CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DA PRAIA GRANDE
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE
E DESENVOLVIMENTO DE SISTEMAS

RELATÓRIO TÉCNICO DE CONCLUSÃO DE CURSO

Utilização do Arduino para simular o trabalho das engrenagens no torno mecânico

DANILO PELOZONE LIMA
RODRIGO AUGUSTO A. DA SILVA

Orientador: Prof. Me. Fernando Bacic Mendes
Co orientador: Prof. Me. Jonatas Cerqueira Dias

PRAIA GRANDE

DEZEMBRO DE 2018
DANILO PELOZONE LIMA
RODRIGO AUGUSTO A. DA SILVA

Utilização do Arduino para simular o trabalho das engrenagens no torno mecânico.

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Praia Grande, como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Fernando Bacic Mendes
Co orientador: Prof. Me. Jonatas Cerqueira Dias

PRAIA GRANDE
DEZEMBRO DE 2018

DANILO PELOZONE LIMA
RODRIGO AUGUSTO A. DA SILVA

Utilização do Arduino para simular o trabalho das engrenagens no torno mecânico.

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia de Praia Grande, como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Praia Grande, _____de _____ de _____.

Banca de Avaliadores.

_____

Prof. Me. Fernando Bacic Mendes

Orientador

_____

Prof. _____

Avaliador

_____

Prof. _____

Avaliador

AGRADECIMENTOS

Agradeço a Deus por fazer ser possível e pela ajuda. Meus pais pela confiança, paciência e incentivo.

Gostaríamos de agradecer a todos os professores, colegas e orientadores que estiveram presentes durante todo o processo de aprendizado nos últimos 3 anos. Sempre dando apoio e motivando a buscar melhorar e apresentar o melhor resultado possível.

Em especial gostaríamos de agradecer ao professor Fernando Mendes Bacic. Sem ele o desenvolvimento desse relatório não teria sido possível, ele sempre foi um professor presente em nosso desenvolvimento não só como profissional, mas como pessoa, guiando e apoiando no processo de amadurecimento.

Também gostaríamos de agradecer o professor Jonatas Cerqueira Dias e a professora Renata Neves, que sempre se dispuseram para nos ajudar e guiar com criticismo construtivo e ensinamentos importantes.

OS AUTORES

RESUMO

A engenharia evoluiu bastante ultimamente com auxílio da informática e uma das áreas da engenharia que mais se valeu dessa junção foi a automação. Esse projeto visa propor o uso do Arduino que é uma ferramenta *open-source* de prototipagem para criação de uma estrutura de mecatrônica, de baixo custo, para a realização de rotinas de engrenagem para criação de roscas em tornos mecânicos. O protótipo montado usando Arduino uno e um motor de passo permite a usinagem roscas que anteriormente não seriam possíveis em certos modelos de torno, possibilitando a usuários com baixa renda ou com interesse de economizar reutilizar e atualizar tornos já existentes. Atualmente alguns modelos de tornos existentes não possuem a possibilidade de usinagem de todos os tipos de roscas, a melhoria e automatização de processos é algo que é constantemente buscado no mercado. Tornos mecânicos automatizam o processo de usinagem para criação de roscas e parafusos, o custo da máquina é alto e traz um grande benefício para quem trabalha na área de construção, marcenaria, prototipagem e mecatrônica. Quando um torno deixa de se tornar ideal com o maior arranjo de possibilidades para usinagem as únicas escolhas no mercado disponíveis são tornos mais modernos de alto custo ou impressoras 3D, ambas as opções possuem um alto custo de aquisição que pode não ser realista ao consumidor com baixo poder aquisitivo.

O projeto tem como principal objetivo uma solução de investimento reduzido usando peças de fácil obtenção no mercado, mostrando uma solução viável para a reutilização e atualização de tornos, não computadorizados, mas que não atendem as necessidades do torneiro.

Para o desenvolvimento da solução, como previamente mencionado, foi usada a placa Arduino de prototipagem, o código desenvolvido foi feito na linguagem C/C++. Foram documentados diagramas de acordo com a UML para a documentação do sistema, mostrando seus atores e casos de uso. Os testes foram implementados com foco tanto em requisitos funcionais e não funcionais, evitando fios expostos e design final do produto de maneira que ele seja compacto e não apresente riscos.

Os resultados apresentados foram que a utilização do Arduino com outros componentes como motores de passo é capaz de fazer o processo de usinagem. A infraestrutura foi capaz de criar o resultado desejado sem problemas, essa solução então mostra a possibilidade da introdução de uma solução barata a questão de atualização de tornos. Usuários com baixo poder aquisitivo, tanto que usam o torno por motivos pessoais como entusiastas, quanto os que usam para motivos profissionais como eletricistas ou marceneiros, teriam acesso a essa solução. O maior investimento da solução, é principalmente representado, pela peça mais cara que é o próprio Arduino.

Palavras-chaves: Arduino uno, Automação, motor de passo, mecatrônica**;**

ABSTRACT

Engineering evolved a lot in the last few years with the help of technology, and one of the areas of knowledge that got benefited the most from this was automation. This project proposes the use of Arduino which is an open-source prototyping tool for the creation of a low cost mechatronics structure for the reproduction of gear routines for the creation of threads in mechanical lathes. The prototype created by using Arduino uno and a stepper motor allows the machining of screws that previously couldn't be done by certain mechanical lathe models, making it possible for users with low-income and with interest in saving money to re utilize and update their old lathe models. Currently, some existing lathe models don't allow for the machining of all kinds of screw threads, the improvement of automatization of processes is something that is always sought after in technology. Mechanical lathes make the machining process automatic for the creation of screw threads and bolts, a lathe is expensive and brings a lot of value to those who work in the areas related to construction work, woodwork, prototyping and mechatronics. When the lathe stops being ideal for the ideal options of thread, the available options for consumers are either more modern lathes or 3D printers, both which are expensive for the average customer with low acquisitive power.

The project has the primary objective of being a small-budget solution that uses electronic components that are easily obtainable, showcasing a cheap solution for the re utilization and upgrade of lathes that no longer attends to the needs of its user.

For the development of the solution, as previously mentioned, it was used the Arduino prototyping board, the code was developed in the C/C++ language. The planning of the project was made using the SCRUM framework. It was used the UML model for documentation of the system and its components, showcasing actors and use cases. Tests were implemented with the focus on both the functional

requirements as well as the nonfunctional requirements of the system, bringing attention to the infrastructure of the project.

The results obtained showed that the use of Arduino in conjunction with other components like the stepper mother is capable of machining. The infrastructure was able to reproduce the expected result with no problems, this solution showcased a cheap option for upgrading lathes. Users with low acquisitive power that uses the lathe in small business or for personal projects like mechatronic enthusiasts or woodworkers, electricians would have access to it. The biggest investment that has to be made is the main component of the project, which is the Arduino board.

Keywords: Arduino uno, Stepper motor, Automatization, mechatronics**;**

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURA E SIGLAS

DIY - Do It Yourself (tradução: Faça Você Mesmo).

IDE - Integrated development environment (tradução: Ambiente de Desenvolvimento Integrado).

LCD - Liquid Crystal Display (Tradução: tela de cristal líquido).

SMD - Surface Mount Device (componente menor, soldado na superfície da placa).

PWM - Pulse Width Modulation.

CNC – Computer numerical control.

SUMÁRIO

1 INTRODUÇÃO

O torno é uma das ferramentas mais antigas criadas para modelagem de peças. Os primeiros criados foram tornos de vara na Idade Média. Conforme a tecnologia foi evoluindo os tornos sofreram melhorias. Os modelos de tornos atuais foram criados a partir da revolução industrial, quando James Watt criou a máquina a vapor e outros processos foram atualizados de acordo.

Ao longo do tempo, o meio de funcionamento dos tornos evoluiu e acompanhou os avanços tecnológicos. A produção de peças sempre teve um grande foco em automatização de processos, o torno originalmente era movido por um pedal, depois a vapor e agora eletricidade. Com a evolução das tecnologias com o passar dos anos alguns tornos mecânicos não são capazes de fazer a usinagem de peças variadas devido as suas limitações.

Neste relatório técnico será descrito desenvolvimento e funcionamento do “gear.simulator”, junção do Arduino, de um motor de passo e um encoder rotativo em tornos antigos para dar suporte a maior usinagem de rosca. Atualmente o torneiro precisa de várias combinações de engrenagens para realizar uma maior variedade de roscas. Os custos gerados pela aquisição de novo equipamento como impressoras 3Ds ou tornos mais modernos se mostram muito elevados, o que nem sempre é viável para torneiros menores com recursos escassos.

Grandes industrias podem simplesmente substituir seus tornos, mas usuários como marceneiros ou mecânicos nem sempre possuem a mesma facilidade para a substituição de equipamento. A infraestrutura do “gear.simulator” propõe uma solução mais acessível, financeiramente, na questão de melhoria de tornos.

As opções disponíveis no mercado, atualmente, são a substituição dos tornos antigos por tornos mais modernos ou impressoras 3D. Contudo certos modelos de impressoras 3D podem não satisfazer a necessidade que um torno supre além de ser cara, e a aquisição de um torno moderno muitas vezes apresenta um custo muito elevado para o usuário autônomo ou para uma empresa de pequeno porte. Remediar essa situação é o principal objetivo do “gear.simulator”.

1.1 OBJETIVO GERAL

Este projeto tem como proposta usar o Arduino para ser uma opção de automação no torneamento que é feito normalmente por engrenagens em tornos mais avançados, tornando possível atualizar tornos mais antigos para fazer roscas de acordo com os valores informados pelo usuário. Ele usa uma tela LCD e um menu simples com botões mecânicos para que o usuário entre com sua escolha, os dados usados no cálculo são introduzidos pelo próprio usuário por meio do teclado e então os motores recebendo as informações da placa de Arduino realiza a usinagem da peça.

O principal objetivo com isso é a criação de um sistema de baixo custo que torne possível a melhoria e automação de processos de tornos antigos. A melhoria de processos de mecanização é algo que se demonstra atrativo no mercado, e a criação dessa solução agrega um grande valor a torneiros para todo aquele que não possui um torno computadorizado ou impressoras 3D.

1.1.1 OBJETIVOS ESPECIFICOS

- Realizar o processo de usinagem com uma placa de prototipação.

A Adoção do Arduino se justifica, no intuito de oferecer novos movimentos de rotação e usinagem para tornos antigos, para a criação de uma rosca. Usa cálculos que são recebidos no Arduino por meio de uma interface para a que as peças ligadas possam fazer os movimentos calculados por meio do código em C++. O retorno esperado é uma rosca de acordo com o que foi entrado pelo usuário, podendo ser aplicado a tornos antigos para que os mesmos tenham uma área maior de atuação.

- Apresentar uma interface de fácil uso

É importante a infraestrutura apresentar uma interface de fácil uso e navegação para o usuário, levando em consideração que o mesmo muitas vezes é leigo em relação a tecnologia. Levando em consideração esse fator foi utilizado um menu simples e uma tela LCD que acompanham instruções de uso. Devido ser uma infraestrutura a facilidade na usabilidade do mesmo é de grande importância.

- Disponibilizar para usuários de baixa renda/poder aquisitivo uma solução viável

Atualmente não existem muitas opções para torneiros caso eles possuam um torno antigo além de comprar um torno mais recente ou uma impressora 3D. Ambas opções possuem um alto custo e muitas vezes não são viáveis a todos os usuários.

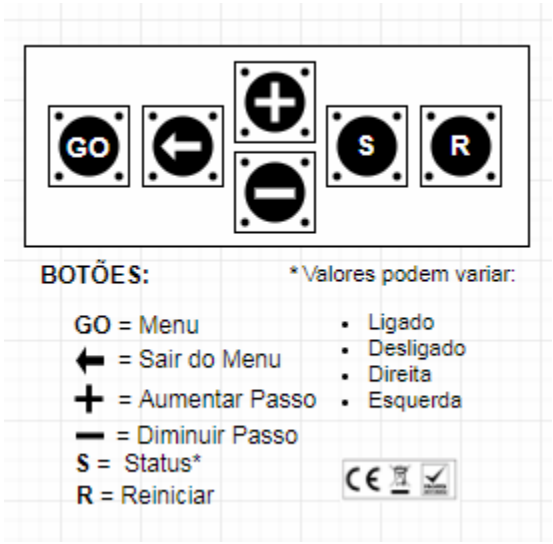
Impressoras 3D é uma tecnologia nova, que não é de fácil acesso a pessoas de baixa renda, o mesmo pode ser dito em relação a um torno automatizado moderno. Ambas soluções presentes não são viáveis para uma grande parcela do mercado, mas a usinagem é um processo útil para várias áreas de desenvolvimento. Muitos processos fabris que usam roscas e parafusos para seu trabalho e aqueles que desenvolvem soluções de uso pessoal podem se beneficiar.

O projeto tem como público alvo dono de tornos antigos com menos opções de usinagem, esse público alvo inclui tantos entusiastas quanto autônomos que possuem uma renda mais baixa. É de interesse a esse público pois um torno moderno é muito caro, e o projeto de baixo custo apresenta um grande custo benefício em pequena escala.

O projeto é feito em Arduino, a ferramenta fritzing e programado em C++ que foi versionado com a ferramenta do Git, foi usado SCRUM para o planejamento do projeto em que foram feitas várias entregas objetivas e detalhadas com uma fase de teste e implantação. As entregas foram planejadas num modelo Kanban com as tarefas e fases de projeto especificadas.

O sistema é um dispositivo que possui um teclado e uma tela de LCD que apresenta opções para manipulação de entrada do usuário do torno. Ao total de 4 opções de funções para escolha eles sendo a direção; passo desejado; rosca polegada; rosca métrica. O usuário tem controle dos menus por meio de 6 botões, entrada, saída, mais/aumenta, menos/diminui, Status/Set, reiniciar/parar emergência. Como descrito no colante (adesivo fixado no case).

