HELDER ESCOBAR SALES

DESENVOLVIMENTO DE UM DISPOSITIVO ELETRÔNICO EMBARCADO PARA O TRATAMENTO DE INFORMAÇÕES UTILIZADAS NO PROCESSO DE BALANCEAMENTO DINÂMICO DE PEÇAS ROTATIVAS

Curitiba

2020

HELDER ESCOBAR SALES

DESENVOLVIMENTO DE UM DISPOSITIVO ELETRÔNICO EMBARCADO PARA O TRATAMENTO DE INFORMAÇÕES UTILIZADAS NO PROCESSO DE BALANCEAMENTO DINÂMICO DE PEÇAS ROTATIVAS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica, Área de Concentração Eletrônica, Departamento de Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, como parte das exigências para a obtenção do título de Engenheiro Eletricista.

Universidade Federal do Paraná

Departamento de Engenharia Elétrica

Orientador: Prof. Ph.D. André Augusto Mariano

Coorientador: Eng. Ricardo Augusto Borba

Curitiba

2020

HELDER ESCOBAR SALES
DESENVOLVIMENTO DE UM DISPOSITIVO ELETRÔNICO EMBARCADO PARA O TRATAMENTO DE INFORMAÇÕES UTILIZADAS NO PROCESSO DE BALANCEAMENTO DINÂMICO DE PEÇAS ROTATIVAS/ HELDER ESCOBAR SALES. – Curitiba, 2020-
79 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Ph.D. André Augusto Mariano
Coorientador: Eng. Ricardo Augusto Borba

Trabalho de Conclusão de Curso – Universidade Federal do Paraná
Departamento de Engenharia Elétrica, 2020.

1. Aplicativos móveis. 2. Balanceamento de peças rotativas. 3. Desgaste mecânico. 4. Sistema eletrônico embarcado. I. Prof P.hD. André Augusto Mariano. II. Universidade Federal do Paraná. III. Departamento de Engenharia Elétrica. IV. Desenvolvimento de um dispositivo eletrônico embarcado para o tratamento de informações utilizadas no processo de balanceamento dinâmico de peças rotativas

TERMO DE APROVAÇÃO

HELDER ESCOBAR SALES

DESENVOLVIMENTO DE UM DISPOSITIVO ELETRÔNICO EMBARCADO PARA O TRATAMENTO DE INFORMAÇÕES UTILIZADAS NO PROCESSO DE BALANCEAMENTO DINÂMICO DE PEÇAS ROTATIVAS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica, Área de Concentração Eletrônica, Departamento de Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, como parte das exigências para a obtenção do título de Engenheiro Eletricista.

.

Prof. Ph.D. André Augusto Mariano
Orientador

Eng. Ricardo Augusto Borba
Coorientador

Prof. Dr. César Augusto Dartora
Departamento de Engenharia Elétrica – UFPR

Prof. Dr. Rodrigo Jardim Riella
Departamento de Engenharia Elétrica – UFPR

Curitiba
2020

Dedico este trabalho aos meus pais,
à minha irmã,
meus professores
e meus amigos

Agradecimentos

Agradeço a Deus pela vida e criação do nosso fascinante e misterioso Universo.

Aos meu pais Ezequiel e Antonia por terem me proporcionado educação, carinho e amor.

À minha irmã Kelly pelo companheirismo, apoio e carinho.

Ao meu professor orientador André Augusto Mariano pela oportunidade, confiança e conhecimentos transmitidos.

Ao meu coorientador e amigo Ricardo cuja amizade, confiança e companheirismo são inestimáveis.

Aos meus familiares pelo apoio e carinho.

Aos meus amigos pelos bons momentos que passamos juntos na universidade.

Às pessoas do departamento de Engenharia Elétrica da UFPR e a todos os professores que contribuíram para a minha formação como engenheiro eletricista.

"A sabedoria com as coisas da vida não consiste,
ao que me parece, em saber o que é preciso fazer,
mas em saber o que é preciso fazer antes
e o que fazer depois."
(Liev Nikoláievich Tolstói)

Resumo

A necessidade de utilização de sistemas mecânicos rotativos livres de perturbações vibratórias é desejável por vários motivos, entre os quais estão inclusos: o conforto humano, o aumento da vida útil dos equipamentos, a diminuição dos efeitos indesejados sobre outras medidas, a redução das perdas rotacionais em máquinas elétricas, além de outros. Dessa forma, o objetivo deste trabalho é desenvolver um sistema eletrônico embarcado que auxilie no processo de balanceamento de peças rotativas. Assim, projetou-se um protótipo que, através de sensores externos, realiza o processamento de dados para a determinação do ângulo e a massa necessária para a correção do desbalanceamento. Através dele, as variáveis de interesse obtidas são enviadas à uma tela LCD e a um aplicativo de celular utilizando um módulo de rede sem fio. Construiu-se um protótipo, com apenas dois microcontroladores e alguns componentes passivos, além de sensores externos. Através de testes verificou-se que o objetivo proposto foi cumprido com o sistema tendo baixo custo e boa portabilidade, além de apresentar erro médio de medições de ângulo abaixo de 2% em comparação com os medidos no osciloscópio.

Palavras-chave: Aplicativos móveis. Balanceamento de peças rotativas. Desgaste mecânico. Sistema eletrônico embarcado.

Abstract

The need of rotational mechanical systems free of vibratory disturbances are desirable for several reasons, which includes: the human confort, increased lifespan of equipments, reducing undesirable effects over measurements, reduction of rotational losses in electrical machines, and others. Thus, the objective of this work is develop an embedded system that auxiliates in the balancing process of rotational parts. It was designed a embedded system which, using sensors, process data to determine the angle and mass necessary to correct unbalances. After obtaining the variables of interest, a wireless module sends the information to a LCD and a mobile application. It was built a prototype, with only two microcontrollers and a few passive components, besides external sensors. Through tests it was verified that the proposed objective was accomplished, with the system having low cost and good portability, showing less than 2% mean error in angle measurements comparing with a oscilloscope.

Keywords: Balancing rotational parts. Embedded system. Mechanical wear. Mobile apps.

Lista de ilustrações

Figura 1 – Balanceador industrial de rodas 19
Figura 2 – Diagrama de blocos do sistema de balanceamento 22
Figura 3 – Esquemático do circuito amplificador não inversor 24
Figura 4 – Esquemático do circuito seguidor de tensão 25
Figura 5 – Esquemático do circuito somador de tensão 26
Figura 6 – Diferenças entre as topologias de filtros 27
Figura 7 – Execução de tarefas em um *RTOS* 29
Figura 8 – Diagrama de blocos do microcontrolador *STM32G474RE* 33
Figura 9 – Diagrama de blocos do microcontrolador *ESP-32* 36
Figura 10 – Placa de desenvolvimento *NUCLEO-G474RE* 38
Figura 11 – Placa de desenvolvimento *LOLIN-32 Lite* 39
Figura 12 – Disco piezoelétrico 40
Figura 13 – Encapsulamento do sensor *DRV5023* modelo *TO-92 LPG* 41
Figura 14 – Diagrama de blocos funcional do sensor *DRV5023* 41
Figura 15 – Saída do sensor em função da densidade de fluxo magnético 42
Figura 16 – Influência das orientações dos polos magnéticos 42
Figura 17 – Onda senoidal de 20 Hz contendo 12 harmônicas 44
Figura 18 – Resposta em frequência da amplitude do sinal na saída do filtro 45
Figura 19 – Resposta em frequência da fase do sinal na saída do filtro 46
Figura 20 – Sinal de entrada com ruído e sinal de saída filtrado 47
Figura 21 – Roda de desbalanceamento impresso através de impressora *3D* 48
Figura 22 – Protótipo do sistema eletrônico 49
Figura 23 – Tela inicial do aplicativo *Balanceador* no dispositivo virtual *Pixel 2* . . 52
Figura 24 – Requisição da chave pública pelo aplicativo do celular 53
Figura 25 – Envio da chave pública pelo microcontrolador *ESP-32* 54
Figura 26 – Envio da chave privada *AES* pelo aplicativo do celular 55
Figura 27 – Envio dos dados de roteador criptografados pelo aplicativo do celular . 55
Figura 28 – Envio de datagramas *UDP* pelo microcontrolador 56
Figura 29 – Aplicativo indicando conexão com o *ESP-32* 57
Figura 30 – Onda senoidal com ruído na saída do gerador de sinal 58
Figura 31 – Onda quadrada na saída do gerador de sinal 58
Figura 32 – Onda senoidal de 20 *Hz* contendo 10 harmônicas de diferentes amplitudes 59
Figura 33 – Onda senoidal simulando vibração e onda quadrada simulando posição 60
Figura 34 – Protótipo montado 61
Figura 35 – Tela inicial do *Método ref. não balanceada* 62
Figura 36 – *Método ref. não balanceada* após três medições 63

Figura 37 – Resultado após a quarta medição do *Método ref. não balanceada* 64
Figura 38 – Resultado final do *Método ref. não balanceada* 65
Figura 39 – Calibração do ângulo 66
Figura 40 – Janela de calibração 67
Figura 41 – Ângulo e massa de correção do desbalanceamento 68
Figura 42 – Display *LCD* com o resultado da calibração utilizando somente um parafuso posicionado a 0° da origem 69
Figura 43 – Sistema rotativo balanceado 70
Figura 44 – Criptografia *AES* modo *CBC* 77

Lista de quadros

Quadro 1 – Diferenças entre os filtros tipo *FIR* e *IIR* 27
Quadro 2 – Diferenças entre as topologias de filtros 27

Lista de tabelas

Tabela 1 – Tabela de conversão para *Base64* 31
Tabela 2 – Componentes da senoide simulada 44
Tabela 3 – Características do filtro projetado 46
Tabela 4 – Características dos sinais simulados 47
Tabela 5 – Custos aproximados dos componentes utilizados no projeto 49
Tabela 6 – Características dos sinais senoidais com ruído e filtrados 59
Tabela 7 – Diferença entre os ângulos medidos no osciloscópio e no microcontrolador 60

Lista de abreviaturas e siglas

ADC	*Analog-to-Digital Converter*
AES	*Advanced Encryption Standard*
AP	*Access Point*
API	*Application Programming Interface*
CBC	*Cipher Block Chaining*
CORDIC	*COordinate Rotation DIgital Computer*
DAC	*Digital-to-Analog Converter*
DMA	*Direct Memory Access*
ECC	*Error Correcting Code*
ESP-IDF	*Espressif IoT Development Framework*
FIR	*Finite Impulse Response*
FMAC	*Filter Mathematical Accelerator*
FPU	*Floating Point Unit*
GPIO	*General Purpose Input/Output*
IC	*Input Capture*
IDE	*Integrated Development Environment*
IIR	*Infinite Impulse Response*
IoT	*Internet of Things*
IP	*Internet Protocol*
IV	*Initialization Vector*
JTAG	*Joint Action Test Group*
LCD	*Liquid Crystal Display*
MIT	*Massachusetts Institute of Technology*

MSPS	*Mega Samples Per Second*
PCROP	*Proprietary Code Readout Protection*
PGA	*Programmable Gain Amplifier*
PLA	*Polylactic Acid*
RMS	*Root Mean Square*
ROM	*Read Only Memory*
RPM	Rotações Por Minuto
RSA	*Rivest-Shamir-Adleman*
RTOS	*Real Time Operational System*
SHA	*Secure Hash Algorithm*
SMD	*Surface Mount Device*
SPI	*Serial Peripheral Interface*
SPIFFS	*SPI Flash File System*
SRAM	*Static Random Access Memory*
STA	*Station*
TCP	*Transport Layer Protocol*
UART	*Universal Asynchronous Receiver/Transmitter*
UDP	*User Datagram Protocol*
USART	*Universal Synchronous/Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*

Lista de símbolos

$+VCC$	Tensão de alimentação positiva
$-VCC$	Tensão de alimentação negativa
B_{OP}	*Operate point*
B_{RP}	*Release point*
B_{hys}	*Hysteresis*; $B_{hys} = (B_{OP} - B_{RP})$

Sumário

1 INTRODUÇÃO . . . 19

2 FUNDAMENTAÇÃO TEÓRICA . . . 22
2.1 Visão geral . . . 22
2.1.1 Bloco físico . . . 23
2.1.2 Bloco de processamento analógico-digital . . . 23
2.1.3 Bloco de comunicação . . . 23
2.2 Amplificador operacional . . . 23
2.2.1 Amplificador não inversor . . . 24
2.2.2 Seguidor de tensão . . . 24
2.2.3 Somador de tensão . . . 25
2.3 Filtros digitais . . . 26
2.4 *RTOS* . . . 28
2.4.1 *FreeRTOS* . . . 28
2.4.2 Tarefas . . . 28
2.4.3 Escalonamento . . . 29
2.4.4 Semáforos . . . 29
2.4.4.1 *Mutex* . . . 30
2.5 Codificação *Base64* . . . 30
2.6 Protocolos de comunicação . . . 31
2.6.1 *Sockets* . . . 31
2.6.1.1 *TCP* . . . 31
2.6.1.2 *UDP* . . . 31
2.7 *STMicroelectronics STM32G474RE* . . . 32
2.7.1 Memória . . . 34
2.7.2 *Timers* . . . 34
2.7.3 Periféricos analógicos . . . 34
2.7.3.1 *ADC* . . . 34
2.7.3.2 *DAC* . . . 35
2.7.3.3 Amplificadores operacionais e comparadores . . . 35
2.7.4 Periféricos de comunicação . . . 35
2.8 *Espressif Systems ESP-32* . . . 35
2.8.1 Memória . . . 35
2.8.2 Aceleradores de criptografia . . . 35

3 FERRAMENTAS E PROCEDIMENTOS . . . 37

3.1 Placas de desenvolvimento **37**
3.1.1 *NUCLEO-G474RE* 37
3.1.2 *LOLIN-32 Lite* 37
3.2 Sensores **40**
3.2.1 Disco Piezoelétrico 40
3.2.2 Sensor *Hall DRV5023* 40
3.3 *Sofwares* de desenvolvimento, simulação e modelagem **43**
3.3.1 Plataformas de desenvolvimento para microcontroladores 43
3.3.2 Plataforma de desenvolvimento de aplicativos móveis 43
3.3.3 Software de simulação e cálculo númerico MATLAB 43
3.3.4 *Software* de modelagem de peças *SolidWorks* 47
3.4 Funcionamento do sistema de balanceamento **48**
3.4.1 Ângulo e intensidade de um desbalanceamento 48
3.4.2 Descrição geral do sistema implementado 48
3.4.3 Métodos de balanceamento 50
3.4.3.1 Balanceamento de peças rotativas sem uma peça padrão balanceada de comparação 50
3.4.3.2 Balanceamento de peças rotativas com uma peça padrão balanceada de comparação 51

4 TESTES **52**
4.1 Testes de comunicação **53**
4.2 Testes de funcionalidade com gerador de sinal **57**
4.3 Testes com ventilador de mesa **60**
4.3.1 Balanceamento de peças rotativas sem uma peça padrão balanceada de comparação 61
4.3.2 Balanceamento de peças rotativas com uma peça padrão balanceada de comparação 65

5 CONCLUSÃO **71**
5.1 Possibilidade de melhorias futuras **71**

REFERÊNCIAS **72**

APÊNDICES **74**

APÊNDICE A – CONCEITOS REFERENTES AOS SENSORES DE VIBRAÇÃO E POSIÇÃO **75**
A.1 Piezoeletricidade **75**
A.2 Efeito Hall **75**

APÊNDICE B – CRIPTOGRAFIAS **76**

B.1 ***AES*** **76**
B.1.1 *AES* modo *CBC* 76
B.2 ***RSA*** **77**

APÊNDICE C – DETALHAMENTO TÉCNICO DA RODA DE DESBALANCEAMENTO **78**

1 Introdução

A necessidade de utilização de sistemas mecânicos rotativos livres de perturbações vibratórias é desejável por vários motivos, entre os quais estão inclusos: o conforto humano, o aumento da vida útil dos equipamentos, a diminuição dos efeitos indesejados sobre outras medidas, a redução das perdas rotacionais em máquinas elétricas, além de outros.

Com o objetivo de minimizar as perturbações decorrente das vibrações mecânicas de rotação, são utilizados equipamentos eletromecânicos e circuitos eletrônicos para processar as informações obtidas possibilitando fazer correções de desbalanceamentos.

A maioria dos equipamentos que realizam medições de desbalanceamento são de uso industrial, com grandes dimensões, alto custo e são projetados para usos específicos. Um exemplo de balanceador usado para o balanceamento de rodas está apresentado na Figura 1.

Figura 1 – Balanceador industrial de rodas

Fonte: Adaptado de Alibaba (2020).

Desta forma, deseja-se projetar um equipamento com baixo custo e dimensões reduzidas, aumentando a portabilidade, além de oferecer uma maior flexibilidade de operação, a fim de se abranger uma vasta gama de aplicações.

O conjunto é baseado em:

- Bloco físico, formado por sensores de posição e vibração;
- Bloco de processamento analógico-digital, constituído de um circuito que possui um microcontrolador para processar os dados através de um programa de filtragem digital, amplificação e comparação, o qual fornecerá os ângulos e variáveis de interesse;
- Bloco de comunicação, que é formado por um dispositivo de rede sem fio, que processa as informações e as envia a uma tela *LCD* e a um aplicativo de celular baseado no sistema operacional *Android*.

Objetivos

O objetivo geral deste trabalho é desenvolver uma solução de eletrônica embarcada para aplicações em sistemas rotativos de modo a auxiliar no balanceamento dinâmico das partes girantes, utilizando interfaces convencionais *LCD* e integração com redes *IoT*.

Para simplificar o desenvolvimento do projeto, limitou-se a faixa de aplicação para sistemas girantes que possuem rotações em torno de 1400 *RPM*.

Os objetivos específicos são:

- Definir um microcontrolador adequado para o tratamento dos sinais presentes no sistema;
- Selecionar sensores para a captação de vibrações decorrentes do desbalanceamento;
- Utilizar dispositivos para interface sem fio do dispositivo, além de mostradores LCD para caso o usuário não possua acesso uma rede sem fio;
- Realizar o tratamento dos sinais em um protótipo;
- Montar uma interface para o usuário.

Organização do trabalho de conclusão de curso

Este trabalho se divide em cinco capítulos, sendo o primeiro a introdução.

O segundo capítulo aborda os principais conceitos envolvidos para a realização do trabalho.

O terceiro capítulo apresenta os procedimentos efetuados e quais ferramentas foram utilizadas para a obtenção dos resultados a serem atingidos.

O quarto capítulo aborda os testes envolvendo alguns dos cenários possíveis, de modo a serem analisados a funcionalidade do protótipo e da interface com o usuário além de expor os resultados e discussões acerca destes.

O último capítulo é destinado às conclusões do trabalho desenvolvido.

