UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
BACHARELADO EM ENGENHARIA ELETRÔNICA

GUSTAVO CHICHANOSKI

DESENVOLVIMENTO DE INSTRUMENTAÇÃO ELETRÔNICA COM FPGA: VAZÃO E TEMPERATURA

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2018

GUSTAVO CHICHANOSKI

DESENVOLVIMENTO DE INSTRUMENTAÇÃO ELETRÔNICA COM FPGA: VAZÃO E TEMPERATURA

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de Trabalho de Conclusão de Curso, do curso de Graduação em Engenharia Eletrônica do Departamento Acadêmico de Eletrônica (DAELN) da Universidade Tecnológica Federal do Paraná (UTFPR), como requisito parcial para obtenção do título de Engenheiro Eletrônico.

Orientador: Dr. Márcio Rodrigues da Cunha

CAMPO MOURÃO
2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Campo Mourão
Coordenação do Curso de Engenharia Eletrônica

TERMO DE APROVAÇÃO

DO TRABALHO DE CONCLUSÃO DE CURSO INTITULADO

DESENVOLVIMENTO DE INSTRUMENTAÇÃO ELETRÔNICA COM FPGA: VAZÃO E TEMPERATURA

por

GUSTAVO CHICHANOSKI

Trabalho de Conclusão de Curso apresentado no dia 23 de Novembro de 2018 ao Curso Superior de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Campus Campo Mourão. O Candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____

Prof. Lucas Ricken Garcia
(UTFPR)

_____

Prof. Gilson Junior Schiavon
(UTFPR)

_____

Prof. Marcio Rodrigues da Cunha
(UTFPR)
Orientador

A Folha de Aprovação assinada encontra-se na Coordenação do Curso

RESUMO

CHICHANOSKI, GUSTAVO. DESENVOLVIMENTO DE INSTRUMENTAÇÃO ELETRÔNICA COM FPGA. Trabalho de Conclusão do Curso - Bacharelado em Engenharia Eletrônica, Universidade Tecnológica Federal do Paraná. Campo Mourão, 2018. O objetivo desse trabalho foi projetar um controlador de temperatura e vazão implementado em FPGA. Realizando as etapas de identificação, projeto e implementação de um controlador discreto com saída modulada em PWM, utilizando os sensores de vazão YF-S401 e como atuador as bombas d'água automotivas de 24V. Para a temperatura foi utilizado o controlador ON/OFF, o sensor de temperatura DS18B20 e de atuador, um aquecedor de 1000 W ligado a rede elétrica.

Palavras-chave: Controlador,Identificação de Sistemas, Vazão, Temperatura, FPGA

ABSTRACT

CHICHANOSKI, GUSTAVO. DEVELOPMENT OF ELECTRONIC INSTRUMENTATION WITH FPGA. Course Completion Work - Bachelor of Electronic Engineering, Federal Technological University of Paraná. Campo Mourão, 2016. The objective of this work was to design a temperature and flow controller implemented in FPGA. In this work an identification and distance control project is presented using a modulation through PWM modulation, using flow sensors YF-S401 and as a 24V water pump actuator. For the temperature was controlled by an ON / OFF controller, temperature sensor DS18B20 and a heater of 1000W connected to a network as **actuator**.

Key-words: FPGA, Discrete Controller, System Identification, Flow, Temperature

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, fonte de toda a sua ciência, que por sua benção, pude trilhar o caminho do conhecimento durante toda a graduação. Agradeço a minha família pelo suporte, ajuda e compreensão nos momentos mais complicados.

Agradeço aos meus professores, que me auxiliaram até esse momento, mas principalmente, agradeço ao meu orientador Dr. Márcio Rodrigues da Cunha, pela fé depositada em mim.

Agradeço a Universidade Tecnológica Federal do Paraná campus Campo Mourão, pela disponibilização de sua estrutura.

Por fim, agradeço aos meus colegas de faculdade, que me auxiliaram durante todo esse período.

Agradeço a todos que compartilharam esse caminho comigo, em especial a minha esposa Andressa Santos Leão por me fazer companhia nas noites frias de estudos, espero o melhor para todos, muito obrigado por tudo.

FIGURAS

Figura 1 – EVOLUÇÃO DOS PLDs. 6
Figura 2 – ESTRUTURA INTERNA PAL. 7
Figura 3 – Esquema simplificado do GAL. 7
Figura 4 – CPLD Altera família MAX 7000. 8
Figura 5 – Arquitetura básica de uma Fpga. 9
Figura 6 – Bloco lógico. 10
Figura 7 – Esquema de núcleo rígido em uma FPGA. 11
Figura 8 – Placa de desenvolvimento Cyclone IV 12
Figura 9 – Cálculo da vazão. 13
Figura 10 – Gráfico de processos mais medidos nas indústrias. 13
Figura 11 – Bomba d'água do lavador de para-brisa 0 392 003 501 12V. 14
Figura 12 – Curva de vazão por tensão da bomba d'água. 15
Figura 13 – Figura do motor para o projeto. 16
Figura 14 – Tipos de sensores de vazão existentes. 18
Figura 15 – Funcionamento do tipo turbina. 18
Figura 16 – Sensor de vazão YF-S401. 19
Figura 17 – Resistor para aquecimento do misturador. 20
Figura 18 – Sensor de temperatura DS18B20. 21
Figura 19 – Diagrama de blocos do controlador de vazão da bomba d'água. 23
Figura 20 – Diagrama de blocos do controlador de temperatura do aquecedor. 24
Figura 21 – Diagrama de blocos do controlador de temperatura do aquecedor. 24
Figura 22 – Diagrama de Processos e Instrumentação P&DI. 25
Figura 23 – Desenho esquemático de um sistema para realizar a produção química. 26
Figura 24 – Gráfico entre vazão e frequência. 27
Figura 25 – Comparação entre o modelo matemático e a bomba real. 30
Figura 26 – O lugar das raízes no sistema de vazão. 31
Figura 27 – Limites do lugar das raízes com os limites dos parâmetros. 32
Figura 28 – Root Locus com o integrador e os limites. 33
Figura 29 – Resposta ao degrau do sistema com compensador. 34
Figura 30 – Comparativo de PMW com 25, 50 e 75%. 35
Figura 31 – Relação entre tensão, porcentagem e PWM. 35
Figura 32 – Driver para o acionamento do MOSFET. 37
Figura 33 – Placa de acionamento da bomba. 39
Figura 34 – Placa de acionamento do aquecedor. 40
Figura 35 – Placa fabricada do esquemático. 41
Figura 36 – Protótipo fabricado para o trabalho. 42
Figura 37 – Organização dos blocos do VHDL. 43
Figura 38 – Representação do bloco Leitor. 43
Figura 39 – Representação do bloco PID. 44
Figura 40 – Representação do bloco PWM 45
Figura 41 – Representação do bloco controleTemp. 46
Figura 42 – Representação do bloco pwmTemp 48
Figura 43 – Resposta do sistema obtida pelo osciloscópio para uma referência de 10Hz. O sinal em amarelo é a saída PWM do FPGA e o sinal em verde é a saída do sensor de vazão. 49
Figura 44 – Resposta do sistema para uma referência de 10Hz. 50

Figura 45 – Resposta do sistema para uma referência de 5Hz obtida pelo osciloscópio, o sinal em amarelo representa a saída PWM do FPGA e o sinal em verde representa o sinal do sensor de vazão. 50
Figura 46 – Resposta do sistema para uma referência de 5Hz. 51
Figura 47 – Resposta do sistema para uma referência de 5Hz obtida pelo osciloscópio, o sinal em amarelo representa a saída PWM do FPGA e o sinal em verde representa o sinal do sensor de vazão. 51
Figura 48 – Resposta do sistema para uma referência de 5Hz. 52
Figura 49 – Saída PWM do FPGA lida no osciloscópio. 53
Figura 50 – Resposta do sistema para uma referência de 60 C. 53
Figura 51 – Resposta do sistema para uma referência de 55 C. 54
Figura 52 – Comparação entre o aquecedor real e o virtual. 55
Figura 53 – Design final dos blocos de FPGA. 55

LISTA DE TABELAS

Tabela 1 – Recursos da placa de desenvolvimento do Cyclone IV 12
Tabela 2 – Características da bomba 0 392 003 501. 14
Tabela 3 – Características do sensor de Vazão. 19
Tabela 4 – Características do sensor de vazão. 21
Tabela 5 – Uso do FPGA pelo bloco leitor. 44
Tabela 6 – Uso do FPGA. 45
Tabela 7 – Uso do FPGA pelo bloco PWM2. 46
Tabela 8 – Uso do FPGA. 47
Tabela 9 – Uso do FPGA pelo bloco pwmTemp. 48
Tabela 10 – Utilização da FPGA. 49
Tabela 11 – Pinos no FPGA. 56
Tabela 12 – Uso do FPGA. 56

LISTA DE SÍMBOLOS

CI	Circuito Integrado
FPGA	Field Programmable Gate Array ou Arranjos de Portas Programáveis em Campo
PID	Proporcional Integral Derivativo
VHDL	*VHSIC Hardware Description Language*
DARPA	Departamento de Defesa dos Estados Unidos
Q_M	Vazão mássica
Q_V	Vazão volumétrica
M	Massa
V	Volume
t	Intervalo de tempo
SI	Sistema Internacional
s	Segundos
gal	Galões = 3,785411784 litros
m^3	Metros cúbicos

SUMÁRIO

1 INTRODUÇÃO **1**
1.1 TEMA 1
1.1.1 Delimitação do tema 1
1.1.1.1 Identificação de sistemas 1
1.1.1.2 FPGA 2
1.1.1.3 Microeletrônica 3
1.2 OBJETIVOS 4
1.2.1 Objetivo geral 4
1.2.2 Objetivos específicos 4
1.3 PROBLEMAS E PREMISSAS 5
1.4 JUSTIFICATIVA 5
2 FUNDAMENTAÇÃO TEÓRICA **6**
2.1 DISPOSIVOS LÓGICOS PROGRAMÁVEIS 6
2.1.1 SPLD 6
2.1.2 CPLD 7
2.2 FPGA 8
2.2.1 Blocos lógicos Configuráveis 8
2.2.2 Técnicas de programação 9
2.2.3 Núcleo de FPGA 11
2.2.4 Família Cyclone IV 11
2.3 VAZÃO 12
2.3.1 Sensor vazão 17
2.3.2 Sensor do tipo turbina 18
2.4 TEMPERATURA 19
2.4.1 Atuador resistivo 20
2.4.2 Sensor de temperatura 20
2.5 VHDL 21
2.6 CONTROLADOR 22
2.6.1 Controlador ON/OFF 23
2.6.2 Protótipo 23
3 METODOLOGIA **26**
3.1 DESCRIÇÃO DO PROTÓTIPO 26
3.2 CÁLCULO DO SENSOR DE VAZÃO 26
3.2.1 Cálculo da frequência 27
3.3 MODELAGEM MATEMÁTICA DA BOMBA D'ÁGUA 28
3.4 CÁLCULO DO CONTROLADOR DE VAZÃO 30
3.4.1 Discretização do controlador 32
3.5 MODULAÇÃO POR LARGURA DE PULSO 33
3.6 PLACA DE ACIONAMENTO DA BOMBA 36
3.6.0.1 Placa do MOSFET 39
3.7 PLACA DE ACIONAMENTO DO AQUECEDOR 39
3.7.0.1 Placa projetada 41
4 RESULTADOS **42**
4.1 PROTÓTIPO 42
4.2 VAZÃO 42
4.2.1 Leitor 42

4.2.2 PID 44
4.2.3 PWM 45
4.3 TEMPERATURA 46
4.3.1 controleTemp 46
4.3.2 pwmTemp 47
4.4 RESULTADO 48
4.4.1 Vazão 48
4.4.2 Temperatura 49
4.5 RESULTADOS DO FPGA 55
5 CONCLUSÃO 57

1 INTRODUÇÃO

1.1 TEMA

Desenvolvimento de um sistema de controle implementado em FPGA, para controlar parâmetros de um processo químico, vazão e temperatura.

1.1.1 Delimitação do tema

Um processo químico é uma operação ou um conjunto de operações coordenadas que causam uma transformação química ou físico-químico em determinado material ou mistura de materiais. O objetivo dos processos químicos é a obtenção de produtos desejados à partir de matérias primas selecionadas ou disponíveis.

A velocidade desses processos pode ser controlada variando alguns parâmetros de processos tais como, temperatura e concentração dos reagentes. As reações químicas podem ser classificadas em endotérmicas e exotérmicas. As reações endotérmicas absorvem calor do meio enquanto as reações exotérmicas liberam energia para o meio.

Este trabalho visa controlar os parâmetros de um processo químico, o fluxo dos reagentes e a temperatura da reação. Para tanto, foram utilizados componentes que possam alterar tais variáveis, os atuadores, uma bomba d'água, fornecer vazão de reagentes, e um aquecedor para gerar calor a reação. Esses atuadores serão controlados por um controlador implementado em FPGA, Field Programmable Gate Array ou Arranjos de Portas Programáveis em Campo, que terá como entrada a comparação da saída dos sensores de fluxo dos reagentes e a temperatura, com a referência do projeto.

1.1.1.1 Identificação de sistemas

Um processo tanto químico, físico ou elétrico, podem ser modelados matematicamente para descrever as principais características físicas. Os modelos podem ser retirados de simples sistemas lineares a máquinas complexas, os modelos são utilizados para análise, otimização, simulação e controle.

Nesses processos são conhecido a entrada e saída, sendo necessário explorar o funcionamento do processo para realizar o controle. A dificuldade do modelo varia conforme o estudo do processo, sendo classificado em três tipos.

- **Caixa Branca**: quando se entende os fenômenos que influenciam o processo, possi-

bilitando cálculo do modelo matemático que simule o funcionamento do processo, como a carga e descarga de um capacitor.

- **Caixa Cinza**: quando apesar dos fenômenos envolta do processo não serem totalmente conhecidos é possível realizar um modelo matemático do processo.
- **Caixa Preta**: quando se desconhece o funcionamento do processo estudado, conhecendo apenas a sua saída e entrada.

Para resolver a dificuldade de cálculo do modelo em caixas cinzas e pretas, pode-se realizar uma aproximação utilizando os métodos de identificação de sistemas. Através das entradas e saídas da planta, o algoritmo de identificação determina os parâmetros da função de transferência, baseado no critério de minimização da função custo.

1.1.1.2 FPGA

A grande maioria dos Circuito Integrados (CI), vendidos no mercado são destinados a aplicações específicas, denominados de *Application Specific Integrated Circuit* (ASIC), onde a programação é realizada no ato da fabricação ou posteriormente. Em relação aos ASICs programados no ato de fabricação existe um elevado custo de produção, exigindo que se produza em massa cada novo circuito, para que o produto seja economicamente viável. Entretanto caso haja um erro de programação na hora da produção, não há alternativas para a correção do erro.

Devido as desvantagens aos ASICs programados no ato da fabricação, em 1983 o cofundador da empresa Xiling Inc., Ross Freeman, criou um novo circuito integrado que poderia ser programado posteriormente, o FPGA, que são chips de silício reprogramáveis.

Segundo (COSTA, 2011), os FPGAs fornecem velocidade temporizada por *hardware* de alta confiabilidade, assim não necessitando de produção em altos volumes para justificar a despesa de um projeto ASICs customizado. Já em comparação com processadores de uso geral, como a família 8051, AVR e outros, que também podem ser programados posteriormente, os FPGAs possuem a mesma flexibilidade de *software*, porem são verdadeiramente paralelos por natureza, logo diferentes operações não tem que competir pelo mesmo recurso. Onde cada tarefa de processamento é enviada para uma seção dedicada do chip e pode funcionar de modo autônomo sem nenhuma influência de outros blocos lógicos.

Entretanto, o FPGA enfrenta dificuldades no processamento e conectividade de E/S no seu sistema, pois os FPGAs não têm ecossistema de *drivers* e a base de IP/códigos que as arquiteturas de microprocessadores e sistemas operacionais possuem, e ainda, microprocessadores com sistemas operacionais fornecem a fundação para estruturas de

arquivo e comunicação com periféricos usados por muitas tarefas, geralmente essenciais, como registro de dados no disco.

Para configurar o FPGA, é necessário definir tarefas de computação digital em *software* utilizando ferramentas de desenvolvimento, e então gerando um arquivo de configuração ou *bitstream* que contém informações sobre como os componentes devem ser conectados. Assim surgiu a linguagem usada para facilitar a programação de FPGA, o VHDL, uma linguagem com paralelismo nativo. Desenvolvida pelo DARPA, Departamento de Defesa dos Estados Unidos, em meados de 1980, para documentar o comportamento de ASICs que compunham os equipamentos vendidos às Forças Armadas Americanas.

1.1.1.3 Microeletrônica

Segundo (SWART, 2016), a microeletrônica surgiu em 1947, na Bell Labs, com o descobrimento do efeito transistor e o desenvolvimento do processo planar para fabricação de CI's em 1959 na Fairchild, resultando nos primeiros CI's comerciais em 1962.

O primeiro circuito integrado digital com tecnologia TTL(Transistor Transistor Logic) surgiu nos anos 60. Posteriormente, surgiram os circuitos integrados CMOS (Complementary MOS Logic). No começo os circuitos integrados incorporavam um pequeno número de portas lógicas, até o surgimento dos microprocessadores em 1970, (SWART, 2016).

A introdução dos microprocessadores e o aumento da capacidade da indústria de componentes eletrônicos de criar circuitos de memória formaram a base para rápida expansão da indústria de computadores e sistemas digitais complexos.

Segundo (COSTA, 2011), um circuito digital pode ser caracterizado em três tipos, cada qual projetado e produzido dentro de um circuito integrado, com os seguintes objetivos:

- **Lógica combinacional:** circuito de lógica digital cuja saída é uma expressão booleana em função de suas entradas. A saída do circuito responde imediatamente para qualquer mudança nas suas entradas.
- **Lógica Sequencial**: circuito lógico cuja resposta é baseada no estado atual e, em algumas vezes, no estado de suas entradas. A lógica pode ser síncrona ou assíncrona. Na lógica sequencial síncrona, as mudanças dos estados lógicos ocorrem sempre que um sinal de *clock* comum é aplicado a todos os setores de um circuito. Na lógica sequencial assíncrona, os estados lógicos mudam em função de sinais independentes para cada setor de um circuito.

- **Memória:** circuito lógico em que um valor digital pode ser armazenado(escrito) e retirado(lido) posteriormente. Para o usuário a memória pode ser somente de leitura ROM(Read Only Memory) ou de acesso aleatório RAM(Random Access Memory). Na ROM os dados armazenados são inicialmente gravados na memória e podem ser lidos pelo usuário. Os dados não podem ser alterados pelo circuito de aplicação. Na RAM o usuário pode escrever os dados na memória e depois retirá-los quando o circuito de aplicação necessitar deles.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Neste contexto o objetivo deste trabalho decorre da convergência tecnológica das três áreas contextualizadas, instrumentação, FPGA e controle. O escopo do trabalho é projetar controladores de vazão e temperatura implementado-os em FPGA.

1.2.2 Objetivos específicos

O desenvolvimento do projeto contempla alguns objetivos específicos, onde cada um deles representa uma parcela para alcançar o resultado desejado citado no objetivo geral. Os objetivos específicos são:

- Pesquisar artigos e trabalhos relacionados ao tema proposto.
- Realizar um estudo sobre o funcionamento e implementação dos atuadores.
- Realizar um estudo sobre o funcionamento e implementação dos sensores.
- Identificar a planta da vazão.
- Projetar um controlador para os atuadores utilizados.
- Desenvolver o software de controle implementado no FPGA.
- Confeccionar o protótipo do *hardware*.

1.3 PROBLEMAS E PREMISSAS

Para fazer o controle de vazão e temperatura é necessário além do correto dimensionamento da planta, encontrar sensores que satisfaçam o projeto do controlador.

Utilizando sensores pré-estabelecidos no mercado, podemos determinar a planta com precisão e realizar o correto controle.

1.4 JUSTIFICATIVA

A indústria química abrange áreas como a petroquímica, insumos agrícolas, medicamentos, tintas, entre outras. Essas indústrias baseiam sua velocidade de produção em função da velocidade das reações envolvidas no processo, ou seja, quanto maior a velocidade de reação, maior a velocidade de produção.

Portanto, a justificativa deste trabalho consiste em contribuir com o projeto de plantas inteligentes e integradas com todas as operações unitárias, atuadores, sensores e circuitos eletrônicos.

