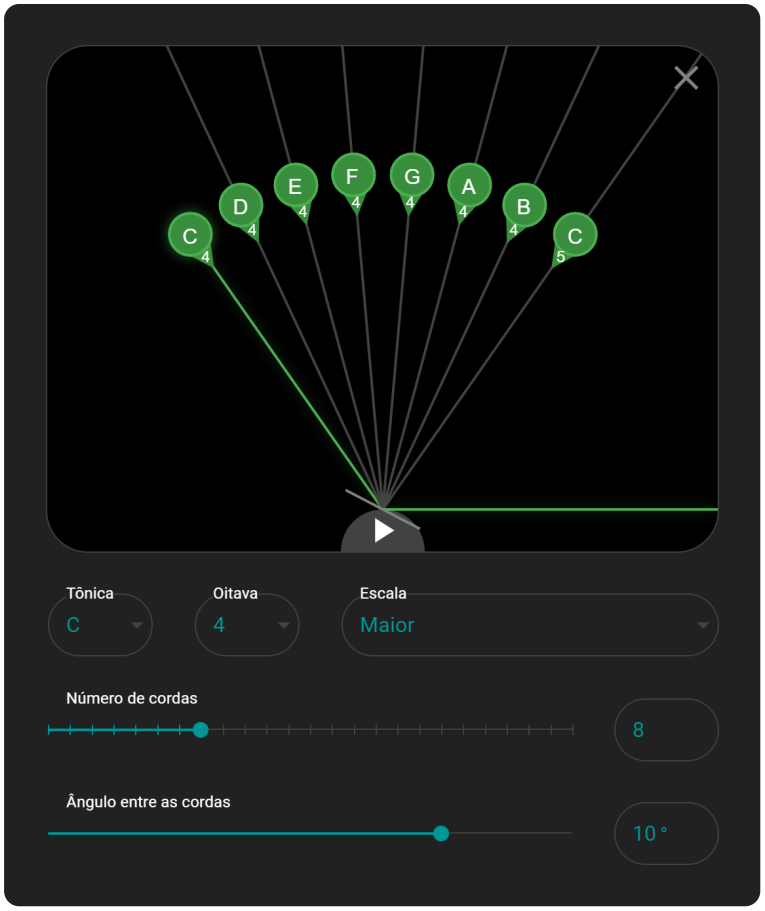


Figura 4.12: *O laser C4 foi clicado, gerando o comando MIDI visto no Programa 4.2.*

4.6 MIDI

MIDI (*Musical Instrument Digital Interface*) é um protocolo que permite a comunicação entre equipamentos musicais e um computador. Normalmente este tipo de comunicação é usado com teclados musicais, mas pode ser usado também com outros instrumentos, como, por exemplo, com uma bateria eletrônica.

Cada nota tocada pelo instrumento gera um comando que é enviado ao computador para que ele decida o que fazer. Geralmente ele apenas reproduz a nota tocada. A vantagem é que, pelo computador, é possível escolher qual será o timbre da nota. Isto significa que é possível reproduzir o som de um saxofone com um teclado musical, por exemplo, ou combinações mais exóticas, como tocar uma flauta com uma bateria eletrônica.

Na internet existem diversos sites[32] gratuitos que conseguem se conectar a instrumentos musicais ligados no computador para reproduzir os mais variados sons. Também é possível gravar as sequências de comandos enviados pelos instrumentos e, com isso, criar uma música. Para fazer esse tipo de coisa, geralmente se utiliza programas chamados DAW (*Digital Audio Workstation*). Os mais conhecidos são:

- Pro Tools[7]
- FL Studio[18]
- Audacity[5]
- GarageBand[1]

O que o software da LaserHarp se propõe a fazer é ser um instrumento MIDI virtual, ou seja, ele gerará comandos MIDI para que outros programas os execute. Estes comandos são de dois tipos: `noteon` e `noteoff`, um para tocar e outro para parar de tocar uma nota, respectivamente.

Ambos os comandos necessitam que sejam passados 3 parâmetros:

- **`note`**
  Um número de 0 a 127 representando uma nota musical. Por exemplo, o número 60 representa a nota C da oitava 4.
- **`velocity`**
  Um número de 0 a 127 representando a velocidade em que a nota é tocada, o que na pratica representa o volume dela. Quanto maior o número, maior o volume.
- **`channel`**
  Um número de 0 a 15 representanto em qual canal a nota deve ser reproduzida. Isso é útil para programas em que é possível designar cada canal para tocar um timbre diferente. Mas, no caso, todas as notas serão tocadas em um único canal (canal 0).

Conforme mencionado na Seção 4.2, o pacote escolhido para lidar com comandos MIDI foi o `easymidi`[21]. Um código de exemplo do uso desse pacote pode ser visto no Programa 4.2.

Programa 4.2 Comando MIDI para tocar a nota C da oitava 4 com volume máximo.

```
1  import easymidi from "easymidi"
2  let output = new easymidi.Output("LASERHARP", true)
3  output.send("noteon", {
4     note: 60,
5     velocity: 127,
6     channel: 0,
7  })
```

4.6.1 Funções

No software da LaserHarp, os códigos que envolvem o protocolo MIDI estão no arquivo `src/components/Midi.js`. Nele temos a classe `Midi` com as seguintes funções:

- **`connect()`**
  Cria uma conexão com uma porta MIDI virtual. Aqui é importante notar que a função se comporta de maneira diferente de acordo com o sistema operacional que estiver sendo usado no momento. No Windows, diferentemente dos demais sistemas, não é possível criar uma porta MIDI virtual nativamente. Para isso deve ser usado

um software externo capaz de criar essa porta. O software escolhido para cumprir tal função, como visto na Seção 4.3, chama-se `loopMIDI`[11].