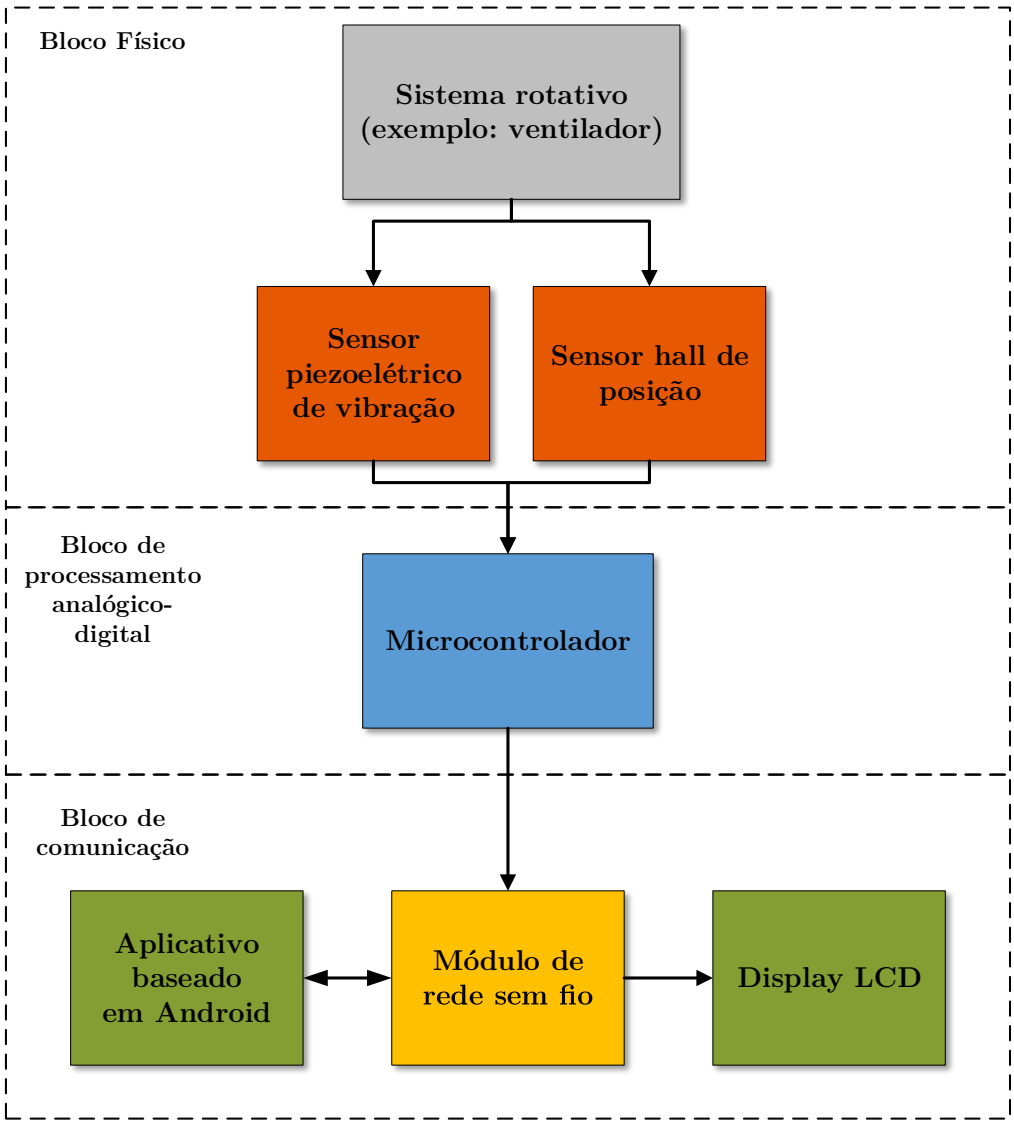
2 Fundamentação Teórica

Neste capítulo será descrito o funcionamento geral do sistema, os conceitos envolvidos e apresentar um detalhamento dos microcontroladores a serem utilizados.

2.1 Visão geral

Nesta seção será apresentada a visão geral do projeto separado em blocos. Um diagrama mostrando de forma resumida o projeto está apresentado na Figura 2.

Figura 2 – Diagrama de blocos do sistema de balanceamento



Fonte: O autor (2020).

2.1.1 Bloco físico

O bloco físico é composto de um sistema rotativo, que podem ser partes de motores, peças de eletrodomésticos, hélices de *drones*, entre outros.

Quando o sistema não está balanceado são geradas perturbações que resultam em vibrações e desgastes mecânicos das peças.

No caso do presente trabalho, utilizou-se um ventilador com uma hélice desbalanceada. Quando o ventilador é ligado, a hélice gira e o desbalanceamento faz com que o aparelho vibre.

São utilizados sensores do tipo *switch* unipolar *Hall* para medições de posicionamento e sensores discoidais piezoelétricos para a captação da vibração, que auxiliam na caracterização do desbalanceamento transformando as grandezas físicas em grandezas elétricas.

Um detalhamento dos conceitos que englobam os sensores estão apresentados no Apêndice A.

2.1.2 Bloco de processamento analógico-digital

O objetivo do bloco de processamento analógico-digital é manipular os sinais elétricos decorrentes das medições de vibrações e realizar o processamento de forma a estabelecer os valores das grandezas necessárias para a correção do desbalanceamento.

É utilizado um microcontrolador *STM32G474RE* que contém periféricos analógicos necessários para a manipulação dos sinais elétricos.

2.1.3 Bloco de comunicação

O bloco de comunicação tem a função de estabelecer um meio de entrada e saída de informações. É composto por um módulo de rede sem fio *ESP-32* que direciona as informações processadas no bloco da subseção 2.1.2 para uma tela *LCD* e para um aplicativo de celular, de modo a prover uma interface entre o usuário e o sistema de correção de desbalanceamento.

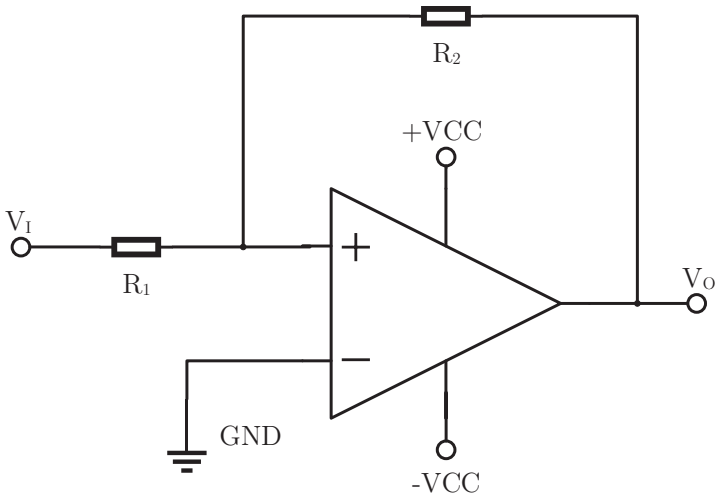
2.2 Amplificador operacional

Nesta seção serão apresentados os modelos de amplificadores utilizados neste trabalho.

2.2.1 Amplificador não inversor

A configuração de um amplificador operacional no modo não inversor está apresentado na Figura 3.

Figura 3 – Esquemático do circuito amplificador não inversor



Fonte: O autor (2020).

Onde:

- V_i é a tensão de entrada;
- V_o é a tensão de saída;
- R_1 é o resistor de entrada;
- R_2 é o resistor de realimentação.

Um amplificador não inversor tem como finalidade amplificar os sinais de um circuito, sem que ocorra a inversão da fase. Idealmente, apresenta impedância infinita na entrada e nula na saída. A Equação 2.1 determina o ganho de malha fechada do amplificador (V_O/V_I), através das resistências de realimentação positiva R_1 e R_2 (SEDRA; SMITH, 2012).

$$\frac{V_O}{V_I} = 1 + \frac{R_2}{R_1} \tag{2.1}$$

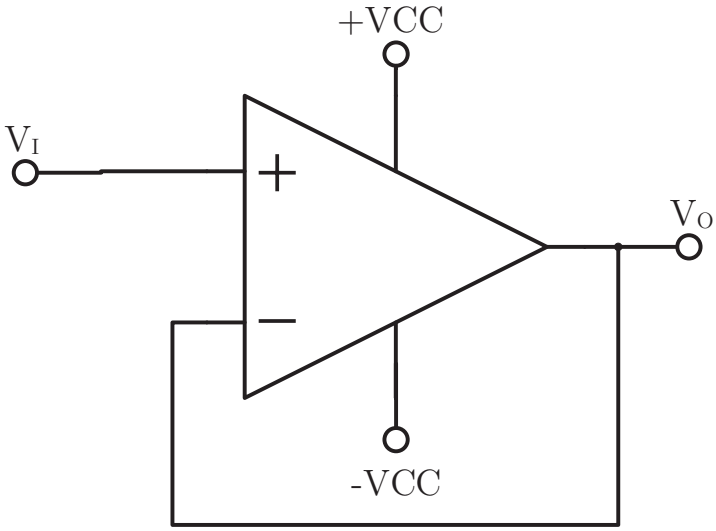
2.2.2 Seguidor de tensão

Um seguidor de tensão, também chamado *buffer*, serve para o isolamento do circuito sob o ponto de vista da impedância. Consiste num amplificador não inversor de ganho

unitário ($V_O = V_I$), e algumas aplicações desta topologia envolvem a transformação de impedâncias ou a amplificação de potência (SEDRA; SMITH, 2012).

O esquemático de um seguidor de tensão está apresentado na Figura 4.

Figura 4 – Esquemático do circuito seguidor de tensão



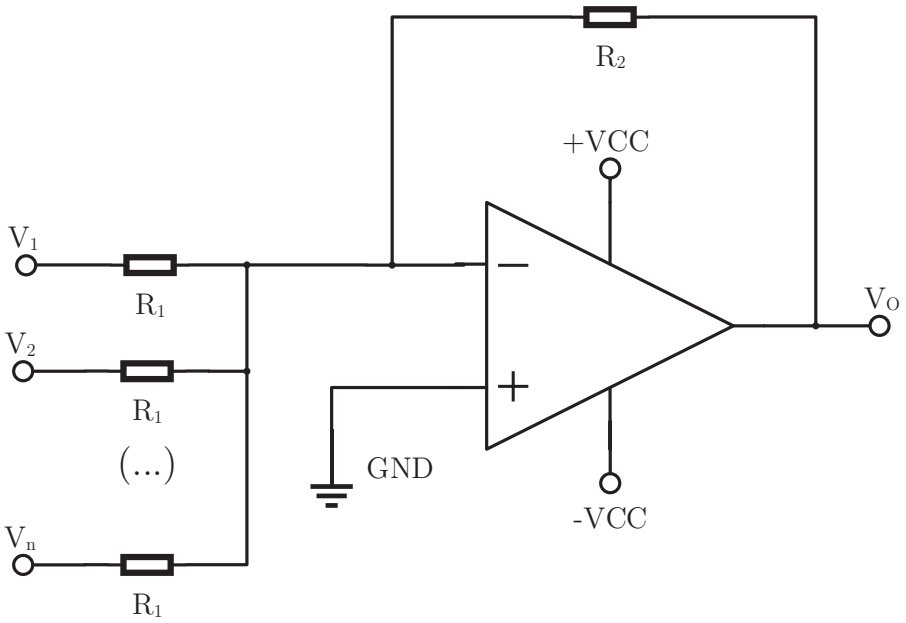
Fonte: O autor (2020).

2.2.3 Somador de tensão

Um somador de tensão é um circuito que contém um amplificador operacional e componentes passivos de maneira a ter-se na saída a soma algébrica das tensões de entrada (V_1, V_2, $\cdots$, V_n) (SEDRA; SMITH, 2012). O modelo matemático que descreve o comportamento do somador de tensão está demonstrado na Equação 2.2 e a representação do circuito está mostrado na Figura 5.

$$V_O = \frac{R_2}{R_1} \times (V_1 + V_2 + ... + V_n) \tag{2.2}$$

Figura 5 – Esquemático do circuito somador de tensão



Fonte: O autor (2020).

2.3 Filtros digitais

Filtros digitais são implementações de operações matemáticas aplicado a um sinal amostrado, a fim de ter o mesmo efeito que se teria em um filtro analógico equivalente.

A utilização de filtros digitais tem como vantagem a maior simplicidade do circuito, às custas de um tempo de processamento via *software* pelo dispositivo.

Os filtros digitais podem ser classificados de acordo com sua resposta ao impulso (BARBOSA, 2006):

- *FIR*, cuja estrutura é baseada em equações lineares com coeficientes não-recursivos;
- *IIR*, cuja estrutura é baseada em equações lineares com coeficientes recursivos.

A utilização de um ou de outro depende de fatores como velocidade, variação da fase para diferentes frequências e a estabilidade (SIEMENS, 2020). O Quadro 1 mostra as diferenças entre os tipos de filtro digital.

Quadro 1 – Diferenças entre os filtros tipo *FIR* e *IIR*

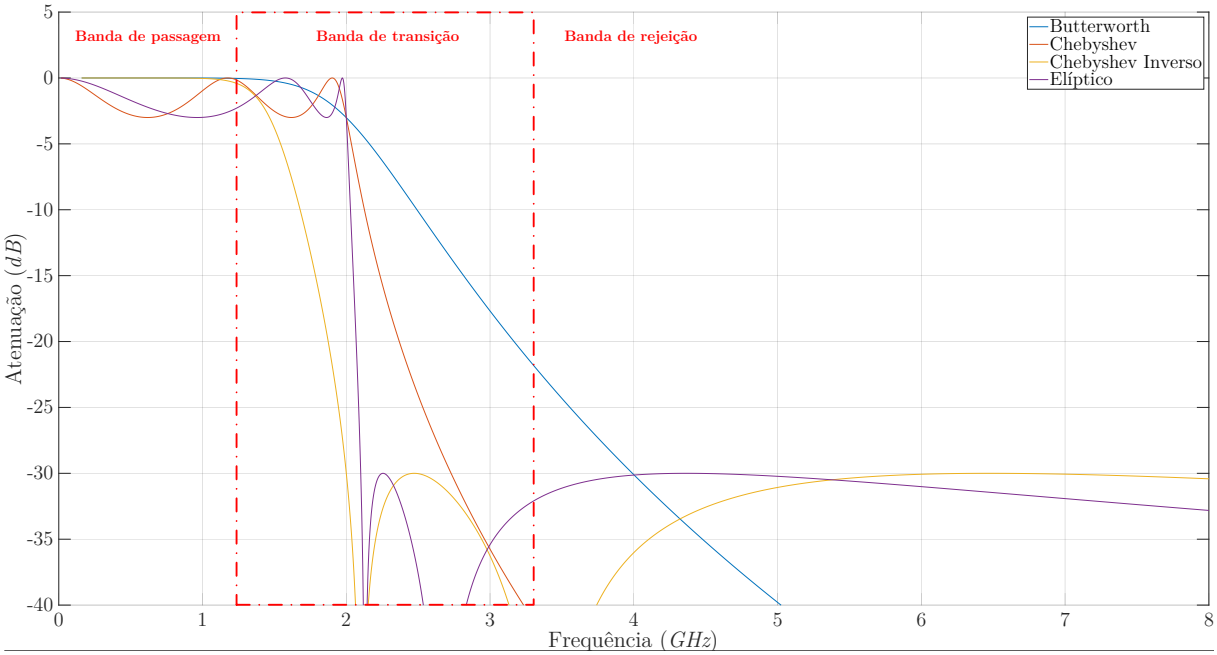
Tipo de filtro	Velocidade Computacional	Fase/Atraso	Estabilidade
FIR	Lento	Constante	Sempre estável
IIR	Rápido	Não constante	Nem sempre estável

Fonte: O autor (2020).

Os filtros digitais podem ter diferentes tipos de resposta de acordo com suas topologias. As mais comuns incluem: *butterworth*, *elíptico*, *chebyshev* e *chebyshev* inverso (SIEMENS, 2020).

A Figura 6 e o Quadro 2 mostram as diferenças entre os filtros.

Figura 6 – Diferenças entre as topologias de filtros



Fonte: O autor (2020).

Quadro 2 – Diferenças entre as topologias de filtros

Topologia	Banda de passagem	Banda de transição	Banda de rejeição
Butterworth	Plana	Larga	Monotônica
Chebyshev inverso	Plana	Estreita	Com *ripple*
Chebyshev	Com *ripple*	Estreita	Monotônica
Elíptico	Com *ripple*	Muito estreita	Com *ripple*

Fonte: O autor (2020).

2.4 *RTOS*

Sistemas operacionais de tempo real, ou *RTOS*, são sistemas que possuem a finalidade de facilitar ao programador a criação de *softwares* multitarefa que precisam ser executadas em um tempo pré-definido, ou seja, possui características determinísticas.

As explicações das seções seguintes têm como referência o manual de Services (2017) e (BARRY, 2016).

2.4.1 *FreeRTOS*

O *FreeRTOS* é um tipo de *RTOS* com foco em sistemas embarcados que possui licença open source *MIT*, não requerendo que o usuário exponha sua propriedade intelectual ao usá-lo. É destinado à sistemas embarcados que possuem recursos de memória e processamento restritos, com algumas das principais características listadas a seguir:

- Prover uma solução única e independente para várias arquiteturas e ambientes de desenvolvimento;
- Possuir uma implementação compacta, normalmente ocupa entre 6 e 12 *Kbytes* de memória;
- Facilitar a implementação nos microcontroladores, com exemplos pré-configurados para cada arquitetura;

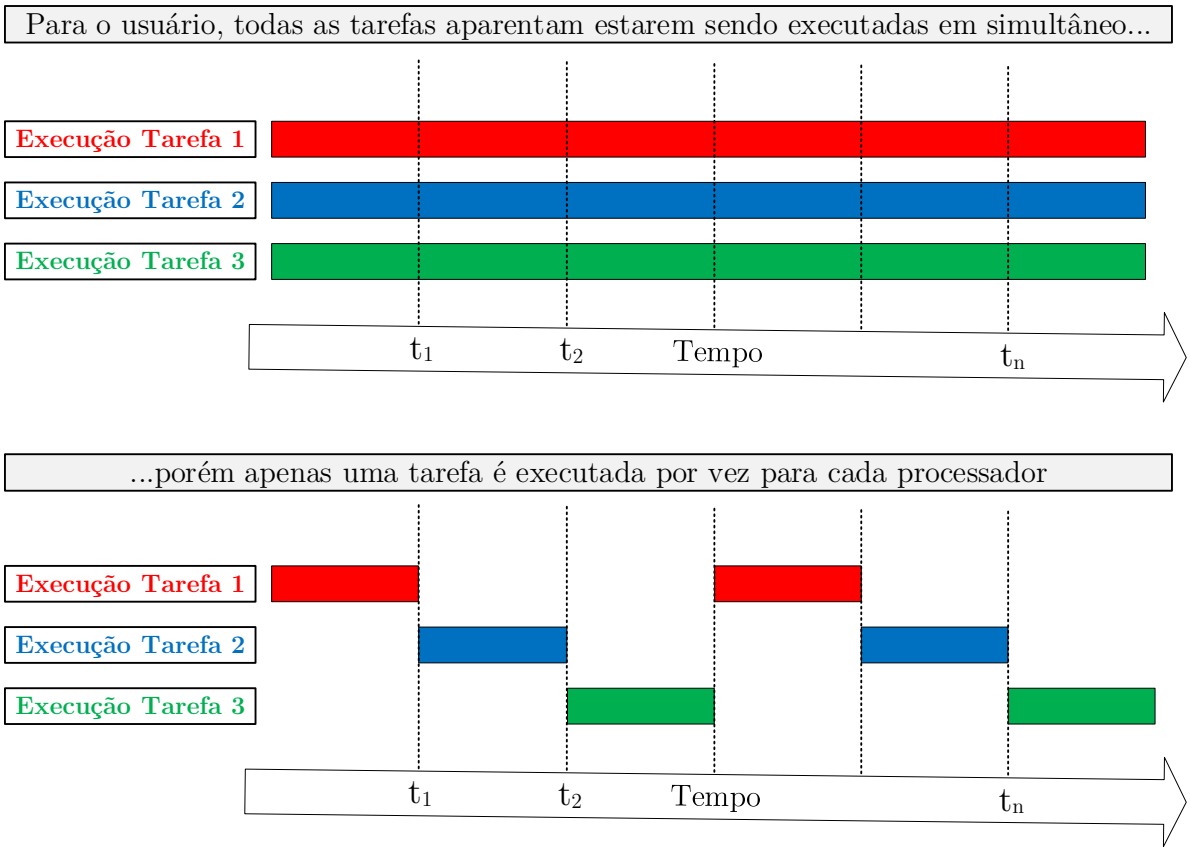
Além disso, apresenta variantes com licenças e suporte comerciais tais como *OPENRTOS* e *SAFERTOS*, sendo este último destinado à aplicações onde a segurança e confiabilidade são imprescindíveis. O núcleo do sistema operacional é chamado de *kernel*.

