UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

ARTUR PIANA BROILO - 00302491

BOMBA DE INFUSÃO DE BAIXO CUSTO

Porto Alegre
2024

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

ARTUR PIANA BROILO - 00302491

BOMBA DE INFUSÃO DE BAIXO CUSTO

Trabalho de Conclusão de Curso (TCC-CCA) apresentado à COMGRAD-CCA da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de *Bacharel em Eng. de Controle e Automação* .

ORIENTADOR:
Prof. Dr. Mário Roland Sobczyk Sobrinho

Porto Alegre
2024

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

ARTUR PIANA BROILO - 00302491

BOMBA DE INFUSÃO DE BAIXO CUSTO

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção dos créditos da Disciplina de TCC do curso *Eng. de Controle e Automação* e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Mário Roland Sobczyk Sobrinho, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Banca Examinadora:

Prof. Dr. Mário Roland Sobczyk Sobrinho, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Fabiano Disconzi Wildner, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Walter Fetter Lages, UFRGS
Doutor pelo Instituto Tecnológico de Aeronáutica (ITA) – São José dos Campos, Brasil

_____
Alceu Heinke Frigeri
Coordenador de Curso
Eng. de Controle e Automação

Porto Alegre, Agosto 2024

RESUMO

Bombas de infusão são usadas tanto no ambiente científico para pesquisa em microfluídica, como a simulação de órgãos em aplicações *patient-on-a-chip* (Q. et al., 2023), como no ambiente médico, tanto para terapia intravenosa quanto para gotejar medicamentos em um equipo. Apesar da aparente simplicidade das partes constituintes da bomba de infusão, os equipamentos encontrados comercialmente têm custo significativamente elevado. Este trabalho tem por objetivo estudar os principais tipos de bombas de infusão utilizados comercialmente, seus mecanismos de acionamento e suas partes integrantes. Em sequência, propõe-se desenvolver e construir um protótipo funcional do equipamento de baixo custo. A versão proposta nesse trabalho da bomba de infusão deve ser capaz de atender aos requisitos básicos de um laboratório de pesquisa, onde uma determinada vazão e uma quantidade total de fluido são previamente configurados e seguidos dentro de um erro aceitável pelo equipamento.

Palavras-chave: Bomba de Infusão, Controle, Microfluídica.

ABSTRACT

Infusion pumps are used in both scientific environments for research in microfluidics, such as organ simulation in patient-on-a-chip applications (Q. et al., 2023), and medical environments for both intravenous therapy and medication drip. Despite the apparent simplicity of the constituent parts of an infusion pump, commercially available equipment has a significantly high cost. This work aims to study the main types of commercially used infusion pumps, their actuation mechanisms, and their integral parts. Subsequently, it proposes to develop and construct a functional low-cost prototype of the equipment. The version of the infusion pump proposed in this work should be capable of meeting the basic requirements of a research laboratory, where a specific flow rate and total fluid volume are pre-configured and followed within an acceptable error margin by the equipment.

Keywords: Syringe Pump, Control Engineering, Microfluidics.

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo discutem-se os principais tipos do equipamento bomba de infusão, seu funcionamento, e aplicações na indústria médica e no meio científico. Levanta-se a revisão bibliográfica de trabalhos conduzidos tanto no Brasil como no exterior sobre o estudo e desenvolvimento de protótipos de baixo custo do equipamento.

O equipamento bomba de infusão é responsável por garantir uma vazão controlada e precisa de fluido por um determinado período de tempo através de pressão positiva (DIAS, 2019). No ambiente médico, o fluído é geralmente composto por medicamentos ou nutrientes (JUAREZ et al., 2016), enquanto no ambiente laboratorial químico ou biológico, o fluído pode conter reagentes, microrganismos ou algas (AMARANTE et al., 2019) (BOOESHAGHI et al., 2019).

A necessidade da bomba de infusão surge quando o tempo de vazão controlada do fluido é prolongado e não há disponibilidade de operação humana contínua, ou quando essa disponibilidade tem valor proibitivo. Ainda, o equipamento é necessário quando a precisão de vazão é elevada, onde o erro decorrente da operação humana não atende aos requisitos da aplicação (ARÊDES et al., s.d.[a]). No contexto de Engenharia Biomédica, o sistema de administração intravenosa de medicamentos utilizando bombas de infusão difere-se do sistema de administração tradicional no âmbito de que a pressão é gerada por um mecanismo, e não pela gravidade. Esse mecanismo pode ser um motor elétrico acionando uma seringa, peristáltico, ou por pistão. Ainda, é possível utilizar a bomba de infusão para controlar a taxa de gotejamento do medicamento ao equipo. Em Estabelecimentos Assistenciais de Saúde (EAS), as bombas de infusão são identificadas em três tipos: bomba de infusão volumétrica, onde o medicamento é administrado de forma intravenosa e a vazão é selecionada pelo operador e indicada em unidades de volume por tempo ; bomba de seringa, onde a pressão para vazão é gerada pelo acionamento de uma ou mais seringas, usualmente por motor de passo, tanto para terapia intravenosa quanto para gotejamento do medicamento no equipo para mistura com soro fisiológico ; e bomba de infusão ambulatorial, onde a vazão é programada em sequências tanto contínuas quanto não-contínuas (ALVES, 2002) (DIAS, 2019).

As bombas de infusão desempenham um papel crescente na pesquisa em microfluídica (BOOESHAGHI et al., 2019) (AKKOYUN; OZÇELIK, 2020), uma área que envolve o estudo e manipulação de fluidos em escalas sensivelmente pequenas, frequentemente em canais com dimensões de micrômetros. Elas são essenciais para garantir a entrega precisa e controlada de vazões ultra baixas necessários para experimentos de alta precisão. Na microfluídica, as bombas de infusão são usadas para introduzir reagentes, células, microrganismos e outras substâncias em dispositivos microfluídicos, facilitando a investigação de fenômenos biológicos, químicos e físicos em escalas microscópicas (LAKE; HEYDE; RUDER, 2017). A capacidade de controlar com precisão a vazão e a pressão dos fluidos torna as bombas de infusão ferramentas indispensáveis para a execução de experimentos

complexos e para o desenvolvimento de novas tecnologias em diagnóstico, terapias médicas e síntese química em microescala (AMARANTE et al., 2019).

A motivação desse trabalho é derivada do potencial de reduzir custos médicos e científicos através do desenvolvimento de uma bomba de infusão de baixo custo, de *software* aberto, e cujos componentes sejam de fácil acesso em uma região extensa do Brasil. Para isso, estudam-se os tipos de bombas de infusão, os principais mecanismos de aplicações, seu funcionamento e aplicabilidade.

1.1 Estado da Arte

No Brasil, Arêdes desenvolveu uma bomba de infusão de baixo custo de mecanismo peristáltico utilizando apenas componentes do mercado nacional. A bomba opera em malha fechada através da estimação da vazão por detecção do tempo entre gotas com sensor infravermelho, garantindo alta precisão de operação. (ARÊDES et al., s.d.[b]) (ARÊDES et al., s.d.[a]). Em 2002, Márcio Alexandre de Castro Alves escreveu sobre a operação de bombas de infusão no âmbito brasileiro, destacando os principais tipos, componentes essenciais, problemas encontrados e avaliações de segurança relevantes. (ALVES, 2002) Em 2019, Dias projetou e construiu uma bomba de infusão, do tipo bomba de seringa, em malha aberta operada remotamente para aplicação em biossensores (DIAS, 2019). Em 2014, Amorin fez um trabalho similar com conexão por *bluetooth* (DE AMORIN, 2014).

O interesse crescente em microfluídica combinado com avanços tanto em Design Assistido por Computador *(CAD)* quanto em impressão 3D aumentaram significativamente o número artigos descrevendo o desenvolvimento e implementação de tecnologias de infusão, em particular de bombas de seringa, de baixo custo. Por ser um campo recente, a maior parte da literatura é encontrada em artigos do exterior. Por exemplo, Lake desenvolveu uma bomba de seringa controlada em laço fechado por um controlador *PID* (Proporcional, Integral e Derivativo) para aplicações microfluídicas (LAKE; HEYDE; RUDER, 2017). Nesse trabalho, sensores de vazão piezoresistivos foram usados para medir e corrigir desvios de vazão dos *set-points* determinados para que a vazão pretendida fosse próxima à vazão real. Um microcontrolador *Arduino* e um *driver* de motor de passo foram empregado para controlar a vazão da seringa com erro de pressão de 1%.

No trabalho citado de (WIJNEN et al., 2014), o autor concebeu e produziu uma bomba de seringa de código aberto usando software de Design Assistido por Computador *(CAD)* e impressão 3D. Para controlar um motor de passo, utilizou-se um *Raspberry Pi* em conjunto com um *driver* Pololu A4988. O movimento linear, encarregado de deslocar o êmbolo da seringa para extrair o fluido, foi gerado por meio da conexão de uma haste roscada ao motor de passo. Esse mecanismo rotaciona a haste, movimentando linearmente uma plataforma vertical através de um acoplamento com uma porca embutida na parte inferior da plataforma. Como resultado, a plataforma aplica pressão no êmbolo, permitindo a liberação do fluido. Além disso, os autores desenvolveram uma interface gráfica baseada em uma página web para permitir a supervisão e configuração da bomba pelo usuário.

Em um trabalho diferente, foi apresentada uma bomba de seringa acionada por pressão que não requer eletricidade ou microcontroladores para operar (DYLAN et al., 2017). Uma válvula reguladora de pressão e uma fonte constante de ar pressurizado foram utilizadas para mover o êmbolo da seringa. Este sistema forneceu vazões de 0,5 a 8 mL/h com desvio menor que 10% em relação às vazões desejadas.

Uma bomba de seringa de baixo custo direcionada a infusões intravenosas em

hospitais foi apresentada em (JUAREZ et al., 2016). O princípio de funcionamento dessa bomba foi baseado na liberação controlada de um mecanismo de mola. O mecanismo é comprimido através de um motor de passo acionado por um microcontrolador. A eficácia desta bomba de seringa foi posteriormente testada através da administração intravenosa de sulfato de magnésio em pacientes em condição de Pré-eclâmpsia (SKERRETT et al., 2017). No total, 22 pacientes mulheres foram incluídas em um estudo de 466 horas, onde erros de vazão menores que 2% foram registrados. As vazões obteníveis para este sistema foram entre 3 e 60 mL/hora, o que é relativamente alto para aplicações microfluídicas tendo em vista os requisitos baixos dessa aplicação como por exemplo em (MENGXI et al., 2019).

Outros projetos são encontrados em (AMARANTE et al., 2019), onde os autores desenvolvem uma bomba com administração de volumes configurável para aplicação em fluidos de recompensa para roedores, em (BOOESHAGHI et al., 2019), unindo uma bomba de infusão feita em laboratório com um sistema microfluídico monitorado por microscópio, em (PUSCH; HINTON; FEINBERG, 2018) para projeto de bomba de infusão para impressora 3D, e em (AKKOYUN; OZÇELIK, 2020), onde múltiplas bombas de infusão são controladas por um único sistema. A integração do equipamento controlado por uma Raspberry Pi com uma tela sensível ao toque foi realizada em (GARCIA; LIU; DERISI, 2018).

Por ser um campo de estudo de emergência recente e de aplicações variadas, não existem regulações no Brasil quanto a normas para bombas de infusão voltadas para a microfluídica. Todavia, no Brasil, a bomba de infusão para aplicações na área da saúde é considerada um equipamento eletromédico (EEM) e é fortemente regulamentada (ALVES, 2002). A Lei 6360 de 23 de setembro de 1976 exige o registro e inspeção desses equipamentos pelo Ministério da Saúde, enquanto que a comercialização é regulamentada pela portaria 2043 de 12 de dezembro de 1994, também do Ministério da Saúde, onde o Sistema Nacional de Metrologia, Normalização e Qualidade Industrial (Sinmetro) foi adotado para verificação da qualidade desses produtos. As principais normas técnicas para o desenvolvimento e comercialização da bomba de infusão são: ABNT NBR IEC 60101-1: Equipamento Eletromédico Parte 1: Requisitos essenciais para segurança básica e desempenho essencial (ABNT, 2016); e a ABNT NBR IEC 60101-2-24: Equipamento Eletromédico Parte 2-24: Requisitos particulares para segurança básica e desempenho essencial de bombas de infusão e de controladores de infusão (ABNT, 2015). Ambas as normas são vastas e orientadas para produtos comerciais, dessa forma, a realização da totalidade de testes e certificações necessárias foge do escopo da confecção do protótipo do equipamento desenvolvido neste trabalho.

1.2 Objetivos Específicos

Os objetivos deste trabalho são os seguintes:

1. Estudar diferentes sistemas eletromecânicos de bombas de infusão, identificar os componentes e escolher o mecanismo mais adequado considerando desempenho, complexidade e custo.

2. Definir e especificar os componentes e procedimentos necessários para a montagem do protótipo, incluindo as partes mecânicas, modelagem, seleção de sensores de vazão, projeto do circuito elétrico e projeto do controlador.

3. Desenvolver o *firmware* para acionar o atuador conforme especificações e possibilitar a comunicação com outros dispositivos, além de criar um programa supervisório para comunicação com a bomba.

4. Adquirir as peças e montar o protótipo, verificar a funcionalidade do equipamento e realizar ensaios em malha aberta e fechada para avaliar o desempenho e aperfeiçoar o controlador.

Neste trabalho, é apresentado o processo de desenvolvimento do equipamento, incluindo a montagem mecânica, diagrama elétrico, funcionamento do *firmware*, Supervisório, leis de controle propostas e desempenho do protótipo. O *hardware* é disponibilizado de forma aberta, assim como o *firmware* e o Supervisório, seguindo a metodologia *FOSS (Free and Open Source Software)*.

FOSS é um *software* disponibilizado como código-fonte (código aberto) que pode ser usado, estudado, copiado, modificado e redistribuído sem restrições, ou com restrições que apenas garantam que outros destinatários tenham os mesmos direitos sob os quais o *software* foi obtido (SILVA; COUTINHO; COSTA, 2023). Segundo dados de 2014, 94% dos 500 mais poderosos supercomputadores do mundo, 75% dos 10000 sites com mais acessos e 98% das empresas usam *software* aberto (WIJNEN et al., 2014). Um estudo sobre os fatores que influenciam a escolha do *FOSS* em países em desenvolvimento classifica o seu uso como um pilar chave para organizações tanto públicas quanto privadas, principalmente pois acelera a descoberta de novas tecnologias e promove a inovação (SILVA; COUTINHO; COSTA, 2023).

1.3 Organização do Trabalho

O Capítulo 2 trata dos principais tipos de Bomba de Infusão, seu funcionamento, e a escolha do mecanismo melhor adequado para o projeto. O Capítulo 3 detalha o desenvolvimento do protótipo, incluindo a derivação de um modelo simples a partir dos princípios físicos envolvidos, os componentes escolhidos e as relações entre eles, o *firmware* desenvolvido e o programa Supervisório desenvolvido. O Capítulo 4 mostra o desempenho do equipamento construído tanto em malha aberta quanto em malha fechada, detalhando os controladores projetados, e o Capítulo 5 discute as conclusões alcançadas com o projeto e sugestões para trabalhos futuros. O Apêndice A contém as tabelas de peças mecânicas e elétricas necessárias para a reprodução da bomba, assim como instruções de montagem e operação. O diagrama elétrico e imagens do protótipo construído também são inclusos. O Apêndice B apresenta o código do *firmware* da bomba, e o Apêndice C contém o código do Supervisório.

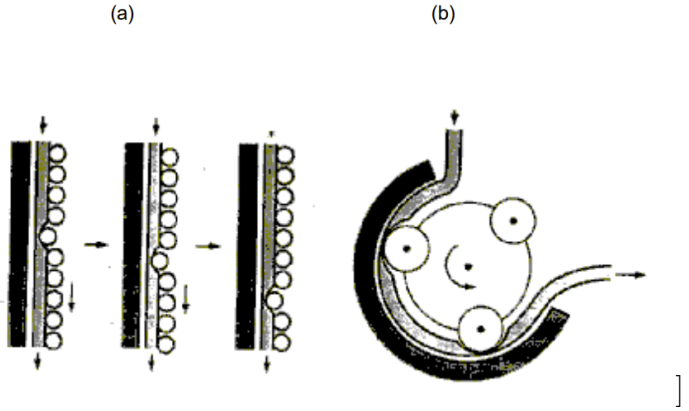
2 BOMBA DE INFUSÃO - FUNCIONAMENTO

Diferentes tipos de bombas de infusão apresentam desempenhos de velocidade e precisão de infusão distintos, essa diferença é derivada de como o fluido é impulsionado ao longo do equipo. A principal característica diferenciadora do equipamento é o mecanismo de geração de vazão, onde os três principais tipos são: peristáltico, pistão e bomba de seringa(DIAS, 2019). A escolha do mecanismo do protótipo é uma função do escopo da aplicação, e de requisitos de desempenho, custo, e integração com outros componentes. A seguir, detalham-se os três tipos principais de mecanismos de bomba de infusão encontrados comercialmente em ambientes médicos (ALVES, 2002).

2.1 Direcionamento por Mecanismo Peristáltico

Nesse mecanismo, o tubo ou equipo por onde o fluido a ser infundido percorre é comprimido de forma cadenciada por uma série de roletes rotatórios (peristáltico rotativo), ou por uma série de pulsos (peristáltico linear). Essa ação é responsável por criar um movimento peristáltico que impulsiona o fluido ao longo do tubo. A Figura 1 ilustra esse funcionamento, onde os mecanismos são frequentemente direcionados por um motor de passo controlado por um microcontrolador (DIAS, 2019).

Figura 1: *Direcionamento peristáltico (a) Pulsos (b) Roletes*



Fonte: (MOYLE; DAVEY, 2000)

As bombas de infusão peristálticas são dispositivos amplamente utilizados na administração de fluidos médicos devido à sua capacidade única de evitar a contaminação do fluido. O mecanismo peristáltico faz com que o líquido a ser infundido não entre em contato direto com o mecanismo da bomba, mantendo a esterilidade e evitando a

contaminação cruzada, o que é essencial em ambientes médicos e laboratoriais (KLESPITZ; KOVÁCS, 2014).

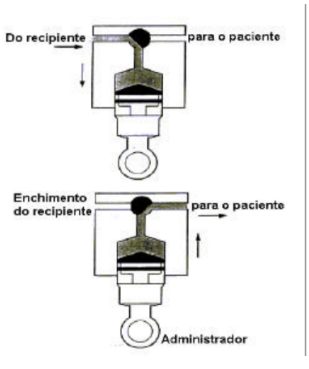
Um dos principais benefícios das bombas peristálticas é a sua manutenção simples. Como o desgaste ocorre principalmente no tubo, este pode ser facilmente substituído, garantindo a longevidade do equipamento e a continuidade do tratamento sem a necessidade de interrupções prolongadas. Além disso, as bombas peristálticas têm boa confiabilidade e precisão média na administração de fluidos, podendo ser utilizadas em uma variedade de aplicações, como a administração de nutrientes e medicamentos e fluidos intravenosos em pacientes hospitalizados, bem como em laboratórios de pesquisa e na indústria farmacêutica para a transferência segura de líquidos sensíveis. Outra vantagem das bombas de infusão peristálticas é a sua flexibilidade em termos de tipos de fluidos que podem manejar. Elas são capazes de infundir uma ampla gama de líquidos, incluindo soluções viscosas, suspensões, e fluidos contendo partículas, sem comprometer a precisão ou a segurança do processo (KLESPITZ; KOVÁCS, 2014).

Em contrapartida, a principal limitação da bomba peristáltica é a vazão pulsada de fluido. A compressão periódica do tubo causa variações na taxa de fluido infundido, o que não é ideal em situações que requerem uma dosagem significativamente precisa, como na administração de medicamentos com margens terapêuticas estreitas, ou na infusão de chips microfluídicos. Outra desvantagem potencial das bombas peristálticas é a baixa pressão disponível quando comparada ao mecanismo por pistão e ao mecanismo por bomba de seringa. Como o tubo é repetidamente comprimido e descomprimido para gerar a vazão, pressões elevadas aumentam a fadiga e podem levar ao seu rompimento (KLESPITZ; KOVÁCS, 2014).