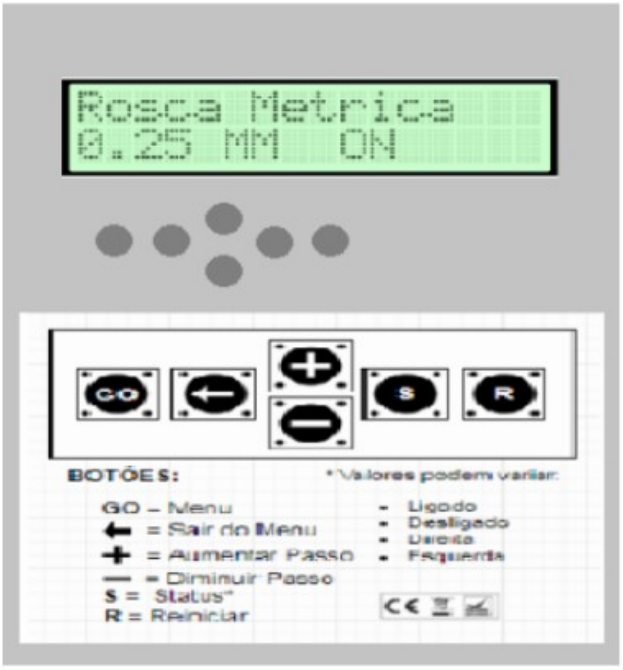
Figura 1 - Adesivo de descrição das funcionalidades dos botões



Fonte: (Autores, 2018)

O usuário inicia o Arduino, sendo a primeira tela de navegação de menus, o primeiro menu a aparecer é a Direção, a direção configurada por padrão é o movimento para a esquerda. Apertando o botão - ou + (como mostrado na imagem figura 1 - Adesivo de descrição das funcionalidades dos botões) há a troca dos menus.

Figura 2 - Protótipo de infraestrutura



Fonte: (Autores, 2018)

2 Planejamento

Para o planejamento do projeto foram levantados o ambiente em que ele foi feito, as linguagens usadas, os componentes eletrônicos, seu funcionamento e metodologia de documentação que é explicada a seguir.

2.1 SISTEMA OPERACIONAL

Sistema operacional usado Windows 7, por oferecer o suporte mais facilitado a maioria das ide de desenvolvimento da linguagem C/C++, facilidade de instalação e reconhecimento do driver FTDI para comunicação via serial com a plataforma Arduino. Além do Windows ser mais popular e intuitivo para novos usuários do Arduino, maior conteúdo de tutoriais, maior suporte de drivers e programas para desenvolvimento.

2.2 PLATAFORMA DE DESENVOLVIMENTO

A plataforma Arduino oferece uma IDE Arduino 1.8.4 para o desenvolvimento e compilação, a compilação é a tradução da linguagem alto nível compreensível ao ser humano para linguagem máquina especificada pela arquitetura do processador, traduzindo o código fonte para código de máquina, esse código é feito de 0 e 1, chamado de binário. O código já traduzido, resultando no binário é passado para o microcontrolador atmega328p presente no Arduino uno via USB, usando o protocolo serial.

2.3 LINGUAGEM C/C++

Segundo Schildt (1996) a linguagem C é estruturada enquanto C++ é orientada a objetos, ambas são compiladas. O C++ é versão estendida e melhorada do C, ela acopla o C mais extensões para orientação a objetos. é possível programar para muitas plataformas computacionais e embarcados. Na programação para o Arduino pode se utilizar o C ANSI ou C++, para facilitar a implementação do código em vários modelos de Arduino e não precisar de conhecimento técnico foi usado a biblioteca LiquidCrystal que cria um objeto. Esse objeto criado na lógica algorítmica representa um LCD da vida real. O uso desta biblioteca cria um objeto, então a programação é feita em C++.

2.4 FRITZING

O fritzing é uma ferramenta de automação de design eletrônico, é uma inciativa open-source que é utilizada para criação de protótipos, documentação de modelos de hardware. O mesmo foi utilizado no projeto para fins de documentação e diagramação de componentes em um nível eletrônico.

Para a prototipagem dos componentes, visualização das ligações por fio, foi usado o fritzing na versão 0.9.3b para windows 64 bits, sendo disponível para os sistemas operacionais linux e macOS e não necessita de instalação. O fritzing é de uso gratuito.

2.5 COMPONENTES ELETRONICOS

Os componentes eletrônicos utilizados são especificados a seguir, explicando a maneira com qual ele é relevante ao projeto e onde foi utilizado.

2.5.1 ENCODER ROTATIVO INCREMENTAL - 600 pulsos por giro

Figura 3 - Encoder Rotativo Incremental

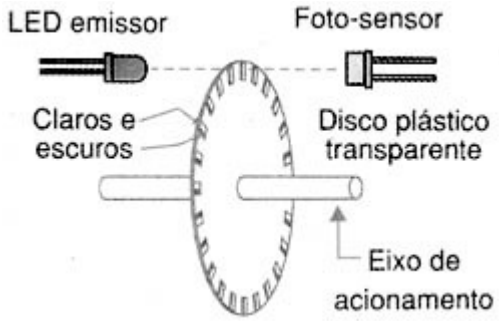
Fonte: (MP Automação, 2017)

Converte a rotação do eixo com pulsos elétricos, encoders são geralmente usados para monitorar eletronicamente a posição da rotação do eixo. Será usado para monitoramento da rotação e posição do eixo árvore do torno mecânico. (DYNAPAR)

Ele é construído usando um disco de material rígido, que há tiras que impedem a passagem da luz ao fotossensor, é exemplificado na Figura 4 o funcionamento de um sensor rotativo simples. O sensor rotativo incremental possui 2

fotosensores, que possibilita a identificação do sentido de rotação. Essa identificação é feita de forma lógica pelo algoritmo.

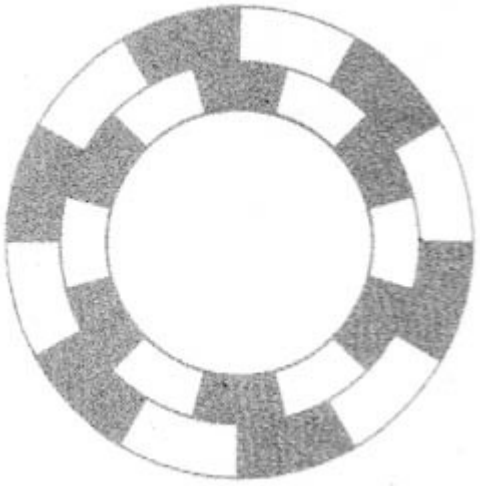
Figura 4 - Encoder óptico por dentro



Fonte: (NEWTON BRAGA, 2015)

A figura número 5 exemplifica a dessincronia das barreiras no disco, essa diferença possibilita o reconhecimento via algoritmo do sentido da rotação, pelos padrões de sentido (Sentido horário A0011 B1001) (Sentido Anti-Horário A1001 B0011).

Figura 5 - Disco encoder rotativo Incremental

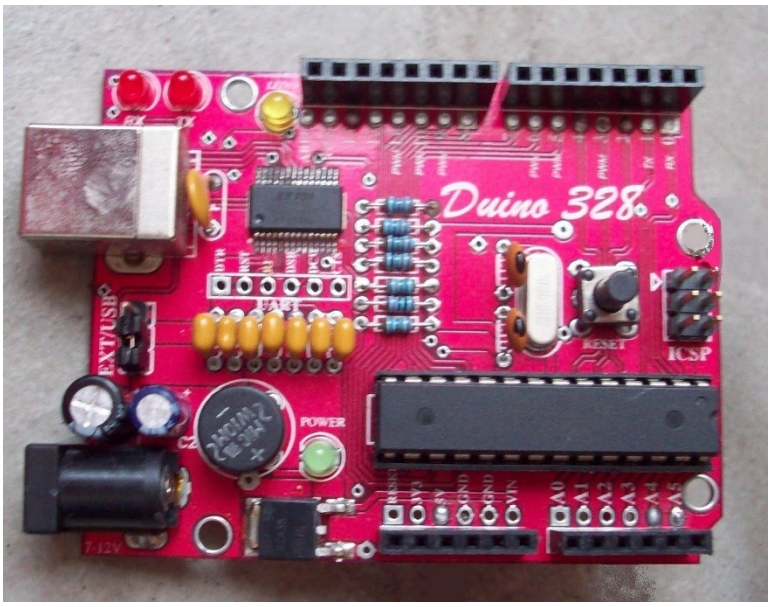


Fonte: (NEWTON BRAGA, 2015)

Usado com muita importância na indústria para automação e controle, é um sensor que identifica o ângulo exato, velocidade, sentido da rotação.

2.5.2 ARDUINO UNO Rev3 R3 Atmega328

Figura 6 - Arduino uno

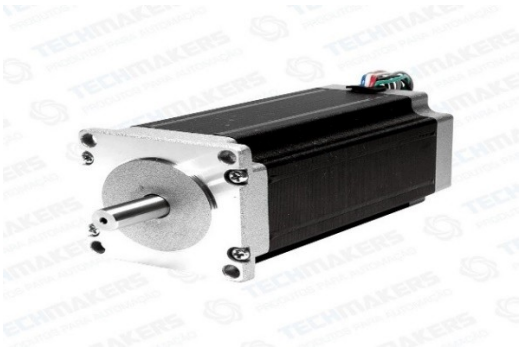


Fonte: (Autores, 2018)

O Arduino é uma placa controladora de prototipagem eletrônica de hardware e software livre para programação, com entradas e saídas de dados. Por meio dessa placa é possível conectar outros circuitos externos como sensores, LEDs, chaves, relés e pequenos motores. Pode ser programado em linguagens como C, C++, Java e outras. A placa contém 6 portas analógicas e 14 portas digitais, ambos tipos sendo de entrada e saída de pulsos. Contém 5,3.3 e voltagem de referência para alimentação de periféricos como sensores; há o atmega328p como controlador, havendo variações do Atmega328p, não influencia na programação. (arduino.cc)

2.5.3 MOTOR DE PASSO Nema 23 - 30kgf.cm

Figura 7 - Motor de Passo



Fonte: (POLICOMPCOMPONENTES, 2017)

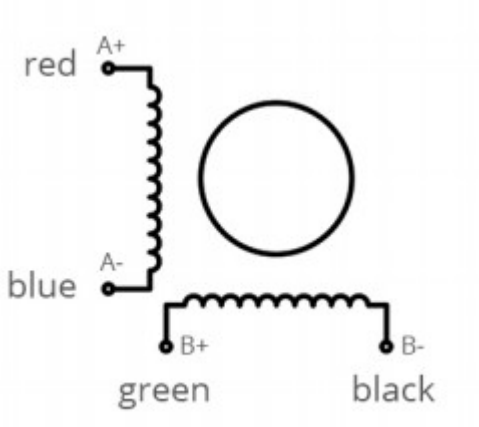
Motor de passo reproduz por meio de pulsos PWM (*pulse width modulation*) que controlam a velocidade do motor, giros com grande precisão com velocidade controlada. Os motores de passo tecnicamente giram o número de pulsos recebido, velocidade também sendo controlada pela frequência dos pulsos. O ponto negativo do motor de pulso é que sua força de giro (torque) diminui exponencialmente pela velocidade; quanto maior a velocidade menor é o torque (figura 8). O motor usado tem a resolução de 200 pulsos por rotação, é bipolar e tem 4 fios, a alimentação de energia e pulsos é feita pelo Driver específicos para cada motor. (KALATEC)

Figura 8 - Especificações Motor

Standard	NEMA 23
Step angle	1.8° ±5%
Current / Phase	3.0A
Voltage / Phase	3.9V
Phase No.	2
Resistance	1.3 ±10% Ω
Insulation resistance	100MOhm (500V DC)
Inductance	6.5 ±20%mH
Insulation class	B
Holding torque	30kgf.cm

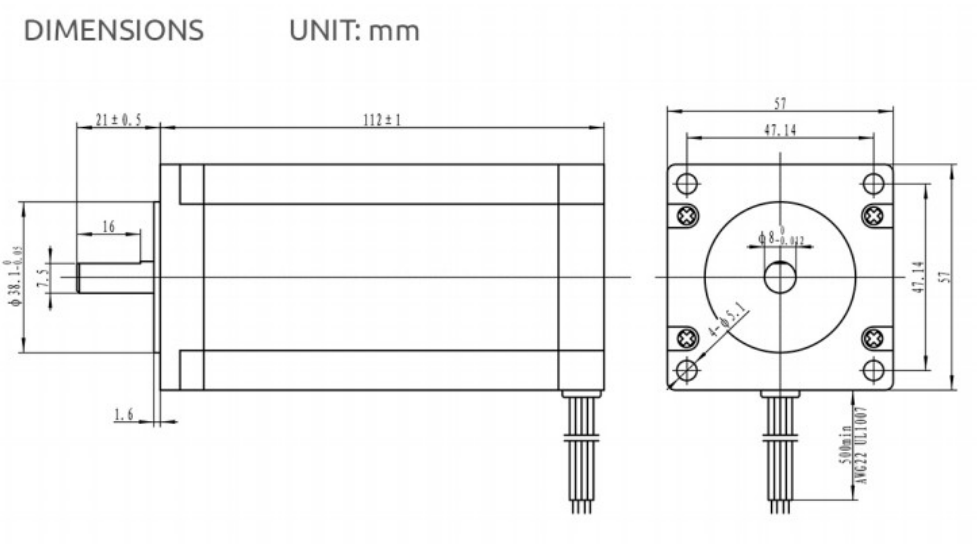
Fonte: (DATASHEET WS23-0300-30-4, 2015)

Figura 9 - Ligação bobinas



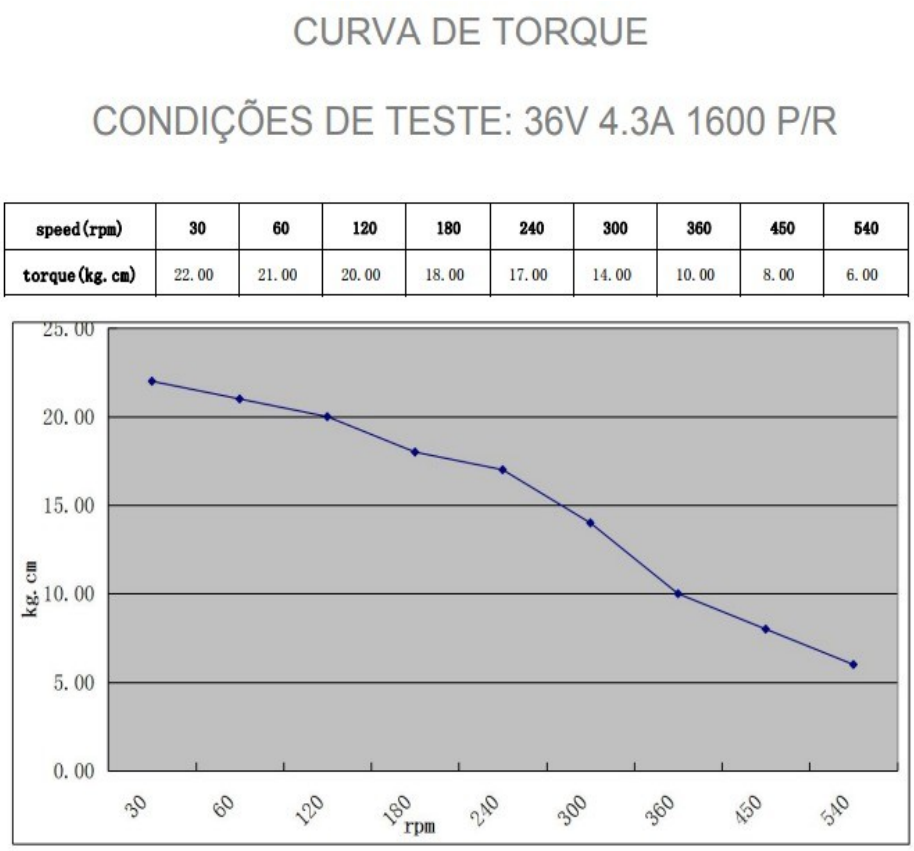
Fonte: (DATASHEET WS23-0300-30-4, 2015)

Figura 10 - Dimensões em milímetros



Fonte: (DATASHEET WS23-0300-30-4, 2015)

Figura 11 - Gráfico de torque nema 23 - 30kgf.cm



Fonte: (POLICOMPCOMPONENTES, 2018)

2.5.4 SHIELD DISPLAY LCD 1602 16x02 Keypad

Figura 12 - Shield LCD com botões

Fonte: (Arduinoecia, 2018)

O shield display lcd é uma placa que foi desenhada para ser acoplada no Arduino uno e dueminove, pode exibir textos, caracteres separados, caracteres especiais e desenhos, isso sendo separado por 16 colunas e duas linhas.