2.4.2 Tarefas

Cada programa executado é chamado de *task* (tarefa), ou *thread* em algumas implementações. É possível gerenciar várias tarefas, a fim de se facilitar programações mais complexas.

Um processador convencional normalmente executa apenas uma tarefa por vez, entretanto, quando rapidamente alterna a execução entre diferentes tarefas, um sistema operacional pode passar a impressão de estar executando todas simultaneamente. O diagrama que apresenta este fenômeno está explicitado na Figura 7.

Figura 7 – Execução de tarefas em um *RTOS*



Fonte: O autor (2020).

2.4.3 Escalonamento

O escalonador é a parte do *kernel* responsável por decidir qual tarefa deve ser executada em um tempo específico. O *kernel* pode suspender e depois continuar uma tarefa várias vezes durante o ciclo de vida desta. Além disso, a própria tarefa pode decidir pela suspensão dela mesma, seja para gerar um atraso (*sleep*) ou para bloquear o estado até que um recurso esteja disponível ou ocorra algum evento.

2.4.4 Semáforos

Um semáforo é um recurso utilizado para controlar acesso à recursos ou realizar o sincronismo de tarefas. Consiste num objeto do *kernel* que possui uma identificação única, um valor binário associado e uma lista de tarefas em espera. Uma analogia deste recurso seria a de uma chave em que permite a tarefa executar alguma operação ou acessar algum recurso.

O semáforo é inicializado com valor maior ou igual a um, e cada tarefa que faz o uso do semáforo decrementa seu valor até que se chegue em zero. Quando o valor do semáforo é zero, todas as tarefas que tentarem fazer uso deste serão bloqueadas até que

o valor seja novamente não nulo. Cabe então às tarefas que fizeram o uso do semáforo retorná-lo à condição em que se encontrava anteriormente.

2.4.4.1 *Mutex*

Mutex (*mutually exclusive*) é um tipo especial de semáforo, utilizado para controlar o acesso à uma variável que é compartilhada entre várias tarefas. Possuem apenas dois estados, disponível e indisponível. Quando uma tarefa precisa acessar um recurso que potencialmente possa ser modificado por outras tarefas enquanto esta faz operações, usa-se o *mutex*, fazendo com que as demais tarefas permaneçam bloqueadas até a liberação do recurso.

2.5 Codificação *Base64*

A codificação *Base64* tem como propósito a transmissão de dados de texto em meios onde os valores de *bytes* originais são importantes. Como diferentes sistemas podem ter diferentes codificações para o tratamento de textos, estes podem alterar os *bytes* originais recebidos de uma cadeia de caracteres (TOOLKITS, 2020).

Para a codificação de um texto para *Base64*, utiliza-se o seguinte algoritmo (GURU, 2020):

1. Obtém-se a cadeia de *bytes* de informação em uma única longa sequência de *bits*;
2. Divide-se da esquerda para a direita os números binários em grupos de seis *bits*, preenchendo com zeros no final, se necessário;
3. O valor em decimal dos grupos de *bits* representa o índice da *Base64*, o qual é convertido para o caractere equivalente de acordo com a Tabela 1;
4. Se a informação a ser codificada não for múltiplo de 3 *bytes*, adicionam-se caracteres especiais "=" ao fim da codificação, um para cada *byte* faltante de modo que fique divisível por três.

Tabela 1 – Tabela de conversão para *Base64*

Índice	Caractere Base64	Índice	Caractere Base64	Índice	Caractere Base64	Índice	Caractere Base64
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Fonte: O autor (2020).

Para a decodificação executa-se os passos descritos na ordem inversa.

2.6 Protocolos de comunicação

Nesta seção serão apresentados os protocolos de comunicação utilizados para a o envio e recebimento de informações.

2.6.1 *Sockets*

A maioria das aplicações de rede envolvem processos que se comunicam entre si, normalmente em pares. Os processos enviam e recebem mensagens da rede através de interfaces, denominadas de *sockets* (KUROSE; ROSS, 2014).

2.6.1.1 *TCP*

O *TCP* é um protocolo de transporte orientado à conexão, o qual é utilizado para a transferência de dados de uma forma confiável. Possui mecanismos para o recebimento de dados de forma ordenada e sem erros, além de possuir um controle de congestionamento para a manutenção da qualidade da *Internet* (KUROSE; ROSS, 2014).

2.6.1.2 *UDP*

O *UDP* é um protocolo de transporte não orientado à conexão, utilizado para a transferência de dados em que a confiabilidade não é um fator importante. Por não possuir

mecanismos de controle de erros e confirmação de recepção, é um protocolo bastante simples e leve (KUROSE; ROSS, 2014), podendo ser usado em situações como chamadas de voz ou videoconferências onde a transmissão em tempo real é mais importante que a garantia da recepção de todos os dados e de forma correta.

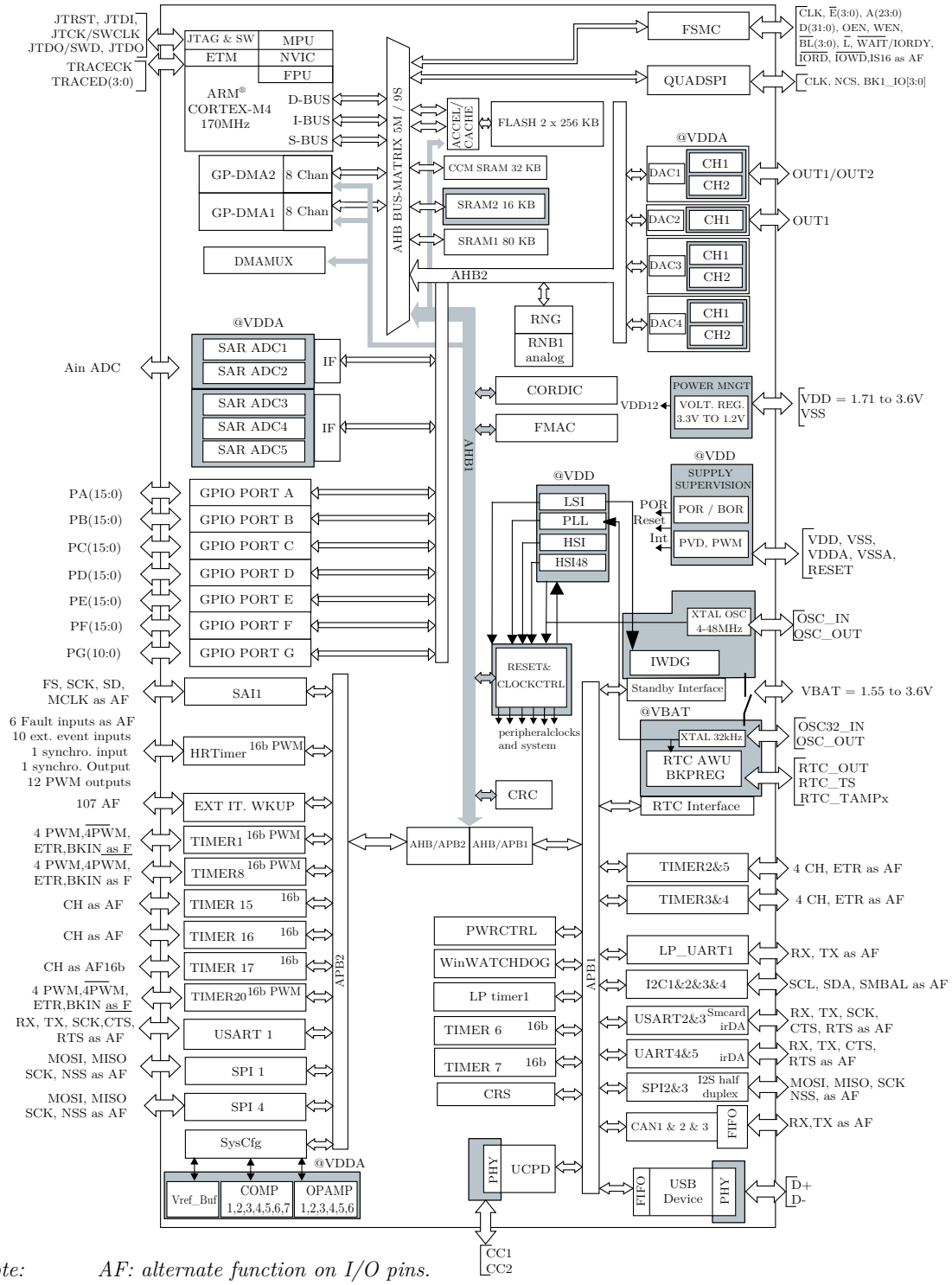
2.7 *STMicroelectronics STM32G474RE*

Para reduzir o custos de produção que são proporcionais à quantidade de componentes eletrônicos discretos utilizados numa confecção de placa de circuito impresso, economizar espaço físico e se beneficiar de uma fácil manipulação dos parâmetros dos elementos do circuito, optou-se pelo microcontrolador *STM32G474RE*, que possui alta integração de periféricos analógicos dedicados ao processamento de sinais elétricos.

O microcontrolador *STM32G474RE* possui um núcleo de processamento baseado na tecnologia *ARM Cortex-M4*, cujas principais características são a alta performance com baixo custo e consumo. Possui arquitetura de barramento de 32 *bits*, *clock* de 170 MHz e um *FPU* integrado. Também possui periféricos para cálculos trigonométricos *CORDIC* e cálculo de filtros digitais *FMAC*.

O diagrama de blocos geral do *STM32G474RE* é apresentado na Figura 8.

Figura 8 – Diagrama de blocos do microcontrolador *STM32G474RE*



Note: *AF: alternate function on I/O pins.*

Fonte: Adaptado de STMicroelectronics (2019b).

A seguir são detalhadas algumas das principais especificações técnicas e funcionalidades do microcontrolador.

2.7.1 Memória

Para a gravação ou leitura de dados, o microcontrolador conta com uma memória Flash de 512 *kbytes* e 128 *kbytes* de *SRAM*.

A memória flash possui ainda suporte a *ECC*, *PCROP*, *securable memory area* e é dividida em dois bancos que permite ler enquanto escreve.

A memória *SRAM* é dividida em 96 *kbytes* com checagem de paridade via hardware nos 32 primeiros *kbytes* e o restante é implementada em uma secção especial do barramento do microcontrolador para diminuir a latência e tempo de execução dos códigos.

Possui um *DMA* de 16 canais para transferência de dados sem a interferência do processador.

2.7.2 *Timers*

O *STM32G474RE* possui 17 *timers*, sendo eles:

- Um *timer* de alta resolução (184 ps), com 6 contadores de 16 *bits*;
- Dois *timers* de 32 *bits*;
- Onze timers de 16 *bits*;
- Dois *watchdog timers*;
- Um *SysTick timer*.

Alguns *timers* possuem a capacidade de capturar eventos externos, tais como mudança de tensão em pinos específicos do microcontrolador, e também funcionam como contadores de pulso.

2.7.3 Periféricos analógicos

2.7.3.1 *ADC*

A leitura de tensões analógicas é feita através de *ADC*s, dos quais o microcontrolador conta com cinco unidades de 12 *bits* de precisão com *hardware oversampling* integrado, podendo estender para 16 *bits*.

A taxa de conversão máxima do *ADC* é de 4 MHz, possui até 42 canais compartilhados e a faixa de conversão abrange 0 a 3,6 *V*.

2.7.3.2 *DAC*

O *STM32G474RE* possui 4 *DAC*s totalizando 7 canais, sendo 3 canais externos com *buffer* e velocidade máxima de 1 *MSPS* e 4 canais internos sem *buffer* com velocidade máxima de 15 *MSPS*.

2.7.3.3 Amplificadores operacionais e comparadores

O microcontrolador conta com 6 amplificadores operacionais *rail-to-rail* que podem funcionar no modo *PGA* com todos os terminais acessíveis. Também possui 7 comparadores analógicos ultra-rápidos com histerese programável e *rail-to-rail*.

2.7.4 Periféricos de comunicação

O *STM32G474RE* possui cinco interfaces tipo *USART*/*UART*, com suporte configurável entre 7 a 9 *bits* de transmissão (incluindo *bits* de paridade), além de uma conexão *USB* em conformidade com a especificação 2.0 *full-speed*.

2.8 *Espressif Systems ESP-32*

A escolha do microcontrolador *ESP-32* teve como base o baixo custo e alta velocidade por possuir dois núcleos de processamento, sendo um dedicado às operações de rede sem fio que aumenta a estabilidade da conexão *wireless*.

O microcontrolador *ESP-32* possui dois microprocessadores de 32 *bits Cadence Tensilica Xtensa LX6* com arquitetura *Harvard*. Os núcleos de processamento podem rodar a velocidades de até 240 MHz e possuem um *FPU* integrado, que consiste num hardware dedicado para executar operações de ponto flutuante.

O diagrama de blocos que ilustra o núcleo e os periféricos presentes no *ESP-32* está mostrado na Figura 9.

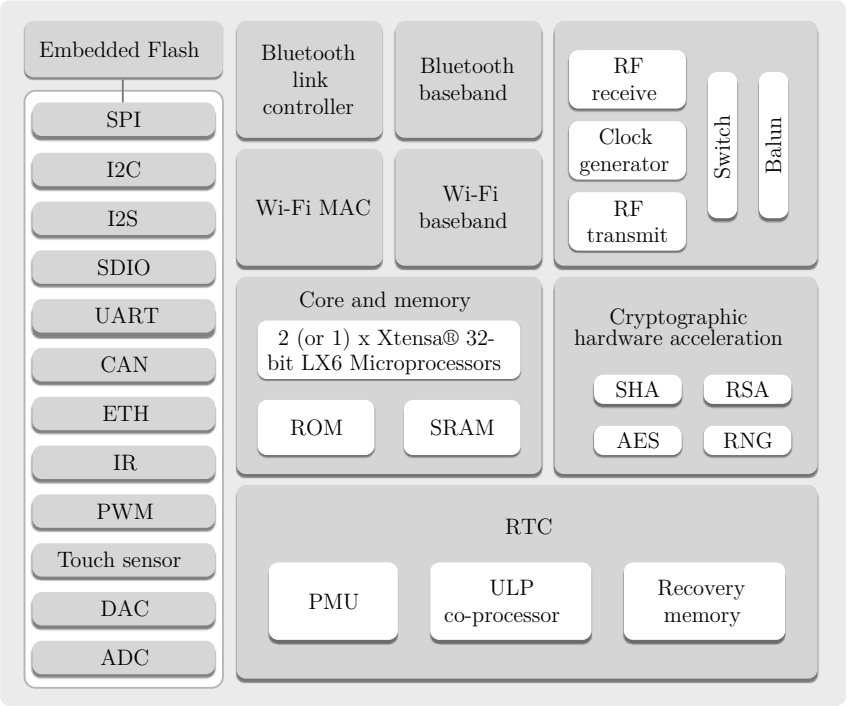
2.8.1 Memória

A memória embarcada conta com 448 *kbytes* de *ROM* e 520 *kbytes* de *SRAM*, sendo estas de uso interno. Para gravação e leitura de dados é preciso de uma memória Flash *SPI* externa, o qual é suportado até 16 *Mbytes*.

2.8.2 Aceleradores de criptografia

Por ser um microcontrolador voltado à comunicação, possui aceleradores de criptografia *AES*, *RSA*, *ECC* e *SHA*. O tamanho máximo para as operações criptográficas *RSA* e *ECC* são 4096 *bits*.

Figura 9 – Diagrama de blocos do microcontrolador *ESP-32*



Fonte: (SYSTEMS, 2020).

3 Ferramentas e procedimentos

Neste capítulo será abordado as ferramentas utilizadas e os procedimentos realizados para a validação do projeto.

3.1 Placas de desenvolvimento

Nesta seção serão listadas as placas de desenvolvimento utilizadas para o processamento dos sinais do sistema, comunicação entre dispositivos e interface para o usuário.

3.1.1 *NUCLEO-G474RE*

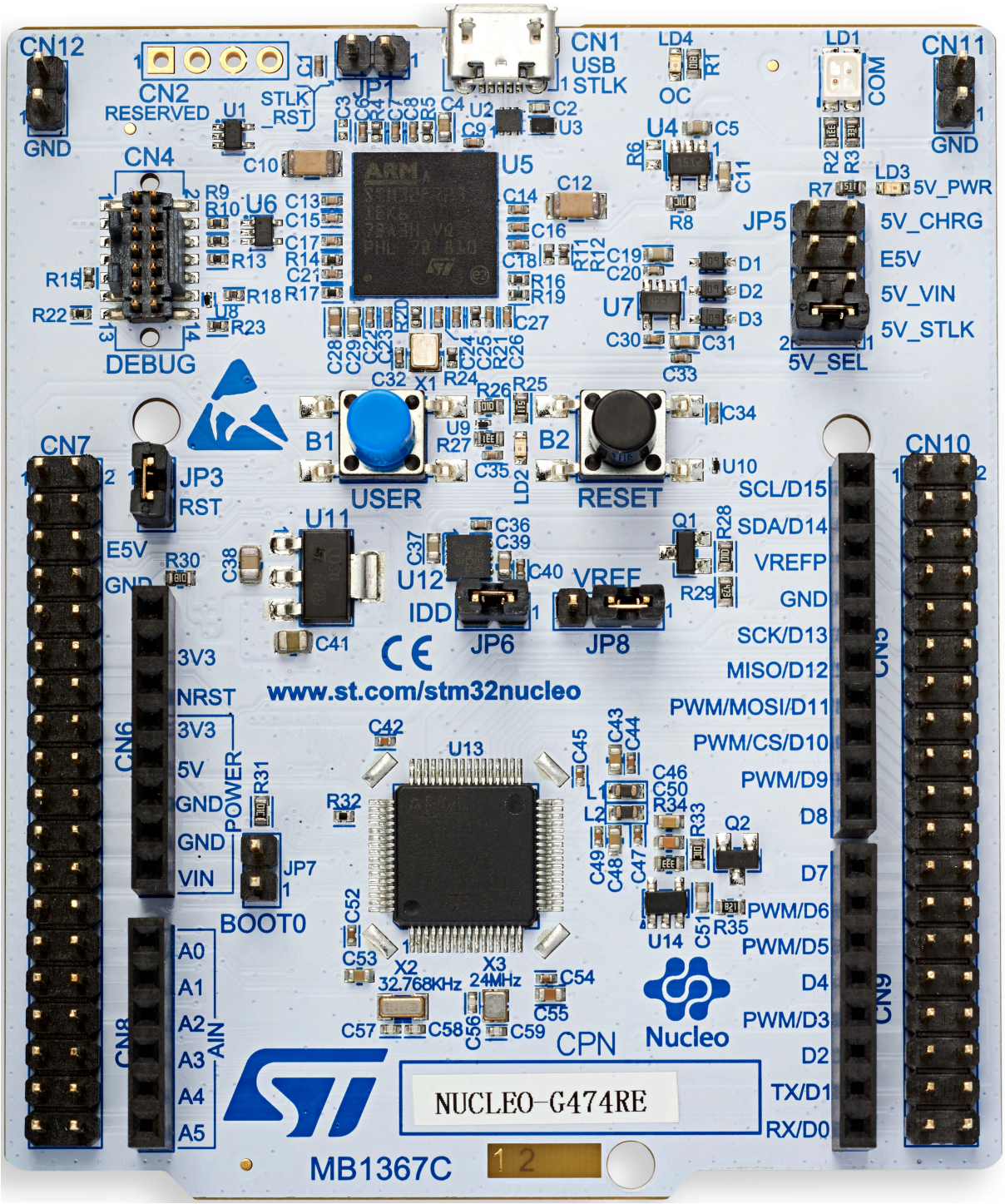
Esta placa de desenvolvimento da *STMicroelectronics* conta com o processador *STM32G474RE*, em um formato com 64 pinos disponíveis para serem utilizados e também um layout que permite a inserção de *shields* para a plataforma *Arduino*.

