LEONARDO FONCESCA FINARDI

Projeto e construção de um protótipo de *dip coating* utilizando a plataforma de prototipagem Arduino

Santos

2019

LEONARDO FONCESCA FINARDI

Projeto e construção de um protótipo de *dip coating* utilizando a plataforma de prototipagem Arduino

Monografia apresentada à Escola Politécnica da Universidade de São Paulo como requisito parcial para obtenção do título de Bacharel em Engenharia de Petróleo

Área de concentração: eletrônica/programação

Orientador: Prof. Dr. Jean Vicente Ferrari

Santos

2019

Catalogação-na-publicação

Finardi, Leonardo
Projeto e construção de um protótipo de dip coating utilizando a plataforma de prototipagem Arduino / L. Finardi -- São Paulo, 2019.
46 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Minas e de Petróleo.

1.Dip coating 2.Arduino 3.Servo motor 4.Programação I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Minas e de Petróleo II.t.

Santos

2019

RESUMO

O *dip coating* é um processo que consiste na aplicação de revestimentos finos em sólidos de diversas geometrias por meio da realização de ciclos de mergulho e evaporação em soluções específicas preparadas previamente, as quais após a secagem manifestam sua natureza química diferente na camada aplicada, permitindo a manipulação das propriedades superficiais do sólido de acordo com o fluido usado para o revestimento. O presente trabalho tem como objetivo tanto desenvolver um projeto de um equipamento de *dip coating* econômico quanto de construir um protótipo a fim de comprovar a funcionalidade do projeto. Para isso, é utilizada a plataforma de *hardware* livre Arduino para a prototipagem e programação do equipamento, bem como um Servo motor de rotação contínua para a realização da carga mecânica necessária. São abordadas as teorias fundamentais para o entendimento e descrição do processo, os motivos por trás da escolha das peças (bem como suas aplicações e funcionalidades), a metodologia das ligações eletrônicas, a programação necessária no *Arduino* e o processo de calibragem de velocidades. O código desenvolvido em conjunto com as calibrações permitiram produzir um equipamento de *dip coating* que opera com precisão em velocidades de 10 a 40 milímetros por segundo.

Palavras-Chave: *Dip coating*. Arduino. Servo motor.

ABSTRACT

Dip coating is a process that consists on the controlled application of thin coating films in solids of various geometries through the performing of dip and evaporation cycles in an preprepared specific solution, which after drying manifests the different chemical nature in the applied coat, allowing the manipulation of the superficial properties from the solid according to the fluid used for the coating. The objectives of the work are develop a project of an economic dip coating machine and construct a prototype to prove the project functionality. For this it will be used the free hardware platform Arduino' for the prototyping and programming and a continuous rotation 'Servo motor' for perform the mechanical load necessary. It's described the fundamental theory for the understanding and description of the process, the motivation beyond the parts definition (as well as the applications and functionalities), the methodology of the eletronic disposition, the necessary programming in the Arduino and the process of velocity calibration. The code developed with the velocity calibration enabled to produce a dip coating unit that operate with precision in velocities from 10 to 40 millimeters per second.

Keywords: Dip coating. Arduino. Servo motor.

SUMÁRIO

1 INTRODUÇÃO

O processo de *dip coating*, ou revestimento por imersão, é um dos métodos utilizados atualmente para a aplicação de revestimentos finos em sólidos. Os revestimentos possuem funcionalidades ligadas diretamente à natureza química do material utilizado. Serve como uma alternativa a outros métodos de aplicações de películas finas pois possui uma maior precisão de espessura aplicada, o que deve ser controlado por exemplo a fim da redução de gastos e de desperdício de material. Nos casos em que o revestimento possui um valor monetário significativo ou que se produza em larga escala, por exemplo, se justifica o uso do equipamento.

Filmes finos são lâminas delgadas que, após aplicadas em sólidos, possuem espessuras que variam de nanômetros até micrômetros. Tais películas finas são de extrema utilidade em casos que o material de revestimento possui um valor monetário considerável.

Existem outras técnicas de aplicação de filmes finos, que são mais caras, como PVD[1] (vaporização do material de revestimento e condensação sobre a superfície do substrato para a formação do filme sólido) e CVD[2] (o filme é depositado a partir de reações químicas entre o vapor do revestimento e o substrato).

O *dip coating* é um processo de deposição molhada de filmes finos e é um dos métodos mais antigos aplicados comercialmente de forma eficaz. Os primeiros registros do uso do *dip coating* são da produção de velas, quando o pavio era repetidamente mergulhado em uma resina feita predominantemente de sebo animal e cera, até que se atingisse a espessura desejada.

Hoje em dia, a técnica é utilizada por diversas áreas da ciência, com diferentes aplicações. Alguns exemplos são:

- Em cerâmicas, o revestimento permite a compatibilização entre a superfície da peça e a cobertura final de esmalte.

[1] *Physical vapor deposition*
[2] *Chemical Vapour Deposition*

- Em Placas de Circuito Impresso (placas de eletrônicos para a ligação de fios) o revestimento aumenta a vida útil e a proteção, bem como permite isolamentos elétricos e previne corrosões.

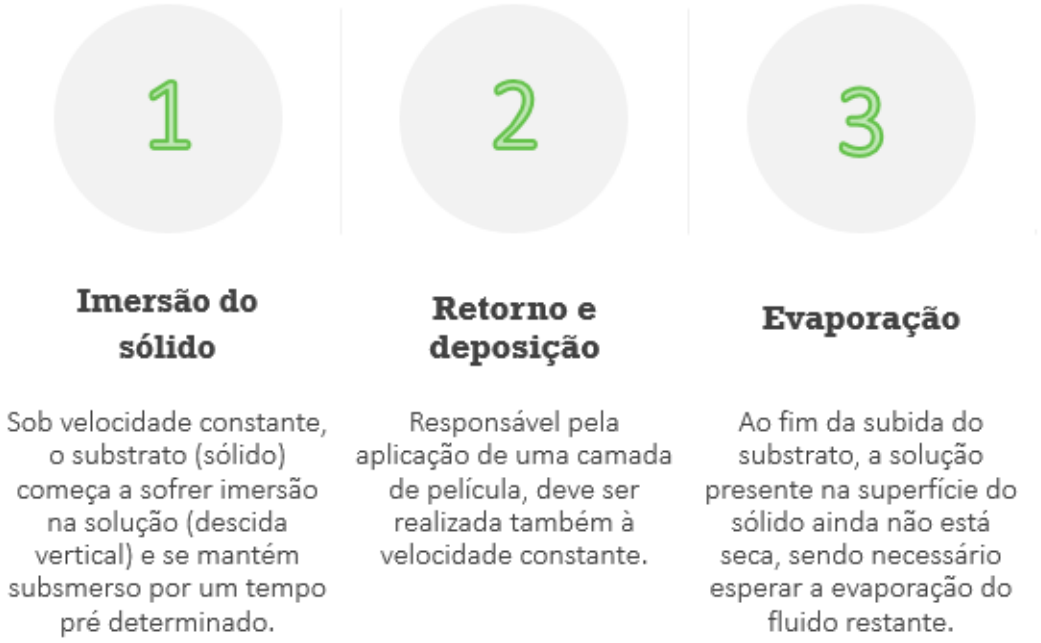
- No processo de galvanização por imersão a quente, o *dip coating* é uma alternativa para a deposição do metal (em estado fundido) sobre a superfície do substrato.

- Na indústria de tintas e vernizes, permite uma maior facilidade no revestimento de superfícies com formatos complexos.

- Para a aplicação de filmes finos sobre substratos metálicos para proteção contra a corrosão.

Pode-se resumir o processo de *dip coating* em 3 etapas principais, ilustradas na Figura 1.

Figura 1 – Etapas do processo de *dip coating.*



Fonte: Próprio autor.

A técnica de revestimento por imersão fornece filmes de qualidade e uniformes que podem ser aplicados inclusive em superfícies não homogêneas, com formatos complexos. O processo *dip coating* é então repetido até que se alcance a espessura de revestimento desejado.

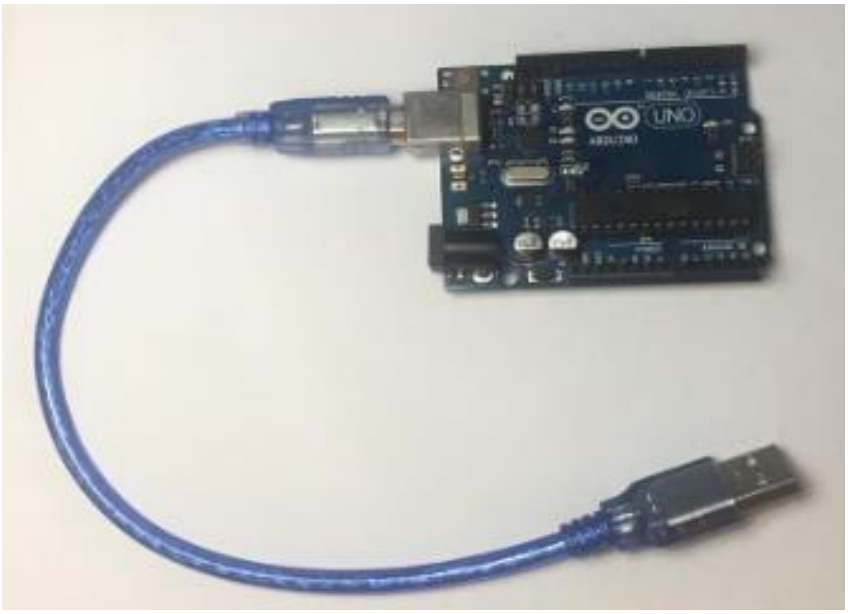
A primeira patente baseada no processo de dip coating foi criada em 1939 para a aplicação de filmes de sílica (GEFFCKEN e BERGER, 1939 apud BRINKER e HURD, 1994). Hoje em dia, revestimentos por sol-gel são estudados para uma vasta gama de aplicações, incluindo revestimentos de proteção, camadas apassivadoras, ferroelétricas, sensores e membranas (BRINKER e HURD, 1994).

Como o equipamento descrito pode ser construído de diferentes formas, o foco do trabalho será tanto para o desenvolvimento de um projeto de *dip coating* de baixo custo quanto para a construção de um protótipo representativo.

Para a construção do protótipo será utilizada uma placa Arduino UNO R3, ilustrada na Figura 2. O Arduino foi criado em 2005 como um *hardware* livre. A placa é constituída por um micro controlador Amtel e circuitos de entrada/saída. O dispositivo possui saídas PWM[3] (*Pulse Widht Modulation*), analógicas e digitais.

A placa Arduino é baseada num microcontrolador muito versátil que potencializa suas funções para além de uma simples interface passiva de aquisição de dados, podendo operar sozinha no controle de vários dispositivos e tendo assim aplicações em instrumentação embarcada e robótica (SOUZA et al, 2011).

Figura 2 – Placa Arduino UNO R3.



Fonte: Próprio autor.

[3] O PWM é uma técnica utilizada para obter resultados analógicos por meio de entradas digitais (geração de onda quadrada). As saídas PWM do Arduino UNO, figura 2, são as 3, 5, 6, 9, 10, 11. Elas se diferenciam das saídas digitais normais pois possuem um ~ antes do número do pino.

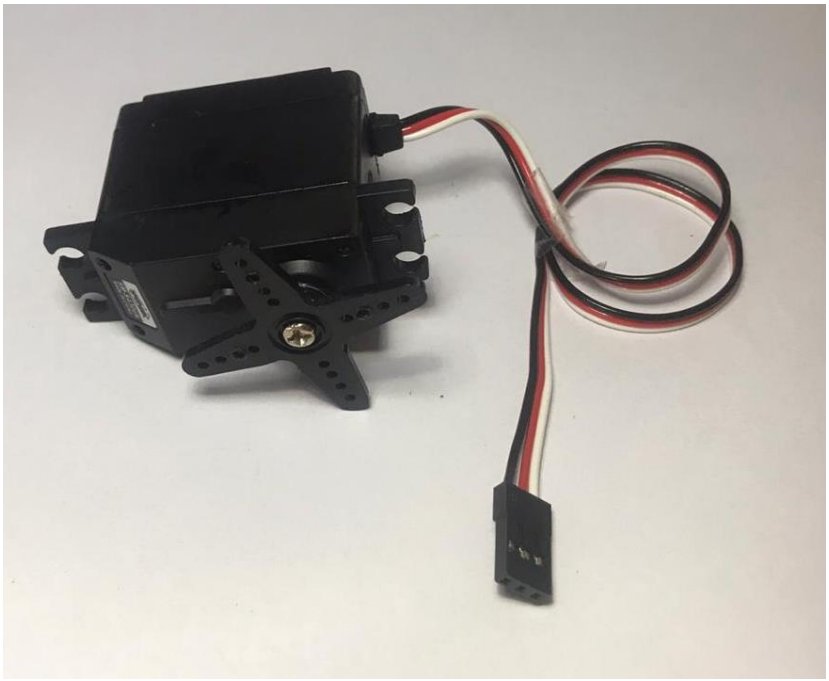
O Arduino é gerenciado por meio de programação, feita através de um computador, que será utilizado tanto como fonte de energia para a placa quanto como interface entre o usuário e o equipamento.

Em conjunto com o Arduino é necessário um motor, que é responsável pela movimentação vertical do substrato (sólido) a ser revestido. Assim, além de ser necessário escolher um tipo de motor compatível com a placa do Arduino, é necessário também desenvolver uma interface na qual o torque do motor consiga resultar na elevação do substrato.

O processo de seleção do tipo de motor pode ser subdividido em duas partes. A primeira é achar todos os motores capazes de lidar com o carregamento desejado e a segunda é, com os possíveis motores selecionados, escolher aquele que satisfaz melhor algum critério adicional (preço, peso, volume, etc) (VAN DE STRAETE et al, 1998).

Visto a finalidade do trabalho, será utilizado o Servo Motor da Figura 3. Esse dispositivo é basicamente um motor que possui o controle angular por meio de um sinal PWM. Além de possuir um preço de mercado compatível com o projeto, é um atuador eletromecânico que gira e determina a posição angular do rotor.

Figura 3 – Kit Servo Motor SM-S4306R**.**



Fonte: Próprio autor.