2.2 Direcionamento por Pistão ou Diafragma

O mecanismo da Figura 2 funciona em duas fases. Na primeira fase, o movimento para baixo do pistão direciona rapidamente o fluido de uma bolsa reservatória para um recipiente chamado diafragma. Na segunda fase, o movimento para cima do pistão impulsiona o fluido com velocidade regulada para o paciente. Uma válvula é implementada para garantir o sentido correto de vazão (DIAS, 2019).

Figura 2: *Mecanismo de direcionamento por pistão ou diafragma.*
(a) Pulsos (b) Roletes



Fonte: (MOYLE; DAVEY, 2000)

As bombas de infusão por pistão ou diafragma são conhecidas por sua alta precisão e controle rigoroso sobre a taxa de infusão (ALVES, 2002), tornando-as ideais para a

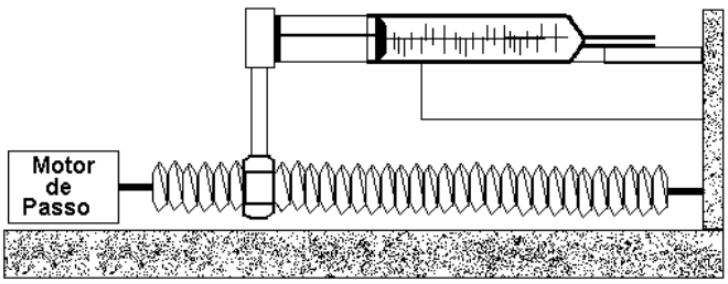
administração de fluidos que requerem dosagens muito precisas. A capacidade de manipular uma variedade de fluidos, desde soluções aquosas até líquidos mais viscosos, garante flexibilidade no tratamento de diferentes condições médicas. O design mecânico dessas bombas proporciona uma operação confiável e consistente, minimizando a variação na taxa de infusão (MOYLE; DAVEY, 2000).

No entanto, as bombas de pistão ou diafragma apresentam desvantagens. Uma das principais desvantagens é a complexidade e o custo. Estas bombas tendem a ser mais complexas e caras em comparação com outros tipos de bombas de infusão, resultando em custos iniciais mais altos e a necessidade de manutenção especializada (MANDEL, 2018). Além disso, devido ao seu design mecânico, essas bombas podem ser maiores e menos portáteis. A presença de componentes que podem se desgastar com o tempo, como pistões e válvulas, necessitam de manutenção regular para garantir um desempenho ótimo e evitar falhas durante a infusão. Outra desvantagem potencial das bombas de pistão ou diafragma é a sensibilidade à presença de bolhas de ar no fluido infundido. Bolhas de ar podem afetar a precisão da dose administrada, exigindo um cuidado adicional para garantir que o fluido esteja livre de ar antes da infusão. Além disso, ajustar a taxa de infusão com alta precisão pode ser mais desafiador em comparação com outros tipos de bombas de infusão, como as de seringa. Por fim, é comum haver contato direto do mecanismo da bomba com o fluido, diminuindo a esterilização e tornando o processo de manutenção e limpeza entre cada uso mais complexo.

2.3 Direcionamento por Bomba de Seringa

A bomba de seringa da Figura 3 utiliza o mecanismo de rosca sem fim. Esse mecanismo é composto por uma haste roscada e uma plataforma acoplada a haste por porca. Conecta-se a haste roscada a um motor de passo através de um acoplamento flexível para compensar desalinhamentos. Posiciona-se a plataforma de forma que ela faça contato com o êmbolo da seringa. O mecanismo funciona de forma que quando a haste é rotacionada pelo motor, a plataforma é deslocada linearmente com o êmbolo da seringa, provocando vazão de saída do fluido quando o deslocamento é em direção ao bico (DIAS, 2019).

Figura 3: *Mecanismo de geração de vazão da bomba de seringa com motor de passo e haste roscada.*



Fonte: Alves (2002)

Para aplicações em laboratório, em especial em microfluídica, o líquido vazado da seringa é injetado diretamente no chip . Em contrapartida, na área da saúde, a bomba

de seringa regula a frequência de queda por gravidade das gotas do medicamento a serem dissolvidas no soro administrado ao paciente. Em bombas mais complexas e com certificações adicionais, a bomba de seringa pode ser usada para infundir diretamente o paciente por meio intravenoso. Nesse caso, há controle de pressão e de formação de bolhas além da instrumentação da vazão. Uma das principais vantagens das bombas de seringa é a precisão que oferecem para administrar pequenas quantidades de medicamentos, o que é relevante em tratamentos controlados. Esta precisão diminui o risco de sobredosagem ou subdosagem, proporcionando um tratamento mais seguro e eficaz para o paciente (MANDEL, 2018) (JUAREZ et al., 2016) (SKERRETT et al., 2017). Além disso, as bombas de seringa são versáteis e podem ser utilizadas em uma gama de aplicações; em especial, são o tipo mais comum encontrado na literatura relativa a pesquisa microfluídica, a qual requer dosagens precisas. São também especialmente úteis em situações onde é necessário ajustar a taxa de infusão de maneira dinâmica, respondendo rapidamente às mudanças nas necessidades da aplicação.

Uma das principais desvantagens da bomba de seringa é a limitação em termos de volume. As seringas geralmente possuem um volume limitado, o que significa que a bomba precisa ser recarregada ou a seringa substituída com mais frequência em comparação com outros tipos de bombas de infusão que podem acomodar volumes maiores de fluido. Isso pode ser inconveniente em tratamentos que requerem infusões contínuas de grandes volumes. Outra desvantagem é a presença de atrito estático entre o êmbolo e a seringa. Para iniciar o movimento, a força aplicada precisa superar esse atrito estático, que é geralmente maior que o atrito dinâmico que ocorre durante o movimento contínuo. Essa diferença entre atrito estático e dinâmico causa uma transição abrupta que resulta no efeito *stick-slip*. Movimentos lentos, especialmente importantes ao contexto de operação da bomba, são mais suscetíveis ao *stick-slip* uma vez que a força necessária para manter o movimento constante pode não ser suficiente para superar de maneira consistente o atrito estático, causando perda de suavidade na vazão (BERMAN; DUCKER; ISRAELACHVILI, 1996). O movimento de precessão da rosca sem fim decorrente das tolerâncias dimensionais ou do acoplamento flexível da rosca com o motor de passo podem aumentar esse efeito. O efeito de *stick-slip* também pode ocorrer entre eixos lineares da bomba e rolamentos, todavia nesse caso o efeito tende a ser reduzido se a lubrificação for adequada. Além disso, as bombas de seringa podem ser mais sensíveis à presença de bolhas de ar no fluido infundido. Bolhas de ar podem interferir na precisão da dosagem administrada, exigindo um cuidado adicional na preparação da seringa para garantir que o fluido esteja completamente livre de ar antes da infusão (MANDEL, 2018).

2.4 Comparação dos Mecanismos de Direcionamento

Os diferentes mecanismos de bombas de infusão apresentam vantagens e desvantagens distintas. O mecanismo peristáltico, caracterizado pela ausência de contato direto entre o fluido e as partes mecânicas, minimiza o risco de contaminação, sendo adequado para infusões gerais em que a precisão elevada não é crucial. No entanto, sua vazão pulsada e precisão reduzida são limitantes para a aplicação em microfluídica. Já o mecanismo de pistão oferece boa precisão de vazão e é indicado para aplicações que exigem doses precisas. Todavia, as partes móveis podem sofrer desgaste ao longo do tempo, exigindo manutenção, e dependendo do design, pode haver contato direto do fluido com partes mecânicas. Além disso, o volume total de infusão pode ser limitado dependendo do tamanho do diafragma.

O mecanismo de bomba de seringa é versátil e adaptável, sendo comumente utilizado em diversos contextos clínicos para administração geral de fluidos por gotejamento, em especial, contextos onde uma medicação deve ser administrada em baixa taxa por um logo período de tempo (DIAS, 2019) (ALVES, 2002). No contexto da área da saúde, a limitação de quantidade de volume a ser infundido torna-se relevante se a dose de medicação for elevada, exigindo trocas da seringa durante a infusão e possivelmente inserindo erros na dosagem. Essa limitação é menos importante no contexto de pesquisa em microfluídica, uma vez que o volume a ser infundido é usualmente baixo (AKKOYUN; OZÇELIK, 2020). A inserção de imprecisões na vazão devido ao efeito de *stick-slip* é mais significativa, especialmente no contexto de vazões baixas. Esse efeito deve ser levado em conta no projeto e idealmente, através de uma lei de controle, ou outro recurso, remediado. Além disso, há um risco potencial de contaminação devido ao contato direto do fluido com a seringa (ALVES, 2002).

Considerando as características de cada mecanismo, escolhe-se implementar uma bomba de vazão do tipo bomba de seringa em virtude dos seguintes motivos:

1. Precisão suficiente. Apesar das não-idealidades provenientes do atrito do êmbolo com a seringa e dos rolamentos com os eixos, a bomba de seringa é capaz de sustentar vazão suave e menos pulsada que a bomba peristáltica. Embora seja possível que haja uma redução de precisão quando comparado com a bomba acionada por pistão, espera-se que o desempenho seja suficiente devido ao uso comum da bomba de infusão do tipo bomba de seringa em aplicações críticas (JUAREZ et al., 2016) (ALVES, 2002).

2. Simplicidade de montagem e manutenção. O mecanismo da bomba de seringa com rosca sem fim é relativamente simples, e pode ser facilmente reproduzido e modificado. Há pouco desgaste dos componentes, reduzindo a manutenção necessária, aumentando a expectativa de vida do equipamento e reduzindo custos. Também não há contato do fluido com as partes integrantes da bomba, exceto com a seringa descartável, o que são vantagens pertinentes em relação ao mecanismo de acionamento por pistão.

3. Grande quantidade de literatura nacional e internacional cobrindo diversos projetos e implementações do equipamento, fornecendo amplo material para pesquisa e aprimoramento do projeto.

3 DETALHAMENTO DO PROJETO

3.1 Atuador

De acordo com o Capítulo 2, escolhe-se desenvolver uma bomba de seringa com mecanismo de rosca sem fim. A fim de impulsionar a plataforma responsável por movimentar o êmbolo, é necessário um atuador. Para a bomba de seringa, é crítico ter controle preciso sobre a posição do êmbolo para garantir dosagens consistentes, por isso, o seguimento de referências de posição em baixas velocidades será priorizado. Consideram-se diferentes tipos de motores elétricos disponíveis para a atuação.

Motores de Corrente Contínua (CC) de ímã permanente com escovas: simples e amplamente utilizados, esses motores possuem modelos lineares com funções de transferência conhecidas da tensão de armadura para a velocidade, além de ter torque elevado em baixas rotações, rápida velocidade de resposta e baixo custo (HUGHES, 2005) (LINARES-FLORES; SIRA-RAMIREZ, 2004). Todavia, controlar a posição de forma precisa tem complexidade aumentada devido a presença de um integrador da velocidade para a posição no modelo. Além disso, motores CC com escovas sofrem desgastes com o tempo de operação, sendo necessária manutenção ou troca do motor após certo período. Caso esse motor seja escolhido, será necessária uma realimentação da posição do motor e uma lei de controle para a corrente de armadura.

Motores de Corrente Contínua (CC) sem escovas: esse motor pode ser visto como um motor de corrente contínua comutado eletronicamente "de dentro para fora" (HUGHES, 2005), portanto, pode ser controlado de forma semelhante a um motor de corrente contínua convencional. A principal vantagem desse motor é o menor desgaste dos componentes mecânicos, enquanto a desvantagem é o maior custo e menor disponibilidade comercial.

Motores de Corrente Alternada (AC): a velocidade do motor AC, seja síncrono ou assíncrono, é determinada pela frequência da tensão de armadura e pelo número de par de polos magnéticos. O controle fino de posição é mais complexo, exigindo sistemas de controle vetorial ou de frequência variável. Um inversor é necessário caso a bomba opere com tensão de alimentação CC (HUGHES, 2005).

Servo motores: esse tipo de motor, que pode ser tanto CC quanto AC, incorpora eletrônica e instrumentação complementar para adicionar um sistema de realimentação. Esse sistema permite que o motor servo tenha um controle de posição interno, e permite movimentos que não seriam possíveis nos motores tradicionais sem a malha de realimentação (HUGHES, 2005). Contudo, ressalta-se que a carga acoplada ao eixo pode exceder as especificações do controlador projetado internamente, deteriorando o seguimento de referências de posição nesses casos.

Motores de passo: o eixo de saída desse motor gira em uma série de intervalos angulares discretos fixos, ou passos, onde um passo é gerado pelo acionamento das bobinas do motor em uma sequência específica. O motor de passo tem a vantagem de poder

ser controlado diretamente por microcontrolador ou computador, em conjunto com uma interface de potência. Quando um número definido de passos é acionado, o eixo gira um ângulo conhecido, o que torna o motor ideal para controle de posição (HUGHES, 2005). Além disso, os motores de passo têm um torque alto em baixas velocidades, com decréscimo do torque a medida que a velocidade aumenta (IQTEIT et al., 2022), sendo adequados para aplicações que requerem movimento lento e controlado. Por fim, os motores de passo são altamente confiáveis e duráveis, com menos componentes mecânicos sujeitos a desgaste em comparação com outros tipos de motores, como motores DC com escovas (ATHANI, 1997).

Dadas as características discutidas, a escolha do tipo de motor para aplicação pode ser reduzida para o motor servo e o motor de passo. Uma vantagem do motor de passo é o maior torque de retenção, que é eficaz em manter a posição com precisão em baixas velocidade. Essa característica é reduzida significativamente a medida que a velocidade aumenta. Em contrapartida, o motor servo tem torque de retenção em função da corrente dinamicamente ajustada pelo circuito de realimentação, sendo eficaz em uma gama maior de velocidades. O motor servo também é melhor adaptado em condições onde a carga varia de forma dinâmica, pois enquanto o motor de passo pode perder passos, o motor servo tende a compensar o aumento da carga através da eletrônica de controle. Também considera-se que o motor de passo pode ter escopo de vida superior ao motor servo devido ao baixo atrito entre os componentes e ao motor não ser danificado em casos de sobrecarga (HUGHES, 2005) (ATHANI, 1997). Enquanto ambos os motores são adequados para a aplicação, por fim escolhe-se o motor de passo para a atuação da bomba de seringa. As características de alto torque de retenção e alta precisão em baixas velocidades, combinadas com menor valor comercial do componente e maior disponibilidade fundamentam a escolha.

A fim de simplificar a lógica de acionamento do motor de passo e priorizar o nível correto e estável de corrente, e também considerando o baixo custo e alta disponibilidade do componente no mercado, opta-se por utilizar um circuito comercial *driver* para o acionamento. O *driver* é responsável por energizar as bobinas do motor de passo em função de um sinal pulsado de entrada. Para cada pulso do sinal de entrada, o motor fará um avanço, onde o ângulo do avanço é determinado pela configuração de seleção de passo, ou *stepping*, enviada ao *driver* por pinos digitais. Essa configuração determina a sequência de acionamento das bobinas, de forma a girar o rotor em ângulos iguais ou menores que a resolução angular do motor. Para isso, o *driver* aplica correntes de magnitudes controladas aos enrolamentos de entrada (DANOWITZ, 2019) (HUGHES, 2005). A Equação 1 quantifica o deslocamento angular do motor para cada pulso do sinal de entrada do *driver*.

$$\alpha_s = \frac{\alpha_m}{S} \tag{1}$$

Nessa Equação, α_s é o ângulo de rotação para um pulso, α_m é o ângulo de passo do motor encontrado no *datasheet* do motor e S é a configuração de *stepping* do driver.

3.2 Modelagem

Para prever o comportamento da vazão quando o motor for acionado, de forma que o projeto do controlador em malha fechada torne-se específico e adequado ao sistema, busca-se entender a dinâmica do processo através de sua modelagem matemática. Para isso, propõe-se utilizar a modelagem caixa branca.

A modelagem caixa branca (ou modelagem de sistemas baseada em física) envolve a criação de modelos matemáticos dos sistemas dinâmicos com base em um conhecimento da física subjacente, dos componentes e das interações do sistema. A característica principal deste tipo de modelagem é a utilização de equações, equações diferenciais, leis de conservação, e outras ferramentas matemáticas para descrever o comportamento do sistema. A modelagem caixa branca é transparente e parametrizável, e pode fornecer modelos distintos de um mesmo sistema para diferentes aplicações, onde cada modelo captura os componentes essenciais da física para aquele caso (CHEVET et al., 2020).

O primeiro passo para a modelagem caixa branca é analisar os sinais transmitidos entre os componentes do sistema dinâmico da bomba de seringa, onde o sinal de entrada é a tensão pulsada proveniente do microcontrolador para o *driver* do motor de passo, e a saída é a vazão gerada pela bomba. É comum *driver* receber diretamente a tensão de alimentação do motor, sinais de controle para as configurações de *stepping*, e um sinal pulsado para cada passo do motor.

Como discutido na Seção 3.1, o *driver* é responsável pela lógica sequencial de acionamento das bobinas do motor, e é conectado a ele por quatro fios no caso de um motor de passo bipolar, dois para cada bobina. A rotação do motor de passo sofre o efeito de quantização devido aos passos do motor serem discretos, onde o deslocamento angular por passo é dado pela Equação 1. O motor de passo pode ser acoplado à barra roscada através de um acoplamento rígido ou flexível. O acoplamento flexível pode ser necessário devido às tolerâncias dimensionais das peças da bomba, onde o acoplamento rígido geraria tensões que podem comprometer a suavidade do movimento. A rotação da barra deve movimentar uma plataforma, ou dispositivo de avanço, através de uma porca acoplada a sua estrutura.

O dispositivo de avanço é transladado em decorrência da rotação da barra roscada. Nota-se que é necessário um tipo de eixo para garantir que a plataforma se movimente em um movimento linear puro. Para isso, pode-se utilizar rolamentos na plataforma que deslizam em eixos lineares. Cabe notar que há atrito entre os rolamentos e os eixos. Esse atrito pode ser dinâmico, aumentando o torque necessário para impulsionar a plataforma, ou estático, gerando o efeito de *stick-slip* e "saltos"durante o movimento.

Por consequência da translação, a plataforma exerce força sobre o êmbolo da seringa, onde a pressão gerada impulsiona o fluido para fora da seringa com uma determinada vazão. Novamente está presente o efeito de *stick-slip* entre o êmbolo da seringa e a sua parede externa. As Figuras 3 e 6 ilustram o mecanismo descrito. Por fim, o fluido percorre o tubo até existir a formação de uma gota, que despenca ao atingir determinado volume.

A fim de obter um modelo inicial simples, considerando os acoplamentos mecânicos ideais, o efeito do atrito estático e dinâmico entre os rolamentos e os eixos lineares desprezíveis, e o efeito do atrito estático e dinâmico entre o êmbolo e o cilindro da seringa desprezíveis, desenha-se um modelo equivalente da saída de tensão do microcontrolador para o *driver* até a vazão de saída na Figura 4.

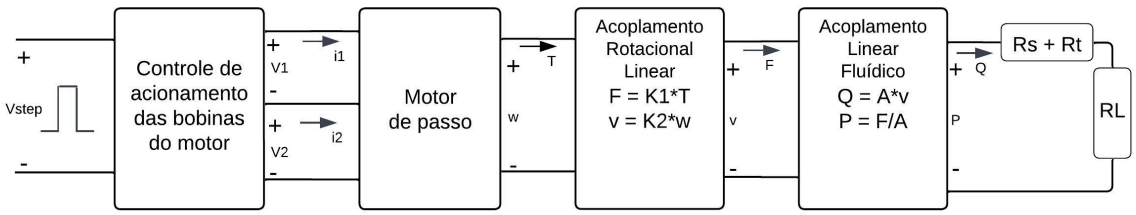
Na Figura 4:

1. V_{step} é a tensão pulsada de saída do microcontrolador de frequência f_s.

2. V_1, V_2, i_1 e i_2 são as tensões e correntes das fases de um motor de passo bipolar.

3. ω e T são a velocidade angular de rotação da haste roscada em $rad\,s^{-1}$ e o torque do motor de passo em Nm, respectivamente.

Figura 4: *Diagrama de Blocos Equivalente do Modelo Dinâmico*



Fonte: O Autor

4. K_2 é a relação entre a velocidade angular da barra roscada e velocidade linear do dispositivo de avanço da bomba de infusão. É possível verificar que $K_2 = \frac{d}{2\pi}$, onde d é igual ao passo da rosca da barra roscada.

5. K_1 é a relação entre o torque do motor e a força F exercida no êmbolo da seringa. Como $\omega T = vF$ para conservação ideal de potência, segue que $K_1 = \frac{1}{K_2} = \frac{2\pi}{d}$.