- **`getVirtualPort(name)`**
  Procura por uma porta MIDI com o nome "`name`". É utilizada quando se está no sistema Windows. No caso, a função procura a porta criada pelo `loopMIDI`.
- **`isOpen()`**
  Verifica se a conexão com a porta virtual está ativa.
- **`play(note, intensity)`**
  Manda o comando `noteon` para a porta virtual com os parâmetros "`note`" e "`intensity`" (volume da nota).
- **`release(note, intensity)`**
  Manda o comando `noteoff` para a porta virtual com os parâmetros "`note`" e "`intensity`" (tais parâmetros são necessários para se saber qual nota deve ser interrompida).
- **`beep(note, intensity)`**
  Executa as funções `play` e `release` uma seguida da outra, causando um som curto (beep!). É útil para testes.

4.7 Câmera

Como visto na Seção 4.1, o software possui uma classe `Camera` que contém as funções relacionadas com o pacote `opencv4nodejs`[20]. Embora o programa esteja pronto para captar as imagens da câmera, nenhum processamento é feito sobre essas imagens até o momento da publicação deste projeto. Como o foco do trabalho foi dado à construção da LaserHarp e deste software que a controla, a parte do código que identifica quais lasers foram tocados foi deixada para ser implementada futuramente.

Na interface gráfica (Figura 4.13) existem alguns parâmetros que podem ser alterados pelo usuário:

- **Dispositivo de captura** (`device_id`)
  - A câmera que será utilizada para captar as imagens.
  - Ao iniciar o programa, um vetor com todas as câmeras disponíveis é criado. A variável `device_id` indica o índice da câmera escolhida.
- **Brilho** (`brightness`)
  - Número real de -100 a 100 que indica a quantidade de brilho que deve ser adicionado à imagem da câmera (caso o valor seja negativo, o brilho é retirado).
- **Contraste** (`contrast`)
  - Número real de -100 a 100 que indica a quantidade de contraste que deve ser adicionado à imagem da câmera (caso o valor seja negativo, o contraste é retirado).

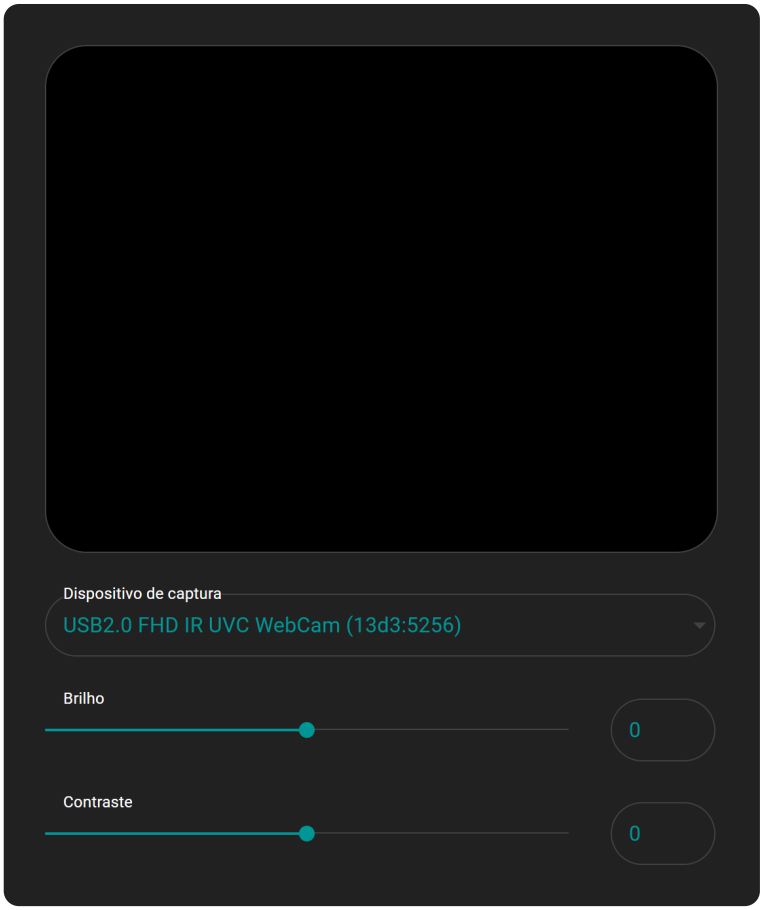


Figura 4.13: *Área de controle da câmera.*

Para se processar as imagens, basta manipular, dentro da função `processVideo()`, a variável `frame`, que é uma matriz representando um quadro capturado pela câmera escolhida (ver Programa 4.3). Ao se identificar que um laser está sendo interrompido, o programa deverá executar a funçao `play()` da classe `Midi` (ver Seção 4.6). Analogamente, ao se identificar que um laser deixou de ser interrompido, a funçao `release()` deverá ser executada. Em ambas as funções, deve-se passar como parâmetro a nota em questão.

Programa 4.3 Função `processVideo()` da classe `Camera`.

```
1  processVideo = (capture) => () => {
2    let begin = Date.now()
3
4    let frame = capture.read()
5
6    this.renderFrame(frame, document.getElementById("camera"))
7    let delay = 33 - (Date.now() - begin)
8    setTimeout(this.processVideo(capture), delay)
9  }
```

4.8 Comunicação serial

Um dos objetivos do software em questão é controlar a LaserHarp. Para isso, com a ajuda do pacote `serialport`[25], foi usado um tipo de comunicação chamado serial[3]. Ela consiste na transferência de dados um `bit` de cada vez. A comunicação se dá fisicamente através de um cabo USB (*Universal Serial Bus*) (Ver Subseção 3.4.1) conectado ao Arduino da LaserHarp e ao computador. Por meio dela, é possível enviar e receber *strings*.

A Figura 4.14 exemplifica resumidamente o fluxo de dados que ocorre entre a harpa e o computador quando as configurações escolhidas para a harpa são as mesmas mostradas na Figura 4.16.