A Figura 10 mostra a parte frontal da placa.

3.1.2 *LOLIN-32 Lite*

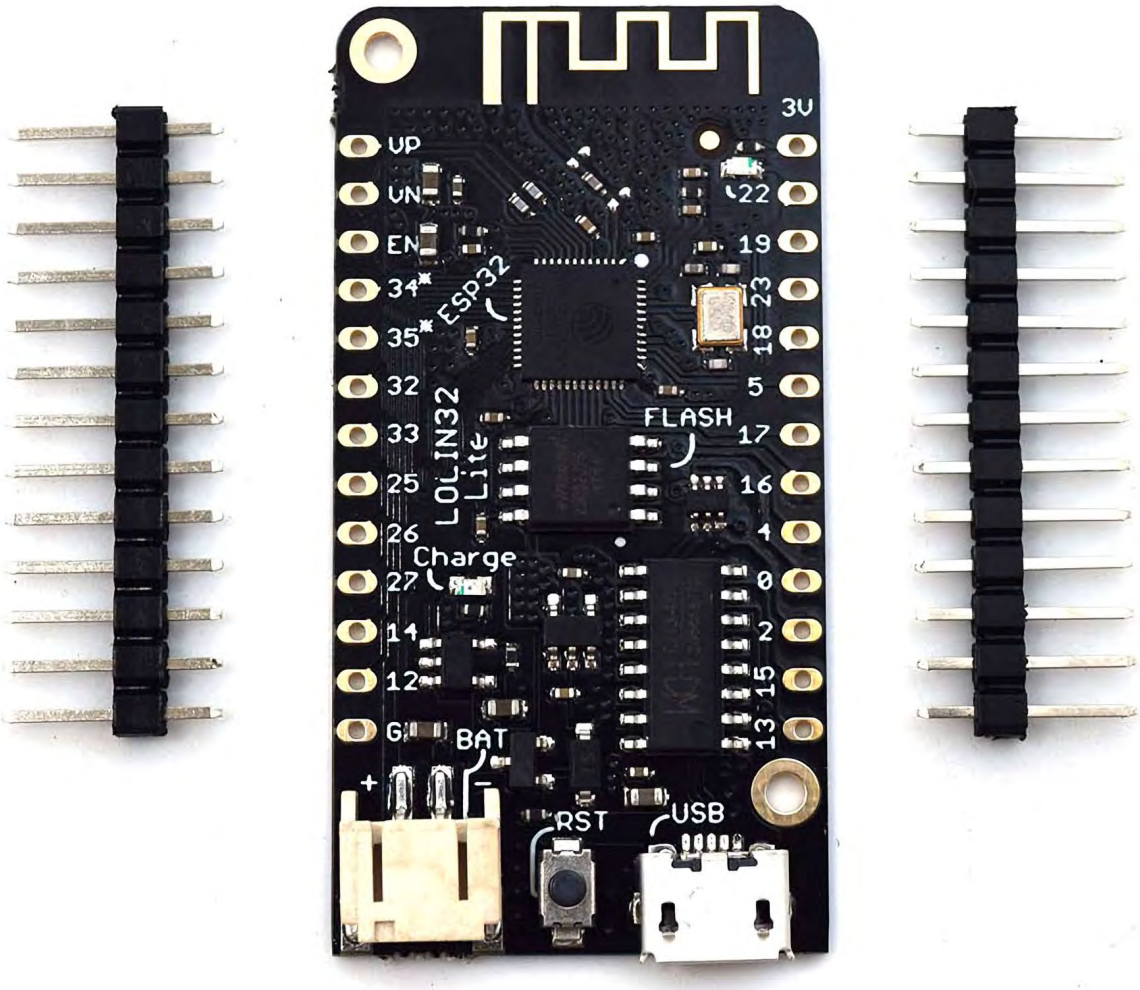
O módulo *LOLIN-32 Lite* é fabricado pela empresa *WEMOS* e integra um microcontrolador *ESP-32 WROOM rev.1*. Possui 26 pinos, saída para carregar bateria ou utilizá-la como fonte de energia e uma antena embutida na placa de circuito impresso.

Figura 10 – Placa de desenvolvimento *NUCLEO-G474RE*



Fonte: (STMICROELECTRONICS, 2019a)

Figura 11 – Placa de desenvolvimento *LOLIN-32 Lite*



Fonte: Adaptado de Projects (2020).

3.2 Sensores

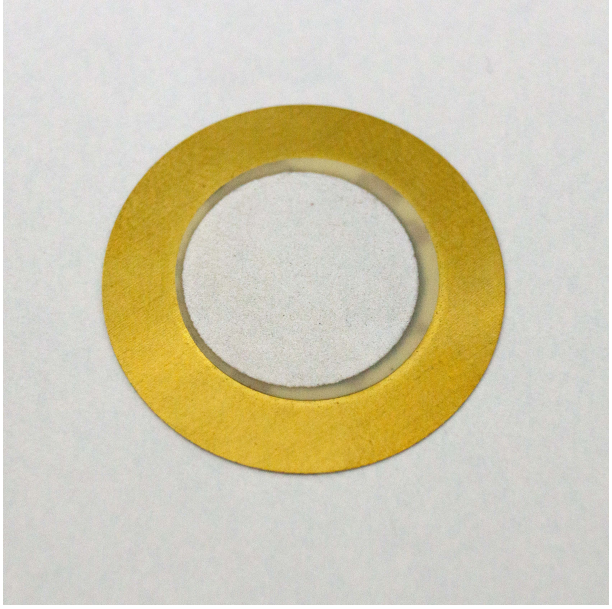
Nesta seção serão listados os sensores utilizados na captação dos sinais provenientes do sistema.

3.2.1 Disco Piezoelétrico

O sensor piezoelétrico utilizado tem como finalidade medir as vibrações do sistema em análise. Foi utilizado um modelo discoidal de 35 mm de diâmetro, no qual foi posteriormente soldado fios de cobre esmaltado para realizar as ligações necessárias.

A Figura 12 mostra o disco piezoelétrico utilizado neste trabalho.

Figura 12 – Disco piezoelétrico

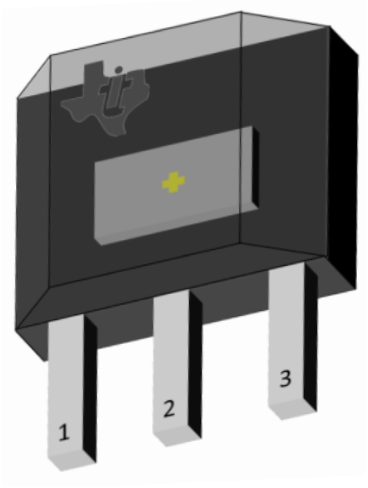


Fonte: O autor (2020).

3.2.2 Sensor *Hall DRV5023*

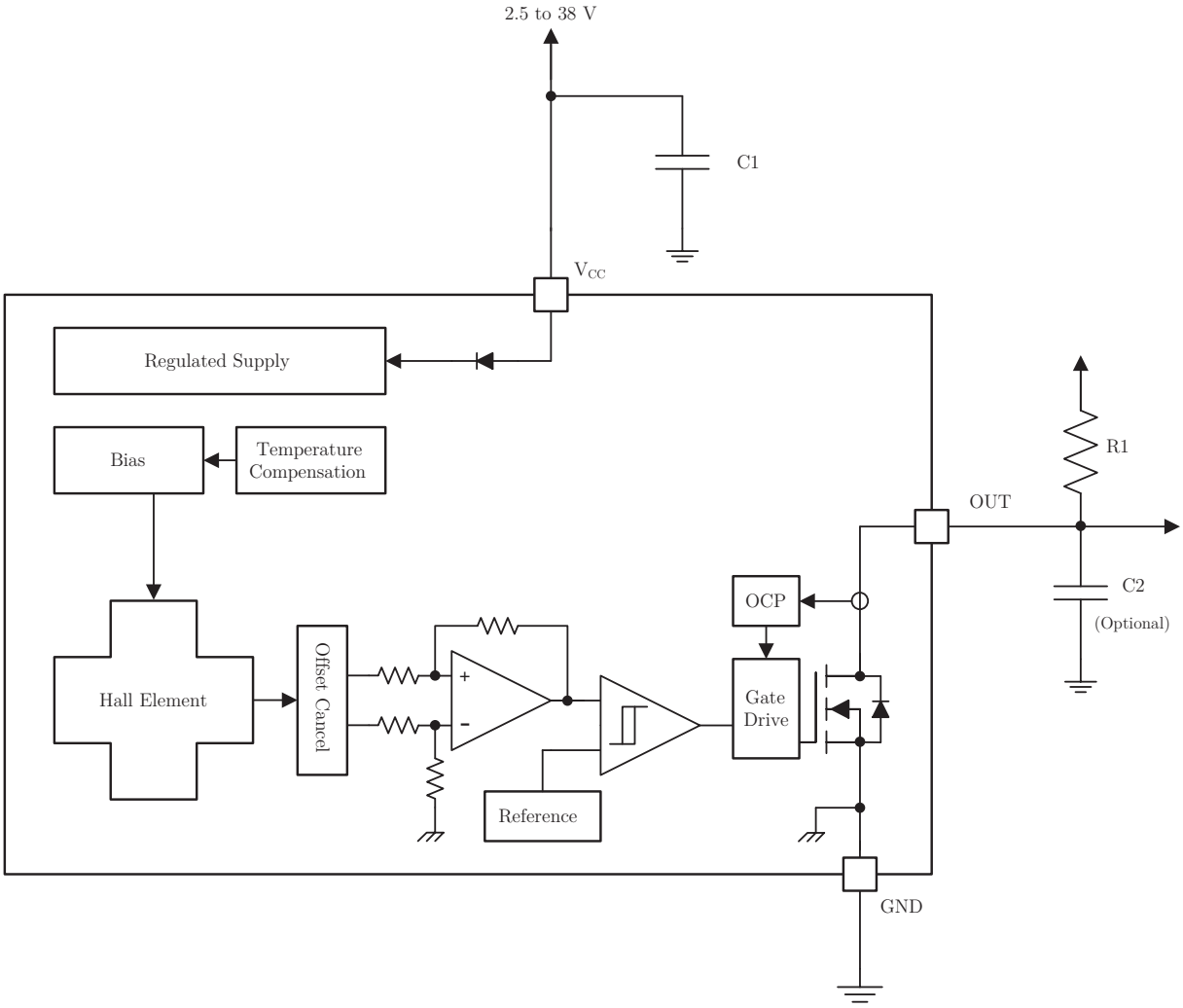
O sensor *Hall DRV5023* é um *switch* digital unipolar com estabilização *chopper* que possui saída dreno aberto, cujo encapsulamento é mostrado na Figura 13 e seu diagrama de blocos representado na Figura 14.

Figura 13 – Encapsulamento do sensor *DRV5023* modelo *TO-92 LPG*



Fonte: (INSTRUMENTS, 2016)

Figura 14 – Diagrama de blocos funcional do sensor *DRV5023*

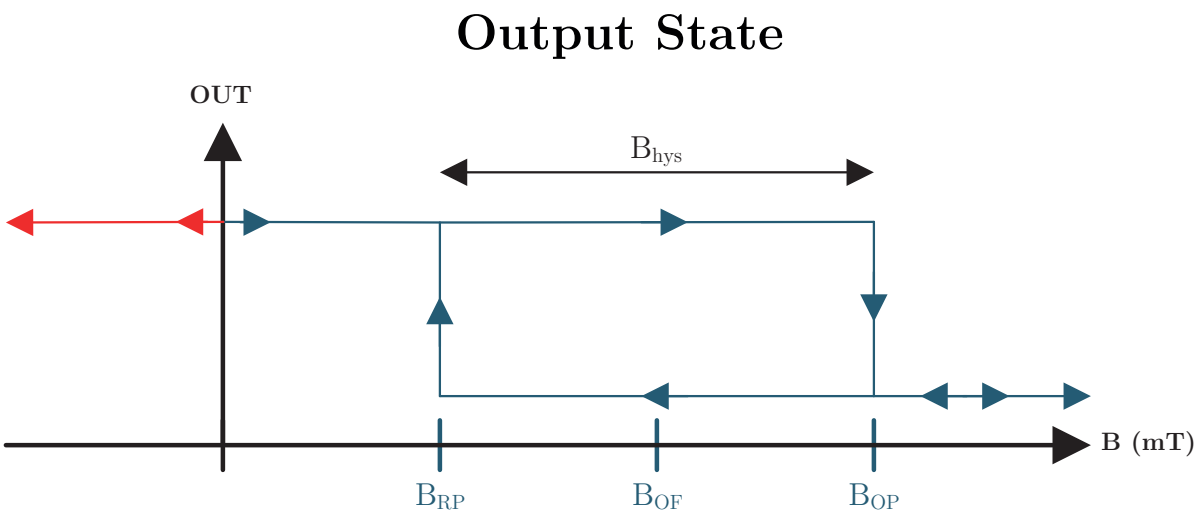


Fonte: (INSTRUMENTS, 2016)

Quando uma certa intensidade magnética B_{OP} é aplicada ao sensor, a sua saída muda para nível baixo, permanecendo assim até um novo nível de intensidade B_{RP} um pouco menor. Os níveis de gatilho do modelo usado é de $B_{OP} = 14,5$ mT para o nível superior e $B_{RP} = 6$ mT para o nível inferior, conforme ilustra a Figura 15.

A Figura 16 mostra os efeitos das linhas de campo magnético no sensor conforme a orientação do ímã.

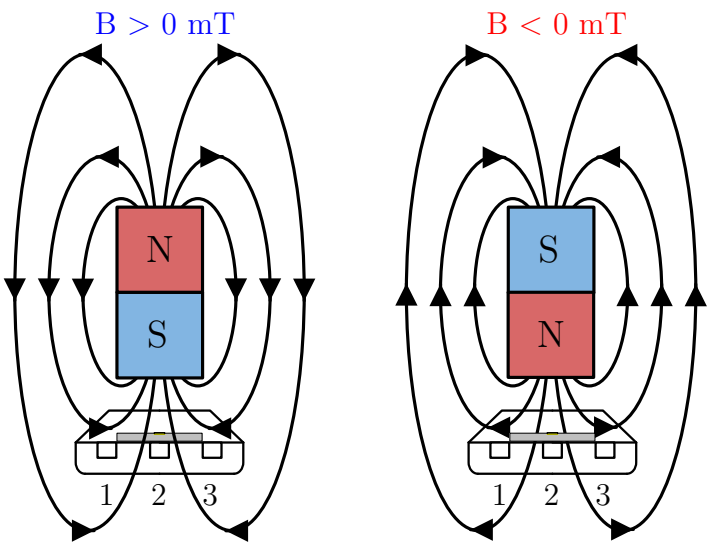
Figura 15 – Saída do sensor em função da densidade de fluxo magnético



Fonte: (INSTRUMENTS, 2016)

Figura 16 – Influência das orientações dos polos magnéticos

N = Polo norte, S = Polo sul



(Vista inferior)

Fonte: Adaptado de Instruments (2016)

3.3 *Sofwares* de desenvolvimento, simulação e modelagem

Nesta seção serão descritos as ferramentas utilizadas na programação de microcontroladores, simulações de circuitos e modelagem de peças para a realização do protótipo.

3.3.1 Plataformas de desenvolvimento para microcontroladores

Para programar os microcontroladores *ESP-32* e *STM32G4* usa-se a linguagem *C*. No caso do *ESP-32*, utilizou-se o ambiente de desenvolvimento *open-source Eclipse IDE*, com toda biblioteca baseada no *framework ESP-IDF* que utiliza o sistema operacional de tempo real *FreeRTOS*. Para o microcontrolador da *STMicroelectronics*, usou-se o *STM32CubeIDE*, baseado no *Eclipse*, o qual provê um conjunto de *API*s para auxiliar a programação da série *G4* através do *STM32CubeG4*.

3.3.2 Plataforma de desenvolvimento de aplicativos móveis

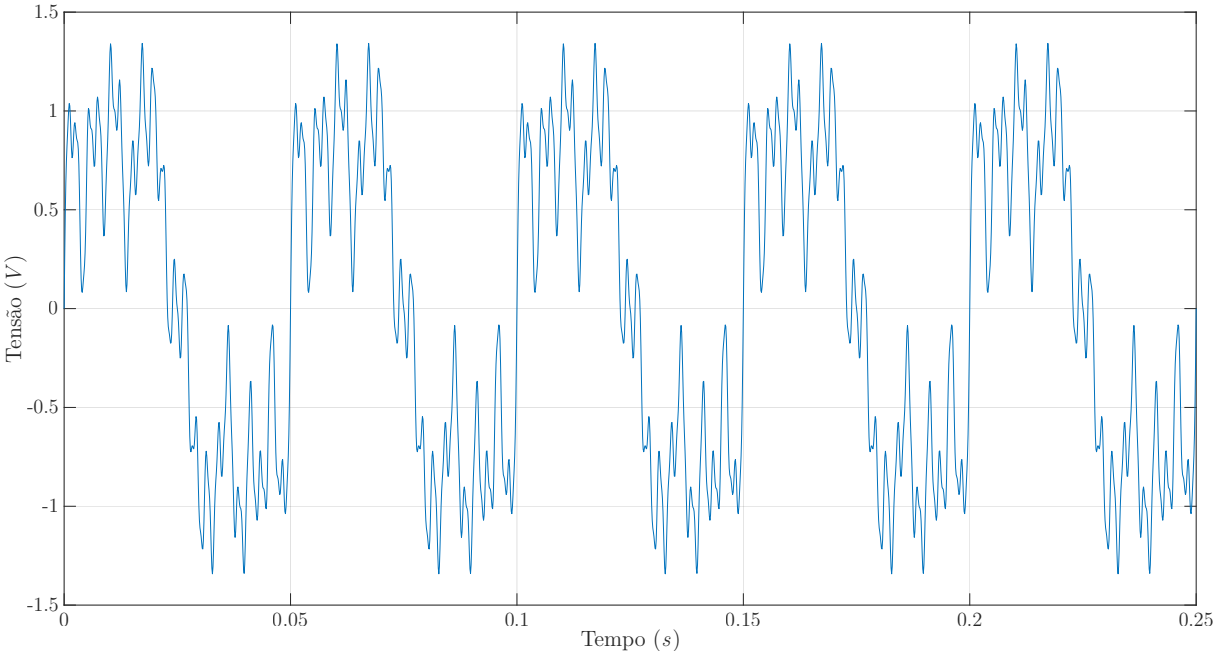
A programação do aplicativo para *Android* é feita usando-se a linguagem de programação *Java*, através do ambiente de desenvolvimento *Android Studio*, fornecido pela *Google*.