6. Q é a vazão da seringa, dada pela área do êmbolo multiplicada pela velocidade de avanço. Adicionalmente, a pressão P exercida pelo êmbolo da seringa ao fluido é calculada dividindo-se a força pela área.

7. $R_s + R_t$ representa a resistência hidráulica combinada da seringa e do tubo por onde o fluido vaza, e R_L a resistência hidráulica da carga receptora do fluido infundido, como um chip microfluídico por exemplo. De acordo com (DYLAN et al., 2017), considerando fluxo laminar, a resistência hidráulica pode ser calculada pela Equação 2 pela Lei de Poiseuille quando o tubo é cilíndrico. Nessa equação, l é o comprimento do tubo, d é o diâmetro do tubo, e ν é a viscosidade do fluido. Caso a resistência hidráulica da carga tenha características que fogem desses pressupostos, a queda de pressão em função da vazão deve ser modelada separadamente.

$$\frac{\Delta P}{Q} = \frac{128 l \nu}{\pi d^4} \tag{2}$$

Deseja-se determinar o comportamento do sistema *driver* e motor de passo. Modelos não lineares existem na literatura para o motor de passo, como discutido em (IQTEIT et al., 2022), e na documentação do *Simulink* em (MATHWORKS, 2023). Todavia, considerando a velocidade de resposta do motor de passo ordens de grandeza superior aos tempos de infusão operados pela bomba, considerando a regulagem de corrente exercida pelo *driver*, e considerando que o torque para acionamento e para compensação das perdas de carga não exceda o torque de retenção do motor de passo, encontrado no *datasheet*, assume-se, nesse caso que o motor reage instantaneamente a V_{step}, e sua velocidade em função do tempo é dada pela Equação 3.

$$\omega(t) \approx f_s \alpha_s \tag{3}$$

Sendo f_s a frequência de pulsos de V_{step} e α_s o ângulo de rotação por pulso em função do ângulo de rotação por pulso nominal do motor α_m e da seleção de *step* escolhida para o *driver*.

Considerando essas hipóteses simplificadores, e assumindo área circular do êmbolo da seringa pressionado pela plataforma, é possível reduzir o diagrama da Figura 4 para a Equação 4.

$$\begin{aligned} Q &= v(t)A \\ Q &= \omega(t)K_2A \\ Q &= f_s\alpha_s\frac{d}{2\pi}\pi\left(\frac{D}{2}\right)^2 \end{aligned} \tag{4}$$

Nessa Equação, $v(t)$ é a velocidade linear da plataforma (ou dispositivo de avanço) responsável por pressionar o êmbolo da seringa, A é a área do êmbolo da seringa, $\omega(t)$ é a velocidade angular da barra roscada acoplada ao eixo do motor de passo, K_2 é a relação entre a velocidade angular da barra roscada e velocidade linear da plataforma, α_s é o ângulo de uma rotação do motor por passo, d é o passo de rosca da barra roscada e D é o diâmetro do êmbolo da seringa. f_s é a frequência do sinal pulsado do microcontrolador para o *driver* e Q é a vazão resultante da bomba de seringa.

Dadas as hipóteses simplificadoras, o sistema dinâmico é proporcional, ou seja, segue perfeitamente uma referência ajustada por um ganho em malha aberta. É em função desse resultado que motores de passo podem ser usados sem malha de realimentação para aplicações que tolerem erros de posição moderados, sendo bem sucedidos em manter uma precisão aceitável nesses caso. Tal resultado é verificado experimentalmente no trabalho de (DIAS, 2019).

Todavia, tanto em aplicações na indústria microfluídica quanto na indústria médica, o zelo pela precisão de vazão é prioritário. Sabe-se que as não idealidades dos acoplamentos, atrito dinâmico e estático do êmbolo da seringa com o cilindro, desalinhamento do motor com a barra roscada, vibrações e outras perturbações, vão afetar o ponto de operação e a precisão da infusão. Nesses casos, a operação em malha aberta não é suficiente. Por esse motivo, propõe-se incorporar um sensor de vazão ao projeto, e implementar uma lei de controle com o fim de reduzir esses efeitos e melhorar a qualidade da saída.

Para medição da vazão, é utilizado um sensor de contagem de gotas infravermelho na saída da bomba capaz de estimar a vazão através da frequência de gotas, caso o volume de cada gota seja conhecido. Dessa forma, faz-se a hipótese de que a variação do volume da gota é baixa e pode ser estimada experimentalmente (GENNES; BROCHARD-WYART; QUÉRÉ, 2004). O sensor infravermelho possui as vantagens de ser comercialmente acessível, ter baixo valor de mercado e não ser invasivo - não ter contato com o fluido evitando desgastes e contaminações. Entretanto, a contagem de gotas é menos precisa que a medição direta de vazão ou pressão pois não é possível medir a variável do processo continuamente, apenas de forma amostrada pela presença da gota. Além disso, espera-se que o período de queda entre cada gota seja da ordem de segundos para vazões da ordem de $\mu L/min$, considerando algumas dezenas de microlitros por gota, logo espera-se um problema de *subamostragem* da vazão caso as perturbações durante a operação ocorram entre os períodos de amostragem. Apesar dessas limitações, o trabalho de (ARÊDES et al., s.d.[a]) sugere que pode-se obter um erro inferior a 5% se a calibração for adequada.

3.3 Partes Integrantes

A busca por um motor de passo adequado levou à consideração do NEMA 17 bipolar com passo de 1,8 graus, tensão de alimentação de 12 V e corrente de até 2 A. Motores dessa categoria são amplamente utilizados em aplicações de impressão 3D, robótica e automação por possuírem equilíbrio entre tamanho e desempenho (PUTRI et al., 2023) (LAKE; HEYDE; RUDER, 2017) (WIJNEN et al., 2014) (AKKOYUN; OZÇELIK, 2020). *NEMA* (National Electrical Manufacturers Association) se refere aos padrões dimensionais do motor, "17" indica que o motor possui uma flange de montagem de 1,7 polegadas (42 mm) de largura. Esta padronização facilita a integração do motor em uma variedade de projetos e aplicações. Além disso, o motor NEMA 17 considerado possui torque de retenção de $T = 59\,Ncm$. Assumindo uma barra roscada com passo de rosca $d = 0,8\,mm = 0,08\,cm$, considerando o acoplamento rotacional linear proposto na Figura 4 $F = 2\pi T/d$ e considerando um coeficiente de segurança $n = 5$, é necessário aplicar $927\,N$ à plataforma para exceder o torque de retenção motor. Dessa forma, estima-se que o torque de retenção é suficiente para acionar o êmbolo da seringa, superando a resistência do fluido e não idealidades em quaisquer regiões de operação.

O *driver* A4988 é considerado como a escolha para o acionamento do motor de passo NEMA 17. Este *driver* é amplamente utilizado devido à sua simplicidade e robustez (DANOWITZ, 2019), (LAKE; HEYDE; RUDER, 2017)., oferecendo a capacidade de seleção de passo, ou *stepping*, até 1/16, onde $S \in \{1, 2, 4, 8, 16\}$. A função de *stepping* permite que o motor pare em posições intermediárias de passos completos, aumentando significativamente a resolução rotacional do motor e consequentemente reduzindo o efeito de quantização da vazão da bomba. Todavia, quanto menor é o ângulo de rotação para um pulso, menor é a velocidade alcançável pelo motor e menor o torque de carga (BEDNARSKI; JACKIEWICZ; GAŁECKI, 2021) (IQTEIT et al., 2022). De acordo com a discussão relativa à ordem de grandeza do torque do motor NEMA 17 utilizado, espera-se que a redução de torque não afete a operação da bomba. O A4988 também suporta tensões de operação de até 35 V e correntes de saída de até 2 A por bobina, proporcionando flexibilidade no controle de motores com diferentes especificações, em especial sendo suficiente para o motor NEMA 17 escolhido. Além disso, a disponibilidade e a documentação extensa do A4988 tornam a sua integração no projeto mais acessível e facilitam a resolução de problemas durante o desenvolvimento.

O *driver* é conectado diretamente a um microcontrolador que terá a função de gerar os sinais de controle do driver, incluindo os níveis lógicos que determinam a configuração de *stepping* e o sinal pulsado. A frequência de pulso gerada determinará a velocidade de rotação do motor, e por consequência a vazão da bomba.

A fim de poder controlar a bomba manualmente caso ocorra uma emergência ou para economizar tempo de posicionamento, incluem-se no equipamento dois botões cujos sinais gerados são avaliados continuamente. O primeiro botão desloca o êmbolo da seringa para frente se pressionado, enquanto segundo botão o desloca para trás. Os sinais comandados pelos botões estão em configuração *pull-up*, ou seja, têm sinal alto quando o botão estiver aberto. Além disso, decidiu-se conectar um filtro passa-baixo do tipo RC, cuja constante de tempo é igual a 10 ms, em série com cada botão. Os filtros têm a função de atenuar ruídos de alta frequência conhecidos como *bouncing* quando o botão é pressionado, proporcionando apenas uma detecção de sinal para cada acionamento manual. Uma chave $S1$ é conectada entre a fonte e a entrada de tensão do microcontrolador para ligar e desligar o equipamento. Também são conectados capacitores entre as tensões de alimentação e o GND com o

propósito de filtragem de ruído na linha.

Para a medição da vazão é usado um sensor óptico de contagem de gotas de modelo $KY-032$. O sensor funciona através de um sistema de reflexão infravermelho, onde um LED (*Light-Emitting Diode*) emissor e um fotorreceptor ficam lado a lado. Quando um obstáculo ou objeto atravessa um cone de 35 graus em frente ao sensor, a saída muda de nível lógico. Além disso, ele possui dois potenciômetros ajustáveis, sendo um para controle da frequência de operação e outro para controle da sensibilidade da distância de detecção, que pode ficar entre $2\,cm$ e $20\,cm$ (JOY-IT, 2024). O sinal do sensor é conectado ao microcontrolador de forma que ele gere uma interrupção de prioridade elevada quando houver mudança de nível lógico.

Para acionar o *driver* conectado ao motor de passo, ler os sinais originados dos botões, processar as interrupções geradas pelo sensor, calcular o sinal de controle e receber e enviar informações para outros dispositivos, desenvolveu-se o *firmware* da bomba de infusão. O *firmware* é o código embarcado que deve ser executado a todo momento pelo microcontrolador. Ele é desenvolvido para atender a necessidade específica do sistema, de forma que a leitura do sinal do sensor e o cálculo da lei de controle ocorram em *tempo real*.

A fim de reduzir custos e priorizar componentes facilmente acessíveis no mercado, escolhe-se o microcontrolador *Arduino UNO*. O *Arduino* faz o processamento do *firmware*, a comunicação com os periféricos, o envio dos sinais de controle ao *driver* A4988 e o recebimento dos sinais do sensor. A placa possui 14 pinos digitais de entrada/saída, dos quais 6 podem ser usados como saídas PWM, além de 6 entradas analógicas. A placa também oferece comunicação serial *(UART)*, *SPI* e *I2C*, permitindo a conexão com sensores e módulos externos (ARDUINO, 2024).

O processador do *Arduino UNO*, *ATmega328P*, é um processador simples de oito bits cuja arquitetura é do tipo *RISC* (*Reduced Instruction Set Computer*). A arquitetura *RISC* utiliza um conjunto pequeno de instruções altamente otimizadas, a maioria executada em apenas um ciclo de *clock* (operação atômica) (LU, 2021). Ele possui 32 registradores de propósito geral, frequência de 16 MHz, memória *flash* de 32 kilobytes, responsável por armazenar o código, memória *EEPROM (Electrically Erasable Programmable Read-Only Memory)* de um kilobyte, para armazenar outros dados não voláteis, e memória $SRAM$ *(Static Random-Access Memory)* de dois kilobytes, para os dados voláteis, como variáveis, a pilha (ou *stack*), e o monte (ou *heap*) (INC., 2024).

Considera-se: a utilização de cerca de 10 pinos digitais para os sinais de controle do *driver* e leitura de botões; o uso de 200 variáveis do tipo *float*, ou *uint32_t*, de quatro bytes por variável, para armazenamento dos dados de controle e operação da máquina de estados do *firmware*; um pino PWM para acionamento da entrada *step* do *driver*, um *Timer* para rastreamento do tempo de operação, um *Timer* para geração do sinal PWM do pino de *step*, e um pino digital capaz de gerar interrupções ao detectar mudança de nível lógico do sensor digital. Estimando que a leitura do sensor e o cálculo do sinal de controle não exceda 1000 instruções após a compilação, assumindo 1,5 ciclos de *clock* em média por instrução, e considerando um coeficiente de segurança de cinco, estima-se que o atraso entre o recebimento do sinal do sensor e a mudança de frequência PWM do pino *step* não ultrapasse $1\,ms$. Como esse atraso tem ordens de grandeza menor que o período de detecção de gotas do sensor infravermelho, e como as características estimadas do *firmware* e do *hardware* não excedem as especificações do *Arduino*, estima-se que o microcontrolador, embora simples, é suficiente para aplicação.

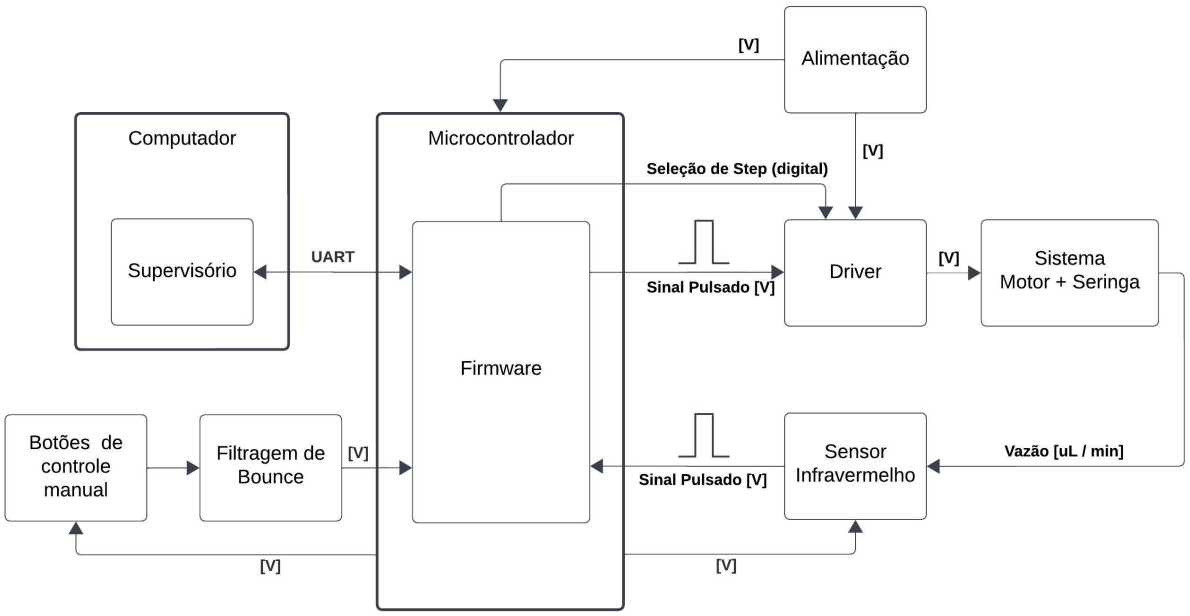
Utiliza-se uma fonte de tensão contínua de 12 V e 2 A para a alimentação do motor através do *driver* e do microcontrolador. A tensão e corrente são adequadas uma vez que

o *Arduino UNO* possui tensão recomendada entre 7 a 12 V, e máxima corrente de 500 mA, enquanto o *driver* deve ter tensão entre 8 e 32 V, e 1,5 A de corrente para o motor. Adicionalmente, fontes de 12 V são amplamente utilizadas em equipamentos eletrônicos e tem valor de compra acessível.

Por fim, para o usuário entrar com as configurações de vazão e tempo de infusão, é desenvolvido um programa Supervisório capaz de enviar os dados de configuração para o microcontrolador da bomba através do sistema *UART*. Esse sistema é escolhido devido a sua simplicidade e características suficientes de transmissão de dados. O *firmware* da bomba em contrapartida envia continuamente os dados de vazão atual e total infundido.

A Figura 5 apresenta os diferentes componentes integrantes do equipamento organizados como diagrama de blocos.

Figura 5: *Diagrama de Blocos da Bomba de Infusão*



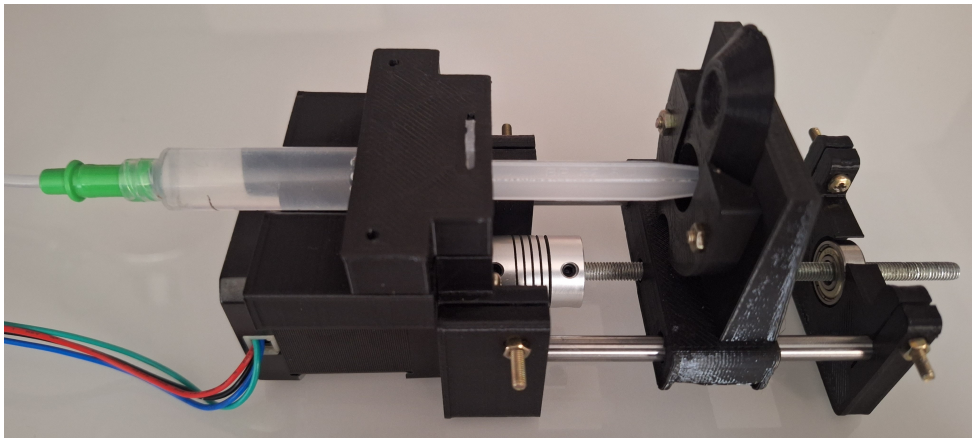
Fonte: O Autor

3.4 Montagem

Para o projeto da montagem mecânica da bomba de infusão, o artigo de (lake; heyde; ruder, 2017) apresenta um design satisfatório de bomba de infusão de seringa de baixo custo e de uso amplo, combinando e aperfeiçoando construções anteriores como a de (wijnen et al., 2014). Como o design é aberto e cumpre os requisitos propostos neste trabalho, ele será usado para a construção física da bomba de infusão, onde os arquivos .stl são encontrados no artigo citado. A montagem completa da bomba de infusão é dada pela Figura 6.

No Apêndice A encontram-se os componentes mecânicos e elétricos necessários para a construção da bomba, as peças impressas 3D necessárias para a estrutura, as instruções de montagem e considerações para a operação da bomba, e o diagrama elétrico do projeto. Além disso, incluem-se imagens da realização do diagrama elétrico e do posicionamento do sensor infravermelho.

Figura 6: *Bomba de Infusão*



Fonte: O Autor

3.5 Firmware

Três funcionalidades principais são implementadas no *firmware* para a operação do equipamento:

1. **Infusão**: Leitura do sinal do sensor através da geração de uma interrupção, estimação da vazão, e cálculo do sinal de controle. A frequência calculada pelo controlador é transformada no número de contagens que o *Timer 1* do *Arduino* deve alcançar para gerar uma interrupção de comparação, ou *Compare Match Interrupt*. Essa interrupção de comparação muda o nível lógico do pino nove, que é conectado à entrada *step* do *driver* do motor de passo. A inversão do nível lógico do pino nove em intervalos de tempo determinados permite a geração de um sinal *PWM* de frequência regulada, e por consequência o controle da velocidade do motor de passo.

2. **Verificação de entradas**: Recebimento dos pacotes de dados do supervisório, e leitura dos botões de acionamento manual. Essas verificações não devem interromper ou atrasar a tarefa de Infusão, por isso, são colocadas dentro da função *loop()* do *Arduino* e executadas através de comparações temporais utilizando a função *millis()*.

3. **Transmissão de dados**: O *firmware* envia informações relativas ao estado da operação da bomba de seringa ao Supervisório. Da mesma forma como na tarefa de Verificação de Entradas, essa funcionalidade tem baixa prioridade e é chamada na função *loop()*.

Considerando que apenas três tarefas são executadas concomitantemente, onde apenas uma delas tem alta prioridade, optou-se pela utilização das funcionalidades nativas do microcontrolador, como rotinas de interrupção geradas pelos contadores e funções como a *loop*() e a *millis*(), em lugar do emprego de uma biblioteca de sistema operacional de tempo real, como o *FreeRTOS (Free Real-Time Operating System)*. As vantagens dessa escolha são o desenvolvimento do *firmware* com o mínimo de *overhead* possível, ou seja, com o mínimo de recursos de processamento e armazenamento necessários para a aplicação, além de menor complexidade de código. Todavia, essa estrutura seria limitada se o número de tarefas gerenciadas fosse maior.