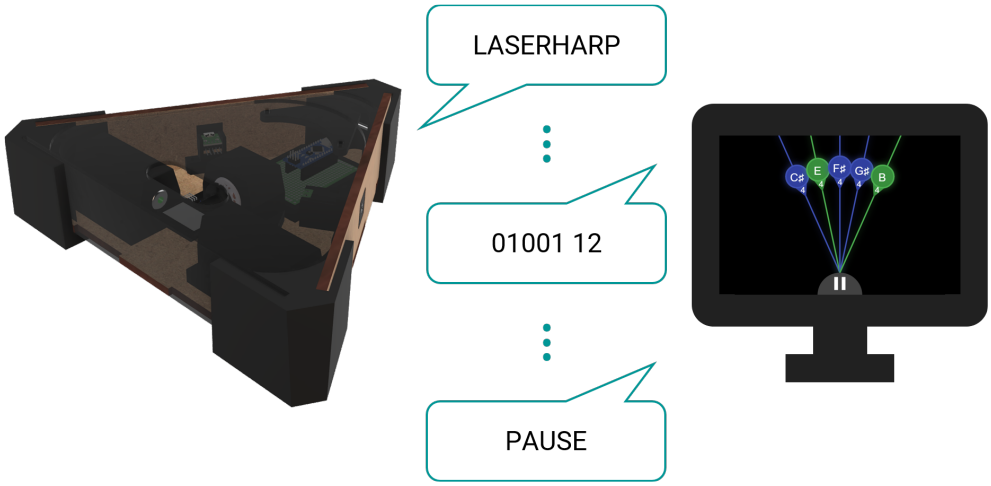


Figura 4.14: *Comunicação serial entre a LaserHarp e o computador.*

- Ao iniciar, o programa tenta se conectar a todos os dispositivos USB conectados ao computador até que algum deles, a LaserHarp, envie a mensagem "**LASERHARP**".
  - Caso nenhum dispositivo envie a mensagem em 5 segundos, todas as conexões são finalizadas e o botão "`Procurar LaserHarp`" (Figura 4.15a) fica disponível na interface para que o programa procure novamente.
  - Caso encontre a LaserHarp, a conexão é então estabelecida e surge na tela o desenho da harpa.
- Ao pressionar o botão "`Iniciar`" (Figura 4.15b), uma *string* contendo 2 números separados por espaço é enviada para a LaserHarp. Na Figura 4.14 temos um exemplo dessa *string*: "`01001 12`". Este é o **comando de inicialização** da LaserHarp.
  - O primeiro número é uma codificação binária referente aos lasers que a harpa deve criar.
    - 1: Laser verde.
    - 0: Laser azul.
  - O segundo número é o valor do ângulo que deve ter entre os lasers da harpa.
  - Após pressionar o botão, os *inputs* ficam desabilitados pois não é possível alterar os valores da harpa enquanto ela estiver gerando os lasers.

- Ao pressionar o botão "`Pausar`" (Figura 4.15c), uma *string* "*PAUSE*" é enviada para a LaserHarp, fazendo com que ela pare de gerar os lasers. Este é o **comando de pausa** da LaserHarp.
  - Após pressionar o botão, os *inputs* são habilitados novamente para que o usuário possa escolher outras configurações caso queira.
- Com a harpa parada, é possível desconectá-la do computador pressionando o botão "`Desconectar`" (Figura 4.15d). No caso, nenhuma mensagem é enviada para a LaserHarp e nem recebida dela. Após isso, o programa volta ao estado em que o botão "`Procurar LaserHarp`" é exibido.
- A qualquer momento o usuário pode desconectar a LaserHarp do computador retirando o cabo USB que liga os dois. Ao fazer isso, a harpa para imediatamente de gerar os lasers e o programa exibe novamente o botão "`Procurar LaserHarp`".

