UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ENGENHARIA ELETRÔNICA

CRISTIANE BARBOSA PRADO

AUTOMAÇÃO DO DISPOSITIVO CONCHA DE CASAGRANDE PARA O ENSAIO DE LIMITE DE LIQUIDEZ DO SOLO

TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
2019

CRISTIANE BARBOSA PRADO

AUTOMAÇÃO DO DISPOSITIVO CONCHA DE CASAGRANDE PARA O ENSAIO DE LIMITE DE LIQUIDEZ DO SOLO

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Trabalho de Conclusão de Curso 2, do curso de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Bacharel.

Orientador: Prof. Dr. Marcos Roberto Bombacini

TOLEDO
2019

TERMO DE APROVAÇÃO

Título do Trabalho de Conclusão de Curso Nº 93

Automação do Dispositivo Concha Casagrande para o Ensaio de Limite de Liquidez do Solo

Por

Cristiane Barbosa Prado

Esse Trabalho de Conclusão de Curso foi apresentado às 8h do dia 3 como requisito parcial para a obtenção do título Bacharel em Engenharia Eletrônica. Após deliberação da Banca Examinadora, composta pelos professores abaixo assinados, o trabalho foi considerado APROVADO.

_____

Alberto Vinicius De Oliveira
UTFPR

_____

Marcos Vinícius
UTFPR

_____

Marcos Roberto Bombacini
COELE
Orientador (a)

_____

Fabio Rizental Coutinho
Coordenador(a) da COELE

Toledo, 04/07/2019
O termo de aprovação assinado encontra-se na coordenação do curso

Dedico este trabalho à minha mãe, que sempre me apoiou e acreditou em mim.

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Dr. Marcos Bombacini, por todo esforço desprendido nesse projeto.

Agradeço também a banca que avalia meu trabalho. Ambos foram essenciais no desenvolvimento desse projeto.

Aos meus amigos que não hesitaram em me ajudar a completar esse projeto, especialmente o William Fernandes.

Aos meus amigos, colegas e professores que disponibilizaram materiais e componentes eletrônicos quando precisei, especialmente o Prof. Dr. Felipe Pfrimer, minha amiga Camilla Parmignani Afonso e minha amiga Jéssica Cristina Pereira.

Gostaria de agradecer meu namorado, Wagner Eduardo de Souza da Silva, que sempre se disponibilizou a me ajudar da forma que podia e esteve ao meu lado, nesse projeto tão importante.

Agradeço também a todos os professores, servidores e amigos que, durante o desenvolvimento desse projeto, me incentivaram e me tranquilizaram, de modo que eu me sentisse confiante e segura para finalizar essa jornada.

Por fim, agradeço minha família, especialmente minha mãe, que sempre esteve ao meu lado e soube entender a importância desse projeto, para mim.

RESUMO

Foi desenvolvido um sistema eletrônico para automatizar o dispositivo concha de Casagrande. O sistema como um todo deve ser funcional e prático. A rotação do eixo do motor utilizado deve se manter constante em duas voltas por segundo, além de que deve ser exibido o número de rotações completadas ao usuário. E, por fim, foi necessário desenvolver uma lógica para o usuário operar o sistema. Foi desenvolvido um eixo com sistema de acoplamento específico para esse projeto. O motor DC utilizado é controlado por uma técnica de controle de tensão denominada PWM (*Pulse With Modulation*), bem como pela ponte H. E, para otimizar o tempo de resposta do motor DC, foi projetado um controlador PI. Além disso, o número de rotações do motor DC é exibido em um *display*, e considerado como número de golpes, para facilitar o registro. O sistema tem duas chaves *push buttons:* uma liga e desligar o motor DC, em um sistema de *play/pause* (um clique para ligar motor DC e outro clique para desligar o motor DC); e, uma chave reinicia o *display* que exibe o número de rotações para o usuário. O microprocessamento de todo o sistema foi realizado através da plataforma Arduino Nano. O acoplamento do motor junto ao sistema se mostrou prático e eficiente. O eixo do motor DC atingiu a velocidade angular esperada, com oscilações na resposta obtida em torno do *setpoint*. Os botões foram tratados e funcionaram corretamente. O dispositivo automatizado foi comparado com um dispositivo manual, através da realização do ensaio de limite de liquidez do solo, no qual os dados obtidos pelo sistema automatizado se mostraram mais dispersos que o os dados obtidos pelo sistema manual, entretanto a linha de tendência de ambos os sistemas é muito semelhante.

Palavras-chave: Automação. Concha Casagrande. Controlador PI. Limite de liquidez. Motor DC.

ABSTRACT

It was developed an electronic system to automate the Casagrande-device. The whole system must be functional and practical. The motor's angular velocity must be kept constant at two turns per second, and the completed rotation's number must be shown to the user. And, finally, it was necessary to develop a logic to the user to operate the system. An axis with specific coupling system was developed for this project. The DC motor used is controlled by a voltage control technique called Pulse With Modulation (PWM), with the aid of an H bridge. And, to optimize the answer time of the DC motor, a PI controller has been designed. In addition, the number of rotations of the DC motor is shown on a display, and considered as number of strokes, to facilitate the register. The system has two push-button keys: one of them turn on and turn off the DC motor, in a play/pause logic (one click to turn on DC motor and another click to turn off DC motor); and, the other push-button restarts the display that show the number of rotation to the user. The microprocessing of the entire system was done through the Arduino Nano platform. The coupling of the engine next to the system proved to be practical and efficient. The DC motor axis reached the expected angular velocity, with oscillations in the answer signal obtained around the setpoint. The buttons were treated and worked correctly. The automated device was compared with a manual device, by performing the soil liquidity limit test. The data obtained by the automated system were more dispersed than the data obtained by the manual system, however the trend line of both systems is very similar.

Keywords: Automation. Casagrande-device. PI Controller. Liquid limit. Motor DC.

LISTA DE FIGURAS

LISTA DE SIGLAS

AC	*Alternating Current* (Corrente alternada)
DC	*Direct Current* (Corrente contínua)
GND	*Ground* (terra)
ISR	*Interrupt Service Routine* (Rotina de nterrupção do serviço)
I/O	*Input/Output* (Entradas/saídas)
LC	Limite de Contração
LED	Light Emitting Diode (Diodo emissor de luz)
LL	Limite de Liquidez
LP	Limite de Plasticidade
NA	Normalmente Aberto
NBR	Norma Brasileira
NF	Normalmente Fechado
NR	Norma Regulamentadora
PCI	Placa de Circuito Impresso
PD	Proporcional Derivativo
PI	Proporcional Integral
PID	Proporcional Integrador Derivativo
PWM	*Pulse Width Modulation* (Modulação de largura de pulso)
RPM	Rotação por Minuto
SSD	*Seven Segment Display* (*Display* de sete segmentos)

LISTA DE SÍMBOLOS

g(t)	Sinal de controle no domínio do tempo
e(t)	Sinal de erro no domínio do tempo
r(t)	Sinal de referência no domínio do tempo
c(t)	Sinal de saída controlada no domínio do tempo
u(t)	Sinal degrau
E(s)	Sinal de erro no domínio da frequência
L	Constante de atraso de tempo
S	Frequência
T	Constante de tempo
V_{in}	Tensão de entrada
Vcc	Alimentação do circuito
T_i	Constante de tempo integral
T_d	Constante de tempo diferencial
K_P	Ganho proporcional
K_i	Ganho proporcional Integral
K_d	Ganho proporcional Derivativo
Ω	Ohm (Unidade de resistência)
Δ	Diferença da variável que a acompanhar em situações distintas
α	Ângulo formado na amostra de solo preparada para o ensaio

LISTA DE QUADROS

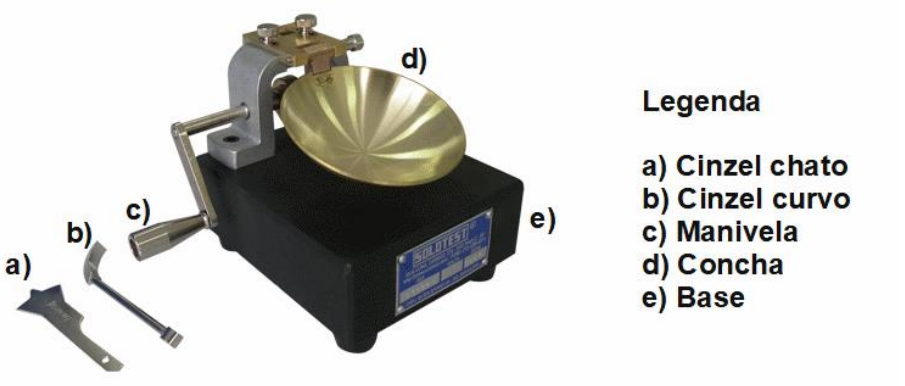
SUMÁRIO

1 INTRODUÇÃO

O limite de liquidez delimita o teor de umidade do solo, no qual o solo deixa de se comportar de forma plástica e passa a ter comportamentos semelhantes a líquidos. Os diferentes tipos solos, mesmo que semelhantes, nunca apresentam exatamente as mesmas características físicas. Porém, determiná-las é essencial para a construção de obras civis.

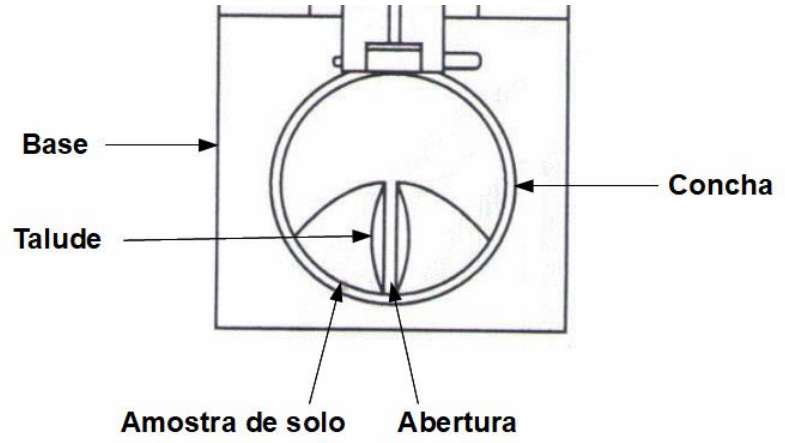
Um dos métodos para determinação de limite de liquidez utiliza a concha de Casagrande (Figura 1). Regulamentado pela NBR 6459/2016, o ensaio exige que o solo seja preparado e depositado sobre a concha do dispositivo. Em seguida, é feito uma abertura na amostra utilizando o cinzel, conforme é apresentado na Figura 2, que, se observado em corte, formam dois taludes com ângulos determinados, que podem ser analisados na Figura 3.

Figura 1 - Dispositivo concha de Casagrande e componentes.



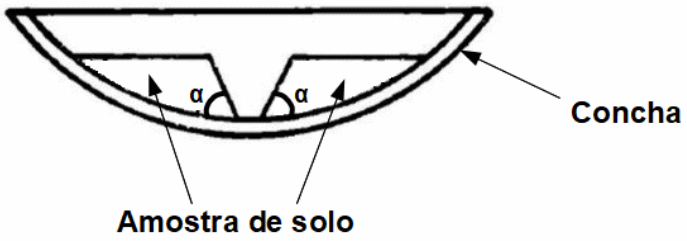
Fonte: SOLOTEST.

Figura 2 - Vista superior da concha com solo preparado para ensaio.



Fonte: Adaptado de NBR 6459/2016.

Figura 3 - Vista em corte da concha com amostra preparada para o ensaio.



Fonte: Adaptado de NBR 6459/2016.

O dispositivo fornece golpes na amostra de solo, através da queda da concha numa altura de 10 mm até a base do dispositivo, a qual é gerada pelo acionamento da manivela e de um came acoplado ao eixo da mesma.

Esse procedimento faz com que aconteça o deslocamento da massa de solo perpendicularmente à abertura realizada.

Durante os consecutivos golpes as bases dos dois taludes tendem a se encontrar, e quando esse contato atinge a extensão aproximada de 13 mm é interrompido o movimento da manivela e registrado o número de golpes. Na sequência é retirada uma amostra da área de contato para determinação da umidade.

Será desenvolvido um sistema eletrônico para automatizar o ensaio e fazer o controle da cadência da concha, além de registrar o número de golpes realizados ao longo do ensaio.

O sistema terá um botão *push-button,* com a função de *play/pause*. Ou seja, ao ser pressionado uma vez, inicia o processo de cadência da concha, e ao ser pressionado pela segunda vez, paralisa o processo. Outro botão, do mesmo modelo, reiniciará os contadores (de golpes e de controle) se o sistema estiver pausado.

Para aumentar confiabilidade e diminuir o tempo de resposta do sistema, será implementado um controlador PI.

1.1 Objetivos

O objetivo principal deste trabalho é automatizar o dispositivo concha de Casagrande, com um sistema prático. De modo que, diminua a interferência do usuário nos resultados e facilita o manuseio e a limpeza do dispositivo.

Para isso, serão concluídos os objetivos específicos:

- Substituir a manivela por um motor DC com tacômetro acoplado para provocar a cadência da concha duas vezes por segundo;
- Implementar um controlador para manter a velocidade do eixo constante em 2 golpes por segundo;
- Incluir um *display* digital para exibir o número de golpes dispendidos durante o ensaio;
- Instalar um botão que ativará o motor, ao ser pressionado, e desativará o motor quando o botão for pressionado novamente;
- Instalar um botão que zera o contador de voltas;
- Projetar placas de circuito impresso de 50 mm x 100 mm;
- Projetar um eixo e um acoplador para o motor DC de modo que, o motor transmita velocidade angular através do encaixe;

1.2 Justificativa

Atualmente o dispositivo concha de Casagrande é operado manualmente por um laboratorista que fica incumbido de três distintas e simultâneas tarefas:

1. Controlar a frequência da cadência da concha;
2. Contar a quantidade de golpes;
3. Perceber quando o material atinge o comprimento de 13 mm de contato.

Kestler (1982) evidencia que testes feitos com amostras similares têm resultados não muito precisos entre diferentes laboratoristas, quando comparados com outro método para obter o limite de liquidez. Kestler ressalta, inclusive, que há variações nos resultados obtidos até quando é o mesmo laboratorista.

Souza (2011) considera que, a frequência da cadência dos golpes influencia diretamente nos resultados do ensaio, além de considerar a técnica do operador como uma das variáveis mais importantes e que provoca maior variação nos resultados obtidos.

Sabe-se que, já existem dispositivos concha de Casagrande automatizados, como o aparelho da Enge Totus mostrado na Figura 4 e da Solotest mostrado na Figura 5. Porém, ambos têm a motorização fixa e sistema de contagem de golpes mecânicos, o que dificulta seu manuseio.

Figura 4 - Dispositivo automático da Enge Totus

Fonte: Enge Totus.

Figura 5 - Dispositivo automático da Solotest

Fonte: Solotest.

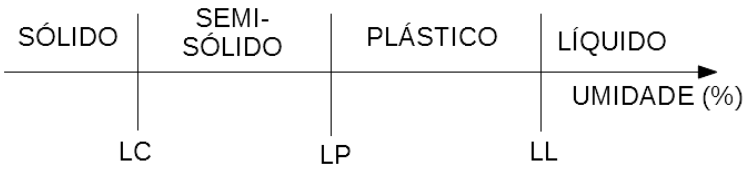
Desse modo, percebe-se que é interessante automatizar o ensaio de Limite de Liquidez do Solo, utilizando-se um sistema estável e de prático acoplamento. E assim, o laboratorista poderá executar o ensaio com maior precisão, tranquilidade e agilidade.

2 FUNDAMENTAÇÃO TEÓRICA

Os solos apresentam quatro estados físicos de acordo com a porcentagem de umidade de cada material. Do menos úmido para o mais úmido, são eles: estado sólido; semi-sólido; plástico; e líquido.

Os limites de umidade, onde cada solo troca de estado, são dados obtidos em ensaios. Também denominados limites de Attemberg, ou limites de consistência (Figura 6), eles são extremamente relevantes para entender como o solo se comportará mecanicamente com o aumento da umidade.

Figura 6 - Limites e estados físicos do solo em relação à porcentagem de umidade.



Fonte: Adaptado *de* Bianchi e Ramos (2013).

Com base no conhecimento prévio das características do material, poderão ser feitas alterações nas propriedades dos solos, ou destiná-los a uma aplicação mais adequada, como: construção de rodovias; barragens; prédios. Além de saber se o material tem capacidade para suportar os projetos civis desenvolvidos.

.

2.1 Ensaio

O ensaio realizado para encontrar o limite de liquidez do solo utilizando o dispositivo concha de Casagrande fornece resultados que variam por consequência de n aspectos. Alguns destes são levantados nessa pesquisa com a finalidade de projetar um sistema que reduz a variação dos resultados.

O dispositivo concha de Casagrande (Figura 1) consiste, segundo Caputo (1988, p. 54), em: “[...] um prato de latão, em forma de concha, sobre um suporte de ebonite”.

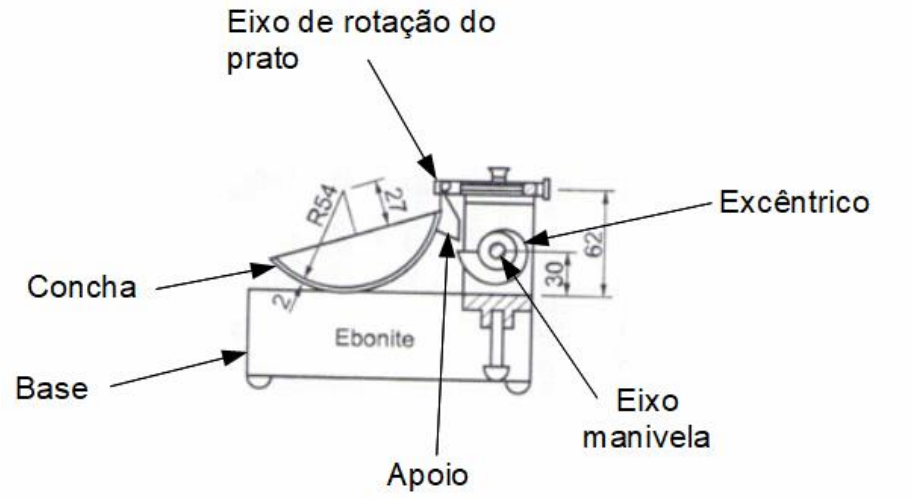
Através do eixo com um came excêntrico acoplado, é imprimido ao prato quedas de 10 mm duas vezes por segundo, com intensidade constante. A NBR 6459/2016 (Solo – Determinação do limite de liquidez) padroniza as dimensões

do aparelho em milímetros, e traz uma vista lateral de como as engrenagens conectadas a um eixo controlado por uma manivela, provocam a queda do prato de latão em 10 mm.

Na vista lateral, mostrada pela Figura 7, pode-se ver como o sistema acoplado ao eixo manivela levanta o prato em 10 mm para, em seguida, provocar uma queda na concha, ao longo de uma volta da manivela.