3.6 Supervisório

O Supervisório é um programa cuja funcionalidade é receber entradas do usuário e mostrar de forma periódica informações de operação. As informações incluem tempo de execução, vazão de referência, vazão estimada, erro de vazão e sinal de controle. O usuário pode, através do supervisório, enviar uma Configuração de Infusão ao *firmware* embarcado da bomba. Essa configuração contém a vazão desejada, o tempo de infusão, o sentido de giro, e a seleção de passo. Adicionalmente, é possível enviar um comando de movimentação manual, e sobrescrever a rotina de infusão em andamento, ao pressionar-se as teclas "A"ou "D"do teclado. A tecla "A"move a plataforma que pressiona o êmbolo da seringa no sentido oposto da infusão (*puxar*), enquanto a tecla "D"move a plataforma no sentido da infusão (*empurrar*). Essa funcionalidade é utilizada caso seja necessário posicionar o êmbolo da seringa em um ponto específico, ou abortar uma rotina em andamento.

Devido à ausência do requisito de alto desempenho para o Supervisório, o *software* é desenvolvido em *Python*, onde o alto nível de abstração da linguagem permite a implementação mais rápida e compacta de funcionalidades no decorrer do projeto. No caso do protótipo desenvolvido, o Supervisório é executado em um computador de propósito geral do tipo *laptop*, todavia, ele também pode ser executado em um *SBC (Single-Board Computer)*, como por exemplo o *Raspberry Pi*.

A comunicação entre o Supervisório e o *firmware* embarcado é realizada pela conexão USB do Arduino através do sistema UART, com *baudrate* escolhida de 115200 *bps* . Define-se o pacote de configuração enviado pelo Supervisório para o *firmware* da bomba na Figura 7. Nele, é possível informar a seleção de passo no primeiro byte, o sentido de giro do motor no segundo byte (*char* '0' para empurrar, *char* '1' para puxar), a vazão de referência desejada em $\mu L\ min^{-1}$ e o tempo de operação em min.

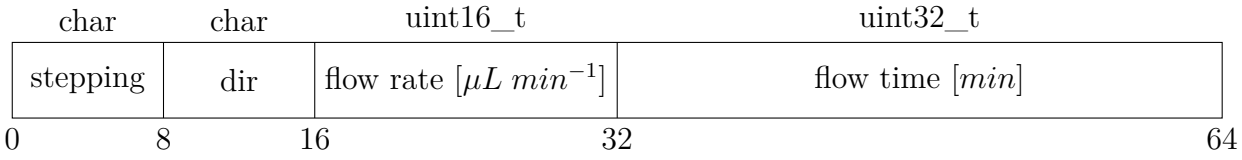


Figura 7: *Pacote serial de configuração para a Bomba de Infusão*

Fonte: O Autor

Se a bomba for operada manualmente pelo Supervisório, apenas o *char* 'dir' é transmitido pela linha serial. Nesse caso, a configuração de infusão é sobrescrita e a bomba move-se de forma proporcional ao tempo que a tecla é pressionada.

4 RESULTADOS

4.1 Estimação do Volume da Gota

A fim de estimar a vazão através da contagem de gotas, é necessário ter uma medida confiável do volume médio de uma gota. Uma prática comum de laboratório é considerar o volume de uma gota de água como $0,05\,cm^3$, todavia, essa estimação não é adequada no contexto onde uma vazão precisa é necessária. Por isso, sugere-se realizar um experimento com o fim de estimar o volume médio de uma gota de água no contexto da bomba de seringa construída. De acordo com (GENNES; BROCHARD-WYART; QUÉRÉ, 2004), volume de uma gota formada em um tubo de diâmetro constante, e mantida a uma temperatura constante, tende a ser aproximadamente constante. Esse fenômeno é explicado pela tensão superficial do fluido, que atua para minimizar a área da superfície da gota, resultando em gotas de volume similar sob condições constantes.

Propõe-se infundir um número N de gotas em um recipiente inicialmente seco. O recipiente tem sua massa medida antes e depois da infusão, o que permite determinar a massa do fluido infundido. A balança utilizada é a *SC-009* de resolução, de 0,1 grama, calibrada imediatamente antes do experimento, que é conduzido em pressão atmosférica e temperatura aproximadamente constante de 24 graus Celsius. Como a resolução da balança é relativamente baixa em relação ao volume esperado de uma gota, escolhe-se o número $N = 120$ para diminuir as incertezas geradas pela escala. Além disso, o experimento é repetido cinco vezes. O resultado das medições é dado pela Tabela 1.

Tabela 1: *Resultados experimentais da estimação da massa de gotas*

Ensaio	Massa medida
1	4,8 g
2	5,0 g
3	5,4 g
4	5,3 g
5	5,3 g

Fonte: O Autor

Através dos ensaios, é possível determinar a massa média $\bar{x} = 5,2\,g$ e o desvio padrão experimental $s = 0,3\,g$. Busca-se determinar a incerteza associada à medição. De acordo com (EVALUATION..., 2008), a incerteza é um parâmetro associado ao resultado de uma medição que caracteriza a dispersão dos valores que poderiam razoavelmente ser atribuídos ao mensurando. Ela é essencial para estabelecer a confiabilidade dos resultados encontrados, uma vez que toda medida encontrada empiricamente não é, a rigor, correta

(TAYLOR, 1997). Dois tipos de incertezas são destacados. A incerteza de tipo A é derivada da análise das estatísticas das observações, e é dada pela Equação 5.

$$\begin{aligned} u_A &= \frac{s}{\sqrt{n}} \\ u_A &= 0,13\,g \end{aligned} \tag{5}$$

Já a incerteza de tipo B provém de fontes que não sejam a análise estatística das observações. Para esse experimento, assume-se que a maior fonte de incertezas tipo B é a resolução da balança r, dessa forma, a incerteza de tipo B é calculada pela Equação 6 (EVALUATION..., 2008).

$$\begin{aligned} u_B &= \frac{r}{2} \\ u_B &= 0,05\,g \end{aligned} \tag{6}$$

A incerteza combinada leva em consideração múltiplas fontes de incerteza que afetam essa medição. Ela é calculada combinando as incertezas individuais de várias fontes, que podem incluir tanto incertezas do tipo A (estatísticas) quanto do tipo B (não estatísticas) (EVALUATION..., 2008). A incerteza combinada é dada pela Equação 7.

$$\begin{aligned} u_C &= \sqrt{u_A^2 + u_B^2} \\ u_C &= 0,14\,g \end{aligned} \tag{7}$$

A fim de realizar uma análise semelhante para apenas uma única gota, é feita uma divisão das massa média e das incertezas calculadas pelas Equações 5, 6, 7 pelo número $N = 120$ de gotas por ensaio. Dessa forma, obtém-se $\bar{x}_{\text{gota}} = 0,043\,g$, $u_{A,\text{gota}} = 0,001\,g$, $u_{B,\text{gota}} = 0,000\,g$ e $u_{C,\text{gota}} = 0,001\,g$. A massa média de uma gota estimada por esse experimento é então data:

$$\bar{m}_{\text{gota}} = 0,043\,g \pm 0,001\,g \tag{8}$$

De acordo com (HAYNES, 2015), seção 6-7, a densidade da água é $0,9973\,g/cm^3$ a 24,0 graus Celsius e pressão atmosférica. Dessa forma, o volume médio de uma gota é estimado:

$$\bar{V}_{\text{gota}} = 0,043\,cm^3 \pm 0,001\,cm^3 \tag{9}$$

Esse valor apresenta uma incerteza relativa de $2,3\%$. Essa incerteza relativa é considerada alta para aplicações críticas, como dosagem de medicamentos, mas é aceitável como primeira estimação para o protótipo de bomba de seringa desenvolvido nesse trabalho. Além disso, é necessário destacar que existem outras fontes de incertezas de tipo B que não foram consideradas, como incerteza de calibração da balança, incerteza da temperatura, incerteza do usuário e outras. Propõe-se a realização de experimentos com equipamentos mais precisos e com número de amostras superior para diminuir a incerteza relativa a pelo menos $< 1\%$ em trabalhos futuros.

4.2 Ensaio em Malha Aberta

Considerando a Equação 1 que define a rotação angular do motor para um passo dada uma certa configuração de *stepping*, e a Equação 4 que estima, dadas as hipóteses simplificadoras discutidas na Seção 3.2, a vazão da bomba em malha aberta para uma determinada frequência de passos por segundo define-se a Equação 10:

$$\begin{aligned} Q &= \frac{f_s \alpha_m d D^2}{8S} \\ f_s &= K_m S\, Q,\ K_m = \frac{8}{\alpha_m d D^2} \end{aligned} \tag{10}$$

Nessa Equação, $D = \{13 \pm 0,5\}\, mm$ é o diâmetro medido do êmbolo da seringa de capacidade de 5 cm^3 utilizada, $d = 0,8\, mm$ é o passo de rosca da haste roscada, $\alpha_m = 1,8° = 0.0314159\, rad$ é o ângulo rotacionado por passo do motor de passo NEMA 17, S é a configuração de *stepping* e Q é a vazão em mm^3/s ou $\mu L/s$. Para uma escolha de $S \in \{1, 2, 4, 8, 16\}$ e de vazão de referência Q, é possível determinar a frequência do sinal pulsado a ser enviado ao *driver* do motor de passo. Define-se $K_m = 1,8835\, mm^{-3}$ como uma constante derivada das partes construtivas da bomba de seringa. É necessário que a frequência seja inferior a $31,25\, kHz$, que é o limite de frequência PWM que pode ser gerada pelo *timer 1* através da interrupção de comparação. Todavia, mesmo considerando $S = 16$, é necessária uma vazão superior a $1\, cm^3/s$ para atingir essa frequência, superior a região de operação da bomba.

Deseja-se verificar o desempenho da bomba de infusão construída em malha aberta para analisar a necessidade da implementação de um controlador para a vazão. Para isso, propõe-se realizar um ensaio de infusão de 120 segundos escolhendo $S = 16$ e definindo o *set-point* de vazão $Q = 300\, \mu L/min = 5\, \mu L/s$. A vazão real é estimada pela contagem de gotas do sensor infravermelho KY-032. Quando uma gota é detectada, o sensor gera uma interrupção ao microcontrolador, que por sua vez salva o tempo de ocorrência da gota desde o início do experimento. A vazão é estimada dividindo-se o volume médio de uma gota, dado pela Equação 9, pelo tempo transpassado entre as detecções das duas últimas gotas.

Observou-se durante os ensaios o efeito de *bouncing* no sinal do sensor, onde duas ou mais interrupções poderiam ser geradas devido à queda de uma única gota. Por isso, modificou-se o *firmware* do microcontrolador para que a interrupção gerada apenas mude o valor de uma variável *dropDetected* de *false* para *true*. A cada $5\, ms$, caso *dropDetected* esteja *true*, o *firmware* registra a queda da gota e retorna o estado da variável para *false*. Escolheu-se o período de $5\, ms$ para essa funcionalidade pois foi constatada a confiabilidade da geração de pelo menos uma interrupção durante a queda da gota nesse período. Além disso, implementou-se o descarte de gotas de período inferior a $50\, ms$, uma vez que empiricamente foi observado que as gotas de baixo período detectadas eram provindas do efeito de *bouncing*, pois não houve a queda real de duas gotas em sequência. Uma vez feitas essas mudanças, constatou-se que o *firmware* identificou corretamente a queda de cada gota.

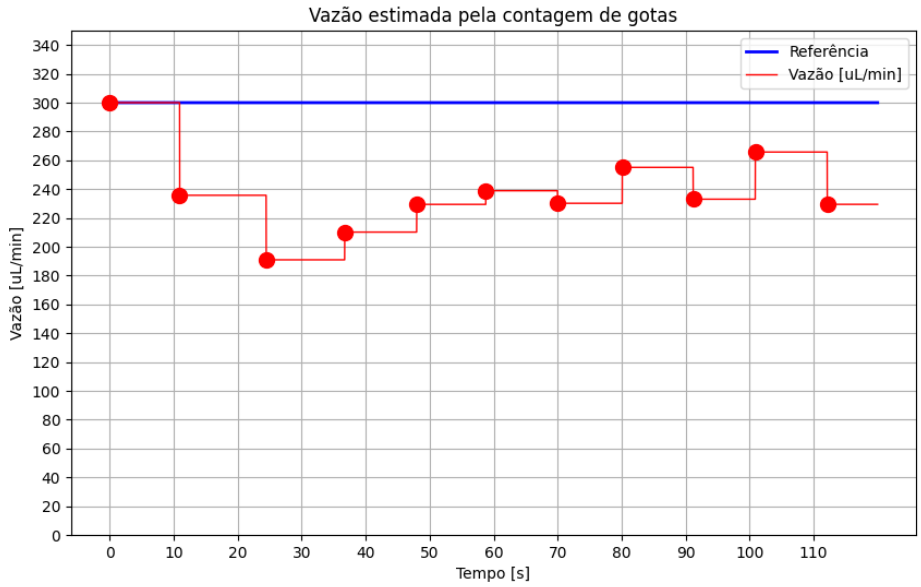
Nota-se que do início do experimento até a queda da primeira gota é impossível estimar a vazão, por isso, decidiu-se assumir a estimativa nesse caso igual à referência. Repete-se o experimento duas vezes. Os resultados são experimento são dados pelas Figuras 8 e 9.

Detectou-se nos dois ensaios a queda de 10 gotas, totalizando aproximadamente

430 μL, significativamente menos que os 600 μL esperados. Nota-se a presença de um erro sistemático cometido pela bomba de seringa, onde a vazão medida está inferior à vazão de referência na maior parte do ensaio. Suspeita-se que o erro sistemático seja principalmente proveniente das incertezas das variáveis da Equação 10. Exceto na primeira gota do segundo ensaio, nos dois experimentos o período de queda das gotas iniciais foi superior ao período de queda das gotas subsequentes. Estima-se que essa diferença seja em decorrência do acomodamento da plataforma que pressiona o êmbolo, onde ocorre uma deformação inicial antes da infusão iniciar com o início da ação do motor de passo. Pensa-se que o período de queda da primeira gota do segundo ensaio ser inferior às demais seja devido a formação parcial de uma gota no tubo antes do experimento, acelerando sua queda.

Também percebe-se uma parcela de erro aleatório, onde há uma oscilação da vazão estimada na parte final dos ensaios devido ao período de queda das gotas ora aumentar, ora diminuir. Estima-se que o erro aleatório seja em decorrência das não idealidades do sistema, como acoplamento flexível entre o motor de passo e a barra roscada, atrito estático e efeito de *stick-slip* tanto entre os rolamentos e os eixos lineares quanto entre o êmbolo e o cilindro da seringa. Também pensa-se que uma parcela do erro aleatório seja ocasionada pelas vibrações do motor de passo, fazendo a gota se desprender do tubo em tempos ligeiramente diferentes.

Figura 8: *Estimação da vazão da bomba em malha aberta (1)*

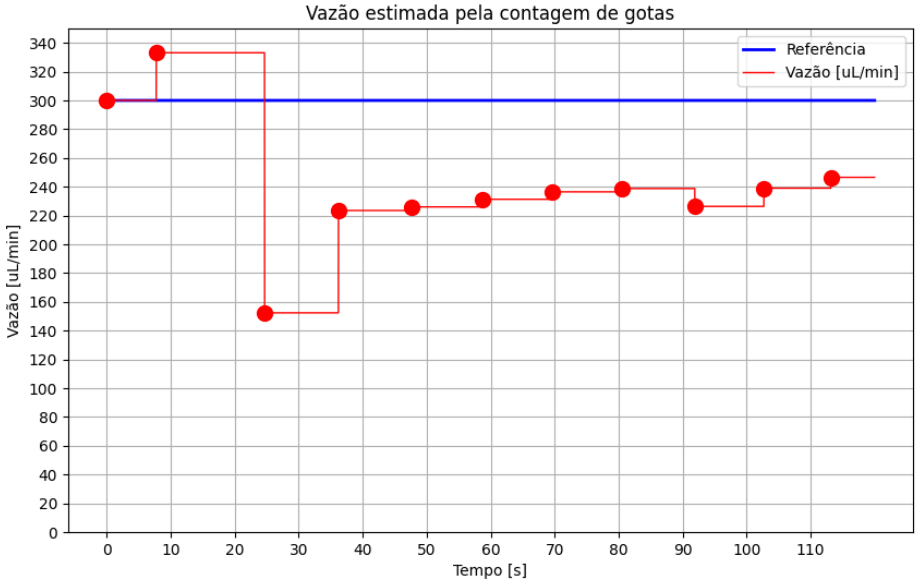


Fonte: O Autor

4.3 Ensaio em Malha Fechada com controlador proporcional

A fim de diminuir o erro sistemático identificado na Seção 4.2, propõe-se a implementação de um controlador em malha fechada. O sinal de controle é recalculado sempre que o

Figura 9: *Estimação da vazão da bomba em malha aberta (2)*



Fonte: O Autor

firmware detecte uma nova gota. Antecipadamente ao projeto do controlador, implementa-se uma saturação na saída de forma que o sinal de controle f_s esteja contido entre $0,5 f_{s,\text{ref}}$ e $1,5 f_{s,\text{ref}}$, onde $f_{s,\text{ref}}$ é a frequência de referência calculada pela Equação 10, definindo-se $Q = Q_{\text{ref}}$ em $\mu L/s$. A saturação está presente como um mecanismo de segurança para o motor de passo, impedindo que ele seja acionado excessivamente abaixo ou acima da frequência teórica de operação. O primeiro controlador considerado é o proporcional, onde o sinal de controle é dado pela Equação 11.

$$f_s[k] = K_p\, S(Q_{\text{ref}}[k] - Q_{\text{est}}[k]) \tag{11}$$

Nessa Equação, k é o número de gotas detectadas, f_s é a frequência a ser enviada ao driver do motor de passo, S é a configuração de *stepping*, Q_{ref} é a referência de vazão em $\mu L/s$, Q_{est} é a vazão estimada pelo período de queda da última gota em $\mu L/s$, e K_p é o ganho proporcional do controlador discreto. Inclui-se S na lei de controle pois uma configuração maior de *stepping* requer uma frequência maior de acionamento, sem alterar as características de controle.

Após a realização de testes para diferentes valores de K_p, percebe-se que a lei de controle proposta pela Equação 11 não consegue controlar devidamente o sistema. Devido ao período de queda das gotas ser elevado, uma vez que o erro torna-se pequeno, o sinal de controle cai bruscamente, e induz o período da gota seguinte a ser bastante elevado. Por consequência, a próxima ação de controle aumentará a vazão de saída excessivamente. Esse comportamento leva a um sinal oscilatório cujo desempenho não é adequado ao projeto.