3.3.3 Software de simulação e cálculo númerico MATLAB

As simulações e design do filtro digital *FIR* foram feitas utilizando-se o *software MATLAB*, juntamente com o complemento *DSP Toolbox*.

Para realizar os testes de bancada com gerador de sinal arbitrário, foi criado uma forma de onda que simula uma senoide com 12 harmônicas através do *MATLAB*, para representar os sinais de vibração provenientes do sensor piezoelétrico juntamente com ruídos. A forma de onda está mostrada na Figura 17.

Figura 17 – Onda senoidal de 20 Hz contendo 12 harmônicas



Fonte: O autor (2020).

As harmônicas foram introduzidas de modo a ter-se uma forma de onda bastante desfigurada em relação à uma senoide pura, sem necessariamente obedecer um critério rigoroso, mas que extrapolasse um pouco alguma situação real. A Tabela 2 lista as componentes da senoide simulada.

Tabela 2 – Componentes da senoide simulada

Onda	Frequência	Amplitude
Fundamental	20 *Hz*	1,000 *V*
1ª *harmônica*	40 *Hz*	0,001 *V*
2ª *harmônica*	60 *Hz*	0,250 *V*
3ª *harmônica*	80 *Hz*	0,004 *V*
4ª *harmônica*	100 *Hz*	0,008 *V*
5ª *harmônica*	120 *Hz*	0,250 *V*
6ª *harmônica*	140 *Hz*	0,031 *V*
7ª *harmônica*	160 *Hz*	0,063 *V*
8ª *harmônica*	180 *Hz*	0,125 *V*
9ª *harmônica*	200 *Hz*	0,250 *V*
10ª *harmônica*	220 *Hz*	0,063 *V*
20ª *harmônica*	420 *Hz*	0,250 *V*
49ª *harmônica*	1000 *Hz*	0,063 *V*

Fonte: O autor (2020).

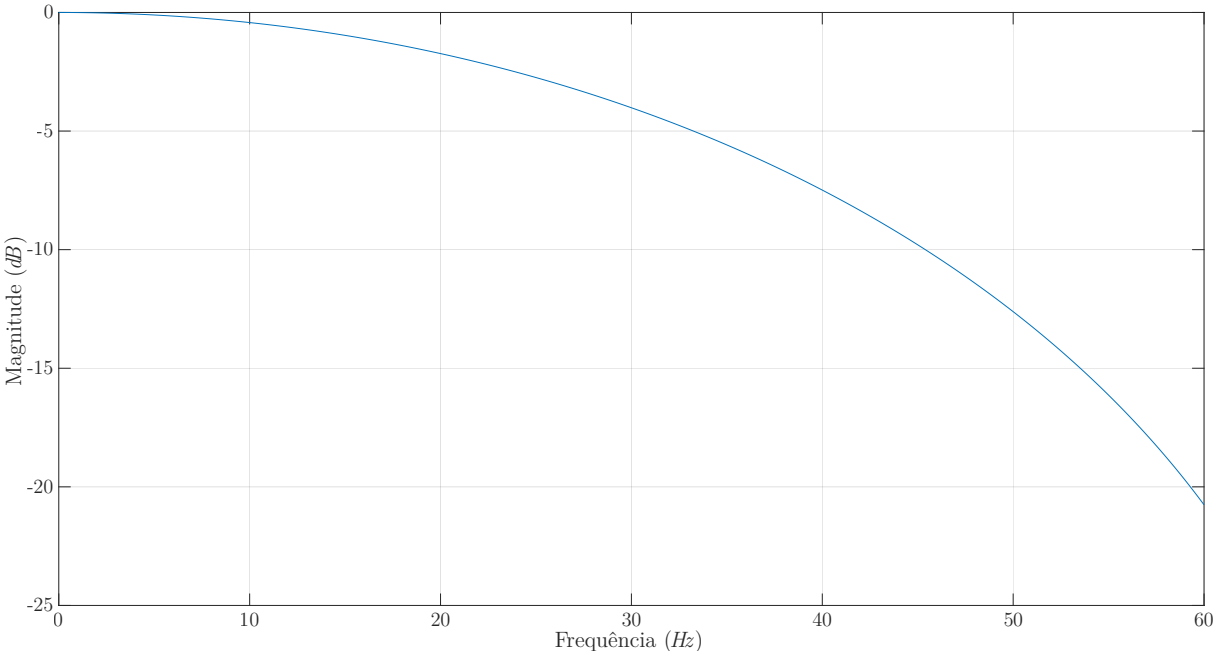
Para a filtragem das componentes harmônicas, foi utilizado um filtro passa baixas

FIR Chebyshev através do *filterDesigner*, com frequência de corte de 50 *Hz* e ordem 200.

A escolha da ordem do filtro foi feita arbitrariamente, suficientemente grande para que o sinal apresentasse boa resposta qualitativamente sem comprometer o tempo de execução do código, enquanto que a frequência foi escolhida próxima de 50–60 *Hz* considerando que o ruído mais baixo é proveniente da alimentação dos sistemas em análise.

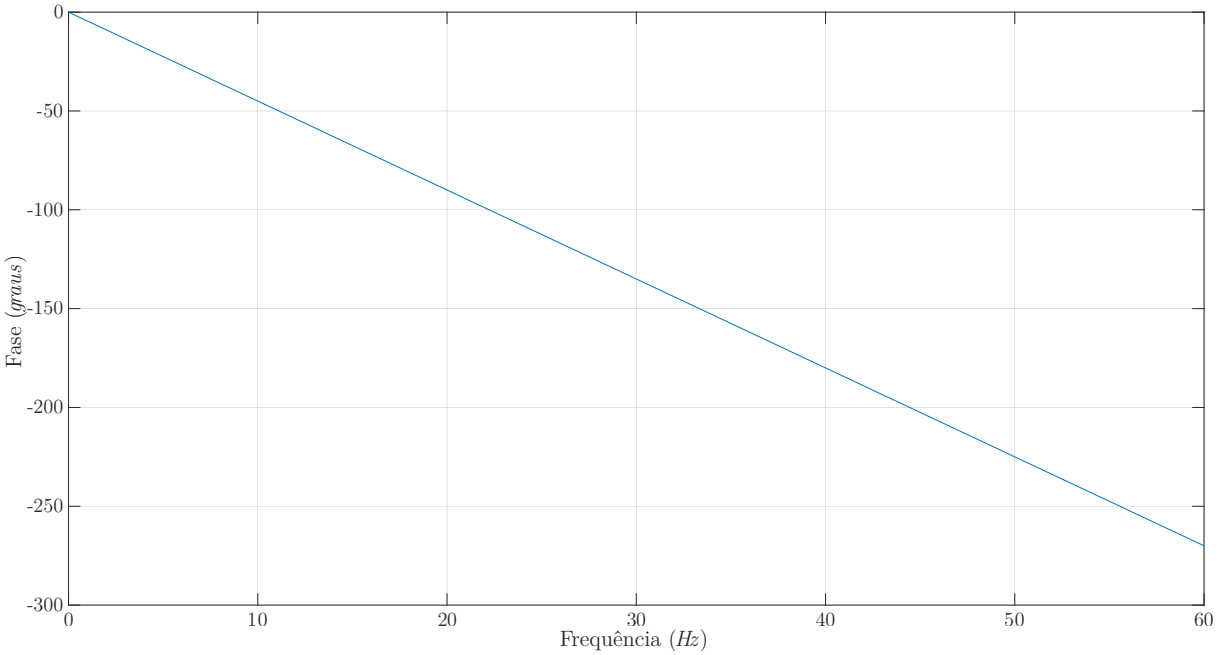
A topologia *Chebyshev* foi escolhida por apresentar uma banda de transição estreita. Na Figura 18 é mostrada a resposta em frequência da magnitude do sinal de saída do filtro e na Figura 19 a resposta em frequência da fase.

Figura 18 – Resposta em frequência da amplitude do sinal na saída do filtro



Fonte: O autor (2020).

Figura 19 – Resposta em frequência da fase do sinal na saída do filtro



Fonte: O autor (2020).

A Tabela 3 mostra os valores no ponto de projeto do filtro, bem como o valor esperado quando aplicado um sinal de 20 *Hz* em sua entrada, que é a frequência usada para os testes e a segurança do usuário que irá balancear as peças.

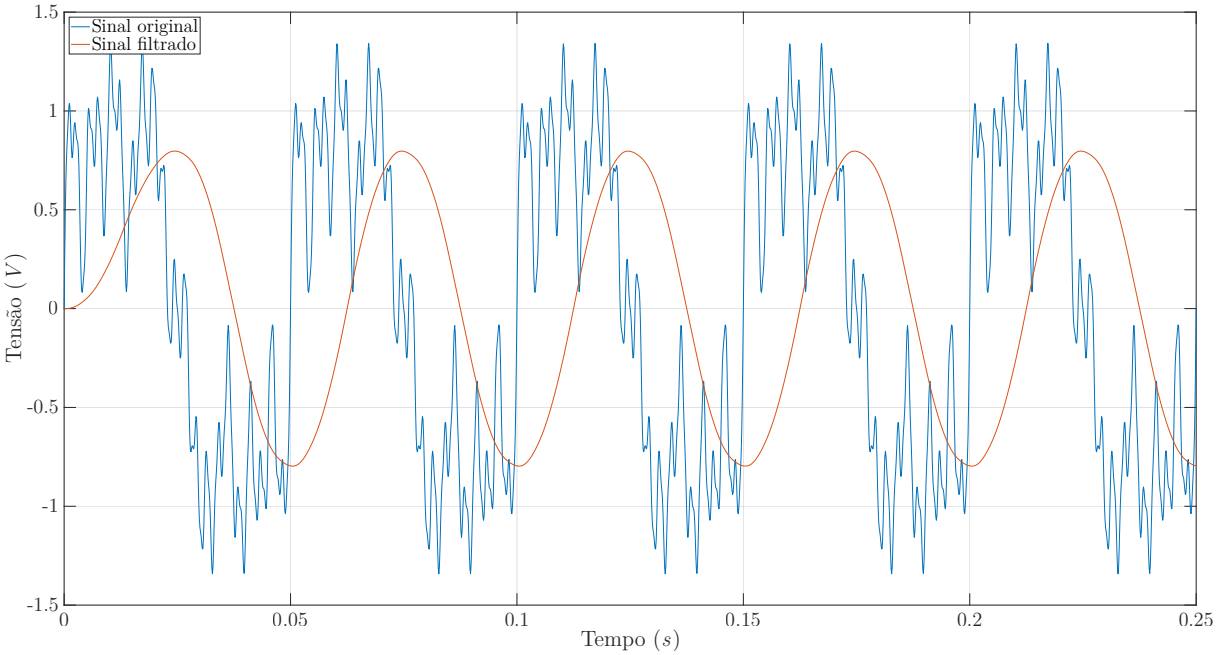
Tabela 3 – Características do filtro projetado

Frequência	Magnitude	Fase
50 *Hz*	-12,61 *dB*	-225,00°
20 *Hz*	-1,73 *dB*	-90,00°

Fonte: O autor (2020).

Com o filtro projetado, foi simulado o sinal de saída tendo como sinal de entrada a forma de onda de 20 Hz contendo 10 harmônicas simulada previamente. A Figura 20 mostra o sinal original de entrada e o de saída filtrado.

Figura 20 – Sinal de entrada com ruído e sinal de saída filtrado



Fonte: O autor (2020).

A partir da análise do gráfico, podemos determinar algumas características, tomando o sinal original como referência, como as listadas na Tabela 4.

Tabela 4 – Características dos sinais simulados

Sinal	Tensão *RMS*	Fase relativa
Original	0,7993 V	0°
Filtrado	0,5701 V	86,4592°

Fonte: O autor (2020).

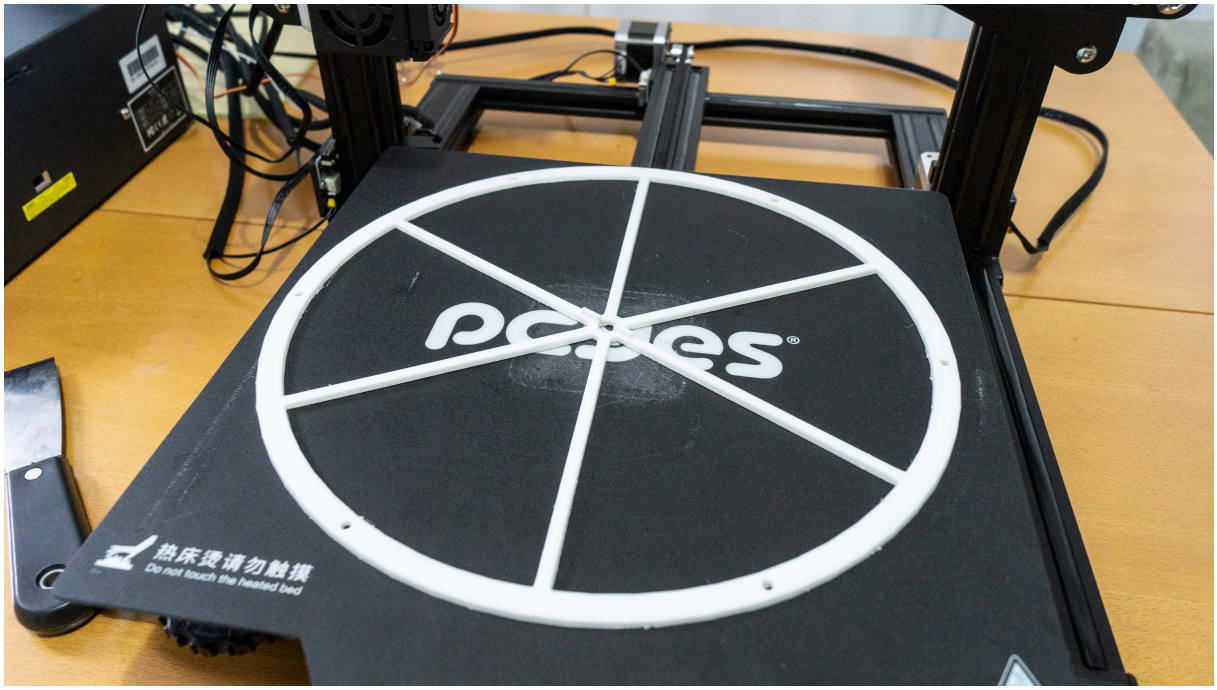
3.3.4 *Software* de modelagem de peças *SolidWorks*

Para simular situações de desbalanceamento no protótipo, envolvendo um ventilador de mesa, projetou-se uma roda que possui furos localizados à 120° um do outro, com aproximadamente 292 mm de diâmetro, através do software de modelagem de peças *SolidWorks* 2020. O detalhamento técnico encontra-se disponível no Apêndice C.

O modelo da roda foi então impresso em uma impressora *3D PCYES Faber-10*, com material termoplástico *PLA* que oferece as condições necessárias para o teste e é biodegradável.

Uma foto da roda confeccionada está representada na Figura 21.

Figura 21 – Roda de desbalanceamento impresso através de impressora *3D*



Fonte: O autor (2020).

3.4 Funcionamento do sistema de balanceamento

Nesta seção serão descritos o funcionamento do sistema completo.

3.4.1 Ângulo e intensidade de um desbalanceamento

Para determinar o ângulo em que se encontra um desbalanceamento de um sistema rotativo, realizam-se duas medições:

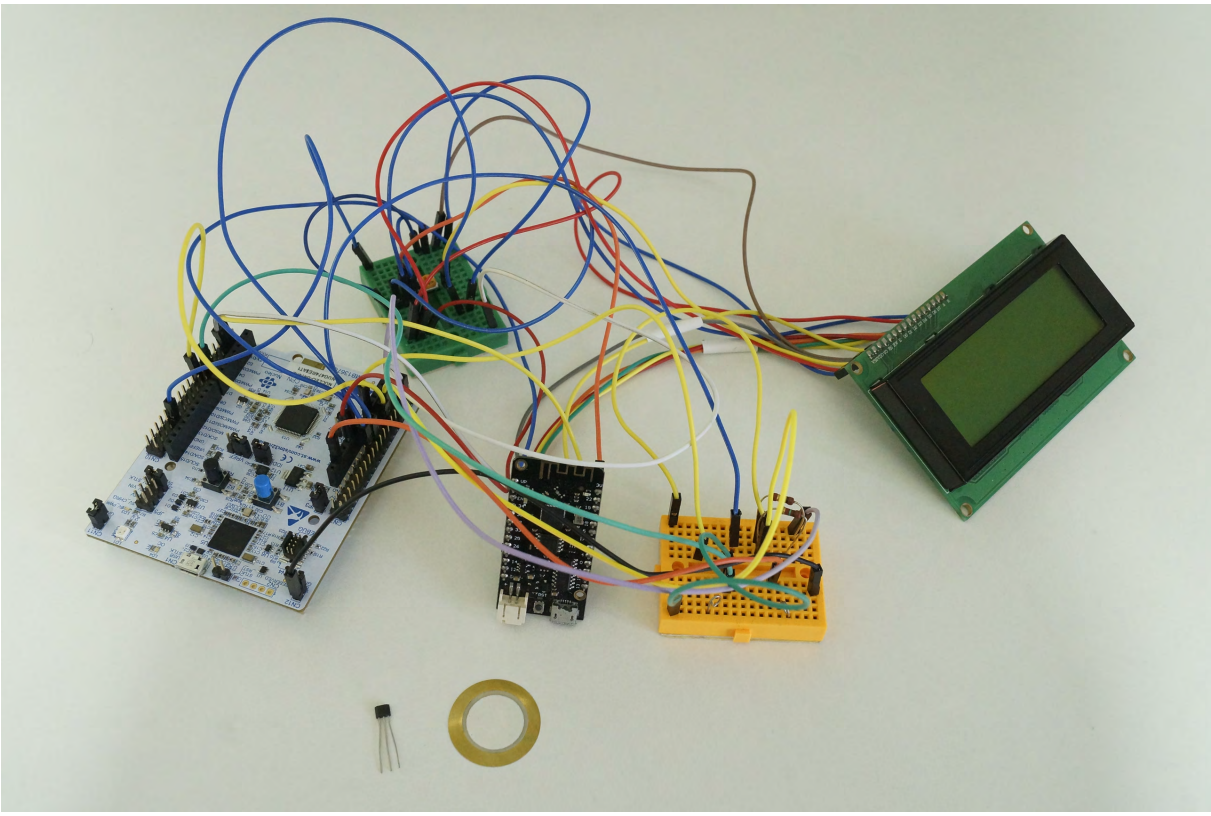
1. Mede-se a rotação, através do *Hall* que gera uma onda quadrada conforme o sistema completa uma volta em torno do seu eixo;
2. Mede-se a vibração proveniente do desbalanceamento da peça girante, através do disco piezoelétrico o qual gera uma senoide proporcional à sua intensidade.

A diferença angular entre o zero da senoide e da borda de subida da onda quadrada fornece o ângulo de desbalanceamento e a magnitude da senoide indica a intensidade da vibração.

3.4.2 Descrição geral do sistema implementado

O protótipo com o microcontrolador e os circuitos auxiliares estão mostrados na Figura 22.

Figura 22 – Protótipo do sistema eletrônico



Fonte: O autor (2020).

O custo aproximado de cada componente utilizado está mostrado na Tabela 5.