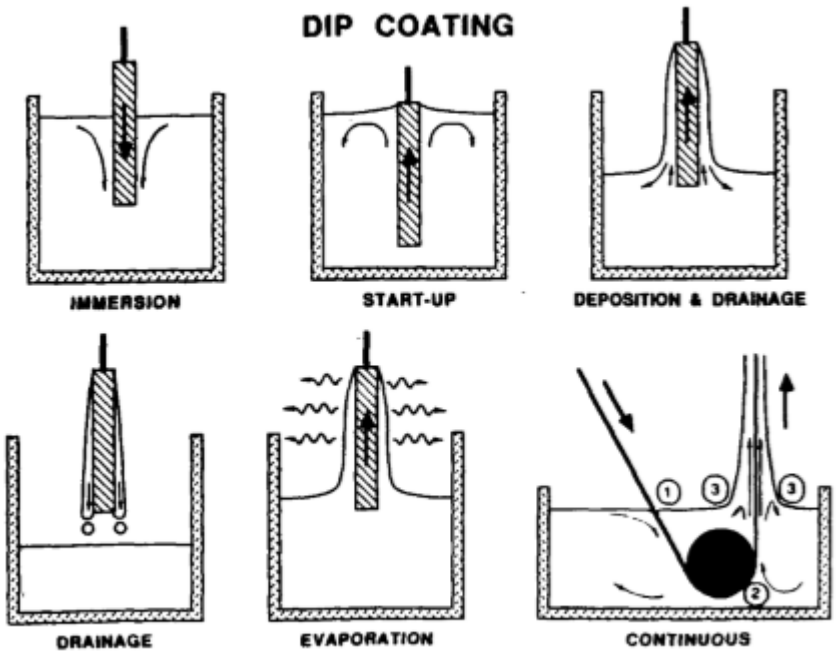
Assim, os objetivos do trabalho são tanto de criar um projeto de equipamento de *dip coating* econômico quanto de construir um protótipo representativo, a fim de validar a eficiência do projeto.

2 REVISÃO DA LITERATURA

2.1 *Dip Coating*

O revestimento por imersão é uma maneira simples e antiga de deposição em substratos de filmes líquidos finos e uniformes que por solidificação geram o revestimento. O fluxo básico é laminar e a espessura do filme é definida pela competição entre as forças viscosas, forças capilares (tensão superficial) e gravidade. Quanto mais rápido o substrato é submergido, mais espesso se torna o filme depositado. Isso pode ser compensado pelo uso de soluções voláteis e processos rápidos de secagem (SCRIVEN, 1988). O processo é ilustrado na Figura 4.

Figura 4 – estágios do dip coating convencional e contínuo.



Fonte: **SCRIVEN, 1988.**

O revestimento por imersão é um método simples para a criação de grandes áreas de arranjo líquido microestrutural para aplicações envolvendo análises químicas e sintéticas, ensaios bioquímicos ou *wet printing* de líquidos poliméricos ou de tintas. A espessura máxima do filme depositado depende criticamente da velocidade de mergulho, bem como do tamanho, geometria e orientação angular do substrato (DARHUBER et al, 2000).

Quanto mais lenta é a submersão do substrato, menor é a força viscosa de arrasto (pela qual o líquido é elevado), e então mais fino será o filme depositado. Esse comportamento atrapalha o revestimento por imersão na prática: quanto mais fino for o revestimento desejado, mais devagar será a taxa de produção, o processo será mais sensitivo às perturbações externas, mais difícil será o comprometimento com a qualidade e maior será o custo (SCRIVEN, 1988).

O *dip coating* é um processo sem desperdícios, de baixo custo, com boa repetibilidade e capaz de oferecer um excelente controle da espessura do revestimento. Como pontos negativos, destaca-se a impossibilidade de assegurar a uniformidade da espessura ao longo de toda a placa, bem como a necessidade de um eficiente sistema/método de controle da viscosidade do fluido, dado esta alterar-se com a natural evaporação de solvente do fluido (JONES, 2010 apud CRUZ, 2014).

Para um melhor entendimento do *dip coating*, é importante destacar um processo fundamental que pode estar envolvido: o *sol-gel*. A técnica *sol-gel* é um método de deposição amplamente utilizado na ciência de materiais para a criação de revestimentos. O termo *sol* é geralmente empregado para definir uma dispersão de partículas coloidais estável em um fluido, enquanto o termo *gel* pode ser visto como sendo um sistema formado pela estrutura rígida de partículas coloidais ou de cadeias poliméricas que imobilizam a fase líquida em seus interstícios. O processo inicia-se com a hidrólise de um precursor líquido (sol), que por poli condensação forma gradativamente um gel. Tal gel é um sistema bifásico composto de líquido (solvente) e sólido, o qual é utilizado para a aplicação do revestimento que, passo a passo, vai sofrendo redução do volume líquido (podendo ser acelerado por secagem) (ILER, 1979 apud HIRATSUKA et al, 1994).

Durante um revestimento de dip coating com *sol-gel*, os componentes inorgânicos são rapidamente concentrados na superfície do substrato por drenagem gravitacional com decorrentes reações de evaporação e condensação. Alguns exemplos dos vários fatores que influenciam na estrutura dos filmes depositados são: taxas relativas de evaporação e condensação, pressão capilar, tamanho e opacidade dos componentes fractais, velocidade de mergulho. Através do controle desses fatores é possível ajustar o tamanho de poro, área de superfície, volume de poro e também índice de refração do filme depositado (BRINKER et al., 1991).

O processo *sol-gel* é bastante interessante pois permite preparar materiais com estruturas distintas a partir do controle da cinética de transformação. Deste ponto de vista, este processo assemelha-se à transformação líquido-sólido e pode ser razoavelmente bem compreendido através da teoria termodinâmica dos fenômenos críticos e dos modelos cinéticos de agregação. Isto permite projetar novos materiais com propriedades peculiares. Além disto, como a cinética deste processo é lenta, é possível analisar *in-situ* as reações de hidrólise e condensação, permitindo a obtenção de informações valiosas sobre os mecanismos reacionais (HIRATSUKA et al, 1995).

2.2 Arduino

A tecnologia da placa Arduino é baseada em um micro controlador muito versátil que potencializa suas funções para além de uma simples interface passiva de aquisição de dados, podendo operar sozinha no controle de vários dispositivos e tendo assim aplicações em instrumentação embarcada e robótica. O Arduino é uma plataforma de *hardware open source* de fácil utilização, ideal para a criação de dispositivos que permitam a interação com o ambiente, com uma interface no computador baseada na linguagem C/C++ e com diferentes versões nacionais da placa. As restrições ao acesso dessa tecnologia, em geral, se devem ao desconhecimento e à baixa oferta no mercado nacional, e por isso se torna essencial o estudo e trabalho nesse assunto (SOUZA et al., 2011).

A maior vantagem do Arduino em relação a outras plataformas de desenvolvimento de micro controladores é a sua facilidade de utilização, o que permite que pessoas que não sejam de áreas técnicas possam aprender o básico e criar seus próprios projetos em um período relativamente curto e sem a necessidade de um conhecimento especializado em eletrônica. Há uma imensa comunidade de pessoas usando Arduinos e compartilhando seus códigos e diagramas de circuito para que outros copiem e modifiquem (MCROBERTS, 2015).

O Arduino é um *open source* que foi projetado para introduzir a programação para pessoas não familiarizadas com o desenvolvimento de *software*, o que torna conveniente a estudantes. Ele inclui um editor de código que contém recursos como realce de sintaxe, correspondência de chaves e recuo automático, sendo também capaz de compilar e fazer *upload* de programas para a placa com um simples *click*. Um programa ou código escrito no Arduino é chamado de "*sketch*" (GALADIMA, 2014).

2.3 Servo Motor

O servomotor é uma máquina síncrona composta por uma parte fixa (o estator) e outra móvel (rotor). O estator é bobinado como no motor elétrico convencional, porém, apesar de utilizar alimentação trifásica, não pode ser ligado diretamente à rede, pois utiliza uma bobinagem especialmente confeccionada para proporcionar alta dinâmica ao sistema. O rotor é composto por imãs permanentes dispostos linearmente sobre o mesmo e com um gerador de sinais chamado resolver instalado para fornecer sinais de velocidade e posição (MORETO, 2007 apud CAROLINO, 2014).

O paradigma da programação na qual o Servo motor se enquadra é na orientação a objetos. A orientação a objetos é uma tecnologia que enxerga os sistemas como sendo coleção de objetos integrantes e consiste em considerar os sistemas computacionais como uma coleção de objetos que interagem de maneira organizada entre si. Os programas orientados a objetos são programas estruturados em módulos que agrupam um estado e operações sobre este estado, apresentando ênfase em reutilização do código (FARINELLI, 2007).

Os servomotores são maquinas síncronas, compostas de seis polos no estator, de alimentação trifásica, imãs permanentes dispostos linearmente sobre a face do rotor e

um sensor analógico chamado feedback para realimentação de posicionamento e velocidade (FANUC, 1998 apud MORETO, 2007).

3 MATERIAIS E MÉTODOS

3.1 Materiais

A primeira etapa do trabalho foi a criação do projeto. A ideia geral do equipamento de *dip coating* é administrar a rotação do motor de forma que a velocidade de movimentação vertical do substrato seja constante e pré-definida, bem como os tempos dos intervalos. Dessa forma, inicialmente foi feito o dimensionamento do motor.

Como o intuito do projeto é reproduzir um equipamento de *dip coating* de forma econômica, foi feito um estudo dos motores elétricos mais simples que pudessem suportar a carga[4] e estivessem de acordo com a tensão de alimentação proveniente do Arduino (5 volts). Três opções eram possíveis: motor DC, motor de passo e servo motor (de rotação contínua). O motor DC opera sob tensão de 5 volts e possui uma rotação aproximada de 330 rpm. O motor de passo opera a 6 volts e é mais indicado para projetos de movimentação angular controlada, com foco na precisão do posicionamento. O servo motor opera de 4,8 a 6 volts e possui uma rotação aproximada de 50 rpm (a 4,8 volts). Dessa forma, como o *dip coating* trabalha com velocidades baixas, o servo motor se destaca em relação ao motor DC por possuir menor velocidade de rotação. Assim, foi decidido utilizar o servo motor de rotação contínua (pois normalmente são travados entre 0º e 180º), que possui um torque de 6,2 kg/cm e está de acordo com as condições de operação.

Antes do início da programação no Arduino, é necessário projetar a interface entre a rotação do rotor e a descida do substrato. Para a solução desse problema foi utilizada uma ideia parecida com a da vara de pescar: se o substrato estiver preso em uma das extremidades de uma linha enquanto na outra extremidade estiver o rotor, tal linha pode ser enrolada no rotor (como um carretel) e, conforme o sentido de rotação, implica na subida ou descida do substrato. Dessa forma, são necessárias 4 peças:

[4] A aplicação normalmente utilizada da carga é no revestimento de chapas metálicas, como aço-carbono ou alumínio, de áreas de até 10 cm².

a) Extensão do rotor - a fim de alocar o carretel. Para a decisão da extensão do motor deve-se considerar a alocação do carretel. Assim, pode ser utilizado um parafuso contínuo, pois além de ser cortável em qualquer tamanho, é fixável ao rotor. Para fixar o parafuso contínuo ao rotor foram utilizadas tanto as peças de extensão do próprio servo quanto parafusos pequenos de fixação.
b) b) Carretel - a fim de limitar e enrolar a linha de forma adequada. Para o carretel, aproveitando o formato de parafuso na extensão do rotor, é possível rosquear peças (como roscas e rodelas) a fim de fixar os dois limites laterais para o enrolamento da linha. Dessa forma, é possível determinar diferentes tamanhos de carretel (utilizando as rodelas), bem como a distância até a base do rotor (importante pois quanto maior o tamanho do rotor maior será a carga mecânica resultante no motor).
c) c) Linha – tendo que ser inextensível e flexível. Em um primeiro momento foi cogitado usar algum tipo cabo de aço, devido à resistência a tração. No entanto, o cabo de aço ao sofrer o enrolamento no carretel não se mantém estabilizado, pois o cabo tende a ficar esticado. Dessa forma, foi decidido utilizar um cabo de náilon, que é utilizado para amarrar sacos de materiais de construção (suporta altos carregamentos de areia, garantindo a aplicabilidade para os esforços do projeto), por ser inextensível e não oferecer resistência ao sofrer enrolamento.
d) d) Haste de metal – para melhorar a estabilidade. A finalidade da haste de metal é de ser a parte que será passada pelo guia linear (a fim de garantir a descida vertical sem oscilações pendulares principalmente). Dessa forma, será presa à ponta da linha e será responsável pela estabilização do processo. Assim, basta que a haste seja rígida, reta e fina o suficiente para passar no guia linear. Uma vantagem importante do uso da haste de metal é a de evitar que a flexibilidade do cabo influencie de forma pendular durante o processo de mergulho.

A fim de garantir maior estabilidade, em um primeiro momento foi cogitado utilizar apenas um guia linear para reduzir o movimento pendular, localizado próximo à região de mergulho. Porém, esse mecanismo permite micro oscilações que podem ser evitadas. Dessa forma, ao invés de descer o cabo por apenas um guia linear, foi decidido descer a haste de metal fina por dentro de dois guias lineares pois, assim, a região entre a superfície externa da haste e a superfície interna do guia, que antes permitia um micro movimento pendular, agora possui liberdade limitada devido ao outro guia linear presente mais abaixo. Assim, o cabo de náilon servirá apenas para a

conversão da energia entre o toque do motor a subida do cabo, visto que a haste de metal que é fixada ao cabo e que passa pelos guias lineares.

Com a parte móvel do projeto definida, visto que o motor gerenciado pelo Arduino fará a subida e descida do cabo de náilon por meio de enrolamento, limitado pelo carretel, com velocidade de rotação constante e pré-definida, pode-se iniciar o desenvolvimento do suporte do equipamento.

A fim de situar todas as peças em uma máquina consolidada e estabilizada, deve-se projetar a maneira como todos os componentes serão dispostos e em que tipo de suporte eles estarão anexados. Dessa forma, é importante destacar alguns fatos:

- O suporte deverá ter boa estabilidade e resistência para que se tenha uma melhor qualidade dos filmes aplicados.

- Uma superfície rígida deverá suportar tanto a placa Arduino quanto o servo motor.

- O servo motor deve estar disposto de forma que o rotor possa desenrolar o cabo sem resistências externas e fora da superfície do item acima.

- A altura mínima do equipamento deve considerar o tamanho da haste de metal, pois ela deverá passar por dois guias lineares e sempre se manter em contato com ambos.