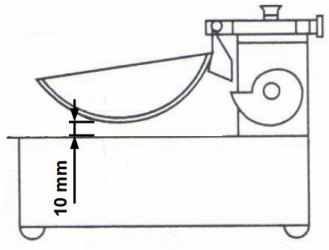
Nota-se que existe um excêntrico fixo ao eixo de forma circular que aumenta seu raio até que exerça na concha a elevação de 10 mm, que pode ser vista na Figura 8, e provoque uma queda abrupta contra a base pela descontinuidade do raio. Consequentemente, ao girar o eixo, esse ressalto empurra o prato através do apoio gerando a elevação da concha em torno do eixo do prato localizado no ponto mais alto do aparelho.

Figura 7 - Vista lateral do aparelho de Casagrande (dimensões em mm).



Fonte: Adaptado de NBR 6459/2016.

Figura 8 - Vista lateral do dispositivo com a concha elevada em 10 mm



Fonte: Adaptado de NBR 6459/2016.

O solo preparado para o ensaio deve ser depositado no prato e, em seguida, é aberto um sulco com o cinzel, de forma a dividi-lo, conforme mostra a Figura 9.

Figura 9 - Material de ensaio sobre o prato (vista em planta).

Fonte: Autoria própria.

Durante o ensaio, parte do material volta a se unir, conforme é apresentado na Figura 10, devido à ocorrência dos impactos contra a base.

Figura 10 - Vista em corte do Prato com material de ensaio dentro.

Fonte: NBR 6459 (2016).

2.1.1 Interferências nos Resultados

Quando se utiliza o dispositivo concha de Casagrande para realizar o ensaio de limite de liquidez do solo, admite-se que, a ele estão associadas algumas variáveis que influenciam no resultado final. Reduz então, a confiabilidade do ensaio. Sendo assim, se faz necessário repetir o ensaio diversas vezes, para analisar a variação dos resultados e obter uma conclusão coerente.

Souza (2011) constata que, a discrepância entre os resultados esperados e encontrado é mais acentuada se a cadência dos golpes for abaixo de dois golpes por segundo. Por fim, Souza conclui que a técnica do operador chega a ser considerada a interferência que mais pode causar variação nos resultados.

A técnica do operador é importante no sentido de que a ele são atribuídas muitas funções e responsabilidades ao mesmo tempo. O laboratorista deve se concentrar no que está fazendo e ser acostumado com o ensaio, para causar o mínimo de interferências possíveis.

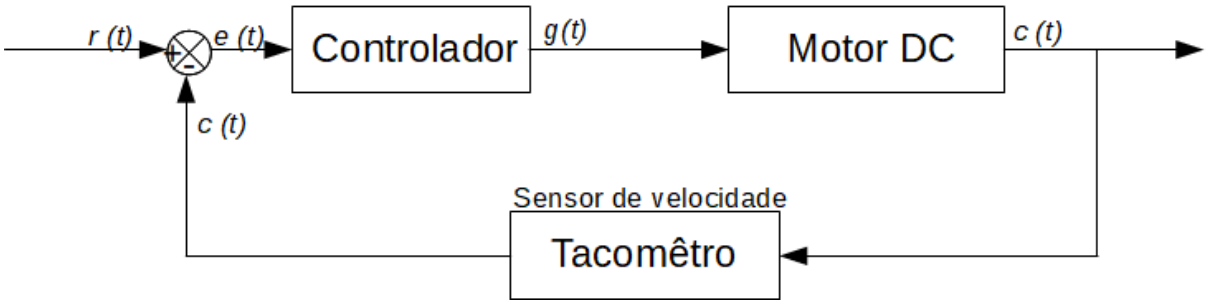
2.2 Projeto Eletrônico

2.2.1 Tacômetro e *Encoder*

O sistema desenvolvido para automação do dispositivo concha de Casagrande deverá controlar a velocidade de rotação do motor e, por consequência, a cadência da concha do dispositivo. A resposta do sistema será a frequência de rotação do eixo, que incide diretamente na cadência da concha do dispositivo. Para coletar a posição de saída do sistema será utilizado um tacômetro acoplado ao motor DC.

O sensor de velocidade enviará o dado coletado da saída controlada do sistema c(t) para o microcontrolador que, por sua vez, irá comparar o sinal c(t) com o sinal de referência r(t). A diferença entre r(t) e a saída controlada é descrita como um erro e(t) que é enviado para o controlador PI. O controlador, por sua vez, emitirá um sinal de controle u(t) para o sistema, de forma a otimizar o tempo de resposta e a estabilidade do sistema. A Figura 11 descreve o sistema em diagrama de blocos.

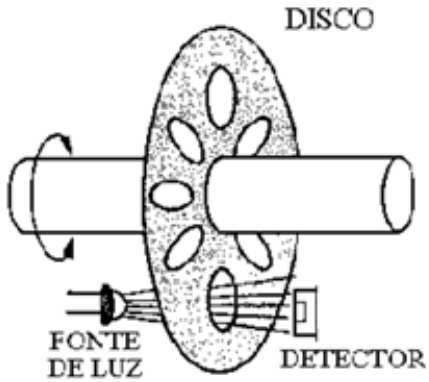
Figura 11 - Diagrama de blocos que descreve a automação do dispositivo concha de Casagrande.



Fonte: Autoria própria.

O tacômetro coleta o sinal de saída da seguinte forma: próximo ao tacômetro tem o dispositivo *encoder* que, segundo Zucatelli e Oliveira (2007), é um disco perfurado anexo ao eixo do motor. O tacômetro emite luz em um ponto e detecta em outro ponto, e o disco *encod*er tem a função de controlar a passagem de luz. Como o disco *encoder* está fixo ao eixo do motor, o tacômetro recebe a informação de velocidade do eixo. Na Figura 12, pode-se observar como o disco *encoder* controla a passagem de luz.

Figura 12 - Esquema com disco *encoder.*



Fonte: Adaptado de Zucatelli e Oliveira (2007).

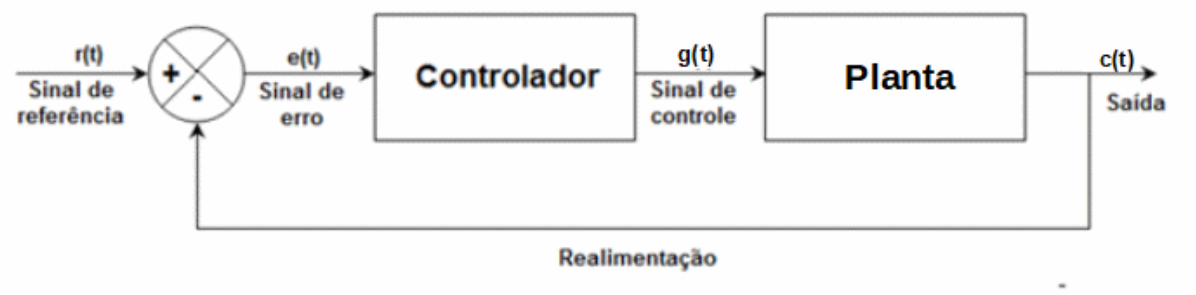
2.2.2 Controlador PI

Baltazar, Andrade e Gouveia (2013) dizem que para tornar um processo mais preciso e sensível às perturbações externas o sinal de saída deve ser comparado com um sinal de referência r(t). A diferença entre esses sinais gera um sinal denominado erro e(t).

O controlador, também chamado de compensador, é o dispositivo que recebe a variação entre os sinais de saída e de referência e calcula o sinal de controle que deve ser aplicado ao processo propriamente dito, com o objetivo de corrigir a variação entre o sinal de saída e o sinal de referência.

. A Figura 13 mostra o diagrama de blocos que descreve o modelo de um sistema controlado em malha fechada.

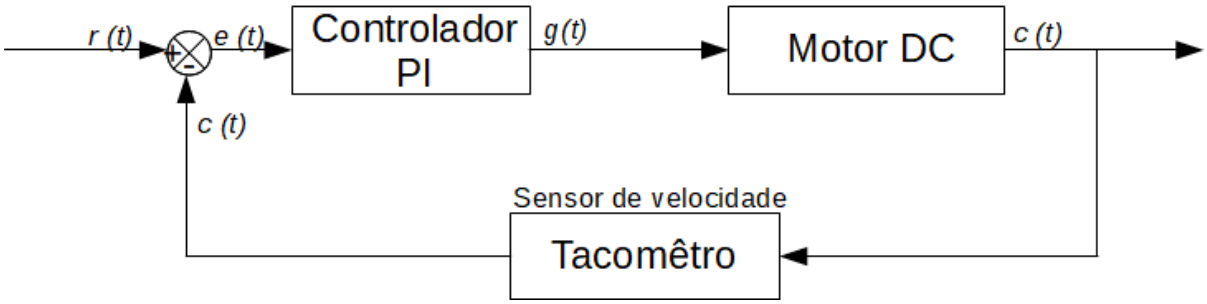
Figura 13 - Diagrama de um sistema de controle em malha fechada.



Fonte: Adaptado de Baltazar, Andrade e Gouveia.

Segundo Zucatelli e Oliveira (2007), um controlador PI tem como objetivo fazer com que os processos sigam com erro nulo. O diagrama apresentado na Figura 14 mostra onde o controlador PI atuará.

Figura 14 - Diagrama de blocos com controlador PI.



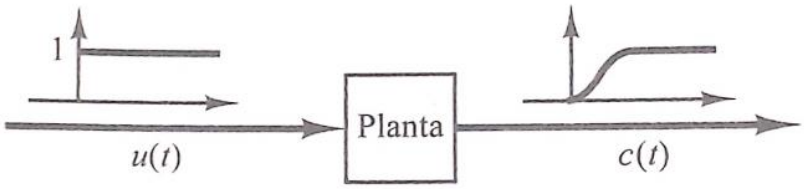
Fonte: Adaptado de Ogata (2010).

O controlador apresentado na Figura 13 é descrito matematicamente pela Equação (1), que está no domínio da frequência, representado pelo símbolo S, através dos parâmetros K_p (ganho proporcional), T_i (constante de tempo integral), T_d (constante de tempo diferencial):

$$K_p(1 + \frac{1}{T_i S} + T_d S) \quad (1)$$

Nesse projeto, será empregada uma das regras de sintonia de Ziegler-Nichols. De acordo com o Ogata (2010), a resposta da planta a uma entrada de degrau unitário, deve ser obtida experimentalmente. Se a planta não possui integradores ou polos complexos conjugados dominantes, a curva de resposta ao degrau unitário muda de sinal, conforme Figura 15.

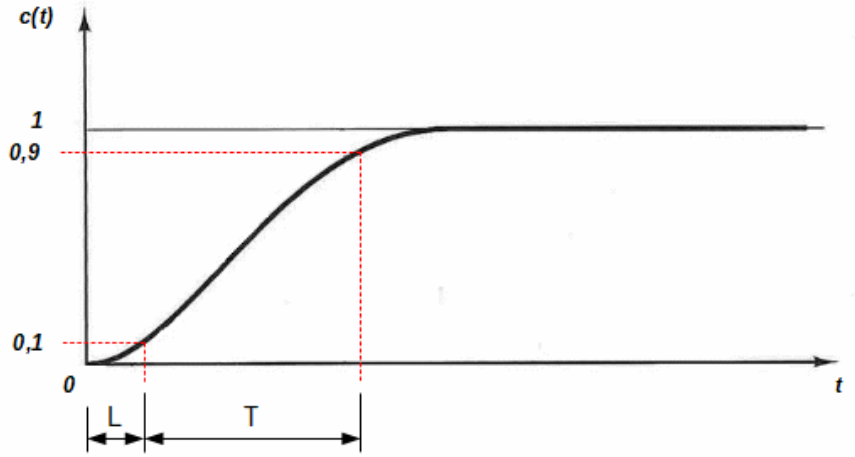
Figura 15 - Resposta ao degrau unitário de uma planta.



Fonte: Ogata (2010).

O Ogata (2010) diz que: “A curva com o formato em S pode ser caracterizada por duas constantes, o atraso L e a constante de tempo T”. O atraso e a constante de tempo são determinados de acordo com o tempo de resposta do sistema, como mostra a Figura 16.

Figura 16 - Curva de resposta em forma de S.



Fonte: Adaptado de Ogata (2010).

Com os dados obtidos experimentalmente (o atraso L e constante de tempo T), é possível encontrar os valores de K_p, T_i e T_d, da Equação (1), através do preenchimento do Quadro 1, de acordo com o controlador desejado.

Quadro 1 - Regra de sintonia de Ziegler-Nichols baseada na resposta ao degrau da planta (primeiro método).

Tipo de controlador	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0{,}9\frac{T}{L}$	$\frac{L}{0{,}3}$	0
PID	$1{,}2\frac{T}{L}$	2L	0,5L

Fonte: Ogata (2010).

O controlador também pode ser descrito pelos parâmetros K_p (ganho proporcional), K_i (ganho proporcional integral), K_d (ganho proporcional derivativo). Para encontrá-los basta multiplicar, na equação (1), o parâmetro K_p que está em evidência, pela função que está entre parênteses.

2.2.3 Resistor *pull-up*

Essa configuração é utilizada nos pinos de entrada do microcontrolador, para leituras de chaves, com o objetivo de eliminar ruídos de informação. De acordo com o site oficial da plataforma Arduino (2019):

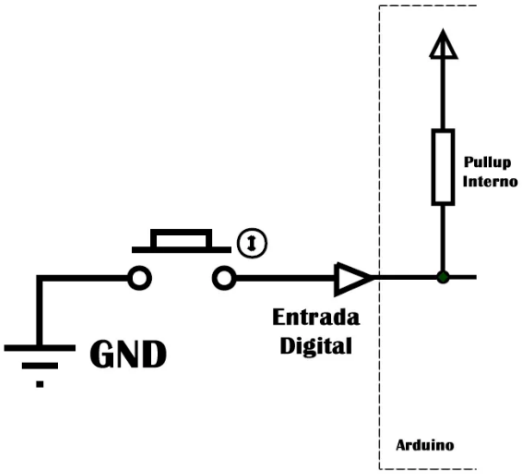
> [...] Os pinos de entrada fazem demandas extremamente pequenas no circuito que estão amostrando [...]. Isso significa que é preciso muito pouca corrente para mover o pino de entrada de um estado para outro [...].
>
> Isso também significa, no entanto, que pinos configurados como pinMode (pin, INPUT) sem nada conectado a eles, ou com fios conectados a eles que não estejam conectados a outros circuitos, reportarão mudanças aparentemente aleatórias no estado do pino, captando ruído elétrico do ambiente ou acoplamento capacitivo do estado de um pino próximo.

Para diminuir oscilações indesejadas na leitura do pino, recomenda-se direcioná-lo para um estado conhecido. Um dos métodos que a plataforma Arduino dispõe para tal é o resistor *pull-up*.

Um resistor interno de alta impedância, ligado em 5 V, é conectado ao pino de entrada. Enquanto que, a entrada de leitura da chave, deve ser ligada ao terra. Dessa forma, se a chave estiver pressionada, a leitura é 0 V e, se não estiver

pressionada, a leitura é 5 V. A Figura 17 mostra o esquema elétrico de ligação *pull-up*.

Figura 17 - Esquema de ligação *pull-up.*



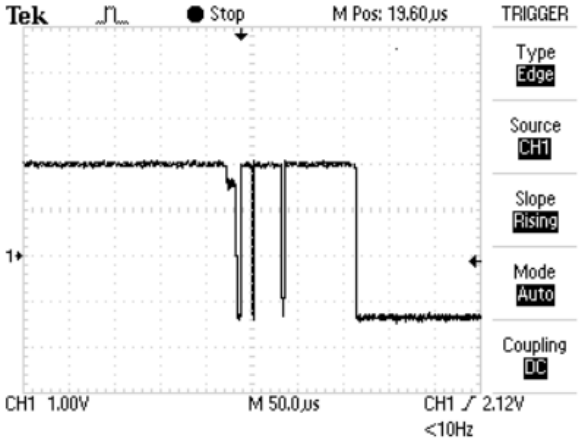
Fonte: Arduino e Tecnologia.

2.2.4 Debounce

Devido às suas características mecânicas e físicas, os botões geram oscilações quando pressionados. Esse efeito, conhecido como bouncing, pode causar leituras equivocadas. Greensted (2010) diz que: “Se esse sinal fosse usado como entrada para um contador digital, por exemplo, você receberia várias contagens em vez da contagem individual esperada”.

Na Figura 18 é possível ver o sinal de entrada oscilando enquanto o botão é pressionado.

Figura 18 - Oscilação do sinal de entrada no momento do chaveamento.



Fonte: The Lab Book Pages.

Existem algumas formas de minimizar esse efeito. Para esse projeto, será feito tratamento via software, através de leituras sequenciais, com um mesmo intervalo de tempo. É uma solução que não gera custo adicional com material, porém acrescenta um tempo de processamento e resposta do sistema. Apesar das desvantagens, é uma limitação quase imperceptível para o usuário, devido à frequência de processamento ser superior a vinte e quatro vezes por segundo, que Rodrigues (2003) classifica como frequência mínima para que o olho humano não perceba as interrupções.

2.2.5 PWM

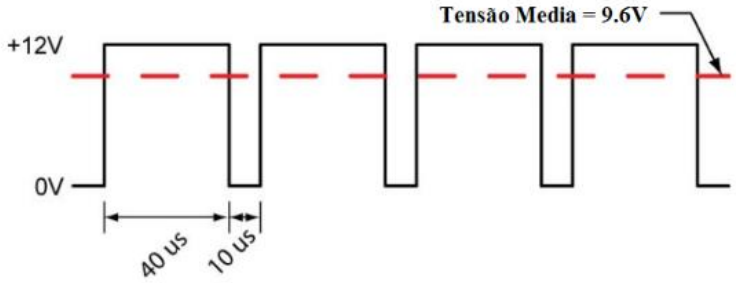
PWM significa “*Pulse With Modulation*” ou modulação por largura de pulso. É uma técnica usada para controlar a tensão entregue à carga. Carga esta que, de acordo com Silveira (2019), pode ser motores elétricos, aquecedores, LEDs ou luzes, entre outras.

A técnica consiste em: controlar o tempo que uma chave fica ligada, em relação a um período. Se o período é repetido numa frequência acima de 24 Hz, de modo que o usuário não processe o chaveamento como interrupções, a tensão de saída para a carga pode ser considerada tensão média do período entre as subidas de pulsos. Ou seja, o tempo de largura de pulso dividido pelo período entre cada subida de pulso é proporcional à tensão média de saída.

A Figura 19 mostra um exemplo onde essa relação é de 0,8. Nota-se que o período tem 50 µs e a largura de pulso é de 40 µs. Já a tensão máxima é de 12 V,

enquanto que a tensão modulada é de 9,6 V. Essa relação proporcional é chamada de *duty cycle* (ciclo de trabalho). Como deseja-se 80% da tensão máxima, o *duty cycle* é 80% do período.

Figura 19 - Exemplo de Modulação por Largura de Pulso.