Tabela 5 – Custos aproximados dos componentes utilizados no projeto

Componente	Custo aproximado
NUCLEO-G474RE	R$ 140,00
LOLIN32-LITE	R$ 30,00
Disco Piezoelétrico 35 *mm*	R$ 1,06
DRV5023	R$ 3,30

Fonte: O autor (2020).

A sequência lógica de funcionamento segue as seguintes etapas:

1. O sinal do piezoelétrico é lido pelo pino *PA1* que está configurado como amplificador somador, cujo circuito está realizado com divisores resistivos externamente de modo a realizar um *offset* do sinal em 1,65 V para que haja a máxima excursão do sinal na entrada do *ADC*.

2. O microcontrolador então realiza a filtragem do sinal lido pelo *ADC* através do filtro digital *FIR*, cujo sinal resultante é roteado para um comparador interno através

do *DAC* com buffer também interno. O comparador tem como referência 1,65 V, através de um segundo *DAC* interno, e uma histerese programada em 70 mV, para que se obtenha o zero da onda.

3. Em paralelo, o sinal do sensor *Hall* entra pela porta *PB10* usada como *IC* e este sinal é então comparado com o sinal de saída do comparador interno, assim obtendo-se a defasagem do sinal original com o sinal lido pelo sensor *Hall*.

4. Na sequência, as informações de defasagem e amplitude do sinal do disco piezoelétrico é enviado ao ESP-32 usando comunicação *UART*, e do ESP-32 é enviado via *WiFi* para um aplicativo de celular baseado em *Android* e também mostrado as informações num display *LCD*. A comunicação entre o aplicativo e o ESP-32 é baseado no protocolo *TCP*, e as informações de *login* da rede são criptografadas quando trafegam entre estes dispositivos.

5. Através do aplicativo e as informações obtidas do *NUCLEO-G474RE*, os dados são manipulados de forma a ser mostrado intuitivamente para o usuário final qual é o ângulo de desbalanceamento e a massa necessária para o balancear.

3.4.3 Métodos de balanceamento

Na preparação do sistema para as medições é estabelecido um ponto de referência na peça girante para um sensor de posição que irá atuar a cada rotação. Além disso, posiciona-se o sensor de vibração em uma região fixa do aparelho em análise que se maximize a sensibilidade com relação à vibração do sistema.

A seguir serão apresentados dois métodos utilizados no balanceamento de um aparelho que possui peça rotativa.

3.4.3.1 Balanceamento de peças rotativas sem uma peça padrão balanceada de comparação

Para medir o desbalanceamento de um aparelho que possui peça rotativa que está vibrando, sem que haja uma peça padrão balanceada para comparação, utiliza-se o método que envolve a realização de quatro medições, três delas para definir o ângulo e uma para a massa. As medições resultam em vetores, cujos passos envolvem:

1. A inserção de uma massa conhecida em um ponto qualquer da peça rotativa (preferencialmente na periferia da peça), sendo este ponto considerado a referência (ponto zero);

2. Remove-se a massa e se insere novamente na peça rotativa, desta vez a 120° em relação à referência;

3. Repete-se mais uma vez o processo, desta vez a 240° em relação ao ponto zero;

4. A soma vetorial das três medidas executadas nos passos anteriores irá gerar uma resultante que terá um ângulo em relação à referência. Esse ângulo indica a reta suporte do centro de massa do desbalanceamento.

5. Por fim, à 180° deste local deve-se inserir a massa para a correção do desbalanceamento. Caso as vibrações persistam, será indicado a quantidade de massa a ser inserida ou retirada deste local para que o sistema fique balanceado.

3.4.3.2 Balanceamento de peças rotativas com uma peça padrão balanceada de comparação

Quando se deseja balancear várias peças iguais, como por exemplo em uma linha de produção, é desejável reduzir o número de passos para o balanceamento. Assim, deve-se utilizar uma peça padrão previamente balanceada, para então executar-se os seguintes procedimentos:

1. Na peça padrão balanceada, realiza-se uma medição da intensidade de vibração com esta girando;

2. Adiciona-se uma massa em um ponto qualquer da peça girante (preferencialmente na periferia), fazendo com que a peça fique desbalanceada, e mede-se novamente;

3. Após duas medições, é obtido uma equação linear que irá representar a quantidade de massa para diferentes intensidades de vibrações.

Após a execução dos procedimentos, cada peça desbalanceada que for analisada irá gerar um vetor cujo ângulo estará defasado em relação ao ponto de referência do sensor, com a magnitude obtida através da equação obtida no passo 3.

4 Testes

Para verificar a funcionalidade do protótipo, foram feitos testes em bancada e posteriormente num sistema que tenta simular uma condição o mais próximo possível da realidade.

Nos testes feitos usando o aplicativo de celular, foi virtualizado um telefone da *Google* modelo *Pixel 2* através da *IDE Android Studio*. A Figura 23 mostra a janela principal do aplicativo aberto no telefone virtual.

Figura 23 – Tela inicial do aplicativo *Balanceador* no dispositivo virtual *Pixel 2*

Fonte: O autor (2020).

Inicialmente verificou-se a funcionalidade do sistema *WiFi* e as interações entre o aplicativo e o microcontrolador *ESP-32*.

A seguir, utilizou-se um gerador de sinais arbitrários *FeelTech* modelo *FY6800*, com o primeiro canal simulando o sensor *Hall* e o segundo canal as vibrações do disco

piezoelétrico.

Por fim, utilizou-se um ventilador de mesa foi adicionado à roda da subseção 3.3.4 que possui pontos onde se inserem massas para desbalancear o sistema, instalado atrás da hélice.

4.1 Testes de comunicação

Foi utilizado o software analisador de protocolo *Wireshark* para verificar os pacotes de transmissão enviados entre o microcontrolador e o celular, de forma a verificar que os dados de login do roteador estão criptografados e não possam ser descobertos por terceiros. Um detalhamento das criptografias utilizadas está apresentado no Apêndice B

O microcontrolador *ESP-32* opera no modo *STA+AP*, ou seja, simultaneamente este é um ponto de acesso e também se conecta a um ponto de acesso externo (por exemplo, um roteador).

Através do celular, foi conectado ao ponto de acesso *ESP_Balancer* gerado pelo microcontrolador. Na sequência, foram inseridos as informações de *login* da rede *WiFi* do autor para que o *ESP-32* se conecte à internet e compartilhe as informações pelo roteador.

Todos os pacotes de dados trafegados, excetuando-se os *bytes* de controle, estão representados em vermelho sempre que aparecerem. Os passos a seguir representam a sequência de eventos para a troca de informações de *login*.

1. Assim que o aplicativo do celular é aberto e se conecta à um ponto de acesso, o aplicativo solicita ao microcontrolador *ESP-32* a chave pública da criptografia *RSA*, conforme mostra a Figura 24;

Figura 24 – Requisição da chave pública pelo aplicativo do celular

```
No.  Time        Source          Destination     Protocol  Length  Info
162  8.436540    192.168.15.7    192.168.15.21   TCP       64      56523 → 49152 [PSH, ACK] Seq=1 Ack=452 Win=63789 Len=10

Frame 162: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface \Device\NPF_{0E5BC0EB-5433-40DB-8BE6-AA782E45028A}, id 0
Ethernet II, Src: Micro-St_38:4e:28 (d8:cb:8a:38:4e:28), Dst: Espressi_89:15:25 (b4:e6:2d:89:15:25)
Internet Protocol Version 4, Src: 192.168.15.7, Dst: 192.168.15.21
Transmission Control Protocol, Src Port: 56523, Dst Port: 49152, Seq: 1, Ack: 452, Len: 10
Data (10 bytes)

0000  b4 e6 2d 89 15 25 d8 cb 8a 38 4e 28 08 00 45 00   ..-..%...8N(..E.
0010  00 32 c7 de 40 00 80 06 00 00 c0 a8 0f 07 c0 a8   .2..@..........
0020  0f 15 dc cb c0 00 71 a2 98 20 00 00 1b 42 50 18   ......q.. ...BP.
0030  f9 2d 9f 91 00 00 53 45 4e 44 50 55 42 4b 45 59   .-....SENDPUBKEY
```

Fonte: O autor (2020).

2. O microcontrolador *ESP-32* envia ao aplicativo do celular a chave pública, conforme mostra a Figura 25.

Figura 25 – Envio da chave pública pelo microcontrolador *ESP-32*

```
No.   Time        Source          Destination     Protocol  Length  Info
170   8.445575    192.168.15.21   192.168.15.7    TCP       505     49152 → 56524 [PSH, ACK] Seq=1 Ack=1 Win=5744 Len=451

Frame 170: 505 bytes on wire (4040 bits), 505 bytes captured (4040 bits) on interface \Device\NPF_{0E5BC0EB-5433-40DB-8BE6-AA782E45028A}, id 0
Ethernet II, Src: Espressi_89:15:25 (b4:e6:2d:89:15:25), Dst: Micro-St_38:4e:28 (d8:cb:8a:38:4e:28)
Internet Protocol Version 4, Src: 192.168.15.21, Dst: 192.168.15.7
Transmission Control Protocol, Src Port: 49152, Dst Port: 56524, Seq: 1, Ack: 1, Len: 451
Data (451 bytes)

0000  d8 cb 8a 38 4e 28 b4 e6 2d 89 15 25 08 00 45 00   ...8N(..-..%..E.
0010  01 eb 00 3f 00 00    06 1a 61 c0 a8 0f 15 c0 a8   ...?.....a......
0020  0f 07 c0 00 dc cc 00 00 19 83 b0 b9 65 90 50 18   ............e.P.
0030  16 70 f3 6f 00 00 2d 2d 2d 2d 2d 42 45 47 49 4e   .p.o..-----BEGIN
0040  20 50 55 42 4c 49 43 20 4b 45 59 2d 2d 2d 2d 2d    PUBLIC KEY-----
0050  0a 4d 49 49 42 49 6a 41 4e 42 67 6b 71 68 6b 69   .MIIBIjANBgkqhki
0060  47 39 77 30 42 41 51 45 46 41 41 4f 43 41 51 38   G9w0BAQEFAAOCAQ8
0070  41 4d 49 49 42 43 67 4b 43 41 51 45 41 33 47 36   AMIIBCgKCAQEA3G6
0080  39 59 4c 49 38 48 58 67 47 32 4e 50 69 6b 70 47   9YLI8HXgG2NPikpG
0090  4a 0a 2b 4b 55 71 56 50 42 47 2f 6a 34 72 6f 4d   J.+KUqVPBG/j4roM
00a0  47 4c 76 71 61 76 2f 47 39 68 48 6a 4d 64 42 2f   GLvqav/G9hHjMdB/
00b0  32 79 6d 4d 41 6f 64 52 66 67 78 55 74 4e 42 55   2ymMAodRfgxUtNBU
00c0  53 4c 49 2b 6a 6f 77 4a 4b 72 39 35 52 76 63 65   SLI+jowJKr95Rvce
00d0  76 6a 0a 35 71 72 78 57 2b 6b 6c 4e 45 7a 6e 6a   vj.5qrxW+klNEznj
00e0  32 37 41 31 48 68 47 76 4b 6d 55 46 6e 67 66 61   27A1HhGvKmUFngfa
00f0  43 6f 4a 45 42 6a 65 6e 6d 2f 79 7a 4e 6d 44 7a   CoJEBjenm/yzNmDz
0100  4c 5a 43 57 4c 67 65 78 44 59 45 56 56 2b 6e 48   LZCWLgexDYEVV+nH
0110  75 4e 42 0a 64 4f 5a 43 6b 5a 4e 5a 6c 4a 6a 6c   uNB.dOZCkZNZlJjl
0120  78 78 77 76 43 72 6a 6a 73 36 47 43 47 57 30 42   xxwvCrjjs6GCGW0B
0130  6f 70 57 6f 32 4d 58 49 63 71 58 2f 64 6b 5a 7a   opWo2MXIcqX/dkZz
0140  49 42 52 51 52 54 70 4b 53 43 52 2b 4f 55 4d 4e   IBRQRTpKSCR+OUMN
0150  31 33 54 70 0a 75 77 38 51 45 42 78 6a 4f 64 47   13Tp.uw8QEBxjOdG
0160  48 35 50 4e 78 71 54 42 35 6b 78 78 4d 41 55 4f   H5PNxqTB5kxxMAUO
0170  59 6c 34 6e 61 54 38 47 52 44 4d 4f 44 71 69 2b   Yl4naT8GRDMODqi+
0180  59 33 52 53 6b 71 6a 79 43 74 69 4f 64 54 4c 6a   Y3RSkqjyCtiOdTLj
0190  67 78 53 2f 6a 0a 4b 2b 42 48 37 46 66 69 73 44   gxS/j.K+BH7F  sD
01a0  41 6d 7a 54 74 67 57 45 6f 53 7a 78 58 68 6c 56   AmzTtgWEoSzxXhlV
01b0  71 36 6f 4e 36 65 33 65 34 35 75 2b 72 78 66 50   q6oN6e3e45u+rxfP
01c0  64 36 67 2b 63 70 44 77 61 47 57 65 42 31 79 5a   d6g+cpDwaGWeB1yZ
01d0  38 42 75 6f 62 61 0a 42 77 49 44 41 51 41 42 0a   8Buoba.BwIDAQAB.
01e0  2d 2d 2d 2d 2d 45 4e 44 20 50 55 42 4c 49 43 20   -----END PUBLIC
01f0  4b 45 59 2d 2d 2d 2d 2d 0a                        KEY-----.
```

Fonte: O autor (2020).

3. O aplicativo do celular gera de forma aleatória uma chave privada *AES*, criptografa utilizando a chave pública *RSA* recebida anteriormente e envia ao *ESP-32*, conforme mostra a Figura 26;

Figura 26 – Envio da chave privada *AES* pelo aplicativo do celular

```
No.  Time       Source        Destination     Protocol  Length  Info
171  8.453234   192.168.15.7  192.168.15.21   TCP       405     56524 → 49152 [PSH, ACK] Seq=1 Ack=452 Win=63789 Len=351

Frame 171: 405 bytes on wire (3240 bits), 405 bytes captured (3240 bits) on interface \Device\NPF_{0E5BC0EB-5433-40DB-8BE6-AA782E45028A}, id 0
Ethernet II, Src: Micro-St_38:4e:28 (d8:cb:8a:38:4e:28), Dst: Espressi_89:15:25 (b4:e6:2d:89:15:25)
Internet Protocol Version 4, Src: 192.168.15.7, Dst: 192.168.15.21
Transmission Control Protocol, Src Port: 56524, Dst Port: 49152, Seq: 1, Ack: 452, Len: 351
Data (351 bytes)

0000  b4 e6 2d 89 15 25 d8 cb 8a 38 4e 28 08 00 45 00   ..-..%...8N(..E.
0010  01 87 c7 e3 40 00 80 06 00 00 c0 a8 0f 07 c0 a8   ....@...........
0020  0f 15 dc cc c0 00 b0 b9 65 90 00 00 1b 46 50 18   ........e....FP.
0030  f9 2d a0 e6 00 00 41 45 53 4b 45 59 3a 41 62 79   .-....AESKEY:Aby
0040  65 58 69 59 2f 6d 71 34 33 6f 38 31 64 49 56 6e   eXiY/mq43o81dIVn
0050  47 4f 65 65 63 79 6a 41 34 36 6d 31 79 6c 58 78   GOeecyjA46m1ylXx
0060  35 51 54 45 34 38 72 4b 46 55 77 6d 51 74 72 42   5QTE48rKFUwmQtrB
0070  66 2b 39 2f 6e 2b 6b 6e 72 52 2b 6c 72 4d 34 56   f+9/n+knrR+lrM4V
0080  6f 36 56 68 57 4c 50 6d 4d 37 43 78 6d 4f 67 65   o6VhWLPmM7CxmOge
0090  35 50 2b 63 54 6d 42 61 78 45 56 59 41 66 31 75   5P+cTmBaxEVYAf1u
00a0  34 49 59 35 56 4f 4e 4f 59 56 6d 79 79 44 52 42   4IY5VONOYVmyyDRB
00b0  62 36 34 57 42 77 59 58 4c 35 63 30 7a 61 30 57   b64WBwYXL5c0za0W
00c0  38 2b 65 38 71 70 2b 37 73 4c 66 70 57 6a 6a 68   8+e8qp+7sLfpWjjh
00d0  38 4f 69 63 65 63 38 33 37 4f 57 66 49 76 31 75   8Oicec837OWfIv1u
00e0  73 43 62 7a 30 6d 42 62 6f 57 70 31 2b 31 6d 34   sCbz0mBboWp1+1m4
00f0  63 50 77 36 6b 2b 6a 48 58 54 33 37 43 56 37 59   cPw6k+jHXT37CV7Y
0100  4f 6f 69 43 73 6d 45 52 55 59 4d 67 46 73 6c 50   OoiCsmERUYMgFslP
0110  37 50 36 6a 6e 78 43 4d 6a 70 64 79 66 47 4a 78   7P6jnxCMjpdyfGJx
0120  6c 49 73 4e 4e 31 4e 38 4f 51 64 42 30 33 52 65   lIsNN1N8OQdB03Re
0130  45 50 71 61 4a 5a 4d 6f 50 37 4b 4b 34 37 75 77   EPqaJZMoP7KK47uw
0140  4a 49 47 31 70 73 44 4a 6f 61 4c 56 79 64 79 39   JIG1psDJoaLVydy9
0150  78 49 78 6e 76 6b 44 68 74 2f 2f 57 37 41 63 63   xIxnvkDht//W7Acc
0160  47 68 65 54 39 35 63 79 5a 71 2b 55 54 4f 65 54   GheT95cyZq+UTOeT
0170  44 74 55 70 31 4d 44 37 49 56 30 56 6a 61 4f 4c   DtUp1MD7IV0VjaOL
0180  34 6a 55 54 65 48 71 78 6f 7a 79 31 61 42 7a 4e   4jUTeHqxozy1aBzN
0190  34 4a 77 3d 3d                                    4Jw==
```

Fonte: O autor (2020).

4. Quando o usuário insere as informações de *login* do roteador, esta informação é criptografada utilizando *AES* e convertida para *Base64* de forma a preservar todos os *bits* na transmissão, a qual na sequência é enviada ao microcontrolador, conforme mostra a Figura 27;

Figura 27 – Envio dos dados de roteador criptografados pelo aplicativo do celular

```
No.  Time       Source        Destination     Protocol  Length  Info
435  26.448359  192.168.15.7  192.168.15.21   TCP       134     56537 → 49152 [PSH, ACK] Seq=1 Ack=19 Win=64222 Len=80

Frame 435: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits) on interface \Device\NPF_{0E5BC0EB-5433-40DB-8BE6-AA782E45028A}, id 0
Ethernet II, Src: Micro-St_38:4e:28 (d8:cb:8a:38:4e:28), Dst: Espressi_89:15:25 (b4:e6:2d:89:15:25)
Internet Protocol Version 4, Src: 192.168.15.7, Dst: 192.168.15.21
Transmission Control Protocol, Src Port: 56537, Dst Port: 49152, Seq: 1, Ack: 19, Len: 80
Data (80 bytes)

0000  b4 e6 2d 89 15 25 d8 cb 8a 38 4e 28 08 00 45 00   ..-..%...8N(..E.
0010  00 78 c7 fc 40 00 80 06 00 00 c0 a8 0f 07 c0 a8   .x..@...........
0020  0f 15 dc d9 c0 00 f2 f2 dd d4 00 00 19 d9 50 18   ..............P.
0030  fa de 9f d7 00 00 52 4f 55 54 45 52 44 41 54 41   ......ROUTERDATA
0040  3a 67 4c 4d 63 2f 45 43 66 73 43 36 72 38 49 64   :gLMc/ECfsC6r8Id
0050  70 6a 2b 74 66 4a 77 3d 3d 2c 6b 4b 4a 6e 77 74   pj+tfJw==,kKJnwt
0060  68 41 41 74 6c 64 35 32 53 34 6a 34 5a 63 68 5a   hAAtld52S4j4ZchZ
0070  41 7a 50 42 52 59 30 49 2b 45 69 68 32 48 70 38   AzPBRY0I+Eih2Hp8
0080  38 52 61 47 49 3d                                 8RaGI=
```

Fonte: O autor (2020).