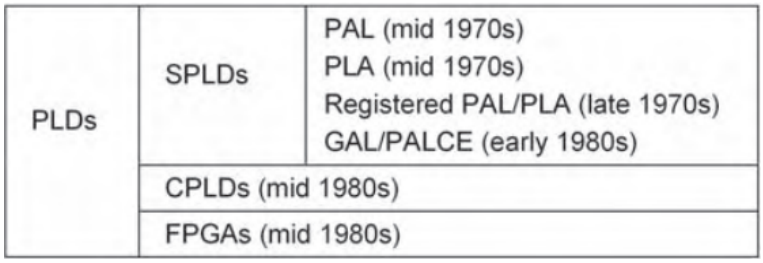
2 FUNDAMENTAÇÃO TEÓRICA

2.1 DISPOSIVOS LÓGICOS PROGRAMÁVEIS

Os dispositivos lógicos programáveis, conhecidos como *Programable Logic devices* (PLD), nasceram da necessidade de *hardwares* que pudessem ser programados para atender uma determinada especificação. Os primeiros PLDs podiam implementar apenas circuitos combinacionais simples, porém, graças à evolução da microeletrônica, estes dispositivos tornaram-se mais eficazes, abrangendo aplicações complexas, como por exemplo na área de telecomunicações.

O quadro abaixo ilustra a sequência da evolução dos PLDs.

Figura 1 – EVOLUÇÃO DOS PLDs.



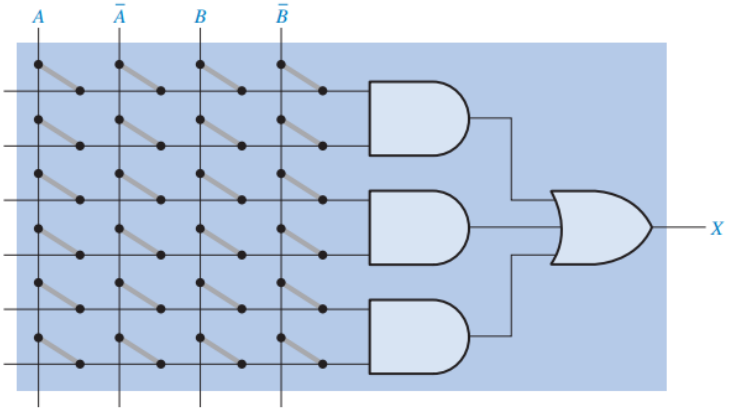
Fonte:(PEDRONI, 2016).

2.1.1 SPLD

São chamados de SPLD(*Simple Program Logic Devices*) os primeiros PLDs, cuja estrutura interna baseia-se em arranjos lógicos de portas *And/Or*. Os principais integrantes desta família são:

- PAL: Estes dispositivos são compostos por um arranjo programável de portas AND seguidas de um arranjo fixo de portas OR, podendo implementar expressões lógicas de soma-de-produtos com um determinado número de variável. O arranjo programável desta estrutura consiste em uma matriz de linha es colunas condutoras com fusíveis nas intersecções. A figura a seguir mostra a organização básica de um dispositivo PAL.
- PLA: Nestes dispositivos os arranjos de *Or* também são programáveis, permitindo desta forma, um maior número de combinações lógicas para uma mesma quantia de *hardware*. Entretanto, com o aumento das conexões programáveis aumentou a propagação de atrasos, diminuindo a velocidade do sistema.

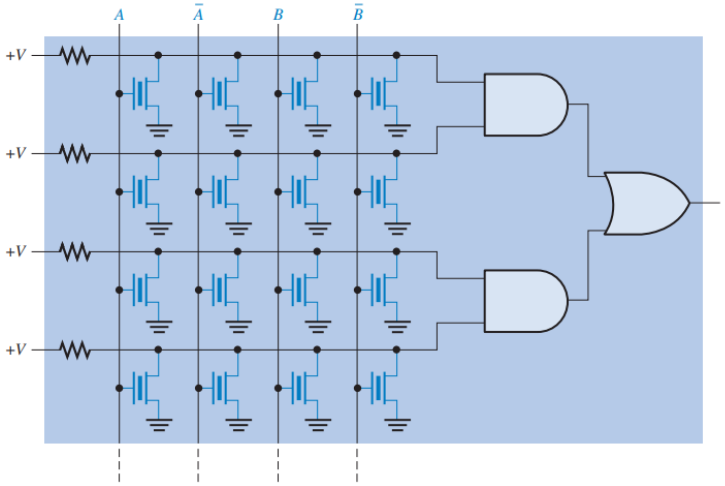
Figura 2 – ESTRUTURA INTERNA PAL.



Fonte: (FLOYD, 2007).

- GAL: Sua principal diferença para os PLDs anteriores, é a possibilidade de reprograma-lo. Desta forma, o dispositivo GAL possui essencialmente os mesmos arranjos AND/OR que os PAL, mas, com tecnologia de processo reprogramável como EEPROM em vez de fusíveis. A grande vantagem da GAL é a sua capacidade de também programar circuitos sequenciais.

Figura 3 – Esquema simplificado do GAL.



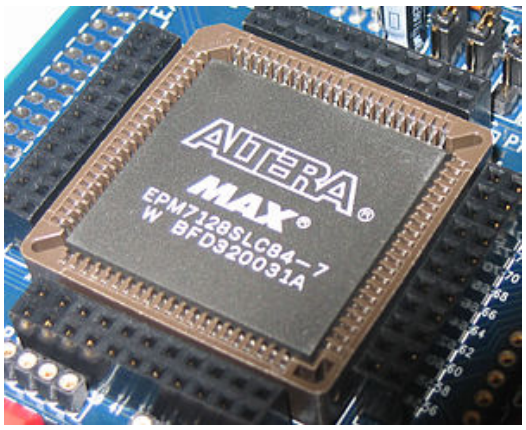
Fonte: (FLOYD, 2007).

2.1.2 CPLD

A CPLD(Complex PLD) é um dispositivo construído com vários SPLDs interligados no mesmo chip, onde estas ligações podem ser programadas. As CPLDs possuem muitas características adicionais em relação aos SPLDs, tais como maior número de E/S e suporte para vários padrões lógicos. Os dispositivos da família MAX fabricados pela

Atera são um exemplo de CPLDs comercializadas atualmente.

Figura 4 – CPLD Altera família MAX 7000.



Fonte: (WIKIPéDIA, 2013).

2.2 FPGA

No ano de 1985 a empresa Xilinx Inc. lançou a primeira FPGA, abreviação para Field Programable Gate Array. Este dispositivo consiste em um conjunto de blocos lógicos alocados em forma de matriz, (FLOYD, 2007).

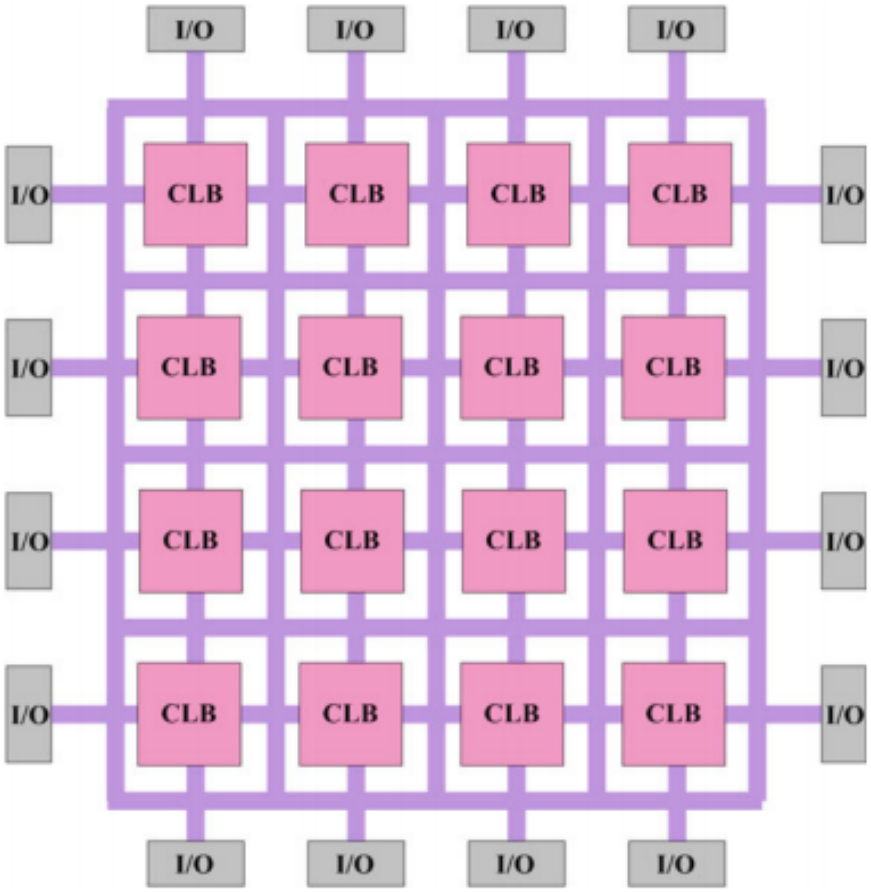
A configuração interna de uma FPGA é diferente para cada fabricante, e também pode variar em cada família de dispositivos, no entanto, três elementos estão presentes em todos os modelos: Bloco Lógico Configurável (CLB), interconexões e blocos de entrada e saída (I/O). A Figura 5 mostra a arquitetura básica de uma FPGA.

2.2.1 Blocos lógicos Configuráveis

Os blocos lógicos programáveis são os componentes básicos de uma FPGA, formando a unidade mínima de lógica e armazenamento para uma determinada aplicação. Os CLBs podem desempenhar desde funções lógicas simples, como a de um transistor, até as funções lógicas de um processador.

Quando os CLBs são relativamente simples, a arquitetura do FPGA é denominada granulação fina. Já quando os CLBs são complexos, é dito que o FPGA possui granulação grossa. Geralmente opta-se por uma granulação intermediária, pois, uma granulação muito fina implica em um grande amontoado de interconexões programáveis resultando em perda de área útil, atrasos e alto consumo de energia. Por outro lado, a granulação grossa acarretaria no desperdício de recursos ao implementar funções menores.

Figura 5 – Arquitetura básica de uma Fpga.



Fonte: (FAROOQ; MARRAKCHI; MEHREZ, 2012).

A maior parte dos fabricantes de FPGA utilizam CLBs com vários módulos lógicos menores e uma interconexão local que os interliga, as vezes são baseados em LUT.

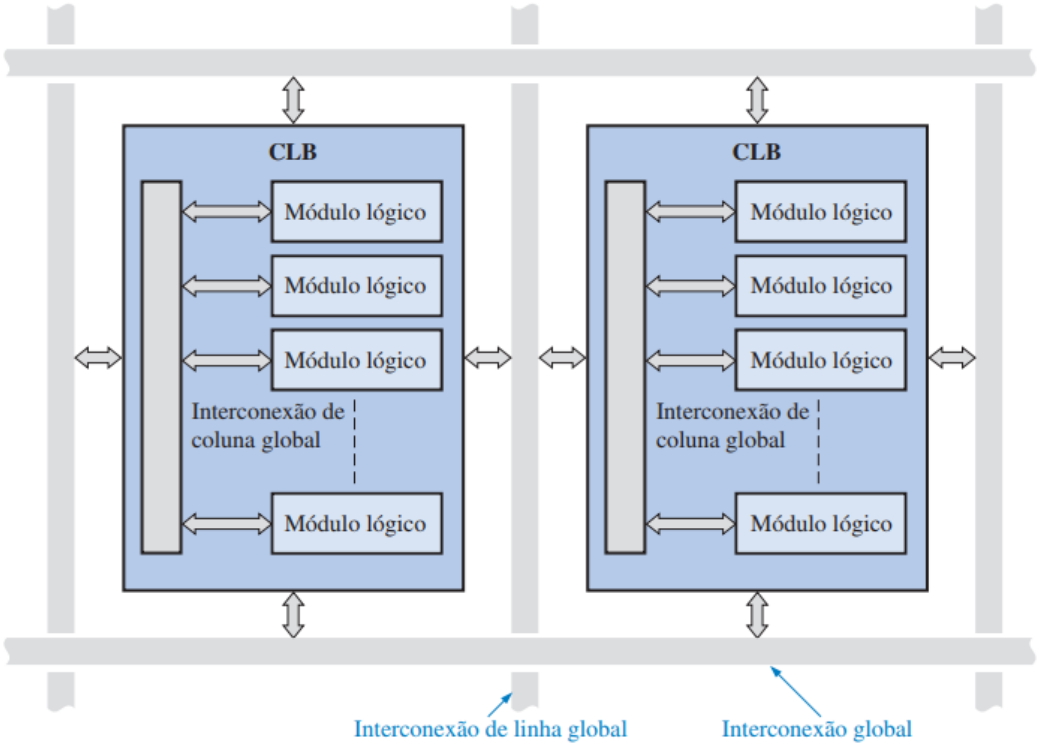
As LUTs são memórias voláteis programáveis capazes de implementar lógicas combinacionais de soma-de-produtos, comportando-se de forma semelhante à um dispositivo PLA ou PAL.

2.2.2 Técnicas de programação

As arquitetura reprogramadas podem ser obtidas utilizando várias técnicas, os quais destacam-se:

- Antifuse: Na tecnologia antifuse emprega-se dispositivos que possuem alta impedância em seu estado não programado, entretanto, ao aplicarmos uma tensão sobre ele, sua impedância diminui. A maior vantagem desta técnica é a baixa área ocupada. Apesar disso, esta tecnologia é essencialmente não-volátil, utilizando dispositivos

Figura 6 – Bloco lógico.



Fonte:(FLOYD, 2007)

que não permitem a reprogramação.

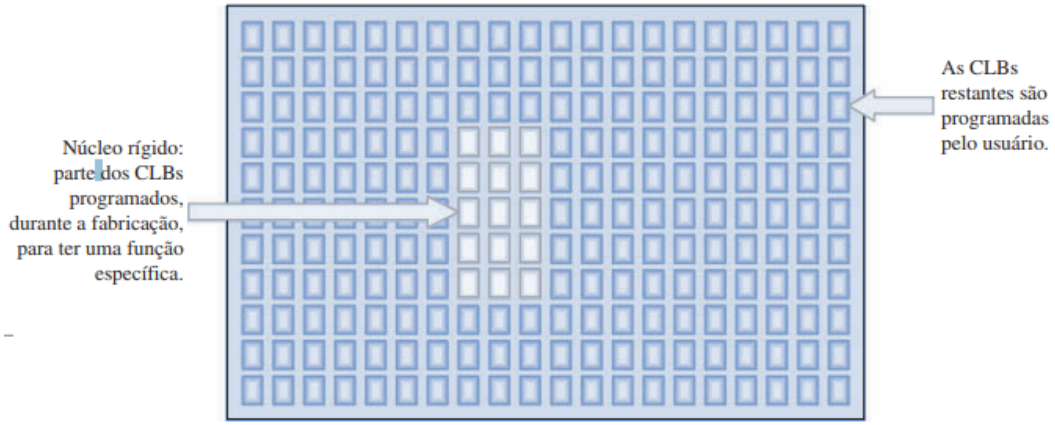
- Gate flutuante: Esta tecnologia é semelhante à usada na fabricação de EEPROMs, baseada em transistores MOS construídos com dois gates flutuantes. A desvantagem de tecnologias FLASH é que apensar de serem não voláteis, estas memórias não podem ser infinitamente reprogramadas.

- SRAM: Através da memória estática de acesso randômico controla-se um transistor de passagem ou multiplexador. Desta forma, as SRAMs são utilizadas para programar as interconexões dentro de uma FPGA. A grande maioria dos fabricantes utilizam esta tecnologia por causa de sua capacidade de reprogramação e o uso do padrão CMOS, levando assim à uma maior velocidade e à um menor consumo de energia. Por serem memórias voláteis, as FPGAs baseadas em SRAM necessitam de uma memória não volátil para sua inicialização. Esta memória pode ser incrustada no mesmo chip ou pode ser externa, com a transferência de dados controlada por um processador hospedeiro.

2.2.3 Núcleo de FPGA

As FPGAs comercializadas possuem núcleo rígido de programação fixa em meio à dispositivos totalmente programáveis. Este núcleo contém especificações do fabricante, interfaces padrão de entrada/saída, além de microprocessador e processadores de sinais. A vantagem das funções pré-programadas é a rapidez no desenvolvimento de projetos, pois como as funções são testadas, o usuário pode economizar o tempo que demoraria para desenvolve-las. A Figura 7 mostra o esquema de núcleo em uma FPGA.

Figura 7 – Esquema de núcleo rígido em uma FPGA.



Fonte: (FLOYD, 2007).

2.2.4 Família Cyclone IV

Na família Cyclone IV da Altera, as menores unidades lógicas são chamadas de Logics Elements(LE). Estas unidades lógicas são compostas por LUTs de 4 entradas e um registrador programável (flip-flop). OS LE são baseados em SRAM, necessitando de um dispositivo memória externa por serem voláteis.

Os blocos de arranjo lógico (LABs) são arranjos com dezesseis LE, conectadas com alta integração de projeto, aumentando o desempenho. Além disso cada LAB possui uma lógica dedicada para tratar cada sinal das LEs.

Este trabalho irá utilizar o processador Cyclone IV da família Cyclone da Altera como FPGA. Nele será executado o programa principal, que entre outras tarefas, realizará o controle da vazão e temperatura do sistema. A versão do Cyclone IV utilizada foi a EP4CE6, Figura 8 com as características mostrada na Tabela 1. A escolha deste dispositivo para o projeto foi motivada principalmente por seu baixo custo.

Figura 8 – Placa de desenvolvimento Cyclone IV

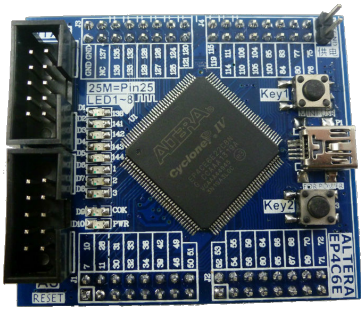


Tabela 1 – Recursos da placa de desenvolvimento do Cyclone IV

Recursos	EP4CE6
Elementos Lógicos	6.272
Memoria Embarcada (Kbits)	270
Multiplicadores Embarcados 18x18	15
PLLs de uso geral	2
Rede de clock global	10
Banco de I/O do usuário	8
I/O máximo usuário	179

Fonte: Autoria Própria.

2.3 VAZÃO

Segundo (BRUNETTI, 2008), a vazão (Q) pode ser definida como o volume ou massa do fluido que atravessa uma certa seção do escoamento por unidade de tempo. A vazão para as massas pode ser expressa pela equação 2.1, enquanto a vazão volumétrica pode ser definida pela equação 2.2.

$$Q_M = \frac{M}{t} \qquad (2.1) \qquad\qquad Q_V = \frac{V}{t} = \frac{s \cdot A}{t} \qquad (2.2)$$

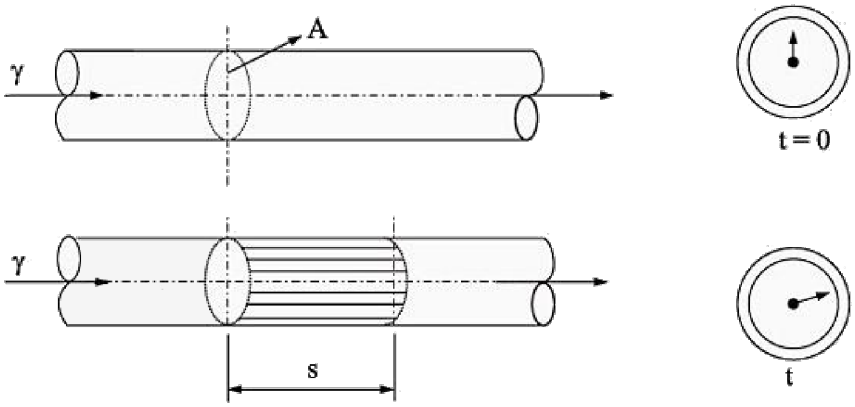
A Figura 9 representa como é obtida a expressão matemática da vazão. No instante t igual a 0s, existe um escoamento no sentido indicado por γ através da área **A**. Após um período de tempo **t**, o líquido percorre um comprimento **s**. Ao multiplicar-se a área **A** pelo comprimento **s** e posteriormente dividirmos pelo tempo **t**, obteremos a vazão volumétrica do sistema.

O conhecimento da vazão é fundamental nos processos indústria, sendo a terceira grandeza mais mensurada nas indústrias, como mostra a figura 10. A vazão pode ser geradas por máquinas, pela diferença de altura ou pressão.

Pode-se chamar de máquinas, qualquer dispositivo introduzido no escoamento, os quais forneçam ou retirem energia dele na forma de trabalho. Segundo (BRUNETTI, 2008), é possível dividir essas máquinas em dois grupos:

- **Bombas**: qualquer máquina que forneça energia ao fluido, $H_M > 0$.