No algoritmo a leitura e identificação do botão é pela entrada em um teclado quando acionada o pino analógico A0 que é o resultado elétrico da resistência dos resistores que são ligados em série, e a cada ligação de resistor um botão é colocado. Quanto mais ligações de resistores, maior é a resistência, essa resistência resultante nessas ligações identifica o botão apertado.

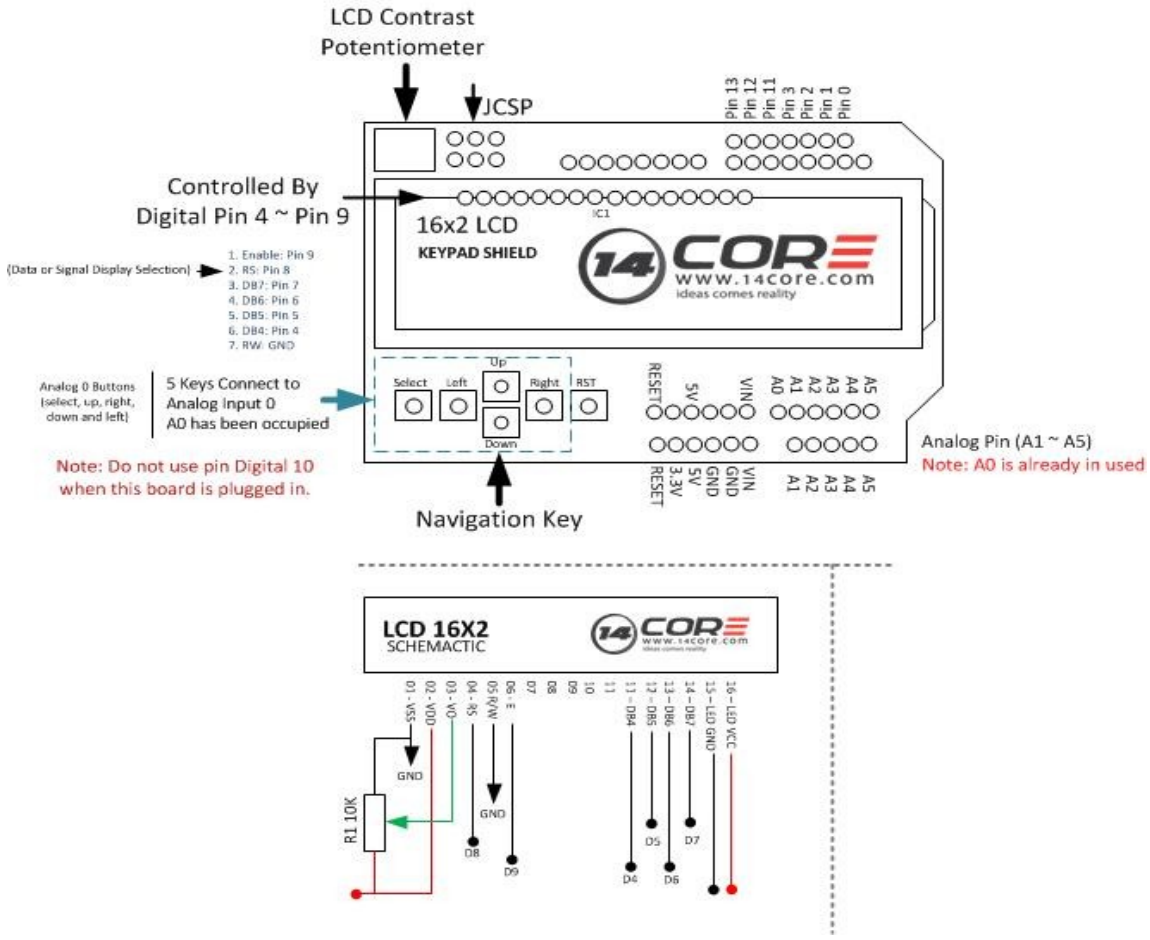
Os valores de resistência são: botão up de 142, down de 326, right de 0, left de 503, select de 741, esses valores podendo variar de shield para shield. Quando não há botão sendo apertado, não há resistência.

Na programação essa identificação pode ser feita usando IF, usando um valor mais próximo de maior valor, exemplo:

```
1. --if(AnalogRead(A0) < 200 ){
2. botão up apertado,
3. se 142 é menor que 200}
4. if else(AnalogRead(A0) < 800){
5. botão select(selecionar)
```

O botão up tem resistência de 142, é menor que 800, mas não é o mais próximo desse número, já o select tem resistência 741, é menor e mais próximo de 800 (DATASHEET LCD-KEYBOARD).

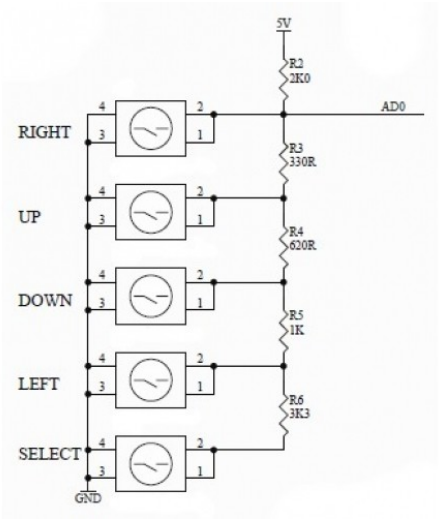
Figura 13 - Ligação pinagem Shield keyboard com Arduino uno



Fonte: (14CORE EDITOR, 2016)

Contém 6 botões ao total, sendo 5 configuráveis logicamente pelo valor resistivo que cada botão tem pela ligação de resistores; outro não configurável destinado ao reset do Arduino, como mostrado na Figura 13.

Figura 14 - Ligação elétrica



Fonte: (ARDUINO E CIA, 2018)

São presentes botões para escolha da opção em texto exibida na tela. Contém 16 colunas e 2 linhas; fundo cor azul, letras pretas. (ARDUINO E CIA).

No algoritmo a leitura e identificação do botão é pelo pino analógico A0 do display que é o resultado elétrico da resistência dos resistores que são ligados em série, e a cada ligação de resistor um botão é colocado.

Quanto mais ligações de resistores, maior é a resistência, essa resistência resultante nessas ligações identifica o botão pressionado. Resistência está presente no shield usado neste projeto, esses valores podendo variar de shield para shield. (DATASHEET LCD-KEYBOARD).

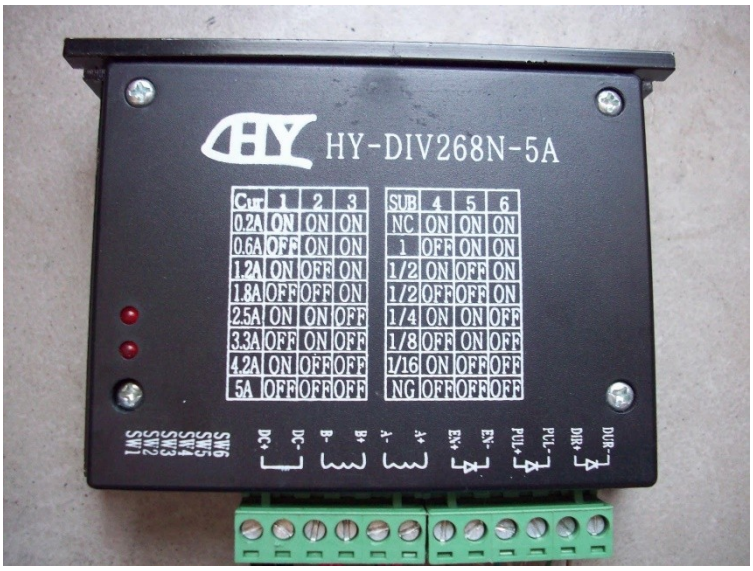
Tabela 1 - Valores de cada botão do shield LCD

Botão	UP	DOWN	RIGHT	LEFT	SELECT	NENHUM*
Valor	142	326	0	503	741	1000

Fonte: (Autores, 2018)

2.5.5 DRIVER MOTOR PASSO Hy-div268n-5A

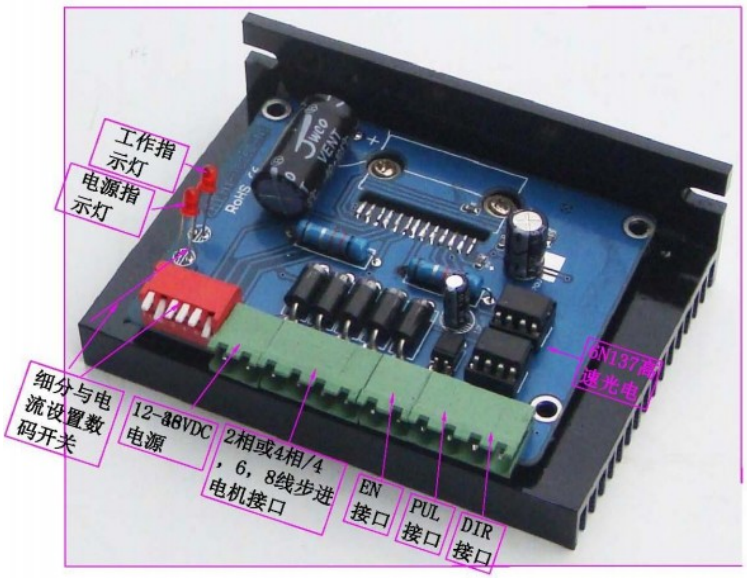
Figura 15 - HY-DIV268N-5A



Fonte: (Autores, 2018)

Hy-div268n-5a é baseado no CI Toshiba WD-6600 Driver para motor de passo é o intermediário entre controlador e o motor. Responsável por fornecer energia controlada e pulsos eletrônicos para o funcionamento programado do motor. É controlado pelo controlador central, no caso o Arduino Uno. (BAÚ DA ELETRÔNICA)

Figura 16 - Por dentro HY-DIV268N-5A



Fonte: (DATASHEET HY-DIV268N-5ª, 2016)

Figura 17 - Possíveis ligações dos minis interruptores do HY-DIV268N-5A

细分	1	2	3	电流	4	5	6
NG	ON	ON	ON	0.2A	ON	ON	ON
1	OFF	ON	ON	0.6A	OFF	ON	ON
1/2	ON	OFF	ON	1.2A	ON	OFF	ON
1/2	OFF	OFF	ON	1.8A	OFF	OFF	ON
1/4	ON	ON	OFF	2.5A	ON	ON	OFF
1/8	OFF	ON	OFF	3.3A	OFF	ON	OFF
1/16	ON	OFF	OFF	4.2A	ON	OFF	OFF
NG	OFF	OFF	OFF	5A	OFF	OFF	OFF

Fonte: (DATASHEET HY-DIV268N-5ª, 2016)

2.5.6 Funcionamento Técnico de cada componente

O funcionamento técnico das funcionalidades e como os componentes de hardware interagem entre si especificados e exemplificados seguem nos próximos tópicos.

2.5.6.1 Menu Passo Desejado

Na opção de PASSO DESEJADO é a escolha para desbaste da usinagem, não existe sincronização com o eixo árvore. A velocidade é controlada pelo número do delay dado para função Delay() para dar cada pulso ao Driver, esse delay é feito usando a fórmula 10/passo, o passo corresponde pelo passo desejado escolhido no menu PASSO DESEJADO. Um exemplo: Caso queira um passo de 1,0mm, 10/1,0 resulta em 10 milissegundos de delay; caso queira 0,1 mm resulta 100 milissegundos; caso queira 9,9mm, 10/9,9 resulta 1,0101 milissegundos delay.

2.5.6.2 Menu Rosca Milímetro

No menu ROSCA MILÍMETRO a rotação do motor de passo, é feita por pulsos de acordo com os intervalos recebidos pelo encoder, a cada movimento do eixo árvore o encoder envia pulsos para o Arduino, pulsos esses recebidos e contados para depois do número total ser alcançado, mandar um pulso para o driver que rotacionam o motor de passo. O encoder rotativo incremental divide uma volta completa em 600 ângulos, cada ângulo corresponde a um pulso elétrico, a cada volta completa são 600 pulsos enviados ao arduino. O motor de passo por padrão a 200 pulsos gira uma volta completa; o Driver HY-DIV268N-5A pode ser configurado para micro passos, dividindo os 200 passos por 16, nesta configuração para dar um giro completo precisa dar 200*16= 3200 pulsos.

Tabela 2 - Configuração de pulsos por rotação

Configurações Driver	Pulsos por giro
1 full step	200
1/2 half step	400
1/4 quarter step	800
1/8 eighth step	1600
1/16 sixteenth step	3200

Fonte: (Autores, 2018)

Isto depende de Driver e motor. Neste projeto é recomendado o full Step (passo completo) que é 200 pulsos por giro completo, pois essa configuração nos dá maior torque.