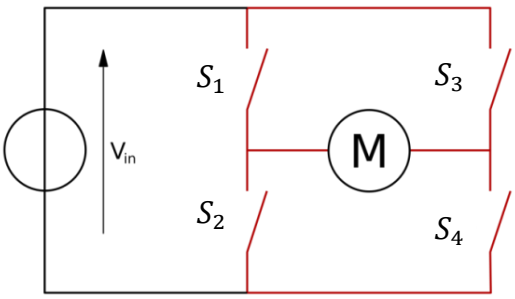


Fonte: Meca Web.

2.2.6 Ponte H

Recebe esse nome porque o esquemático da sua configuração remete a letra “H”, conforme mostra a Figura 20. Ponte H é um circuito que permite variar o sentido de corrente e a polaridade da tensão sobre uma carga, através do chaveamento de componentes eletrônicos. Uma de suas funcionalidades é controlar sentido de rotação e a velocidade de motores DC.

Figura 20 - Esquemático de ponte H.



Fonte: Blog Silvatronics.

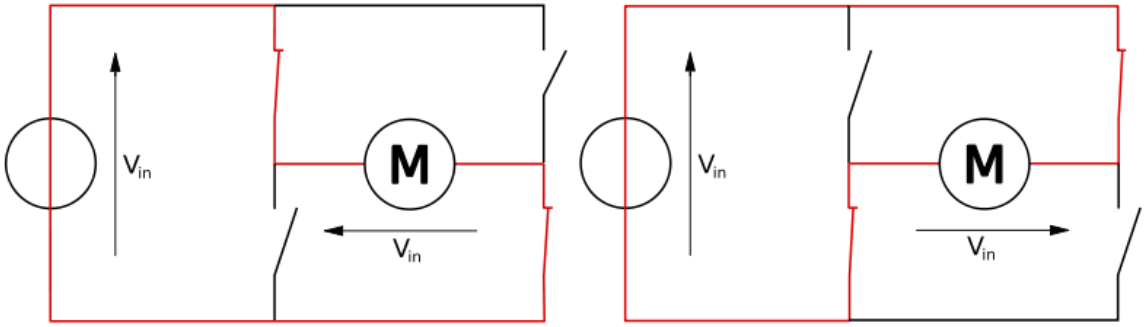
Observa-se que ao conectar o motor nessa configuração, é possível alterar sua polaridade apenas alternando as chaves ligadas. As chaves devem ser ligadas aos pares:

I. S_1 e S_4 Motor ligado;

II. S_2 e S_3 Motor ligado com sentido inverso à configuração anterior;
III. S_1 e S_3 Motor desligado;
IV. S_2 e S_4 Motor desligado;
V. S_1 e S_2 ou S_3 e S_4 não devem ser ligadas juntas porque provocam um curto circuito na fonte.

A Figura 21 evidencia como a polaridade do motor - e por consequência seu sentido de giro - é alterada através do chaveamento da configuração de ponte H.

Figura 21 - Modos de operação da ponte H.



a) Configuração I. b) Configuração II.

Fonte: Blog Silvatronics.

Outro detalhe relevante da ponte H é que ela possibilita o controle da frequência de chaveamento. Ou seja, permite utilizar a teoria do PWM para atingir uma velocidade angular próxima da desejada.

2.2.7 Interrupção

Interrupção é um modo de tratar um evento, externo ou interno ao microcontrolador. De acordo com Angelo (2018): “Interrupção é a quebra da sequência de operação de forma a atender eventos especiais (entrada de dados atendendo solicitações de periféricos, sinal de sensor, sinais internos, etc)”.

Funciona da seguinte forma: quando ocorre determinado evento de interesse, o programa principal é interrompido para que seja executada a rotina de tratamento da interrupção, denominada ISR (*Interrupt Service Routine*).

Para utilizar esse recurso, é preciso configurar a interrupção e a sua rotina de tratamento. Então, assim que o evento ocorrer, ele será tratado. Dessa forma, o programa ganha flexibilidade no processo de execução, uma vez que, não é mais necessário verificar a todo o momento se o evento ocorreu.

Seu uso é interessante para projetos que “esperam” eventos importantes, quase que ao mesmo tempo, e não se tem o controle de quando ocorrerão. Otimiza-se assim, o processamento de eventos.

3 METODOLOGIA

Os métodos utilizados para aplicar a teoria abordada anteriormente foram escolhidos conforme viabilidade de custo e de disponibilidade, além de pesquisa prévia sobre *softwares* e especificidades de *hardwares* que atendem os requisitos do projeto.

3.1 Processador e Microcontrolador

Para escolher o melhor microprocessador para este projeto é preciso saber quais objetivos pretende-se atingir, analisar todos os componentes elétricos e eletrônicos já definidos, além de fazer um breve estudo do que será necessário utilizar na programação do microcontrolador. Com isso, chega-se a seguinte lista de especificidades para o microcontrolador:

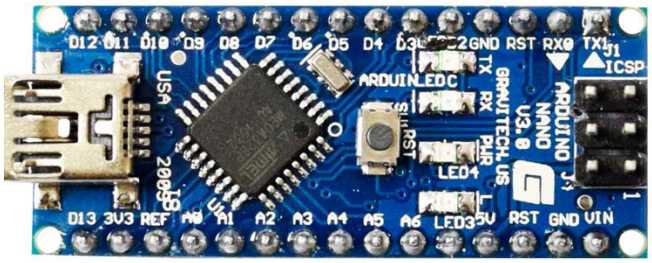
- Dimensões físicas limitadas em 5,3 cm x 6,8 cm x 1,0 cm;
- uma saída PWM, no mínimo;
- 15 entradas e saídas digitais (Portas I/O), no mínimo;
- Facilidade para encontrar componentes compatíveis ou adaptáveis;
- Custo de até R$100,00.

Com essa lista de requisitos, a plataforma Arduino nano, que tem o microprocessador ATmega 328, foi a escolhida para o projeto. Os fatores que mais influenciaram na escolha da plataforma Arduino Nano (Figura 22) foram: a facilidade de encontrar componentes compatíveis com a plataforma; e, apesar de custar em torno de R$ 35,00, o que o classifica como mais caro se comparado com outros microprocessadores, como o PIC16F887 que custa R$17,91, não é necessário comprar demasiados acessórios complementares para possibilitar as conexões.

As dimensões limitantes pré-definidas são equivalentes à plataforma Arduino Uno, que também atende os requisitos do projeto, porém dimensionalmente é maior, comparado à plataforma Arduino Nano. Outra questão é que o Arduino Nano não

tem 15 portas I/O digitais, entretanto suas portas analógicas podem ser configuradas para realizar “leitura” e “escrita” de dados digitais.

Figura 22 - Placa arduino nano.

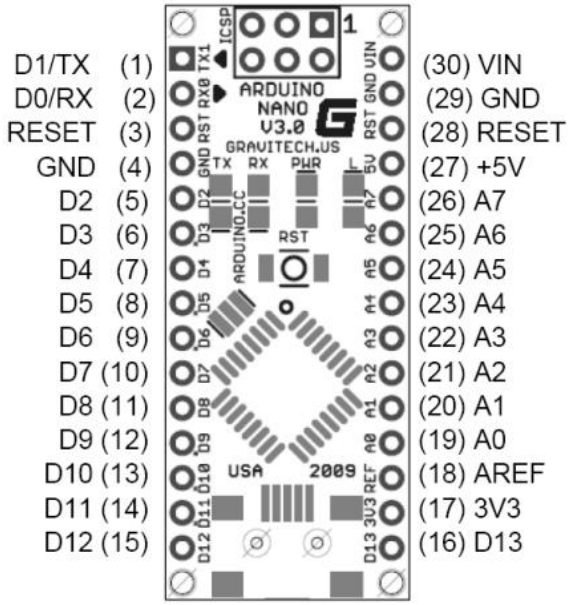


Fonte: Makerlab Electronics.

Isso significa que a plataforma Arduino Nano propicia um dos melhores custos-benefícios para o projeto.

Na Figura 23 pode-se ver como são distribuídos e denominados os pinos da plataforma que foi escolhida para a automação do dispositivo concha de Casagrande.

Figura 23 - Identificação dos pinos do Arduino Nano 3.0.



Fonte: Adri Robot.

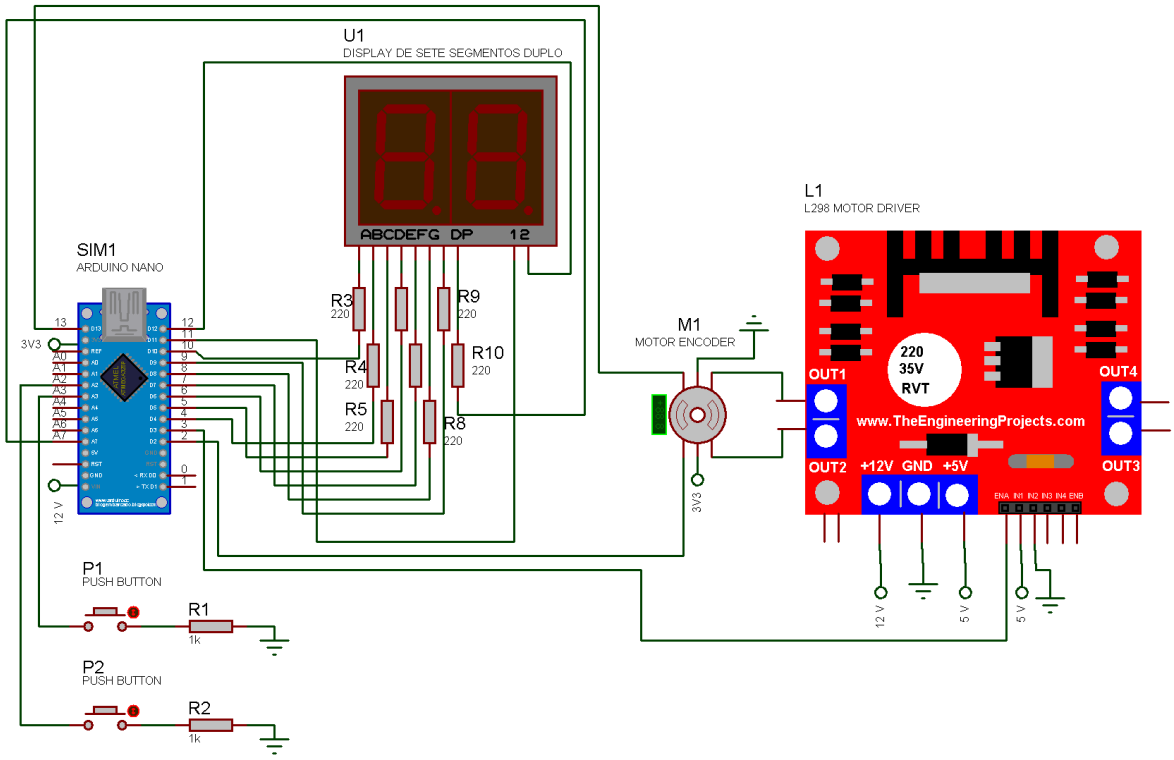
Ao passo que foi definido o microprocessador, também foram definidos os componentes que compõem o sistema: o motor; o tacômetro; o *display*; e os botões. Esses componentes serão descritos e esmiuçados nos tópicos subsequentes. Por

ora, observa-se como ficaram distribuídos os pinos da plataforma, de acordo com a Figura 23:

- D3 conectado ao pino de entrada PWM do módulo l298n (ponte H);
- D2 e D13 conectados às fases do *encoder* do motor;
- D4 - D10 e A7 conectados aos ânodos dos LED’*s* do *display;*
- D10 e D11 conectados nos cátodos dos LED’s do *display*;
- A2 e A3 conectados aos botões configurados como entradas *pull-ups*;
- 3V3 conectado no tacômetro;
- +5V conectado ao módulo l298n (ponte H) em dois pontos, para garantir o funcionamento e o sentido de giro do motor;
- VIN conectado diretamente com a fonte de energia de 12 V DC.

Na Figura 24 pode-se visualizar o esquema elétrico completo em simulação realizada no *software* Proteus. A imagem evidencia a distribuição de pinos da plataforma Arduino Nano e dos componentes eletrônicos selecionados, conforme descrito nos itens acima. Essa simulação foi usada para testar as ligações elétricas dos componentes, bem como botões, o motor DC, a ponte H e o *display*.

Figura 24 - Esquema elétrico do sistema, para simulação.



Fonte: Autoria própria.

Vale ressaltar que grande parte das saídas da plataforma Arduino Nano e suas funcionalidades, são utilizadas no projeto desenvolvido. Sendo assim, a escolha da plataforma se mostra coerente.

3.2 Motor com tacômetro acoplado

Com o objetivo de fornecer o torque necessário para girar o eixo que controla a cadência da concha, será utilizado um motor de corrente contínua, 6 V, 210 RPM, com máximo torque de 0,2941995 N.m e diâmetro do eixo de 4 mm, como o mostrado na Figura 25.

Figura 25 - Motor *encoder* DC 6V.

Fonte: FILIPEFLOP.

Esse motor estará acoplado ao eixo, que tem sua seção transversal mostrada na Figura 7. Eixo este, que incide diretamente na frequência de golpes. Sendo assim, a rotação do eixo deverá ter sua velocidade monitorada pelo tacômetro, para coletar o sinal de saída do sistema.

Mesmo com custo mais elevado, se comparado com um motor DC e um sistema *encoder* externo, o motor com um sistema *encoder* integrado, como o da Figura 25, se mostrou mais interessante para o projeto, pois apresenta menos erros mecânicos.

Uma observação relevante é que, o disco *encoder* interno do motor DC escolhido contém 516 marcações, ou seja, cada volta do eixo do motor gera 516 pulsos.

Observa-se que o motor utilizado tem seis saídas, três relativas ao motor e três relativas ao *encoder*. A Figura 26 mostra a identificação de cada pino do motor, fornecida pelo fabricante.

Figura 26 - Vista traseira do motor DC *encoder* com identificação dos pinos.

Fonte: Fabricante.

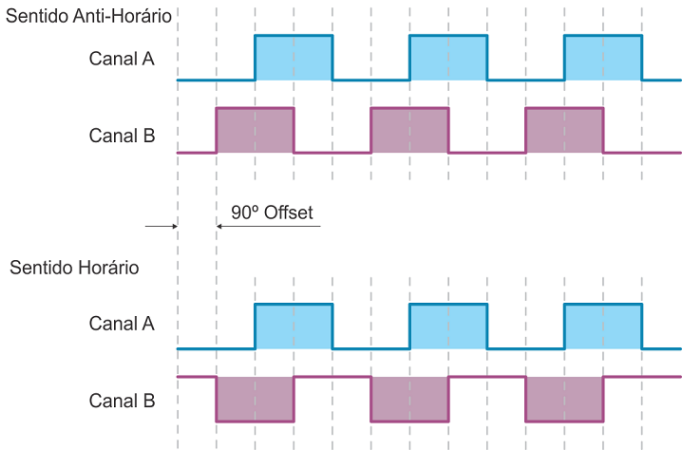
Foram desenvolvidos dois códigos semelhantes para o projeto. O primeiro foi desenvolvido para obter a resposta da planta à entrada degrau u(t). Com os dados coletados da saída c(t) foi projetado o controlador. Para implementar o controlador PI foi desenvolvido o segundo código.

São dois códigos muito semelhantes entre si. O que muda é o acréscimo das configurações necessárias para implementar o controlador PI. O código completo com o controlador PI pode ser visto no APÊNDICE A.

Sabe-se que a saída PWM do Arduino tem 8 bits. Sendo assim, comporta valores entre 0 e 255. Como a tensão mínima de saída do Arduino é de 0 V e a máxima é de 5 V, a resolução da saída PWM é de 19,6 mV por bit. No APÊNDICE B é possível observar como foi configurada a saída PWM do Arduino na fase inicial do projeto.

Após enviar o sinal u(t), é preciso verificar se o motor está funcionando como esperado. O *encoder* integrado ao motor tem duas fases de saída fase A e fase B. Quando a borda de subida de um sinal da fase A se encontra com o sinal da fase B no estado alto o motor gira no sentido anti-horário. Analogamente, quando o sinal da fase B estiver baixo durante a borda de subida do sinal da fase A, o *encoder* gira no sentido horário. A Figura 27 descreve as formas de ondas das saídas de cada fase, conforme o sentido de giro do motor.

Figura 27 - Formas de ondas das fases do *encoder* incremental.



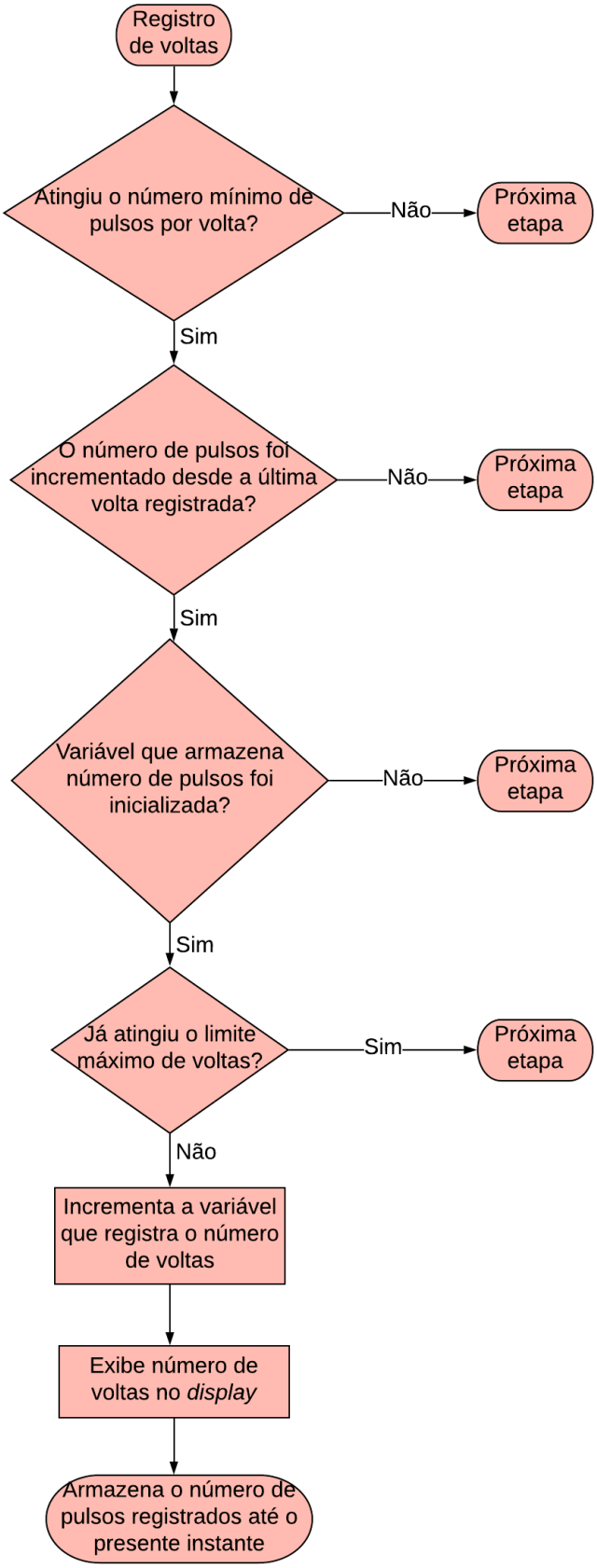
Fonte: HI tecnologia.