(a) *Botão "`Procurar LaserHarp`"*

(b) *Botão "`Iniciar`"*

(c) *Botão "`Pausar`"*

(d) *Botão "`Desconectar`"*

Figura 4.15: *Botões do programa.*

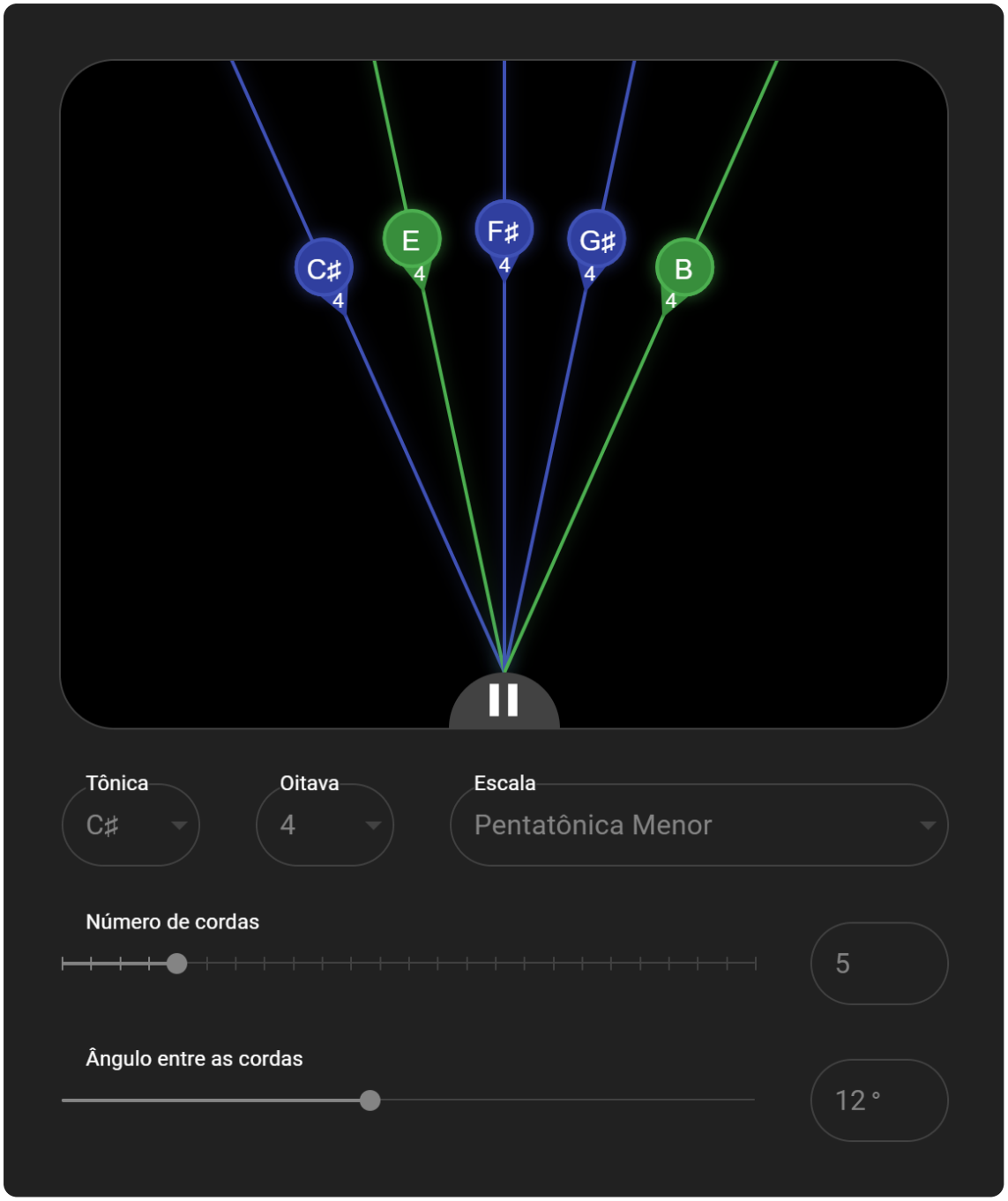


Figura 4.16: *Exemplo de configuração da LaserHarp.*

4.8.1 Funções

No código do software, todas as funções referentes à comunicação serial estão na classe `Hardware` do arquivo "`src/components/Hardware.js`". As funções são as seguintes:

- **`setSerial(serial)`**
  Define a porta serial que será utilizada.
- **`isActive()`**
  Verifica se a conexão com a porta serial está ativa.
- **`isConnected()`**
  Verifica se a porta serial atual não é nula.
- **`disconnect()`**
  Desconecta-se da porta serial.
- **`start(params)`**
  Envia o comando de inicialização para a LaserHarp.
- **`pause()`**
  Envia o comando de pausa para a LaserHarp.

Um código de exemplo do uso do pacote `serialport` pode ser visto no Programa 4.4.

Programa 4.4 Envia a *string* "*01001 12*" para a porta serial de nome "*COM9*".

```
1  import SerialPort from "serialport"
2  let serial = new SerialPort("COM9", { baudRate: 9600 })
3  serial.write("01001 12")
```

Capítulo 5

Conclusões

Ao longo deste trabalho, foi desenvolvido todo o hardware da LaserHarp, juntamente com sua estrutura física modelada em 3D e o software para computador responsável pelo controle da harpa.

Alguns problemas durante o desenvolvimento do hardware acabaram fazendo a LaserHarp não ser capaz de cumprir seu objetivo principal: tocar música. Mas o trabalho foi importante para mostrar como é possível, com um orçamento baixo[1], construir equipamentos complexos como uma harpa laser. A LaserHarp foi desenvolvida até um certo ponto em que, após os problemas serem solucionados, bastará trabalhar na área de processamento de imagem para se ter um instrumento pronto para o uso.

A Figura 5.1 ilustra o funcionamento geral da LaserHarp. Até o momento, só faltando desenvolver a seta vermelha de número 2 para que se tenha uma harpa funcional.

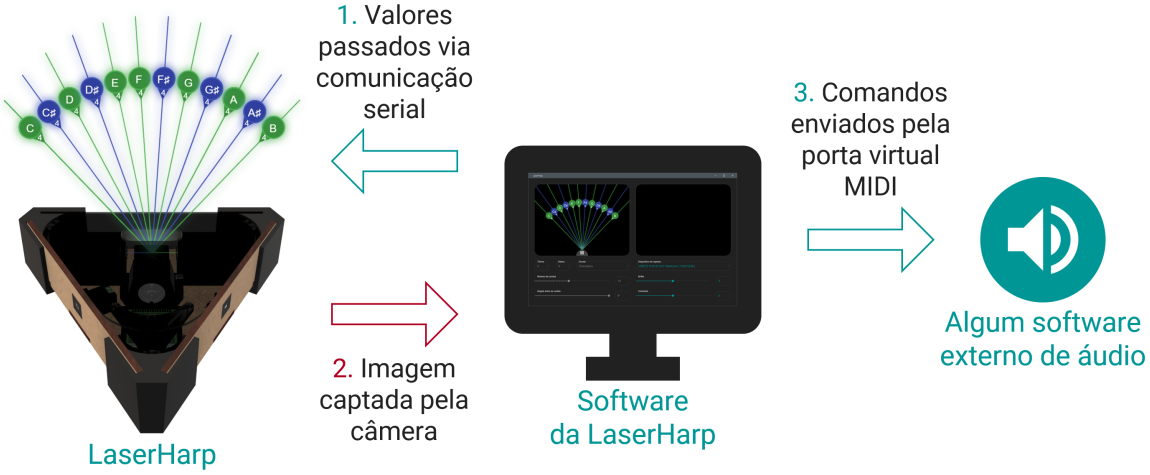


Figura 5.1: *Esquema do funcionamento geral da LaserHarp.*

[1]Em comparação com o preço de uma harpa laser profissional.

5.1 Resultados

O resultado da construção da LaserHarp, partindo do zero, pode ser visto na Figura 5.2.