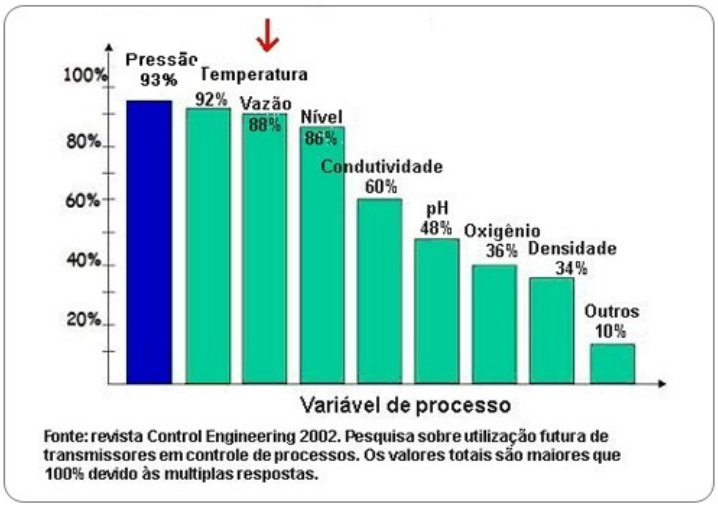
Figura 9 – Cálculo da vazão.



Fonte: (BRUNETTI, 2008).

- **Turbinas**: qualquer máquina que retire energia do fluido, $H_M < 0$.

Figura 10 – Gráfico de processos mais medidos nas indústrias.



Fonte: (CASSIOLATO; ORELLANA, 2010).

Neste trabalho será feito o controle de duas bombas d'águas, representados na Figura 11. Nessas bombas, a água entra no centro de um rotor de plástico em alta rotação, impulsionado por motor elétrico de corrente contínua, movimentando o líquido e gerando uma força centrífuga que se transforma em pressão. A entrada de água é denominada de sucção e possui o diâmetro externo de 8mm e interno de 6mm, no qual a saída é localiza na lateral, possuindo o mesmo diâmetro da entrada. A tabela 2 mostra as características da bomba d'água que será utilizada no trabalho.

Foi realizado um ensaio na bomba d'água, para levantar sua curva de resposta para a variação de tensão, mostrado na Figura 12.

O escoamento pode ser avaliado por meio da equação de Reynolds definida por 2.3, através do qual podemos classificar o regime de escoamento.

Figura 11 – Bomba d'água do lavador de para-brisa 0 392 003 501 12V.

Fonte:(BOSCH, 2013).

Tabela 2 – Características da bomba 0 392 003 501.

Características	Valor	Unidade
Tensão nominal	12	**V**
Fluxo impulsionado	0.5	**L/H**
Pressão de impulsão	1.5	**bar**
Sentido de giro	L(esquerdo)	
Tipo de serviço	S2 1,5	**min**
Grau de proteção	P 54	
Peso	0.09	**g**

Fonte:(BOSCH, 2013).

$$Re = \frac{\rho \cdot v \cdot D}{\mu} \tag{2.3}$$

Os parâmetros definido na equação 2.3 são:

v - Velocidade média do fluido $[\frac{m}{s}]$.

D - Diâmetro para o fluxo no tubo $8 \cdot 10^{-3}$ $[m]$.

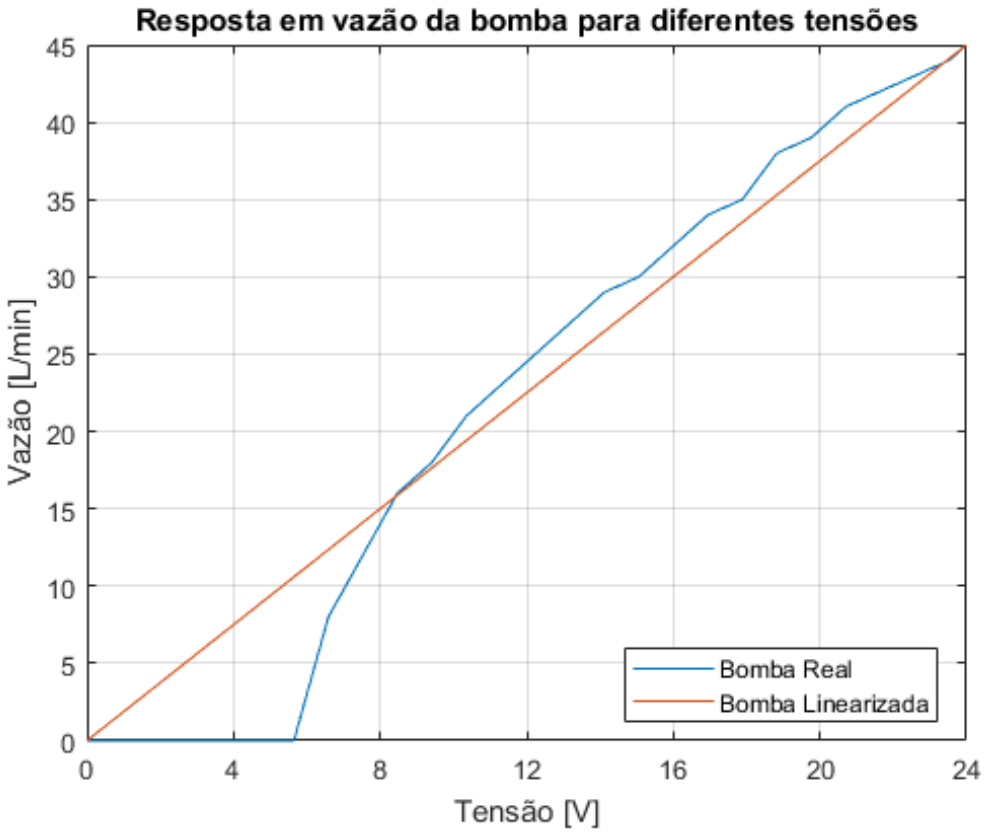
μ - Viscosidade dinâmica do fluido, para água é $1,0030 \cdot 10^{-3}$ $[\frac{m^2}{s}]$.

ρ - Massa específica do fluido, para água o valor é de 1000 $[\frac{kg}{m^3}]$.

Através do resultado da equação de Reynolds, avaliar o regime de escoamento, em laminar, Re $<$ 2000, transitório, 2000 $<$ Re $<$ 2400, e turbulento, Re $>$ 2400. Sendo as características:

- **Laminar**: Regime onde o fluido move-se ao longo de trajetórias bem definidas, apresentando lâminas ou camadas, cada uma delas preservando sua característica no meio. Onde o fluido age no sentido de amortecer a tendência de surgimento da turbulência. O escoamento laminar é o menos comum na prática, podendo ocorre a baixas velocidades ou em fluídos que apresentam grande viscosidade.
- **Transitório**: Fluido se encontra entre as regiões
- **Turbulento**: Regime onde o fluído descreve trajetórias irregulares, produzindo uma transferência de quantidade de movimentos entre regiões de massa líquida. Ocorre a altas velocidades ou em fluídos de baixa viscosidade.

Figura 12 – Curva de vazão por tensão da bomba d'água.



Fonte: Autoria própria.

A velocidade do fluido, considerando a características da água, pode ser definido isolando o valor de v, definindo pela equação 2.4.

$$v = \frac{Re \cdot \mu}{\rho \cdot D} = \frac{1}{80}\frac{m}{s} \tag{2.4}$$

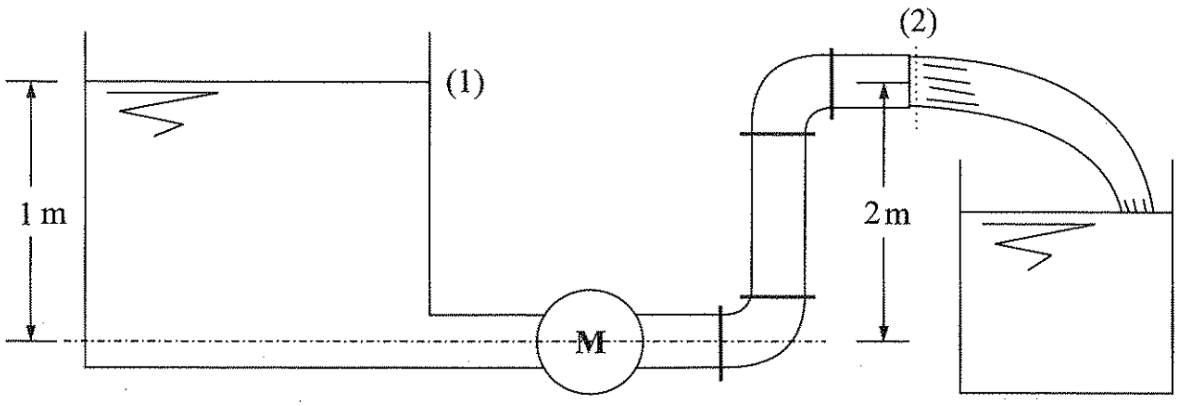
Segundo o (CUNHA, 2012), um valor adequado para o número de Reynolds é de 100, valor que permite uma melhor mistura do líquido. Substituindo os valores de Re, μ, ρ e D, tem-se que a velocidade do fluido será de $0.0125\frac{m}{s}$. Assim determinou uma vazão volumétrica de referência pela equação 2.5, substituindo os de v e D obtêm-se o valor de Q_V.

$$Q_V = \frac{v \cdot \pi \cdot D^2}{4} = 6.3020 \cdot 10^{-7}\frac{m^3}{s} = 0.6302\frac{mL}{s} \tag{2.5}$$

Conforme (BRUNETTI, 2008) explica, como o fluido é considerado ideal, pode-se aplicar a equação de Bernoulli entre as seções (1) e (2), mostrado na Figura 13, lembrando que entre as duas existe a máquina M. Mesmo que o reservatório da esquerda não tenha o nível constante, será adotada a hipótese de regime permanente com a seguinte consideração: o reservatório, sendo de grandes dimensões, levará muito tempo para que seu nível seja alterado sensivelmente pela água descarregada por (2).

Logo dentro de um certo intervalo de tempo, pode-se considerar que o seu nível é constante, mantendo dessa forma a hipótese de regime permanente. Com essas considerações para um primeiro modelo, pode-se definir a equação de Bernoulli mostrado a seguir.

Figura 13 – Figura do motor para o projeto.



Fonte: (BRUNETTI, 2008).

$$H_1 + H_M = H_2 \tag{2.6}$$

$$H_1 = \frac{p_1}{\gamma} + \frac{{v_1}^2}{2g} + z_1 \tag{2.7}$$

$$H_2 = \frac{p_2}{\gamma} + \frac{{v_2}^2}{2g} + z_2 \tag{2.8}$$

- H_1 - Energia total por unidade de peso na seção 1 $[m]$.
- z_1 - Carga potencial $[m]$.
- g - Aceleração da gravidade $\left[\frac{m}{s^2}\right]$.
- v_1 - Velocidade do fluido na seção 1 $\left[\frac{m}{s}\right]$.
- p_1 - Pressão do fluido no reservatório 1 $[Pa]$.
- γ - Peso especifico do fluido $\left[\frac{kg}{m^3}\right]$.
- H_2 - Energia total por unidade de peso na seção 2 $[m]$.
- v_2 - Velocidade do fluido na seção 2 $\left[\frac{m}{s}\right]$.
- p_2 - Pressão do fluido no reservatório 2 $[Pa]$.
- H_M - Energia total por unidade de peso do motor $[m]$.

$$z_1 = 1m \qquad (2.9) \qquad\qquad z_2 = 2m \qquad (2.10)$$

Adotando o PHR na base do reservatório (1), tem-se:

A pressão, tanto na seção (1) como na seção (2), é igual a pressão atmosférica, logo $p_1 = 0$ e $p_2 = 0$ na escala efetiva.

$$H_1 = 0 + 0 + 1 = 1m \qquad (2.11)$$

$$H_2 = 0 + \frac{(0.0125)^2}{2 \cdot 9.8} + 2 \cong 2m \qquad (2.12)$$

$$H_M = H_2 - H_1 = 1m \qquad (2.13)$$

Como no sentido do escoamento H_M é positivo, conclui-se que a máquina necessária para obter a vazão desejada é uma bomba. Então $H_B = 1m$, nesse caso a potência necessária para a bomba será de:

$$N = \gamma \cdot Q \cdot H_B = 10^4 \cdot 6.3020 \cdot 10^{-7} \cdot 1 = 6.302W \qquad (2.14)$$

- N - Potência da bomba $[W]$.

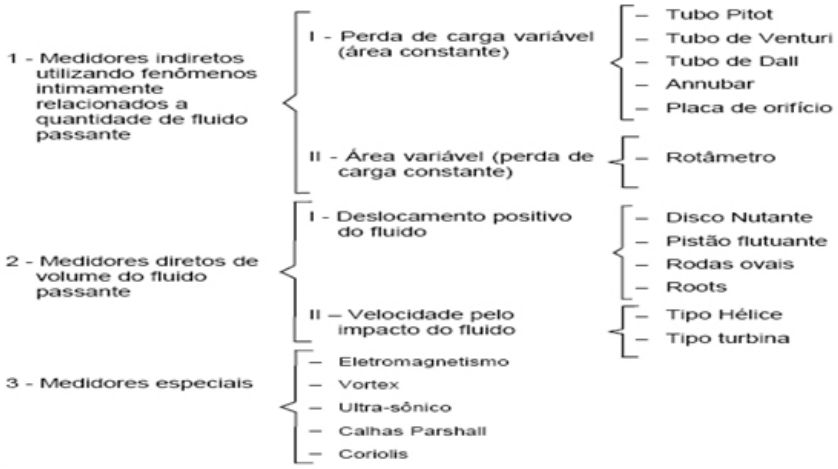
Pode-se notar a capacidade da bomba, Figura 11, de produzir a vazão e potência necessária para o projeto.

2.3.1 Sensor vazão

Exitem diversos tipos de sensores, Figura 14, tais como os sensores o tubo de Pilot, tubo de Venturi, tubo de Dali, annubar, placa de orifício, rotâmetros, disco nutante, pistão flutuante, rodas ovais, roots, tipo hélice, tipo turbina, eletromagnéticos, ópticos, vortex, ultrassônicos, calhas Parshall, coriolis e ópticos.

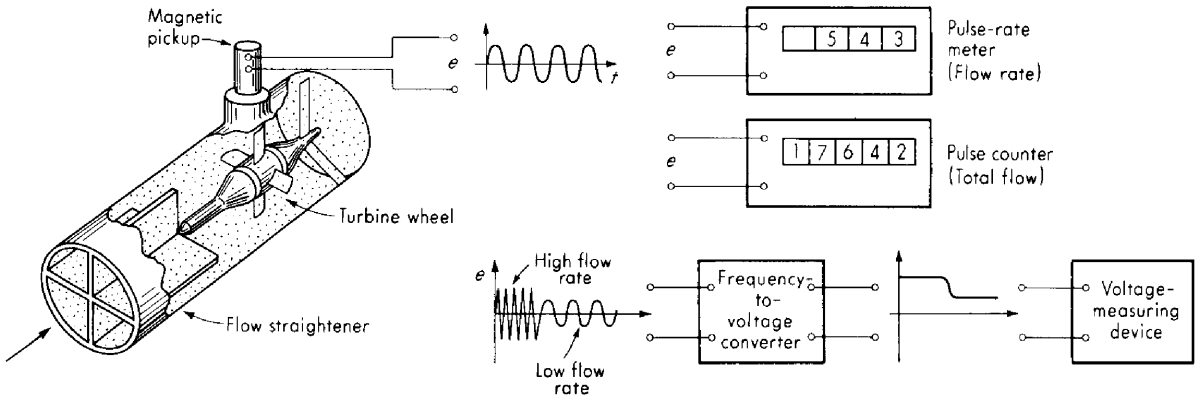
Em particular será estudado o sensor de tipo turbina. Segundo o autor (DOEBELIN, 1990), a velocidade de rotação da turbina depende da vazão do fluido. Considerando-se o atrito do rolamento e outras perdas desprezíveis, pode-se conceber uma turbina cuja a velocidade varia linearmente com o fluxo do líquido. Esse tipo de sensor é simples e permite uma boa acurácia contando o número de vezes que as laminas das hélice passam pelo ponto que contem o sensor de proximidade magnético, que produz pulsos de tensão. Se usar um circuito eletrônico para contar e acumular o número total de pulsos durante um determinado intervalo de tempo obtemos a vazão. Por causa dessa natureza digital permite fabricar medidores de alta acurácia. Na Figura 15, mostra funcionamento do sensor de vazão do tipo turbina.

Figura 14 – Tipos de sensores de vazão existentes.



Fonte: (CASSIOLATO; ORELLANA, 2010).

Figura 15 – Funcionamento do tipo turbina.



Fonte:(DOEBELIN, 1990).

2.3.2 Sensor do tipo turbina

O sensor de vazão utilizado neste trabalho é o YF-S401, Figura 16, de 7mm no tubo de entrada e saída, Tabela 3.

O YF-S401 é uma estrutura de engrenagem em formato de catavento com um imã acoplado no rotor que trabalha em conjunto com um sensor hall, emitindo um sinal pulsado com frequência proporcional a vazão. Através destes pulsos é possível mensurar a vazão do líquido, por sua frequência. Utilizando a equação 2.15, retirado de (LEONG *et al.*, 2016), obtêm-se a vazão desejada.

$$Q = \frac{98}{F} \tag{2.15}$$

- **Q** - Vazão volumétrica do fluido. $\left[\frac{L}{min}\right]$
- **F** - Frequência do sinal de saída. [Hz]

Figura 16 – Sensor de vazão YF-S401.

Fonte: (TRONIC, 2016).

Tabela 3 – Características do sensor de Vazão.

Características	**Valor**	**Unidade**
Pressão máxima	0.8	**MPa**
Faixa de trabalho	0.3 a 30	**L/min**
Tensão de operação	5 a 24	**V**
Corrente máxima	10	**mA**
Faixa de Umidade	35 a 90	**% RH**
Exatidão	2	**%**
Pulsos por litro	5880	**Pulsos**

Fonte: (TRONIC, 2016).

2.4 TEMPERATURA

O controle de temperatura será realizado através de um FPGA, e deverá ficar em torno de 50 a 80 °C, para o trabalho fixou-se a referência para 65 °C. O calor será produzido através de um atuador resistivo, que aquecerá a mistura no misturador.

$$C = m \cdot c \cdot (T - T_O) \tag{2.16}$$

- $C \rightarrow$ de calor sensível [J].
- c $\rightarrow$ *calor* específico da substância que constitui o corpo [*cal*].
- $m \rightarrow$ Vazão mássica do sistema $\left[\frac{kg}{s}\right]$.
- T $\rightarrow$ Temperatura de referência [°C].
- $T_O \rightarrow$ Temperatura do líquido no misturador [°C].

Para calcular o resistor necessário para aquecer a água do misturador, considera o pior caso, onde a água está na temperatura ambiente de 25 °C e completando o reservatório de 2 L, como para água 1 L é equivalente a 1 kg, 2 L são 2 kg d'água para aquecer. Sendo assim determina-se que:

$$C = 2 \cdot 1 \cdot (65 - 25) = 2 \cdot 40 = 80J \tag{2.17}$$

Logo é necessário produzir 80 J para aquecer a água de 25 °C para 65 °C, como J é $\frac{W}{s}$, assim um resistor de 80 W atenderá os requisitos do sistema. Como a potência no resistor é dado pela equação 2.18, isolando a resistência e substituindo o valor de V_{RMS} pela tensão residencial de 127 V.

$$P = \frac{V_R^2}{R} \tag{2.18}$$

$$R = \frac{{V_R}^2}{P} = \frac{127^2}{1000} \approx 16,129\Omega \tag{2.19}$$

2.4.1 Atuador resistivo

Já obtido o parâmetro atuador, escolheu-se uma resistência de 201,6125 Ω, facilmente encontrado comercialmente, que suporta uma potência máxima de 80 W, o resistor comercial mostrado na Figura 17.

Figura 17 – Resistor para aquecimento do misturador.

Para o controle do atuador, pode-se escolher um gradador, onde se controla o ângulo de acionamento, ou um relé, usando modulação PWM para o controle.

2.4.2 Sensor de temperatura

O sensor de temperatura utilizado no projeto foi o DS18B20, Figura 18. Para ler o valor de temperatura, utiliza o protocolo OneWire para realizar a comunicação com outros dispositivos.

Figura 18 – Sensor de temperatura DS18B20.

Tabela 4 – Características do sensor de vazão.

Características	Valor	Unidade
Faixa de temperatura	-55 a 125	°C
Tensão de operação	3 a 5,5	**V**
Acurácia	0.5	°C

Fonte: Autoria própria.