Como o motor terá seu sentido de giro controlado pela ponte H, não é extremamente necessário checar a cada pulso essa informação.

É preciso configurar duas portas I/O para leitura da forma de onda das fases do *encoder, c*om atenção especial para a porta que faz a leitura da fase A, pois, para esse projeto, é necessário que seja uma porta habilitada para interrupções externas.

Para incrementar uma volta são feitos quatro testes, conforme o fluxograma apresentado na Figura 28. Caso os testes sejam verdadeiros é realizado o tratamento dos dados.

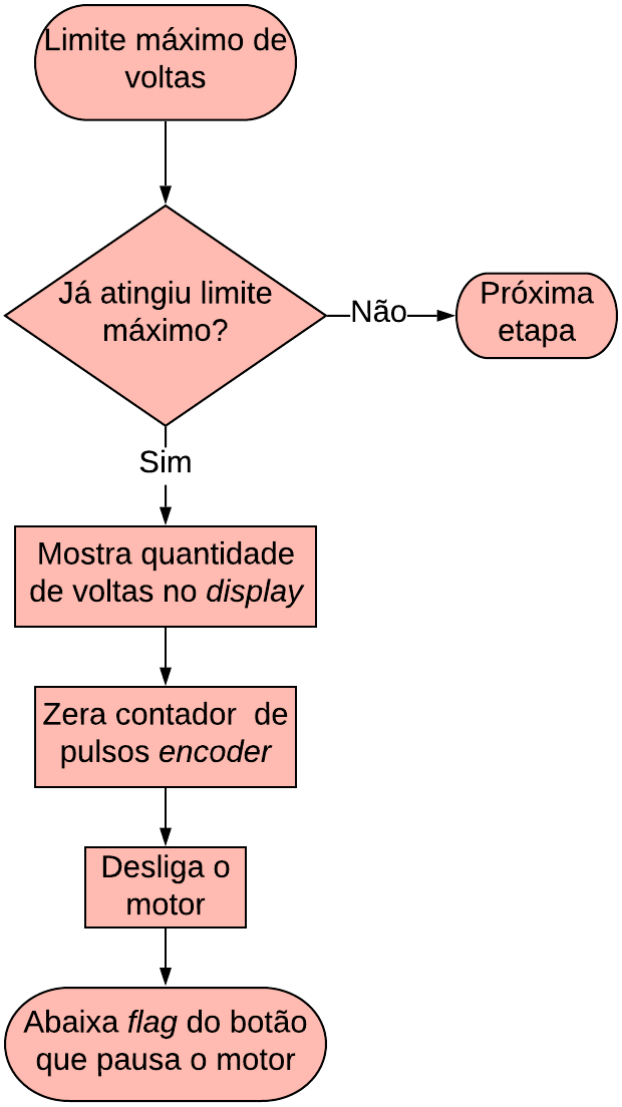
Figura 28 - Fluxograma para registro de voltas completas.



Fonte: Autoria própria.

Baseado na média de golpes necessários por ensaio de limite de liquidez do solo, que variam entre oito e trinta e cinco golpes, foi estabelecido o número máximo de 60 golpes por ciclo do sistema de controle. O fluxograma apresentado na Figura 29 mostra a lógica utilizada para limitar o número de voltas.

Figura 29 - Fluxograma da lógica que limita o número de voltas.

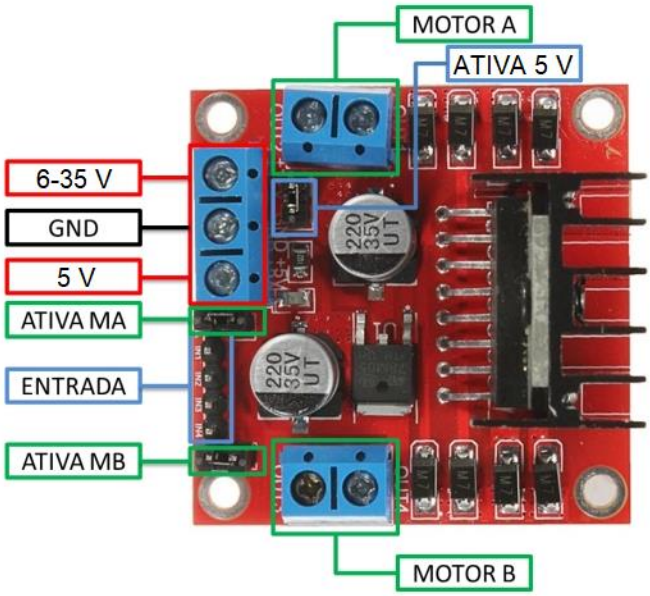


Fonte: Autoria própria.

3.3 Ponte H

Para este projeto foi utilizado o módulo de ponte H l298n (Figura 30). Este módulo é robusto e consegue trabalhar com dois motores ao mesmo tempo.

Figura 30 - Módulo ponte H l298n com identificação dos pinos.

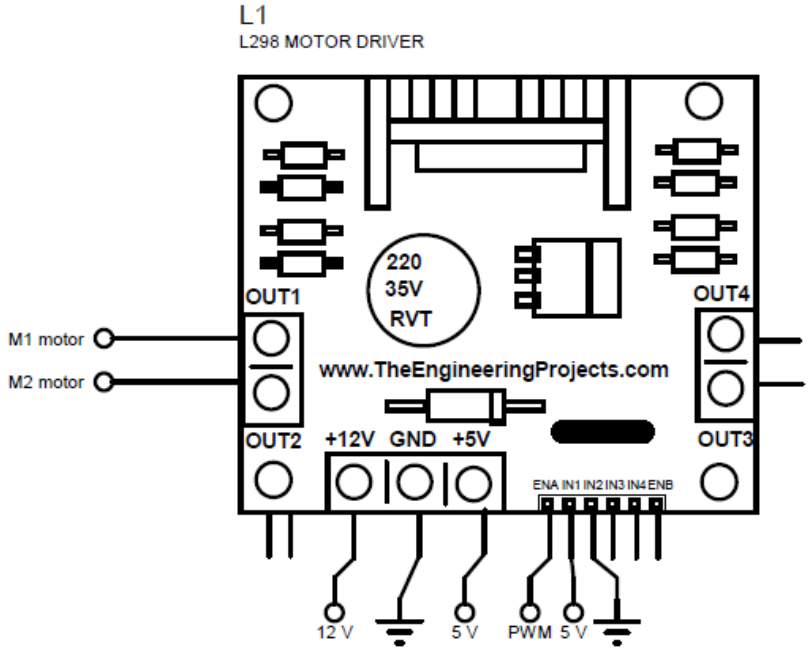


Fonte: FilipeFlop.

Ele foi escolhido por dois motivos: seria difícil desenvolver uma ponte H com dimensões muito menores que esse módulo manualmente; além de que, é um módulo de menor custo comparado com o custo para produzir uma ponte H manualmente.

Para trabalhar com esse módulo deve-se atentar a identificação dos pinos, conforme a Figura 30. A Figura 31 mostra as ligações feitas no módulo. Nota-se que a entrada +12V será ligada direto na alimentação da fonte, enquanto que os pontos conectados à *label* 5 V estão ligados ao pino +5V do Arduino. Os pinos OUT1 e OUT2 estão ligados à entrada M1 e M2 do motor, respectivamente. Enquanto que o pino “ENA” será conectado a saída PWM do Arduino (pino D3). Desse modo será garantido o sentido de giro do motor DC.

Figura 31 - Ligação eletrônica do motor *encoder.*



Fonte: Autoria própria.

3.4 Display de sete segmentos (SSD)

Com a intenção de otimizar as funções disponíveis no controlador e reduzir a quantidade de saídas utilizadas, o modelo escolhido 5621 ASR (Figura 32), que atende as expectativas, foi o escolhido para exibir a quantidade de golpes dispendidos. O *display* terá a função de deixar evidente ao operador do dispositivo o número de golpes já realizados, de forma a facilitar a verificação e registros das quantidades.

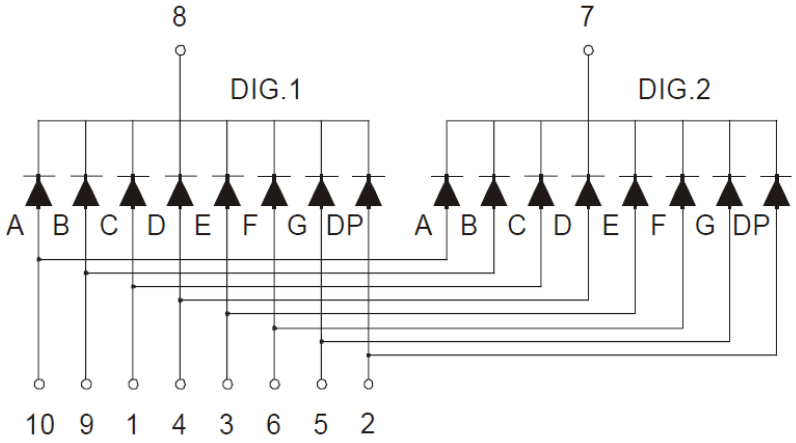
Figura 32 - *Display* de sete segmentos modelo 5621 ASR.

Fonte: Alibaba.

O modelo apresentado na Figura 32 contém dez pinos para controle dos LED's, e são divididos da seguinte forma: dois pinos controlam a polarização dos

dígitos, individualmente; 8 pinos controlam os LED's individualmente, como descreve a Figura 33. Os pinos são ordenados de 1 a 10, a iniciar do pino no canto inferior esquerdo e, terminar no canto superior esquerdo.

Figura 33 - Esquema elétrico do SSD 5621 ASR.



Fonte: MCD ELectronics Inc.

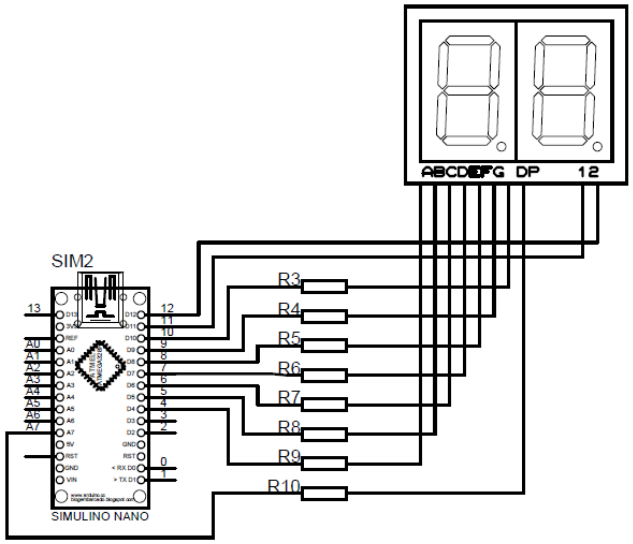
Uma observação relevante no esquema elétrico desse componente é que ele necessita de uma polarização por meio do cátodo, nos pinos 7 e 8. Isso o caracteriza como um modelo de cátodo comum. Essa informação é relevante na polarização do *display* via *software*.

Outro detalhe relevante para utilizar esse modelo no projeto, é que, como não é possível polarizar ambos os dígitos ao mesmo tempo com LED's diferentes entre si, recomenda-se polarizá-los numa frequência maior que 24 Hz. Dessa forma, pode-se polarizar um dígito de cada vez, porém numa velocidade imperceptível ao usuário.

Por último, o fabricante recomenda que a corrente que circula por cada LED seja de 20 mA. Sabe-se que a tensão de saída dos pinos do Arduino é de, aproximadamente, 5 V. Para assegurar um valor próximo dessa corrente será colocado um resistor de 220 Ω em série com cada LED.

A Figura 34 mostra como são feitas as ligações dos resistores em série com os pinos do *display*, bem como, em quais portas I/O eles estão conectados.

Figura 34 - Conexão do display de sete segmentos.



Fonte: Autoria própria.

Como o *display* precisa ser ativado numa frequência maior ou igual que 24 Hz, para obter um melhor desempenho, uma das ferramentas disponíveis na plataforma Arduino é a biblioteca “*SevSeg*”, desenvolvida pelo engenheiro de *Firmware* e *Hardware* Dean Reading.

3.5 Botão *push-button*

O modelo de botão utilizado no projeto pode ser visto na Figura 35. É um botão que possibilita seu encaixe em um envoltório, e dificulta sua conexão a uma PCI. Atende assim as necessidades do projeto.

Figura 35 - Botão.

Fonte: Só tudo.

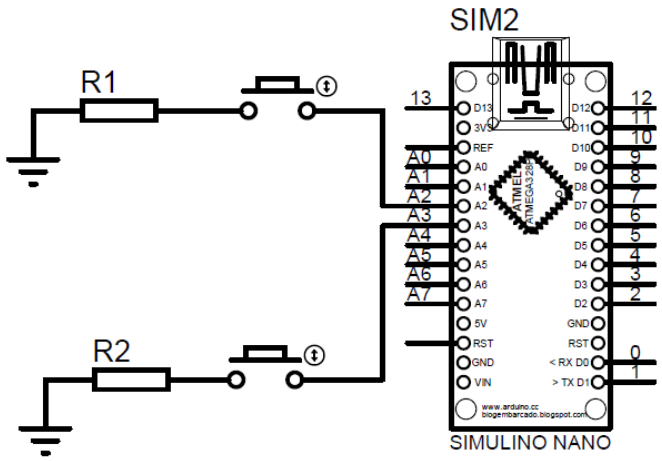
O projeto necessita de dois botões *push-buttons* para facilitar o controle do usuário: um dos botões controla o acionamento do motor, bem como seu desacionamento, num sistema *play/pause*; e o outro reinicia a contagem de golpes dispendidos.

Os botões são conectados aos pinos analógicos do Arduino - que estão configurados com entradas *pull-up*. Em série com o botão é conectado um resistor de 1 kΩ para cada botão, que se encontram ligados ao terra,. Esses resistores tem como função, controlar a corrente que entra no Arduino e evitar danos no sistema.

Sabe-se que o Arduino fornece 5 V de saída, então, ao colocar o resistor de 1 kΩ, a corrente fica limitada em 5 mA. Como o resistor *pull-up* é de alta impedância, não altera a lógica de leitura do botão.

Na Figura 36, pode-se observar como e onde são feitas as conexões eletrônicas dos botões.

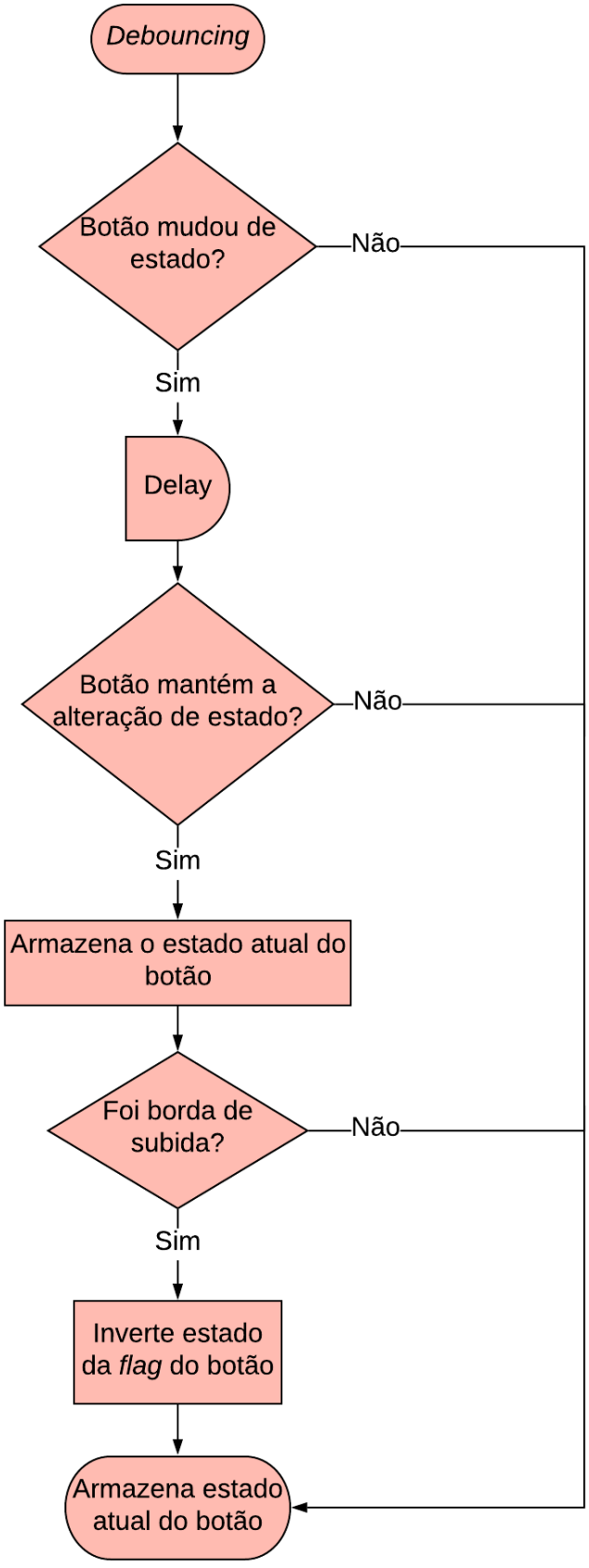
Figura 36 - Conexão eletrônica dos botões.



Fonte: Autoria própria.

As entradas de leitura dos botões foram configuradas, internamente, como *pull_up* para garantir que a porta de leitura do botão não receba ruídos não previstos. Entretanto, os ruídos do efeito *debouncing* são previstos e foram tratados em código, conforme lógica descrita no fluxograma da Figura 37, que descreve exatamente como é realizada a leitura do botão *play/pause*.

Figura 37 - Fluxograma para tratamento do efeito *deboucing.*



Fonte: Autoria própria.

O que muda na lógica de leitura do botão que zera os contadores é que, no lugar de inverter a *flag* do botão, ela é colocada em estado alto.

O artífício de inverter a *flag* foi utilizado no tratamento do efeito *deboucing* do botão *play/pause* para simplificar o processo de ligar e desligar o motor. Pois, uma vez que esse botão é pressionado, não precisa verificar o estado do motor para alterá-lo, já que o estado do motor é condicionado ao estado da *flag*. Por sua vez, sempre que o botão que zera os contadores for pressionado, os contadores devem ser zerados, desde que o motor esteja desativado.

3.6 Fonte de alimentação do sistema

Decidir a forma de alimentação do circuito é relativamente complexo. Pois, existem diversas opções de alimentação, ao passo que deve ser analisada qual a opção que melhor atende as especificidades do projeto. Sendo assim, a escolha se deu através de uma análise específica do local onde o dispositivo é utilizado e suas necessidades, que serão detalhadas a seguir.

Condições do local:

- Laboratório acadêmico;
- Ambiente fechado;
- Bancadas e tomadas elétricas disponíveis.