A. Fórmula rosca polegada

A formula usada para o cálculo no menu de rosca polegada usada é:

Intervalo = pulsos por giro completo encoder / (pulsos por giro completo motor passo / avanço completo do furo por giro *(25. 4 (medida de uma polegada em milimetros) / dentes por polegada)))).

B. Fórmula rosca Milímetro

A formula usada para o cálculo no menu de rosca métrica é:

Intervalo = pulsos por giro completo encoder / (pulsos por giro completo motor passo / avanço completo do furo por giro * rosca em milímetros))).

C. Cálculo feito pelo algoritmo em C ANSI

```
1.  {
2.  #define tam(vet) (sizeof((vet)) / sizeof(int))
3.  void main(){
4.  float rosc_mm[] = {0.25,0.50,0.85,1.0,1.25,1.50,1.75,2.0};
5.  int rosc_ii[] ={8,9,10,11,12,13,14,15,16,17,18,19,20};
6.  for(int i =0 ; i<tam(rosc_mm);i++){
7.  printf("Rosca Milimetro MM: %f \t Espacos: %f \n",rosc_mm[i],(1200/(200/(3.18*rosc_mm[i]))));
8.  }
9.  for(int i =0 ; i<tam(rosc_ii);i++){
10. printf("Rosca Polegada TPI: %d \t Espacos: %f \n",rosc_ii[i],(1200/(200/(3.18*(25.4/rosc_ii[i])))));
11. }
12. system("pause");
13. }
```

Tabela 3 - Roscas em milímetro mais comuns

Roscas Milímetros	**Espaços ()**
0.25	4.770000
0.50	9.540000
0.85	16.218000

1.00	19.080000
1.25	23.850000
1.50	28.620000
1.75	33.390000
2.00	38.160000

Fonte: (Autores, 2018)

Tabela 4 - Roscas fios por polegada (TPI)

Roscas Polegadas	**Espaços ()**
8	60.579000
9	53.848000
10	48.463200
11	44.057455
12	40.386000
13	37.279385
14	34.616571
15	32.308800
16	30.289500
17	28.507765
18	26.924000
19	25.506947
20	24.231600

Fonte: (Autores, 2018)

3 DESENVOLVIMENTO DO PROJETO

O sistema tem finalidade de atualizar tornos mecânicos antigos ou novos que não possuem o conjunto completo de engrenagem para fazer os mais amplos tipos de rosca.

Torneiros e entusiastas que usinam metais e plásticos no torno mecânico precisam de engrenagens para efetuar o desbaste e dependendo do serviço a rosca no material; esse trabalho é feito pelas engrenagens que são elementos de transmissão de movimento do eixo principal a qual a peça é fixada para o fuso que controla o movimento do carro principal. O recâmbio no torno mecânico é o conjunto de engrenagens que transmitem o movimento do eixo principal ao fuso (ELTON).

O projeto “Gear.Simulator” visa a facilidade e melhor custo benefício para a colocação, trabalho e manutenção ao usuário final.

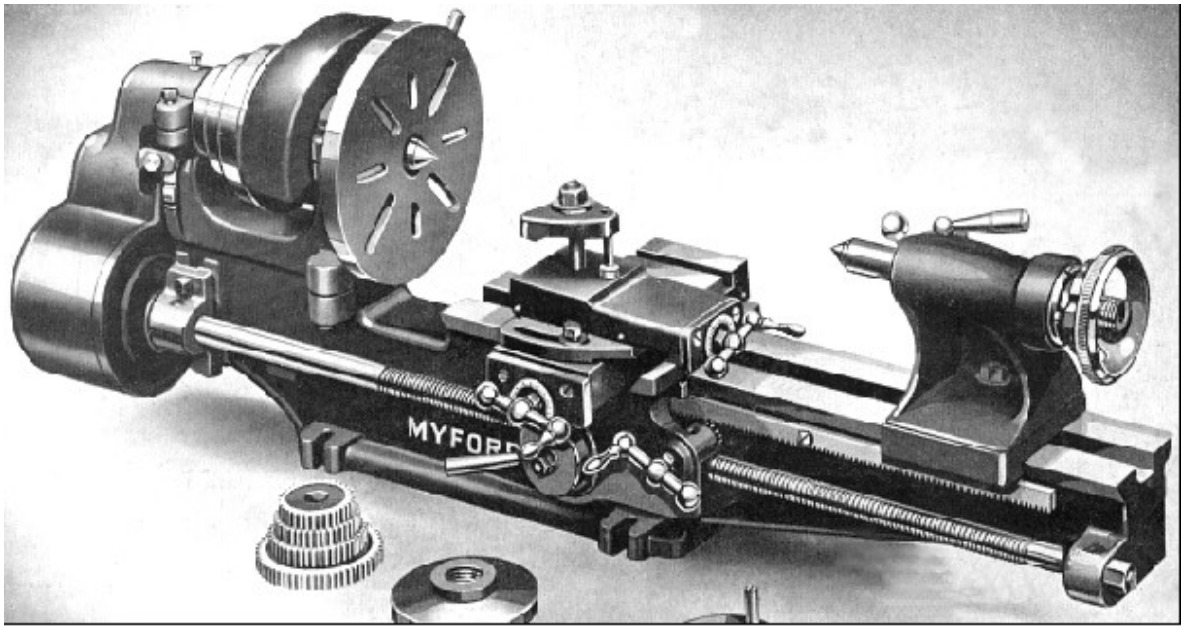
3.1 ETAPAS DE DESENVOLVIMENTO

O projeto foi planejado e desenvolvido no modelo SCRUM, com entregas planejadas e etapas definidas. Baseado nisso, foi usado o modelo de kanban para gerenciamento de tarefas e etapas de desenvolvimento. Foi levantada a viabilidade do projeto, os requisitos funcionais e caso de uso e então foi iniciado o desenvolvimento do mesmo.

3.1.1 VIABILIDADE DO PROJETO

Os principais interessados no produto são torneiros mecânicos e entusiastas que possuem um torno mecânico que não possua todas engrenagens do conjunto ou querem mais praticidade na usinagem de rosca ou desbaste. O foco é fazer um software que possua uma interação com o usuário de forma simples e objetiva, com a escolha de trabalhos que originalmente as engrenagens fazem no torno. Garantindo que o software faça os cálculos corretos para cada rosca e que no trabalho, tenha a precisão e sincronia.

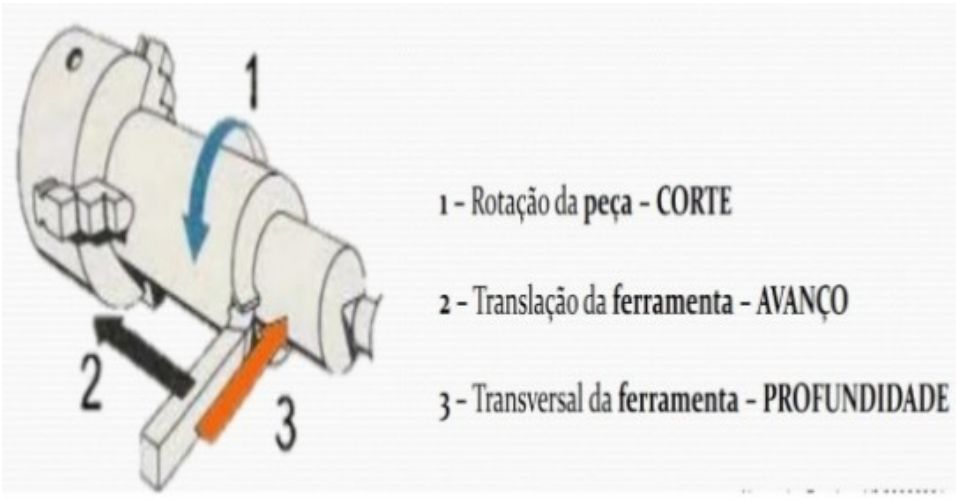
Figura 18 - Torno Myford ML.2



Fonte: (lathes.co.uk, 2015)

Torno mecânico é uma ferramenta eletromotriz que tem como finalidade principal usinar materiais cilíndricos. O material ou peça é fixada na placa do eixo; sendo girada a uma rotação específica por material a usinar. A peça pode ser fixada no carro longitudinal por meio de acessórios ou pelo suporte que alguns modelos já possuem. Pode fazer vários trabalhos, como fresagem, rosca, usinagem de peças cônicas, furos, etc.

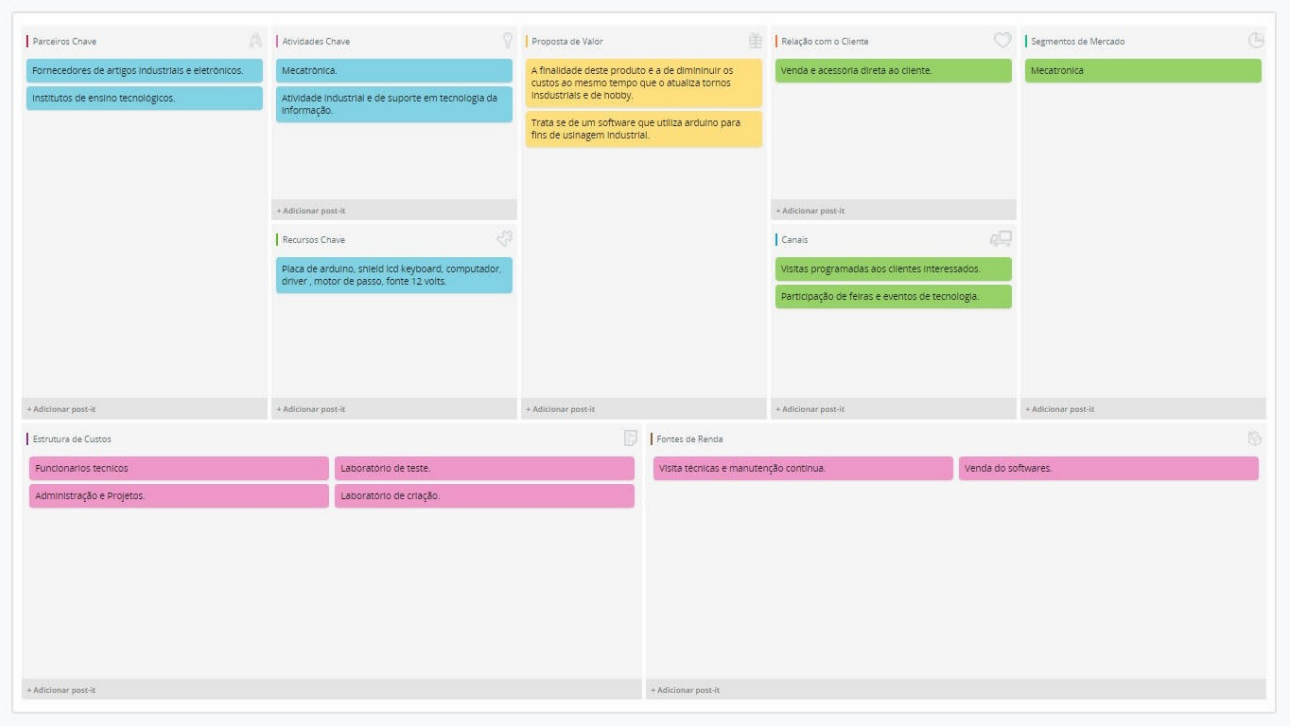
Figura 19 - Movimentos de usinagem realizados pelo torno



Fonte: (CARLOS, 2014)

Foi levantado um canvas de projeto para apresentação explicando os pontos de importância e viabilidade do projeto e seu segmento de mercado usando o modelo disponibilizado pelo SENAI.

Figura 20 - Canvas de negócio

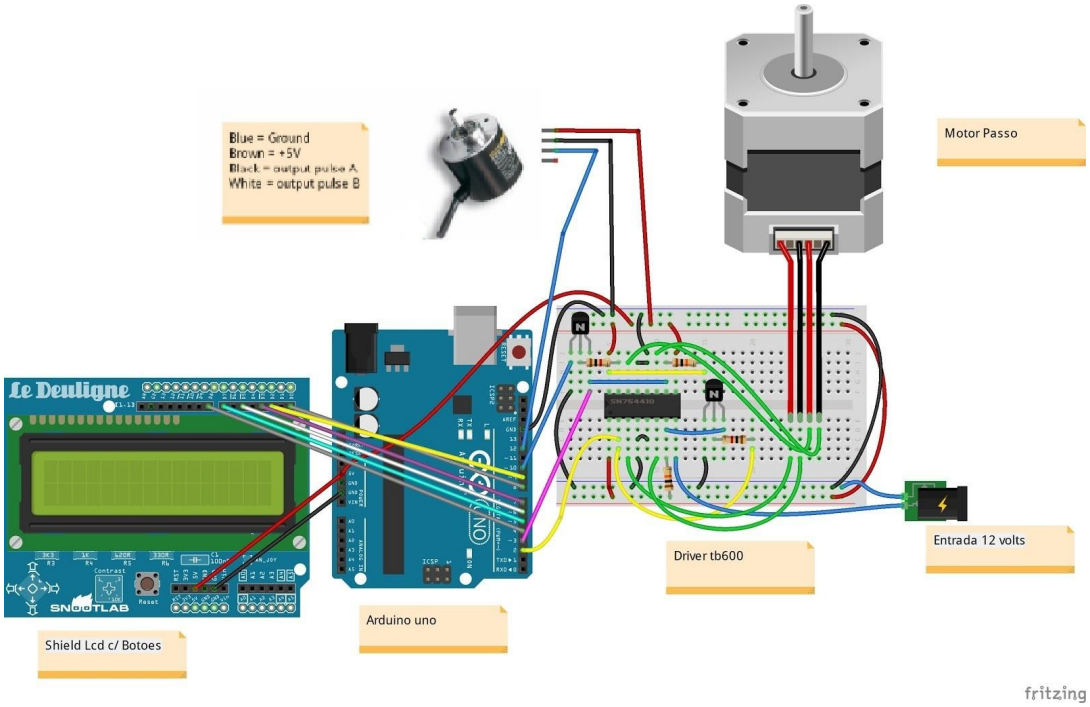


Fonte: (Autores, 2018)