2.5 VHDL

VHDL é a linguagem usada para o projeto de circuitos digitais em CPLDs, FPGAs e ASICs. Desenvolvida sob o comando do Departamento de Defesa dos Estados Unidos (DARPA), na década de 1980, para documentar o comportamento de ASICs que compunham os equipamentos vendidos às Forças Armadas americanas. Para a compilação e simulação do trabalho, foi utilizado o programa *Quartus Prime Lite Edition Version* 15.1.0, fornecido gratuitamente pela *Altera Corporation*.

Apesar de existir outras linguagens de descrição de *hardware*, como VERILOG, Handel-C, SDL, ISP, entre outros. A linguagem VHDL possui como vantagem seu design de proposito geral se comparado as outras ferramentas existentes, sendo uma das mais usadas para o design de PLD e FPGA, porem outras linguagens são mais especializadas para acessar dispositivos de características comuns (flip-flops e I/O buffers).

A linguagem VHDL é dividida em bibliotecas e pacotes, entidade e arquitetura, contendo atributos, operadores e entre outros.

Para exemplo de um código VHDL, será criado uma porta AND com duas entradas A e B, uma saída S. Primeiramente se declara as bibliotecas e os pacotes a serem utilizados no programa.

```
1 library IEEE;
```

```
2 use IEEE.std_logic_1164.all;
```

Logo após se declara a entidade, onde deve conter as portas da AND.

```
1 entity AND is
2    port (
3       A : in  std_logic;
4       B : in  std_logic;
5       S : out std_logic
6    );
7 end entity AND;
```

Por fim se declara a arquitetura da porta AND, como mostrado a seguir.

```
1 architecture RTL of AND is begin
2    S <= A and B;
3 end architecture RTL;
```

Sendo assim, a criação do componente ficará como mostrado a seguir.