5. O *ESP-32* então recebe as informações do roteador e grava dentro da memória *flash* as configurações de *login* através do *SPIFFS*, um sistema de arquivos construído para ler e gravar em memórias *flash* externas em microcontroladores. Após o salvamento dos dados na memória *flash*, o *ESP-32* auto-reinicia e conecta-se à rede.

Após o *ESP-32* conectar-se à rede fornecida pelo usuário, é possível utilizar o celular nesta mesma rede de modo a poder receber dados da internet em simultaneidade com os dados provenientes do microcontrolador, assumindo que o roteador tenha acesso à internet.

Para que o microcontrolador e o celular consigam se comunicar entre si usando a rede *WiFi* do roteador, o *ESP-32* envia seu endereço *IP* através de datagramas *UDP* no endereço *multicast 224.0.0.1*, que está mostrado na Figura 28.

Figura 28 – Envio de datagramas *UDP* pelo microcontrolador

```
No.   Time        Source          Destination     Protocol  Length  Info
18    6.041730    192.168.15.21   224.0.0.1       UDP       43      49153 → 49153 Len=1

Frame 18: 43 bytes on wire (344 bits), 43 bytes captured (344 bits) on interface \Device\NPF_{CFA80614-B1CE-44C2-8085-51431582B74A}, id 0
Ethernet II, Src: Espressi_89:15:25 (b4:e6:2d:89:15:25), Dst: IPv4mcast_01 (01:00:5e:00:00:01)
Internet Protocol Version 4, Src: 192.168.15.21, Dst: 224.0.0.1
User Datagram Protocol, Src Port: 49153, Dst Port: 49153
Data (1 byte)

0000  01 00 5e 00 00 01 b4 e6 2d 89 15 25 08 00 45 00   ..^.....-..%..E.
0010  00 1d 01 43 00 00     11 0a  ce c0 a8 0f 15 e0 00    ...C...........
0020  00 01 c0 01 c0 01 00 09 9f 19 31                    ..........1

No.   Time        Source          Destination     Protocol  Length  Info
19    7.065693    192.168.15.21   224.0.0.1       UDP       43      49153 → 49153 Len=1

Frame 19: 43 bytes on wire (344 bits), 43 bytes captured (344 bits) on interface \Device\NPF_{CFA80614-B1CE-44C2-8085-51431582B74A}, id 0
Ethernet II, Src: Espressi_89:15:25 (b4:e6:2d:89:15:25), Dst: IPv4mcast_01 (01:00:5e:00:00:01)
Internet Protocol Version 4, Src: 192.168.15.21, Dst: 224.0.0.1
User Datagram Protocol, Src Port: 49153, Dst Port: 49153
Data (1 byte)

0000  01 00 5e 00 00 01 b4 e6 2d 89 15 25 08 00 45 00   ..^.....-..%..E.
0010  00 1d 01 44 00 00     11 0a cd c0 a8 0f  15 e0 00    ...D...........
0020  00 01 c0 01 c0 01 00 09 9f 19 31                    ..........1

No.   Time        Source          Destination     Protocol  Length  Info
20    7.987313    192.168.15.21   224.0.0.1       UDP       43      49153 → 49153 Len=1

Frame 20: 43 bytes on wire (344 bits), 43 bytes captured (344 bits) on interface \Device\NPF_{CFA80614-B1CE-44C2-8085-51431582B74A}, id 0
Ethernet II, Src: Espressi_89:15:25 (b4:e6:2d:89:15:25), Dst: IPv4mcast_01 (01:00:5e:00:00:01)
Internet Protocol Version 4, Src: 192.168.15.21, Dst: 224.0.0.1
User Datagram Protocol, Src Port: 49153, Dst Port: 49153
Data (1 byte)

0000  01 00 5e 00 00 01 b4 e6 2d 89 15 25 08 00 45 00   ..^.....-..%..E.
0010  00 1d 01 45 00 00     11 0a cc  c0 a8 0f 15 e0 00    ...E...........
0020  00 01 c0 01 c0 01 00 09 9f 19 31                    ..........1
```

Fonte: O autor (2020).

Através destes passos, verificamos no aplicativo do celular a conexão estabelecida, conforme mostrado na Figura 29.

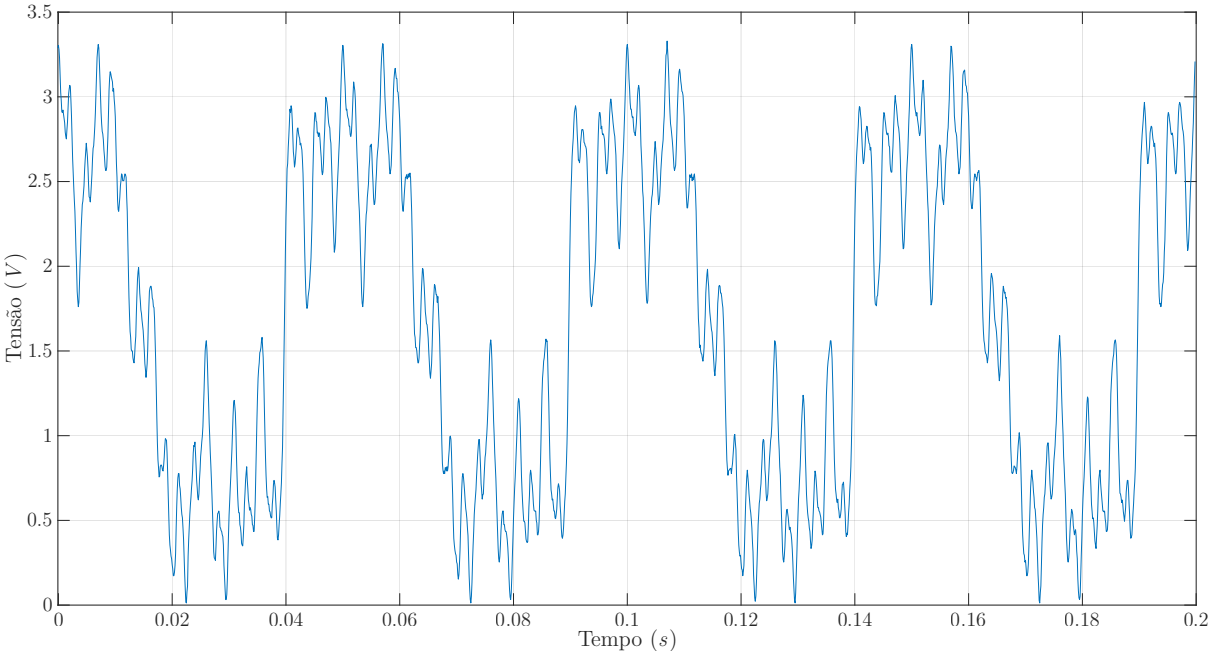
Figura 29 – Aplicativo indicando conexão com o *ESP-32*

Fonte: O autor (2020).

4.2 Testes de funcionalidade com gerador de sinal

Utilizando o gerador de sinais, o primeiro canal foi definido para gerar a senoide descrita na subseção 3.3.3 e sua saída está representada na Figura 30. Esta onda visa simular a vibração de uma peça rotativa e também incluir efeitos de ruídos, respeitando-se os limites de 0 a 3,3 V de entrada do *ADC* do *NUCLEO-G474RE*.

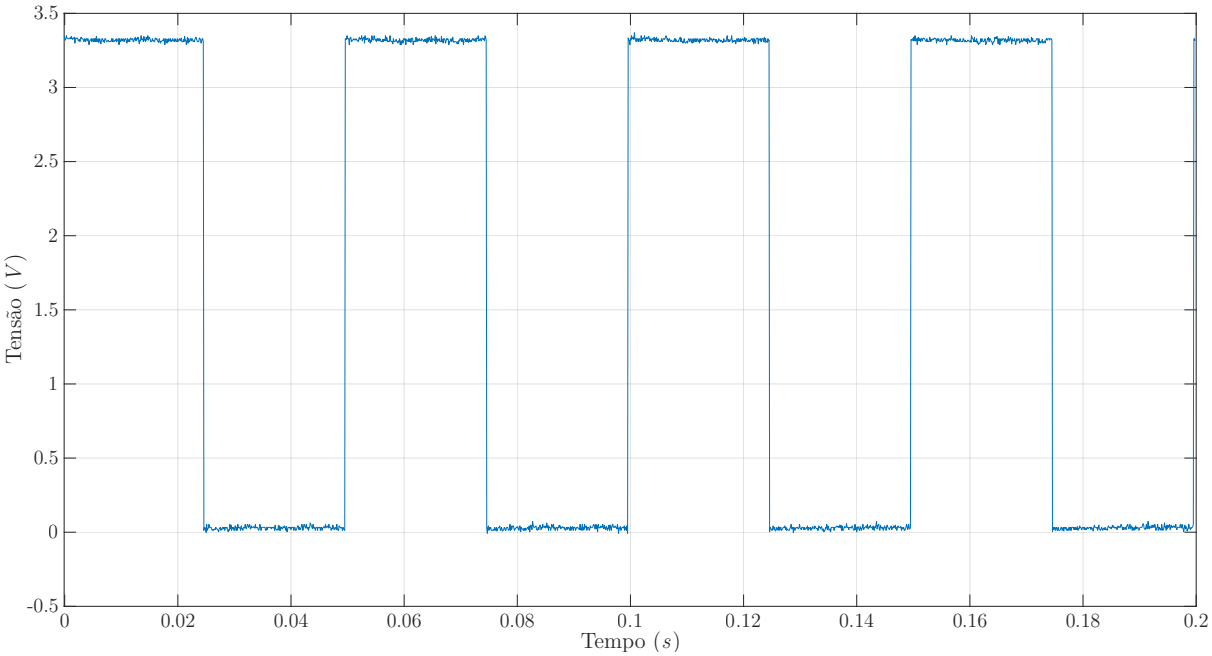
Figura 30 – Onda senoidal com ruído na saída do gerador de sinal



Fonte: O autor (2020).

No segundo canal, é gerado uma onda quadrada com a mesma frequência e amplitude do primeiro, para simular o comportamento do sensor *Hall*, ilustrado na Figura 31.

Figura 31 – Onda quadrada na saída do gerador de sinal

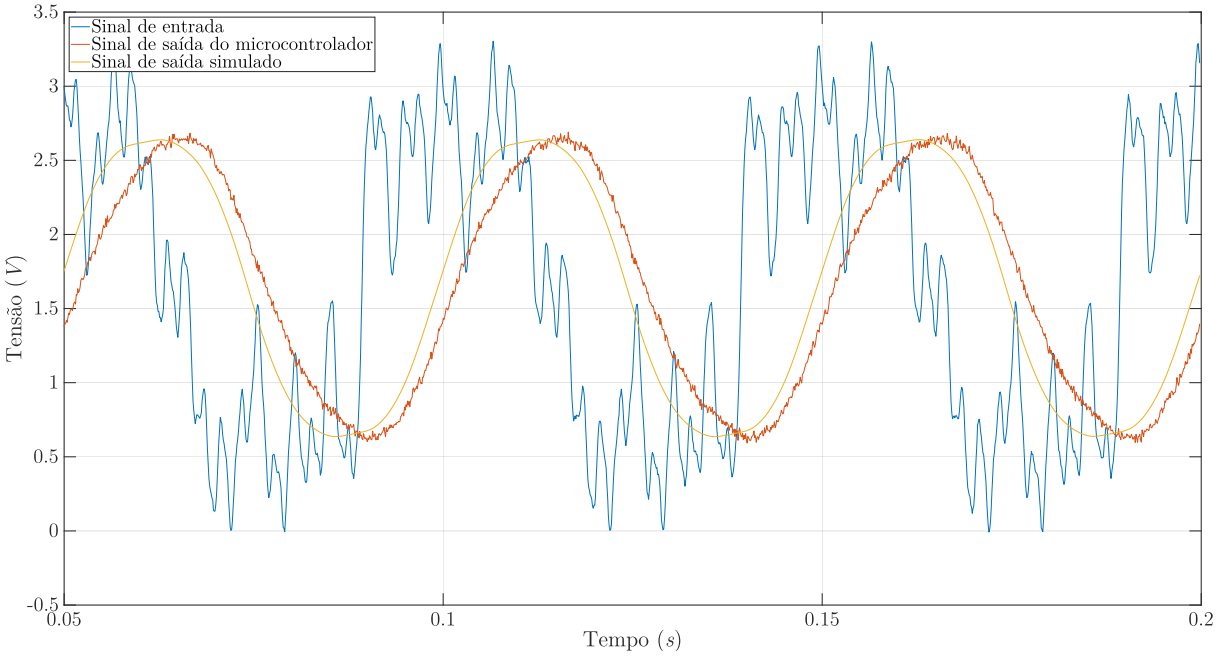


Fonte: O autor (2020).

A partir da forma de onda do primeiro canal, fez-se o processo de filtragem digital e a saída resultante foi direcionada à um *GPIO* através de um *DAC*, para poder ser analisado

o comportamento do filtro. A entrada com ruído e a saída filtrada estão mostrados na Figura 32, juntamente com o sinal de saída filtrada através de simulação para fins de comparação.

Figura 32 – Onda senoidal de 20 *Hz* contendo 10 harmônicas de diferentes amplitudes



Fonte: O autor (2020).

A diferença de fase e amplitude entre o sinais estão mostrados na Tabela 6.

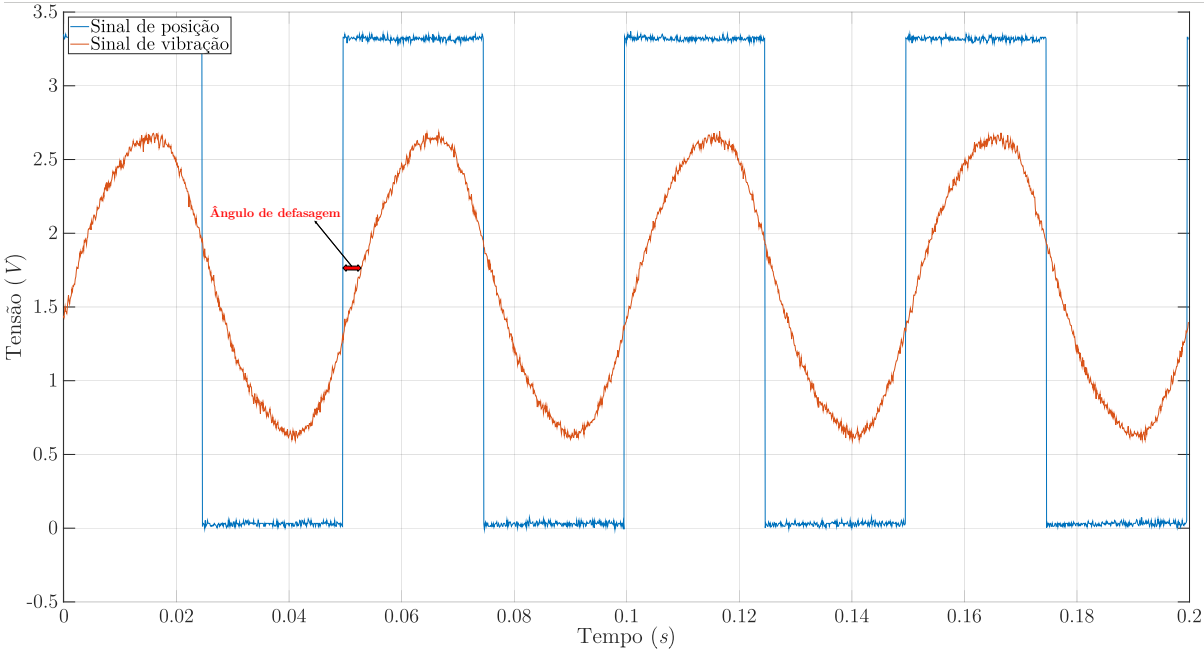
Tabela 6 – Características dos sinais senoidais com ruído e filtrados

Sinal	Tensão *RMS*	Fase relativa
Gerador de sinal	1,9092 V	0°
Microcontrolador	1,7936 V	91,7224°
Simulado	1,7071 V	80,3656°

Fonte: O autor (2020).

Através de testes variando o ângulo de defasagem entre a onda quadrada e a senoide filtrada, foram comparados os resultados obtidos entre microcontrolador e o osciloscópio. A Figura 33 mostra um exemplo para o cálculo do ângulo entre os sinais e a Tabela 7 ilustra os ângulos medidos em ambos dispositivos para diferentes casos de defasagem.

Figura 33 – Onda senoidal simulando vibração e onda quadrada simulando posição



Fonte: O autor (2020).

Tabela 7 – Diferença entre os ângulos medidos no osciloscópio e no microcontrolador

Ângulo medido no osciloscópio	Ângulo medido no microcontrolador	Diferença	Erro percentual
91°	89°	2°	2,20 %
135°	135°	0°	0,00 %
178°	177°	1°	0,56 %
307°	308°	1°	0,32 %

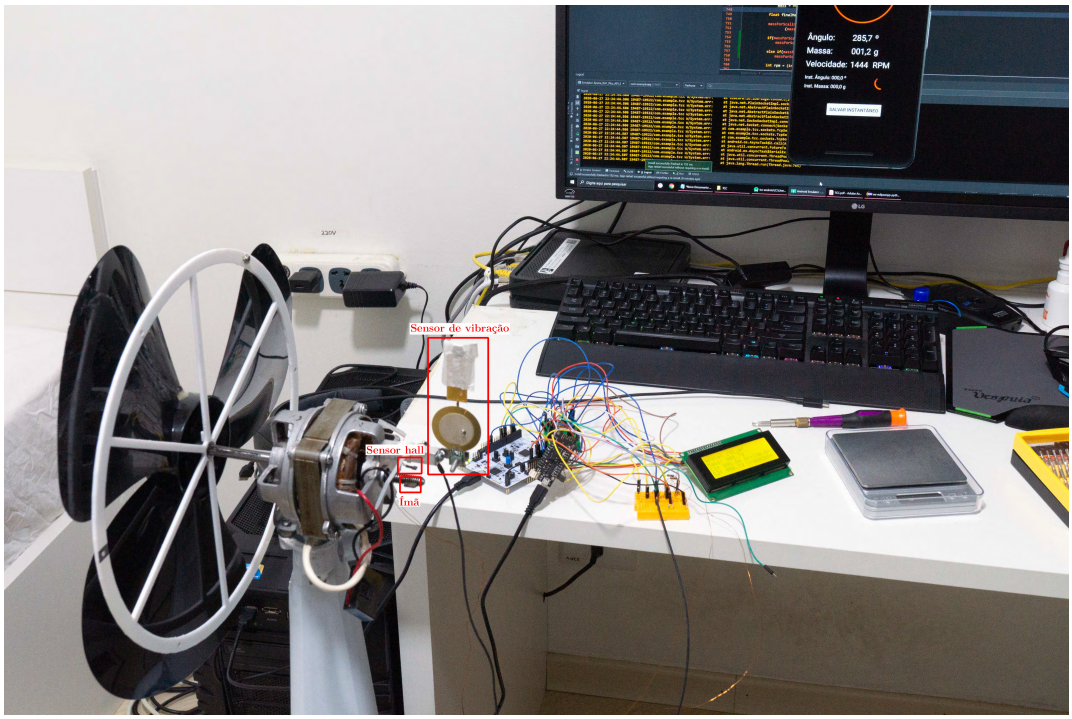
Fonte: O autor (2020).