- A horizontalidade de todas as peças é fundamental (bem como a verticalidade do cabo/haste).

Considerando tais necessidades e analisando as diferentes possibilidades no mercado, a escolha da base do equipamento foi de usar um suporte de monitor por dois motivos principais: o peso de um monitor é maior do que as peças a sofrem *dip coating* (normalmente), garantindo a estabilidade, e porque pode-se aproveitar o molde de parafusos de fixação do monitor (no caso da escolha, *VESA*[5]) para parafusar a superfície rígida, que comportará o Arduino e o servo motor.

Dessa forma, após a análise de diferentes opções, um modelo destacou a atenção, ilustrado na Figura 5 pois, além de permitir a portabilidade da máquina como um todo, visto que se instala em praticamente qualquer mesa, possui uma longa articulação

[5] *VESA* é um padrão internacional de furos, localizado no verso de TVs, que serve para a fixação de parafusos. Permite checar a compatibilidade entre a TV e o suporte.

horizontal, que permite a descida do substrato em um béquer, por exemplo, sem se preocupar com possíveis obstáculos verticais.

Figura 5 – Suporte Articulado de Mesa para monitor F50N.

Fonte: Próprio autor.

Com base na escolha do modelo acima, é necessário projetar um suporte rígido. Ele serve para apoiar a placa Arduino e servo motor de forma mais simples e prática, visto que isso não pode ser feito apenas com o F50N[6]. Tal suporte será parafusado no F50N e deve possuir uma superfície horizontal para alocação das peças.

Um solução encontrada foi a de utilizar uma peça de madeira em formato de L (parafusando um lado interno do L no padrão VESA e o outro na barra de coluna do suporte). Dessa forma, os parafusos garantem a estabilidade e há uma superfície rígida e horizontal no topo do suporte para a alocação da parte eletrônica.

Outro motivo para a utilização de uma madeira em L como suporte rígido é o fato de que os dois guias lineares verticais que limitarão o movimento da haste de metal deverão estar próximos à região de mergulho. Assim, com a peça em L, é possível parafusar os guias na própria madeira, garantindo estabilidade e permitindo o controle

[6] Modelo do suporte articulado de mesa mostrado na Figura 5.

de verticalidade. Define-se assim, também, a fixação dos guias lineares por meio de parafusos na madeira em L.

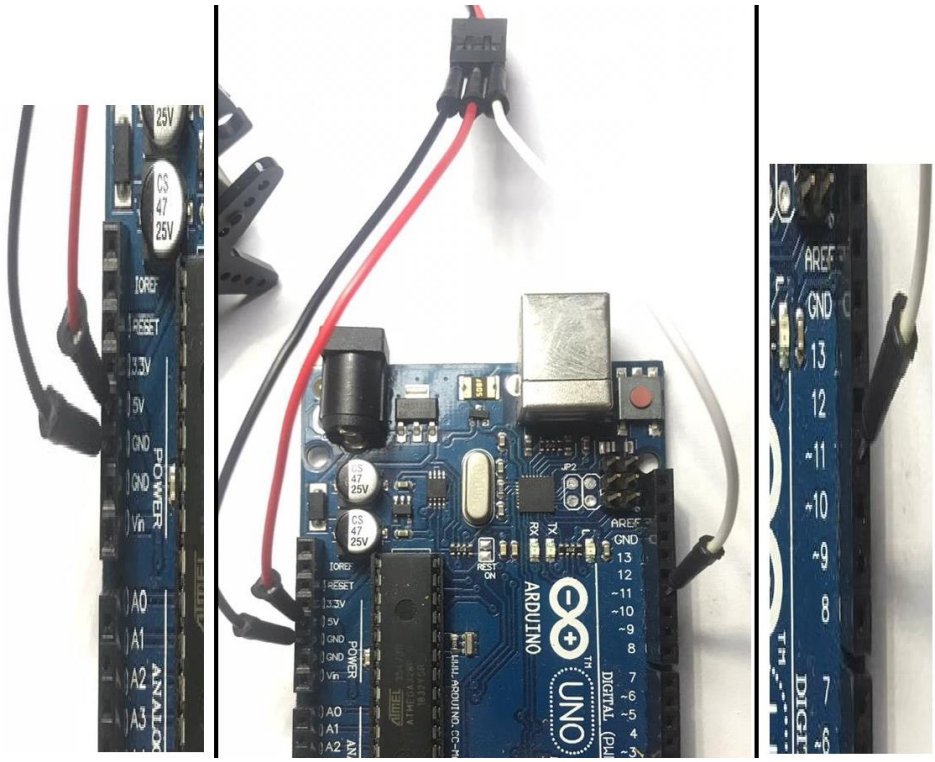
3.2 Métodos

3.2.1 Conexão entre o Arduino e o Servo motor

Para início da programação, é necessário baixar o próprio software do Arduino, feito de forma gratuita online no site <www.arduino.cc>. Nele, são armazenadas as funções e realizadas as programações, que serão enviadas ao Arduino via USB.

Para que se teste um código, é necessário conectar a placa Arduino à alimentação (no caso o computador) e fazer a conexão dos fios entre o Servo motor e o Arduino. A funcionalidade do motor depende diretamente da programação feita na placa, a qual manipula a energia dos fios conectados. Para a conexão do servo motor, deverão ser utilizados *Jumpers*[7]. O modo correto de se fazer a ligação é o ilustrado na Figura 6.

Figura 6 – Ligação entre Arduino e servo motor.



Fonte: Próprio autor.

[7] *Jumper* é a denominação do fio utilizado em prototipagem. Existem jumpers fêmea e macho, que se diferenciam pelo tipo de conector presente na ponta. No caso do jumper macho, o cabo possui o metal da ponta exposto para a conexão, enquanto o cabo fêmea possui o encaixe onde o cabo macho entra (sendo usado como forma de extensão de fio).

Nota-se que 3 cabos são necessários no mínimo para essa ligação. O cabo preto, conectado ao pino GND do Arduino, que significa *ground* e é a entrada do terra. O cabo vermelho, conectado ao pino 5V, que é o cabo de alimentação do motor e que recebe 5 volts do Arduino. O cabo branco, ligado ao pino 11, que é o sinal PWM e será responsável pelo gerenciamento do motor. No exemplo da Figura 6, o cabo branco poderia estar conectado também nas portas 9,10 ou 11, pois elas também são PWM (marcadas com um '~' na placa). Assim, com as conexões feitas e o software baixado, pode-se iniciar a parte de programação e, em seguida, a fase de testes.

3.2.2 Programação

Para se iniciar a programação, é importante destacar que a linguagem utilizada é a C/C++. A plataforma Arduino trabalha com uma estrutura de programação na qual existem duas funções principais no programa. A primeira é a função *void setup ()*, que é a utilizada para fazer a definição dos parâmetros iniciais do programa e dos valores que são utilizados durante o código principal. A segunda é a função *void loop ()*, que é a função principal, na qual o programa deve ser redigido, e sua estrutura é no formato de *loop*, ou seja, a função fica repetindo o código de forma infinita a menos que alguma coisa cause o *break* desse *loop*.

A plataforma Arduino possui bibliotecas prontas, que são módulos contendo funções variadas que são fundamentais para maiores funcionalidades. Além das que já existem no software do Arduino, diferentes bibliotecas podem ser baixadas e adicionadas de forma livre a fim de aprimorar o código. Como se utiliza um servo motor, foi utilizada a biblioteca "Servo.h", que é própria do Arduino para servo motores.

Assim, para o desenvolvimento do código é fundamental entender as funções que foram utilizadas. Elas estão descritas na Tabela 1 (as funções que possuem * na explicação são as específicas da biblioteca <Servo.h>). Nessa tabela as variáveis x e y são utilizadas de forma ilustrativa, visto que x representa qualquer nomenclatura e y representa qualquer valor. A variável S também é ilustrativa e representa o nome dado ao servo motor no programa. A letra k representa uma condição, geralmente de desigualdade ou de valor de variáveis.

Tabela 1 – funções fundamentais do projeto e suas utilidades.

Função	**Utilidade**
int **x**=**y**	Declara uma variável inteira **x** na memória do programa que recebe o valor de **y**.
float **x**=**y**	Declara uma variável racional **x** na memória do programa que recebe o valor de **y** (usada devido a valores decimais).
Servo **S**	*Declara um servo motor (objeto) na variável **S**, para enviar os comandos à ela e discriminar no programa qual motor está sendo utilizado (poderia ser mais de um).
S.attach(**y**)	*Utilizada para anexar o pino **y** do Arduino em que será ligado o servo motor **S**.
S.write(**y**)	*Envia um valor **y** ao servo **S**. No caso de um servo motor continuo, **y** varia de 0 a 180, sendo 0 a maior velocidade em uma direção, 90 a velocidade zero e 180 a maior velocidade na outra direção.
delay (**y**)	Utilizada para definir o tempo **y** (em milissegundos) que os comandos se manterão ativos.
while (**k**)	Cria um *loop* enquanto a expressão **k** que estiver entre parêntesis se torne falsa (repete a função enquanto a condição **k** seja verdadeira).
if (**k**)	Executa os comandos escritos no seu interior uma vez caso a expressão **k** que estiver entre parêntesis seja verdadeira.

Assim, utilizando as funções acima e a lógica da necessidade do equipamento de *dip coating*, um primeiro código foi desenvolvido a fim de realizar a calibração das velocidades. O motivo disso é o fato do *input*[8] de 0 a 180 da função do servo motor ser adimensional e não possui relação direta com algum parâmetro no formato rad/s por exemplo, o que impede a noção da velocidade real do rotor. Além disso, destaca-se que a rotação do motor é afetada pelo aumento de carga no rotor. Assim, é necessário descobrir quanto vale a relação entre o *input* do servo e a velocidade real, até porque o dado que o operador vai digitar no equipamento é de dimensão [mm/s] por exemplo. Inicia-se, então, o processo de calibragem das velocidades, descrito no tópico seguinte.

[8] O *input* do motor é o valor inserido na função "*write*" que está relacionado com a intensidade do pulso enviado, que deve ser um valor entre 0 e 180.

3.3 Calibração de velocidades

A calibração das velocidades depende da forma como as funções da biblioteca <Servo.h> trabalham. No caso, a função *write* insere um *input* de velocidade no motor que mantém o comando acionado pelo tempo estipulado na função *delay*, que é colocada na linha do código logo abaixo.

Assim, para descobrir essa relação, será utilizada a fórmula de velocidade constante (variação de espaço dividida pela variação de tempo, pois a velocidade inserida em um *write* é realmente constante durante o *delay*), pois como o tempo será um dado inserido, basta saber a distância que a haste de metal desce verticalmente nesse tempo inserido para se calcular a velocidade real de descida (e, da mesma forma, pra subida).

Então, utilizando a fórmula da velocidade constante, o tempo inserido na função *delay* e a distância percorrida, estima-se a velocidade real para cada *input* de velocidade. Uma coisa importante a se destacar é o fato de que um mesmo *input* aplicado para a subida e descida terá resultados diferentes de velocidade. Isso decorre do fato que, para a descida, a força peso da linha e da haste de metal atuam a favor da rotação, enquanto na subida elas atuam contra a rotação. Isso implica que, comparadas com a velocidade do motor sem carregamento nenhum, a de velocidade descida será maior e a velocidade de subida será menor.

Assim a primeira medição de velocidades foi realizada e os valores foram colocados em uma planilha de Excel. As medições foram colocadas em gráfico e se mostraram aproximadamente com um caráter de reta, com alguns valores discrepantes. No entanto, como a regressão linear não representava de forma fiel os dados, além do fato da possibilidade das medições terem sido feitas de forma imprecisa, se tornou necessário um novo procedimento experimental, agora com maior precisão nas medições.

Como em procedimentos experimentais desse tipo a qualidade das leituras é muito afetada por conta da deflexão, se tornou necessária uma adaptação. Então, para a redução das imprecisões no processo experimental e principalmente o efeito da deflexão, foram usados basicamente 2 objetos: uma luminária de mesa e uma régua. A ideia é, ao invés de medir a distância de descida com a régua suspensa próxima a linha ou algo parecido, localizar a luminária de mesa na frente do equipamento a fim de utilizar a sombra da linha (projetada na madeira em L atrás) como parâmetro de

distância percorrida, utilizando uma régua fixada também na madeira em L paralelamente próximo à sombra da linha, para que assim a medição da distância percorrida por tempo pelo *input* aplicado seja visto de acordo com a movimentação da sombra. Essa técnica, ilustrada na Figura 7, reduz muito os efeitos da deflexão do experimento e permite uma maior representatividade das medições feitas.

É útil fixar algum sólido pequeno na lateral da haste de metal apenas para se destacar da sombra da haste pois, como a ela é cilíndrica, sua sombra é retangular e homogênea, o que impede de medir a movimentação de um ponto qualquer que não seja do topo nem da base (a ideia é gerar uma saliência na sombra da haste que será usada como parâmetro de deslocamento vertical usando a régua). Na Figura 7, o sólido usado como referência é uma abraçadeira de nylon amarela, que está cortada a fim da sombra da ponta ser usada como uma espécie de ‘ponteiro’.

Figura 7 – Técnica de medição de deslocamento vertical utilizada.

Fonte: Próprio autor.

Assim, utilizando a técnica descrita acima, as medições foram refeitas. Novamente os dados foram colocados em uma planilha do Excel a fim de gerar um gráfico para, assim, tentar gerar uma aproximação que represente a relação entre a velocidade e o *input*. Então, com as medidas realizadas, foram feitas regressões lineares (tanto para a subida quanto para a descida) a fim de descrever as retas em expressões, que serão colocadas na programação no Arduino. Os dados de medição, as regressões lineares e o código final estão expostos no tópico a seguir.

4 RESULTADOS

4.1 Calibração de velocidades

Conforme exposto no item anterior, foram realizadas duas medições de subida e descida, sendo que as segundas medições foram mais precisas (devido à metodologia aplicada) e fundamentais devido à grande imprecisão dos dados da primeira medição. É importante salientar que a velocidade máxima que o motor consegue suportar (tanto na descida quanto na subida) é de aproximadamente de 40 milímetros por segundo, enquanto a mínima é de aproximadamente 5 milímetros por segundo. Isso foi descoberto durante a primeira medição de velocidades de descida, devido ao fato de a partir de certo valor de *input* a velocidade se manter aproximadamente constante em 40 milímetros por segundo (se confirmou na subida também), e, por isso, a 1ª medição de descida possui mais valores que os outros. Dessa forma, as medições seguintes tiveram os *inputs* considerados até que atinjam o valor correspondente de velocidade de 40 milímetros por segundo, pois todos os valores de *input* acima disso também resultam em 40.