3.1.2 CONSTRUÇÃO

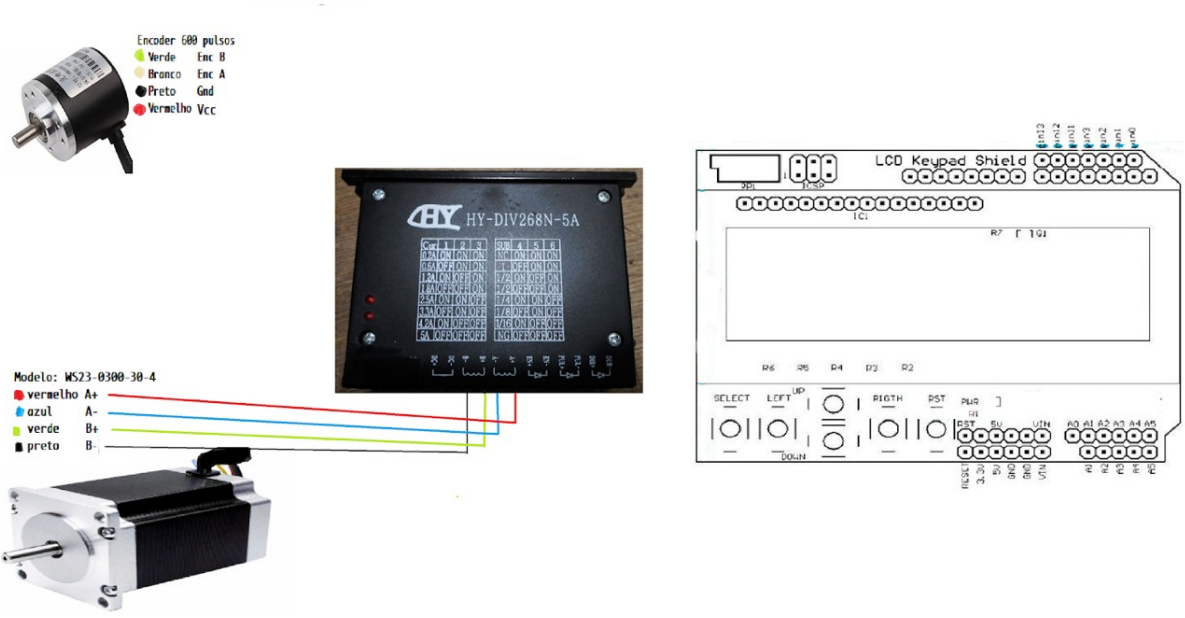
A construção do projeto foi feita por meio do arduino em conjunto de outros componentes eletrônicos como o shield display e o motor de passo, foi documentado com o fritzing a ligação dos componentes Arduino, display e motor de passo na figura 21. A figura 22 apresenta a ligação dos componentes com o outro motor que gera mais pulsos possibilitando o trabalho de usinagem.

Figura 21 - Modelo de construção feito por meio do fritzing



Fonte: (Autores, 2018)

Figura 22 - Modelo de construção



Fonte: (Autores, 2018)

Utilizado a plataforma de versionamento GitHub, para hospedagem e facilidade de download e visualização do código fonte. Como o sistema possui vários cabos, foi feita uma infraestrutura para a proteção dos mesmos.

Foram feitos testes para verificação de resultado e funcionamento das interfaces, além disso foi verificada a segurança do cabeamento usado na infraestrutura e cobertas partes expostas.

3.1.3 IMPLANTAÇÃO

Com o levantamento e planejamento feito a implantação do projeto e desenvolvimento do projeto foi feita, levando em consideração se os componentes funcionavam corretamente entre si e foi feita a documentação de como eles se conectam. Foi avaliada o critério de inovação do projeto, se ele é algo já existente e como agrega valor, e o manuseio e manutenção do usuário final.

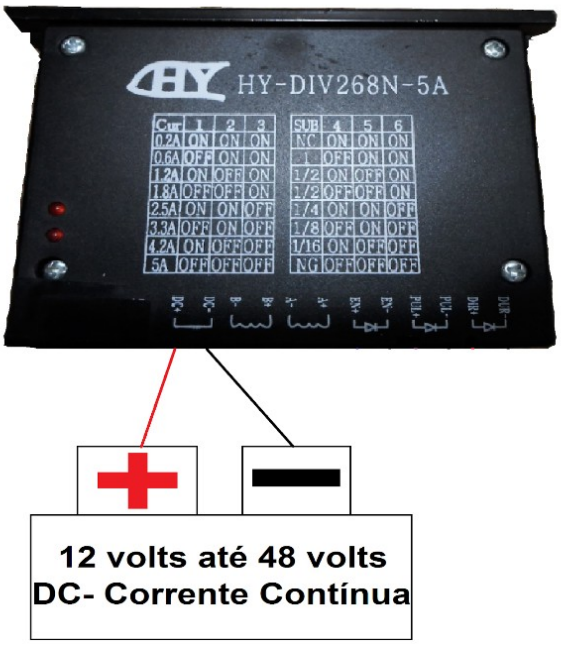
3.1.3.1 TESTES E CORREÇÕES

Durante a implantação foi verificada se o motor girava corretamente de acordo com as especificações. Foram feitos testes de usinagem e diversos tipos de roscas, baseado no resultado apresentado foram feitas correção no código e repostas as peças de acordo com a necessidade encontrada.

3.1.3.2 LIGAÇÃO DOS COMPONENTES

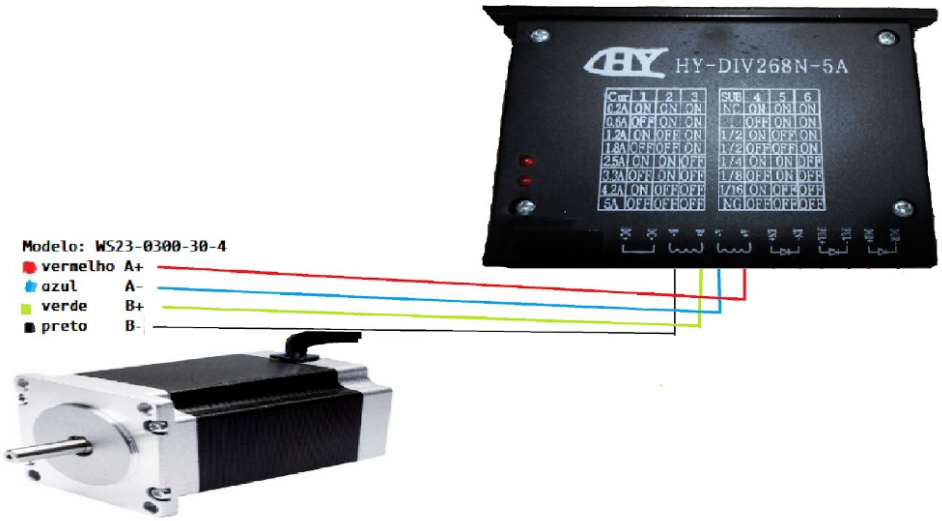
A ligação dos componentes eletrônicos presentes no projeto e voltagens são mostrados nas figuras a seguir, o diagrama foi feito por meio da plataforma de fritzing para exemplificação e documentação.

Figura 23 - Ligação do Driver com Fonte de energia



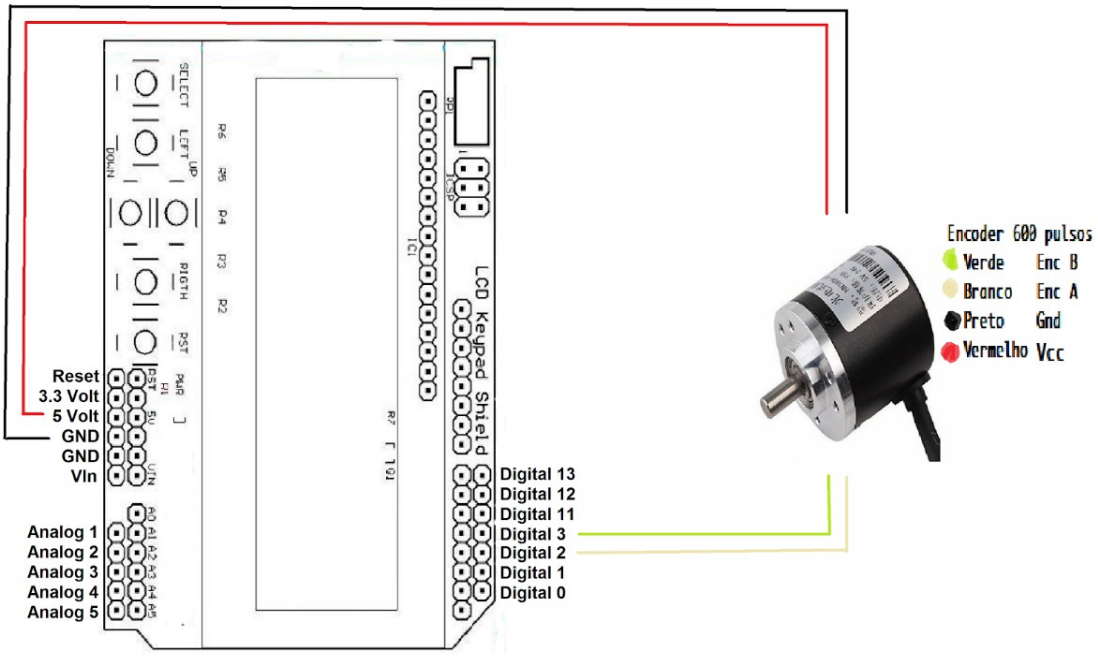
Fonte: (Autores, 2018)

Figura 24 - Ligação do Motor com o Driver



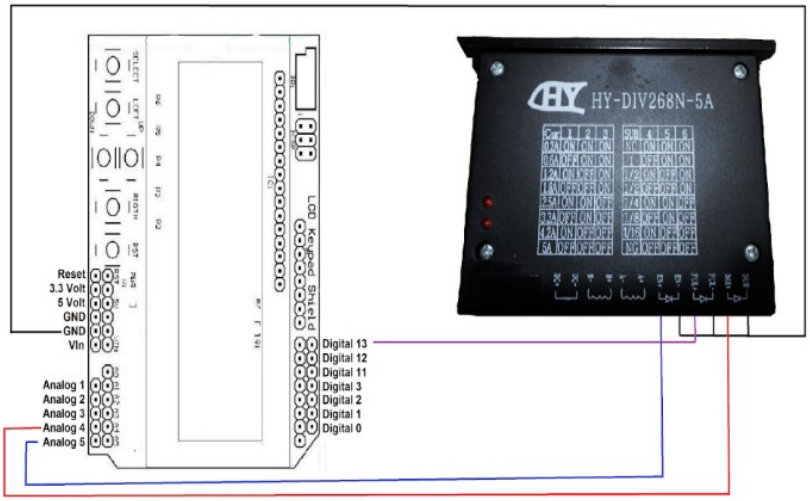
Fonte: (Autores, 2018)

Figura 25 - Ligação do Encoder com o Arduino Uno



Fonte: (Autores, 2018)

Figura 26 - Ligação Driver com o Arduino Uno



Fonte: (Autores, 2018)

3.1.4 AVALIAÇÃO E MANUTENÇÃO

É necessário conhecimento mecânico para manutenção do dispositivo sendo que ele é tem por maior foco o funcionamento do torno feito pelo hardware, as peças usadas no projeto são facilmente encontradas no mercado e não tem um alto custo de manutenção. O encoder com menor sensibilidade tem menos possibilidades de roscas, é necessário levar em consideração esse quesito para o usuário, sendo que caso haja substituição de peças ele entenda essa limitação.

O projeto usa peças de baixo custo, a mais cara sendo a própria placa de prototipagem usada.

3.2 CRITÉRIOS DE INOVAÇÃO

Uma alternativa barata de se ter uma engrenagem em um torno mecânico. O entusiasta, mecânico, marceneiro, etc. pode imprimir a engrenagem ou comprar um torno que possui esse suporte, mas as duas opções têm um custo muito alto, tornando o projeto útil para o uso mais casual do torno. Grandes empresas podem arcar com esses custos, mas usuários mais simples nem sempre, sendo que alguns já possuem um torno anteriormente e só gostariam de ter mais opções.

3.3 DIVULGAÇÃO DO PRODUTO

O produto tem um baixo custo de desenvolvimento, sendo o Arduino um componente open-source de desenvolvimento e prototipagem. O projeto traz o interesse de pessoas que possuem DIY como hobby, sendo que os tornos não possuem sempre a configuração necessária para fazer o torneamento de certa peça, nesse caso é comum a prática de impressão 3D de peças que pode acabar sendo custoso para o usuário final.

4 DESENVOLVIMENTO DO SISTEMA

O desenvolvimento do sistema foi realizado usando a documentação levantada por meio do fritzing e da UML, então ocorreu o desenvolvimento do código que foi compilado a placa do Arduino. Os testes feitos foram de maior parte referentes ao funcionamento do projeto, se seu resultado apresentado estava dentro do esperado e se o aparelho apresenta algum risco ao usuário. Foi considerada a usabilidade do projeto para o usuário, se a mesma está de acordo com o perfil de usuário levantado previamente e como o uso do mesmo é feito.

4.1 LEVANTAMENTO DE REQUISITOS

Foram feitos seguindo os padrões da UML o levantamento de requisitos tanto funcionais quanto não funcionais e o diagrama de caso de uso para entendimento, planejamento e desenvolvimento do projeto.

4.1.1 Definição do sistema

As tecnologias utilizadas no projeto são o Arduino uno, motores de passo, encoder rotativo, driver para motor de passo, e um display de LCD, que são ligados a um torno mecânico já existente. A programação por parte do software é feita em C++.

O Arduino faz a comunicação com os outros componentes como motores de passo, encoder rotativo, tela de LCD e um teclado para input do usuário. Também foi utilizado a plataforma Fritzing para a prototipagem dos componentes e visualização das ligações por fio.

4.1.2 IDENTIFICAÇÃO DOS REQUISITOS

A análise de requisitos funcionais e não funcionais feita seguindo os padrões da UML é apresentada a seguir, explicando o que são requisitos funcionais, não funcionais e apresentando os requisitos levantados em tabelas.

4.1.3 ANÁLISE E CLASSIFICAÇÃO DOS REQUISITOS

Os requisitos são separados em funcionais e não funcionais, requisitos funcionais referem se funções de sistemas ou componentes. Já os requisitos não funcionais referem-se “não ao que o sistema fará, mas como ele fará” (ANTONIO, 2008).