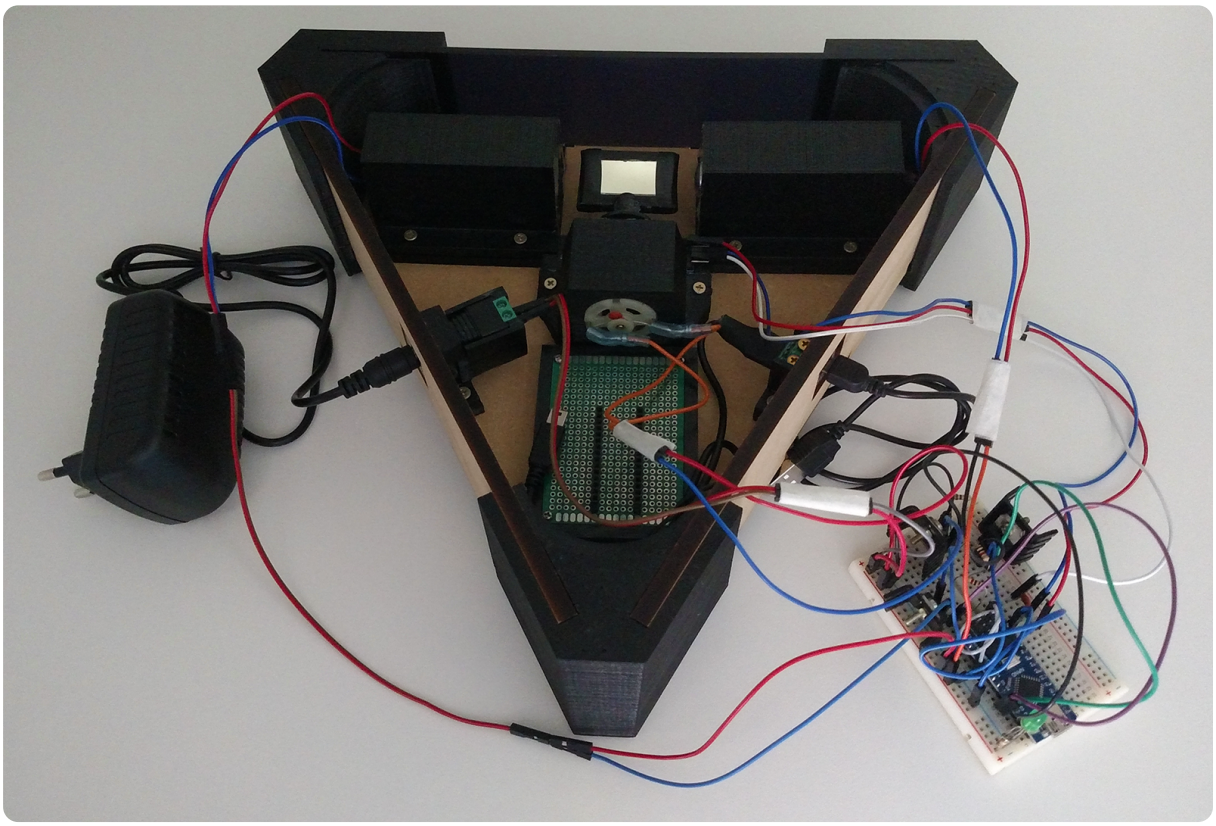


Figura 5.2: *LaserHarp completa vista de cima.*

Todo o circuito da LaserHarp está montado em uma protoboard (Figura 5.3) pois ainda existe a possibilidade de que o hardware seja alterado para solucionar o problema do brilho dos lasers (Subseção 3.4.2).

Alguns detalhes podem ser vistos a seguir:

- A Figura 5.4 mostra a parte de trás do motor. Repare no capacitor 104 de cor laranja ligado aos seus terminais.
- A Figura 5.5 mostra o conector P4 com o seu suporte fixo na estrutura da LaserHarp e com uma fonte conectada a ele.
- As Figuras 5.6 e 5.7 mostram o módulo micro USB com o seu suporte fixo na estrutura da LaserHarp, com o cabo mostrado na Subseção 3.4.1 de um lado e um cabo de celular de outro.