Outro fato importante a se destacar é a influência do carregamento em um motor. Em teoria, o *input* de valor 90 seria a velocidade zero (de repouso) do rotor. Porém, dependendo de fatores como o ambiente e principalmente o carregamento, esse valor discreto de 90 acaba assumindo uma faixa. Dessa forma, não necessariamente o *input* de 89 será a menor velocidade em uma direção ou o 91 será a menor velocidade na outra direção, pois o carregamento influencia diretamente nesse fator. Isso se prova nos valores do procedimento experimental, expostos na Tabela 2 abaixo, pois na subida o *input* mínimo para movimentação do servo é 99 (e

não 91, que seria caso não houvesse carregamento) enquanto na descida o *input* mínimo para movimentação do servo é 89 (como deveria ser, pois o carregamento contribui com a rotação). Assim, estão expostos na Tabela 2 abaixo os resultados das medições realizadas:

Tabela 2 – Medidas do procedimento experimental de calibração de velocidades.

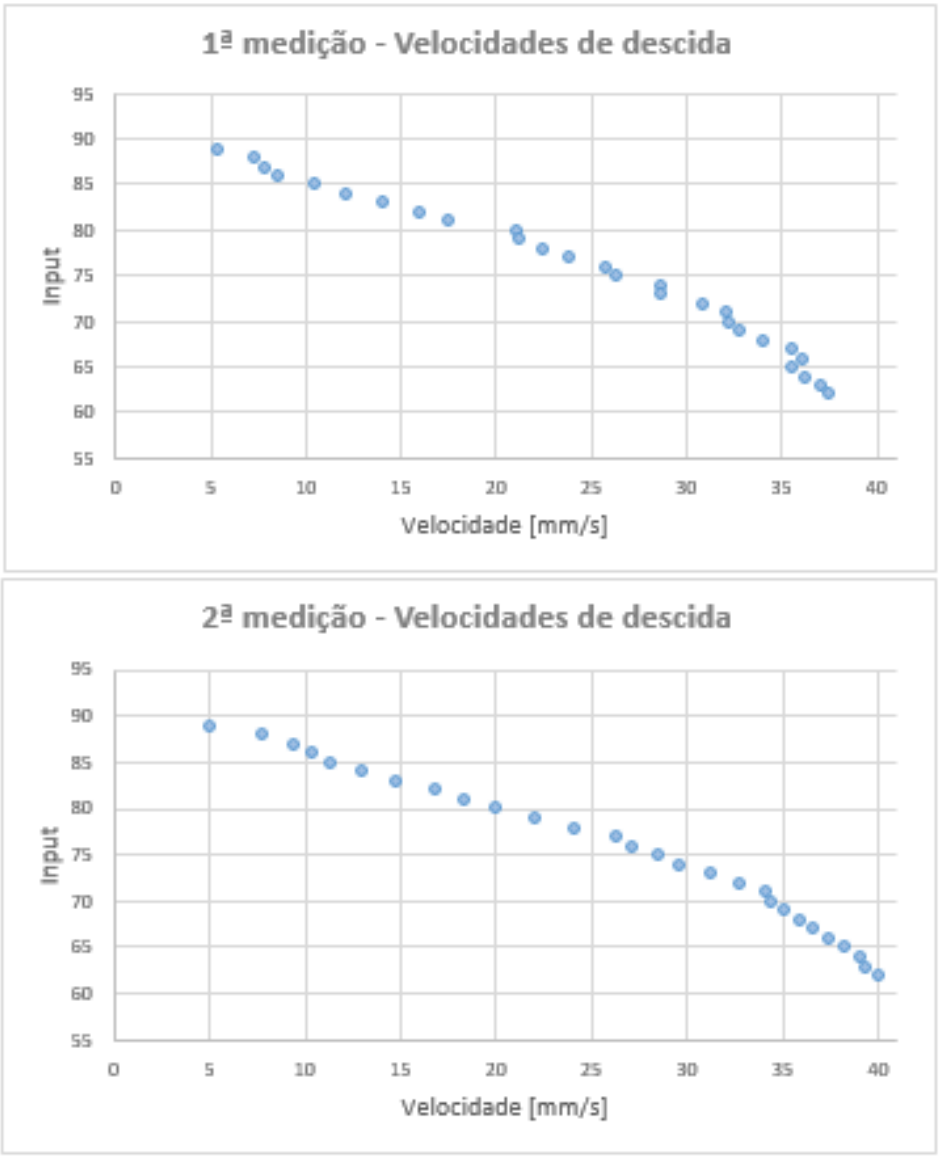
Descida Input	1ª medição [mm/s]	2ª medição [mm/s]
89	5,3	5,0
88	7,2	7,7
87	7,8	9,4
86	8,5	10,3
85	10,4	11,3
84	12,1	13,0
83	14,0	14,8
82	16,0	16,9
81	17,5	18,3
80	21,0	20,0
79	21,1	22,0
78	22,4	24,0
77	23,9	26,3
76	25,7	27,1
75	26,3	28,4
74	28,7	29,6
73	28,6	31,3
72	30,8	32,8
71	32,0	34,1
70	32,3	34,3
69	32,8	35,0
68	34,0	35,9
67	35,5	36,5
66	36,0	37,3
65	35,5	38,2
64	36,3	39,0
63	37,0	39,3
62	37,5	40,0
61	37,3	
60	37,75	
59	38,00	
58	38,00	
57	38,33	
56	39,00	
55	39,33	
54	39,33	
53	40,33	
52	39,00	
51	39,67	
50	39,33	
49	40,00	
48	41,00	
47	40,33	
46	40,00	
45	40,00	
44	40,00	
43	40,30	
42	40,21	

Subida Input	1ª medição [mm/s]	2ª medição [mm/s]
99	4,44	5,0
100	4,94	5,9
101	6,72	7,6
102	7,88	8,7
103	10,69	11,2
104	13,00	12,9
105	14,67	15,1
106	16,40	16,6
107	17,60	18,6
108	19,93	20,3
109	21,75	22,1
110	22,75	23,7
111	24,33	25,4
112	24,83	27,2
113	25,83	28,7
114	27,83	30,2
115	28,17	30,9
116	30,00	32,3
117	30,25	33,4
118	31,50	34,3
119	32,75	34,8
120	33,00	36,0
121	34,00	36,8
122	34,75	39,0
123	35,35	40,1
124	35,65	
125	36,15	
126	36,50	
127	36,70	
128	36,95	
129	37,25	
130	37,95	
131	38,56	
132	39,15	
133	39,50	

Fonte: Próprio autor.

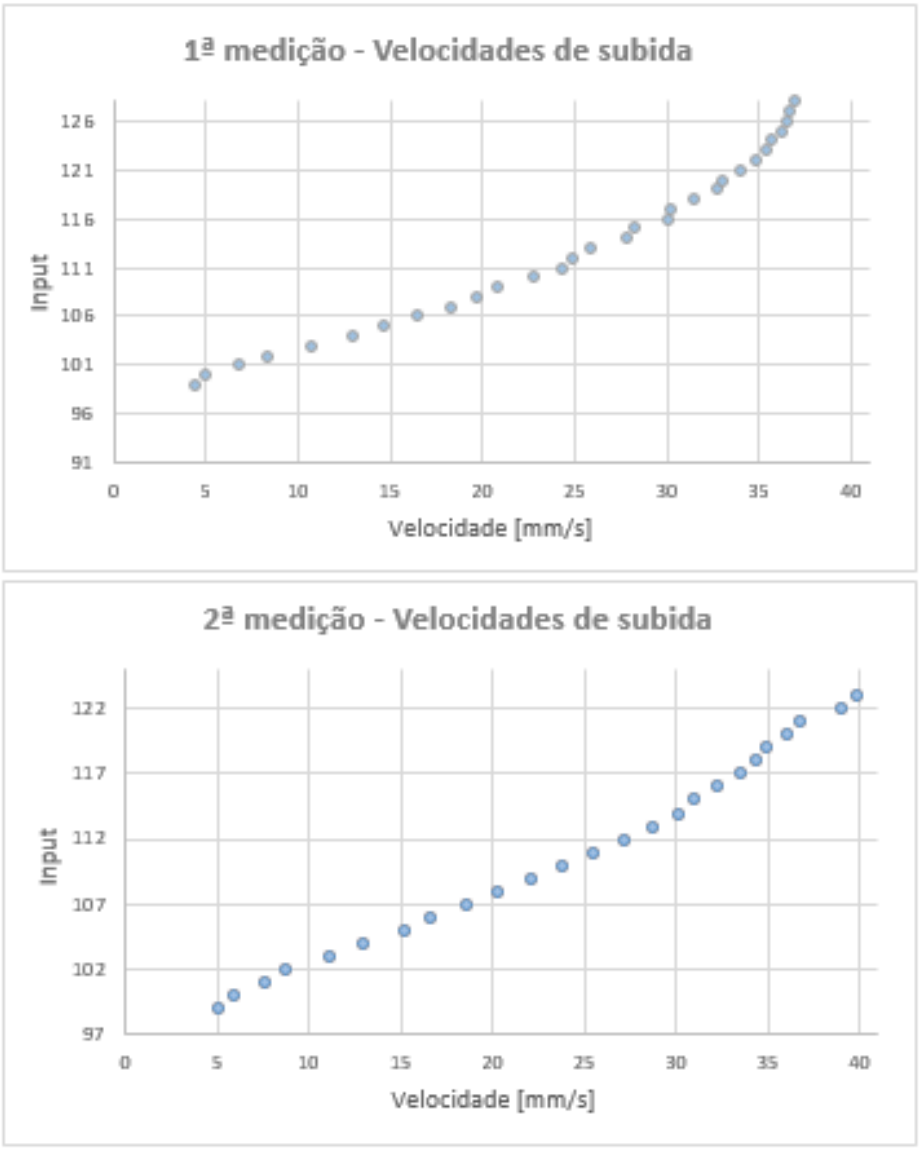
Assim, a fim de uma melhor visualização da evolução de precisão das medições, foram plotados os gráficos dos dados experimentais. Abaixo, na Figura 8, temos os gráficos das duas medições feitas de velocidades de descida e na Figura 9 temos os gráficos das duas medições de velocidades de subida. É importante destacar o aumento de linearidade dos gráficos, por mais que sutil, proveniente de um procedimento experimental mais delicado e fundamental para a realização das regressões lineares, que são o intuito da medição e devem representar da forma mais fiel possível a atuação da velocidade do motor.

Figura 8 – Medições de velocidades de descida realizadas.



Fonte: Próprio autor.

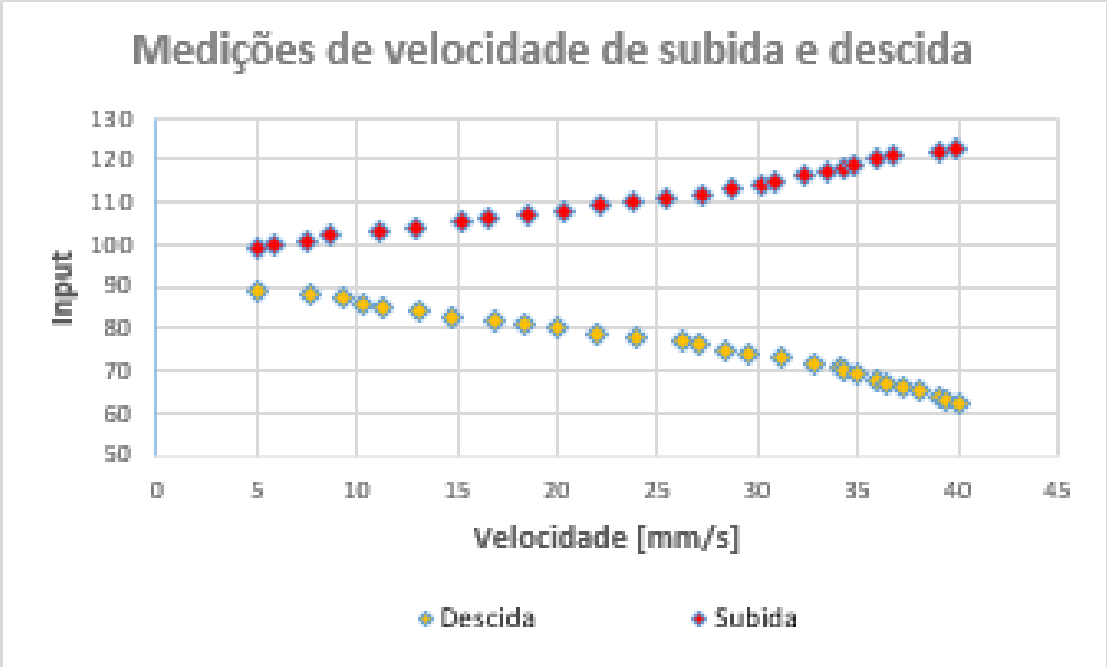
Figura 9 – Medições de velocidade de subida realizadas.



Fonte: Próprio autor.

Então, juntando os valores mais precisos em um único gráfico (incluindo subida e descida), uma característica importante do motor pode ser descoberta: o *input* de repouso, conforme explicado no segundo parágrafo do tópico 4.1. Dessa forma, analisando o gráfico da Figura 10, podemos notar que o input de repouso do servo motor no experimento é um valor entre 90 e 98. Isso se deve principalmente ao fato da força peso da haste de metal e será usado como velocidade zero na programação.

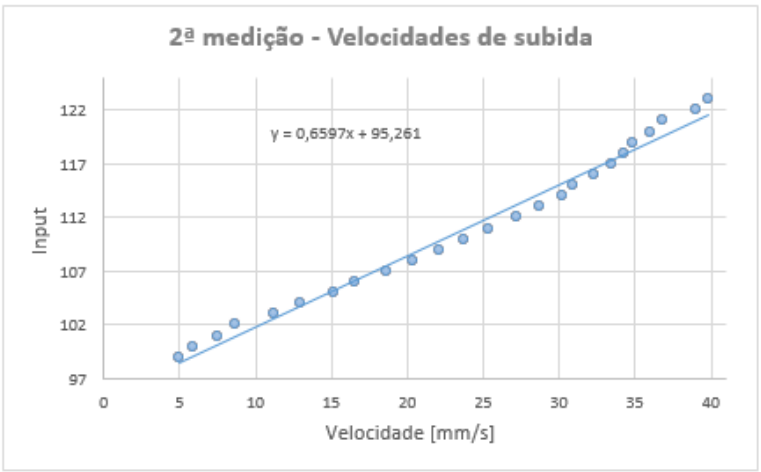
Figura 10 – Dados experimentais da relação entre *input* e velocidade do substrato.