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity AND is
5    port (
6       A : in  std_logic;
7       B : in  std_logic;
8       S : out std_logic
9    );
10 end entity AND;
11
12 architecture RTL of AND is begin
13    S <= A and B;
14 end architecture RTL;
```

Exitem diversos de outros atributos na linguagem VHDL, como sinais,constantes e entre outros, que não foram utilizados no programa, para maiores informações buscar nas referencias.

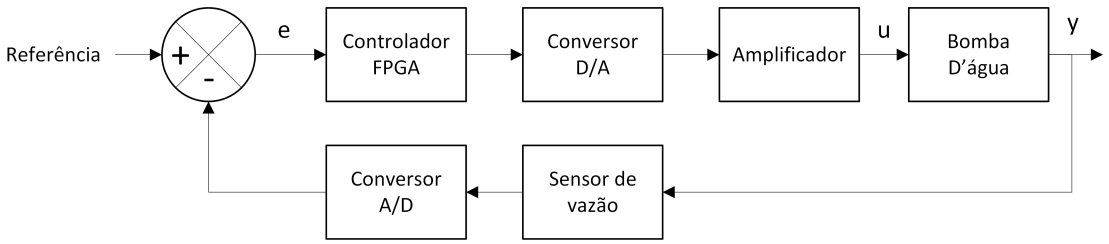
2.6 CONTROLADOR

O objetivo de se controlar um sistema é manter a variável de saída igual a referência, tonando o erro nulo, modificando a variável manipulada. Podendo ser controlado manualmente, malha aberta, ou automaticamente, malha fechada, onde a comparação, controle e sensoriamento são realizados sem a intervenção humana.

Para realizar o controle de vazão no tubo, será utilizado a malha fechada. Onde variável a ser controlada é a tensão na bomba d'água, variável de saída é a vazão no tubo,

o controlador será o FPGA Cyclone IV, Figura 8, o sensor será o sensor de vazão, Figura 16, e o atuador será a bomba d'água, Figura 11, resultando no diagrama de blocos do controlador, mostrado na Figura 19.

Figura 19 – Diagrama de blocos do controlador de vazão da bomba d'água.



Fonte: Autoria própria.

Já para o controle de temperatura no misturador, será utilizado um controlador *ON/OFF*. Onde a variável a ser controlada será a razão cíclica de acionamento do relé, variável de saída é a temperatura do misturador, controlador será o FPGA Cyclone IV, Figura 8, o sensor será LM35, Figura 17, e o atuador será aquecedor mostrado na Figura 17, resultando no diagrama de blocos do controlador mostrado na figura 21.

2.6.1 Controlador ON/OFF

O controlador *ON/OFF* possui dois estados, alto e baixo, ou liga e desliga. Que consiste no controle de uma grandeza, para isso, precisa-se de algumas informações, o valor desejado, real e um algoritmo de controle, essas informações são processados pelo algoritmo de controle que obtêm uma saída que se aproxime do valor desejado.

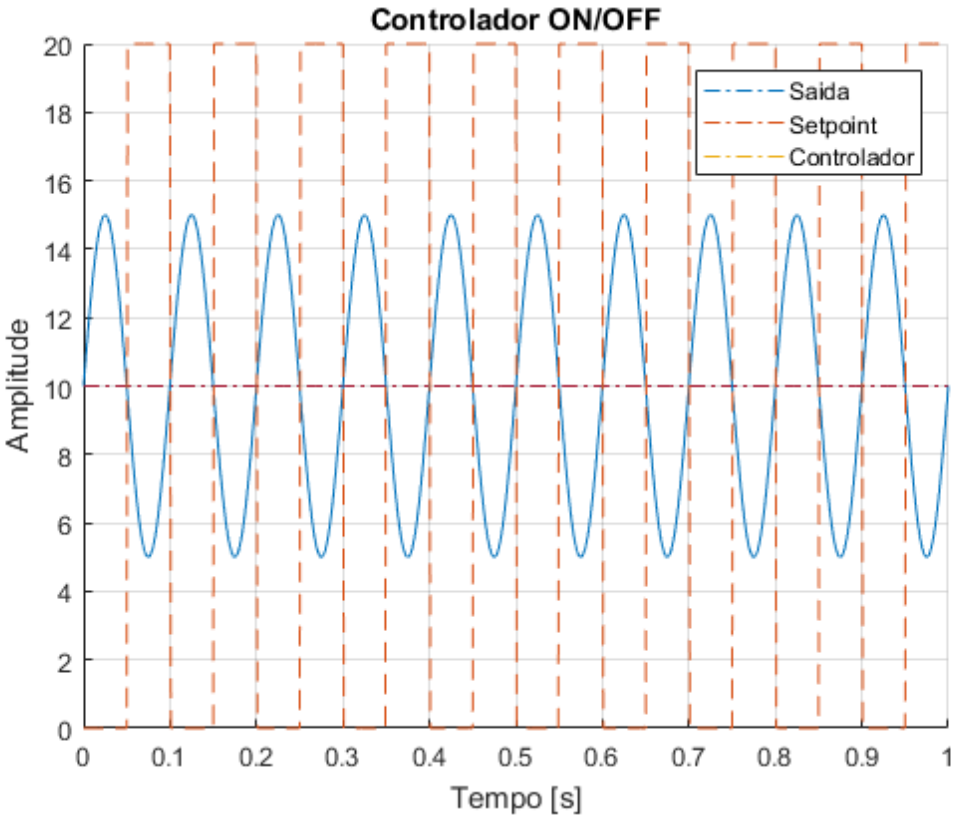
Na Figura 20, pode-se analisar um algoritmo de controlador ON/OFF, nessa figura temos uma saída inicial de 15, o controlador terá que manter a saída do sistema sobre o *setpoint*. Toda vez que o sensor ler um valor de saída menor que o *setpoint*, a saída terá um valor alto, porem quando seu valor for maior que o *setpoint*, sua saída será baixa.

A maior vantagem de se usar esse tipo de controlador esta em sua facilidade de projeto e seu baixo custo, porem sua saída sempre oscilará entorno do *setpoint*, podendo aumentar a amplitude conforme. Esse oscilação pode não ser desejado, necessitando de outro método de controle, como o PID.

2.6.2 Protótipo

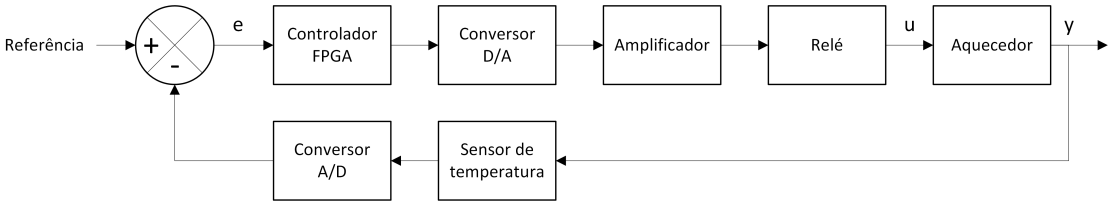
O sistema completo será mostrado na Figura 22, mostrando os locais de sinais elétricos e térmicos, tubulações, sensores de vazão e temperatura, o aquecedor e as bombas

Figura 20 – Diagrama de blocos do controlador de temperatura do aquecedor.



Fonte: Autoria Propriá.

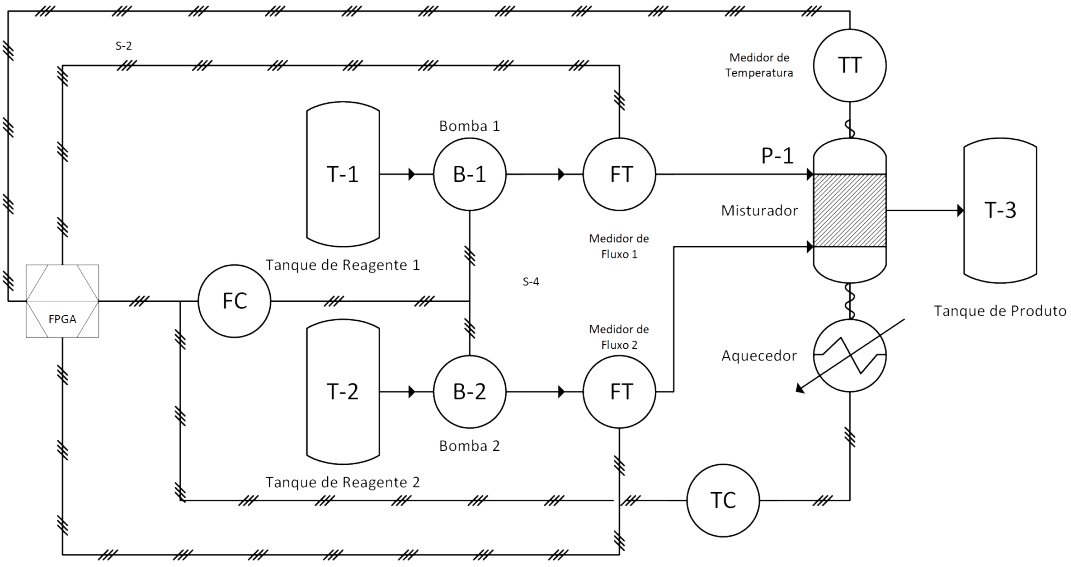
Figura 21 – Diagrama de blocos do controlador de temperatura do aquecedor.



Fonte: Autoria própria.

d'águas.

Figura 22 – Diagrama de Processos e Instrumentação P&DI.



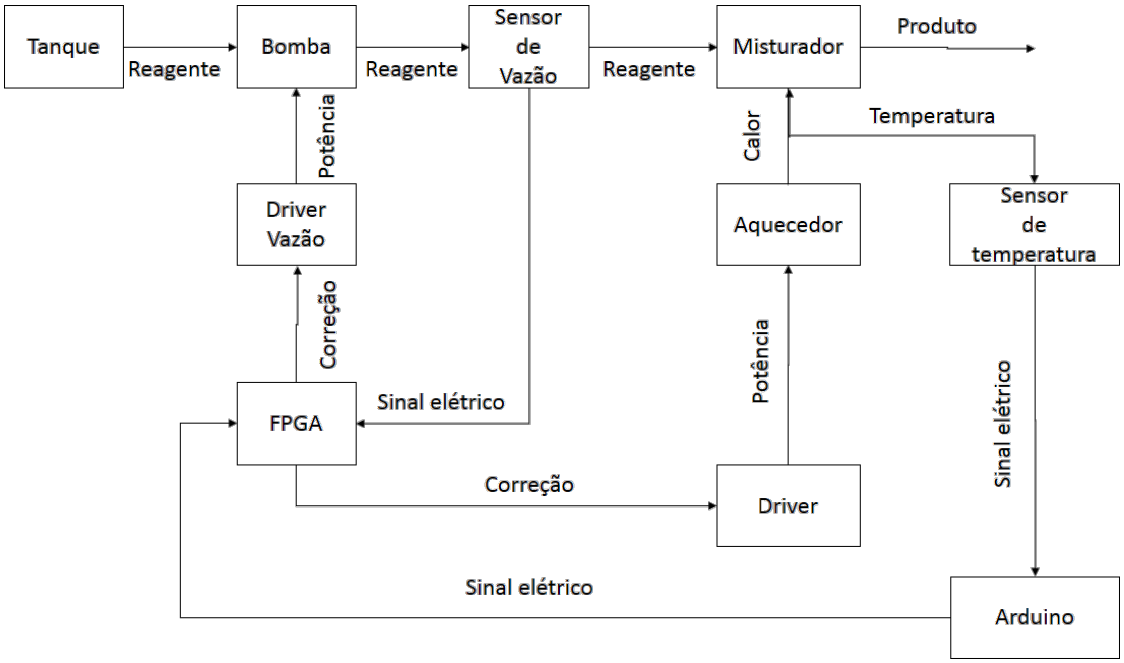
Fonte: Autoria própria.

3 METODOLOGIA

3.1 DESCRIÇÃO DO PROTÓTIPO

Um sistema para controlar parâmetros de uma reação química implementado em uma FPGA é mostrado na Figura 23. Este sistema é composto por controladores de temperatura e vazão. O controle de temperatura é feito por um aquecedor, um sensor e um circuito de controle. O controle da vazão é composto por um sensor, uma bomba de alimentação e um circuito de controle.

Figura 23 – Desenho esquemático de um sistema para realizar a produção química.



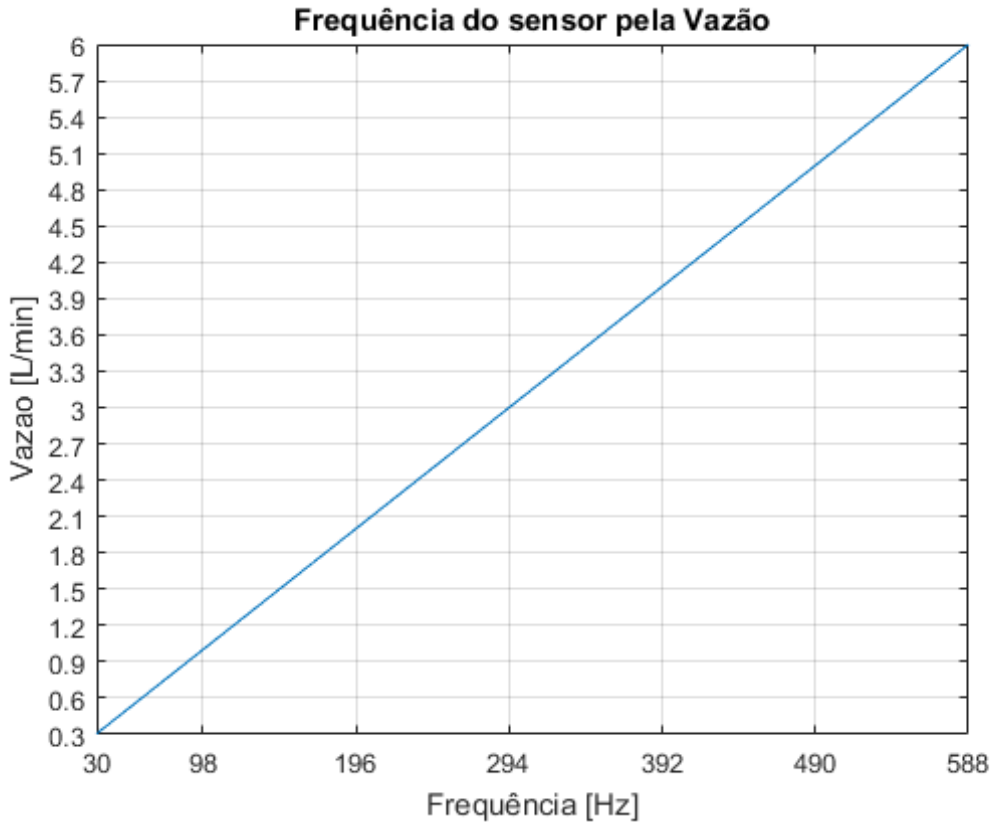
Fonte: Autoria própria.

3.2 CÁLCULO DO SENSOR DE VAZÃO

No sensor de vazão YF-S401, a conversão de vazão (Q) em frequência (F) é linear, como observado no gráfico da Figura 24. Através dos dados do *datasheet*, (LEONG *et al.*, 2016), a vazão é multiplicada por um coeficiente conforme a equação 3.1.

$$Q = \frac{F}{98}\left[\frac{L}{min}\right] \tag{3.1}$$

Figura 24 – Gráfico entre vazão e frequência.



Fonte: Autoria própria.

3.2.1 Cálculo da frequência

Primeiramente, para calcular a frequência do sensor no FPGA, foi necessário um oscilador com uma frequência muito maior que o sensor. Como a vazão máxima medida é de 6 L/min, a maior frequência gerada pelo sensor será de 588 Hz, muito menor que a frequência do oscilador interno (25 MHz) da placa FPGA utilizada.

A definição da frequência do sensor, baseou-se na ideia do divisor de *clock*, que consiste em dividir um *clock* maior de entrada, (F_i), por um determinado valor, para obter um de menor valor na saída, (F_o), conforme a equação 3.2. Para este objetivo, foi implementado um contador de pulsos do *clock* interno da placa FPGA, onde a cada borda de subida incrementa-se o registrador denominado contador. Desta forma, quando há uma borda de subida ou descida do sensor de vazão, o valor da frequência do FPGA é dividido pelo contador.

$$F_o = \frac{F_i}{contador} \tag{3.2}$$

3.3 MODELAGEM MATEMÁTICA DA BOMBA D'ÁGUA

A identificação dos parâmetros da planta foi feita utilizando o Algoritmo de Mínimos Quadrados (AMQ), o qual consiste no critério de minimização da diferença entre parâmetros estimados e reais. Para a aplicação deste método, é necessário considerar a bomba como um sistema causal, linear e invariante no tempo.

A planta pode ser descrita pela equação 3.3, onde $a(z)$ é a matriz de polos, $b(z)$ e de zeros, $u(k)$ as entradas, $y(k)$ as saídas do sistemas e z^{-1} representa o operador de atraso e k o ciclo atual.

$$a(z)y(k) = b(z)u(k) \tag{3.3}$$

Onde $a(z)$ (3.4) e $b(z)$ (3.5) pode ser apresentados como sendo um somatório de 0 a N, sendo N a maior ordem do sistema.

$$a(z) = 1 + \sum_{n=1}^{N} a_n z^{-n} \tag{3.4}$$

$$b(z) = \sum_{n=0}^{N} b_n z^{-n} \tag{3.5}$$

As constantes no projeto estão representadas pela equação 3.6, e o vetor de regressão pela equação 3.7, sendo ambos de ordem $2N - 1$.

$$\theta = \begin{bmatrix} a_1 \\ \vdots \\ a_n \\ b_0 \\ \vdots \\ b_n \end{bmatrix} \tag{3.6}$$

$$\Phi = \begin{bmatrix} -y(k-1) \\ \vdots \\ -y(k-n) \\ u(k) \\ \vdots \\ u(k-n) \end{bmatrix} \tag{3.7}$$

O algoritmo necessita de iterações para aproximar os valores dos parâmetros estimados ($\hat{\theta}$), com condições iniciais nulas, para os valores reais de θ. O vetor $\hat{y}$ representa a saída estimada, e $e(k)$, representa a diferença entre o valor real e o valor estimado, conforme explicitado pelas equações 3.8, 3.9 e 3.10.

$$\hat{\theta} = \left(\sum_{i=0}^{k} \Phi^T(i) y(i) \right)^{-1} \sum_{i=1}^{k} \Phi(i) y(i) \tag{3.8}$$

$$\hat{y} = \theta^T \Phi(k) \tag{3.9}$$

$$e(k) = y(k) - \hat{y}(k) \tag{3.10}$$

A fim de garantir a eficiência da estimação dos parâmetros pelo AMQ, arranja-se a formula de $\Phi(k)$ para uma matriz auxiliar, $P(k)$, de ordem 2NX2N, mostrado na equação 3.11.

$$P(k) = \left(\sum_{i=1}^{k} \Phi(i)\Phi(k)^T \right)^{-1} \tag{3.11}$$

Fazendo os devidos arranjos determina-se os novos valores de P.

$$P(k) = P(k-1) - \frac{P(k-1)\Phi(k)\Phi(k)^T P(k-1)}{1 + \Phi(k)^T P(k-1)\Phi(k)} \tag{3.12}$$

Calculado o valor de $P(k)$ determina-se o novo valor de $\hat{\theta}$, descrita pela equação 3.13.

$$\theta(k) = \hat{\Phi}(k-1) + P(k)\Phi(k)e(k) \tag{3.13}$$

Assegurando que $P(0)$ seja suficientemente grande para garantir a confiança dos parâmetros, foi usado um valor elevado qualquer na matriz de identidade, desta forma, na equação 3.14, o valor escolhido foi de 10k.

$$P(0) = \begin{bmatrix} 10k & 0 & \dots & 0 \\ 0 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & 10k \end{bmatrix} \tag{3.14}$$

Para extrair a resposta do sistema, a bomba foi ligada com uma tensão de 12 V, e desta maneira foi possível retirar os valores da vazão pelo tempo, através do auxilio de um Arduino, que foram salvos em um arquivo e aplicado no algoritmo computacional.

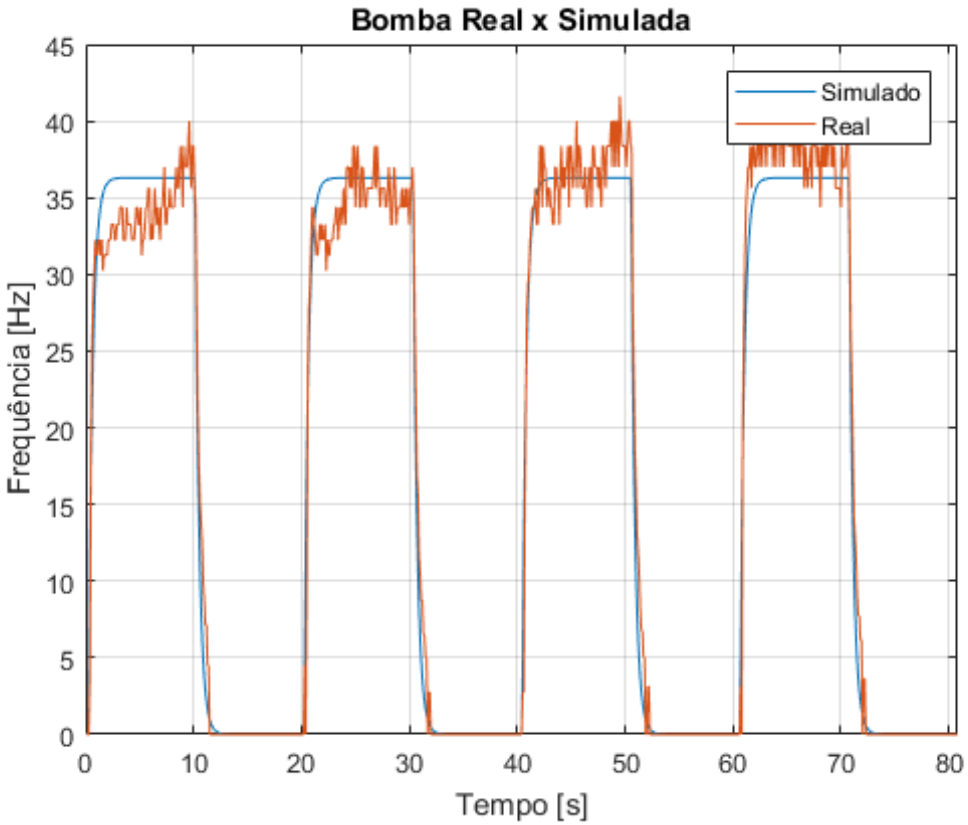
Após esse processo, foi possível obter a planta discretizada do sistema, dada pela equação 3.15, a Figura 25 exibe o comparativo entre comportamento da bomba real e do modelo encontrado pelo método de identificação.

$$H(z) = \frac{0.755z^2 + 0.9965z + 0.1724}{z^3 + 0.5159z^2 + 0.07642z + 0.08965} \tag{3.15}$$

Após a identificação da planta pelo algoritmo de mínimos quadrados, pode-se plotar o lugar das raízes, onde nota-se que a bomba é estável, por não possuir um polo ou zero positivos no eixo real. Para pode-se reescrever a equação para 3.16, afim de realizar os cálculos do controlador utilizando os métodos clássicos do controle.

$$H(s) = \frac{2.463s^2 + 56.5s + 7916}{s^3 + 24.12s^2 + 551.5s + 1309} \tag{3.16}$$

Figura 25 – Comparação entre o modelo matemático e a bomba real.



Fonte: Autoria própria.

3.4 CÁLCULO DO CONTROLADOR DE VAZÃO

Para definir o controlador é necessário ter parâmetros fixos para o inicio dos cálculos. Neste caso o *Overshoot* do sistema (PO) foi escolhido para 10 %, tempo de estabelecimento de 1 s e com erro tendendo a zero.

Com o *Overshoot* determina-se a taxa de amortização, (ζ), definida pela equação 3.18, obtendo o ângulo máximo do polo, θ, para satisfazer o parâmetro, como demonstrado pela equação 3.18.

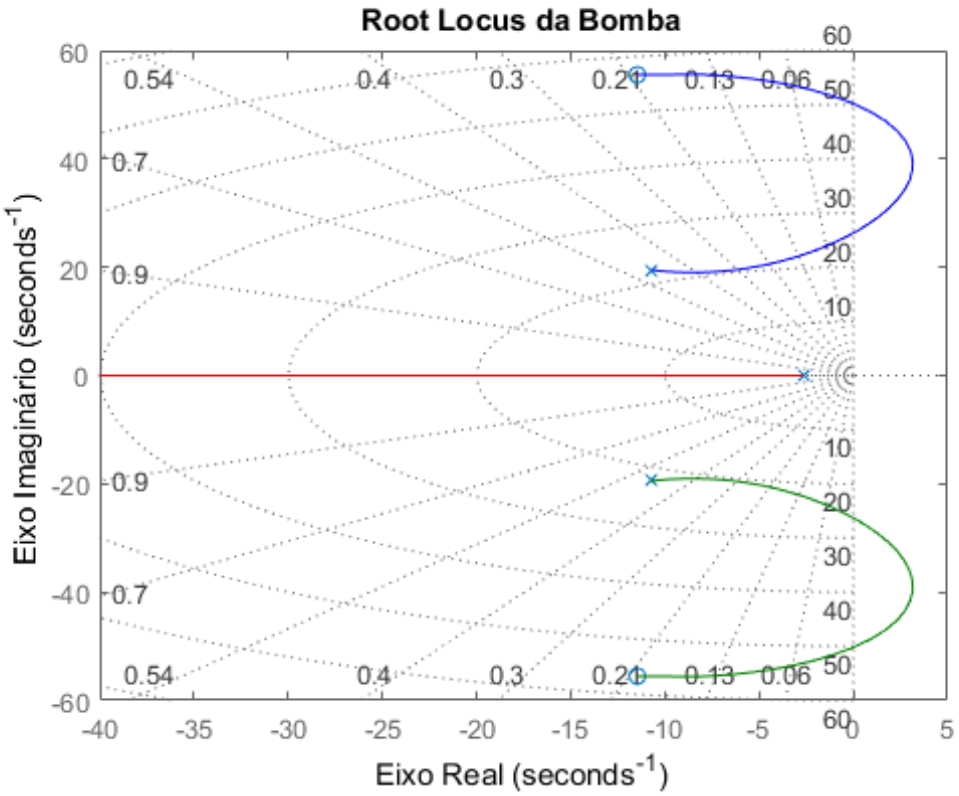
$$\zeta = \sqrt{\frac{\log\left(\frac{PO}{100}\right)^2}{\pi^2 + \log\left(\frac{PO}{100}\right)^2}} = 0.4559 \tag{3.17}$$

$$\theta = tan^{-1}\left(\frac{\sqrt{1-\zeta}}{\zeta}\right) = 62.87 \tag{3.18}$$

Determinado a taxa de amortização e com o tempo de estabelecimento, Te, pode-se determinar a frequência natural, ω_n.

$$Te = \frac{4}{\zeta\omega_n} = 4s \tag{3.19}$$

Figura 26 – O lugar das raízes no sistema de vazão.



Fonte: Autoria própria.

$$\omega_n = \frac{4}{\zeta Te} = 2.1932 \tag{3.20}$$

Redesenhando o *Root Locus* com os limites do parâmetros proposto, tem-se a Figura 27.

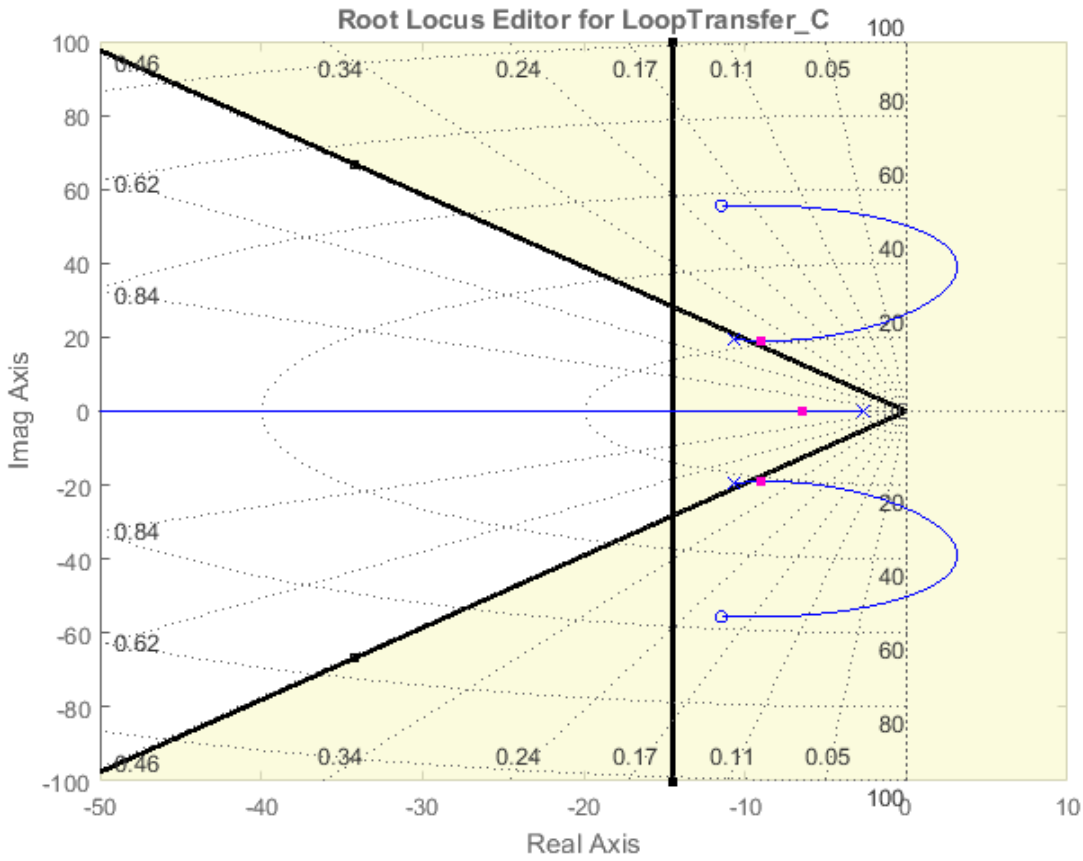
Para obter um sistema com o erro tendendo a zero, foi adicionado um integrador, com polo localizado na origem. Desta forma, obtêm-se o novo *Root Locus*, Figura 28.

Com o auxilio de um *software* matemático, obteve-se um ganho aproximado de 16, que satisfaça os parâmetros do projeto. equação 3.21 representa o controlador encontrado.

$$C\left(s\right) = \frac{0.2}{s} \tag{3.21}$$

Na Figura 29, nota-se que a resposta ao degrau do controlador encontrado, que cumpre os requisito de projeto, o parâmetros do controlador, $Te < 1s$ e $PO < 20$ %.

Figura 27 – Limites do lugar das raízes com os limites dos parâmetros.



Fonte: Autoria própria.

3.4.1 Discretização do controlador

Em dispositivos digitais, tal como o FPGA, é necessário discretizar a função de controle 3.21. O método utilizado no projeto foi o de invariância ao degrau que já leva em consideração o *holder* de ordem zero, equação 3.22, onde $G(s)$ a função de transferência que descreve a planta da bomba no tempo continuo e $G(z)$, equação 3.24, é a planta no tempo discreto.

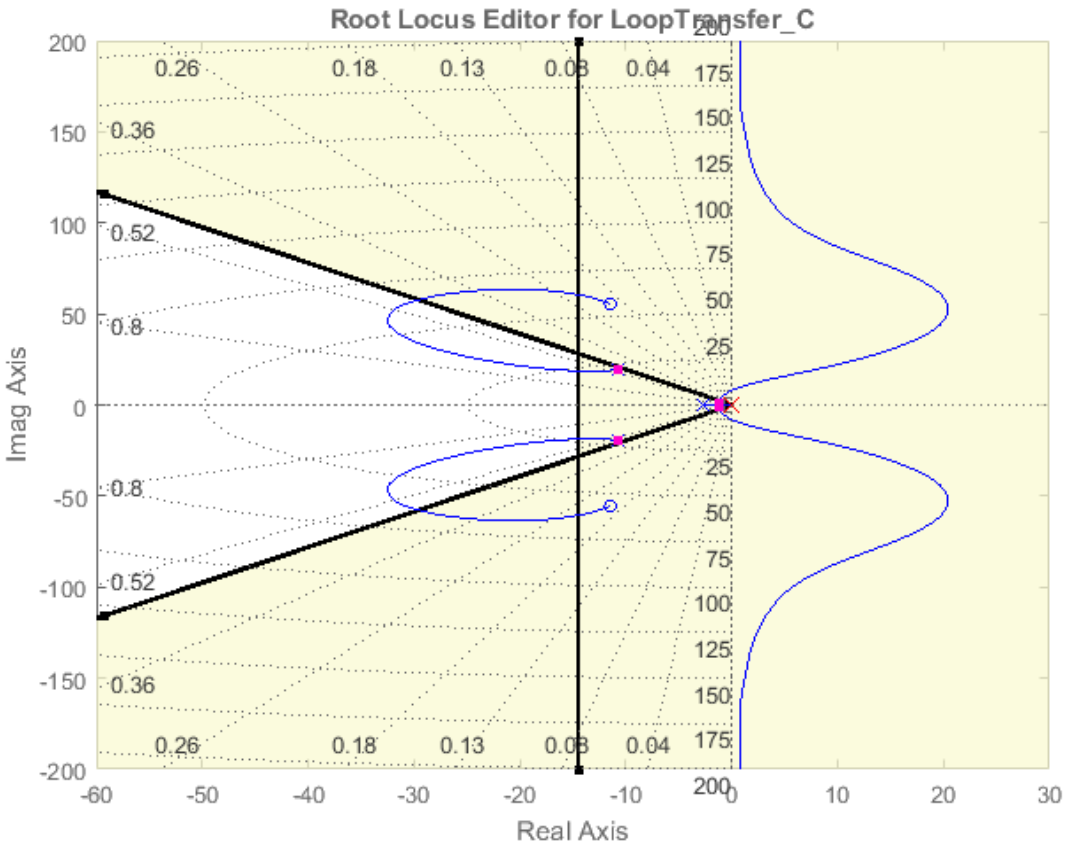
$$G(z) = \left(1 - z^{-1}\right) \mathcal{Z} \left\{ \frac{G(s)}{s} \right\} \tag{3.22}$$

$$G(z) = \left(1 - z^{-1}\right) \mathcal{Z} \left\{ \frac{\left(\frac{0.2}{s}\right)}{s} \right\} \tag{3.23}$$

$$G(z) = \frac{0.02}{z - 1} \tag{3.24}$$

$G(z)$ é descrito pela equação 3.25, onde $U(z)$ representa a entrada e $Y(z)$ a saída do sistema.

Figura 28 – Root Locus com o integrador e os limites.



Fonte: Autoria própria.

$$G(z) = \frac{Y(z)}{U(z)} \tag{3.25}$$

A equação recursiva 3.26, foi reorganizada, tal como mostra a equação 3.27, sendo este controlador implementado no FPGA.

$$u(k) = 0.02e(k) + u(k-1) \tag{3.26}$$

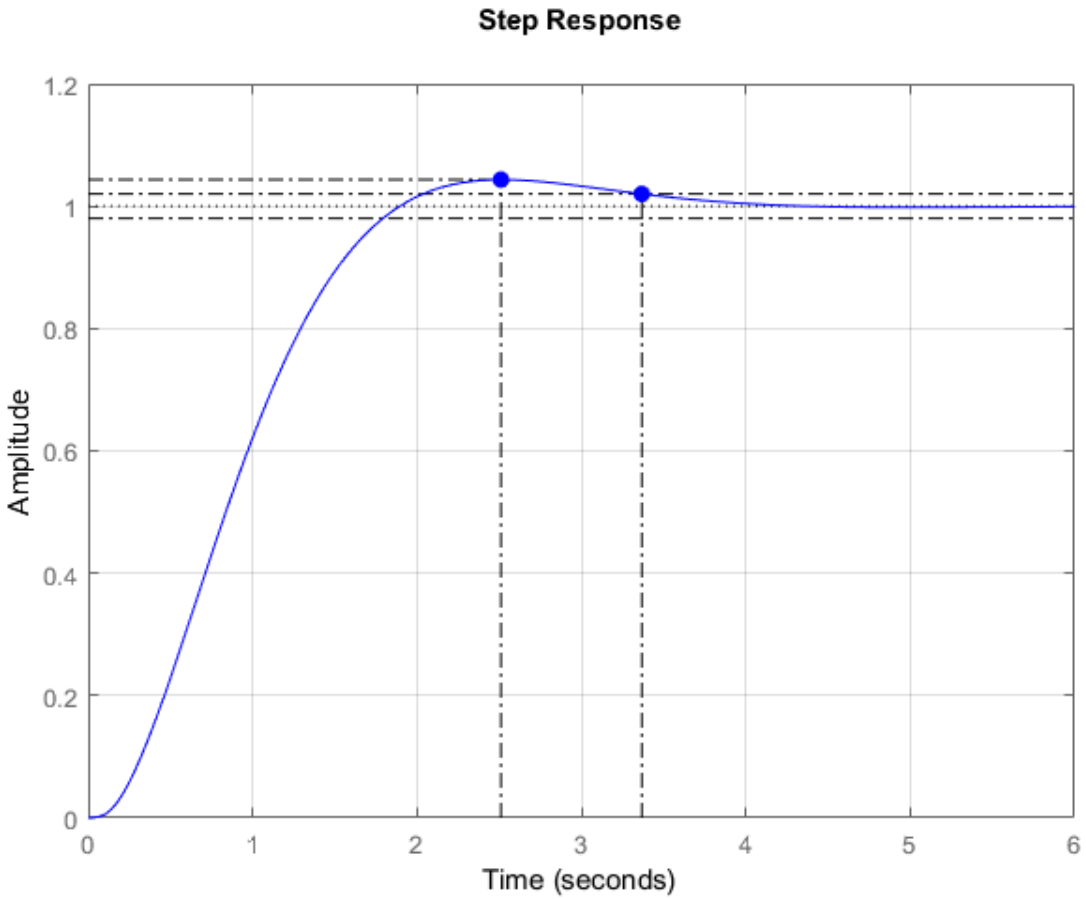
$$u(k) = \frac{e(k)}{50} + u(k-1) \tag{3.27}$$

3.5 MODULAÇÃO POR LARGURA DE PULSO

A bomba d'água utilizada nesse projeto consiste em um motor elétrico de corrente continua acoplado a pás, que empurram a água da entrada para a saída. Para obter diferentes valores de vazão é necessário variar a velocidade do motor, ou seja, é preciso variar a tensão média aplicada.

O FPGA utilizado nesse trabalho não possui uma saída analógica capaz de variar a tensão aplicada sobre a bomba, sendo necessário utilizar um driver para fazer o

Figura 29 – Resposta ao degrau do sistema com compensador.



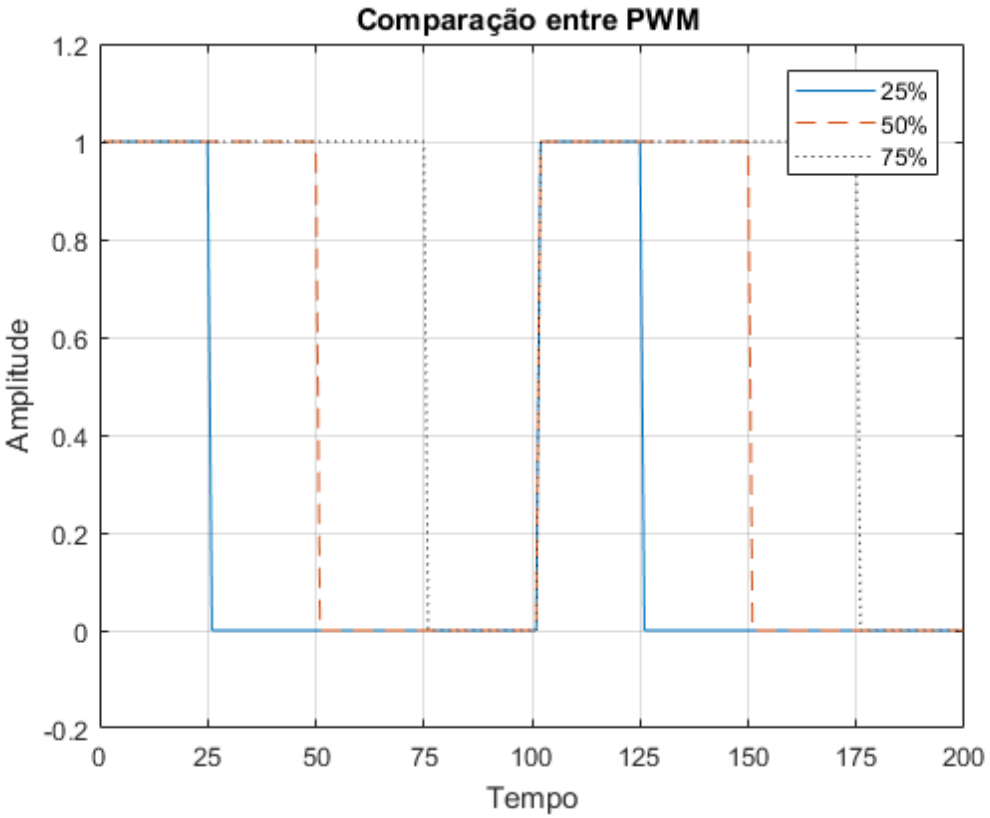
Fonte: Autoria própria.

acionamento. A variação de tensão será implementado através da modulação PWM.

A modulação PWM, ou por largura de pulso, consiste em variar a porcentagem do tempo que o FPGA mantem a saída do pino em valor alto durante um período fixo de tempo. Desta forma, ajusta-se linearmente a tensão média aplicada sobre o motor conforme a porcentagem de tempo que a saída fica elevada. Esta modulação é exemplificada na Figura 30.

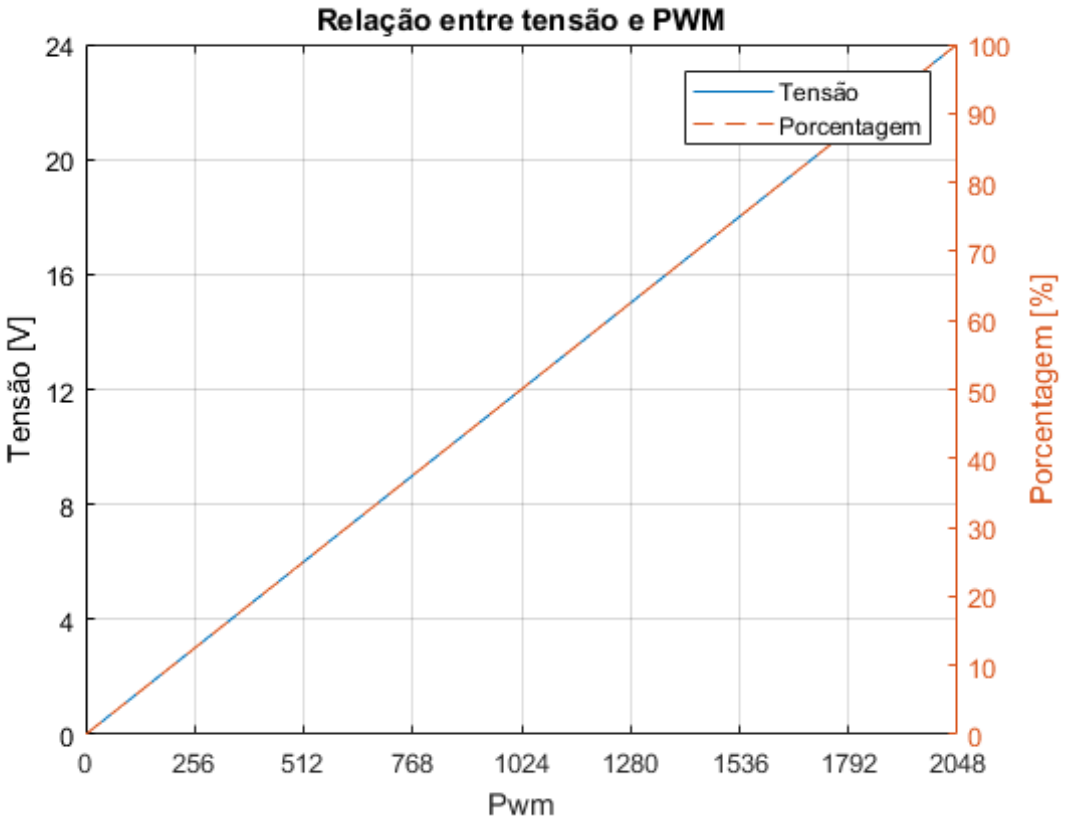
Para implementar um PWM em VHDL, foi criado um registrador de contador que conta as bordas de subida do *clock* de 0 a 2048, que quando chega a 2048, reinicia o contador. A saída do PWM é obtido com a comparação do contador com o valor de entrada fornecido pelo controlador, variando o valor da tensão sobre o motor de 0 V até a alimentação, como mostrado pela Figura 31.

Figura 30 – Comparativo de PMW com 25, 50 e 75%.



Fonte: Autoria própria.

Figura 31 – Relação entre tensão, porcentagem e PWM.



Fonte: Autoria própria.

O *clock* de entrada bloco PWM, cujo o valor é de 25 MHz, é dividido pelo contador do PWM, que no caso contem 2048 estados. A frequência dos pulsos é encontrada pela equação 3.28.

$$F_{PWM} = \frac{F_{CLK}}{contador_{max}} = \frac{25MHz}{2048} \approx 12207Hz \tag{3.28}$$

O motor possui comportamento semelhante à um indutor associado em série com uma resistência, agindo desta forma, como um filtro passa-baixa. Portanto, a tensão média passará completamente para o motor, visto que sua portadora esta localizada em 0 Hz.

Segue abaixo um exemplo de código em VHDL, que implementa a modulação PWM.