4.1.3.1 REQUISITOS FUNCIONAIS.

Tabela 5 - Requisitos funcionais

Requisito	Tipo	Descrição
RF001	Exibir Menu em tela.	O sistema deve mostrar os menus: 1 - direção, 2 - passo desejado, 3 - rosca polegada e 4 - rosca métrica.
RF002	Mudar o sentido de giro do motor.	O sistema deve ser capaz de mudar o sentido de giro do motor para possibilitar a usinagem com os valores inseridos no cálculo.
RF003	Mudar o estado do driver.	O sistema deve ser capaz de iniciar e parar o driver por meio do botão STATUS para possibilitar a usinagem com o motor.
RF004	Gerar pulsos para o motor girar e fazer a usinagem de desbaste.	O sistema deve ser capaz de gerar pulsos para possibilitar o motor de fazer a usinagem com os valores inseridos no menu.
RF005	Sincronizar eixo com fuso	O encoder deve ler a posição do eixo e de acordo com o intervalo girar o fuso. Resultando na sincronização do eixo com o fuso de acordo com a rosca especificada pelo usuário.
RF006	Gerar cálculo de intervalo.	O cálculo se baseia no número total de pulsos por um giro completo e no encoder e no motor de passo. Esse cálculo é usado tanto no RF005 quanto no RF007. Utilizando valores pré-definidos no código.
RF007	Gerar pulsos para o motor girar e fazer a usinagem de rosca.	O sistema deve ser capaz de gerar pulsos de acordo com o intervalo especificado pelo cálculo do RF006 para possibilitar o motor de fazer a usinagem com os valores inseridos no

		menu.
RF008	Iniciar e parar usinagem.	O sistema deve ser capaz de iniciar e parar a usinagem que está sendo efetuado pelo sistema. Isso sendo feito por meio do menu utilizando os botões de status.
RF009	Entrar e sair de menus	O sistema por meio da entrada do teclado deve ser capaz de entrar e sair de menus diferentes.
RF010	Leitura de entrada de valores	O sistema por meio do teclado ser capaz de manipular a entrada do usuário por meio dos botões + e - para aumento e diminuição de passo utilizando valores já pré-definidos pelo sistema.
RF011	Trocar entre menus	O sistema por meio do teclado deve ser capaz de alterar entre os menus por meio dos botões + e - enquanto o usuário não estiver dentro de um menu.

4.1.3.1 REQUISITOS NÃO FUNCIONAIS.

Tabela 6 - Requisitos não funcionais

Requisito	**Tipo**	**Descrição**
RNF001	Usabilidade do usuário.	A interface do sistema deve ser de fácil entendimento para o usuário, a mesma deve ser explicada por meio de um adesivo n
RNF002	Segurança de componentes.	Os componentes não devem estar expostos, o que pode causar riscos e provocar defeitos e mal funcionamento a longo prazo no sistema.
RNF003	Uso de Arduino uno	O sistema deve ser prototipado utilizando uma placa Arduino uno.
RNF004	C/C++	O sistema deve ser desenvolvido em C/C++ devido a plataforma Arduino utilizar o mesmo e

		ser de maior interesse ao projeto.
RNF005	Voltagem do encoder	O enconder deve ser 5 volts, devido esse valor ser o suportado pelo Arduino
RNF006	Voltagem do driver do motor de passo	O driver deve ter o valor de 12 a 48 volts, devido ao chip TB6600 necessitar dessa voltagem. o limite do mesmo é de 48.
RNF007	Rotação lida pelo encoder na hora de fazer a rosca	A rotação do eixo para usinagem deve ser abaixo de 60 RPMs. Devido a limitação do Arduino que ao usar uma linguagem de alto nível, perde um pouco de desempenho não possibilitando rotações acima de 60 RPMs.
RNF008	Tamanho e espessura do fio do encoder	O fio utilizado no encoder deve ser o padrão, pois fios mais grossos e longos perdem o sinal enviado pelo encoder.
RNF009	Fonte de energia que suporte 5 amperes	O motor de passo e o driver do mesmo necessitam de uma fonte que suporte no mínimo 5 amperes para seu funcionamento
RNF010	Shield LCD com teclado	O código deve dar suporte ao shield e o teclado para entradas de comandos pelo usuário e apresentação de dados.
RNF011	Fios do motor de passo	Os fios de cobre do motor de passo devem ser os padrões que vem com a peça. Devido ao fato de que se o fio for muito fino o mesmo derrete, e se for muito grosso perde sinal
RNF012	Limitação de escolha do passo.	O menu passo desejado possui um limite de 0 até 9,9 milímetros. Pois o mesmo é a velocidade máxima necessária para o trabalho.
RNF013	Limitação de número de roscas em milímetros	A limitação é de 0,25 à 3,25 milímetros. Pois não é possível haver uma sincronização acima desse valor devido ao fuso e encoder.

RNF014	Limitação de rosca polegada.	A limitação é a partir de 11 à 27 fios por polegada. Devido ao fuso e encoder.

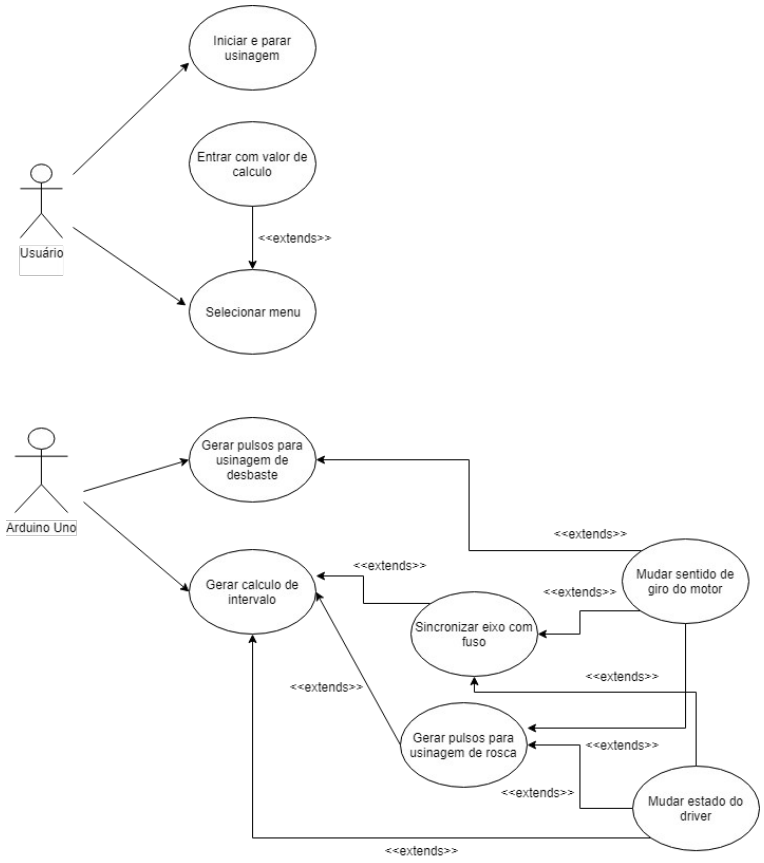
4.2 PROCESSOS

Foi usado o SCRUM para desenvolvimento usando e o modelo KANBAN para planejamento de entregas. Foi usada a UML para a documentação do projeto.

4.2.1 CASO DE USO.

O diagrama de caso de uso auxilia no levantamento dos requisitos do sistema, ele auxilia na comunicação sobre o funcionamento e interações do sistema. (RODRIGO, 2015)

Figura 27 - Diagrama de caso de uso



Fonte: (Autores, 2018)

4.3.1 IMPLEMENTAÇÃO

O desenvolvimento do projeto após a análise e engenharia ser levantada foi feito por meio da linguagem C/C++ e versionado no github. O código então foi compilado na placa Arduino de prototipagem.

4.3.1.1 C/C++

O C++ é uma linguagem de médio nível baseada na linguagem C, ela é uma linguagem compilada multi-paradigma. Na programação para o Arduino pode se utilizar o C ANSI ou C++, para facilitar a implementação do código em vários modelos de Arduino.

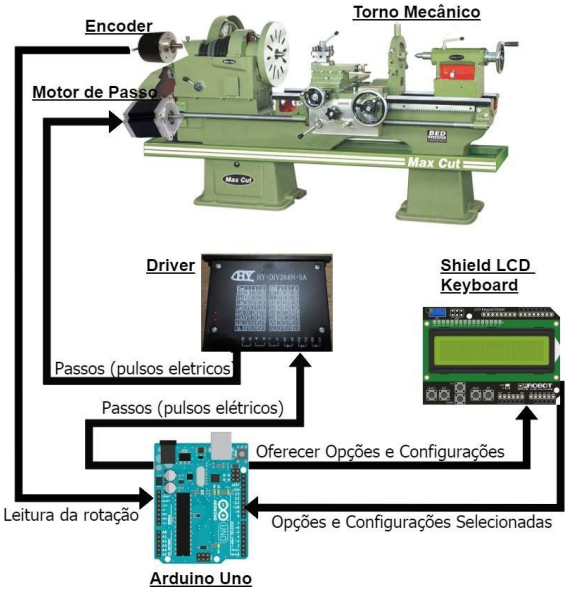
4.3.1.2 CÓDIGO FONTE

O código foi versionado na plataforma GitHub e se encontra para acesso no endereço: <https://github.com/rodalvis/Gear-Simulator>. A aplicação foi feita em C++ que então foi compilada em uma placa Arduino. Ela usa funções para mostrar as informações em uma placa de LCD, receber inputs do usuário e fazer a usinagem de acordo com o que foi requisitado pelo usuário.

4.3.1.3 PROTOTIPAGEM

O protótipo idealizado é representado pela Figura 29, demonstrando como cada componente eletrônico se relaciona e sua funcionalidade no projeto.

Figura 28 - Diagrama do protótipo

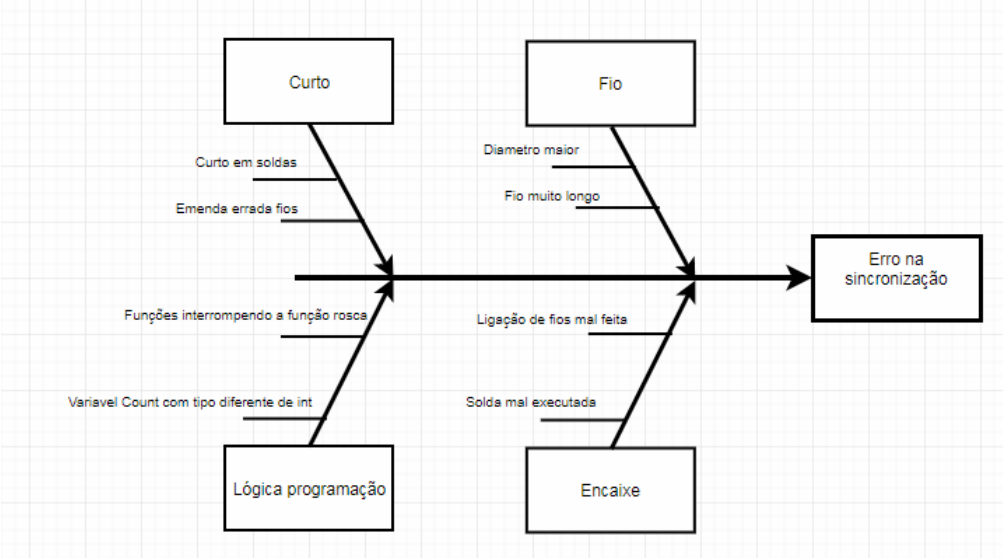


Fonte: (Autores, 2018)

4.4.3 TESTES

Para a documentação foi feito o diagrama Ishikawa para apresentação de possíveis causas de problemas e foi feita a documentação de testes documentando cada teste feito. Além disso foram documentados os testes realizados, os artefatos do sistema, a categoria do teste feito e o responsável pela realização do teste.

Figura 29 - diagrama de ishikawa



Fonte: (Autores, 2018)

Tabela 7 - Tabela de testes

ARTEFATO	CATEGORIA DO TESTE	SITUAÇÃO TESTADA	RESULTADO ESPERADO	RESULTADO OBTIDO	CRITICIDADE	DATA DO TESTE	RESPONSÁVEL PELO TESTE	STATUS DOS TESTE
Shield LCD KeyBoard	USABILIDADE	Teste do funcionamento correto de todos botões	Controle sem interferência/atraso/erro	Erro no componente físico(fabricação), erros esporádicos clicks no botão resultando na	1. MUITO BAIXA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO

				entrada/saída de menus				
Menu Direção	FUNCIONALIDADE	Mudar a direção do carro longitudinal	A direção ser alterada	Direção pode ser alterada normalmente	4. ALTA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO
Menu Rosca Métrica	FUNCIONALIDADE	Configurar a rosca métrica	Escolher o número da rosca, ser calculado e obter o movimento especifico da rosca	Perda de passos, não havendo alteração na velocidade do carro longitudinal por rosca configurada	5. MUITO ALTA	5/25/2018	Rodrigo Augusto A. da Silva	APROVADO
Menu Rosca Polegada	FUNCIONALIDADE	Configurar os fios por polegada da rosca	Escolher o número da rosca, ser calculado e obter o movimento especifico da rosca	Perda de passos, não havendo alteração na velocidade do carro longitudinal por rosca configurada	5. MUITO ALTA	5/25/2018	Rodrigo Augusto A. da Silva	APROVADO
Encoder Rotativo Incremental 600	FUNCIONALIDADE	Leitura correta do giro	600 pulsos por giro completo	600 pulsos por giro completo	4. ALTA	5/15/2018	Rodrigo Augusto A. da Silva	APROVADO
Menu	FUNCIONALI	Configurar	Escolher o	Usinagem	5.	5/20/2	Rodrigo	APROV