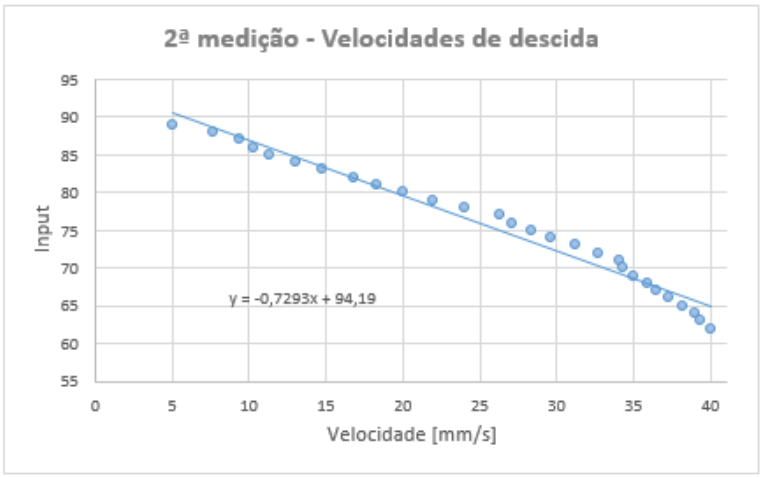


Fonte: Próprio autor.

Assim, analisando o gráfico da Figura 10 (segundas medições) foram realizadas regressões lineares (com as ferramentas do *software* Excel), em busca de uma relação no formato de equação de reta entre o *input* e a velocidade real. Abaixo, na Figura 11, é mostrado graficamente o resultado das regressões, que utilizam todos os dados até a velocidade máxima de 40 mm/s, bem como as equações da reta gerada.

Figura 11 – Regressões lineares das segundas medições de subida e descida.





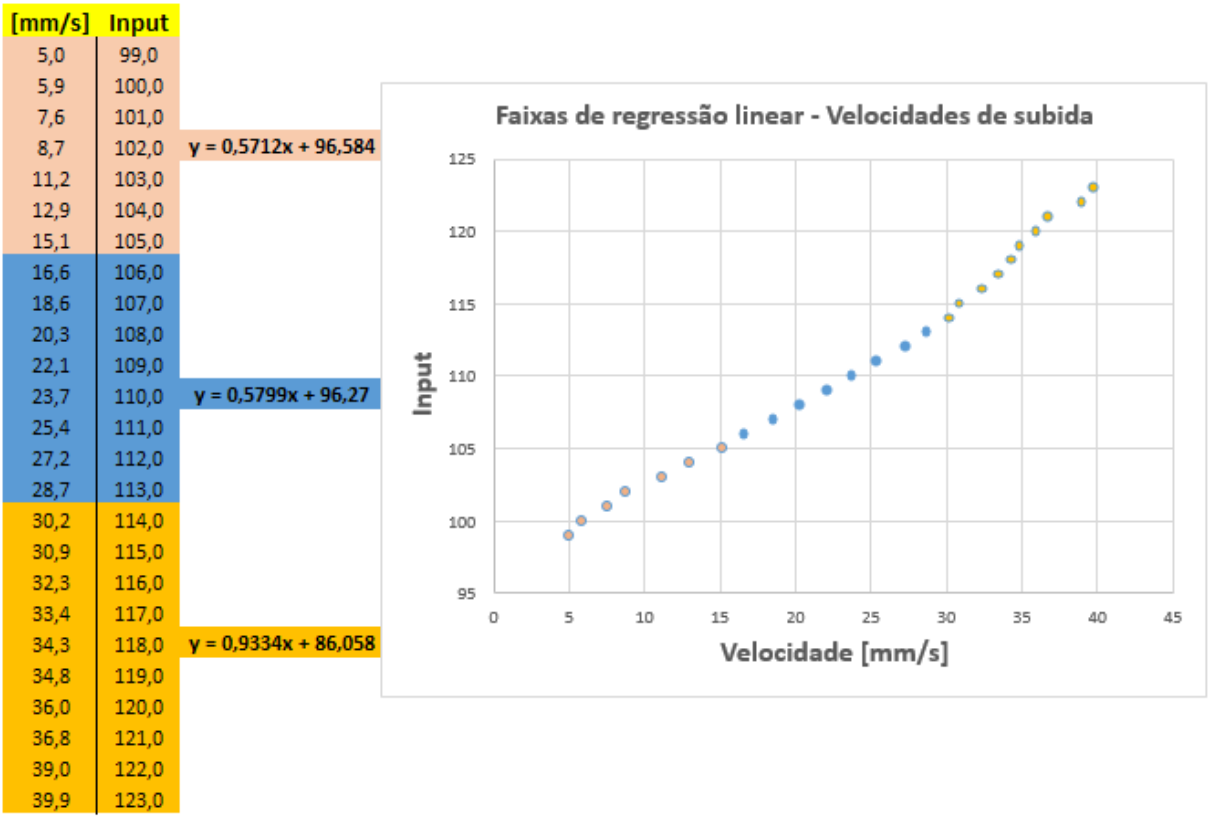
Fonte: Próprio autor.

Dessa forma, como pode-se notar na Figura 11, não se justifica o uso dessa aproximação para descrição da relação devido à má representatividade dos dados. Então, a fim de compensar o problema e considerando a análise visual dos gráficos, foi decidido realizar mais de uma regressão linear por série de dados, dividindo os valores em faixas, devido ao fato de ser possível descrever de forma mais aproximada a evolução dos valores com mais de uma reta.

Assim, a lógica da escolha das faixas foi devido ao fato que, quanto menor a diferença entre duas velocidades, maior será a representatividade caso se utilize uma delas como referência para a outra. Ou seja, é mais eficiente correlacionar valores próximos do que valores distantes. Sendo assim, as faixas foram definidas com base tanto em distribuição homogênea de valores quanto em análise gráfica.

Dessa forma, estão expostas na Figura 12 abaixo as faixas definidas para as velocidades de subida e, na Figura 13, as faixas definidas para as velocidades de descida. As equações geradas com as regressões lineares estão expostas de acordo com a cor (faixa definida) e os dados nos gráficos possuem o mesmo padrão de coloração.

Figura 12 – Faixas definidas para regressão linear nas velocidades de subida.

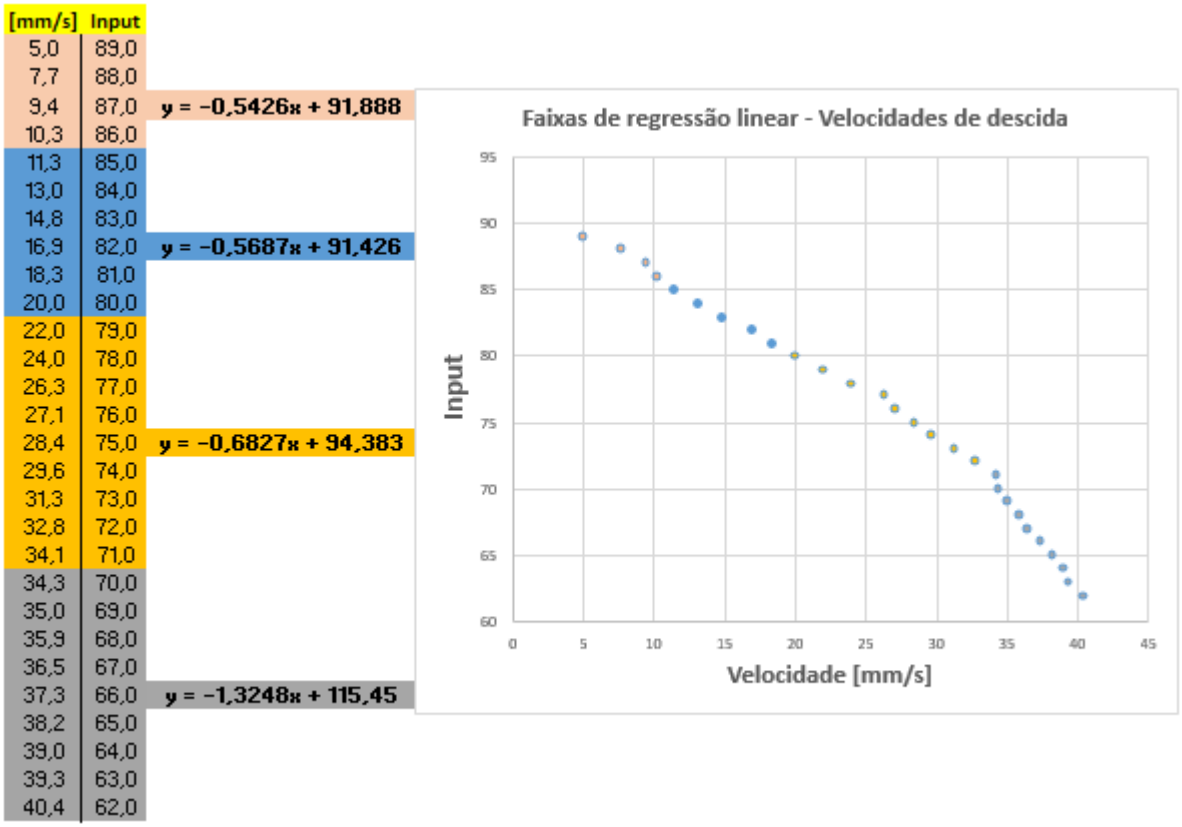


[mm/s]	Input	
5,0	99,0	
5,9	100,0	
7,6	101,0	
8,7	102,0	y = 0,5712x + 96,584
11,2	103,0	
12,9	104,0	
15,1	105,0	
16,6	106,0	
18,6	107,0	
20,3	108,0	
22,1	109,0	
23,7	110,0	y = 0,5799x + 96,27
25,4	111,0	
27,2	112,0	
28,7	113,0	
30,2	114,0	
30,9	115,0	
32,3	116,0	
33,4	117,0	
34,3	118,0	y = 0,9334x + 86,058
34,8	119,0	
36,0	120,0	
36,8	121,0	
39,0	122,0	
39,9	123,0	

Fonte: Próprio autor.

É importante destacar que, como para a velocidade de descida há um número maior de valores, foi realizada uma regressão linear a mais, focada nos valores iniciais que se mostram com maior variância.

Figura 13 – Definição das faixas para regressão linear nas velocidades de descida.



Fonte: Próprio autor.

Assim, as equações obtidas com as regressões lineares para representar a relação entre o *input* do motor e a velocidade real do substrato são inseridas no código, para que o operador possa finalmente digitar a velocidade desejada e a programa faça a conversão e envie ao Servo motor o *input* correto correspondente. Dessa forma, o código final desenvolvido, incluindo as calibrações, está exposto no tópico seguinte.

4.2 Código desenvolvido

Na Figura 14 abaixo temos o código desenvolvido (utilizando as funções descritas no tópico 3.2.2 - Programação), sendo a numeração das linhas a fim de guiar a explicação, a seguir, da lógica de programação e escrita.

Figura 14 – código desenvolvido para o equipamento de *dip coating.*

```
#include <Servo.h>
Servo servomotor;
float velocidade=30;              //em mm/s//
int temporepouso=3000;            //em milissegundos//
int numerodeciclos=3;             //adimensional//
int intervaloentreciclos=2000;    //em milissegundos//
int servoparado=93;               //calculado experimentalmente//
float distanciamaxima=5;          //em centímetros//
void setup() {
  servomotor.attach(11);
  servomotor.write(servoparado);
             }
void loop() {
  while (numerodeciclos>0){
      if ((5<=velocidade) && (velocidade<=10)){
        servomotor.write((-0.5426*velocidade)+91.888);
        delay(((10*distanciamaxima)/velocidade)*1000);
                                                 }
      if ((11<=velocidade) && (velocidade<=21)) {
        servomotor.write((-0.5687*velocidade)+91.426);
        delay(((10*distanciamaxima)/velocidade)*1000);
                                                  }
      if ((22<=velocidade) && (velocidade<=33)){
        servomotor.write((-0.6827*velocidade)+94.383);
        delay(((10*distanciamaxima)/velocidade)*1000);
                                                 }
      if ((34<=velocidade) && (velocidade<=40)){
        servomotor.write((-1.3248*velocidade)+115.45);
        delay(((10*distanciamaxima)/velocidade)*1000);
                                                 }
      servomotor.write(servoparado);
      delay(temporepouso);
      if ((5<=velocidade) && (velocidade<=15)) {
        servomotor.write((0.5712*velocidade)+96.584);
        delay(((10*distanciamaxima)/velocidade)*1000);
                                                 }
      if ((16<=velocidade) && (velocidade<=29)) {
        servomotor.write((0.5799*velocidade)+96.27);
        delay(((10*distanciamaxima)/velocidade)*1000);
                                                  }
      if ((30<=velocidade) && (velocidade<=40)) {
        servomotor.write((0.9334*velocidade)+86.058);
        delay(((10*distanciamaxima)/velocidade)*1000);
                                                  }
      servomotor.write(servoparado);
      delay(intervaloentreciclos);
      numerodeciclos=numerodeciclos-1;
                        }
            }
```

Fonte: Próprio autor.

A fim de descrever a lógica e processo do código, todas as linhas serão explicadas abaixo (é importante destacar que tudo que está entre '//' no código é apenas um comentário).