```
1       process(clk) is begin
2          if(clk'event and clk = '1') then
3             if(contador > 2047) then
4                contador <= 0;
5             else
6                contador <= contador + 1;
7             end if;
8          end if;
9          if(contador < controle) then
10            saida <= '1';
11         else
12            saida <= '0';
13         end if;
14      end process;
```

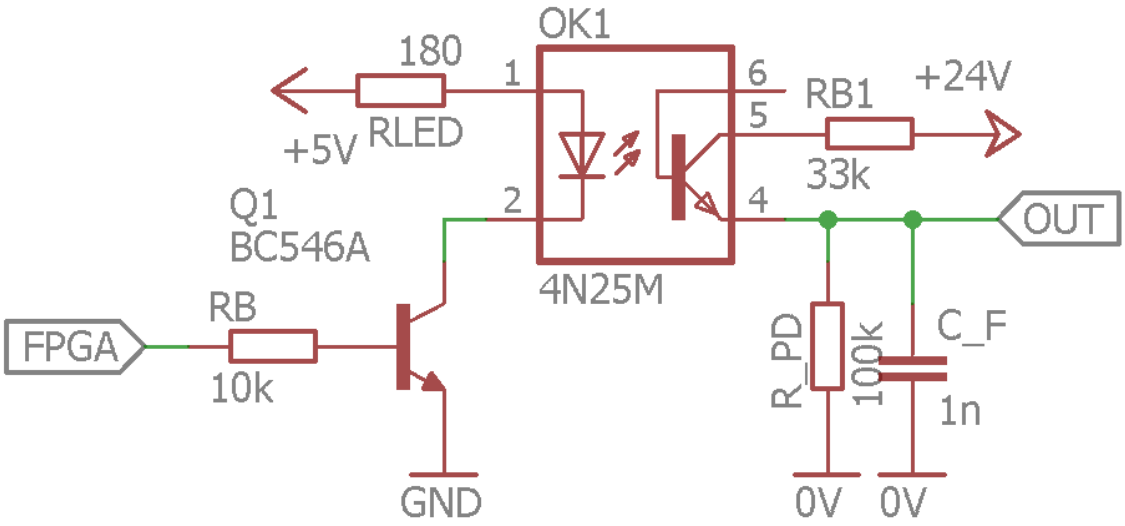
3.6 PLACA DE ACIONAMENTO DA BOMBA

O *MOSFET* é um dispositivo que fornece agilidade e a potência necessária para acionamento da bomba de 24 V. Para acionar é necessário que se aplique uma tensão entre o *Gate* e *Source*, (V_{GS}), maior que o valor de *threshold*, $V_{GS_{TH}}$. Para que a parte de potência não interfira no sinal de controle é necessário isolar eletricamente, sendo assim, utilizou-se um optoacoplador.

O optoacoplador 4N25 possui em seu interior um LED, que quando diretamente polarizado aciona um fototransistor. Este dispositivo será alimentado com 5 V, entretanto há uma queda de tensão no LED de 1,3 V. Definiu-se que a corrente do LED terá o valor de 20 mA, desta forma a resistência entre Vcc e a entrada do optoacoplador é dada pela equação 3.29.

$$R_{LED} = \frac{V_{CC} - V_{LED}}{i_{LED}} = \frac{5 - 1,3}{20m} = 185 \approx 180\Omega \tag{3.29}$$

Figura 32 – Driver para o acionamento do MOSFET.



Fonte: Autoria própria.

Um TBJ é utilizado para amplificar o sinal do FPGA para o optoacoplador. TBJ escolhido foi o BC546A, que possui ganho (β) de 110, uma queda de tensão entre a base e o emissor (V_{BE}) quando polarizado de 0,7 V e uma tensão sobre o coletor e o emissor (V_{CE}) polarizado de 0 V. Sabendo que a corrente desejada é de 20 mA, podemos calcular os resistores pelas equações abaixo.

Como a corrente do coletor do transistor é igual ao *LED*, i_{LED}, pode-se calcular a corrente de base utilizando a relação de ganho entre o coletor e base, equação 3.30.

$$\beta = \frac{i_C}{i_B} \tag{3.30}$$

$$110 = \frac{i_{LED}}{i_B} = \frac{20m}{i_B} \tag{3.31}$$

Isolando a corrente de base, i_B, tem-se a equação 3.32.

$$i_B = \frac{20m}{110} = 181uA \tag{3.32}$$

Quando se aplica a lei de Kirchhoff para tensões obtêm-se a tensão sobre o resistor de base, equação 3.33

$$V_{FPGA} = V_{BE} + R_B iB \tag{3.33}$$

Isolando a resistência de base, R_B, obtemos e a equação 3.35.

$$R_B = \frac{V_{FPGA} - V_{BE}}{i_B} \tag{3.34}$$

Substituindo os valores da equação pelos reais, onde V_{FPGA} é igual a 3,3 V e V_{BE} é igual a 0,7 V, teremos a equação 3.35.

$$R_B = \frac{3,3 - 0,7}{181u} = 14,3k \approx 10k\Omega \tag{3.35}$$

Para o cálculo do resistor de coletor do optoacoplador, considera-se que a tensão entre o gate e *Source* do *MOSFET*, V_{GS}, deve ser menor que 20 V, pois esta é a tensão máxima que pode ser aplicada sobre o *Gate* do *MOSFET*. Neste caso foi utilizado uma tensão V_{GS} de 18 V.

Aplicando a lei de Kirchhoff para tensões obtêm-se a equação 3.36, onde V_{SS} é tensão fornecida pela fonte de potência de 24 V.

$$V_{SS} = R_C i_C + R_{GS} i_{GS} \tag{3.36}$$

Como a corrente sobre o resistor entre o *Gate* e *Source*, R_{GS}, é igual a corrente sobre o coletor, fixando o valor do resistor do emissor de 100 $k\Omega$, tem-se:

$$i_{GS} = \frac{V_{GS}}{R_{GS}} = \frac{18}{100k} = 180uA \tag{3.37}$$

Substituindo o valor de m 3.37 de 3.36, obtêm-se a equação 3.38.

$$24 = R_C i_C + 18 \tag{3.38}$$

Resolvendo a equação 3.38, tem-se a equação 3.39 isolando R_C.

$$R_C = \frac{V_{SS} - V_{GS}}{i_{GS}} = \frac{24 - 18}{180u} = 33.33k \approx 33k\Omega \tag{3.39}$$

A potência aplicada sobre o *LED* pode ser calculada pela equação 3.40.

$$P_{LED} = V_{LED} i_{LED} = 26mW \tag{3.40}$$

A potência sobre o resistor de base pode ser calculada pela equação 3.41.

$$P_{R_B} = R_B i_B^2 = 473uW \tag{3.41}$$

A potência aplicada sobre o transistor pode ser calculada pela equação 3.42.

$$P_T = V_{BE} i_E = 14mW \tag{3.42}$$

A potência aplicada sobre o resistor do coletor do transistor pode ser calculada pela equação 3.43.

$$P_{R_{LED}} = R_{LED} i_{LED}^2 = 74mW \tag{3.43}$$

3.6.0.1 Placa do MOSFET

A placa de acionamento do MOSFET fabricado a partir do esquemático é mostrada na Figura 33.

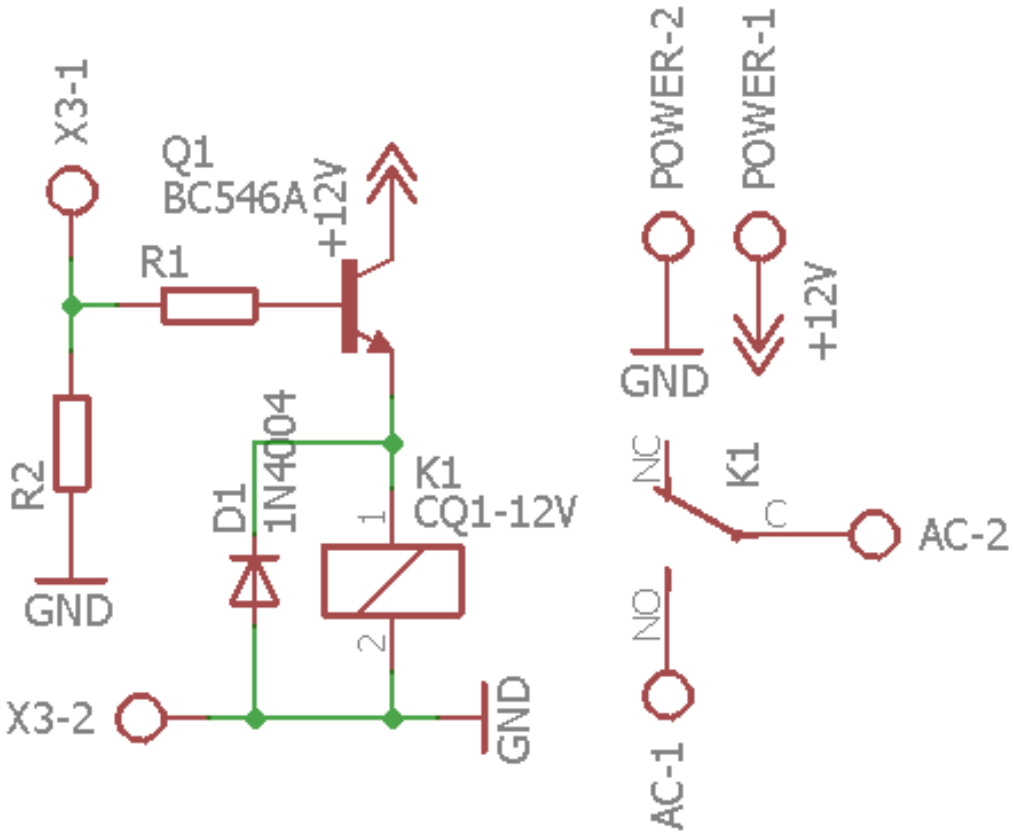
Figura 33 – Placa de acionamento da bomba.

Fonte: Autoria própria.

3.7 PLACA DE ACIONAMENTO DO AQUECEDOR

A placa para o acionamento da resistência que aquece a água foi projetada com um relé para fazer o chaveamento. Para os cálculos da placa do relé, utilizou-se como parâmetros uma tensão aplicada sobre o relé, (V_{CC}), de 12 V, uma resistência do relé, (R_{rele}) de 330 Ω, e uma tensão de acionamento do transistor fornecido pelo FPGA, (V_{FPGA}), de 3,3 V.

Figura 34 – Placa de acionamento do aquecedor.



Fonte: Autoria própria.

Utilizando a regra de Kirchhoff para tensões, temos a equação 3.44.

$$Vcc + Vbe + R_{rele} i_{rele} = 0 \tag{3.44}$$

Sabendo que a tensão sobre o transistor saturado é de aproximadamente de 0 V, quando substituímos este valor na equação 3.44 temos a equação 3.45.

$$irl = \frac{Vcc}{R_{rele}} = \frac{12}{330} = 36,4mA \tag{3.45}$$

O ganho do transistor, β é 110, sendo descrito pela equação 3.30, onde a corrente do emissor é igual a corrente sobre o relé.

$$\beta = \frac{i_e}{i_b} - 1 \tag{3.46}$$

Isolando o valor da corrente de base, i_b, obteve-se a corrente, como visto na equação 3.47.

$$ib = \frac{i_{rele}}{\beta + 1} \frac{36,4m}{110} \approx 330uA \tag{3.47}$$

Calculando o valor do resistor de base utilizando as leis de Kirchoff, temos a equação 3.48.

$$V_{FPGA} + R_B i_B + V_{BE} = 0 \tag{3.48}$$

Isolando a resistência de base na equação 3.48, resulta-se na equação 3.49.

$$R_B = \frac{V_{FPGA} - V_{BE}}{ibr} = \frac{2,6}{330u} \approx 8k\Omega \tag{3.49}$$

3.7.0.1 Placa projetada

A placa fabricada do esquemático é mostrada na Figura 35.

Figura 35 – Placa fabricada do esquemático.

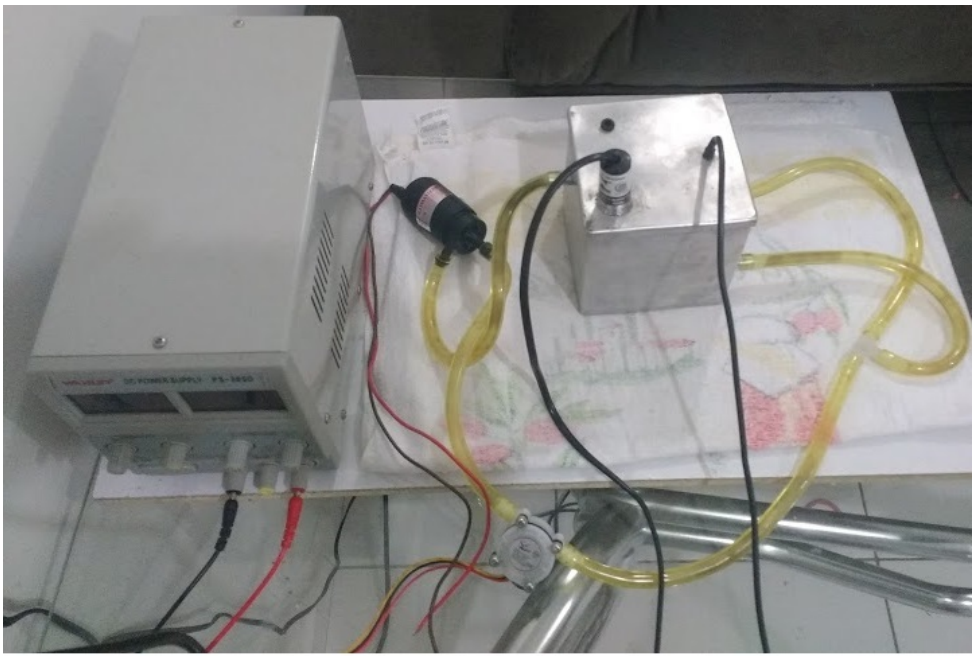
Fonte: Autoria própria.

4 RESULTADOS

4.1 PROTÓTIPO

A Figura 36 mostra o protótipo do trabalho, o sensor de temperatura e aquecedor estão colocados direto no misturador, que foi feito de alumínio com 10 cm de medidas cada lado e 1 L de volume.

Figura 36 – Protótipo fabricado para o trabalho.



Fonte: Autoria própria.

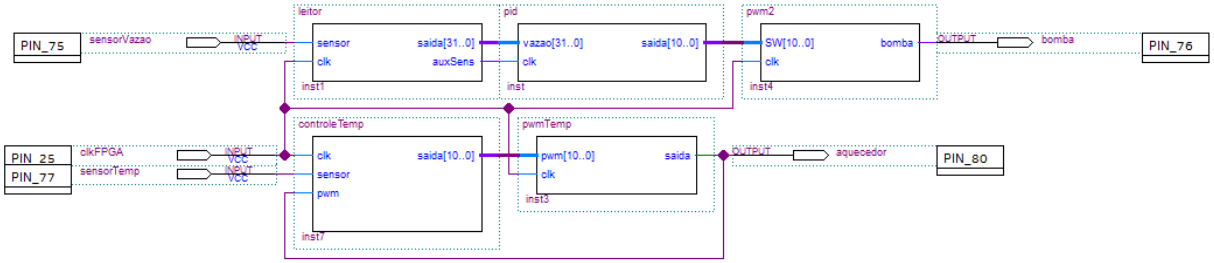
4.2 VAZÃO

O projeto em VHDL, foi descrito e sintetizado utilizando o *software* Quartus II, versão 18. A Figura 37, demonstra a alocação dos blocos utilizada no projeto de vazão.

4.2.1 Leitor

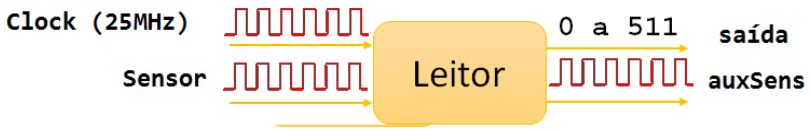
O bloco do leitor, mostrado na Figura 37, realiza a leitura do sensor de vazão, contando quantas bordas de subida de estado do *clock* se passaram entre uma borda do sensor e outra.

Figura 37 – Organização dos blocos do VHDL.



Fonte: Autoria própria.

Figura 38 – Representação do bloco Leitor.



Fonte: Autoria Própria.

Como o contador dividirá a frequência que entra no pino *clock*, que é fixa em 25 MHz, armazenou-se o valor em *fclk*.