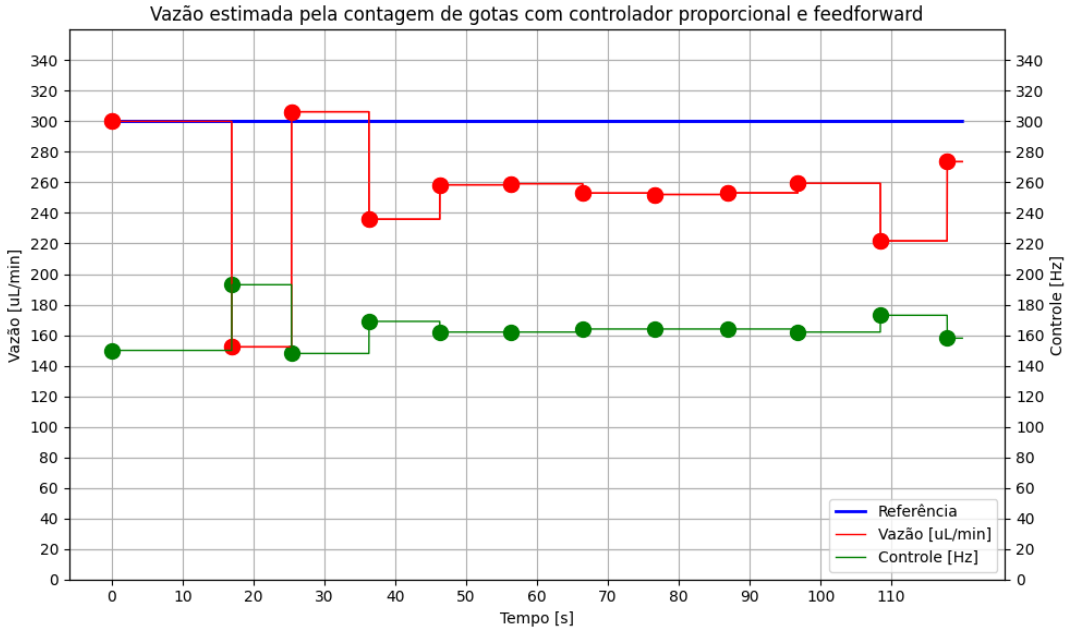
A fim de melhorar a saída, soma-se ao controlador proporcional da Equação 11 uma parcela *Feedforward* do tipo *Compensação de Referência*, de acordo com a Equação 12. A inclusão da componente *feedforward* antecipa as necessidades do sistema com base na referência desejada (BOURDAIS et al., 2020), em particular, sabe-se que a frequência de acionamento não estará distante da frequência teórica calculada pela Equação 10, tanto em virtude da discussão da Seção 3.2 como pelos resultados experimentais da Seção 4.2. Logo,

reduz-se a carga de controle da parcela proporcional ao erro. Escolhe-se para a parcela *feedforward* a frequência de referência calculada pelo modelo simplificado do sistema dado pela Equação 10. Nessa Equação, $K_m = 1,8835\, mm^{-3}$ é uma constante derivada das partes construtivas do equipamento.

$$f_s[k] = K_m S Q_{\text{ref}}[k] + K_p\, S(Q_{\text{ref}}[k] - Q_{\text{est}}[k]) \tag{12}$$

Realiza-se um ensaio em malha fechada com parâmetros semelhantes aos ensaios em malha aberta da seção anterior, onde $S = 16$, $Q_{\text{ref}} = 300\,\mu L/min = 5\,\mu L/s$ e o tempo de ensaio é 120 segundos. O ganho K_p é escolhido $1,09\, mm^{-3}$. Os resultados do ensaio em malha fechada para o controlador proporcional da Equação 12 são dados pela Figura 10. Não houve a presença de saturação durante o ensaio.

Figura 10: *Estimação da vazão da bomba em malha fechada com controlador proporcional e feedforward*



Fonte: O Autor

Nota-se que o controlador proporcional com ação *feedforward* diminuiu o erro sistemático, ou erro de regime, mas não foi suficiente para anulá-lo. Esse resultado é consistente com a Teoria de Controle, uma vez que de acordo com o Teorema dos Modos Internos (BOURDAIS et al., 2020), é necessário que a combinação da planta e controlador incorpore os polos, ou modos, da referência para garantir a ausência de erro de regime. Considerando a referência do tipo salto, que possui um polo em zero, e considerando a ausência de polos em zero na planta pois não há crescimento contínuo da saída para a referência do tipo salto, infere-se a necessidade de um controlador com pelo menos um polo em zero, ou integrador, a fim de seguir referências constantes.

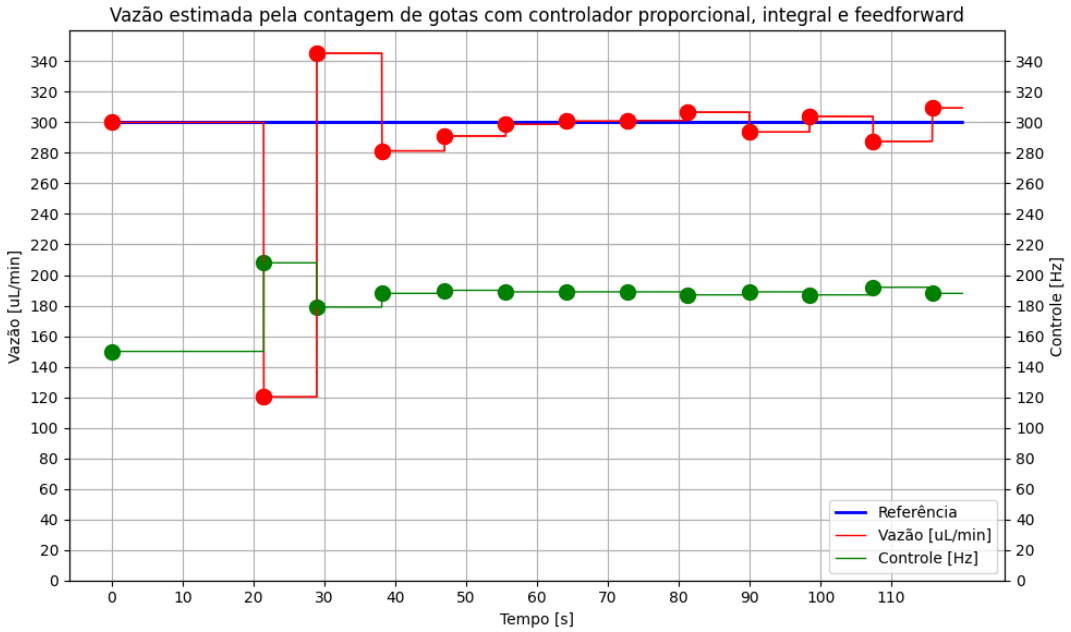
Além disso, observou-se que o período entre a queda da nona e décima gotas, de $t = 96,80\, s$ a $t = 108,44\, s$, foi sensivelmente superior ao período das demais gotas em regime. Esse comportamento pode ser descrito como a presença de uma perturbação do tipo impulso na entrada da bomba durante a formação da gota. Como esperado, o controlador proporcional aumentou o sinal de controle após a detecção da gota de maior período, diminuindo o impacto da perturbação sobre a saída.

4.4 Ensaio em Malha Fechada com controlador proporcional e integral

A fim de obter um controlador capaz de anular o erro de regime da bomba de seringa, projeta-se um controlador do tipo *PI (proporcional e integral)*. O controlador *PI* discreto soma à Equação 12 uma parcela proporcional à soma de todos os erros de vazão detectados no ensaio. Essa soma tem a consequência de manter a mudança do sinal de controle enquanto houver erro. Quando o erro convergir para zero, a parcela proporcional à soma estabiliza, sendo o sinal de controle resultante melhor adaptado para operar a bomba em regime permanente. Utiliza-se uma variável de estado do controlador para armazenar a soma dos erros de vazão, onde essa variável é congelada se a ação de controle atingir a saturação. Esse congelamento é conhecido como mecanismo *Anti-Windup* (Bourdais et al., 2020). A Equação 13 descreve a lei de controle do controlador *PI* utilizado. Faz-se um ensaio em malha fechada com o controlador *PI* projetado, onde escolheu-se $K_p = 0,30\,mm^{-3}$ e $K_i = 0,90\,mm^{-3}$. Os parâmetros do ensaio são semelhantes aos parâmetros utilizados nos ensaios das Seções 4.2 e 4.3. Os dados coletados do ensaio estão na Figura 11.

$$f_s[k] = K_m S Q_{\text{ref}}[k] + K_p S(Q_{\text{ref}}[k] - Q_{\text{est}}[k]) + K_i S \sum_{n=1}^{k-1}(Q_{\text{ref}}[n] - Q_{\text{est}}[n]) \tag{13}$$

Figura 11: *Estimação da vazão da bomba em malha fechada com controlador proporcional, integral e feedforward*



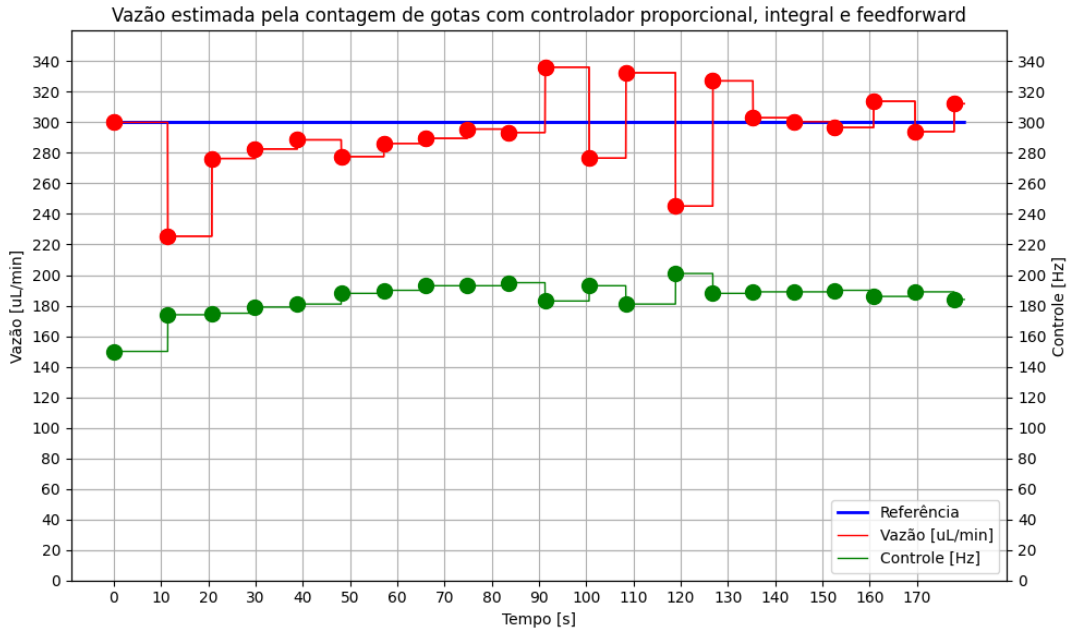
Fonte: O Autor

Nesse ensaio, verifica-se sobrepasso de cerca de 14% após a primeira ação de controle em decorrência do elevado período de queda da primeira gota. Estima-se que esse período alto, da mesma forma que em ensaios anteriores, seja induzido pelo acomodamento da plataforma que impulsiona o êmbolo da seringa no início do teste, pelos atritos estáticos

presentes no equipamento, e pela formação de uma pequena bolha de ar no tubo responsável gotejar o fluido em frente ao sensor infravermelho.

Percebe-se que a adição da ação integral reduziu sensivelmente o erro sistemático após o regime transitório da bomba de seringa, estabilizando a saída próxima à referência. Na região final do teste, notou-se a presença de perturbações aleatórias no período de queda das gotas, onde vê-se que o controlador busca compensar seu efeito através de uma pequena variação do sinal de controle. Essa variação aparenta crescer em magnitude, possivelmente sendo indicativa de uma instabilidade gerada pelas perturbações. Propõe-se a realização de um teste adicional, de maior duração, com o objetivo de investigar a capacidade de rejeição de perturbações em regime do controlador. O resultado do teste é dado pela Figura 12.

Figura 12: *Estimação da vazão da bomba em malha fechada com controlador proporcional, integral e feedforward*



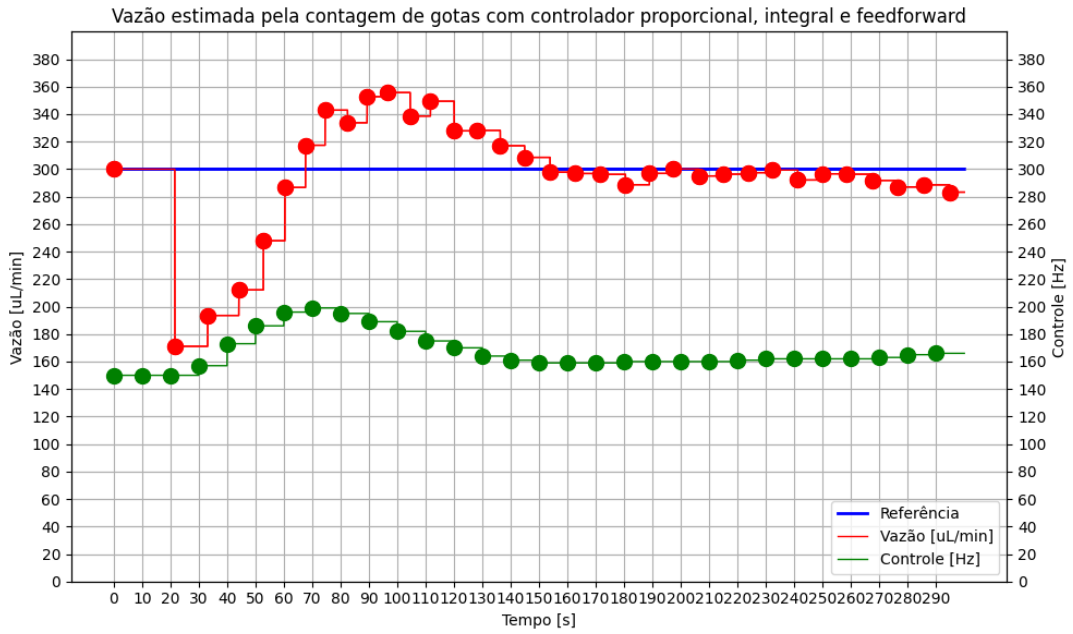
Fonte: O Autor

O ensaio da Figura 12 confirma a suspeita das oscilações induzidas pelas perturbações. Em particular, a correção calculada pelo controlador em decorrência do menor período da gota detectada em $t = 90,92\,s$ leva ao decréscimo da velocidade angular do motor de passo, e, por consequência, uma menor vazão estimada seguinte. Essa vazão reduzida leva o controlador a aumentar a velocidade, aumentando a próxima vazão estimada. O processo é repetido três vezes. As oscilações convergem novamente à referência em $t = 135\,s$, mas novas oscilações parecem começar na região final do teste. A amplitude das oscilações alcança quase 20% da vazão de referência, reduzindo a qualidade do sinal da saída. Ensaiam-se controladores de ganhos K_p e K_i alternativos, onde embora um conjunto de ganhos de menor módulo reduza o efeito visualizado, ele permaneceu presente em testes subsequentes.

Propõe-se reduzir o impacto das perturbações através da implementação de um filtro do tipo *média móvel* na rotina de processamento do sinal do sensor. No algoritmo alterado, consideram-se os períodos das duas últimas gotas detectadas para o cálculo da estimação da vazão, em lugar de apenas o período da última gota. Também, fixa-se o período de cálculo da lei de controle em $T = 10\,s$. O período fixo evita variações de ganhos

do controlador em decorrência do período de amostragem induzido pelo cálculo da lei de controle quando uma nova gota é detectada ser variável, além de manter a função de transferência discreta do controlador constante. Por fim, reduz-se K_p e K_i para $0,20\,mm^{-3}$ e $0,50\,mm^{-3}$, respectivamente, para amortecer a resposta do controlador às perturbações. Espera-se um aumento do regime transiente devido a resposta mais lenta do controlador. O ensaio com as alterações propostas é dado na Figura 13, onde o tempo de operação passou para 300 segundos.

Figura 13: *Estimação da vazão da bomba em malha fechada com controlador proporcional, integral, feedforward, e filtro média móvel*



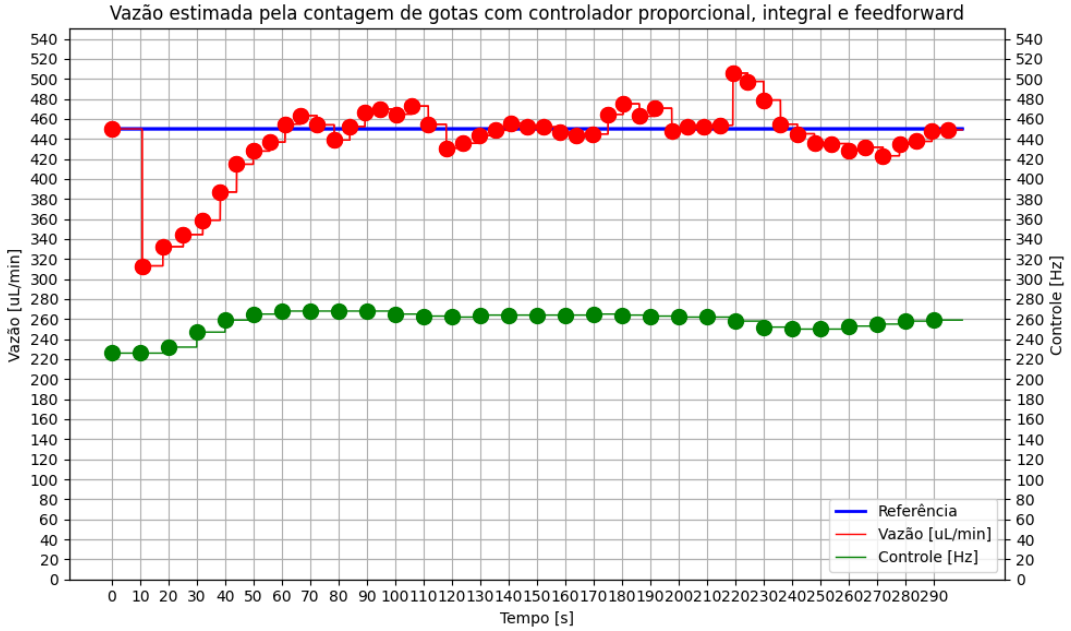
Fonte: O Autor

Percebe-se seguimento de referência de maior robustez no ensaio da Figura 13, onde as perturbações foram melhor amortecidas em relação à Figura 12. Em contrapartida, a diminuição dos ganhos do controlador e a filtragem do sinal do sensor aumentaram o período transiente do teste, que só chegou ao fim em aproximadamente $t = 130\,s$. Essa aumento é ocasionado pelo atraso da estimação da vazão em virtude do emprego do filtro *média móvel*, em conjunto com a maior constante de tempo do controlador. Mais, observou-se um maior sobrepasso, de cerca de 20%, ocasionado pelo acúmulo da ação integral durante o regime transiente do ensaio. Em regime permanente, o erro sistemático da estimação de vazão, calculada a partir do período das últimas duas gotas detectadas, foi inferior a 7%. Por fim, o volume infundido teórico para esse ensaio foi $1500\,\mu L$, ou 35 gotas, de acordo com a Equação 9. Detectou-se a queda de 33 gotas entre o início e o fim do ensaio, aproximadamente 94% do total esperado.

Para validar o controlador projetado, em particular para verificar o desempenho do protótipo para diferentes referências de vazão, realizam-se dois ensaios adicionais. O primeiro ensaio com referência de vazão $Q_{\text{ref}} = 450\,\mu L/min = 7,5\,\mu L/s$, e o segundo ensaio com referência de vazão $Q_{\text{ref}} = 150\,\mu L/min = 2,5\,\mu L/s$. O tempo do segundo ensaio é aumentado para 600 segundos a fim de observar-se um número suficiente de gotas durante a operação. Visualizam-se os experimentos nas Figuras 14 e 15.

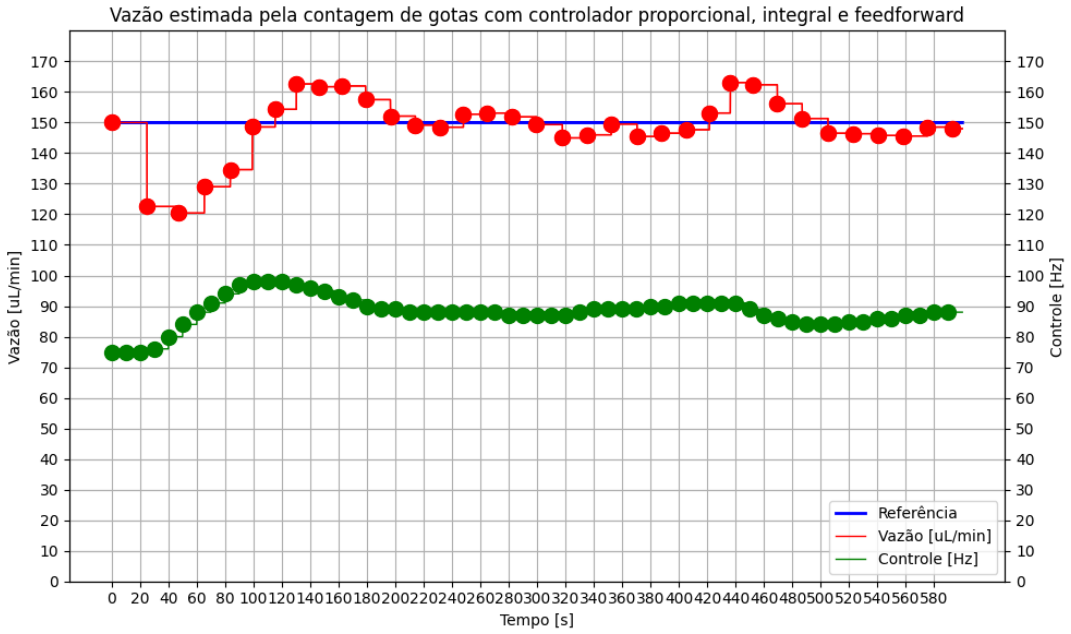
Observa-se no ensaio da Figura 14 menor sobrepasso em relação ao ensaio da Figura

Figura 14: *Estimação da vazão da bomba em malha fechada com controlador proporcional, integral, feedforward, e filtro média móvel*



Fonte: O Autor

Figura 15: *Estimação da vazão da bomba em malha fechada com controlador proporcional, integral, feedforward, e filtro média móvel*



Fonte: O Autor

13. Pensa-se que essa redução é proveniente da vazão estimada das quedas das primeiras gotas serem relativamente mais próximas à referência quando comparadas com as mesmas estimativas na Figura 13. Essa diferença levou a uma diminuição do regime transiente, e por consequência um menor acúmulo da ação integral e menor sobrepasso. A partir de $t = 50\,s$ a vazão estimada apresentou erro de regime predominantemente inferior a 7%, exceto entre os tempos $t = 218\,s$ e $t = 230\,s$, onde perturbações aleatórias levaram à queda mais rápida de duas gotas, e por consequência o aumento do erro para cerca de 12%. O total infundido foi próximo de 96% do total esperado nesse ensaio, sendo 50 gotas detectadas de 52 gotas esperadas.