1 – Inserção da biblioteca Servo.h (inclui as funções *write*, *attach* e *Servo*);

2 – Definição de um servo motor com o nome de 'servomotor';

3 – Definição da velocidade desejada em mm/s (Que deve ser inserida pelo operador. No exemplo da Figura 14, vale 30);

4 – Definição do tempo de repouso submerso em milissegundos (Que deve ser inserido pelo operador. No exemplo da Figura 14 vale 3000, que equivale a 3 segundos);

5 – Número de ciclos de mergulho e secagem (Que deve ser inserido pelo operador. No exemplo da Figura 14, 3 ciclos);

6 – Tempo de repouso para evaporação em milissegundos (Que deve ser inserido pelo operador. No exemplo da Figura 14, 2000, que equivale a 2 segundos);

7 – Criação da variável 'servoparado', que pode receber qualquer valor entre 90 e 98 (conforme visto experimentalmente) e representa o *input* de repouso para o motor;

8 – Definição da distância máxima de descida do substrato em centímetros (Que deve ser inserida pelo operador. No exemplo da Figura 14, vale 5);

9 – Início da função *void setup* (), configuração das condições iniciais;

10 – Vinculação de um pino do Arduino com o objeto 'servomotor' criado (No exemplo da Figura 14, pino 11);

11 – Envia o valor de 'servoparado' para o servo motor, que fará com que a definição inicial do motor seja de repouso;

12 – Fechamento da função *void setup* ();

13 – Início da função *void loop* ();

14 – Criação da função *while* () a fim de definir um *break* para o *loop*, que no caso será quando 'numerodeciclos' for igual a 0;

15 – Criação do primeiro comando *if* () para checar se a velocidade está entre 5 e 10 (primeira regressão linear de descida);

16 – Caso ocorra o *if* () da linha 15, envia ao motor a variável ‘velocidade’ manipulada pela primeira regressão linear de descida, o que gera o Input correspondente do motor;

17 – Definição do tempo em que o comando da linha 16 se manterá ativo. No caso, conforme explicado sobre a fórmula da velocidade constante, temos que o tempo sempre será igual ao quociente entre a distância máxima e a velocidade (respeitando as unidades);

18 – Fechamento da função *if* () aberta na linha 15;

19 – Criação do segundo comando *if* () para checar se a velocidade está entre 11 e 21 (segunda regressão linear de descida);

20 – Caso ocorra o *if* () da linha 19, envia ao motor a variável ‘velocidade’ manipulada pela segunda regressão linear de descida, o que gera o *input* correspondente do motor;

21 – Definição do tempo em que o comando da linha 20 se manterá ativo;

22 – Fechamento da função *if* () aberta na linha 19;

23 – Criação do terceiro comando *if* () para checar se a velocidade está entre 22 e 33 (terceira regressão linear de descida);

24 – Caso ocorra o *if* () da linha 23, envia ao motor a variável ‘velocidade’ manipulada pela terceira regressão linear de descida, o que gera o *input* correspondente do motor;

25 – Definição do tempo em que o comando da linha 24 se manterá ativo;

26 – Fechamento da função *if* () aberta na linha 23;

27 – Criação do quarto comando *if* () para checar se a velocidade está entre 34 e 40 (quarta regressão linear de descida);

28 – Caso ocorra o *if* () da linha 27, envia ao motor a variável ‘velocidade’ manipulada pela quarta regressão linear de descida, o que gera o *input* correspondente do motor;

29 – Definição do tempo em que o comando da linha 28 se manterá ativo;

30 – Fechamento da função *if* () aberta na linha 27;

31 – Envia ao motor a variável ‘servoparado’, que induzirá seu repouso;

32 – Define o tempo em que o comando da linha 31 se manterá ativo, que no caso foi definido na variável ‘temporepouso’;

33 – Criação do quinto comando *if* () para checar se a velocidade está entre 5 e 15 (primeira regressão linear de subida);

34 – Caso o *if* () da linha 33 ocorra, envia ao motor a variável ‘velocidade’ manipulada pela primeira regressão linear de subida, o que gera o *input* correspondente do motor;

35 – Definição do tempo em que o comando da linha 34 se manterá ativo;

36 – Fechamento da função *if* () aberta na linha 33;

37 – Criação do sexto comando *if* () para checar se a velocidade está entre 16 e 29 (segunda regressão linear de subida);

38 – Caso o *if* () da linha 37 ocorra, envia ao motor a variável ‘velocidade’ manipulada pela segunda regressão linear de subida, o que gera o *input* correspondente do motor;

39 – Definição do tempo em que o comando da linha 38 se manterá ativo;

40 – Fechamento da função *if* () aberta na linha 37;

41 – Criação do sétimo comando *if* () para checar se a velocidade está entre 30 e 40 (terceira regressão linear de subida);

42 – Caso o *if* () da linha 41 ocorra, envia ao motor a variável ‘velocidade’ manipulada pela terceira regressão linear de subida, o que gera o *input* correspondente do motor;

43 – Definição do tempo em que o comando da linha 42 se manterá ativo;

44 – Fechamento da função *if* () aberta na linha 41;

45 – Envia ao motor a variável ‘servoparado’, que induzirá seu repouso;

46 – Definição do tempo em que o comando da linha 45 se manterá ativo. No caso, será o tempo de evaporação, no qual o substrato se manterá fora do líquido para a secagem e, por isso, se utiliza a variável ‘intervaloentreciclos’;

47 – Manipulação algébrica do valor de ‘numerodeciclos’. Isso ocorre pois, ao chegar nessa parte do código, um ciclo de mergulho e evaporação terá sido feito e, por isso, se retira 1 do valor de ‘numerodeciclos’ inicial, pois isso será utilizado como *break* do *loop* (pois quando ‘numerodeciclos’ for igual a zero, o *while* () da linha 14 para);

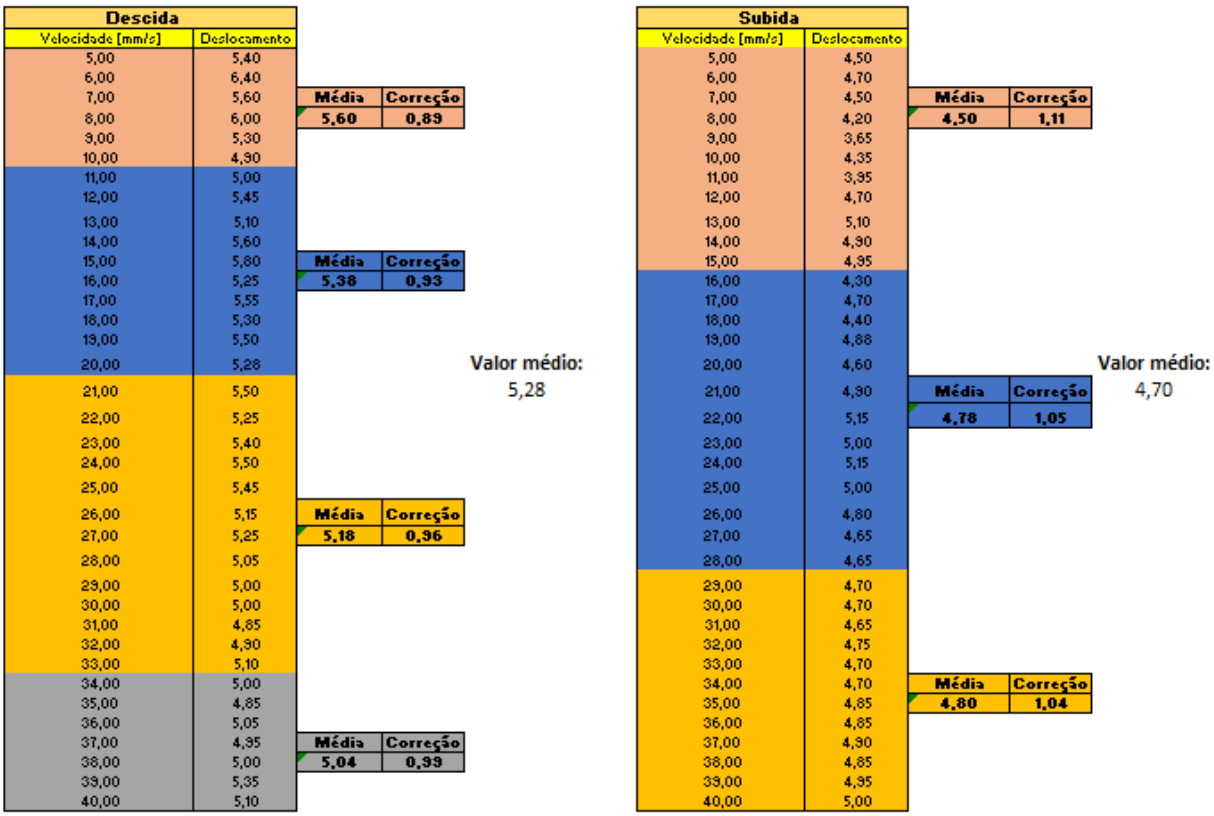
48 – Fechamento da função *while* () aberta na linha 14;

49 – Fechamento da função *void loop* () aberta na linha 13;

Assim, com o código pronto (incluindo a relação entre *input* e velocidade real) e com o protótipo construído, foi possível realizar testes a fim da comprovação da eficácia do projeto. Para isso, se torna necessário mais uma medição de velocidades, agora inserindo a distância de deslocamento vertical desejada (e não o tempo em que o *input* se mantém ativo, como era feito antes da calibração das velocidades).

Dessa forma, utilizando um valor referência de 5 centímetros para a distância de deslocamento inserida no código, foram calculados os deslocamentos reais do substrato a fim de testar o protótipo e mensurar o quanto o projeto comete de erro. Dessa forma, são calculadas as médias das distâncias percorridas por faixa de regressão e, a fim da melhoria dos resultados, é calculado um fator de correção (que compara a média das distâncias percorridas com o valor teoricamente correto, que no caso do teste é de 5 centímetros). Dessa forma, os dados estão expostos na Figura 15 abaixo junto com os parâmetros calculados, sendo que a média exposta sem orientação de cor é a média total dos valores.

Figura 15 – Deslocamentos realizados escolhendo 5 centímetros de referência.



Descida

Velocidade [mm/s]	Deslocamento	Média	Correção
5,00	5,40		
6,00	6,40		
7,00	5,60		
8,00	6,00	5,60	0,89
9,00	5,30		
10,00	4,90		
11,00	5,00		
12,00	5,45		
13,00	5,10		
14,00	5,60		
15,00	5,80		
16,00	5,25	5,38	0,93
17,00	5,55		
18,00	5,30		
19,00	5,50		
20,00	5,28		
21,00	5,50		
22,00	5,25		
23,00	5,40		
24,00	5,50		
25,00	5,45		
26,00	5,15		
27,00	5,25	5,18	0,96
28,00	5,05		
29,00	5,00		
30,00	5,00		
31,00	4,85		
32,00	4,90		
33,00	5,10		
34,00	5,00		
35,00	4,85		
36,00	5,05		
37,00	4,95		
38,00	5,00	5,04	0,99
39,00	5,35		
40,00	5,10		

Valor médio: 5,28

Subida

Velocidade [mm/s]	Deslocamento	Média	Correção
5,00	4,50		
6,00	4,70		
7,00	4,50		
8,00	4,20	4,50	1,11
9,00	3,65		
10,00	4,35		
11,00	3,35		
12,00	4,70		
13,00	5,10		
14,00	4,30		
15,00	4,35		
16,00	4,30		
17,00	4,70		
18,00	4,40		
19,00	4,88		
20,00	4,60		
21,00	4,30		
22,00	5,15	4,78	1,05
23,00	5,00		
24,00	5,15		
25,00	5,00		
26,00	4,80		
27,00	4,65		
28,00	4,65		
29,00	4,70		
30,00	4,70		
31,00	4,65		
32,00	4,75		
33,00	4,70		
34,00	4,70		
35,00	4,85	4,80	1,04
36,00	4,85		
37,00	4,90		
38,00	4,85		
39,00	4,95		
40,00	5,00		

Valor médio: 4,70

Fonte: Próprio autor.

Dessa forma, nota-se que as distâncias de descida estão ligeiramente maiores em média, enquanto as distâncias de subida estão ligeiramente menores em média. Assim, o fator de correção calculado para cada regressão foi utilizado para corrigir o *delay*, em busca de fazer com que o valor médio se aproxime ao máximo da valor ideal (5 cm). Assim, no código explicitado na Figura 15 foi feita uma pequena alteração (a fim da correção), multiplicando-se os *delay's* pelos respectivos fatores de correção. Dessa forma, o código final gerado é o seguinte, exposto na Figura 16. A diferença entre ela e o código da Figura 15 é que na Figura 16 os fatores de correção estão aplicados na expressão do delay.

Figura 16 – Código final do equipamento de *dip coating* com a correção.

```
#include <Servo.h>
Servo servomotor;
float velocidade=30;              //em mm/s//
int temporepouso=3000;            //em milissegundos//
int numerodeciclos=3;             //adimensional//
int intervaloentreciclos=2000;    //em milissegundos//
int servoparado=93;               //calculado experimentalmente//
float distanciamaxima=5;          //em centímetros//
void setup() {
  servomotor.attach(11);
  servomotor.write(servoparado);
              }
void loop() {
  while (numerodeciclos>0){
      if ((5<=velocidade) && (velocidade<=10)){
        servomotor.write((-0.5426*velocidade)+91.888);
        delay(((10*distanciamaxima)/velocidade)*1000*0.893);
                                                 }
      if ((11<=velocidade) && (velocidade<=21)) {
        servomotor.write((-0.5687*velocidade)+91.426);
        delay(((10*distanciamaxima)/velocidade)*1000*0.929);
                                                  }
      if ((22<=velocidade) && (velocidade<=33)){
        servomotor.write((-0.6827*velocidade)+94.383);
        delay(((10*distanciamaxima)/velocidade)*1000*0.964);
                                                 }
      if ((34<=velocidade) && (velocidade<=40)){
        servomotor.write((-1.3248*velocidade)+115.45);
        delay(((10*distanciamaxima)/velocidade)*1000*0.992);
                                                 }
      servomotor.write(servoparado);
      delay(temporepouso);
      if ((5<=velocidade) && (velocidade<=15)) {
        servomotor.write((0.5712*velocidade)+96.584);
        delay(((10*distanciamaxima)/velocidade)*1000*1.111);
                                                 }
      if ((16<=velocidade) && (velocidade<=29)) {
        servomotor.write((0.5799*velocidade)+96.27);
        delay(((10*distanciamaxima)/velocidade)*1000*1.045);
                                                  }
      if ((30<=velocidade) && (velocidade<=40)) {
        servomotor.write((0.9334*velocidade)+86.058);
        delay(((10*distanciamaxima)/velocidade)*1000*1.042);
                                                  }
      servomotor.write(servoparado);
      delay(intervaloentreciclos);
      numerodeciclos=numerodeciclos-1;
                          }
            }
```

Fonte: Próprio autor.