Rosca Métrica	DADE	a rosca métrica	número da rosca, ser calculado e obter o movimento especifico da rosca	correta da rosca escolhida	MUITO ALTA	018	Augusto A. da Silva	ADO
Menu Rosca Métrica	FUNCIONALIDADE	Configurar os fios por polegada da rosca	Escolher o número da rosca, ser calculado e obter o movimento especifico da rosca	Usinagem correta da rosca escolhida	5. MUITO ALTA	5/20/2018	Rodrigo Augusto A. da Silva	APROVADO
Menu Rosca Métrica	USABILIDADE	Testar a usabilidade por pessoas que não são da área de usinagem	Saber quando é possível ou não possível fazer rosca, com base do ícone no menu rosca métrica	Soube quando foi possível fazer ou não a rosca. Pelo ícone de parafuso ou X	3. MÉDIA	5/22/2018	Rodrigo Augusto A. da Silva	APROVADO
Menu Rosca Polegada	USABILIDADE	Testar a usabilidade por pessoas que não são da área de usinagem	Saber quando é possível ou não possível fazer rosca, com base do ícone no menu rosca polegada	Soube quando foi possível fazer ou não a rosca. Pelo ícone de parafuso ou X	3. MÉDIA	5/22/2018	Rodrigo Augusto A. da Silva	APROVADO
Cabo encoder	ESTÉTICA	cabo aparente	Cabo escondido na caixa de engrenagem	Cabo devidamente protegido, com espiral	4. ALTA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO

				chata de plástico cor preto				
Cabos driver	ESTÉTICA	Cabo aparente	Cabo escondido na caixa de engrenagem	Cabo devidamente protegido, com espiral chata de plástico cor preto	4. ALTA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO
Cabo fonte de energia	ESTÉTICA	Cabo aparente	Cabo escondido na caixa de engrenagem	Cabo devidamente protegido, com espiral chata de plástico cor preto	4. ALTA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO
Case Arduino	SEGURANÇA	Arduino exposto	Arduino protegido	Arduino protegido pelo case contra óleo e poeiras.	4. ALTA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO
Correia do encoder	SEGURANÇA	Correia exposta	Correia protegida	Correia protegida pela caixa de engrenagem	4. ALTA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO
Acoplador do motor de passo	SEGURANÇA	Motor de passo funcionando a diferentes	Motor fixado sem folga ou vibração	Fixado, sem folga	6. EXTREMA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO

		tipos de passo						
Engrenagens fuso/motor	SEGURANÇA	Funcionamento a qualquer configuração	Engrenagens fixadas sem qualquer problema	Engrenagens fixadas sem qualquer problema	5. MUITO ALTA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO
Fuso	FUNCIONALIDADE	Giro do fuso a qualquer velocidade	Giro livre sem atrito excessivo	Giro livre sem atrito excessivo	6. EXTREMA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO
Suporte do case Arduino	ESTÉTICA	Case do Arduino fixado na caixa de engrenagem	O case feito de plástico ABS, com Arduino e driver dentro protegido. Fixado perto/ao lado/no torno.	Case fixado na chapa de proteção da parede. Feito como uma central, com o Arduino e driver. Ligação dos cabos protegida. Alimentação é de 12v que energiza o Arduino e driver.	5. MUITO ALTA	5/4/2018	Rodrigo Augusto A. da Silva	APROVADO
Fita de proteção	ESTÉTICA	Motor e encoder envolvidos	Fitas de cor branca oferecendo uma	Protegido de sujeiras leves como	4. ALTA	5/13/2018	Rodrigo Augusto A. da	APROVADO

		por uma fita de proteção	proteção média, como óleo e trepidações do movimento	pó			Silva	

Tabela 8 - Tabela de descrição de artefatos

ARTEFATO	DESCRIÇÃO DO ARTEFATO
Menu Passo Desejado	Menu responsável por configurar o passo(movimento) por segundo do carro longitudinal
Menu Rosca Métrica	Menu de configuração do passo em milímetros do carro longitudinal por giro na placa do torno
Menu Rosca Polegada	Menu de configuração de fios por polegada para avanço do carro longitudinal por giro na placa do torno
Menu Direção	Menu de configuração do sentido do carro longitudinal para debaste ou rosca (Direita/Esquerda)
Shield LCD KeyBoard	Tela lcd de 16 colunas por 2 linhas com 6 botões (5 funcionais + 1 reset)
Cabo encoder	cabos de ligação do encoder com o Arduino
Cabos driver	cabos de ligação do driver com o Arduino
Cabo fonte de energia	cabos de ligação da fonte de energia com o Arduino e driver
Case Arduino	capa de proteção do Arduino
Correia do encoder	borracha de tração do encoder com o eixo arvore
Acoplador do motor de passo	acoplador do motor no corpo do torno mecânico. Suporte
Engrenagens fuso/motor	engrenagens responsáveis por transmitir o movimento do motor de passo para o fuso
Fuso	Fuso do torno. Responsável pelo movimento do carro longitudinal
Suporte do case Arduino	acoplador do case Arduino junto a caixa de engrenagens
Fita de proteção	Fita de proteção que envolve o encoder e motor de passo

Tabela 9 - Tabela de tipo descrição de teste

CATEGORIA DO TESTE	DESCRIÇÃO DA CATEGORIA
FUNCIONALIDADE	O SISTEMA FUNCIONA?
USABILIDADE	É FÁCIL USAR O SISTEMA?
INTERFACE	O SISTEMA É ADEQUADO AO PÚBLICO ALVO?
ESTÉTICA	Parte visual do projeto
SEGURANÇA	Contra acidentes ou quebras inesperadas

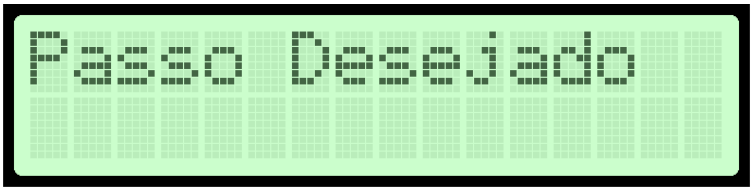
Tabela 10 - Tabela de descrição de teste

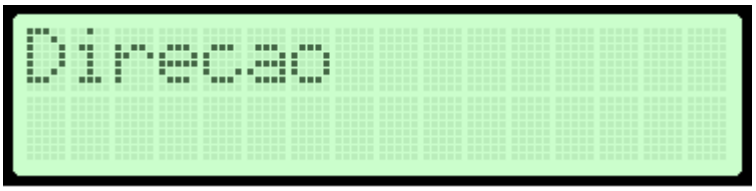
RESPONSÁVEL PELO TESTE	CRITICIDADE	STATUS DO TESTE
Rodrigo Augusto A. da Silva	1. MUITO BAIXA	NÃO INICIADO
	2. BAIXA	EM EXECUÇÃO
	3. MÉDIA	COM PROBLEMAS
	4. ALTA	EM CORREÇÃO
	5. MUITO ALTA	RETESTAR
	6. EXTREMA	APROVADO

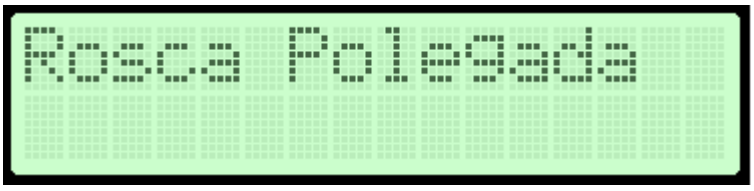
4.4.4 DESCRIÇÃO DAS INTERFACES (telas, relatórios)

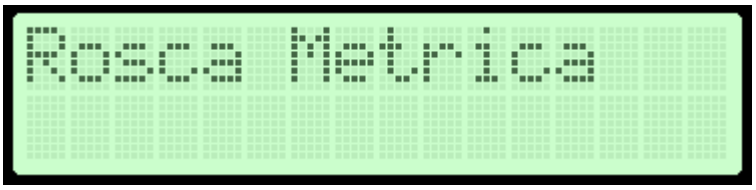
As interfaces foram feitas protótipos das telas pelo fritzing levando em consideração as limitações da tela usadas para o desenvolvimento do protótipo.

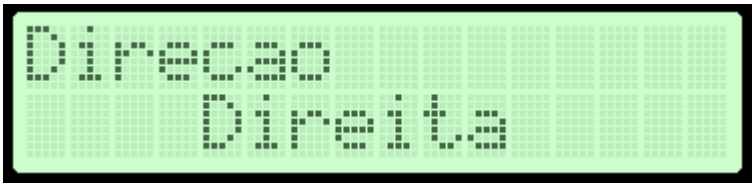
Figura 30 - Protótipos de tela LCD

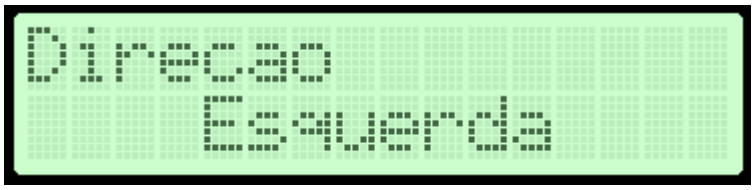


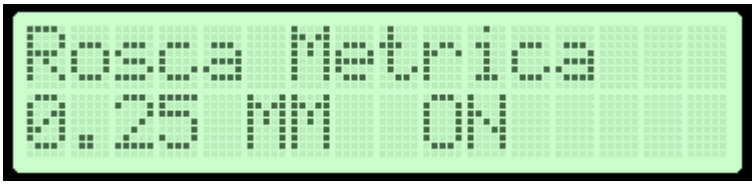

Direcao

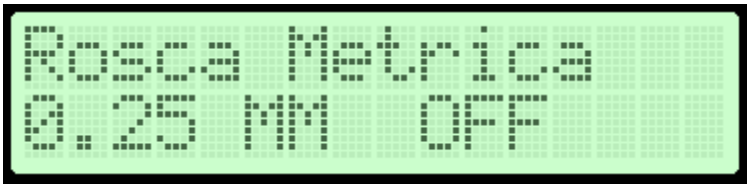

Rosca Polegada


Rosca Metrica

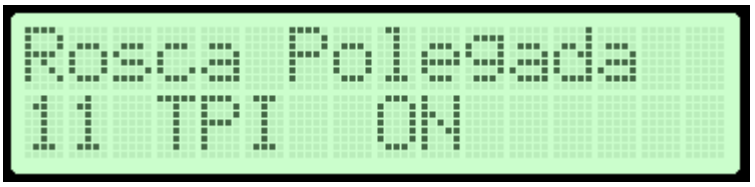

Direcao
Direita

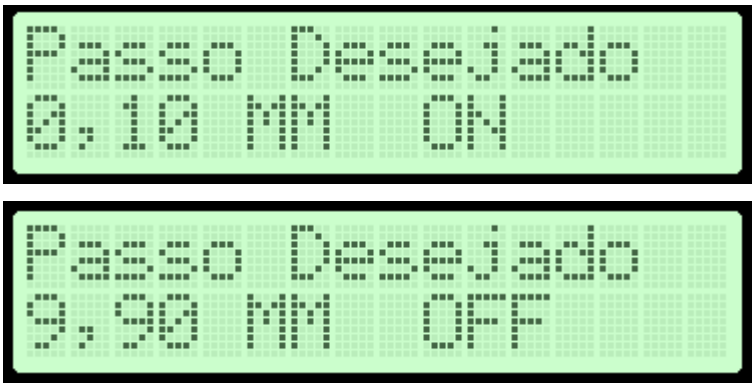

Direcao
Esquerda


Rosca Metrica
0.25 MM ON


Rosca Metrica
0.25 MM OFF


Rosca Polegada
11 TPI OFF

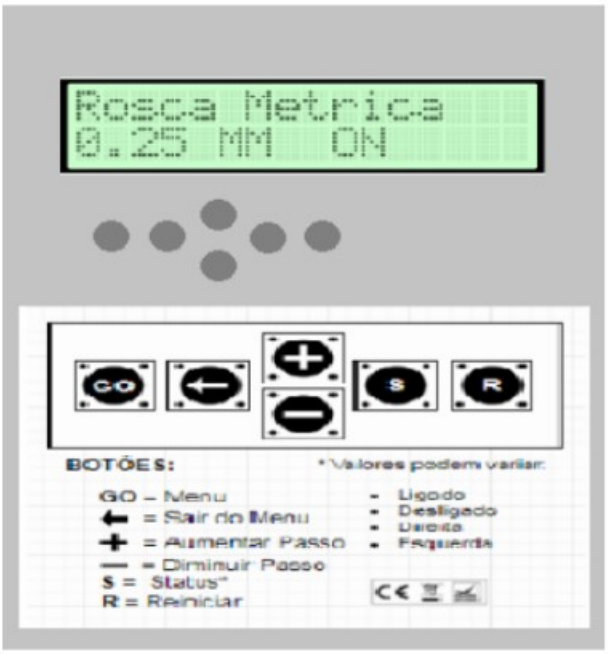

Rosca Polegada
11 TPI ON



Fonte: (Autores, 2018)

Também foi feito o protótipo de infraestrutura do sistema, apresentando sua interface exterior por completo. O protótipo da infraestrutura apresenta a tela, o teclado e o adesivo de instruções de uso que ficam expostos ao usuário.

Figura 31 -Protótipo de infraestrutura



Fonte: (Autores, 2018)

4.4.5 CRITERIOS DE USABILIDADE

O sistema apresenta uma interface de simples uso com um teclado de entrada e um display de LCD, o usuário deve ser capaz de inserir os valores para o cálculo e selecionar o tipo de cálculo para usinagem.

O usuário inicia o Arduino, sendo a primeira tela de navegação de menus, o primeiro menu a aparecer é a Direção, a direção configurada por padrão é o

movimento para a esquerda. O valor do passo pode ser configurado por meio dos botões + e -.

5 CONSIDERAÇÕES FINAIS, RECOMENDAÇÕES E LIMITAÇÕES

Usando o levantamento de viabilidade e interesse do público esse projeto se mostra de interesse para o uso de entusiastas que usam tornos em pequena escala e buscam uma alternativa que não seja custosa, e buscam automação. A construção de um protótipo com a utilização do Arduino demonstrou a possibilidade de desenvolvimento do sistema com um custo baixo e acessível, sendo de interesse a quem trabalha com tornos, mas não tem acesso a equipamentos caros podendo reaproveitar equipamentos já existentes por um baixo custo.

O protótipo desenvolvido neste projeto apresenta uma solução simples que agrega valor em grande escala a pessoas com menos condições financeiras e poder de aquisição. O Arduino combinado com outros componentes se mostrou capaz de realizar as tarefas de usinagem de maneira eficiente e trazendo o retorno esperado pelo escopo criado.

O valor agregado do projeto em relação a entusiastas, marceneiros, mecânicos, etc. que usam o torno para tarefas se mostrou eficiente e uma alternativa de boa qualidade levando em considerações outras opções no mercado. Atualmente as alternativas disponíveis para automação do processo de usinagem e modelagem são impressoras 3D e tornos CNC. As impressoras 3D ou um torno mais novo é um produto caro, no nível de grandes empresas a substituição de peças e aquisição de equipamento mais moderno pode ser considerado algo trivial. Mas para usuários autônomos e pequenas empresas que usam e buscam essa automação o projeto do “gear.simulator” se mostra de grande custo benefício.