```
1    generic(
2        fclk   :   integer := 25_000_000;
3    );
```

As portas **sensor** e **clk** são do tipo entrada, sendo ambas as portas do tipo STD LOGIC. A porta **sensor** está ligada diretamente ao sensor de vazão, e a porta **clk** ligada ao *clock* interno da placa FPGA usada no projeto. A porta **saída** exibe o valor da vazão lida do sensor, e a porta **auxSens** serve de aviso sobre o cálculo da vazão.

```
1    port(
2        sensor    :   in     std_logic;
3        saida     :   out    integer;
4        auxSens   :   out    std_logic;
5        clk       :   in     std_logic;
6    );
```

Para realizar os cálculos em VHDL, foram usados sinais *current*, aux, saux e nova. O sinal *current* conta com um número de borda de subida da porta de entrada clk, o sinal aux armazena o valor prévio da vazão, saux guarda o valor anterior do sensor e nova armazena o valor prévio do auxSens.

```
1    signal    current   :   integer range 0 to 25_000_000;
2    signal    prev      :   integer;
3    signal    aux       :   integer;
4    signal    saux      :   std_logic;
5    signal    nova      :   std_logic;
```

Apesar da lógica do bloco já ter sido exemplificada anteriormente, vale reiterar que o processo de cálculo da vazão, se baseia na ideia de dividir a frequência do *clock* de entrada pelo contador armazenado no sinal *current*. Resultando no código a seguir que junta os blocos explicados acima.

```
1    saida    <= aux;
2    auxSens  <= nova;
3    process(clk) is begin
4       if(clk'event and clk = '1') then
5          current  <= current + 1;
6          if(current > 24_999_999) then
7             current  <= 0;
8             aux      <= 0;
9             nova     <= not nova;
10         elsif(sensor /= saux) then
11            saux     <= sensor;
12            aux      <= 100_000_000/current;
13            current  <= 0;
14            nova     <= not nova;
15         end if;
16      end if;
17   end process;
```

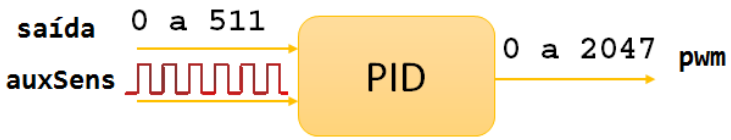
Tabela 5 – Uso do FPGA pelo bloco leitor.

Nome	Usado	Máximo
Elementos Lógicos	1296	6272
Registradores Usados	103	

Fonte: Autoria própria.

4.2.2 PID

Figura 39 – Representação do bloco PID.



Fonte: Autoria Própria.

O controlador seguiu as ideias dos cálculos propostos anteriormente na seção de fundamentação teórica. O bloco tem como portas de entrada a vazão, valor lido pelo bloco **leitor** e passado ao bloco **PID** de tipo STD LOGIC VECTOR com 32 bits, o clk gerado pelo bloco leitor através da saída **auxSens**, que alerta o bloco **pid** sobre o cálculo da vazão,

e a saída **saida**, é o resultado da equação projetada de minimização do erro que vai **PWM**. Como mostrado no código a seguir.

```
1    saida     <= std_logic_vector(to_unsigned(aux2a,saida'length));
2    vazaoi    <= to_integer(unsigned(vazao));
3    process(clk,aux3,erro,aux2,aux2a) is begin
4       if(clk'event and clk = '1') then
5          if(vazaoi < 1000) then
6             aux3  <= vazaoi;
7          end if;
8          if(aux4 > 2047) then
9             aux2a <= 2047;
10         elsif(aux4 < 100) then
11            aux2a <= 100;
12         else
13            aux2a <= aux4;
14         end if;
15      end if;
16      erro  <= 400 - aux3;
17      aux2  <= 1489*erro/10000;
18      aux4  <= aux2 + aux2a;
19   end process;
```

Tabela 6 – Uso do FPGA.

Nome	Usado	Máximo
Elementos Lógicos	656	6272
Registradores Usados	19	

Fonte: Autoria própria.

4.2.3 PWM

Figura 40 – Representação do bloco PWM

Fonte: Autoria Própria.

O bloco **PWM** gera o sinal PWM para disparar a placa de acionamento do sistema de vazão. As portas de entradas SW, valor transmitido pelo bloco de controle **pid** do tipo STD LOGIC VECTOR de 32 bits, e clk, é o *clock* interno gerado pela placa interna do FPGA, e a saída **bomba** é do tipo STD LOGIC que aciona a placa de acionamento.

```
1     bomba <= aux;
2     chave <= to_integer(unsigned(SW));
3     process (clk) is begin
4        if(clk'event and clk = '1') then
5           counter <= counter + 1;
6           if(counter < chave) then
7              aux    <= '1';
8           elsif(counter > 2047) then
9              counter <= 0;
10          else
11             aux    <= '0';
12          end if;
13       end if;
14    end process;
```

Tabela 7 – Uso do FPGA pelo bloco PWM2.

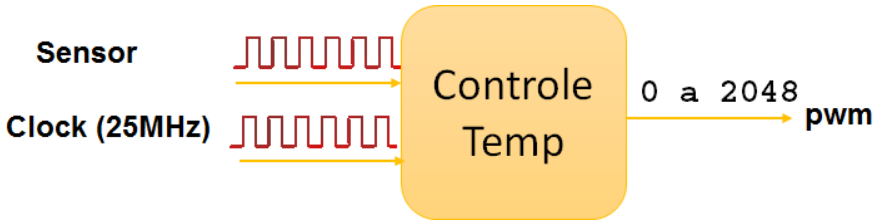
Nome	Usado	Máximo
Elementos Lógicos	22	6272
Registradores Usados	12	

Fonte: Autoria própria.

4.3 TEMPERATURA

4.3.1 controleTemp

Figura 41 – Representação do bloco controleTemp.



Fonte: Autoria Própria.

O bloco **controleTemp** possui 5 portas, sendo 3 pinos de entradas, sensor, entrada do sinal de temperatura fornecido pelo DS18B20 decodificado pelo Arduino, clk, entrada do oscilador fornecido pelo FPGA, e o pwm, gere pulsos para alterar a saída do bloco, sendo 2 pinos de saídas, valor transmitido para o bloco divisorClk para acionar o relé.

A ideia do bloco foi decodificar a informação fornecida pelo Arduino, modulada em PWM, verificar se a temperatura está acima do valor de referência e acionar o bloco PWM caso seja verdade.

A decodificação foi realizada com a ideia de achar o tempo de *Duty Cycle*, que é definido como sendo o período do sinal em alta pelo período total, multiplicado pela resolução da modulação.

```
1   process(clk) is begin
2      if(clk'event and clk = '1') then
3         current <= current+1;
4         if(sensor = '1') then
5            prev <= prev + 1;
6         end if;
7         if(sensor /= saux) then
8            saux  <= sensor;
9            if(sensor = '1') then
10              temp     <= 255*(prev)/current;
11              current  <= 0;
12              prev     <= 0;
13           end if;
14        end if;
15        if(contador < 25_000_000) then
16           contador <= contador + 1;
17        else
18           contador <= 0;
19           if(temp < 55) then
20              saida <= 200;
21           else
22              saida <= 0;
23           end if;
24        end if;
25  end process;
```

Tabela 8 – Uso do FPGA.

Nome	Usado	Máximo
Elementos Lógicos	1311	6272
Registradores Usados	103	

Fonte: Autoria própria.

4.3.2 pwmTemp

O bloco **pwmTemp** é igual ao bloco **PWM2** porem com uma frequência de 1 Hz.

```
1   process(clk) is begin
2      if(clk'event and clk = '1') then
3         if(counter > 3053) then
4            counter  <= 0;
5            aux      <= not aux;
```

Figura 42 – Representação do bloco pwmTemp

Fonte: Autoria Própria.

```
6           else
7              counter <= counter + 1;
8           end if;
9        end if;
10    end process;
11    process(aux) is begin
12       if(aux'event and aux = '1') then
13          contador <= contador + 1;
14          if(contador < pwm) then
15             saida <= '1';
16          else
17             saida <= '0';
18          end if;
19       end if;
20    end process;
```

Tabela 9 – Uso do FPGA pelo bloco pwmTemp.

Nome	Usado	Máximo
Elementos Lógicos	43	6272
Registradores Usados	27	

Fonte: Autoria própria.

4.4 RESULTADO

4.4.1 Vazão

Implementando o código VHDL no FPGA, obteve-se um gasto de 1065 elementos lógicos, 68 registradores e 3 pinos do FPGA, como mostrado na Tabela 10.

Para a validação do projeto proposto, foi realizados diversos testes da bomba com várias referências, para afinar o controle projetado. A fim de averiguar o funcionamento do algoritmo foi realizado testes para os valores de 5 Hz, 10 Hz e 30 Hz.

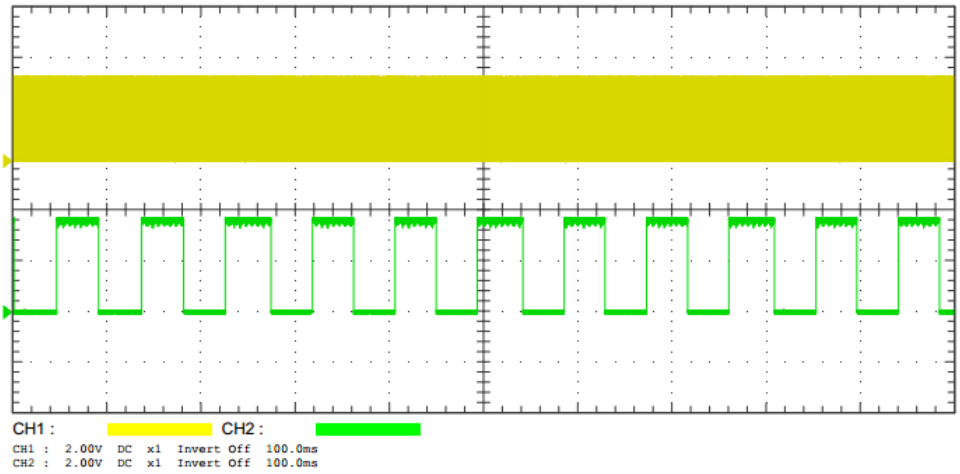
A Figura 43 demonstra o resultado obtido para a referência de 10 Hz no osciloscópio, a Figura 44 é o resposta do manipulado por um *software* matemático para melhor compreensão.

Tabela 10 – Utilização da FPGA.

Material	Quantidade	Total da FPGA
Elementos Lógicos	1065	6272
Registradores	68	
Pinos	3	98

Fonte: Autoria própria.

Figura 43 – Resposta do sistema obtida pelo osciloscópio para uma referência de 10Hz. O sinal em amarelo é a saída PWM do FPGA e o sinal em verde é a saída do sensor de vazão.



Fonte: Autoria própria.

A Figura 45 demonstra o resultado obtido para a referência de 5Hz no osciloscópio, a Figura 46 é o resposta do manipulado por um *software* matemático para melhor compreensão.

A Figura 47 demonstra o resultado obtido para a referência de 30Hz no osciloscópio, a Figura 48 é a resposta manipulada por um *software* matemático para melhor compreensão.

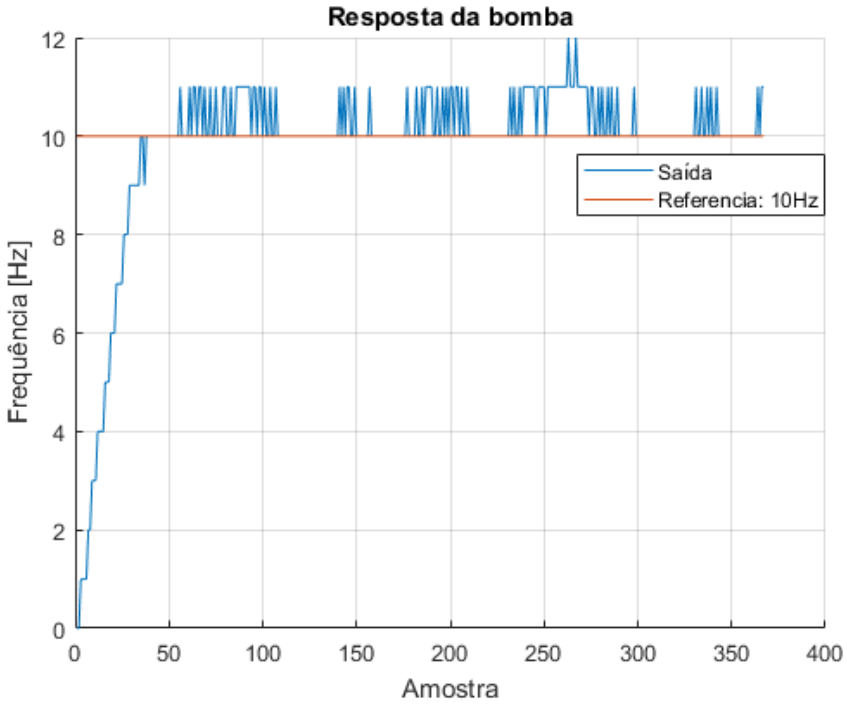
Como o FPGA realiza cálculos inteiros, o resultado da divisão da frequência do *clock* pelo sensor será sempre um número natural, perdendo a informação de frequência entre os inteiros. Como o *Overshoot* projetado era de 20 % do valor estabilizado, ou seja, oscilou no máximo um valor de 1 Hz.

4.4.2 Temperatura

Para realizar o controle de temperatura do sistema, foi utilizado um controlador *ON/OFF*, que liberava a tensão de rede para o aquecedor até que alcançasse a temperatura desejada e desligasse o relé, bloqueando a tensão da rede.

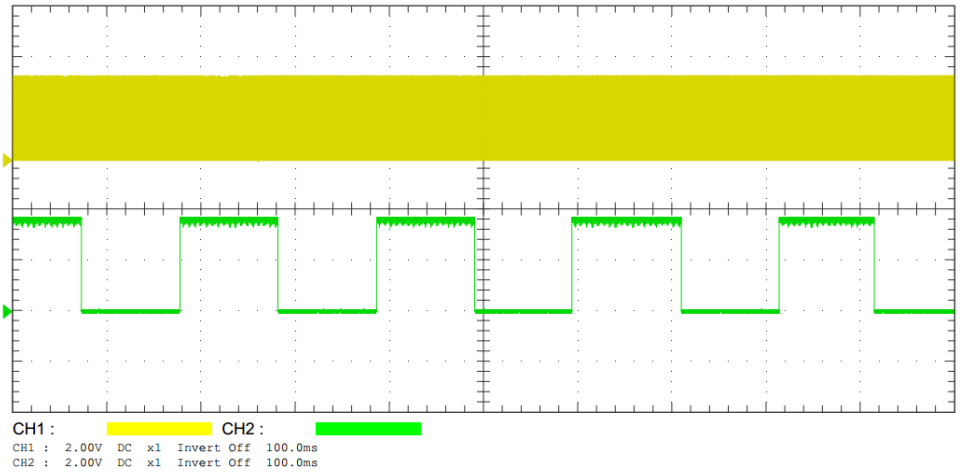
A leitura da temperatura era feito através de um sensor OneWire. Para facilitar a

Figura 44 – Resposta do sistema para uma referência de 10Hz.



Fonte: Autoria própria.

Figura 45 – Resposta do sistema para uma referência de 5Hz obtida pelo osciloscópio, o sinal em amarelo representa a saída PWM do FPGA e o sinal em verde representa o sinal do sensor de vazão.



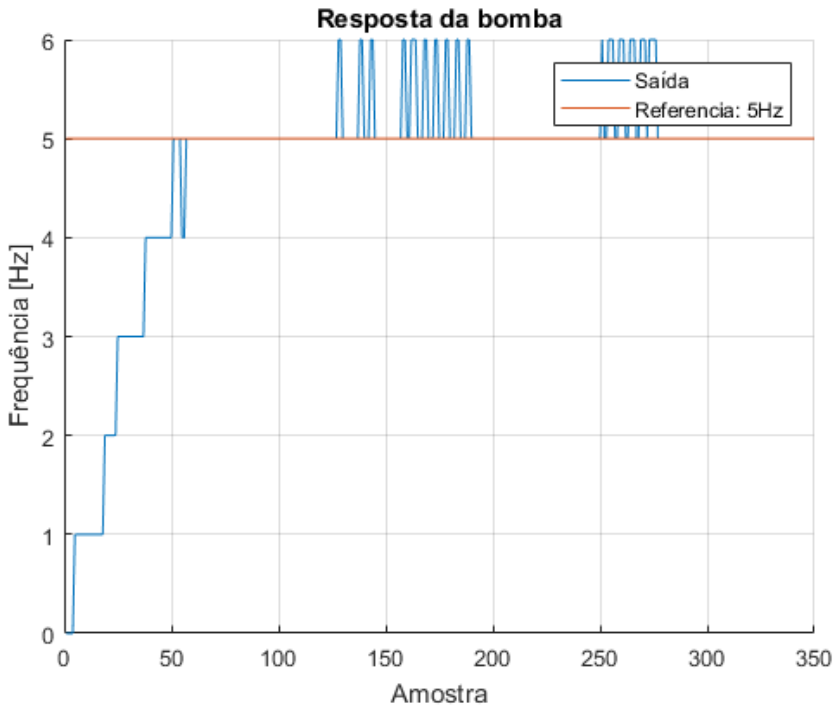
Fonte: Autoria própria.

leitura, foi utilizado um Arduino como decodificador do protocolo e transmitir esse valor para o FPGA.

Para transmitir o dado de temperatura do sensor, usou-se da teoria de transmissão do funcionamento da modulação PWM, na qual é nativa do Arduino, que relaciona a proporção do sinal que mantem em nível lógico alto ao período total da modulação.

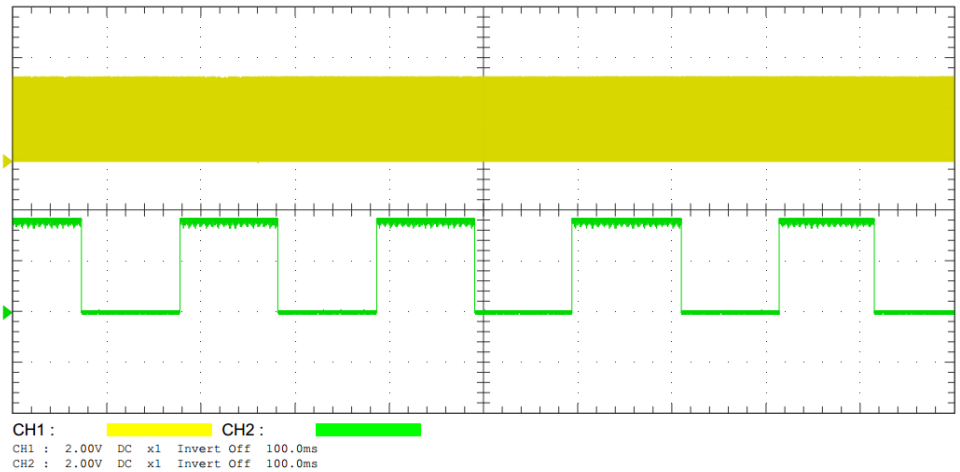
O código para realizar esse cálculo pode ser mostrado a seguir, após achar a

Figura 46 – Resposta do sistema para uma referência de 5Hz.



Fonte: Autoria própria.

Figura 47 – Resposta do sistema para uma referência de 5Hz obtida pelo osciloscópio, o sinal em amarelo representa a saída PWM do FPGA e o sinal em verde representa o sinal do sensor de vazão.

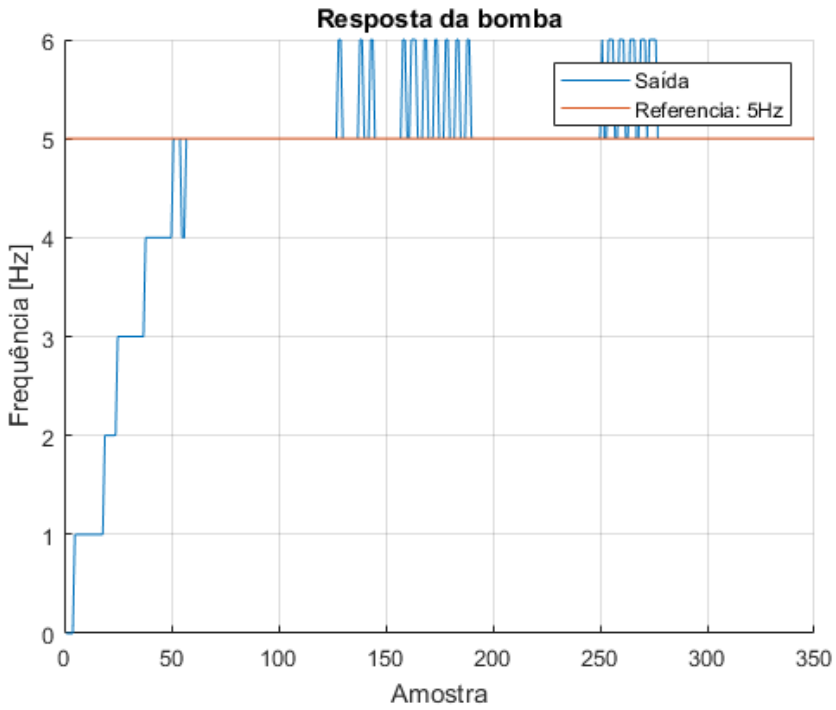


Fonte: Autoria própria.

proporção é multiplicado por 255, que é o valor máximo transmitido pelo valor.