Então, utilizando os fatores de correção nas expressões de delay, as medições foram refeitas, visando mensurar o erro ainda presente no processo. Um problema a se destacar, notado durante o procedimento experimental, é a instabilidade de operação do motor para as velocidades mais baixas, devido provavelmente à qualidade do modelo. Notou-se que a partir de 10 mm/s aproximadamente os valores deixam de sofrer de tal instabilidade. Dessa forma, para não prejudicar a precisão do projeto, será considerado como range de operação apenas as velocidades entre 10 e 40 milímetros por segundo, porque mesmo embora a máquina consiga operar de 5 a 9 milímetros por segundo o erro presente não permite um bom nível de confiança. Tal instabilidade está ilustrada na Tabela 3.

Tabela 3 – Medições experimentais desconsideradas (referência de 5 cm).

Velocidade [mm/s]	Deslocamento [cm]
5,00	4,30
6,00	4,45
7,00	4,40
8,00	4,60
9,00	4,65

Assim, de novo usando a distância referência de 5 centímetros, foram refeitas as medições para todas as velocidades, e os resultados estão expostos na Tabela 4. Destaca-se o aumento significativo de precisão, visto pelo valor médio.

Tabela 4 – Medições experimentais utilizando a correção calculada.

Descida	
Velocidade [mm/s]	Deslocamento [cm]
10,00	4,85
11,00	5,00
12,00	4,95
13,00	4,50
14,00	5,10
15,00	5,40
16,00	4,90
17,00	5,30
18,00	5,00
19,00	5,10
20,00	4,80
21,00	4,95
22,00	5,00
23,00	5,00
24,00	5,20
25,00	5,00
26,00	5,10
27,00	5,10
28,00	4,90
29,00	5,00
30,00	4,95
31,00	4,80
32,00	4,70
33,00	4,80
34,00	4,95
35,00	4,85
36,00	4,95
37,00	4,90
38,00	4,70
39,00	4,75
40,00	4,70

Valor Médio:
4,92
Desvio Padrão:
0,18
Variância:
0,03

Subida	
Velocidade [mm/s]	Deslocamento [cm]
10,00	5,00
11,00	4,40
12,00	4,95
13,00	5,50
14,00	5,00
15,00	5,40
16,00	4,70
17,00	5,10
18,00	4,80
19,00	5,10
20,00	4,75
21,00	5.00
22,00	5,30
23,00	4,90
24,00	5,20
25,00	5,00
26,00	5,10
27,00	4,90
28,00	5,00
29,00	5,05
30,00	5,10
31,00	4,90
32,00	4,80
33,00	4,95
34,00	4,90
35,00	4,85
36,00	4,95
37,00	4,90
38,00	4,75
39,00	4,85
40,00	4,75

Valor Médio:
4,96
Desvio Padrão:
0,21
Variância:
0,04

Fonte: Próprio autor.

O protótipo construído, que foi utilizado para as medições experimentais e calibração da velocidade do Servo motor, está ilustrado na Figura 17. Destaca-se o fato do cabo que liga o Arduino na entrada USB do computador não estar conectado a fim de uma melhor visualização do equipamento, que independe do computador utilizado.

Figura 17 – Protótipo de *dip coating* construído.

Fonte: Próprio autor.

5 DISCUSSÃO

5.1 Projeto

O código escrito em conjunto com as sucessivas calibrações permitiram reproduzir um equipamento de *dip coating* utilizando os materiais descritos. A faixa de velocidade de operação com precisão é de 10 a 40 milímetros por segundo. Isso ocorre pois, conforme explicado, as velocidades mais baixas possuem uma maior variância, o que afetam a precisão da máquina e impedem a consideração delas como possíveis escolhas. Assim, a faixa de atuação com precisão do motor se deve às limitações do modelo selecionado.

Quanto à precisão do projeto, nota-se que as distâncias inseridas no programa, quando realizadas pelo equipamento, reproduzem o valor digitado para a descida e para a subida respectivamente com um desvio padrão de 0,1809 e 0,2130 e uma variância de 0,0327 e 0,0453. Nota-se, assim, a uniformidade dos dados e a boa precisão do projeto desenvolvido. Caso um motor melhor fosse utilizado, além do limite de velocidades de operação aumentar, o erro médio da precisão angular do motor seria reduzido.

A interface construída, bem como o projeto do suporte, são partes relativas do projeto. Seria possível construir a máquina utilizando outra metodologia de conversão da energia do motor para a movimentação do substrato e outro tipo de suporte. Porém, o código descrito e a metodologia aplicada de calibração são partes fundamentais do equipamento, visto a especificidade do processo.

Em relação às melhorias possíveis do projeto, destaca-se a utilização de um sistema de engrenagens a fim da mudança da velocidade angular do rotor, o que permitiria a redução ou aumento da velocidade de descida do substrato, sem necessariamente ter que alterar o motor utilizado. Da mesma forma, a utilização de um motor de mais qualidade permitiria que o limite de operação da máquina fosse alterado, aumentando tanto a precisão do processo quanto o limite de velocidades.

É possível, também, tornar a interface da máquina mais completa, o que não alteraria em nada eficácia do projeto. Por exemplo, poderia ser conectado no Arduino um *display*, um teclado e uma bateria portátil, a fim da digitação dos valores ocorrer diretamente na máquina, sem necessidade de alimentação em um computador. No

entanto, como isso aumentaria bastante o custo do equipamento e aprimoraria o projeto apenas de forma estética, não foi considerado devido ao objetivo do trabalho de reproduzir um equipamento de *dip coating* de forma econômica.

5.2 Protótipo

Com a finalidade de comparar o protótipo desenvolvido com os equipamentos disponíveis no mercado, foi feito um levantamento dos preços de unidades de *dip coating* de diferentes lugares do mundo. Assim, foram selecionadas diferentes empresas para uma melhor estimativa de mercado da máquina e, na Tabela 5 abaixo, temos o resultado do levantamento. Como os preços divulgados das máquinas nos sites é feito em diferentes moedas, será utilizado um fator de conversão a fim de padronizar todos os preços na mesma moeda. Dessa forma, a cotação utilizada para o Dólar será de 1 USD – 4,0449 BRL e a cotação utilizada para a Libra Esterlina será de 1 GBP – 5,1918 BRL (dia 25 de outubro de 2019).

Tabela 5 – Preço de diferentes equipamentos de *dip coating* disponíveis no mercado.

Empresa	Preço	Fonte
VPC tm	R$ 16.159,37	https://www.vpcinc.com/Category/Dip-Coating-Machines-101.cfm
Ossila	R$ 9.345,24	https://www.ossila.com/products/dipcoater?variant=8806309101685
Samkoon	R$ 10.031,35 até R$16.988,58	https://www.alibaba.com/product-detail/200W-Multi-Station-Dip-Coater-Laboratory_62031470672.html?spm=a2700.7724857.normalList.11.3def5265YpQWv h&s=p
Xian Yima Optoelec Co.	R$ 5.258,37	https://portuguese.alibaba.com/product-detail/lom003-dip-coating-machine-withvariable-speed-1-200mm-60538794993.html
MTI Corporation	R$ 14.958,04	https://www.mtixtl.com/MilimeterGradeProgrammableDipCoaterwithDryingOven-PTLMMB01.aspx
MTI Corporation	R$ 37.601,39	https://www.mtixtl.com/MicrometerGradeProgrammableDipCoaterwithDryingOvenPTL-UMB.aspx
TECHSOFT	R$ 7.657,90	https://www.techsoft.co.uk/WorkshopEquipment/ThermoformingEquipment/DipCoatin g
Xiamen Tmax	R$ 4.651,63	https://www.alibaba.com/product-detail/EXW-Price-Desktop-Dip-Coater-Coating_62063669127.html?spm=a2700.7724857.normalList.35.286925730qZUGU

Algumas justificativas para as diferenças entre os preços são:

- Produção em diferentes lugares do mundo.
- Diferentes velocidades de operação.
- Precisão do movimento, bem como o erro percentual.
- Torque máximo do motor, traduzido como a carga suportada.
- Tamanho do tanque de substrato, visto que em certas aplicações são revestidas peças maiores do que as lâminas laboratoriais.
- Peso/tamanho máximo da amostra.
- Dimensões do equipamento.
- Voltagem (120 VAC ou 240 VAC).
- Presença de interfaces interativas.

Pode-se notar então, pela pesquisa realizada, que o preço de um equipamento de *dip coating* atualmente no mercado varia em média de R$ 4.500,00 a R$ 37.500,00, levando em conta apenas os dados levantados na Tabela 5.

O intuito dessa pesquisa realizada é de servir de comparação com o protótipo construído. Vale destacar que o valor dos equipamentos do mercado é um preço e que inclui toda a carga tributária, custo de mão de obra e lucro, por exemplo. Já o valor do protótipo criado é um custo e inclui apenas as peças necessárias. Dessa forma, temos abaixo na Tabela 6 a relação das partes e do custo unitário de cada uma.

Tabela 6 – Custo das peças necessárias para o protótipo

Peça	**Custo Unitário**
Arduino UNO R3 + Cabo USB 2.0	R$ 54,90
Kit Servo Motor SM-S4306R	R$ 89,90
Kit Jumpers Macho/Macho	R$ 10,90
Kit Jumpers Fêmea/Fêmea	R$ 8,90
Suporte Articulado de mesa F50N	R$ 139,90
Suporte rígido de madeira em L	R$ 50,00
Parafusos, linha e rodelas	R$ 5,00
Total:	R$ 359,50

Assim, fazendo a soma dos valores da tabela, podemos notar que o custo total do equipamento foi de R$ 359,50. O preço não leva em conta a mão de obra, visto que tanto a montagem eletrônica quanto a inserção do código no Arduino são processos muito rápidos.

A não consideração do valor da mão de obra se deve ao fato que, utilizando as mesmas peças e o mesmo modelo de Servo motor, basta juntar as peças e colocar o código que já está pronto na Figura 16 que o equipamento funcionará com a mesma precisão descrita nos resultados, o que torna o cálculo do valor da mão de obra muito relativo.

Dessa forma, utilizando o mesmo modelo de motor, a construção do equipamento é algo que leva minutos. Porém, se for alterado o modelo de motor, será necessário uma nova calibração (a fim da descrição da relação entre *input* e velocidade real), o que levará muito mais tempo do que apenas a montagem do mesmo protótipo descrito nesse trabalho (mas, um fator positivo é, uma vez feita a calibração de outro modelo de motor, as novas equações de calibração podem ser utilizada repetidamente para o mesmo modelo em diferentes unidades de *dip coating*, o que garante a reutilização do código, bem como do descrito nesse trabalho). Assim, com a utilização de diferentes motores para as calibrações, é possível otimizar o projeto a fim de permitir saber qual a melhor decisão de motor de se utilizar dependendo da velocidade de substrato desejada e do carregamento presente.

6 CONCLUSÃO

O projeto de equipamento de *dip coating* descrito teve a eficácia comprovada com a aplicação no protótipo e com as medições de calibração, que permitiram reduzir muito a imprecisão do equipamento.

Quanto ao custo, destaca-se que o projeto desenvolvido possui um preço mais de 10 vezes menor que os protótipos mais baratos da pesquisa de mercado feita (chegando a mais de 100 vezes menor que os mais caros). Mesmo apesar das limitações do projeto, a diferença de preço é notável e, com certos aprimoramentos comentados no tópico anterior, é possível melhorar a qualidade do equipamento sem necessariamente precisar de muito mais dinheiro investido.

Destaca-se, então, a importância do estudo dos processos e da otimização, pois mesmo contribuições pequenas no meio tecnológico, quando aplicadas em escala industrial, permitem alterações consideráveis, tanto para a geração de lucros quanto para a redução de desperdícios.

Embora o projeto desenvolvido seja de um equipamento de *dip coating*, qualquer outro tipo de máquina que necessite de cargas mecânicas parecidas ou movimentações do mesmo tipo pode se beneficiar do estudo, visto que a aplicabilidade do código é universal e a utilização de Servo motores e Arduino é algo cada vez mais comum nos equipamentos tecnológicos da atualidade.

7 REFERÊNCIAS

1. BRINKER, C. J. et al. **Fundamentals of sol-gel dip coating**. Thin solid films, volume 201, n. 1, p. 97-108. Albuquerque: 1991.

2. BRINKER, C. J.; HURD, A. J. **Fundamentals of sol-gel dip-coating**. Journal de Physique lll, v. 4, n. 7, p. 1231-1242, 1994.

3. CRUZ, F. J. M. et al. **Projeto de melhoria de equipamento de dip coating**. Tese de doutorado. Brasil: 2014.

4. DARHUBER, A. A. et al. **Selective dip-coating of chemically micropatterned surfaces**. Journal of Applied Physics, vol. 88, issue 9, p. 5119-5126. New Jersey: 2000.

5. FARINELLI, F.; **Conceitos básicos de programação orientada a objetos**. Instituto Federal Sudeste de Minas Gerais. Brasil: 2007.

6. GALADIMA, A. A.; **Arduino as a learning tool**. 11th Internacional Conference on Eletronics, Computer and Computation (ICECCO). IEEE, 2014, p. 1-4.

7. GEFFCKEN, W.; BERGER, E. **Deutsches Reichspatent 736411**: 1939

8. HIRATSUKA, R. S.; SANTILLI, C. V.; PULCINELLI, S. H.; **O processo sol-gel: uma visão físico química**. Química Nova. São Paulo: Sociedade Brasileira Química, v. 18, n. 2, p. 171-180, 1995.

9. MCROBERTS, M. **Arduino básico.** Novatec 2ª ed, 2015.

10. MORETO, M. **Controle de servomotor**. Monografia de conclusão de curso:

    Engenharia elétrica, Universidade São Francisco de Itatiba, 2007.

11. OLIVEIRA, A. R. M.; ZARBIN, A. J. G. **Um procedimento simples e barato para a construção de um equipamento "dip-coating" para deposição de filmes em laboratório**. Química Nova, v. 28, no. 1, São Paulo: 2003.

12. SCRIVEN, L. E. **Physics and applications of dip coating and spin coatin**g**.**

    MRS Online Proceedings Library Archive, v. 121, 1988.

13. SOUZA, A. R. et al. **A placa Arduino: uma opção de baixo custo para experiências de física assistidas pelo PC**. Revista Brasileira de Ensino de Física, v.33, n.1, p. 01-05, 2011.

14. VAN DE STRAETE, H. J. et al. **Servo motor selection criterion for mechatronic applications**. IEEE/ASME Transactions on mechatronics, v. 3, n. 1, p. 43-50, 1998.