O seguimento de referência para uma vazão de $Q_{\text{ref}} = 150\,\mu L/min$ é visualizado no ensaio da Figura 15. Percebe-se que um maior regime transiente nesse ensaio, de cerca de 180 segundos, além de sobrepasso próximo de 8%. Verifica-se que o máximo erro de regime também foi da ordem de 8%, e que foram detectadas 34 gotas de 35 gotas esperadas (volume infundido de aproximadamente 97% do volume total esperado). A partir da comparação dos ensaios das Figuras 13, 14 e 15 constata-se que há uma variação da razão entre a taxa de amostragem do sistema e o período de resposta do controlador. No primeiro ensaio, o controlador calcula a lei de controle em períodos próximos aos períodos de queda das gotas, no segundo ensaio a lei de controle é recalculada após a detecção de cerca de duas gotas, e no terceiro ensaio ocorrem em média dois cálculos da lei de controle para cada gota. Dessa forma, a referência de vazão, as características dinâmicas do protótipo e a ocorrência de perturbações mudam a taxa de amostragem do sistema, o que tem impacto direto no desempenho do controlador linear proposto. Para trabalhos futuros, sugere-se estudar a implementação de técnicas avançadas de controle, como Controle Adaptativo baseado no Tempo de Amostragem, Controle por Reamostragem (*Event-Triggered-Control)*, ou Controle Robusto $H\infty$ ou $\mu - Synthesis$ (BOURDAIS et al., 2020), a fim de incorporar as variações nos períodos de amostragem ao controlador.

Por fim, conclui-se que a Bomba de Infusão construída possui desempenho de vazão de ordem de grandeza similar aos protótipos de baixo custo encontrados na literatura (ARÊDES et al., s.d.[a]) (ARÊDES et al., s.d.[b]) (LAKE; HEYDE; RUDER, 2017) (WIJNEN et al., 2014). Segundo (LAKE; HEYDE; RUDER, 2017), erros dessa magnitude são aceitáveis em diversas aplicações laboratoriais. Todavia, é comum bombas de infusão comerciais utilizadas na área da saúde apresentarem erro de vazão de $0,5\%$ a 5%, controle de total infundido, controle de pressão, detecção de bolhas, alarmes de utilização e outros (ALVES, 2002). Dessa forma, destaca-se que a utilização do protótipo é limitada aos requisitos de desempenho da aplicação, sendo necessários aprimoramentos e certificações para uso comercial.

5 CONCLUSÕES

Neste trabalho, apresentou-se os princípios de funcionamento do equipamento Bomba de Infusão encontrados em produtos comerciais, elencou-se suas vantagens e desvantagens e citou-se algumas de suas aplicações. Revisou-se a literatura disponível, priorizando a ampla gama de trabalhos pertinentes à construção de protótipos do dispositivo. Através da análise realizada, escolheu-se desenvolver e construir uma bomba de infusão do tipo bomba de seringa, com mecanismo de acionamento composto por rosca sem fim, em virtude de ponderação de critérios de desempenho, custo e complexidade.

Identificou-se os componentes que compõe a bomba de seringa e as relações entre eles. Derivou-se um modelo dinâmico genérico do funcionamento do equipamento a partir da análise dos princípios físicos que o descrevem. Consideraram-se hipóteses simplificadoras a fim de obter uma relação entrada/saída para condições ideais de operação. Destacaram-se as limitações do modelo, onde sugeriu-se a instrumentação da vazão de saída através de um sensor de contagem de gotas. A adição do sensor foi proposta com o fim de regular os erros de operação em malha aberta, rejeitar perturbações, e melhorar a qualidade da vazão de saída do protótipo. Encontraram-se peças disponíveis comercialmente para confecção da bomba, como motor, *driver*, microcontrolador, sensor, partes mecânicas e partes elétricas. Ponderou-se a aplicabilidade de cada componente, onde foram descritas as funcionalidades individuais e o processo de integração.

Construiu-se a bomba através da montagem mecânica, realização do circuito elétrico, desenvolvimento do *firmware*, e desenvolvimento do programa supervisório responsável por enviar e receber dados do microcontrolador. Montou-se a instrumentação da saída da bomba, onde o sensor foi posicionado e ajustado de forma que as gotas originadas da pressão positiva induzida pelo mecanismo despenquem em frente ao sensor. Realizaram-se ensaios em malha aberta onde foi possível verificar o desempenho do protótipo, presença de perturbações e limitações do modelo. A partir dos resultados dos ensaios e do modelo, projetaram-se algoritmos de controle cuja entrada foi a diferença entre a referência de vazão e a vazão estimada pelo período de queda das gotas. Obtiveram-se melhores resultados através do uso de um controlador discreto proporcional, integral e com ação *feedforward*. Em conjunto ao controlador, implementou-se filtragem do tipo *média móvel* para a estimação de vazão do sensor com o fim de reduzir o impacto das perturbações e melhorar as características de controle. O sistema de controle foi bem sucedido em regular o erro de vazão do protótipo para diferentes referências, onde mediu-se erro de regime inferior a 7% na maior parte dos ensaios, e picos de até 12% devido a ocorrências de perturbações pontuais. Adicionalmente, o total infundido variou de 94% a 97% do total esperado para diferentes referências de vazão. Por fim, avaliou-se que o protótipo é suficiente para certas aplicações de pesquisa laboratoriais, mas é limitado e requer melhorias e certificações para utilização comercial na área da saúde.

5.1 Sugestões para Trabalhos Futuros

Com base no estudo realizado e no protótipo desenvolvido, sugere-se para trabalhos futuros:

1. Desenvolver o *layout* a partir do diagrama elétrico, possivelmente criando um *shield* para o *Arduino* da Bomba de Infusão. Desenvolver uma interface gráfica para o Supervisório a fim de simplificar o uso do equipamento. Projetar as peças de uma estrutura para o posicionamento do equipo e do sensor infravermelho de forma robusta.

2. Aperfeiçoar o modelo dinâmico apresentado através da inclusão das não idealidades previstas e verificadas experimentalmente. Em especial, estudar a dinâmica do acomodamento do êmbolo no início da operação e o efeito dos atritos durante a infusão.

3. Propor novos métodos de instrumentação para a medição de vazão, como a integração de sensores de pressão, medidores de vazão e sensores de deslocamento do êmbolo. Investigar a viabilidade de incorporar múltiplos sensores e explorar o uso de técnicas avançadas de controle, incluindo controladores robustos e filtros estocásticos, para aprimorar o desempenho do equipamento.

4. Adequar o protótipo às normas técnicas relativas ao uso da Bomba de Infusão como Equipamento Eletromédico (EEM) no Brasil, em particular as normas (ABNT, 2016) e (ABNT, 2015). Estudar e realizar as etapas necessárias para o aprimoramento do projeto para uso em contextos médicos.

REFERÊNCIAS

ABNT. *NBR-IEC-60101-1: Equipamento Eletromédico Parte 1: Requisitos essenciais para segurança básica e desempenho essencial*. Rio de Janeiro, 2016.

ABNT. *NBR-IEC-60101-2-24: Requisitos particulares para segurança básica e desempenho essencial de bombas de infusão e controladores de infusão*. Rio de Janeiro, 2015.

AKKOYUN, F.; OZÇELIK, A. A Simple Approach for Controlling an Open-Source Syringe Pump. *European Mechanical Science*, Ahmet ÇALIK, v. 4, n. 4, p. 166–170, 2020. DOI: 10.26701/ems.769837.

ALVES, M. A. D. C. *Bombas de infusão: operação, funcionalidade e segurança*. 2002. F. 109. Tese (Mestrado em engenharia) – Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal de Santa Catarina. Disponível em: <https://repositorio.ufsc.br/xmlui/handle/123456789/83591>.

AMARANTE, L. M. et al. An Open Source Syringe Pump Controller for Fluid Delivery of Multiple Volumes. *eNeuro*, Society for Neuroscience, v. 6, n. 5, 2019. DOI: 10.1523/ENEURO.0240-19.2019. eprint: https://www.eneuro.org/content/6/5/ENEURO.0240-19.2019.full.pdf. Disponível em: <https://www.eneuro.org/content/6/5/ENEURO.0240-19.2019>.

ARDUINO. *Arduino Uno Rev3*. [S.l.: s.n.], 2024. https://www.arduino.cc/en/Guide/ArduinoUno.

ARÊDES, S. et al. Desenvolvimento de uma bomba infusora volumétrica de custo reduzido e fácil operação. Disponível em: <http://mtc-m16b.sid.inpe.br/col/sid.inpe.br/mtc-m17@80/2006/12.06.16.39/doc/aredes_desenvolvimento.pdf>.

ARÊDES, S. et al. Projeto de uma bomba de infusão de baixo custo. Disponível em: <https://www.inicepg.univap.br/cd/INIC_2005/inic/IC3%20anais/IC3-8.pdf>.

ATHANI, V. *Stepper Motors : Fundamentals, Applications And Design*. [S.l.]: New Age International (P) Ltd., Publishers, 1997. ISBN 9788122410068. Disponível em: <https://books.google.com.br/books?id=0m8NTozFZL8C>.

BEDNARSKI, B.; JACKIEWICZ, K.; GAŁECKI, A. Influence of microstepping signal shape on shaft movement precision and torque variation of the stepper motor. *Energies*, MDPI, v. 14, n. 19, p. 6107, 2021.

BERMAN, A. D.; DUCKER, W. A.; ISRAELACHVILI, J. N. Origin and Characterization of Different Stick−Slip Friction Mechanisms. *Langmuir*, v. 12, n. 19, p. 4559–4563, 1996. DOI: 10.1021/la950896z. eprint: ttps://doi.org/10.1021/la950896z. Disponível em: <https://doi.org/10.1021/la950896z>.

BOOESHAGHI, A. et al. Principles of open source bioinstrumentation applied to the poseidon syringe pump system. *Scientific Reports*, v. 9, ago. 2019. DOI: 10.1038/s41598-019-48815-9.

BOURDAIS, R. et al. *Automatic Control*. [S.l.]: CentraleSupélec, 2020.

CHEVET, T. et al. *Model Representation and Analysis*. [S.l.]: CentraleSupélec, 2020.

DANOWITZ, A. *Dual-Axis Precision Imager*. 2019. F. 18. Tese (Trabalho de graduação em Ciência da Computação) – California Polytechnic State University, San Luis Obispo. Disponível em: <https://digitalcommons.calpoly.edu/cpesp/335>.

DE AMORIN, J. I. L. *DESENVOLVIMENTO DE PROTÓTIPO: SISTEMA DE ACIONAMENTO PARA BOMBA DE INFUSÃO DE SERINGA*. 2014. F. 34. Tese (Trabalho de Conclusão de Curso) – Departamento de Ciência da Computação da Universidade Estadual da Paraíba. Disponível em: <https://dspace.bc.uepb.edu.br/jspui/bitstream/123456789/5039/1/PDF%20-%20Jos%C3%A9%20Izaac%20Leite%20de%20Amorim.pdf>.

DIAS, F. W. L. *DESENVOLVIMENTO DE SISTEMA COM BOMBA DE INFUSÃO DE SERINGA PARA PLICAÇÕES EM BIOSSENSORES*. 2019. F. 50. Tese (Mestrado em engenharia) – Programa de Pós-Graduação em Engenharia Elétrica do Instituto Federal da Paraíba. Disponível em: <hhttps://repositorio.ifpb.edu.br/handle/177683/884>.

DYLAN, G. et al. Design of a Novel, Adjustable Flow Rate, Reusable, Electricity-Free, Low-Cost Syringe Infusion Pump. *Journal of Medical Devices*, v. 11, n. 4, p. 041006, out. 2017. ISSN 1932-6181. DOI: 10.1115/1.4037935. eprint: https://asmedigitalcollection.asme.org/medicaldevices/article-pdf/11/4/041006/6239641/med_011_04_041006.pdf. Disponível em: <https://doi.org/10.1115/1.4037935>.

EVALUATION of measurement data – Guide to the expression of uncertainty in measurement. Sèvres, France: Joint Committee for Guides in Metrology (JCGM), 2008. JCGM 100:2008. Disponível em: <https://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf>.

GARCIA, V. E.; LIU, J.; DERISI, J. L. Low-cost touchscreen driven programmable dual syringe pump for life science applications. *HardwareX*, v. 4, e00027, 2018. ISSN 2468-0672. DOI: https://doi.org/10.1016/j.ohx.2018.e00027. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2468067218300269>.

GENNES, P.-G. DE; BROCHARD-WYART, F.; QUÉRÉ, D. *Capillarity and Wetting Phenomena: Drops, Bubbles, Pearls, Waves*. New York: [s.n.], 2004. ISBN 978-0-387-00592-8.

HAYNES, W. M. *CRC Handbook of Chemistry and Physics*. 96th. Boca Raton, FL: CRC Press, 2015. ISBN 978-1482260960.

HUGHES, A. *Electric Motors and Drives: Fundamentals, Types and Applications*. [S.l.]: Elsevier Science, 2005.

INC., M. T. *ATmega328P Datasheet*. [S.l.: s.n.], 2024. https://www.microchip.com/wwwproducts/en/ATmega328P.

IQTEIT, N. A. et al. Simple Mathematical and Simulink Model of Stepper Motor. *Energies*, v. 15, n. 17, 2022. ISSN 1996-1073. DOI: 10.3390/en15176159. Disponível em: <https://www.mdpi.com/1996-1073/15/17/6159>.

JOY-IT. *KY-032 Infrared Obstacle Avoidance Sensor*. [S.l.: s.n.], 2024. https://joy-it.net/en/products/SEN-KY032IR.

JUAREZ, A. et al. AutoSyP: A Low-Cost, Low-Power Syringe Pump for Use in Low-Resource Settings. *American Journal of Tropical Medicine and Hygiene*, v. 95, jul. 2016. DOI: 10.4269/ajtmh.16-0285.

KLESPITZ, J.; KOVÁCS, L. Peristaltic pumps — A review on working and control possibilities. In: 2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI). [S.l.: s.n.], 2014. P. 191–194. DOI: 10.1109/SAMI.2014.6822404.

LAKE, J. R.; HEYDE, K. C.; RUDER, W. C. Low-cost feedback-controlled syringe pressure pumps for microfluidics applications. *PLOS ONE*, Public Library of Science, v. 12, n. 4, p. 1–12, abr. 2017. DOI: 10.1371/journal.pone.0175089. Disponível em: <https://doi.org/10.1371/journal.pone.0175089>.

LINARES-FLORES, J.; SIRA-RAMIREZ, H. DC motor velocity control through a DC-to-DC power converter. In: 2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601). [S.l.: s.n.], 2004. v. 5, 5297–5302 vol.5. DOI: 10.1109/CDC.2004.1429649.

LU, T. A survey on risc-v security: Hardware and architecture. *arXiv preprint arXiv:2107.04175*, 2021.

MANDEL, J. E. Understanding Infusion Pumps. *Anesthesia & Analgesia*, v. 126, n. 4, p. 1186–1189, 2018. DOI: 10.1213/ANE.0000000000002396.

MATHWORKS. *Stepper Motor*. 2023. Disponível em: <https://www.mathworks.com/help/sps/powersys/ref/steppermotor.html>. Acesso em: 4 jan. 2024.

MENGXI, W. et al. Acoustofluidic separation of cells and particles. *Microsystems and Nanoengineering*, v. 5, dez. 2019. DOI: 10.1038/s41378-019-0064-3.

MOYLE, J.; DAVEY, A. *Equipamentos em anestesia. 1.ed.* [S.l.]: Artmed, 2000.

PUSCH, K.; HINTON, T. J.; FEINBERG, A. W. Large volume syringe pump extruder for desktop 3D printers. *HardwareX*, v. 3, p. 49–61, 2018. ISSN 2468-0672. DOI: https://doi.org/10.1016/j.ohx.2018.02.001. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2468067217300822>.

PUTRI, A. Z. et al. Implementation of Nema-17 Stepper Motor and SG-90 Servo Motor as Mechanical Drivers on Spinal Needle Positioning Test Equipment. In: 2023 IEEE International Biomedical Instrumentation and Technology Conference (IBITeC). [S.l.: s.n.], 2023. P. 52–57. DOI: 10.1109/IBITeC59006.2023.10390937.

Q., S. et al. From organs on a chip to patient on a chip. *Innovation*, out. 2023. DOI: doi:10.1016/j.xinn.2022.100282.

SILVA, D. G.; COUTINHO, C.; COSTA, C. J. Factors influencing free and open-source software adoption in developing countries—an empirical study. *Journal of Open Innovation: Technology, Market, and Complexity*, v. 9, n. 1, p. 100002, 2023. ISSN 2199-8531. DOI: https://doi.org/10.1016/j.joitmc.2023.01.002. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2199853123000021>.

SKERRETT, E. et al. Evaluation of a low-cost, low-power syringe pump to deliver magnesium sulfate intravenously to pre-eclamptic women in a Malawian referral hospital. English. *BMC Pregnancy and Childbirth*, v. 17, n. 1, 2017. Publisher Copyright: © The Author(s). 2017. ISSN 1471-2393. DOI: 10.1186/s12884-017-1382-9.

TAYLOR, J. R. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. 2nd. Sausalito, California: University Science Books, 1997. ISBN 978-0935702750.

WIJNEN, B. et al. Open-Source Syringe Pump Library. *PLOS ONE*, Public Library of Science, v. 9, p. 1–8, set. 2014. DOI: 10.1371/journal.pone.0107216. Disponível em: <https://doi.org/10.1371/journal.pone.0107216>.

Apêndices

Apêndice A - MONTAGEM

A.1 Componentes Mecânicos

Tabela 2: *Componentes mecânicos para a montagem*

Componente	Quantidade	Valor [R$][1]
Motor de passo NEMA-17[2]	1	180,00
Rolamento Linear de Esferas LM6UU 6x12x19 mm	2	10,00
Parafuso de Máquina M3-0,5 x 16 mm	8	0,30
Parafuso de Máquina M3-0,5 x 20 mm	2	0,30
Porca M3-0,6	6	0,20
Porca Quadrada M5-8x8x4 mm	1	3,00
Eixo Linear 6x100 mm	2	60,00
Barra Roscada M5-0,8 de 100 mm	1	17,00
Acoplamento Flexível 5x5 mm	1	15,00
Seringa 5 mL[3]	1	1,00
Sonda de Aspiração Traqueal №10	1	1,00

Fonte: Adaptado de (lake; heyde; ruder, 2017)

Notas: [1] Valores por unidade relativos a 2024
[2] P/N: 17HS19-2004S1; Tipo: Bipolar; Rotação por passo: $1,8°$; Torque de retenção: $59\,N\,cm$ $(6,016\,kgf\,cm)$; Corrente nominal: $2\,A$; Tensão de condução $12\,V$ a $24\,V$.
[3] Bico tipo *Luer Lock*

A.2 Instruções de Montagem

1. Parafusar o motor NEMA-17 ao Conector do Motor da Bomba de Seringa com 4 parafusos M3. Fixar os parafusos na parte traseira do Conector com porcas M3.