Necessidades de projeto:

- A alimentação independente (V_{in}) recomendada para o Arduino é, entre 7 V e 12 V;
- O módulo l298n suporta uma tensão entre 6 V e 35 V;
- O projeto desenvolvido utilizará muitas PORTAS do Arduino, bem como, suas saídas de tensão (5 V e 3V3).

Ao considerar esses pontos, nota-se que não há uma grande necessidade de utilizar energia armazenada (baterias). Então uma solução mais coerente e prática é utilizar a energia fornecida pela rede elétrica.

Estabelecida à fonte de energia, é preciso converter a corrente AC (*Alternating Current*) para DC (*Direct Current*). Integrar um circuito conversor AC/DC

ao protótipo, apesar de interessante, iria aumentar as dimensões físicas do protótipo, além de que, poderia expor componentes mais sensíveis à corrente alternada.

Analisado todos esses aspectos, um conversor externo, facilmente substituível, com conectores de entrada e de saída amplamente conhecidos de fácil acesso, se mostrou uma solução adequada.

A partir disso, o modelo mostrado na Figura 38 foi o escolhido. Com entrada de 100 – 240 V (AC) e saída no padrão P4 de 12 V (DC) e 2 A, o modelo tem um custo acessível, entre R$ 9,00 e R$ 15,00.

Figura 38 - Adaptador AC/DC.

Fonte: Lelis Quase Tudo.

A NR 12 (2009) estabelece alguns parâmetros de segurança para trabalhar com máquinas. A norma diz que é preciso garantir que a máquina seja energizada apenas quando o ambiente for seguro para os trabalhadores, e ainda diz que é preciso estabelecer uma forma de parada de emergência.

As NR’s visam proteger os trabalhadores independentemente do ambiente de trabalho, e das condições das máquinas. Então todo o regulamento é feito de forma genérica e abrangente. Porém é possível garantir a segurança elétrica do equipamento aqui desenvolvido, com os pontos levantados pela norma.

O método utilizado para controlar o fornecimento de energia do sistema é uma chave gangorra (Figura 39), que garantirá que o sistema será energizado se o operador assim estabelecer, bem como, o operador poderá cortar o fornecimento de energia a qualquer momento.

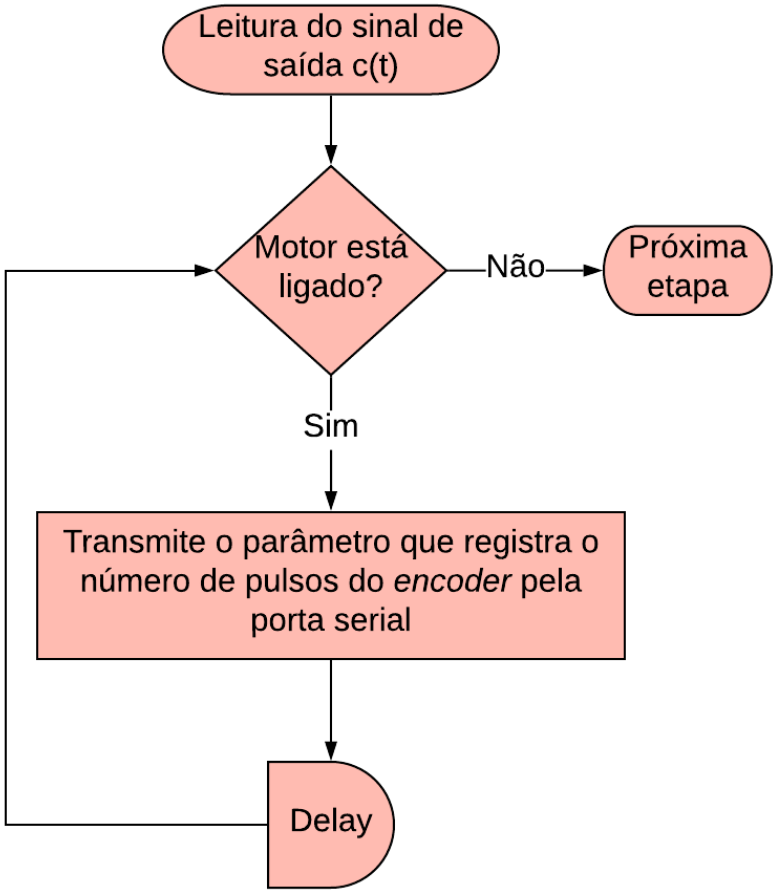
Figura 39 - Chave gangorra de 3 pinos.

Fonte: Eletrogate.

3.7 Controlador

Como já foi dito anteriormente, no tópico 2.2.2, é preciso coletar a resposta da planta à entrada de degrau unitário u(t), para que o controlador seja projetado a partir da curva de resposta obtida. Para isso foi estabelecido um intervalo entre as amostras, de 10 ms, e as amostras foram enviadas pela porta serial seguindo a lógica do fluxograma apresentado na Figura 40.

Figura 40 - Fluxograma da lógica utilizada para leitura do sinal controlado c(t).



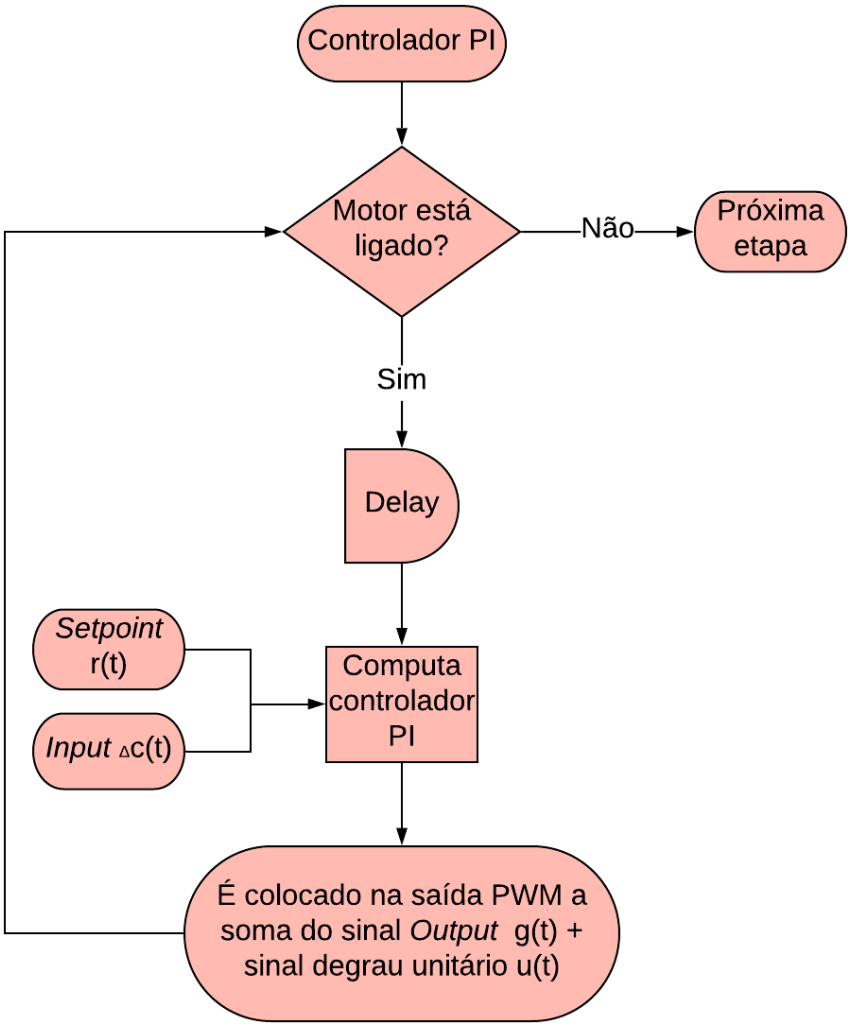
Fonte: Autoria própria.

Projetado o controlador, é necessário implementá-lo. Para isso, será utilizada uma biblioteca própria do Arduino, denominada PID, criada pelo engenheiro químico Brett Beauregard. Para utilizá-la é necessário, de acordo com a plataforma Arduino (2019), usar a sintaxe a seguir, logo após a inclusão da biblioteca: “PID(&*Input*, &*Output*, &*Setpoint,* Kp, Ki, Kd, Direção);”.

A lógica usada para utilizar essa biblioteca é descrita no fluxograma da Figura 41. Observa-se que o parâmetro do controlador “*Input”* recebe a variação do sinal controlado c(t), representado por ∆c(t) no fluxograma, conforme a equação (2).

$$\Delta c(t) = c(t)_i - c(t)_{i-1} \quad (2)$$

Figura 41 - Fluxograma da lógica utilizada para computar controlador PI.



Fonte: Autoria própria.

4 ANÁLISE DOS RESULTADOS

Após aplicar as metodologias apresentadas no Capítulo 3, foram analisados e registrados os resultados obtidos. Os resultados serão apresentados nesse capítulo detalhadamente por tópico, para possibilitar a avaliação do projeto desenvolvido.

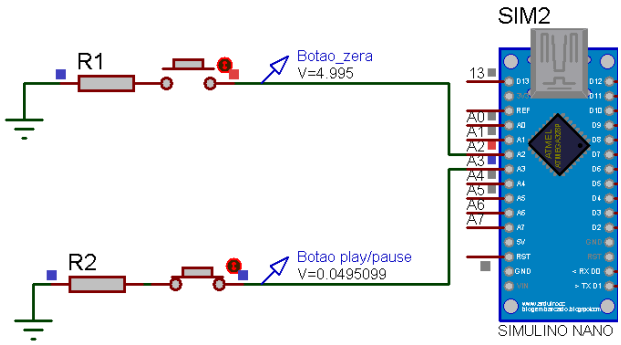
4.1 Simulações

Os componentes foram simulados separadamente em blocos relacionados, para verificação das ligações elétricas dependentes entre si.

4.1.1 Botões

A Figura 42 mostra as ligações elétricas dos botões. Observa-se que um botão está pressionado e o outro não. Foi colocado um medidor de tensão na entrada da plataforma Arduino para analisar a alteração de tensão para essas duas situações, de modo a comparar a tensão no caso em que o botão está pressionado e no caso em que o botão não está pressionado.

Figura 42 - Simulação das ligações dos botões.

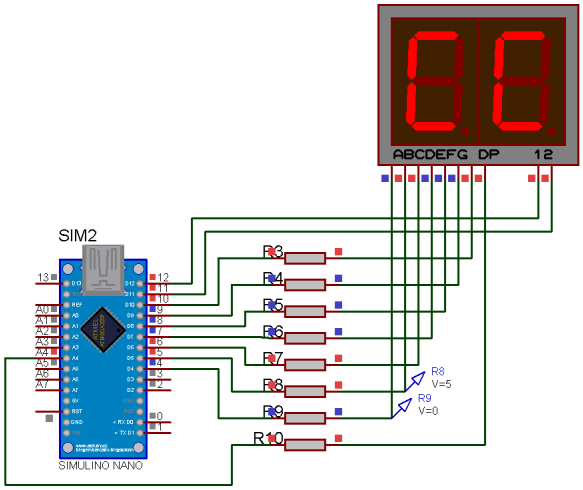


Fonte: Autoria própria.

4.1.2 Display

Foi desenvolvido um código para testar as ligações elétricas do SSD, de modo a formar a letra “C” no *display*. Observa-se na Figura 43 o resultado da simulação, bem como a tensão medida em dois LED’s diferentes, que estão em diferentes estados.

Figura 43 - Simulação das ligações elétricas para *display* de sete segmentos.



Fonte: Autoria própria.

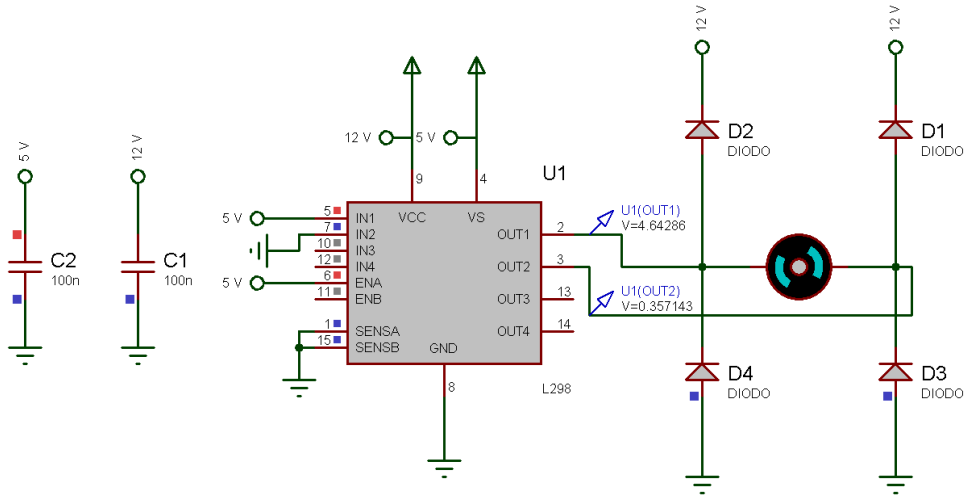
4.1.3 Motor e ponte H

E por fim, foram simuladas as ligações elétricas necessárias para ativar o motor através da ponte H. Na

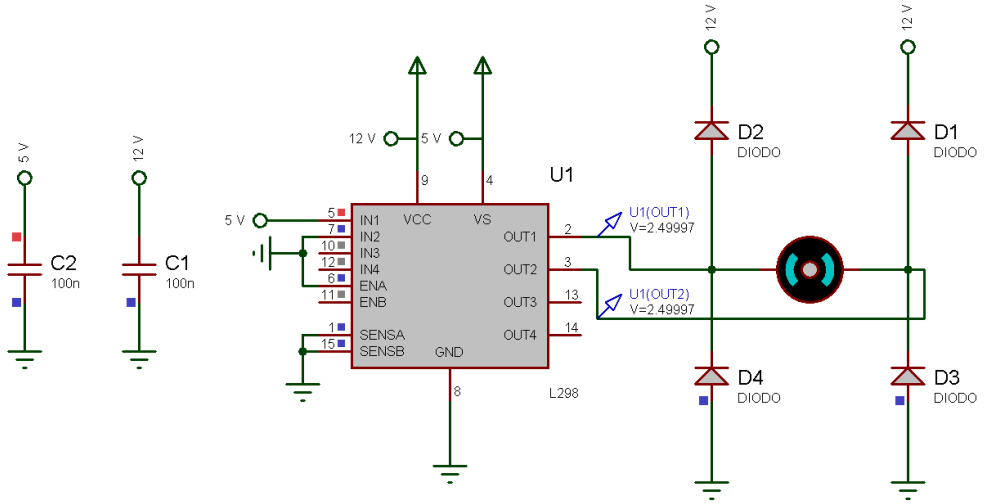
Figura 44 a) é possível observar a queda de tensão sobre o motor DC, que está ativado mediante habilitação da porta “ENA”. E na Figura 44 b) pode-se observar que a porta “ENA” está desabilitada e, portanto, não tem queda de tensão sobre o motor DC.

Figura 44 - Simulação de ligação elétrica do motor DC e ponte H para motor ativado.

a) Motor ativado



b) Motor desativado



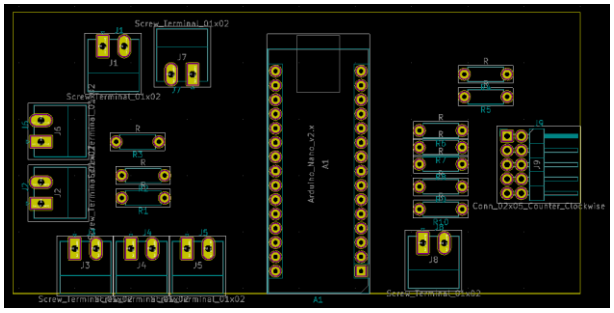
Fonte: Autoria própria.

4.2 PCI (Placa de Circuito Impresso)

Para tornar o projeto físico confiável e estático, foi desenvolvida uma placa de circuito impresso no *software* KiCad, com dimensões de 50 mm x 100 mm. Na Figura 45 é possível ver como foram dispostos os componentes na placa desenvolvida.

Nessa PCI, estão fixados os resistores que são conectados em série com os LED's e os resistores que são conectados em série com os botões *push-button*, além de que a plataforma Arduino Nano também esta acoplada à esta PCI. Portanto, a placa contém vinte e seis pontos de entradas e saídas.

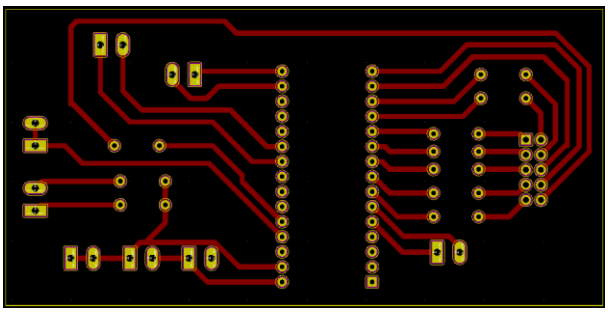
Figura 45 - Posição dos componentes sobre a placa do Arduino.



Fonte: Autoria própria.

Já na Figura 46, é mostrado como foram projetadas as trilhas de cobre da placa. Com o objetivo de diminuir a probabilidade de mau contato nas placas e, por ser um protótipo produzido manualmente, as trilhas foram definidas com 0,9 mm de espessura e os isolamentos entre as trilhas tem 0,3 mm, no mínimo.

Figura 46 - Trilhas de cobre da PCI desenvolvida.

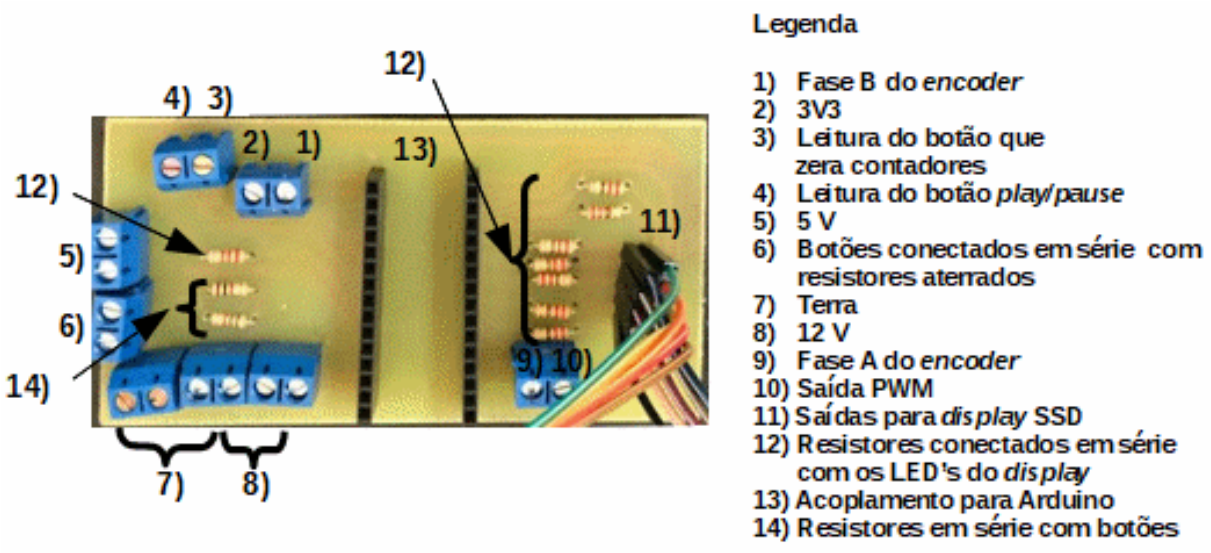


Fonte: Autoria própria.