Universidade de São Paulo

Engenharia de Petróleo – Escola Politécnica

Número: 9017102 USP **Data: 05/12/2019**

Projeto e construção de um protótipo de *dip coating* utilizando a plataforma de prototipagem Arduino.

Leonardo Foncesca Finardi

Orientador: Prof. Dr. Jean Vicente Ferrari

Artigo Sumário referente à disciplina PMI1096 – Trabalho de Formatura para Engenharia de Petróleo II
Este artigo foi preparado como requisito para completar o curso de Engenharia de Petróleo na Escola Politécnica da USP.

Resumo

O *dip coating* é um processo que consiste na aplicação de revestimentos finos em sólidos de diversas geometrias por meio da realização de ciclos de mergulho e evaporação em soluções especificas preparadas previamente, as quais após a secagem manifestam sua natureza química diferente na camada aplicada, permitindo a manipulação das propriedades superficiais do sólido de acordo com o fluido usado para o revestimento. O seguinte trabalho tem como objetivo tanto desenvolver um projeto de um equipamento de *dip coating* econômico quanto de construir de um protótipo a fim de comprovar a funcionalidade do projeto. Para isso, é utilizada a plataforma de *hardware* livre Arduino para a prototipagem e programação do equipamento, bem como um Servo motor de rotação contínua para a realização da carga mecânica necessária. São abordadas as teorias fundamentais para o entendimento e descrição do processo, os motivos por trás da escolha das peças (bem como suas aplicações e funcionalidades), a metodologia das ligações eletrônicas, a programação necessária no *Arduino* e o processo de calibragem de velocidades. O código desenvolvido em conjunto com as calibrações permitiram produzir um equipamento de *dip coating* que opera com precisão em velocidades de 10 a 40 milímetros por segundo.

Abstract

Dip coating is a process that consists on the controlled application of thin coating films in solids of various geometries through the performing of dip and evaporation cycles in an preprepared specific solution, which after drying manifests the different chemical nature in the coat aplied, allowing the manipulation of the superficial properties from the solid according to the fluid used for the coating. The objectives of the work are develop a project of an economic dip coating machine and construct a prototype to prove the project functionality. For this it will be used the free hardware platform ‘Arduino’ for the prototyping and programming and a continuous rotation ‘Servo motor’ for perform the mechanical load necessary. This work

describe the fundamental theory for the understanding and description of the process, the motivation beyond the parts definition (as well as the applications and functionalities), the methodology of the eletronic disposition, the necessary programming in the Arduino and the process of velocity calibration. The code developed with the velocity calibration enabled to produce a dip coating unit that operate with precision in velocities from 10 to 40 millimeters per second.

1. Introdução

O processo de *dip coating*, ou revestimento por imersão, é uma maneira simples e antiga de deposição em substratos de filmes líquidos finos e uniformes que por solidificação geram o revestimento. O fluxo básico é laminar e a espessura do filme é definida pela competição entre as forças viscosas, forças capilares (tensão superficial) e gravidade. Quanto mais rápido o substrato é submergido, mais espesso se torna o filme depositado. Isso pode ser compensado pelo uso de soluções voláteis e processos rápidos de secagem (SCRIVEN, 1988).

O *dip coating* é um processo sem desperdícios, de baixo custo, com boa repetibilidade e capaz de oferecer um excelente controle da espessura do revestimento. Como pontos negativos, destaca-se a impossibilidade de assegurar a uniformidade da espessura ao longo de toda a placa, bem como a necessidade de um eficiente sistema/método de controle da viscosidade do fluido, dado esta alterar-se com a natural evaporação de solvente do fluido (JONES, 2010 apud CRUZ, 2014).

2. Objetivos

Os objetivos do trabalho são tanto de criar um projeto de equipamento de *dip coating* econômico quanto de construir um protótipo representativo, a fim de validar a eficiência do projeto.

3. Materiais e métodos

Como o equipamento descrito pode ser construído de diferentes formas, o foco do trabalho será tanto para o desenvolvimento de um projeto de *dip coating* de baixo custo quanto para a construção de um protótipo representativo. Para a construção do protótipo será utilizada uma placa Arduino UNO R3. O Arduino foi criado em 2005 como um *hardware* livre. A placa é constituída por um micro controlador Amtel e circuitos de entrada/saída. O dispositivo possui saídas PWM[9], analógicas e digitais.

[9] O PWM (*Pulse With Modulation*) é uma técnica utilizada para obter resultados analógicos por meio de entradas digitais (geração de onda quadrada).

Em conjunto com o Arduino é necessário um motor, que é responsável pela movimentação vertical do substrato (sólido) a ser revestido. O modelo de motor escolhido é de um Servo motor (SM-S4306R) de rotação contínua, que pode operar a 5 volts e possui um torque de 6,2 kg

Para que se teste o código, é necessário conectar a placa Arduino à alimentação (no caso o computador) e fazer a conexão dos fios entre o Servo motor e o Arduino. A funcionalidade do motor depende diretamente da programação feita na placa, a qual manipula a energia dos fios conectados. Para a conexão do Servo motor ao Arduino, deverão ser utilizados *Jumpers*[10]. O cabo preto é conectado ao pino GND do Arduino, que significa *ground* e é a entrada do terra. O cabo vermelho é conectado ao pino 5V, que é o cabo de alimentação do motor. O cabo branco é ligado a qualquer porta de sinal PWM do Arduino.

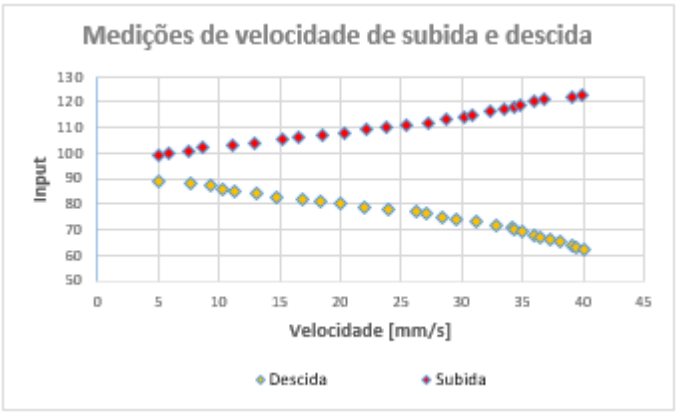
A calibração de velocidades depende das funções da biblioteca <Servo.h>[11]. A função .*write* insere um *input*[12] de velocidade no motor que mantém o comando acionado pelo tempo estipulado na função, que é colocada na linha do código logo abaixo.

Então, para descobrir a relação entre *input* e velocidade real de substrato, é utilizada a fórmula de velocidade constante (variação de espaço dividida pela variação de tempo, pois a velocidade inserida em um *write* é realmente constante durante o *delay*), pois como o tempo será um dado inserido, basta saber a distância que a haste de metal desce verticalmente nesse tempo inserido para se calcular a velocidade real de descida (e, da mesma forma, pra subida).

4. Resultados

O resultado experimental está exposto na Figura 1 em forma de gráfico.

Figura 1 – Dados experimentais da relação entre *input* e velocidade do substrato

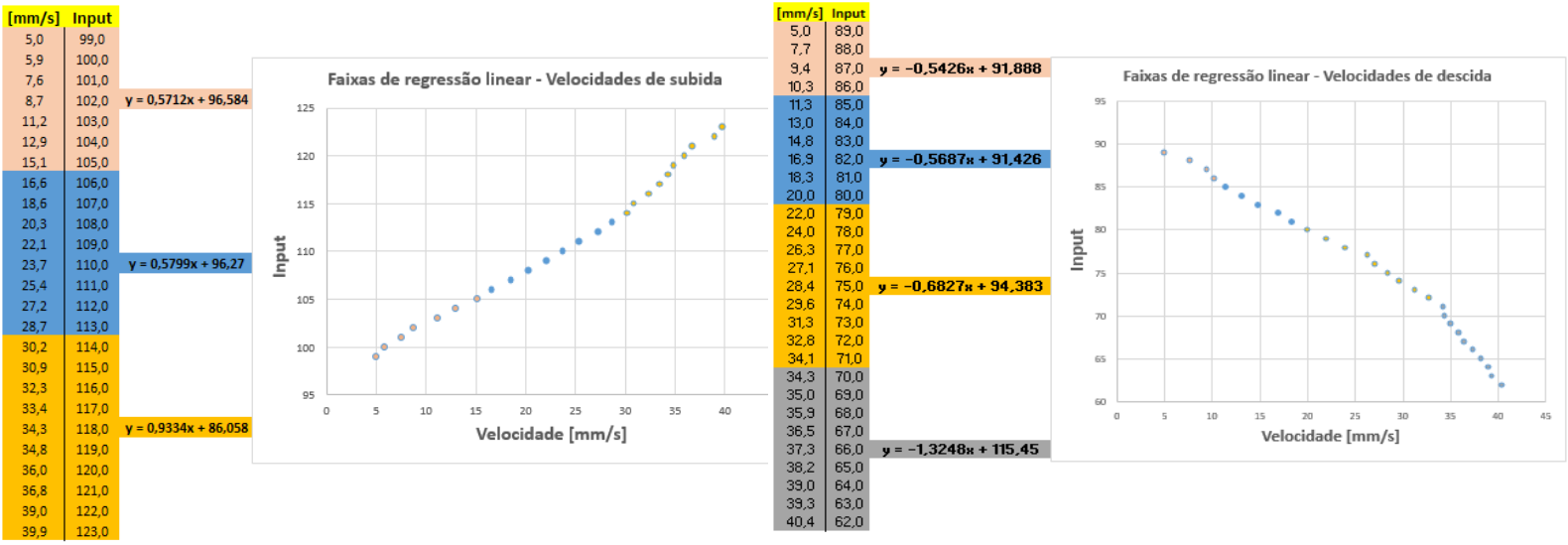


Fonte: Próprio autor

[10] *Jumper* é a denominação do fio utilizado em prototipagem.
[11] A biblioteca <Servo.h> é uma biblioteca própria do Arduino para Servo motores. Possui funções que definem a operação do motor.
[12] Valor de 0 a 180 que define a velocidade do motor, sendo 90 o repouso e as faixas de 0-89 e 91-180 velocidades iguais em módulo, mas opostas em sentido (valores 0 e 180 são teoricamente as maiores velocidades).

Assim, foram realizadas regressões lineares (com as ferramentas do *software* Excel), em busca de uma relação no formato de equação de reta entre o *input* e a velocidade real (para se inserir no código). Na Figura 2 estão definidas as faixas criadas para as velocidades de subida e descida.

Figura 2 – Definição das faixas para regressão linear nas velocidades de descida



Fonte: Próprio autor

Assim, as equações obtidas com as regressões lineares para representar a relação entre o *input* do motor e a velocidade real do substrato são inseridas no código, para que o operador possa digitar a velocidade desejada e a programa faça a conversão correta e envie ao Servo motor o *input* correto correspondente. Com o código incluindo a relação entre *input* e velocidade de substrato e uma correção aplicada para cada faixa de regressão, mede-se a precisão do equipamento e das regressões aplicando todas as velocidades para uma distância referência de 5 cm (arbitrária, é quanto se deseja que o equipamento realize de descida vertical).

Um problema notado durante o procedimento experimental, é a instabilidade de operação do motor para as velocidades mais baixas, devido provavelmente à qualidade do modelo. Dessa forma, para não prejudicar a precisão do projeto, será considerado como *range* de operação apenas as velocidades entre 10 e 40 milímetros por segundo.

O protótipo construído que foi utilizado para as medições experimentais e calibração da velocidade do Servo motor está exposto na Figura 4. Destaca-se o fato do cabo que liga o Arduino na entrada USB do computador não estar conectado a fim de uma melhor visualização do equipamento, que independe do computador utilizado (contanto que possua o software do Arduino).

Figura 4 – protótipo de *dip coating* construído com base no projeto desenvolvido.

Fonte: Próprio autor

5. Discussão

A faixa de velocidade de operação com precisão é de 10 a 40 milímetros por Segundo (de 5 a 9 milimetros por segundo a precisão das distâncias reduz muito, inviabilidando o uso). Quanto à precisão do projeto, nota-se que as distâncias inseridas no programa, quando realizadas pelo equipamento, reproduzem o valor digitado para a descida e para a subida respectivamente com um desvio padrão de 0,1809 e 0,2130 e uma variância de 0,0327 e 0,0453. Nota-se, assim, a uniformidade dos dados e a boa precisão do projeto desenvolvido. Caso um motor melhor fosse utilizado, além do limite de velocidades de operação aumentar, o erro médio da precisão angular do motor seria reduzido. Fazendo a soma dos valores unitários das peças do protítpo, nota-se que o custo total do protótipo foi de R$ 359,50.

Em relação às melhorias possíveis do projeto, destaca-se a utilização de um sistema de engrenagens a fim da mudança da velocidade angular do rotor, o que permitiria a redução ou aumento da velocidade de descida do substrato, sem necessariamente ter que alterar o motor utilizado. Da mesma forma, a utilização de um motor de mais qualidade permitiria que o limite de operação da máquina fosse alterado, aumentando tanto a precisão do processo quanto o limite de velocidades.

6. Conclusão

O projeto de equipamento de *dip coating* descrito teve a eficácia comprovada com a aplicação no protótipo, que possui boa precisão (desvio padrão da ordem de 0,1 e variância da ordem de 0,01 da distância inserida no código, tendo o *range* de velocidades prático de 10 a 40 milímetros por segundo).

O custo do protótipo desenvolvido (considerando apenas as peças) foi de R$ 359,90, sendo que o projeto é replicável e sua montagem é muito simples uma vez possuindo as calibrações de velocidades (que são diretamente relacionadas ao modelo de motor adotado).

Embora o projeto desenvolvido seja a respeito de *dip coating*, qualquer outro tipo de equipamento que necessite de cargas mecânicas parecidas ou movimentações do mesmo tipo pode se beneficiar do estudo, visto que a aplicabilidade do código é universal e a utilização de Servo motores e Arduino é algo cada vez mais comum nos equipamentos tecnológicos da atualidade.

7. Referências

CRUZ, F. J. M. et al. **Projeto de melhoria de equipamento de dip coating**. Tese de doutorado. Brasil: 2014.

SCRIVEN, L. E. **Physics and applications of dip coating and spin coatin**g**.** MRS Online Proceedings Library Archive, v. 121, 1988.