2. Acoplar a Barra Roscada M5 ao eixo do Motor NEMA-17 com o Acoplamento Flexível de 5 mm.

3. Conectar mecanicamente a Plataforma para Seringa de 5 mL ao Conector do Motor da Bomba de Seringa.

Tabela 3: *Peças impressas 3D necessárias*

Componente	Quantidade
Fixador de Seringa de 5 mL	1
Plataforma para Seringa de 5 mL	1
Conectores do êmbolo da seringa	1+1[1]
Fim de Curso da Bomba de Seringa	1
Conector do Motor da Bomba de Seringa	1
Dispositivo de Avanço da Bomba de Seringa	1

Fonte: Adaptado de (LAKE; HEYDE; RUDER, 2017)

Notas: [1] O êmbolo da seringa é fixado ao Dispositivo de Avanço através de dois conectores que têm um encaixe mecânico mútuo.
[2] Valor comercial estimado para impressão das peças, relativo a 2024, de R$ 80,00.

4. Inserir ambos os Rolamentos Lineares de Esferas nas cavidades inferiores do Dispositivo de Avanço da Bomba de Seringa.

5. Inserir a Porca Quadrada M5-0,8 na parte inferior do Dispositivo de Avanço.

6. Roscar o Dispositivo de Avanço na Barra Roscada conectada ao motor pela Porca Quadrada. A parte reta da plataforma do Dispositivo de Avanço deve ficar na direção do motor NEMA-17.

7. Alinhar o Dispositivo de Avanço com o Conector do Motor com os Eixos Lineares de 6x100 mm. Eles devem passar pelos Rolamentos Lineares de Esferas do Dispositivo de Avanço e ser fixados no Conector do Motor.

8. Parafusar um dos Conectores do êmbolo da seringa ao Dispositivo de Avanço.

9. Posicionar a seringa de 5 mL sobre a Plataforma para Seringa. Fixá-la parafusando o Fixador de Seringa sobre a Plataforma para Seringa.

10. Puxar o êmbolo da seringa até encontrar o Dispositivo de Avanço. Fixá-lo ao Dispositivo de Avanço através do segundo Conector do êmbolo.

11. Travar os Eixos Lineares ao Conector do Motor e ao Fim de Curso parafusando quatro parafusos M3 com porcas (dois em cada peça).

12. Conectar a Sonda de Aspiração Traqueal №10 ao bico *Luer Lock* da seringa.

A.3 Componentes Elétricos

Tabela 4: *Componentes elétricos*

Componente	Quantidade	Valor [R$][1]
Microcontrolador Arduino UNO Rev3	1	80,00
Driver de motor de passo A4988	1	15,00
Sensor infravermelho KY-032	1	15,00
Fonte de alimentação $12\,V$ $2\,A$	1	30,00
Chave *SPST*[2]	1	1,50
Botão de 4 pinos	2	0,50
Resistor de $100\,k\Omega$	2	0,15
Resistor de $10\,k\Omega$	2	0,15
Capacitor Eletrolítico de $100\,\mu F$	1	0,40
Capacitor Eletrolítico de $1\,\mu F$	3	0,40
Capacitor Cerâmico de $100\,nF$	2	0,15
Breadboard 830 pontos	1	15,00
Jumpers Macho/Macho	<50	10,00

Fonte: O Autor

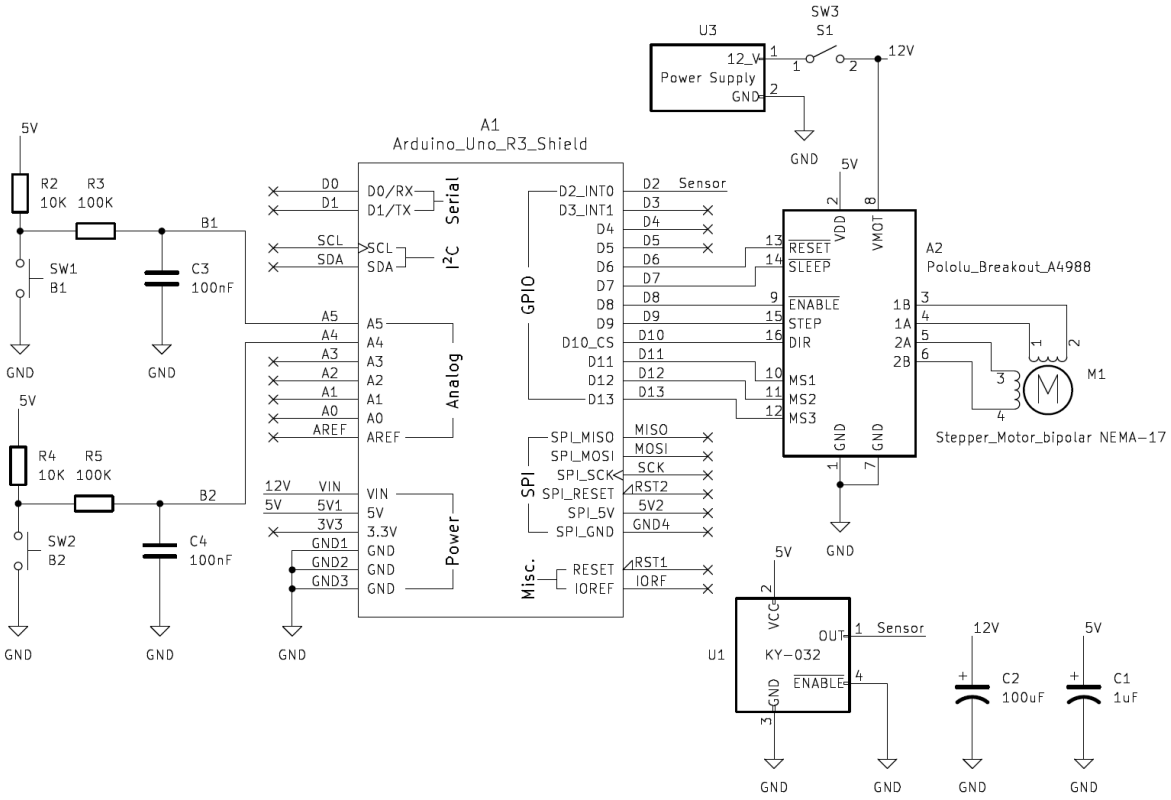
Notas: [1] Valores por unidade relativos a 2024
[2] *Single Pole, Single Throw*

A.4 Consideração sobre Custos

O valor estimado total do protótipo é $R\$\,611,20$. Todavia, prevê-se um aumento de cerca de 10% a fim de considerar taxas de entrega. Os valores dos componentes foram determinados através de uma média de consultas a lojas e distribuidores nacionais. O valor de cada componente pode variar sensivelmente se a peça for consultada em uma loja física. Por exemplo, uma fornecedora de rolamentos de Porto Alegre propôs o valor de $R\$\,38,00$ para cada rolamento linear LM6UU, sendo a mesma peça disponível *online* por cerca de $R\$\,10,00$ em um estabelecimento de componentes eletrônicos de São Paulo.

A.5 Diagrama Elétrico

Figura 16: *Diagrama Elétrico da Bomba de Infusão*



Fonte: O Autor

A.6 Considerações para Operação

1. Acoplar o dissipador de calor do *driver* A4988 ao chip.

2. Ajustar a corrente limite do *driver* A4988 para o valor nominal do motor de passo.

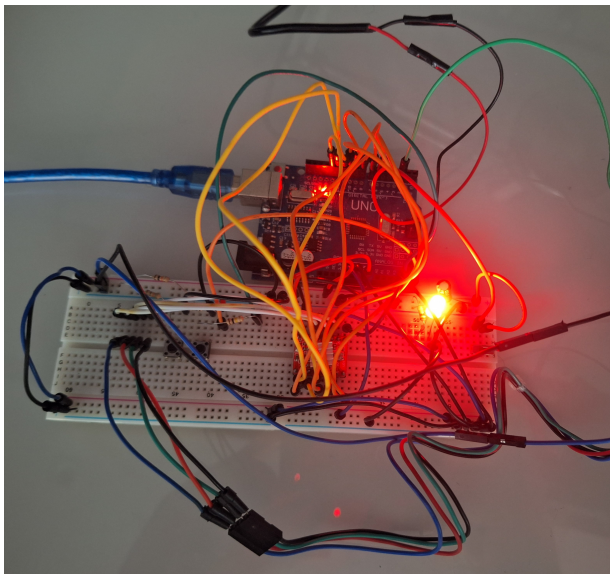
3. Verificar a suavidade do deslizamento da plataforma ao longo dos eixos lineares. Devido às tolerâncias dimensionais das peças impressas 3D, pode ocorrer atrito entre os eixos e o Dispositivo de Avanço. Caso isso aconteça, desgastar levemente a parte da peça em contato com o eixo até a obtenção de movimento suave.

4. Certificar-se que não há ar na Sonda Traqueal. Utilizar os botões de ajuste manual para mover o êmbolo da seringa até verificar-se a queda de algumas gotas, antes de iniciar uma operação de infusão.

5. Certificar-se que a sonda traqueal está posicionada de forma que o sensor infravermelho seja capaz de detectar a queda da gota. Verificar a ativação do LED de detecção do sensor quando a gota passa em frente ao sensor.