A Figura 47 mostra a vista superior da PCI produzida e identifica as entradas e saídas, bem como os componentes nela posicionados. Por sua vez, a

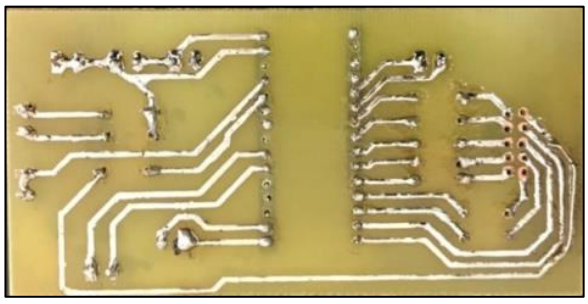
Figura 48 apresenta a vista inferior da placa de circuito impresso desenvolvida, de modo a evidenciar as trilhas e as ilhas de ligações dos componentes.

Figura 47 - Vista superior da PCI produzida.



Fonte: Autoria própria.

Figura 48 - Vista inferior da PCI desenvolvida



Fonte: Autoria própria.

4.3 *Display*

O código desenvolvido para configurar o SSD conforme as especificidades do *hardware* e da biblioteca “*SevSeg*” pode ser visto no APÊNDICE A. Foi necessário configurar o número de dígitos do *display* e os pinos específicos do Arduino que

serão conectados ao *display*. Além disso, foi preciso informar que o *display* é configurado como cátodo comum.

Também foi configurada a intensidade do brilho dos *LED's*, que, pode ser estabelecida dentro de uma escala que varia entre 0 e 100.

Realizadas essas configurações, para alterar o valor exibido no *display*, foi utilizado o comando:

sevseg.setNumber(número_exibido,casas_decimais);

Outro detalhe da biblioteca "*SevSeg*" é que o comando a seguir precisa ser executado frequentemente para "limpar" o *display*:

sevseg.refreshDisplay();

Na Figura 49 é possível observar o SSD ativo formando o numeral "12", através do uso da seguinte linha de código:

sevseg.setNumber(12,0);

.

Figura 49 - Teste das configurações do SSD.

Fonte: Autoria própria.

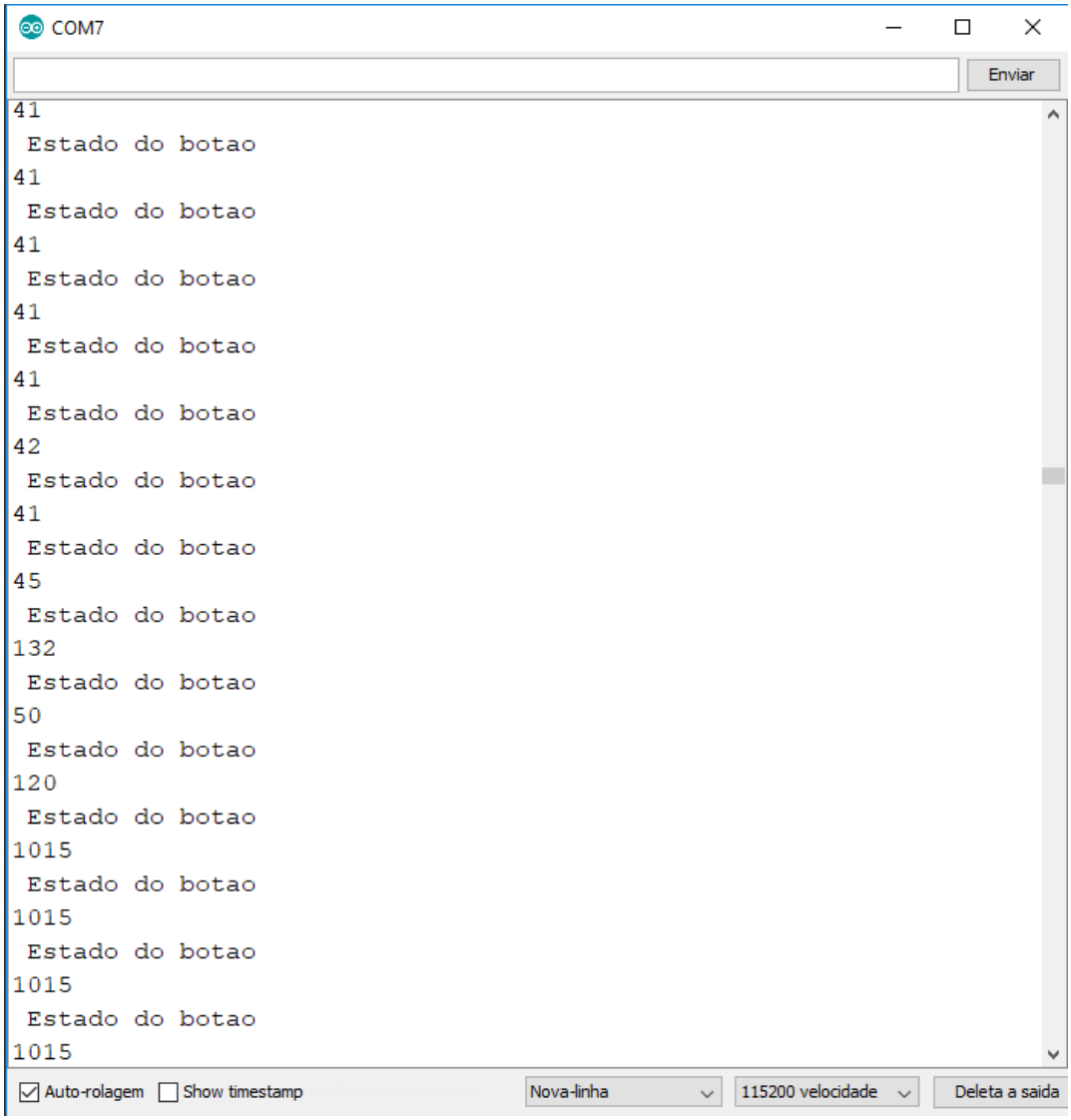
4.4 Botões *push-buttons*

Foi desenvolvido um código para teste de funcionalidade dos botões, que se encontra no APÊNDICE C. O código transmite pela porta serial o estado do pino de leitura do botão. A leitura é transmitida em até 10 bits, ou seja, se a porta de leitura do botão estiver em estado alto o valor transmitido será mais próximo de 1024 e, se estiver em estado baixo o valor transmitido será mais próximo de 0.

Na Figura 50 é possível observar a alteração do dado transmitido, durante uma borda de subida sem tratamento do efeito *debouncing*.

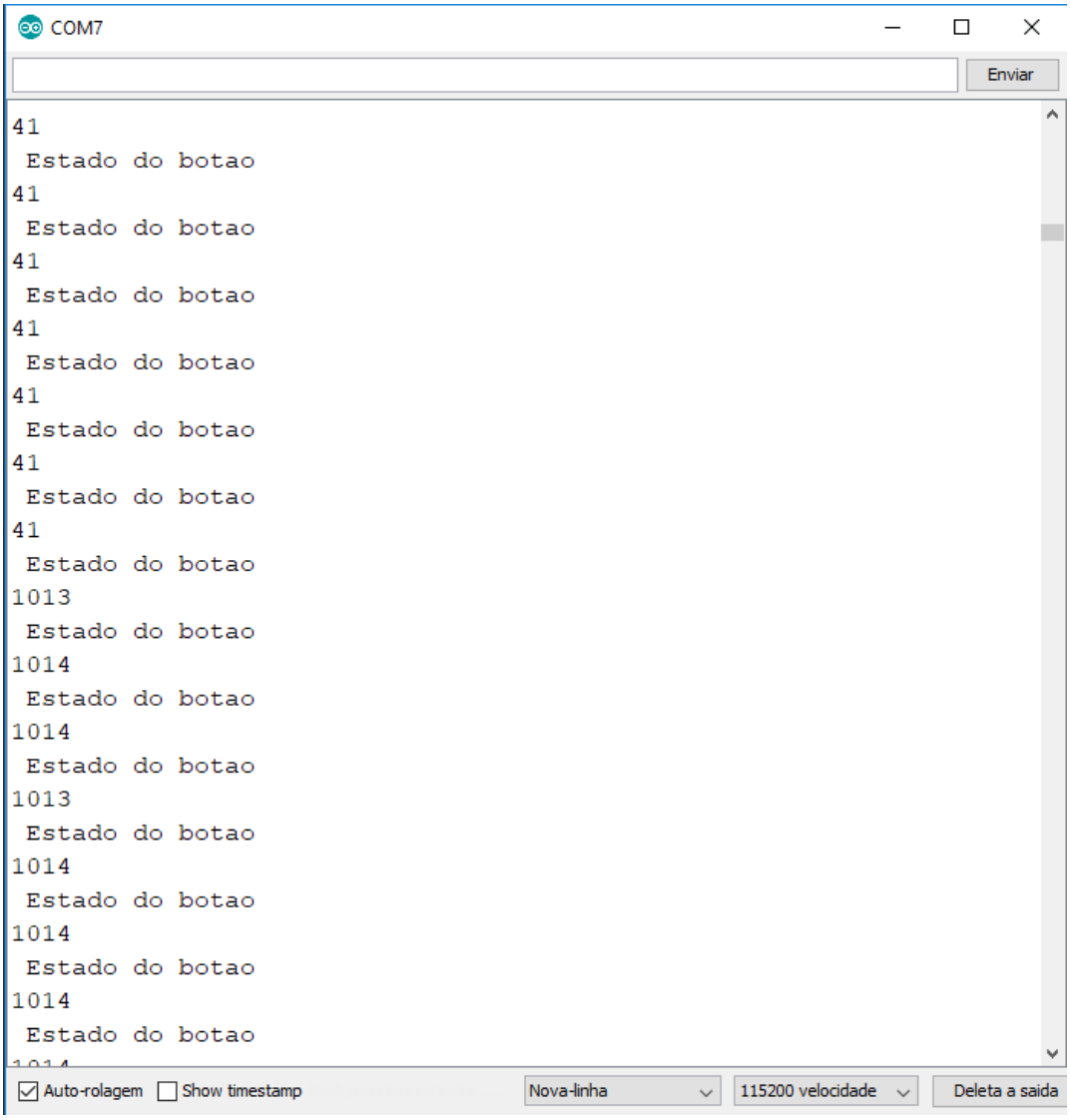
A Figura 51 mostra a transmissão dos dados lidos no pino do botão durante uma borda de subida, sob tratamento do efeito *debouncing.* Observa-se que a alteração dos dados transmitidos é mais abrupta e apresenta menos oscilações, se comparadas com as alterações de leitura apresentadas na Figura 50.

Figura 50 - Leitura do botão transmitida pela porta serial sem tratamento do efeito *deboucing*.



Fonte: Autoria própria.

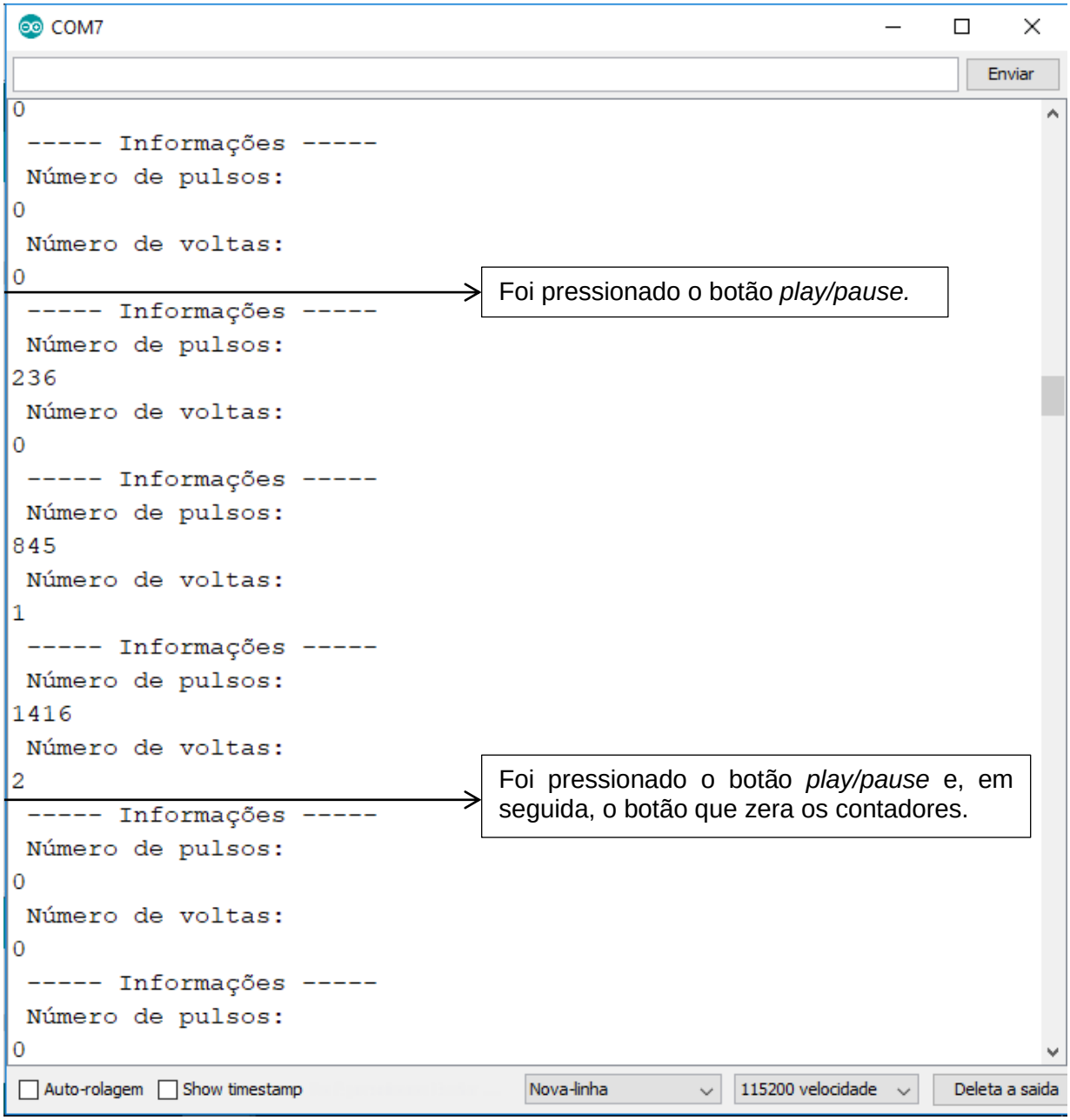
Figura 51 - Leitura do botão transmitida pela porta serial com tratamento do efeito *debounce*.



Fonte: Autoria própria.

Na Figura 52 são apresentados dados transmitidos para a porta serial, com a finalidade de verificar alterações dos dados associados ao pressionamento dos botões, no projeto desenvolvido. Os dados são transmitidos para a porta serial num intervalo de um segundo de diferença entre si. Estão indicados por setas os intervalos que houveram botões pressionados, assim como a identificação de qual botão foi.

Figura 52 - Teste das funcionalidades associadas aos botões.



Fonte: Autoria própria.

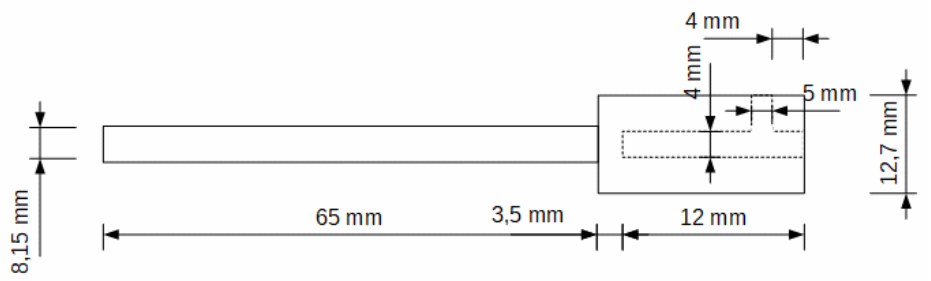
4.5 Eixo para motor DC

Para um acoplamento prático do motor DC foi projetado um eixo que permite a entrada de um parafuso perpendicular. Como o eixo do motor DC é chanfrado, ao rosquear um parafuso no eixo projetado, o motor DC fica encaixado. Ou seja, o eixo do motor DC desacopla facilmente se puxado horizontalmente, mas ao rotacionar, o

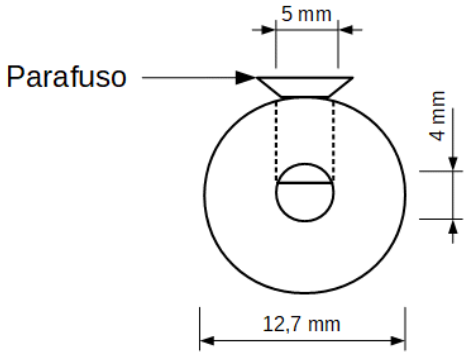
eixo projetado rotacionará junto. A Figura 53 mostra o desenho do eixo projetado para acoplar o motor DC.

Figura 53 - Eixo projetado para acoplar motor DC.

a) Vista lateral



b) Vista frontal

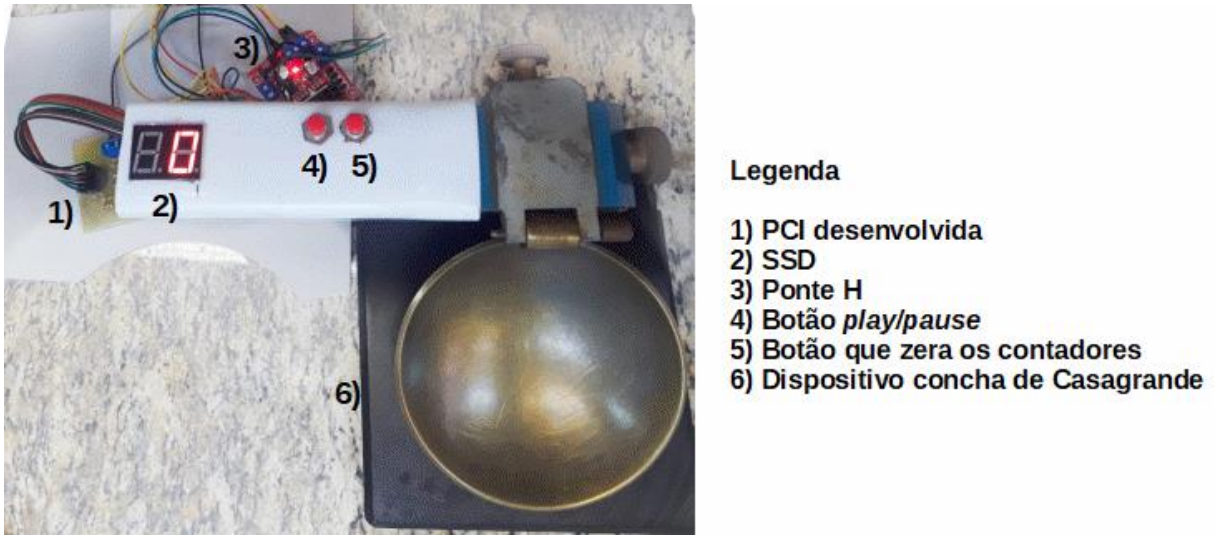


Fonte: Autoria própria.