```
1    process(clk'event and clk = '1') is begin
2       current  <= current + 1;
3       if(sensor = '1') then
4          if(sensor /= saux) then
5             if(sensor = '1') then
6                temp  <= 255*prev/current;
7                current  <= 0;
```

Figura 48 – Resposta do sistema para uma referência de 5Hz.

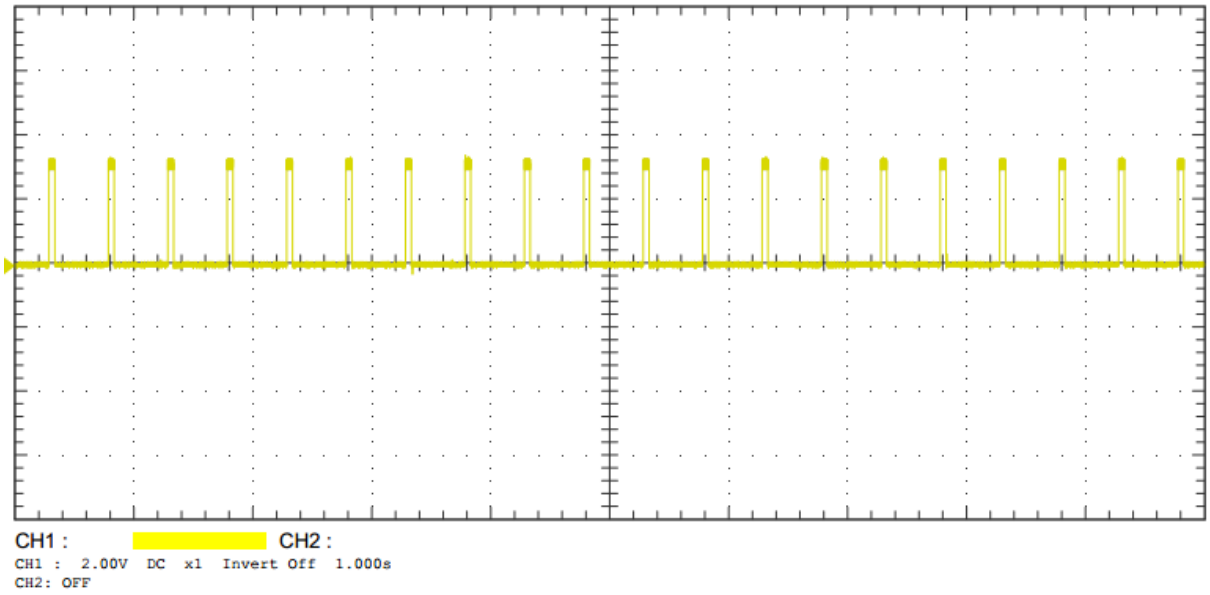


Fonte: Autoria própria.

```
8                prev  <= 0;
9             end if;
10         else
11             prev  <= prev + 1;
12         end if;
13      end if;
14   end process;
```

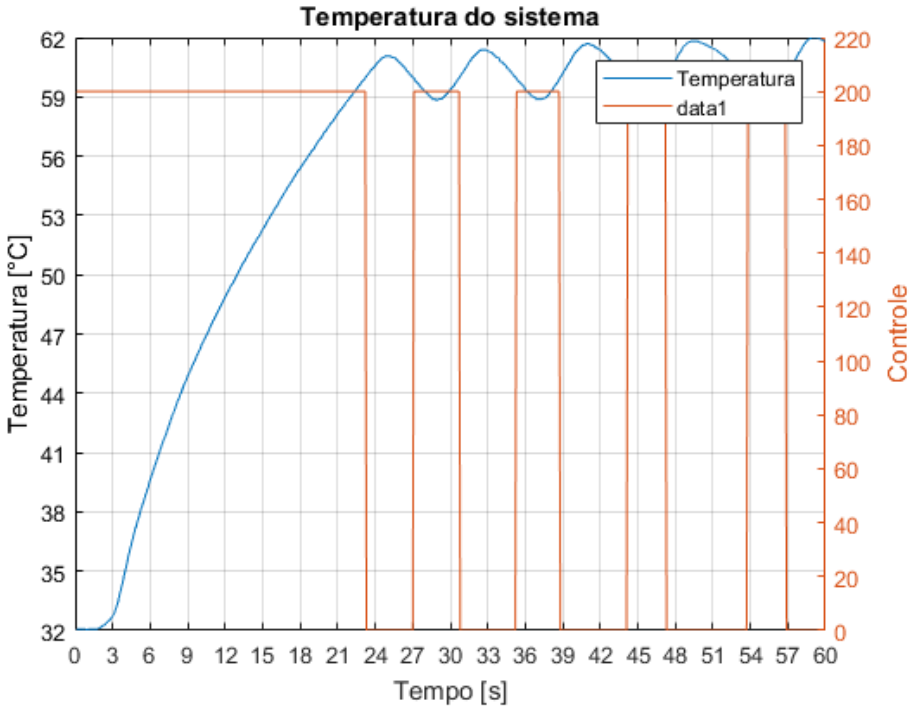
Apesar do controle ON/OFF ser simples e eficaz, ele possui uma grande desvantagem em relação a controle mais complexos, como por exemplo o PID. Quando o controlador envia um sinal baixo para o aquecedor, ele manterá a sua temperatura, que é relativamente maior que a da água. Após um tempo a temperatura do aquecedor se igualará com água, após esse período, como não terá mais o fonte de calor para aquecer a aguá, ela voltará a temperatura ambiente com o passar do tempo, mostrado na Figura 49.

Figura 49 – Saída PWM do FPGA lida no osciloscópio.



Fonte: Autoria própria.

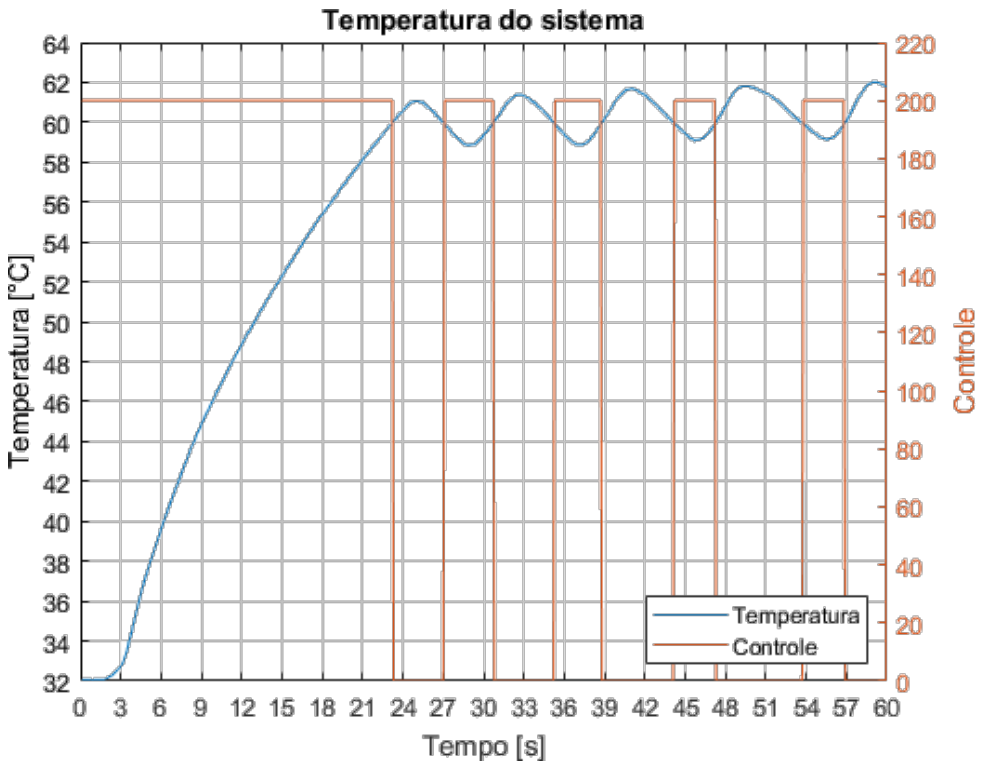
Figura 50 – Resposta do sistema para uma referência de 60 C.



Fonte: Autoria própria.

Esse fenômeno também pode ser notado com outra referência, Figura 51.

Figura 51 – Resposta do sistema para uma referência de 55 C.



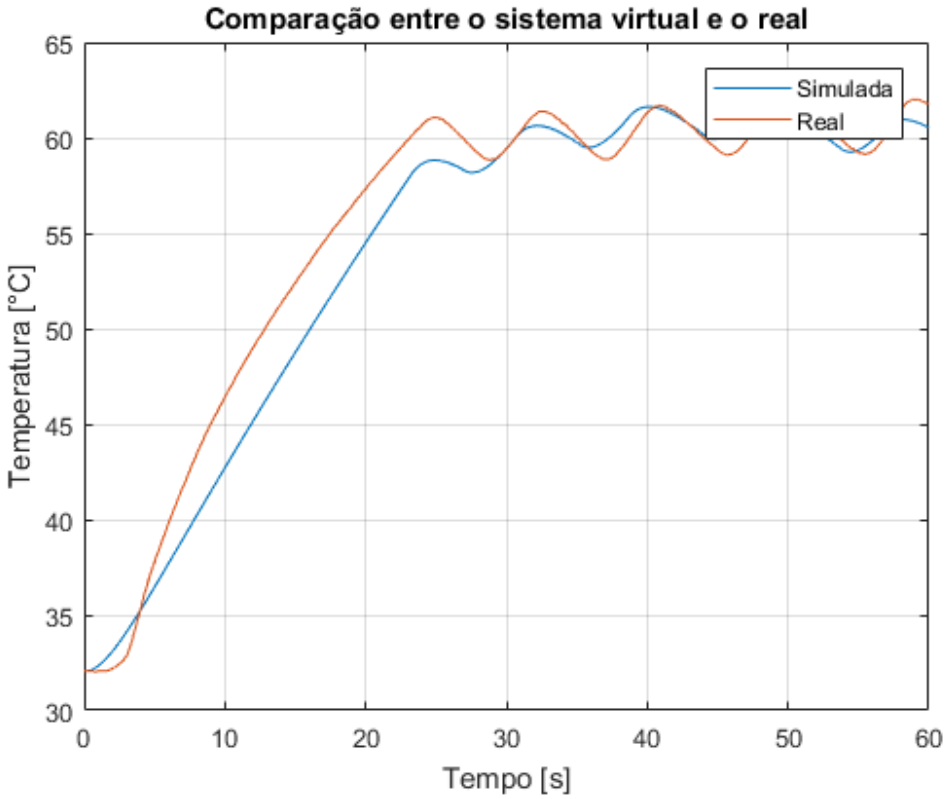
Fonte: Autoria própria.

Com os resultados obtidos, retirou a equação de transferência da planta que se aproximasse do modelo matemático do aquecedor, descrito na equação 3.3, e se comparou o modelo obtido pelo valor real lido da temperatura do aquecedor, mostrado na Figura 52.

$$Aq(z) = \left(\frac{-2.939z^2 + 5.42z + 5.454}{z^3 - 1.326z^2 + 0.5776} \right) 10^{-5} \tag{4.1}$$

..

Figura 52 – Comparação entre o aquecedor real e o virtual.

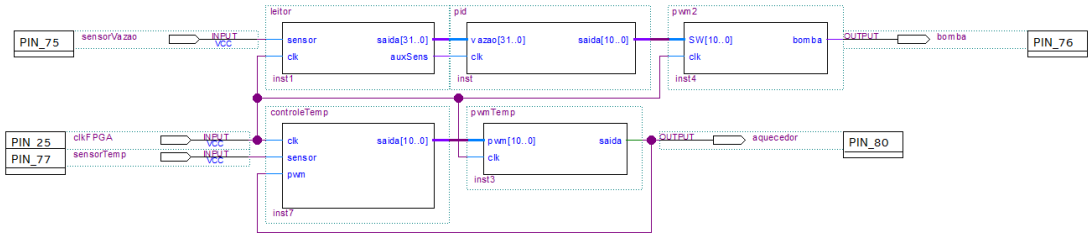


Fonte: Autoria própria.

4.5 RESULTADOS DO FPGA

O design de blocos final do projeto pode ser visto na Figura 53, pode-se notar que há o bloco de leitura do sensor de vazão, controlador e um modulador PWM na parte de vazão, e um bloco de controlador com o leitor do sensor interno e um modulador PWM, na parte de temperatura.

Figura 53 – Design final dos blocos de FPGA.



Fonte: Autoria própria.

Os pinos utilizados no projeto são mostrado na Tabela 11, o número de pinos utilizados no FPGA foram 8 pinos, sendo 3 no controle de vazão (sensor, vazão, bomba), compartilhando o pino do *clock*, e 3 pinos utilizados pelo controle de temperatura (*clock*, sensor, aquecedor), compartilhando o pino de *clock* com o controle de vazão.

Tabela 11 – Pinos no FPGA.

Nome	Tipo	Pino
bomba	Saída	75
clkFPGA	Entrada	25
aquecedor	Saída	80
sensorVazao	Entrada	76
sensorTemp	Entrada	77

Fonte: Autoria própria.

Com relação ao uso da FPGA, foi utilizado ao todo 4052 elementos lógicos, que representou 65 % do total no FPGA e 251 registradores, como descrito na Tabela 12.

Tabela 12 – Uso do FPGA.

Nome	Usado	Máximo
Elementos Lógicos	4052	6272
Registradores Usados	251	

Fonte: Autoria própria.

O desempenho no FPGA da parte prática foi condizente com o esperado da parte teórica, de forma que conseguiu realizar a leitura dos sensores e implementar as funções de controle.

5 CONCLUSÃO

Após a realização dos testes mostrados nos capítulos anteriores, nota-se que os objetivos propostos foram atingidos, tanto no projeto de vazão quanto temperatura. Entretanto, apesar de conseguir atingir a referência desejada, há oscilações em torno da referência.

O atuador de vazão não possuía uma planta conhecida e por isso foi utilizado o método dos mínimos quadrados para a identificação do sistema. A planta obtida pelo algorítimo teve uma excelente aproximação, o que permitiu projetar e implementar o controlador no FPGA.

No controle da vazão, como o valor da frequência de referência é relativamente baixo, no cálculo do FPGA, perde-se informação após a virgula, o que ocasiona oscilações na resposta. Por causa desta perda de dados, o controlador demora mais tempo para atingir a estabilidade. Esse problema poderia ser corrigido pela troca do tipo de sensor de vazão.

No caso da temperatura, o controlador atingiu o desempenho desejado. As oscilações eram esperadas para o tipo de controle aplicado, como explicado no capitulo anterior, no entanto a temperatura não sofreu grande variação, apresentando um *overshoot* de 3,33 %.

Analisando os resultados para o FPGA, observa-se o grande consumo de *hardware* na aplicação, o que representou 65 % do número de elementos lógicos disponíveis. Para tornar o código mais eficiente, poderia utilizar PLL's internos do FPGA para diminuir o uso de elementos lógicos.

Sugestões para trabalhos futuros:

- Projetar um controlador PID para a temperatura em FPGA, afim de diminuir as oscilações entorno da referência.
- Implementar um decodificador do protocolo OneWire em FPGA, para retirar a necessidade de um decodificador.
- Realizar testes com diferentes reagentes, para utilização em diversas plantas.

REFERÊNCIAS

BOSCH, A. **Motores Elétricos**: Inclui motores elétricos, bombas de água e relés. [s.n.], 2013. 7 p. Disponível em: <http://br.bosch-automotive.com/media/parts/download_2/motores_eletricos/Catalogo_Motores_Eletricos_lowres_6_008_CT1_248_2013.pdf>.

BRUNETTI, F. **Mecânica dos fluidos I**. 2 rev.. ed. São Paulo, SP: Pearson Prentice Hall, 2008.

CASSIOLATO, C.; ORELLANA, E. **Medição de vazão**. 2010. Disponível em: <http://www.smar.com/newsletter/marketing/index40.html>.

COSTA, C. da. **Elementos de lógica programável com VHDL e DSP: Teoria e Prática**. 1. ed. São Paulo, SP: Editora Érica, 2011.

CUNHA, M. R. da. **Desenvolvimento de microrreatores em tecnologia LTCC para produção de biodiesel**. Tese ((Doutorado em Microeletrônica) - Escola Politécnica), 2012. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3140/tde-13062013-120728/>.

DOEBELIN, E. O. **Measurement Systems Application and Design**. Fourth edition. [S.l.]: McGraw-Hill Publishing Company, 1990.

FAROOQ, U.; MARRAKCHI, Z.; MEHREZ, H. **Tree-based Heterogeneous FPGA Architectures**: Application specific explotration and optimization. Springer. [S.l.: s.n.], 2012.

FLOYD, T. **Sistemas Digitais: Fundamentos e aplicações.** 9. ed. [S.l.: s.n.], 2007. ISBN 978-85-7780-107-7.

LEONG, W. Y. *et al.* Development of an electronic aerosol system for generating microcapsules. v. 78, n. 5-7, p. 79–85, 2016. ISSN 2180-3722. Disponível em: <http://www.jurnalteknologi.utm.my/index.php/jurnalteknologi/article/download/8718/5181>.

PEDRONI, V. A. **Eletrônica Digital Moderna e VHDL**. [S.l.: s.n.], 2016. 620 p.

SWART, J. W. **Evolução de Microeletrônica a Micro-Sistemas**. 2016. Disponível em: <http://www.ccs.unicamp.br/cursos/fee107/download/cap01.pdf>.

TRONIC, G. **Capteur de débit YFS401 0.3 à 6 l/min**. 2016. Disponível em: <http://www.gotronic.fr/art-capteur-de-debit-yfs401-23851.htm>.

WIKIPéDIA. **CPLD**. 2013. Disponível em: <https://pt.wikipedia.org/wiki/CPLD>.