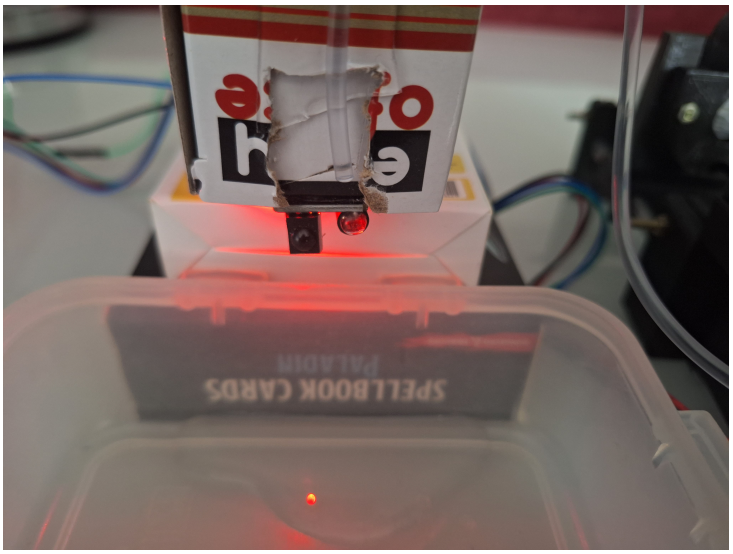
A.7 Circuito e Posicionamento do Sensor

Figura 17: *Realização do diagrama elétrico em breadboard*



Fonte: O Autor

Figura 18: *Posicionamento do sensor KY-032 em frente à queda das gotas*



Fonte: O Autor

Apêndice B - FIRMWARE

```
/*******************************************************************************
                          Open-source Syringe Pump Firmware
                          Author:  Artur Piana Broilo
                          Contact: artur.p.broilo@gmail.com
                          School:  UFRGS
                          Date:    08/2024
*******************************************************************************/
#define PUSH 1
#define PULL -1

#define ANALOG_LOW_THRESHOLD_10_bit 256

// Pin definitions
const uint8_t dirPin = 10;
const uint8_t stepPin = 9;
const uint8_t MS3Pin = 13;
const uint8_t MS2Pin = 12;
const uint8_t MS1Pin = 11;
const uint8_t sleepPin = 7; // Active low
const uint8_t rstPin = 6; // Active low
const uint8_t enablePin = 9; // Active low
const uint8_t manualPULLButton = A4; // Pull UP
const uint8_t manualPUSHButton = A5;  // Pull UP
const uint8_t digitalSensorPin = 2; // Infrared drop sensor

// Serial communications
const uint8_t incomingDataBufferSize = 64;
//const uint16_t readingDelay = 500; // [us]
uint8_t incomingDataBuffer[incomingDataBufferSize] = {0};

// Geometry
const float threadLead = 0.8; // [mm/rotation]
const float syringeDiameter = 13; // [mm]

// Fluid flow
uint16_t maxFlowRateVector[5]; // [uL/min] : max flow rate for each stepping
    configuration, to be computed
uint16_t maxFlowRate = 0; // current maxFlowRate, to be computed
const uint32_t maxFlowTime = 172800; // [s] equivalent of 48 hours of continuous work
const float flowPerRotation = threadLead*PI*sq(syringeDiameter/2); // [uL/rotation]
int32_t totalFlow = 0; // total flow given a setup configuration
volatile double currentFlow = 0; // accumulated flow until present momemt

// Motor constants
const uint8_t steppingVector[5] = {1, 2, 4, 8, 16};
const uint8_t baseStepsPerRotation = 200;
```

```
const uint16_t maxStepsPerSecond = 1000;
const uint8_t maxPossibleRotationsPerSecond = maxStepsPerSecond/baseStepsPerRotation;
const uint8_t motor_interface_type = 1;

// Inputs
int8_t directionRotation = PUSH; // PUSH/PULL
uint8_t stepping = 1;
uint16_t flowRate = 0; // [uL/min]
uint32_t flowTime = 0; // [s]

// Operation variables
double rotationsPerMinute = 0;
double rotationsPerSecond = 0;
double numberOfRotations =  0;
double stepsPerRotation = 0;
int32_t stepsToGo = 0; // [steps]
int32_t newPosition = 0; // [steps]
int16_t motorSpeed = 0; // [steps/s]

// Output flags
uint8_t steppingACK = 0;
uint8_t directionACK = 0;
uint8_t flowRateACK = 0;
uint8_t flowTimeACK = 0;
uint8_t setupACKStart = 0;
uint8_t setupACKPartial = 0;
uint8_t setupACK = 0;
uint8_t resetACK = 0;
uint8_t manualPUSHACK = 0;
uint8_t manualPULLACK = 0;
uint8_t controlACK = 0;

// State variables
uint8_t jobRunning = 0;
bool manualMovement = false;

// Time variables
uint64_t checkCommunicationMillis = 0;
uint8_t checkCommunicationInterval = 5;
volatile uint64_t checkDataTransferMillis = 0;
const uint16_t dataTransferInterval = 1000;
volatile uint64_t startJobTime = 0;
uint64_t manualMovementMillis = 0;

// Sensor variables
const float volumePerDrop = 43.0;        // Microliters
const uint16_t sensorDebounceDelay = 5; // Debounce period in milliseconds
volatile bool dropDetected = false;
volatile uint64_t lastDetectTimeMillis = 0;
volatile uint16_t dropCount = 0;
volatile uint64_t lastDropDetectMillis = 0;
volatile uint32_t dropTime = 0;         // Milliseconds
volatile float estimateFlowRate = 0;    // [uL/min]

// Controller variables
uint8_t control_enable = 1;
uint16_t controllerSampleInterval = 10000;
volatile uint64_t lastControlTimeMillis = 0;
```

```
float Kf = 1.00;
float Kp = 0.20;
float Ki = 0.50;
uint16_t flowControlSignal = 0;
float flowControlState = 0;
float flowControlError = 0;

void setup();
void loop();
void sensorISR();
void enable();
void disable();
void checkPosition();
void checkACK();
void readManualOverrideButtons();
void readSensorData();
void computeControlSignal();
void readSerialInput();
uint32_t getMaxFlowRate(int stepping);
void setOpenLoopParameters();
void resetSetup();
void resetACKSignals();
void resetOpenLoopParameters();
void setStepping(uint8_t stepping);
void printSetupInfo();
void printInfo();
void printOpenLoopParameters();
void printLinearDistance();
void setupPWM(float frequency);
void disablePWM();

// timer 1 compare interrupt service routine
ISR(TIMER1_COMPA_vect)
{
  PORTB ^= (1 << PORTB1);  // Toggle pin 9 (PB1)
}

// Interrupt service routine of sensor
void sensorISR() {
  dropDetected = true;
}


// Power on the motor driver and stepper motor
void enable(){
  digitalWrite(enablePin, LOW);
}


// Power off the motor driver and stepper motor
void disable(){
  digitalWrite(enablePin, HIGH);
}


void setup() {

  Serial.begin(115200);
```

```
162
163   // Wait for communication start
164   while(!Serial){
165     ;
166   }
167   Serial.flush();
168
169   // Set pin modes
170   pinMode(MS3Pin, OUTPUT);
171   pinMode(MS2Pin, OUTPUT);
172   pinMode(MS1Pin, OUTPUT);
173   pinMode(sleepPin, OUTPUT);
174   pinMode(rstPin, OUTPUT);
175   pinMode(enablePin, OUTPUT);
176   pinMode(dirPin, OUTPUT);
177
178   // Compute max flow rates based on max number of steps/s
179   for(int i = 0;i < 5; i++){
180     maxFlowRateVector[i] =
      (maxStepsPerSecond*60/(baseStepsPerRotation*steppingVector[i]))*flowPerRotation;
      // [uL/minute]
181   }
182
183   // Set chip control configuration
184   digitalWrite(sleepPin, HIGH); // Active low
185   digitalWrite(rstPin, HIGH); // Active low
186   digitalWrite(dirPin, HIGH); // HIGH: Push, LOW: Pull
187
188   attachInterrupt(digitalPinToInterrupt(digitalSensorPin), sensorISR, FALLING);
189
190   disable();
191   disablePWM();
192
193 }
194
195
196
197 // Responsible for the Syringe Pump communication
198 void loop() {
199
200   uint64_t currentMillis = millis();
201
202   if(currentMillis - lastDetectTimeMillis > sensorDebounceDelay){
203     lastDetectTimeMillis = currentMillis;
204
205     readSensorData();
206   }
207
208   if(currentMillis - lastControlTimeMillis > controllerSampleInterval){
209     lastControlTimeMillis = currentMillis;
210
211     if(jobRunning && control_enable)
212     {
213       computeControlSignal();
214       setupPWM(flowControlSignal);
215     }
216   }
217
```

```
218   if (currentMillis - checkCommunicationMillis > checkCommunicationInterval){
219
220     checkCommunicationMillis = currentMillis;
221
222     // Buttons
223     readManualOverrideButtons();
224
225     // Serial Communication
226     readSerialInput();
227
228     // Verify and treat incoming signals
229     checkACK();
230
231     // Check if job is complete
232     checkPosition();
233   }
234
235   if(manualMovement){
236     if(currentMillis - manualMovementMillis > 100){
237       manualMovement = false;
238       disablePWM();
239     }
240   }
241
242   if (jobRunning){
243     if (millis() - checkDataTransferMillis > dataTransferInterval)
244     {
245       checkDataTransferMillis = millis();
246
247       currentFlow = (currentMillis -
      startJobTime)*flowPerRotation*rotationsPerSecond/1000.0;
248       printInfo();
249     }
250   }
251 }
252
253 void checkPosition(){
254   if (jobRunning){
255     if (((millis() - startJobTime) > flowTime*1000)){
256       disablePWM();
257       disable();
258       jobRunning = 0;
259       currentFlow = (millis() -
      startJobTime)*flowPerRotation*rotationsPerSecond/1000.0;
260       printInfo();
261       resetSetup();
262       resetOpenLoopParameters();
263       Serial.println("\nEnd of job");
264     }
265   }
266 }
267
268
269 void checkACK(){
270   if (resetACK){
271     disablePWM();
272     disable();
273     jobRunning = 0;
```

```
274     resetSetup();
275     resetOpenLoopParameters();
276     Serial.println("Reset");
277   }
278 
279   if (manualPUSHACK or manualPULLACK){
280     jobRunning = 0;
281     enable();
282     setStepping(16);
283     if (manualPUSHACK){
284       // Serial.println("Manual PUSH");
285       digitalWrite(dirPin, HIGH);
286 
287     }
288     else if (manualPULLACK){
289       // Serial.println("Manual PULL");
290       digitalWrite(dirPin, LOW);
291     }
292     setupPWM(4000);
293     manualMovement = true;
294     manualMovementMillis = millis();
295   }
296 
297   if (setupACKStart){
298     printSetupInfo();
299 
300     if(setupACK){
301       // Set configuration
302       setStepping(stepping);
303       setOpenLoopParameters();
304       printLinearDistance();
305       printOpenLoopParameters();
306       Serial.println("\nSetup Successful\n");
307       enable();
308       Serial.println("");
309       digitalWrite(dirPin, directionRotation);
310       setupPWM(motorSpeed);
311       jobRunning = 1;
312     }
313     else{
314       Serial.println("\nSetup Failed\n");
315     }
316   }
317 
318   if (controlACK or setupACKStart){
319     resetACKSignals();
320   }
321 }
322 
323 void readManualOverrideButtons(){
324   int manualPUSHInput = analogRead(manualPUSHButton);
325   int manualPULLInput = analogRead(manualPULLButton);
326   if(manualPUSHInput <= ANALOG_LOW_THRESHOLD_10_bit){
327     controlACK = 1;
328     manualPUSHACK = 1;
329   }
330   if(manualPULLInput <= ANALOG_LOW_THRESHOLD_10_bit){
331     controlACK = 1;
```

```
332     manualPULLACK = 1;
333   }
334 }
335 
336 void readSensorData(){
337   if(dropDetected){
338     uint64_t currentTime = millis();
339     uint64_t deltaTime = currentTime - lastDropDetectMillis;
340 
341     if(deltaTime >= 50){
342       //dropTime = deltaTime;
343       dropTime = (deltaTime + dropTime)/2.0;
344       estimateFlowRate = 60.0*1000.0*volumePerDrop/((float)dropTime);
345       lastDropDetectMillis = currentTime;
346       dropCount++;
347     }
348 
349     dropDetected = false;
350   }
351 }
352 
353 void computeControlSignal(){
354 
355   flowControlError = (float)flowRate/60.0 - estimateFlowRate/60.0;
356 
357   float newMotorSpeed = Kf*motorSpeed + (float)(Kp*stepping*flowControlError +
      Ki*stepping*flowControlState);
358 
359   if(newMotorSpeed > 1.5*motorSpeed){
360     flowControlSignal = round(1.5*motorSpeed);
361     return;
362   }
363   if(newMotorSpeed < 0.5*motorSpeed){
364     flowControlSignal = round(0.5*motorSpeed);
365     return;
366   }
367 
368   flowControlState += flowControlError;
369 
370   flowControlSignal = round(newMotorSpeed);
371 }
372 
373 void readSerialInput(){
374   if (Serial.available() > 0){
375 
376     delayMicroseconds(500);
377 
378     int bytesRead = Serial.readBytes(incomingDataBuffer, Serial.available());
379 
380     switch(bytesRead){
381 
382       case 1: // control signal
383       {
384         char controlSignal = incomingDataBuffer[0];
385 
386         // reset signal
387         if (controlSignal == 'R' or controlSignal == 'r'){
388           resetACK = 1;
```

```
389         }
390         // manual PUSH signal
391         else if (controlSignal == '0'){
392           manualPUSHACK = 1;
393         }
394         // manual PULL signal
395         else if (controlSignal == '1'){
396           manualPULLACK = 1;
397         }
398         if (resetACK or manualPUSHACK or manualPULLACK){
399           controlACK = 1;
400         }
401       }
402       break;
403 
404       case 8: // setup signal
405       {
406         setupACKStart = 1;
407         stepping = uint8_t(incomingDataBuffer[0]);
408         directionRotation = int8_t(incomingDataBuffer[1]);
409         // arduino is little endian
410         flowRate = uint16_t(incomingDataBuffer[2])<<8 |
    uint16_t(incomingDataBuffer[3]);
411         flowTime = uint32_t(incomingDataBuffer[4])<<24 |
    uint32_t(incomingDataBuffer[5])<<16 | uint32_t(incomingDataBuffer[6])<<8 |
    uint32_t(incomingDataBuffer[7]);
412 
413         maxFlowRate = getMaxFlowRate(stepping);
414 
415         if (stepping == 1 or stepping == 2 or stepping == 4 or stepping == 8 or
    stepping == 16){
416           steppingACK = 1;
417         }
418         if (directionRotation == 1 or directionRotation == -1){
419           directionACK = 1;
420         }
421         if (flowRate >= 0 and flowRate <= maxFlowRate){
422           flowRateACK = 1;
423         }
424         if (flowTime >= 0 and flowTime <= maxFlowTime){
425           flowTimeACK = 1;
426         }
427         if (steppingACK or directionACK or flowRateACK or flowTimeACK){
428           setupACKPartial = 1;
429         }
430         if (steppingACK and directionACK and flowRateACK and flowTimeACK){
431           setupACK = 1;
432         }
433       }
434       break;
435     }
436     // clear data buffer
437     memset(incomingDataBuffer, 0, incomingDataBufferSize);
438   }
439 }
440 
441 uint32_t getMaxFlowRate(int stepping){
442   if (stepping == 1){
```

```
443       return(maxFlowRateVector[0]);
444     }
445     else if(stepping == 2){
446       return(maxFlowRateVector[1]);
447     }
448     else if (stepping == 4){
449       return(maxFlowRateVector[2]);
450     }
451     else if (stepping == 8){
452       return(maxFlowRateVector[3]);
453     }
454     else if (stepping == 16){
455       return(maxFlowRateVector[4]);
456     }
457   }
458
459   // Must be called once there is a change on the inputs
460   void setOpenLoopParameters(){
461     totalFlow = flowRate*flowTime;
462     totalFlow = totalFlow/60;
463     totalFlow = directionRotation*totalFlow;
464     rotationsPerSecond = (double)(flowRate)/flowPerRotation/60.0;
465     numberOfRotations =  (double)(flowTime)*rotationsPerSecond;
466     stepsPerRotation = ((double)baseStepsPerRotation*(double)stepping);
467     stepsToGo = round(stepsPerRotation*numberOfRotations);
468     newPosition = directionRotation*stepsToGo;
469     motorSpeed = stepsPerRotation*rotationsPerSecond;
470     flowControlSignal = motorSpeed;
471     flowControlState = 0;
472     dropCount = 0;
473     estimateFlowRate = flowRate;
474     uint64_t current_time = millis();
475     lastDropDetectMillis = current_time;
476     startJobTime = current_time;
477     checkDataTransferMillis = current_time;
478     lastControlTimeMillis = current_time;
479     dropTime = 1000.0*volumePerDrop/((float)flowRate/60.0);
480   }
481
482   void resetSetup(){
483     flowRate = 0;
484     flowTime = 0;
485     stepping = 1;
486     directionRotation = PUSH;
487     dropCount = 0;
488     dropDetected = false;
489     estimateFlowRate = 0;
490     flowControlSignal = 0;
491     flowControlState = 0;
492   }
493
494   void resetACKSignals(){
495     steppingACK = 0;
496     directionACK = 0;
497     flowRateACK = 0;
498     flowTimeACK = 0;
499     setupACKStart = 0;
500     setupACKPartial = 0;
```

```
501   setupACK = 0;
502   resetACK = 0;
503   manualPUSHACK = 0;
504   manualPULLACK = 0;
505   controlACK = 0;
506 }
507 
508 void resetOpenLoopParameters(){
509   totalFlow = 0;
510   rotationsPerSecond = 0;
511   numberOfRotations =  0;
512   stepsPerRotation = 0;
513   stepsToGo = 0;
514   newPosition = 0;
515   motorSpeed = 0;
516   flowControlSignal = 0;
517   flowControlState = 0;
518 }
519 
520 void setStepping(uint8_t stepping){
521 
522   switch(stepping){
523 
524     // Full Step
525     case 1:
526     digitalWrite(MS1Pin, LOW);
527     digitalWrite(MS2Pin, LOW);
528     digitalWrite(MS3Pin, LOW);
529     break;
530 
531     // Half Step
532     case 2:
533     digitalWrite(MS1Pin, HIGH);
534     digitalWrite(MS2Pin, LOW);
535     digitalWrite(MS3Pin, LOW);
536     break;
537 
538     // Quarter Step
539     case 4:
540     digitalWrite(MS1Pin, LOW);
541     digitalWrite(MS2Pin, HIGH);
542     digitalWrite(MS3Pin, LOW);
543     break;
544 
545     // Eighth Step
546     case 8:
547     digitalWrite(MS1Pin, HIGH);
548     digitalWrite(MS2Pin, HIGH);
549     digitalWrite(MS3Pin, LOW);
550     break;
551 
552     // Sixteenth Step
553     case 16:
554     digitalWrite(MS1Pin, HIGH);
555     digitalWrite(MS2Pin, HIGH);
556     digitalWrite(MS3Pin, HIGH);
557     break;
558 
```

```
559     default:
560     digitalWrite(MS1Pin, LOW);
561     digitalWrite(MS2Pin, LOW);
562     digitalWrite(MS3Pin, LOW);
563     break;
564   }
565 }
566 
567 void printSetupInfo(){
568   Serial.print("Stepping: ");
569   Serial.print(stepping);
570   Serial.println(" steps");
571   Serial.print("Direction: ");
572   if (directionRotation == PUSH)
573   {
574     Serial.println("PUSH");
575   }
576   else if (directionRotation == PULL)
577   {
578     Serial.println("PULL");
579   }
580   Serial.print("Flow rate: ");
581   Serial.print(flowRate);
582   Serial.println(" uL/min");
583   Serial.print("Flow time: ");
584   Serial.print(flowTime);
585   Serial.println(" s");
586 }
587 
588 void printInfo(){
589   Serial.print(String((float)(millis() - startJobTime)/1000.0) + "  ");
590   //Serial.print("Current flow:\t" + String(currentFlow) + " [uL]\t" +
        String(currentFlow*100.0/totalFlow) + "%\tof  " +  String(totalFlow));
591   Serial.print(" Ref: ");
592   Serial.print(flowRate);
593   Serial.print("  [uL/min] Estimate ");
594   Serial.print(estimateFlowRate);
595   Serial.print("  [uL/min]  ");
596   Serial.print("Drops: ");
597   Serial.print(dropCount);
598   Serial.print("  Drop time:  ");
599   Serial.print((float)(lastDropDetectMillis - startJobTime)/1000.0);
600   Serial.print("  Control:  ");
601   Serial.print(flowControlSignal);
602   Serial.print("  Control time:  ");
603   Serial.print((lastControlTimeMillis - startJobTime)/1000.0);
604   Serial.print("  Error: " + String(flowControlError*60));
605   Serial.println("  State:  " + String(flowControlState*60));
606 }
607 
608 void printOpenLoopParameters(){
609   Serial.print("Rotations per second: ");
610   Serial.println(rotationsPerSecond);
611   Serial.print("Number of rotations: ");
612   Serial.println(numberOfRotations);
613   Serial.print("Steps to go: ");
614   Serial.println(stepsToGo);
615   Serial.print("Motor speed: ");
```

```
  Serial.println(motorSpeed);
}

void printLinearDistance(){
  Serial.print("Distance to move: ");
  Serial.print(numberOfRotations*threadLead, 3);
  Serial.println(" mm");
}

// Function to setup and adjust PWM frequency on pin 9
void setupPWM(float frequency) {

  if(frequency == 0){
    return;
  }
  // Disable the interrupt while updating the frequency
  cli();

  // Variables to hold prescaler and compare match value
  uint16_t prescalerValue;
  uint16_t compareMatch;

  // Determine the appropriate prescaler and compare match value
  if (frequency < 1) {
    prescalerValue = 1024;
  } else if (frequency < 8) {
    prescalerValue = 256;
  } else if (frequency < 64) {
    prescalerValue = 64;
  } else if (frequency < 512) {
    prescalerValue = 8;
  } else {
    prescalerValue = 1;
  }

  compareMatch = (16000000 / (2 * prescalerValue * frequency)) - 1;

  // Ensure compareMatch fits within 16-bit register
  if (compareMatch > 65535) {
    compareMatch = 65535;
  }

  // Configure Timer1 for PWM on pin 9
  TCCR1A = (1 << WGM12);  // CTC mode
  TCCR1B = (1 << WGM12);  // CTC mode

  // Set the prescaler
  switch (prescalerValue) {
    case 1:
      TCCR1B |= (1 << CS10);
      break;
    case 8:
      TCCR1B |= (1 << CS11);
      break;
    case 64:
      TCCR1B |= (1 << CS11) | (1 << CS10);
      break;
    case 256:
```

```
674       TCCR1B |= (1 << CS12);
675       break;
676     case 1024:
677       TCCR1B |= (1 << CS12) | (1 << CS10);
678       break;
679   }
680
681   OCR1A = compareMatch;  // Set compare match value
682   TIMSK1 |= (1 << OCIE1A);  // Enable timer compare interrupt
683
684   // Enable the interrupt after updating the frequency
685   sei();
686 }
687
688 // Function to disable PWM on pin 9
689 void disablePWM() {
690   // Disable the interrupt while stopping the PWM
691   cli();
692
693   // Disable Timer1 compare interrupt
694   TIMSK1 &= ~(1 << OCIE1A);
695
696   // Reset Timer1 configuration to stop PWM
697   TCCR1A = 0;
698   TCCR1B = 0;
699
700   // Ensure the output pin is set to LOW
701   digitalWrite(stepPin, LOW);
702
703   // Enable the interrupt after stopping the PWM
704   sei();
705 }
```

Listing B.1: *Firmware*

Apêndice C - SUPERVISÓRIO

```
from asyncio.windows_events import NULL
from curses import baudrate
from socket import timeout
import struct
from math import pi
from math import floor
from tokenize import String
from turtle import reset
import serial
import serial.tools.list_ports
from pynput import keyboard
import threading


class SyringePumpArduino:

    def __init__(self):
        pass

    def controlWrite(self, output):
        if output in controlProtocol:
            self.serialConnection.write(
                controlSignal(controlProtocol[output]))
            return True
        else:
            return False

    def controlQuery(self, output):
        if output in controlProtocol:
            self.serialConnection.write(
                controlSignal(controlProtocol[output]))
            input = self.serialConnection.readline().decode('ascii')
            return input
        else:
            return NULL

    def setupQuery(self, stepping, direction, flowRate, flowTime):
        self.serialConnection.write(setupSignal(
            stepping, direction, flowRate, flowTime))
        input = self.serialConnection.read_all().decode('ascii')
        return input

    def connectToPort(self, baudrate):
        self.serialConnection = NULL
        ports = list(serial.tools.list_ports.comports())
        print("\nAvailable serial ports:")
```

```
47          for p in ports:
48              print(p)
49
50          if len(ports) > 0:
51              connectionPort = str(ports[0]).split()[0]
52              self.serialConnection = serial.Serial(
53                  connectionPort, baudrate, timeout=1)
54              if self.serialConnection.is_open:
55                  print("Connection to: " + self.serialConnection.name)
56                  return True
57              else:
58                  print("Couldn't connect to port: " + connectionPort)
59          else:
60              print("No ports available")
61
62          return False
63
64
65  def printLine(n):
66      print("*" * n)
67
68
69  def setupSignal(stepping, dir, flowRate, flowTime):
70      data = b''
71      data += struct.pack('!B', stepping)
72      data += struct.pack('!b', dir)
73      data += struct.pack('!H', flowRate)
74      data += struct.pack('!I', flowTime)
75      return data
76
77
78  def controlSignal(control):
79      data = b''
80      data += struct.pack('!B', control)
81      return data
82
83
84  def printControls():
85      print("s\tsetup")
86      print("r\treset")
87      print("a\tmanual pull")
88      print("d\tmanual push")
89      print("ESC\texit\n")
90
91
92  def computeMaxFlowRates():
93      maxStepsPerSecond = 1000  # current arduino value
94      baseStepsPerRotation = 200  # stepper motor precision
95      threadLead = 0.8
96      syringeDiameter = 13.0
97      flowPerRotation = threadLead*pi*(syringeDiameter/2)**2
98      maxFlowRatesList = []
99      maxFlowRates = {}
100     for i in range(0, 5):
101         maxFlowRatesList.append(
102             maxStepsPerSecond*60 /
    (baseStepsPerRotation*steppingList[i])*flowPerRotation)
103         maxFlowRates[steppingList[i]] = maxFlowRatesList[i]
```

```
104
105     return maxFlowRates
106
107
108 def getSetupInput():
109     flowRateCorrect = False
110     flowTimeCorrect = False
111     steppingCorrect = False
112     directionCorrect = False
113
114     while not directionCorrect:
115         print("Direction {push, pull}: ", end='')
116         direction = input().upper()
117         if (direction == "PUSH"):
118             direction = 1
119             directionCorrect = True
120         elif(direction == "PULL"):
121             direction = -1
122             directionCorrect = True
123
124     while not steppingCorrect:
125         print("Stepping {1, 2, 4, 8, or 16} (default: 16): ", end='')
126         stepping = input()
127         if stepping.isnumeric():
128             stepping = int(stepping)
129             if stepping in steppingList:
130                 steppingCorrect = True
131
132     maxFlowRate = floor(maxFlowRates[stepping])
133
134     while not flowRateCorrect:
135         print("Flow rate between {} uL/min and {} uL/min (integer): ".format(0,
136               maxFlowRate), end='')
137         flowRate = input()
138         if flowRate.isnumeric():
139             flowRate = int(flowRate)
140             if flowRate >= 0 and flowRate <= maxFlowRate:
141                 flowRateCorrect = True
142
143     while not flowTimeCorrect:
144         print("Flow time between {} s and {} s (integer): ".format(
145             0, maxFlowTime), end='')
146         flowTime = input()
147         if flowTime.isnumeric():
148             flowTime = int(flowTime)
149             if flowTime >= 0 and flowTime <= maxFlowTime:
150                 flowTimeCorrect = True
151
152     return stepping, direction, flowRate, flowTime
153
154 def flush_input():
155     try:
156         import msvcrt
157         while msvcrt.kbhit():
158             msvcrt.getch()
159     except ImportError:
160         import sys
161         import termios  # for linux/unix
```

```
162         termios.tcflush(sys.stdin, termios.TCIOFLUSH)
163
164
165 def on_press(key):
166     global arduino, resetPressed, gettingSetup, endProgram, listener,
    serialMonitorActive
167     try:
168         pressedKey = key.char.lower()
169         if pressedKey == 'r' and not resetPressed:
170             resetPressed = True
171             arduino.controlWrite('r')
172         elif pressedKey == 'a':
173             arduino.controlWrite('a')
174         elif pressedKey == 'd':
175             arduino.controlWrite('d')
176         elif pressedKey == 's' and not gettingSetup:
177             gettingSetup = True
178             # Stop listener
179             return False
180
181     except AttributeError:
182         pass
183
184     if key == keyboard.Key.esc:
185         # Signal end of program
186         endProgram = True
187         # Stop listener
188         return False
189
190
191 def on_release(key):
192     global resetPressed, gettingSetup
193     try:
194         if key.char.lower() == 'r':
195             resetPressed = False
196         elif key.char.lower() == 's':
197             gettingSetup = False
198
199     except AttributeError:
200         pass
201
202
203 def serialMonitor():
204     global serialMonitorActive, arduino
205     with open('flow.txt', 'a') as file:
206         while serialMonitorActive:
207             line = arduino.serialConnection.readline().decode('ascii')
208             print(line, end='')
209             file.write(line)
210
211
212 if __name__ == "__main__":
213
214     controlProtocol = {'r': 114, 'a': 49, 'd': 48}
215     steppingList = [1, 2, 4, 8, 16]
216     baudrate = 115200
217     maxFlowTime = 172800
218     connectionSuccessful = False
```

```
    resetPressed = False
    gettingSetup = False
    serialMonitorActive = False
    endProgram = False

    maxFlowRates = computeMaxFlowRates()

    printLine(80)
    print(" "*33 + " Syringe Pump " + " "*33)
    printLine(80)

    arduino = SyringePumpArduino()
    connectionSuccessful = arduino.connectToPort(baudrate)
    if connectionSuccessful:
        endProgram = False

    else:
        endProgram = True

    print("")

    printControls()

    while not endProgram:

        # Starting the serial monitor
        serialMonitor_t = threading.Thread(target=serialMonitor)
        serialMonitorActive = True
        serialMonitor_t.start()

        # Read keyboard input dynamically
        with keyboard.Listener(on_press=on_press, on_release=on_release) as listener:
            listener.join()

        # If a setup input has been provided, deactivate the keyboard listener and
    the serial monitor until setup finishes
        if(gettingSetup):
            serialMonitorActive = False
            serialMonitor_t.join()
            flush_input()
            printLine(80)
            print("Starting setup configuration\n")
            stepping, direction, flowRate, flowTime = getSetupInput()
            print("")
            printLine(80)
            print(arduino.setupQuery(stepping, direction, flowRate, flowTime))
            gettingSetup = False

    if connectionSuccessful:
        serialMonitorActive = False
        serialMonitor_t.join()
        arduino.serialConnection.close()

    input("Press enter to exit")
```

Listing C.1: *Supervisor*