4.6 Dispositivo concha de Casagrande Automatizado

Na Figura 54 é possível observar como ficou o dispositivo concha de Casagrande com o sistema automático acoplado, com identificação dos componentes exibidos na imagem.

Figura 54 - Dispositivo concha de Casagrande automatizado.



Fonte: Autoria própria.

4.7 Controlador

Foram coletadas 10 amostras do sinal de resposta do sistema c(t) à entrada degrau u(t), para projetar o controlador PI.

A metodologia utilizada para coleta de dados foi descrita no tópico 3.7 e o código construído encontra-se comentado no APÊNCICE A, junto com o código completo, pois não é interessante que essa parte do código esteja ativa durante as execuções do projeto finalizado. Mas pode ser ativada para fins de calibração.

O gráfico apresentado na Figura 55 mostra a curva de tendência das respostas obtidas, bem como o desvio padrão das amostras de c(t). Por sua vez, o gráfico exibido na Figura 56 identifica os parâmetros L e T, necessários para projetar o controlador, na curva de tendência das amostras do sinal c(t).

L = 30 x 10^{-3} s

T = 385 x 10^{-3} s

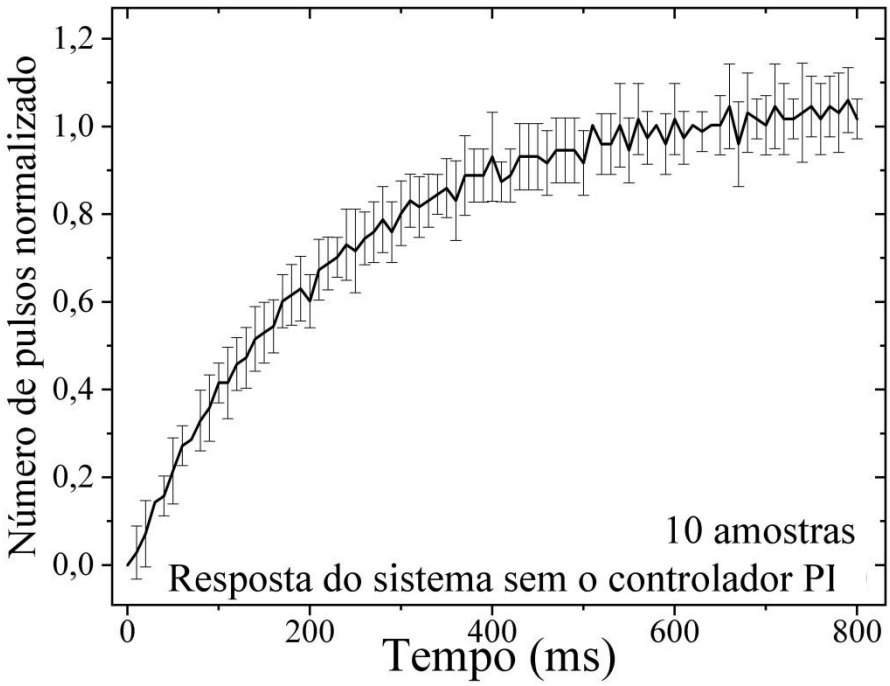
Com os parâmetros T e L é possível obter os parâmetros K_p, T_i e T_d do controlador PI, através do preenchimento do Quadro 1 - Regra de sintonia de Ziegler-Nichols baseada na resposta ao degrau da planta (primeiro método).Quadro 1:

K_p = 11,55

T_i = 0,1 x 10^{-3}

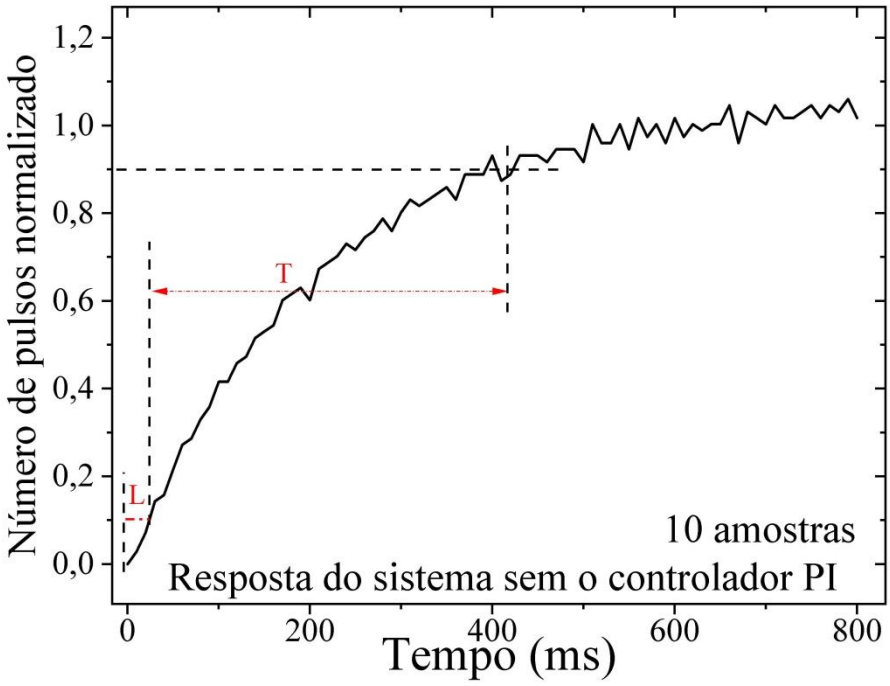
T_d = 0

Figura 55 - Desvio padrão das amostras de resposta do sistema sem o controlador.



Fonte: Autoria própria.

Figura 56 - Identificação dos parâmetros L e T na curva de tendência da resposta do sistema sem o controlador PI.

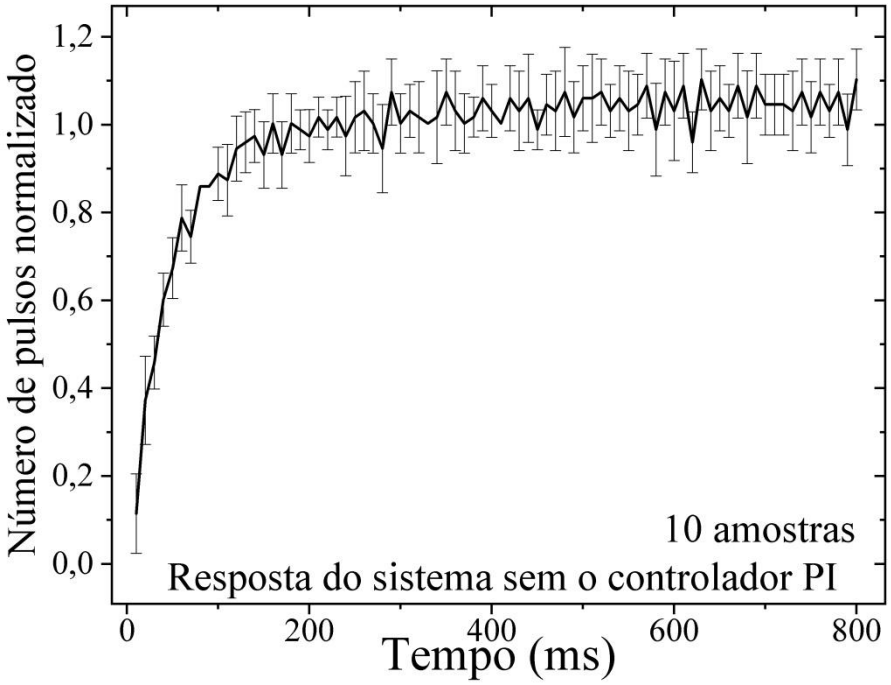


Fonte: Autoria própria.

O código utilizado para implentação do controlador PI pode ser visto no APÊNDICE A.

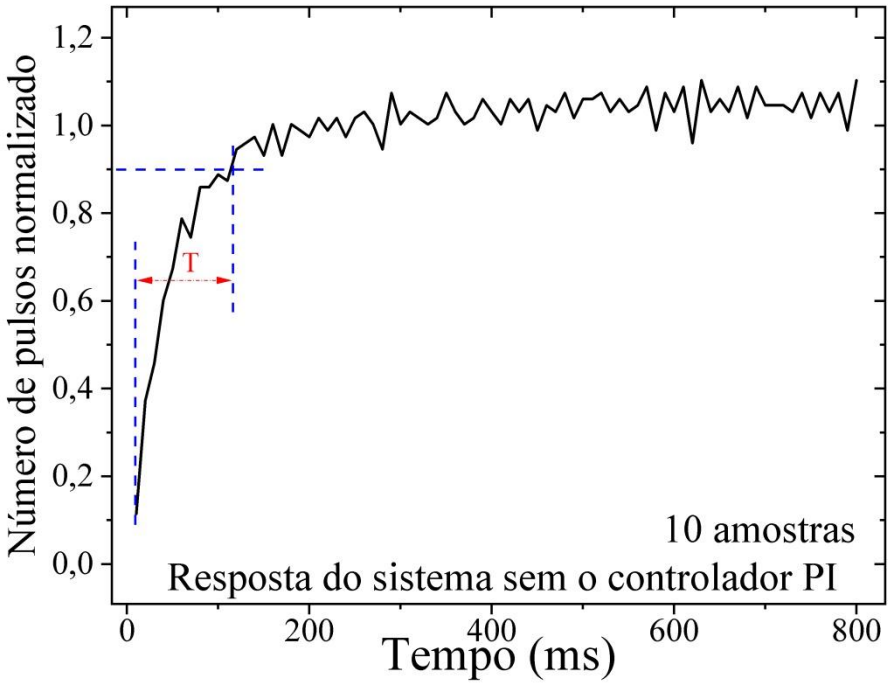
A Figura 57 mostra o desvio padrão de dez amostras do sinal de resposta do sistema c(t), com controlador PI implementado. Observa-se que a curva se aproxima da resposta desejada mais rapidamente, e atinge 90% do valor desejado em 105 ms, conforme é evidenciado na Figura 58..

Figura 57 – Sinal de resposta do sistema c(t) com controlador PI implementado.



Fonte: Autoria própria.

Figura 58 - Curva de tendência da resposta do sistema, com parâmetro T identificado.



Fonte: Autoria própria.

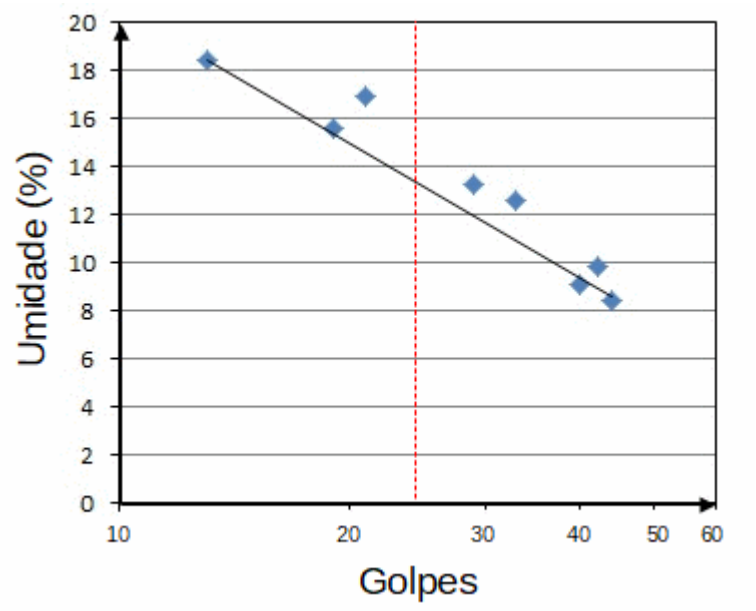
4.8 Validação

Foram realizados dois ensaios simultâneos para obtenção do limite de liquidez de uma amostra de solo, com o objetivo de comparar o resultado obtido com o dispositivo concha de Casagrande manual e com o dispositivo automatizado.

Foi colocada nos gráficos uma linha tracejada em vermelho, perpendicular ao eixo de golpes, no ponto de 25 golpes. A umidade correspondente ao cruzamento da linha de tendência dos dados coletados com a linha tracejada em vermelho é o limite de liquidez do solo.

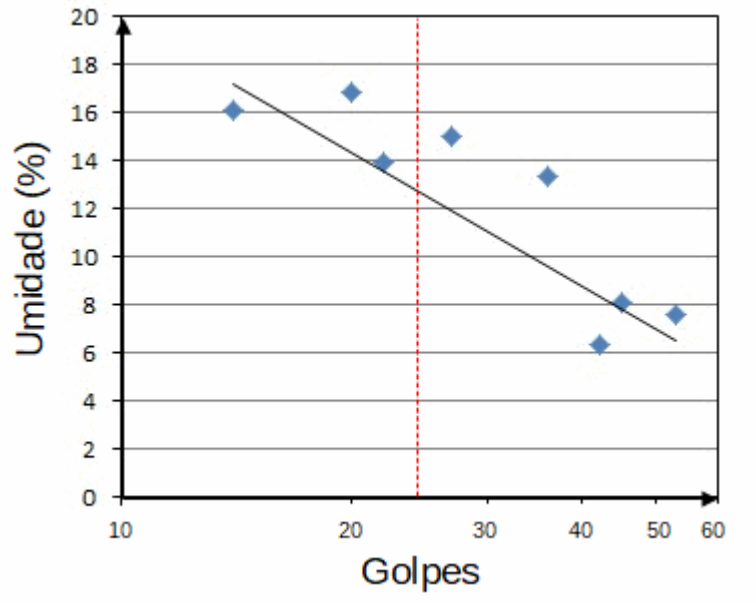
Na Figura 59 podem-se observar os dados obtidos com o dispositivo manual, e sua linha de tendência. Observa-se que o ponto de limite de liquidez do solo encontra-se em 14,6555% de umidade. Enquanto que, na Figura 60, é possível observar os dados obtidos com o sistema automatizado, cujo limite de liquidez foi encontrado em 14,1925% de umidade.

Figura 59 - Dados obtidos em ensaio, com dispositivo concha de Casagrande manual.



Fonte: Autoria própria.

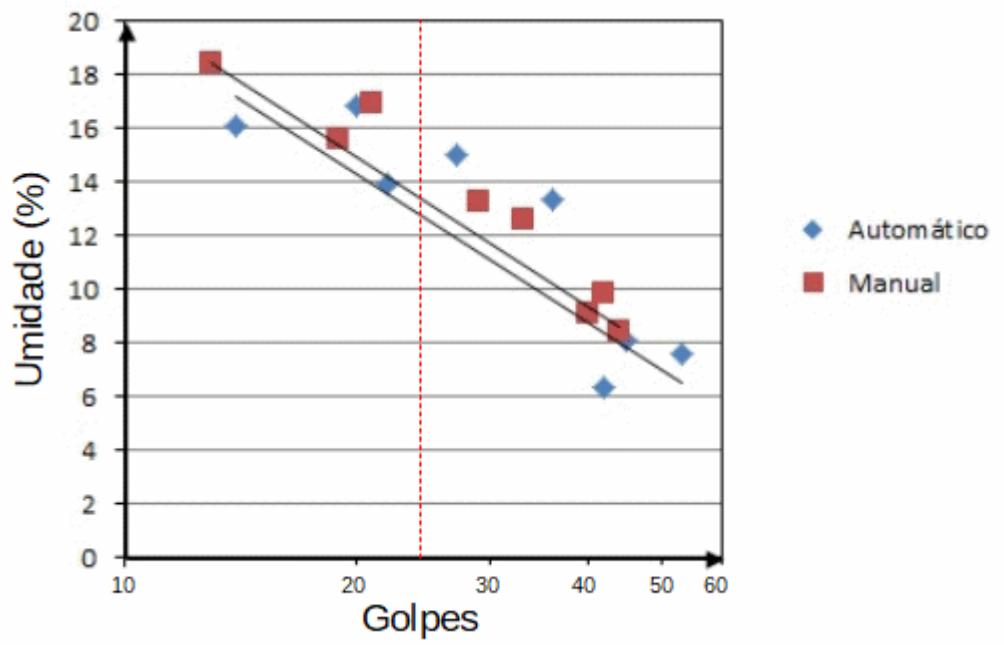
Figura 60 - Dados obtidos em ensaio, com dispositivo concha de Casagrande automatizado.



Fonte: Autoria própria.

Os resultados apresentados na Figura 60 foram obtidos com um sistema de controle PID, que apresentava pouco mais que o dobro de tempo de resposta que o controlador PI projetado no Tópico 4.6. A Figura 61 compara as duas linhas de tendência obtidas com os dados do ensaio realizado.

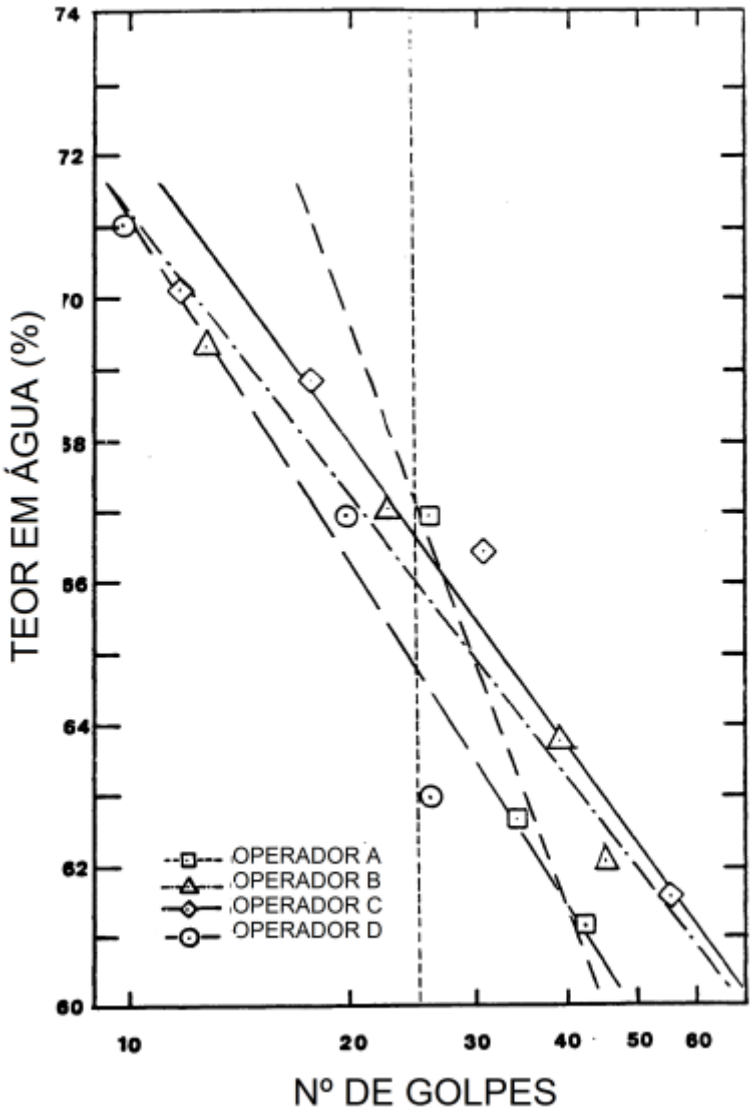
Figura 61 - Comparação entre as linhas de tendência obtidas com o dispositivo manual e com o dispositivo automatizado.