Também é de interesse esse projeto a alunos da área de mecatrônica, pois o modelo feito em Arduino e peças de fácil acesso são de bom valor didático para ensino e construção de projetos piloto, ele apresenta conceitos de eletrônica e programação em um código enxuto que apresenta um grande resultado. O projeto é um bom exemplo de introdução ao Arduino apresentando uso completo da placa para prototipagem de um projeto com viabilidade em um mercado.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS – ABNT. NBR 10719: informação e documentação – Relatório Técnico e/ou científico – Apresentação. Rio de Janeiro, 2011.

SCHILDT, Herbert. C completo e total .3ª ed . São Paulo: Makron BOOkS. 1996.

ELTON, Ricardo. Torno Mecânico, Senai. 12 de abril de 2014. Disponível em: https://pt.slideshare.net/EltonRicardo/aula-02-torno-mecnico. Acesso em: 07 de setembro de 2018

PINTO,Gladimir da Silva. Curso técnico de Electromecánica. Instituto Federal Sul-rio-grandense. Disponível em: http://www2.pelotas.ifsul.edu.br/gladimir/Apostida%20de%20torneamento.pdf. Acesso em: 08 de setembro de 2018

ARDUINO.CC. arduino uno rev3 disponivel em: https://store.arduino.cc/usa/arduino-uno-rev3 Acesso em: 09 de setembro de 2018

KALATEC. Disponível em:http://www.kalatec.com.br/definicao-de-motor-de-passo/ Acesso em: 09 de setembro de 2018

DYNAPAR. Encoder absoluto e incremental: Entenda as diferenças!

Disponível em:https://www.dynaparencoders.com.br/blog/index.php/encoders/diferenca-encoder-incremental-absoluto/ Acesso em: 09 de setembro de 2018

WIKIPÉDIA - Ponte_H Disponivel em: https://pt.wikipedia.org/wiki/Ponte_H. Acesso em: 11 de setembro de 2018

BAÚ DA ELETRÔNICA. Driver para motor de passo 5a wd tb6600 wotiom

Disponível em: http://www.baudaeletronica.com.br/driver-para-motor-de-passo-5a-wd-tb6600-wotiom.html. Acesso em: 11 de setembro de 2018

ARDUINO E CIA. Shield LCD 16x2 com Keypad. Disponível em:https://www.arduinoecia.com.br/2013/08/arduino-shield-lcd-16x2-com-keypad.html Acesso em: 11 de setembro de 2018

NEWTON BRAGA. Como funcionam os encoders (MEC128). Disponível em:http://www.newtoncbraga.com.br/index.php/como-funciona/5454-mec128. Acesso em:20 de setembro de 2018

DATASHEET WS23-0300-30-4. datasheet-motor-WS23-0300-30-4.

DATASHEET HY-DIV268N-5A. div268n-5a-datasheet.

DATASHEET LCD-KEYBOARD. LCD Keypad Shield for Arduino. Disponível em: http://www.rajguruelectronics.com/ Acesso em: 23 de novembro de 2018.

14CORE EDITOR. Wiring the LCD 16x2 Keypad Shield on Arduino

Disponível em: https://www.14core.com/wiring-the-lcd-16x2-keypad-shield-on-arduino/ Acesso em:23 de novembro de 2018

POLICOMPCOMPONENTES. MLB-901729093-motor-de-passo-nema-23-30kgfcm-_JM

Disponível em: https://produto.mercadolivre.com.br/MLB-901729093-motor-de-passo-nema-23-30kgfcm-_JM. Acesso em: 05 de outubro de 2018.

Disponível em: https://www.devmedia.com.br/artigo-engenharia-de-software-3-requisitos-nao-funcionais/9525. Acesso em: 05 de outubro de 2018.

Disponível em: https://medium.com/operacionalti/uml-diagrama-de-casos-de-uso-29f4358ce4d5. Acesso em: 05 de outubro de 2018.

APÊNDICE A – Código fonte em C/C++

```
1.  #include <LiquidCrystal.h>
2.  LiquidCrystal lcd(8,9,4,5,6,7);
3.  #define tamanho_vetor(vetor_tamanho) (sizeof(vetor_tamanho)/sizeof(int))
4.  int rosc_ii[] ={8,9,10,11,12,13,14,15,16,17,18,19,20};  // fios por polegada
5.  float rosc_mm[] = {0.25,0.50,0.85,1.0,1.25,1.50,1.75,2.0};  //Rosca passo em milimetro
6.  volatile short int count = 0;   // SEMPRE DEIXAR A VARIAVEL COMO VOLATILE SHORT INT
7.  //Pinagem do Motor e Encoder
8.  #define motor A5 // EN habilita o motor
9.  #define dir 11 // DIR determina a direção
10. #define pul 13 // PUL envia pulsos
11. #define enca 2 // Fio A encoder
12. #define encb 3 // Fio B encoder
13. //Prototipagem das Funções
```

```
14. //int tamanho_vetor(int vetor_tamanho);
15. void Motor_Estado();
16. void V_On_Off();
17. void V_Dir_Esq();
18. void Direcao();
19. void gira(float passo);
20. void ai0();
21. void ai1();
22. void int2pul();
23. void WaitBtnRelease();
24. char ReadKeypad();
25. void MainMenuBtn();
26. boolean rosca=false;       ////////deixar false FALSE
27. //Variaveis Globais
28. short int intervalo = 100; //-------------------------------------------------------------------------------
29. unsigned int v_bt;
30. boolean pulso = LOW;
31. char btn_push;
32. boolean a = LOW;
33. boolean b = LOW;
34. int keypad_pin = A0;
35. int keypad_value = 0;
36. int keypad_value_old = 0;
37. byte mainMenuPage = 1;
38. byte mainMenuPageOld = 1;
39. byte mainMenuTotal = 4;
40. void MainMenuDisplay();
41. void setup()
42. {
43. pinMode(enca, INPUT_PULLUP); // internal pullup input pin 2
44. pinMode(encb, INPUT_PULLUP); // internal pullup input pin 3
45. attachInterrupt(0, ai0, RISING); //"cadastra" a função ai0 para função a ser executada caso o pino 2
    tenha o estado mudado de HIGH ou LOW(qualquer interação no pino será executado a função ai0)
46. attachInterrupt(1, ai1 , RISING); //"cadastra" a função ai1 para função a ser executada caso o pino 3
    tenha o estado mudado de HIGH ou LOW(qualquer interação no pino será executado a função ai1)
47. pinMode(motor, OUTPUT);
48. pinMode(dir, OUTPUT);
49. pinMode(pul, OUTPUT);
50. Serial.begin(9600);
51. lcd.begin(16,2);  //Initialize a 2x16 type LCD
52. MainMenuDisplay();



53. }
54. void loop()
55. {
56. // Serial.println(count); ------------------------------------------
57. btn_push = ReadKeypad();
58. MainMenuBtn();
59. if(btn_push == 'S')//enter selected menu
60. {
61. WaitBtnRelease();
62. switch (mainMenuPage)
63. {
64. case 1:
65. MenuA();
```

```
66.  break;
67.  case 2:
68.  MenuB();
69.  break;
70.  case 3:
71.  MenuC();
72.  break;
73.  case 4:
74.  MenuD();
75.  break;
76.  }
77.  MainMenuDisplay();
78.  WaitBtnRelease();
79.  }
80.  }//---------- End of loop() loop ----------
81.  void MenuA()                    //Função de Menu de Direção
82.  {
83.  lcd.clear();
84.  lcd.setCursor(0,0);
85.  lcd.print("Direcao < >");
86.  while(ReadKeypad()!= 'L')
87.  {
88.  //Insert Task for Menu A here
89.  V_Dir_Esq();
90.  if(ReadKeypad() == 'R'){
91.  Direcao();
92.  delay(150);}
93.  }
94.  }
95.  int MenuB()                     //Função de Menu da Rosca Metrica
96.  {    unsigned short int r = 0;
97.  while(ReadKeypad()!= 'L')
98.  {
99.  lcd.setCursor(0,0);
100. lcd.print("Rosca Metrica");
101. lcd.setCursor(1,1);
102. lcd.print(rosc_mm[r]);
103. lcd.print("MM");
104. V_On_Off();
105. intervalo = intervalo_mm(rosc_mm[r]);
106. if(ReadKeypad() == 'R' && !(intervalo==0)){
107. delay(100);    //Delay de Segurança NÃO RETIRAR.
108. rosca = !rosca;
109. Motor_Estado();
110. V_On_Off();
111. int2pul();   //envia pulsos a cada N intervalos
112. delay(100);
113. }
114. else if(ReadKeypad() == 'U' && r < (sizeof(rosc_mm)/sizeof(int)) ){
115. r++;
116. delay(100);
117. }
118. else if(ReadKeypad() == 'D' &&  r >= rosc_mm[0] ){
119. lcd.clear();
120. r--;
121. delay(100);
122. }
```

```
123.}
124.}
125.int MenuC(){
126.unsigned short int r = 0 ;//Função de Menu da Rosca Polegada
127.while(ReadKeypad()!= 'L'){
128.lcd.setCursor(0,0);
129.lcd.print("Rosca Polegada");
130.lcd.setCursor(1,1);
131.lcd.print(rosc_ii[r]);
132.lcd.print("TPI");
133.V_On_Off();
134.intervalo = intervalo_ii(rosc_ii[r]);
135.if(ReadKeypad() == 'R'){
136.delay(100);     //Delay de Segurança NÃO RETIRAR.
137.rosca = !rosca;
138.Motor_Estado();
139.V_On_Off();
140.int2pul();    //envia pulsos a cada N intervalos
141.delay(100);
142.}
143.else if(ReadKeypad() == 'U' && r < (sizeof(rosc_ii)/sizeof(int))){
144.r++;
145.delay(100);
146.}
147.else if(ReadKeypad() == 'D' && r >= rosc_ii[0] ){
148.lcd.clear();
149.r--;
150.delay(100);}
151.}
152.}
153.void MenuD(){                    //Função de Menu do Passo Desejado
154.float passo = 0.10;              //passo minimo 1mm
155.while(ReadKeypad()!= 'L'){
156.lcd.setCursor(0,0);
157.lcd.print("Passo Desejado");
158.lcd.setCursor(0,1);
159.lcd.print(passo);
160.lcd.print(" mm");
161.V_On_Off();
162.gira(passo);
163.if(ReadKeypad() == 'R'){
164.Motor_Estado();
165.delay(100);
166.}
167.else if(ReadKeypad() == 'U' && passo <= 9.9){
168.passo+=0.1;
169.delay(100);
170.}
171.else if(ReadKeypad() == 'D' && passo >= 0.1){
172.lcd.clear();
173.passo-=0.1;
174.delay(100);
175.}
176.}
177.}
178.void V_On_Off(){                    //printa na tela lcd o estado do motor
179.lcd.setCursor(8,1);
```

```
a ? lcd.print("ON ") : lcd.print("OFF ");
}
void V_Dir_Esq(){                    //printa na tela lcd o sentido de giro do motor
lcd.setCursor(3,1);
b ? lcd.print("Esquerda") : lcd.print("Direita ");
}
void Direcao(){      //muda o sentido de giro do motor. De direita para esquerda ou esquerda para direita
switch(b){
case true :
digitalWrite(dir,LOW);
b = !b;
break;
case false:
digitalWrite(dir,HIGH);
b = !b;
break;
}
}
void Motor_Estado(){          //muda o estado do motor. De ligado para desligado ou desligado para ligado
switch(a){
case true :
analogWrite(motor,HIGH);
a = !a;
break;
case false:
analogWrite(motor,LOW);
a = !a;
break;
}
}
void gira(float passo){        //Função para gerar pulsos para o motor girar.Usada na função passo desejado
while(a){
pulso = !pulso; //inverte o estado da variável
digitalWrite(pul, pulso); //atribui o novo estado à porta
delay(10/passo);
//delayMicroseconds(900);
if(ReadKeypad() == 'R'){
Motor_Estado();
delay(100);}
}
}



int intervalo_mm(float rosca_milimetro){
return(1200/(200/(3.18*rosca_milimetro)));     //calculo intervalo para rosca milimetro
}
int intervalo_ii(int rosca_polegada){
return(1200/(200/(3.18*(25.4/rosca_polegada)))); //calculo intervalo para rosca polegada
}
void int2pul(){
while(a){
if(ReadKeypad() == 'R'){
rosca = !rosca;
```

```
231.Motor_Estado();
232.delay(100);}
233.}
234.}
235.void ai0()
236.{
237.if(count>=intervalo && rosca){
238.// Serial.println("ai0");
239.// Serial.println(count);
240.count = 0;
241.digitalWrite(pul,HIGH);
242.delay(intervalo);
243.digitalWrite(pul,LOW);
244.}
245.else if (digitalRead(3) == LOW)
246.{  count++; }
247.else
248.{  count--; }
249.}
250.void ai1()
251.{
252.if(count>=intervalo && rosca){
253.//Serial.println("ai1");
254.// Serial.println(count);
255.count = 0;
256.digitalWrite(pul,HIGH);
257.delay(intervalo);
258.digitalWrite(pul,LOW); }
259.else if (digitalRead(2) == LOW)
260.{  count--; }
261.else
262.{  count++; }
263.}
264.void MainMenuDisplay(){
265.lcd.clear();
266.lcd.setCursor(0,0);
267.switch (mainMenuPage)
268.{
269.case 1:
270.lcd.print("Direcao < >");
271.break;
272.case 2:
273.lcd.print("Rosca_Metrica");
274.break;
275.case 3:
276.lcd.print("Rosca_Polegada");
277.break;
278.case 4:
279.lcd.print("Passo_Desejado");
280.break;
281.}
282.}
283.void MainMenuBtn()
284.{
285.WaitBtnRelease();
286.if(btn_push == 'U')
287.{
```

```
mainMenuPage++;
if(mainMenuPage > mainMenuTotal)
mainMenuPage = 1;
}
else if(btn_push == 'D')
{
mainMenuPage--;
if(mainMenuPage == 0)
mainMenuPage = mainMenuTotal;
}
if(mainMenuPage != mainMenuPageOld) //only update display when page change
{
MainMenuDisplay();
mainMenuPageOld = mainMenuPage;
}
}
char ReadKeypad()
{
/* Keypad button analog Value
no button pressed 1023
select  741
left    503
down    326
up      142
right   0
*/
keypad_value = analogRead(keypad_pin);
if(keypad_value < 100)
return 'R';
else if(keypad_value < 200)
return 'U';
else if(keypad_value < 400)
return 'D';
else if(keypad_value < 600)
return 'L';
else if(keypad_value < 800)
return 'S';
else
return 'N';
}
void WaitBtnRelease()
{
while( analogRead(keypad_pin) < 800){}
        }
```