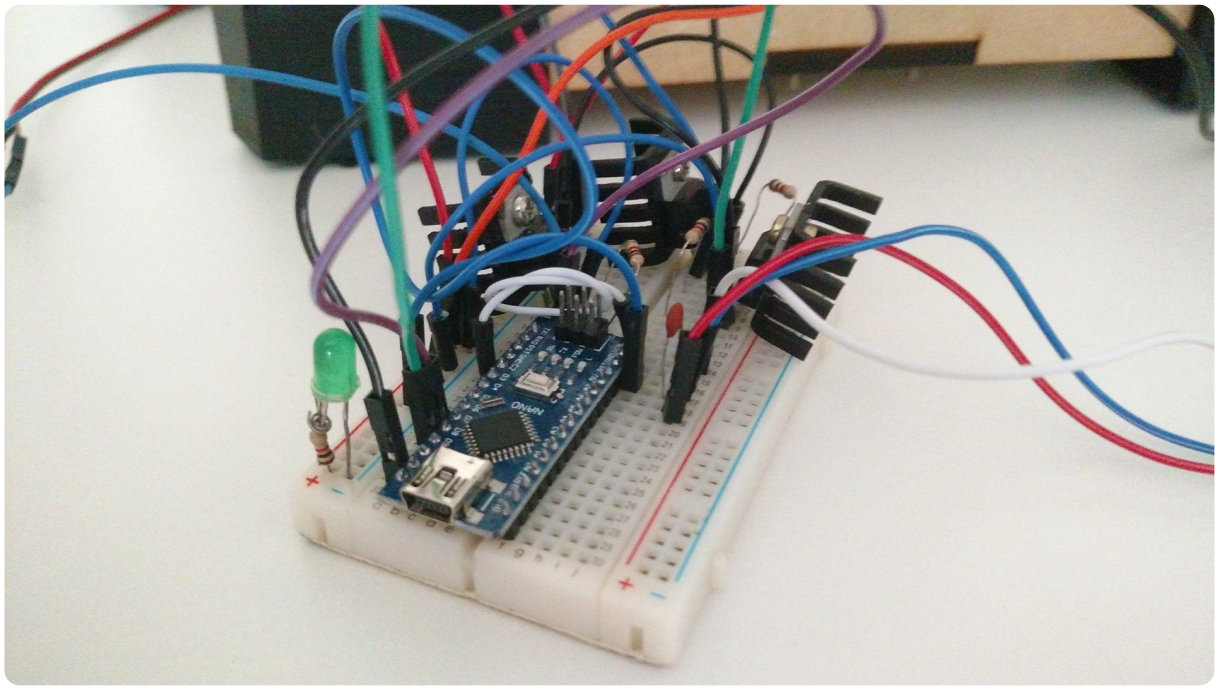


Figura 5.3: *Circuito da LaserHarp montado em uma protoboard.*

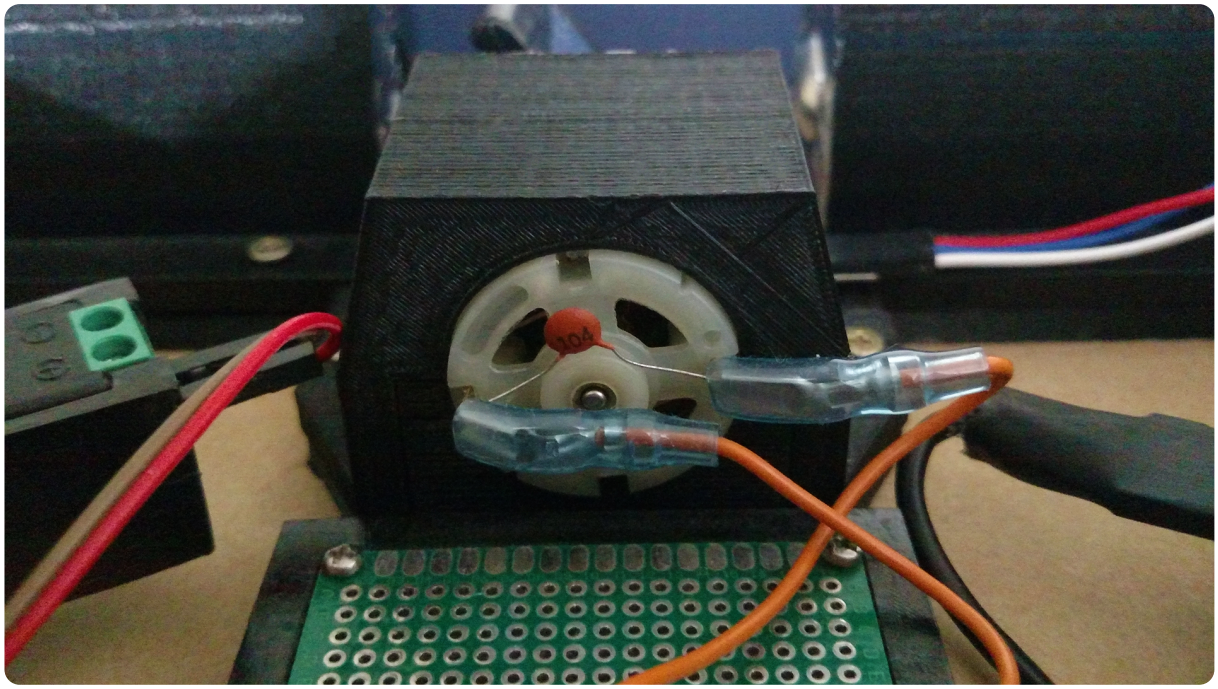


Figura 5.4: *Parte de trás do motor.*

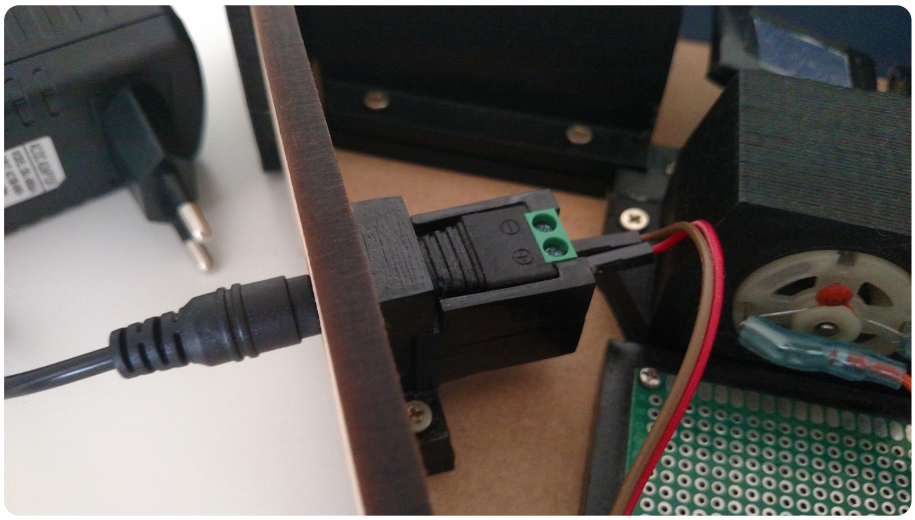


Figura 5.5: *Conector P4 fixo na LaserHarp.*

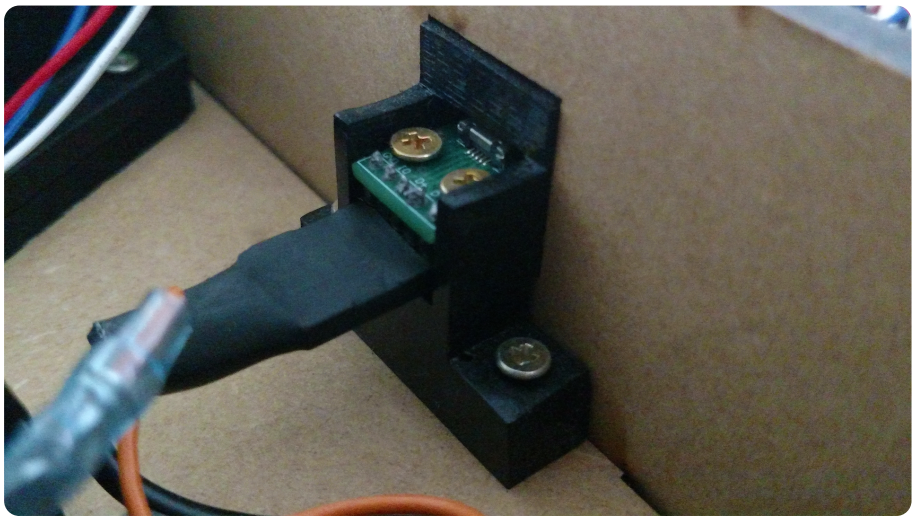


Figura 5.6: *Suporte com o módulo micro USB visto de dentro da LaserHarp.*

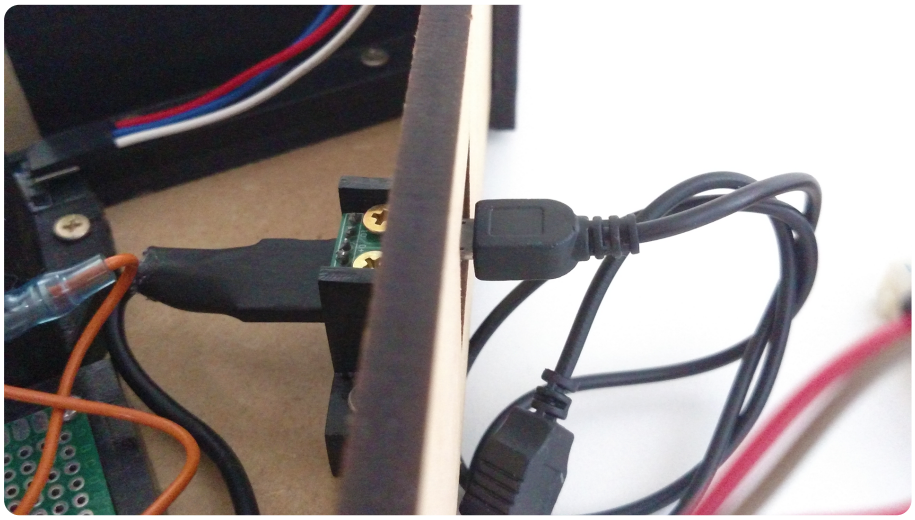


Figura 5.7: *Suporte com o módulo micro USB visto de cima.*

5.1.1 Problemas

Como visto na Subseção 3.4.2, o problema do brilho dos laser, que demora tempo demais para atingir seu máximo, impediu que o desenvolvimento da LaserHarp avançasse mais.

Na Figura 5.9 é possível ver as cordas sendo geradas pela harpa de acordo com as configurações mostradas na Figura 4.16. Como está evidente na foto, apesar de a harpa conseguir gerar os lasers nas posições corretas e no tempo exato, o fato de a luz emitida pelo laser ser fraca acaba deixando toda a imagem escura. A olho nu, com a ajuda de uma máquina de fazer fumaça, é possível ver todas as cordas sem problemas, mas o que importa para o desenvolvimento é a visão da câmera.

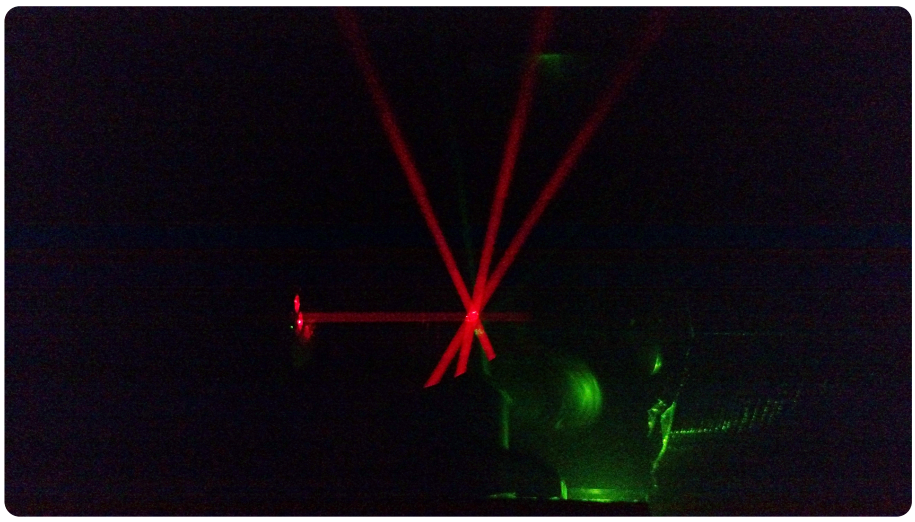


Figura 5.8: *Cordas geradas pela LaserHarp.*