Fonte: Autoria própria.

Para fins de comparação técnica da diferença entre as linhas de tendência obtidas, é apresentado o gráfico que Kestler (1982) comparou resultados obtidos no ensaio de limite de liquidez realizado com o mesmo tipo de amostra de material, em diferentes laboratórios e, portanto, com diferentes operadores, na Figura 62. Nota-se que a diferença encontrada entre os limites de liquidez do solo variou dentro de um intervalo de 2,5% de umidade.

Figura 62 - Influência da técnica do operador da determinação do limite de liquidez através do dispositivo concha de Casagrande.



Fonte: Adaptado de Kestler (1982).

5 CONCLUSÕES

Os métodos de leitura e tratamento de *debouncing* utilizado com os botões se mostraram eficientes e com porcentagem de erro nulo, para o projeto desenvolvido. Bem como o método de leitura do sinal de resposta do sistema c(t), através do sistema *encoder* integrado ao motor DC.

A ponte H associada ao sinal PWM da plataforma Arduino Nano, foram satisfatórias no processo de controle da velocidade angular do eixo do motor DC, pois possibilitaram os ajustes de tensão necessários.

O controlador PI projetado praticamente extinguiu o atraso L do sinal de resposta c(t) e reduziu a constante de tempo T a 105 ms.

O controle de rotação do eixo do motor DC se mostrou eficiente, porém ainda ocorrem oscilações no seguimento do sinal de resposta c(t). E isso se reflete na dispersão dos pontos obtidos no ensaio de limite de liquidez do solo realizado para validação do projeto desenvolvido.

De modo geral, a linha de tendência obtida no ensaio de validação, é satisfatória, baseando-se no estudo realizado por Kestler (1982), pois o ponto do limite de liquidez variou menos de 20% da máxima variação dos resultados apresentados no seu estudo. Todavia, o ensaio foi realizado com um controlado PID, que apresentava maior tempo de assentamento e menos oscilações. Como o ensaio está sujeito a alterações na carga e problemas mecânicos com o acoplamento e alinhamento, o fato de que o tempo de resposta está menor com o controlador PI implementado, deve reduzir a dispersão dos pontos em relação à linha de tendência.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, R. **Leitura de chaves mecânicas e o processo de debounce.** Disponível em: <www.embarcados.com.br/leitura-de-chaves-debounce/>. Acesso em: 29 de maio de 2018.

ANGELO. T. N. **Introdução a Sistemas de Computação Digital**. Disponível em: <www.dca.fee.unicamp.br/~leopini/DISCIPLINAS/EA869/2018-1/k1-interrupcao-geral.pdf> Acesso em: 05 de julho 2019.

ARDUINO. ***AttachInterrupt()*** **[Interrupções externas].** Disponível em: <www.arduino.cc/reference/pt/language/functions/external-interrupts/attachinterrupt/>. Acesso em: 27 de maio de 2019.

ARDUINO. ***Digital Pins*** **[Pinos digitais].** Disponível em: <www.arduino.cc/en/Tutorial/DigitalPins>. Acesso em: 29 de maio de 2019.

ARDUINO. ***PID Library*****.** Disponível em: <playground.arduino.cc/Code/PIDLibrary/>. Acesso em: 01 de junho de 2019.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6459: Solo – Determinação do limite de liquidez**. Rio de Janeiro. 2016.

BALTAZAR, P. M.; ANDRADE, F; DE GOUVEIA, D. C. **Estudo e simulação de controle de torque em sistemas servoacionados.** 2013. 109 f. Trabalho de conclusão de curso - Universidade Tecnológica Federal do Paraná – UTFPR, Curitiba, 2013.

BIANCHI, L. H.; RAMOS. M. S. **Correlações com parâmetros de colapsibilidade de um solo da região de campinas / SP.** 2013. 93 f. Trabalho de conclusão de curso - Universidade Tecnológica Federal do Paraná – UTFPR, Curitiba, 2013.

BOYLESTAD, R. L. **Introdução à Análise de Circuitos**. 10ª ed. São Paulo: Pearson. 848 p.

CAPUTO, Homero Pinto. **Mecânica dos Solos e Suas Aplicações.** Rio de Janeiro: LTC, 6ª edição, 1996**.**

GHIZONI, G. R. S. **Fabricação de pastilhas cerâmicas com lodo proveniente de estações de tratamento de água – estudo de caso.** 2013. 75 f. Trabalho de conclusão de curso - Universidade Tecnológica Federal do Paraná – UTFPR, Pato Branco, 2013.

GREENSTED, A. ***Switch Deboucing***. Disponível em: <http://www.labbookpages.co.uk/electronics/debounce.html> Acesso em: 05 de julho 2019.

KESTLER, M. A. (1982). ***"Correlations and comparisons between the casagrande liquid limit device and the fall Cone".*** M. S. Thesis, Massachusetts Institute of Technology.

NR, Norma Regulamentadora Ministério do Trabalho e Emprego. **NR-12 - Máquinas e Equipamentos**. 2009

OGATA, K. **Engenharia de Controle Moderno**. 5ª. ed. Rio de Janeiro: Prentice-Hall, v. Único, 2010.

ROBOCORE TECNOLOGIA LTDA. **Display de 7 segmentos**. Disponível em: <www.robocore.net/loja/produtos/display-7-segmentos.html> Acesso em: 02 de outubro 2018.

RODRIGUES, T. P. **Física: Persistência da visão**. Disponível em: <www1.folha.uol.com.br/folha/educacao/ult305u13406.shtml> Acesso em: 05 de julho 2019.

SILVEIRA, C. B. **O que é PWM e Para que Serve?.** Disponível em: <www.citisystems.com.br/pwm/>. Acesso em: 05 de julho de 2019.

SOARES, M. J. **MICROCONTROLADORES PIC – TEORIA - PARTE 3 INTERRUPÇÕES - O QUE É ISSO?.** Disponível em: <www.arnerobotics.com.br/eletronica/Microcontrolador_PIC_teoria_3.htm>. Acesso em: 29 de maio de 2019.

SOUZA, P. M. L. P. **Limite de Liquidez – Correlações e Comparações entre os métodos de Fall Cone e da Concha de Casagrande.** 2011. Tese de mestrado - Universidade Nova de Lisboa, Portugal, 2011.

SPARKFUN START SOMETHING. ***Momentary Pushbutton Switch***. Disponível em: <www.sparkfun.com/products/9190> Acesso em: 02 de outubro 2018.

ZUCATELLI, F. H. G.; OLIVEIRA. M. A. V. **Controle de Servomotores CC.** 2007. 24 f. Artigo Científico – Faculdade de Tecnologia Termomecânica, São Bernardo do Campo, 2007.

APÊNDICE A - CÓDIGO PARA O ARDUINO COM CONTROLADOR PI

```
/*
Projeto de Automacao do ensaio do limite de liquidez do solo
para finalizacao do curso de engenharia eletronica - UTFPR

Primeiro semestre de 2019
Autora: Cristiane Barbosa Prado
*/

//=====================================================================
//
// --- Bibliotecas extras adicionadas ---
//
//=====================================================================

//Biblioteca do display de 7 segmentos
#include "SevSeg.h"
SevSeg sevseg;

//Biblioteca PID
#include "PID_v1.h"

//=====================================================================
//
// --- Mapeamento de Hardware ---
//
//=====================================================================

//Definição de pinos para debounce
#define           ZERA_SSD          16
#define           PLAY_PAUSE        17

#define           tempo_de_delay    50

//Pinos para motor
#define           EN                3

//Encoder
#define           encoder_C1        2
#define           encoder_C2        13

// ====================================================================
//
// ---Definição de constantes ---
//
// ====================================================================

//Constante de tempo
#define           um_segundo        1000
#define           t_captacao        10

//Resoluçao do encoder pulsos/volta
#define           pulsos_por_volta  516

//Entrada r(t)
#define degrau              104

//Número de voltas máxima
#define voltas_maxima       60

//=====================================================================
//
```

```
// --- Variáveis Globais ---
//
//=====================================================================

//Flags dos botões
flag_ZERA_SSD = LOW,
flag_PLAY_PAUSE = LOW;

//Variáveis de controle do motor
long              pulse_number = 0;
int                     voltas=0;

//Variáveis do Controlador
double            Setpoint, Input, Output;
long              aux_pulse_number = 0;

//=====================================================================
//
// --- Inicialização myPID ---
//
//=====================================================================

PID myPID(&Input, &Output, &Setpoint,11.55,0.1,0, DIRECT);

//=====================================================================
//
// --- Funções ---
//
//=====================================================================

void  motor_ON();             //função para controle do motor
void  debounce_PLAY_PAUSE();  //detecçao do botao PLAY_PAUSE
void  debounce_ZERA_SSD();    //detecçao do botao que zera o SSD


//=====================================================================
//
// --- Configurações e inicializações ---
//
//=====================================================================

void setup() {

  //Definição das entradas e saidas para debouce
      pinMode(ZERA_SSD, INPUT_PULLUP);    //Resistencia interna
      pinMode(PLAY_PAUSE, INPUT_PULLUP); //Resistencia interna

      // Serial.begin(115200);      //Inicializa comunicação serial

      //Pinos do motor
      pinMode(EN, OUTPUT);          //Saída PWM para habilitar
      pinMode(encoder_C1,  INPUT); //Configura     entrada     C1     para
      //leitura do encoder
      pinMode(encoder_C2,  INPUT); //Configura entrada C2 para //leitura do
      encoder

      //Interrupção externa por borda de subida
      attachInterrupt(digitalPinToInterrupt(encoder_C1),       count_pulses,
      RISING);

      //Configurações para display de 7 segmentos
```

```
      byte numDigits = 2;     //Display de 2 dígitos
      byte digitPins[] = {11,12}; //Pinos do Arduino conectados
                                  //aos dígitos do display
      byte segmentPins[] = {4,5,6,7,8,9,10,21};//Pinos dos Arduino

      //Conectados aos LED's A-DP
      byte hardwareConfig = COMMON_CATHODE;    //Display //Configurado como
                                                          //Cátodo
comum
      sevseg.begin(hardwareConfig,   numDigits,   digitPins,   segmentPins);
      //Começa o display passando os parâmetros de hardware
      sevseg.setBrightness(90); //Brilho de 0 a 100
      sevseg.setNumber(voltas,0);//Inicia display mostrando 00

      //Configurações do Controlador
      Setpoint = 10.45; // r(t) ajustado
      myPID.SetMode(AUTOMATIC);//Configuração para modo automatico
      myPID.SetSampleTime(10); // Computa controlador a cada 10 ms

      Input = aux_pulse_number; //Inicializa entrada
}

void loop() {

      //Verifica se botões foram pressionados
      debounce_PLAY_PAUSE();
      debounce_ZERA_SSD();

      //Caso botão PLAY_PAUSE tenha sido apertado
      if(flag_PLAY_PAUSE == HIGH){

            motor_ON();       //Liga motor

      }else analogWrite(EN,LOW); //Caso botão PLAY_PAUSE não tenha
                                 //sido apertado

      //Caso botao ZERA_SSD apertado
      if(flag_ZERA_SSD == HIGH){

            //Abaixa flag que será tratada
            flag_ZERA_SSD=LOW;

            //Reinicia contagem caso motor desativado
            if(flag_PLAY_PAUSE == LOW){

                  //Zera variáveis
                  voltas = 0;
                  pulse_number = 0;

                  //Zera display
                  sevseg.setNumber(voltas,0);
            }
      }
            sevseg.refreshDisplay();// Deve ocorrer repetidamente
}
 //================================================================
//
// --- Função de controle do motor ---
//
//================================================================
```

```
void motor_ON(){

//-------------------------------------------------------
  //Captação de informações para o controlador
      static unsigned long    guarda_millis_Input = millis();

      //Verificação de pulse_number a cada 10 mS segundos
      if((millis() - guarda_millis_Input)>= t_captacao){

        //reinicia contador
        guarda_millis_Input = millis();

        //captação de amostra
        Input = pulse_number - aux_pulse_number;
        aux_pulse_number=pulse_number;
      }
//-------------------------------------------------------

      myPID.Compute();//Computa controlador

      //variaveis PWM
      int TENSAO = map (degrau, 0,500,0,255);//Função que
                                             //mapeia a variável
                                             //e converte para resolução
                                             //da saída PWM
                                             //de 2^8 (8 bits)

      //u(t) aplicado na planta
      analogWrite(EN,(TENSAO+Output));

      //Variável auxiliar de contagem de voltas
      //que armazena o valor do último pulso que completou
      //a volta anterior
      static int aux_voltas;

      //Teste de controle das voltas
      if((pulse_number % pulsos_por_volta) == 0
      && aux_voltas != pulse_number
      && pulse_number>0 && voltas < voltas_maxima)  {

            //Incrementa e mostra no SSD variavel voltas
            voltas = voltas + 1;
            sevseg.setNumber(voltas,0);

            //Armazena pulse_number
            aux_voltas = pulse_number;

      }

      //Limitação de voltas
      if (voltas>= voltas_maxima){

      sevseg.setNumber(voltas,0); //Imprime voltas máxima
      pulse_number = 0;             //Zera pulse_number
      analogWrite(EN,LOW);          //Desliga motor
      flag_PLAY_PAUSE = LOW;  //Abaixa flag do botão

    }
//=================================================================
//
// --- Usar para conferência, em caso de calibração ---
```

```
//
//=====================================================================

//-------------------------------------------------------------
//     Confere variáveis a cada segundo
//-------------------------------------------------------------
/*    //Controle do tempo
      static unsigned long guarda_millis_encoder = millis();

      //Verificação do rpm a cada 1 segundo
      if((millis() - guarda_millis_encoder)>= um_segundo)
      {
            //reinicia temporizador
            guarda_millis_encoder   =     millis();

            Serial.println(" Informações de 1 segundo: ");

            //Imprime pulsos no monitor serial
            Serial.println(" Pulsos: ");
            Serial.println(pulse_number);

            //Imprime voltas no monitor serial
            Serial.println(" Voltas: ");
            Serial.println(voltas);

      }*/

//-------------------------------------------------------------------
// --- Confere variáveis a cada 10 ms segundo ---
//ATENÇÃO: Ativar essa função prejudica o funcionamento do programa.
//Ativar apenas para verificar o tempo de resposta.
//-------------------------------------------------------------------

      //Captação de informações para o controlador
/*    static unsigned long guarda_millis_controlador = millis();

      //Verificação do rpm a cada 1 segundo
      if((millis() - guarda_millis_controlador)>= t_captacao)
      {
            //reinicia temporizador
            guarda_millis_controlador     =     millis();

            Serial.println(" Informações de 1/100 de segundo: ");

            //Imprime pulsos no monitor serial
            Serial.println(" Pulsos: ");
            Serial.println(pulse_number);

      }*/

}//end motor_ON


//=====================================================================
//
// --- Função de detecção do botao PLAY_PAUSE ---
//
//=====================================================================


void  debounce_PLAY_PAUSE(){
```

```
  //Variáveis para debounce
  static int last_state = HIGH;
  static int  state;
  int reading = digitalRead(PLAY_PAUSE);       //guarda leitura do botão A3
  static unsigned long guarda_millis = millis();//guarda tempo inicial

  if(reading!=last_state){ //testa se botao mudou de estado

    //inicia contador para debounce
    guarda_millis=millis();

  }

  //Primeiro degrau
  if((millis()- guarda_millis)>tempo_de_delay){ //testa se deu o
                       //tempo de delay de 50 milisegundos

      //Caso verdade
      if(reading!=state) {//Testa se botão mudou
                     //Caso verdade
      state=reading;  //Estado recebe leitura

      if(state==HIGH){//Testa se foi borda de subida

        //Condição para botão pressionado
            flag_PLAY_PAUSE = !flag_PLAY_PAUSE;

      }
    }
  }

  last_state=reading;//Último estado recebe leitura mais recente

}//end debounce_PLAY_PAUSE

//=====================================================================
//
// --- Função de detecção do botao ZERA_SSD ---
//
//=====================================================================

void debounce_ZERA_SSD(){

    //Variáveis para debounce
  static int last_state = HIGH;
  static int  state;
  int reading = digitalRead(ZERA_SSD); //Guarda leitura do botão A2
  static unsigned long guarda_millis = millis();//Guarda tempo
                                          //inicial

  if(reading!=last_state){ //Testa se botão mudou de estado

      //Inicia contador para debounce
    guarda_millis=millis();
  }

   //Primeiro degrau
  if((millis()- guarda_millis)>tempo_de_delay){//Testa se deu o
                 //tempo de delay de 50 milisegundos
```

```
      //Caso verdade
    if(reading!=state)         //Testa se botao mudou
    {                          //Caso verdade
      state=reading;         //Estado recebe leitura

      if(state==HIGH){//Testa se foi borda de subida

            //Condição para botão pressionado
            flag_ZERA_SSD = HIGH;

      }
    }
  }

  last_state=reading; //Último estado recebe leitura mais recente

}//end debounce_ZERA_SSD


//=====================================================================
//
// --- Função contadora de pulsos do encoder ---
//
//=====================================================================

void count_pulses()
{

     pulse_number++;//Incrementa número de pulsos
} //end count_pulses
```

APÊNDICE B - CÓDIGO PARA CONFIGURAR SAÍDA PWM DO ARDUINO

```
void verifica_pulsos(){

    //variaveis PWM
    static float PORCENTAGEM = PER; //Constante de entrada dada
                                    //em porcentagem, referente
                                    //a saída do Arduino (5 V)


    int TENSAO = map (PORCENTAGEM, 0,100,0,255);//Função que
                                                //mapeia a variavel
                                                //e converte para resolução
                                                //da saída PWM
                                                //de 2^8 (8 bits)

    //Valor da tensão, já mapeado, na saída PWM (D3)
    analogWrite(EN,TENSAO);
```

APÊNDICE C - CÓDIGO PARA TESTE DE FUNCIONALIDADE DOS BOTÕES

```
//Teste de botões
int teste_botao = analogRead(ZERA_SSD);
//Imprime estado do botão no monitor serial
Serial.println(" Estado do botao ");
Serial.println(teste_botao);
```