Para efeito de comparação, as Figuras 5.9a e 5.9b mostram como deveriam ser os brilhos dos lasers verde e vermelho[2], respectivamente. Repare como a imagem toda fica clara por causa da luz dos lasers. Para que essas fotos fossem tiradas, os lasers foram deixados ligados sem piscar e o espelho foi mantido parado.

Todas as fotos dos lasers foram tiradas em um ambiente repleto de fumaça para que seus feixes ficassem visíveis.

[2]De última hora a caneta laser azul acabou queimando, por conta disso ela teve que ser trocada por uma de cor vermelha.

(a) *Feixe de laser verde.* **(b)** *Feixe de laser vermelho*

Figura 5.9: *Lasers.*

5.2 Futuro

Além de solucionar o problema exposto na Subseção 5.1.1, há algumas outras melhorias que poderiam ser feitas na LaserHarp futuramente:

- Soldar todos os componentes na placa fenolite (Figura 3.12).
  - Assim será possível fechar a estrutura com a tampa sem deixar nenhum fio para fora.
- Dar um acabamento melhor na harpa.
  - Ficou faltando lixar as peças impressas em 3D para deixá-las mais lisas e pintar tudo com alguma tinta de cor metálica. A base e as paredes laterais da estrutura poderiam ser pintadas de preto brilhante.
- Processamento de Imagem
  - Após solucionar o problema do brilho, a harpa estaria pronta para que fosse possível iniciar o desenvolvimento do processamento de imagem com OpenCV[26].
  - Cada feixe de laser interrompido identificado pela câmera faria tocar uma nota musical. A altura da interrupção do feixe poderia indicar o volume da nota.

5.3 Repositório

Todos os códigos do hardware e do software da LaserHarp, assim como os arquivos STL dos modelos 3D de suas peças e os arquivos DXF para corte laser estão disponíveis no repositório do projeto no GitHub (https://github.com/RaphaelRGusmao/LaserHarp).

Referências

[1] Apple. *GarageBand for Mac*. URL: https://www.apple.com/mac/garageband/ (acesso em 20/01/2020) (citado na pg. 52).

[2] Arduino. *Getting Started with the Arduino Nano*. 2019. URL: https://www.arduino.cc/en/Guide/ArduinoNano (acesso em 20/01/2020) (citado na pg. 18).

[3] Arduino. *Serial*. URL: https://www.arduino.cc/reference/en/language/functions/communication/serial/ (acesso em 20/01/2020) (citado nas pgs. 31, 42, 44, 55).

[4] Asic. *Motor control driver chip*. URL: https://www.elecrow.com/download/datasheet-l9110.pdf (citado na pg. 36).

[5] Audacity. *Free, open source, cross-platform audio software*. URL: https://www.audacityteam.org/ (acesso em 20/01/2020) (citado na pg. 52).

[6] Autodesk. *Fusion 360*. URL: https://www.autodesk.com/products/fusion-360/ (acesso em 20/01/2020) (citado na pg. 3).

[7] Avid. *Pro Tools*. URL: https://www.avid.com/pro-tools (acesso em 20/01/2020) (citado na pg. 52).

[8] Babel. *What is Babel?* URL: https://babeljs.io/docs/en/index.html (acesso em 20/01/2020) (citado na pg. 44).

[9] CorelDRAW. *What is a DXF File?* URL: https://www.coreldraw.com/en/pages/dxf-file/ (acesso em 20/01/2020) (citado na pg. 3).

[10] Eckstein Komponente. *1.8° 42mm High Torque Hybrid Stepper Motor*. URL: https://ecksteinimg.de/Datasheet/Schrittmotor/JK42HS40-1004A/JK42HS40-1004AC.pdf (citado na pg. 34).

[11] Tobias Erichsen. *loopMIDI*. URL: http://www.tobias-erichsen.de/software/loopmidi.html (acesso em 20/01/2020) (citado nas pgs. 45, 53).

[12] Facebook. *React: A JavaScript library for building user interfaces*. URL: https://reactjs.org/ (acesso em 20/01/2020) (citado nas pgs. 40, 44).

[13] Fairchild Semiconductor. *LM2903/LM2903I, LM393/LM393A, LM293/LM293A Dual Differential Comparator*. 2001. URL: http://pdf.datasheetcatalog.com/datasheet/fairchild/LM393.pdf (citado na pg. 20).

[14] Fairchild Semiconductor. *TIP120/121/122*. 2001. URL: http://pdf.datasheetcatalog.com/datasheet/fairchild/TIP120.pdf (citado na pg. 21).

[15] Fritzing. *Home*. URL: https://fritzing.org/ (citado na pg. 26).

[16] GitHub. *Electron: Build cross platform desktop apps with JavaScript, HTML, and CSS*. URL: https://electronjs.org/ (acesso em 20/01/2020) (citado nas pgs. 39, 43, 44).

[17] Google. *Create intuitive and beautiful products with Material Design*. URL: https://material.io/design/ (acesso em 20/01/2020) (citado nas pgs. 44, 50).

[18] Image Line. *FL STUDIO*. URL: https://www.image-line.com/flstudio/ (acesso em 20/01/2020) (citado na pg. 52).

[19] Jean Michel Jarre. *Youtube Channel*. URL: https://www.youtube.com/user/jeanmicheljarre/videos (acesso em 20/01/2020) (citado na pg. 2).

[20] justadudewhohacks. *opencv4nodejs*. URL: https://github.com/justadudewhohacks/opencv4nodejs (acesso em 20/01/2020) (citado nas pgs. 44, 53).

[21] David de Kleer. *EasyMIDI*. 2017. URL: https://easymidi.readthedocs.io/en/latest/ (acesso em 20/01/2020) (citado nas pgs. 44, 52).

[22] KonvaJS. *What's Konva?* URL: https://konvajs.org/docs/index.html (acesso em 20/01/2020) (citado nas pgs. 44, 50).

[23] Material-UI. *React components for faster and easier web development*. URL: https://material-ui.com/ (acesso em 20/01/2020) (citado nas pgs. 44, 50).

[24] Mozilla. *JavaScript*. 2019. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (acesso em 20/01/2020) (citado na pg. 39).

[25] Node SerialPort. *Node.js package to access serial ports for Linux, OSX and Windows*. URL: https://serialport.io/ (acesso em 20/01/2020) (citado nas pgs. 44, 55).

[26] OpenCV. *About*. URL: https://opencv.org/about/ (acesso em 20/01/2020) (citado nas pgs. 44, 45, 64).

[27] OpenJS Foundation. *NodeJS*. URL: https://nodejs.org/ (acesso em 20/01/2020) (citado na pg. 44).

[28] Prolight. *Laser Harp Controller*. URL: https://laser-harp.com/ (acesso em 20/01/2020) (citado na pg. 1).

[29] Stratasys. *GrabCAD library*. URL: https://grabcad.com/library (acesso em 20/01/2020) (citado na pg. 3).

[30] Texas Instruments. *DRV8825 Stepper Motor Controller IC*. 2014. URL: http://www.ti.com/lit/ds/symlink/drv8825.pdf (citado na pg. 34).

[31] w3schools. *JSON - Introduction*. URL: https://www.w3schools.com/js/js_json_intro.asp (acesso em 20/01/2020) (citado na pg. 43).

[32] Web Synths. *The browser-based microtonal midi instrument*. URL: https://websynths.com/ (acesso em 20/01/2020) (citado na pg. 51).

[33] Yarn. *Getting Started*. URL: https://yarnpkg.com/en/docs/getting-started (acesso em 20/01/2020) (citado na pg. 44).