A diferença de ângulo entre os dispositivos se manteve praticamente constante, indicando que o microcontrolador está apto a fazer medições confiáveis das defasagens entre os sinais do sensor *Hall* e do piezoelétrico. O erro máximo obtido foi 2,20 %, que é suficiente para várias aplicações onde não se requer uma precisão muito grande.

4.3 Testes com ventilador de mesa

Foram instaladas as peças e sensores no ventilador conforme a Figura 34. O ímã (no eixo do motor na parte de trás) foi posicionado de forma a coincidir com a referência da roda de desbalanceamento (listra preta indicada na roda) e o sensor piezoelétrico foi acoplado de forma a maximizar os efeitos de vibração, dentro dos limites da faixa dinâmica do sistema eletrônico embarcado.

Figura 34 – Protótipo montado



Fonte: O autor (2020).

4.3.1 Balanceamento de peças rotativas sem uma peça padrão balanceada de comparação

Introduziu-se um parafuso de 0,7 g em um dos furos da roda de desbalanceamento para simular um desbalanceamento prévio.

Foi utilizado o aplicativo no modo *Método ref. não balanceada*, cuja tela está representada na Figura 35. A tolerância de erro neste método foi definida como 10% da quantidade de massa necessária para a correção do desbalanceamento.

Figura 35 – Tela inicial do *Método ref. não balanceada*

Fonte: O autor (2020).

Inseriu-se outro parafuso na roda de desbalanceamento em três pontos distintos, distanciados de 120° um do outro, com o resultado das três medições mostrado na Figura 36.

Figura 36 – *Método ref. não balanceada* após três medições

Fonte: O autor (2020).

Após a terceira medição, apresentou-se o ângulo em que se deve inserir uma massa para a correção do desbalanceamento. Tirou-se o parafuso da terceira medição e foi colocado no ângulo indicado pelo celular, para assim determinar-se a quantidade de massa a ser acrescentada ou subtraída caso já não fique balanceado. Obteve-se $-0{,}2$ g, conforme mostrado na Figura 37. O sinal negativo indica que houve um excesso de massa que deveria ser colocado nesse local, e portanto deve ser subtraído uma parcela desta mesma massa.

Figura 37 – Resultado após a quarta medição do *Método ref. não balanceada*

Fonte: O autor (2020).

Portanto, retirou-se o parafuso e colocou-se um pedaço de fita que pesasse 0,5 g, desta vez com o programa apresentando que foi balanceado a peça rotativa, conforme mostra a Figura 38.

Figura 38 – Resultado final do *Método ref. não balanceada*

Fonte: O autor (2020).

4.3.2 Balanceamento de peças rotativas com uma peça padrão balanceada de comparação

Inicialmente foi utilizado um ventilador de mesa com a hélice já balanceada. Para compensar a defasagem de ângulo devido ao filtro digital, introduziu-se uma massa no ponto de referência, e através do aplicativo fixou-se em zero o ângulo lido com a utilização de um *offset*, representado na Figura 39.

Figura 39 – Calibração do ângulo

Fonte: O autor (2020).

Em seguida, iniciou-se o processo de calibração da quantidade de massa do sistema, inserindo-se parafusos na roda de desbalanceamento e gravando os dados no aplicativo para duas massas distintas, uma massa menor no primeiro ponto e uma massa maior no segundo ponto. A Figura 40 mostra a janela de calibração do aplicativo.

Figura 40 – Janela de calibração

Fonte: O autor (2020).

Após a calibração, com o ventilador novamente balanceado, foi inserido um parafuso com uma arruela totalizando 1,6 g posicionado a um ângulo de 240° em relação ao sensor de posição. Em seguida, retirou-se o conjunto de 1,6 g e inseriu-se somente o parafuso de 0,7 g na posição de referência do sensor (0°) para a validação da calibração. Os resultados estão apresentados na Figura 41, com a Figura 42 apresentando o resultado da Figura 41b na tela auxiliar *LCD*.

Figura 41 – Ângulo e massa de correção do desbalanceamento

(a) Resultado para um parafuso com arruela posicionado a 240° da origem

(b) Resultado para somente um parafuso posicionado a 0° da origem

Figura 42 – Display *LCD* com o resultado da calibração utilizando somente um parafuso posicionado a 0° da origem

Fonte: O autor (2020).

Por fim, retirou-se toda a massa da roda de desbalanceamento para eliminar as vibrações, com o programa mostrando que o sistema está balanceado, apresentado na Figura 43.

Figura 43 – Sistema rotativo balanceado

Fonte: O autor (2020).

5 Conclusão

Este trabalho orientou-se no objetivo de desenvolver um sistema eletrônico embarcado para auxiliar no processo de balanceamento dinâmico de peças rotativas. Considerou-se um circuito que fosse o mais abrangente possível e passível de atualizações ou adaptações conforme necessidades de cada sistema e usuário.

O microcontrolador *STM32G474RE* utilizado satisfez os requisitos propostos do projeto, além de eliminar componentes discretos para a etapa de filtragem e manipulação dos sinais de entrada. Por ser de natureza programável, adiciona flexibilidade ao sistema, podendo ser reconfigurado através de códigos os parâmetros do filtro, dos amplificadores e dos comparadores.

Através do microcontrolador *ESP-32*, fez-se a comunicação sem fio de modo a transmitir as informações para dispositivos móveis, adicionando uma flexibilidade ao usuário de modo a não ser necessário estar junto ao sistema para a análise das informações em tempo real.

Com apenas dois microcontroladores e alguns componentes passivos, além de sensores externos que podem ser escolhidos conforme a necessidade da implementação (respeitando-se os limites das entradas dos microcontroladores), o sistema possui topologia simples com dimensões reduzidas, favorecendo a portabilidade e o baixo custo, atingindo-se, assim, os objetivos propostos no trabalho.

5.1 Possibilidade de melhorias futuras

Para melhorias futuras, pode-se considerar o uso da comunicação via *bluetooth*, a integração dos componentes numa única placa de circuito impresso (exceto os sensores), uso de teclado e desenvolvimento de uma interface integrada ao dispositivo, implementação de algoritmos de amplificação automática utilizando os amplificadores operacionais internos do microcontrolador, circuitos de proteção e o desenvolvimento de um invólucro protetor.

Referências

ALIBABA. *Balanceador de rodas automático completo AA-WB205*. 2020. Disponível em: <https://portuguese.alibaba.com/product-detail/full-automatic-wheel-balancer-aa-wb205-60734134528.html>. Acesso em: 31 jul 2020. Citado na página 19.

BARBOSA, E. R. *Filtros Digitais Reconfiguráveis*. 65 p. Monografia (Trabalho de conclusão de curso) — Centro Universitário Positivo, Núcleo de Ciências Exatas e Tecnológicas, Programa de Graduação em Engenharia da Computação, Curitiba, Dezembro 2006. Citado na página 26.

BARRY, R. *Mastering the FreeRTOS™ Real Time Kernel*. [S.l.], 2016. 399 p. Citado na página 28.

CASTRO, F. L. *Criptografia RSA: uma abordagem para professores do ensino básico*. 61 p. Monografia (Trabalho de conclusão de curso) — Universidade Federal do Rio Grande do Sul, Instituto de Matemática, Programa de Graduação em Matemática Pura e Aplicada, Porto Alegre, Dezembro 2014. Citado na página 77.

GURU, B. *Base64 Encode Algorithm*. 2020. Disponível em: <https://base64.guru/learn/base64-algorithm/encode>. Acesso em: 22 jun 2020. Citado na página 30.

INSTRUMENTS, T. *DRV5023 Digital-Switch Hall Effect Sensor*. [S.l.], 2016. (Datasheet). Citado 2 vezes nas páginas 41 e 42.

KUROSE, J. F.; ROSS, K. W. *Redes de computadores e a internet*: uma abordagem top-down. 6. ed. São Paulo: Pearson Education do Brasil, 2014. 65–70 p. Citado 2 vezes nas páginas 31 e 32.

LIMA, A. A. de. *Influência da Desclinação no Efeito Hall Quântico Inteiro*. 55 p. Dissertação (Mestrado) — Universidade Federal de Campina Grande, Fevereiro 2013. Citado na página 75.

MOREIRA, D. C. *Construção e Caracterização do Campo Acústico de Transdutores Ultra-sônico Piezoelétrico de Polarização Variável*. 133 p. Dissertação (Mestrado) — Universidade Federal de São Carlos, Outubro 2008. Citado na página 75.

PROJECTS, D. *#Deal: ESP32 Wemos LoLin32 Lite, more compact ESP32 IoT development board*. 2020. Disponível em: <https://diyprojects.io/product/deal-wemos-lolin32-lite-compact/#.XvLU-yhKguU>. Acesso em: 23 jun 2020. Citado na página 39.

SEDRA, A. S.; SMITH, K. C. *Microeletrônica*. 5. ed. São Paulo: Pearson Education do Brasil, 2012. 47–49 p. Citado 2 vezes nas páginas 24 e 25.

SERVICES, A. W. *The FreeRTOS™ Reference Manual*. 2017. 400 p. Citado na página 28.

SIEMENS. *Introduction to Filters: FIR versus IIR*. 2020. Disponível em: <https://community.sw.siemens.com/s/article/introduction-to-filters-fir-versus-iir>. Acesso em: 23 jun 2020. Citado 2 vezes nas páginas 26 e 27.

STMICROELECTRONICS. *STM32G4 Nucleo-64 boards (MB1367)*. [S.l.], 2019. (User Manual). Citado na página 38.

STMICROELECTRONICS. *STM32G474xB STM32G474xC STM32G474xE*. [S.l.], 2019. (Datasheet - production data). Citado na página 33.

STMICROELECTRONICS. *RM0440 Reference manual*. [S.l.], 2020. (STM32G4 Series advanced Arm®-based 32-bit MCUs). Citado 2 vezes nas páginas 76 e 77.

SYSTEMS, E. *ESP32 Series*. [S.l.], 2020. (Datasheet). Citado na página 36.

TOOLKITS, D. *What is Base64 and How Does It Work?* 2020. Disponível em: <https://www.developertoolkits.com/base64/what-is-base64>. Acesso em: 22 jun 2020. Citado na página 30.

Apêndices

APÊNDICE A – Conceitos referentes aos sensores de vibração e posição

Introdução

O objetivo deste apêndice é familiarizar o leitor com a teoria por trás dos sensores utilizados no trabalho. Serão apresentados resumidamente os conceitos que regem o funcionamento dos sensores.

A.1 Piezoeletricidade

Alguns materiais, quando submetidos à forças externas que os tensionam mecanicamente, podem ficar com cargas de polaridade opostas em suas extremidades, deixando-o polarizado. Este efeito é denominado de piezoelétrico direto (MOREIRA, 2008).

Há também o efeito piezoelétrico inverso, que faz com que o material deforme-se na presença de um campo elétrico (MOREIRA, 2008).

As propriedades piezoelétricas dos materiais têm origem nas estruturas cristalinas das quais não possuem centro de simetria, que quando são deformadas, há uma redistribuição dos elementos com carga de modo a surgir-se cargas em sua superfície (MOREIRA, 2008).

A.2 Efeito Hall

O efeito *Hall* é assim nomeado em homenagem ao físico estadunidense *Edwin Herbert Hall*, que descobriu o fenômeno.

Este efeito consiste no acúmulo de cargas elétricas na periferia de um material condutor, quando este conduz corrente elétrica e está submetido à um campo magnético aplicado perpendicularmente à direção da corrente. Através das cargas acumuladas, é possível determinar o sinal da carga transportada no material (LIMA, 2013).

APÊNDICE B – Criptografias

Introdução

Para a comunicação entre dispositivos através de redes, é importante que os dados privados dos usuários estejam seguros. Será detalhado brevemente as criptografias utilizadas neste trabalho.

B.1 *AES*

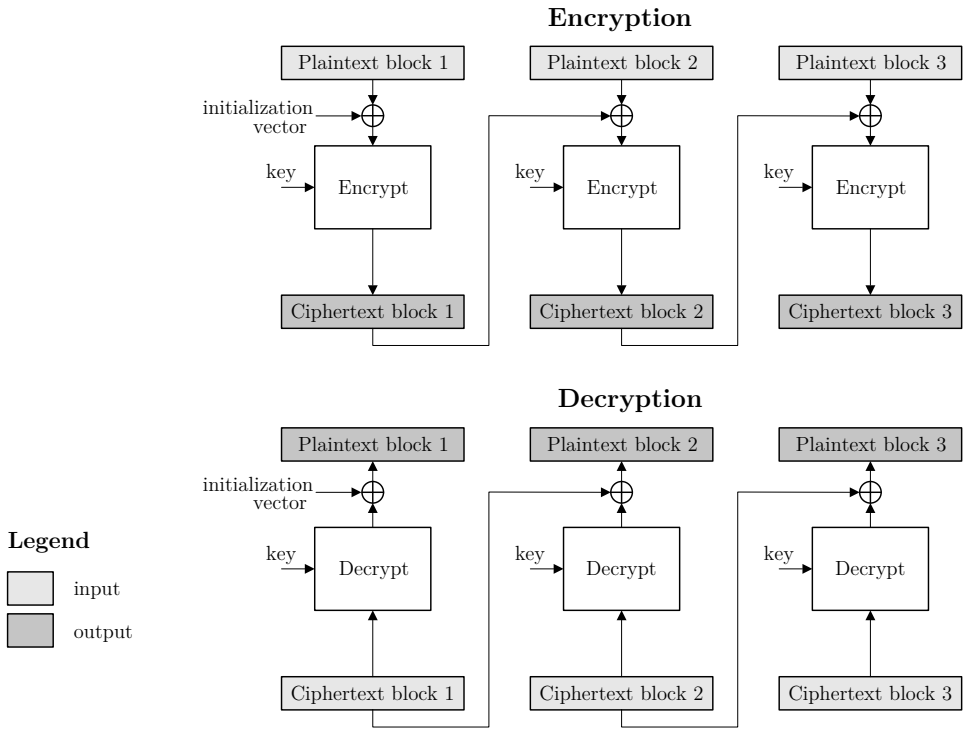
A criptografia *AES* é muito utilizada por apresentar segurança e rapidez. Consiste num algoritmo que implementa chaves privadas, sendo que ambas as partes precisam ter a mesma chave para criptografar e descriptografar a mensagem. Portanto, há de ter um acordo prévio entre as partes, não sendo seguro passar as chaves através de um meio de comunicação público.

B.1.1 *AES* modo *CBC*

Neste modo, a saída de cada bloco de texto criptografado (com a chave privada) é concatenada com a entrada do bloco seguinte. Para que cada mensagem criptografada seja única, um vetor de inicialização (também chamado *IV*) é gerado de forma aleatória e usado durante o processamento do primeiro bloco. O processo de descriptografamento é simplesmente o inverso do criptografamento (STMICROELECTRONICS, 2020).

O modo de criptografia *CBC* pode ser visualizado graficamente pela Figura 44.

Figura 44 – Criptografia *AES* modo *CBC*



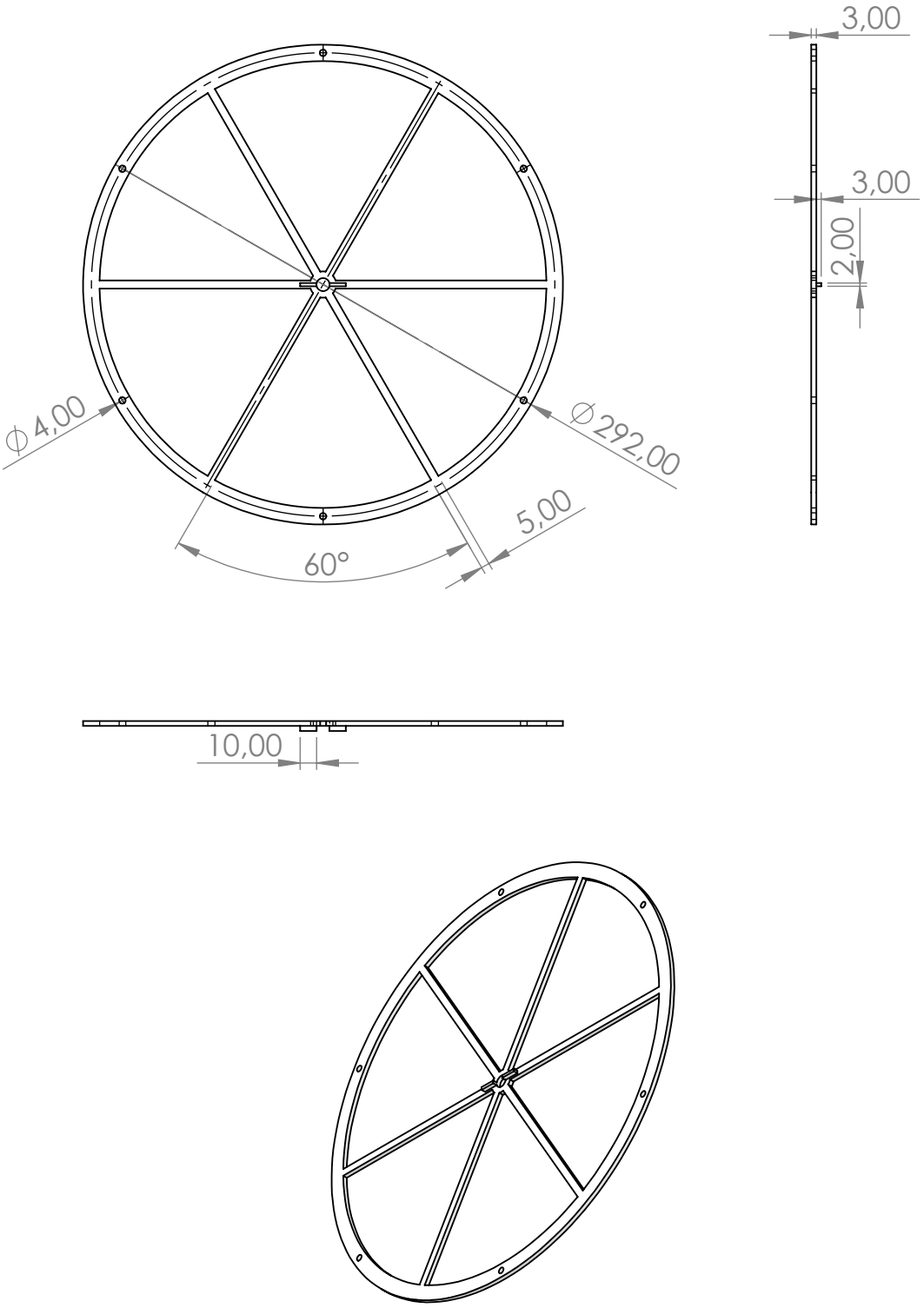
Fonte: Adaptado de STMicroelectronics (2020).

B.2 *RSA*

A criptografia *RSA* é baseada em pares chave pública-privada (assimétrica). É também bastante usada em autenticações ou troca de senhas na rede, devido à utilização de chaves públicas para a criptografia dos dados, sendo que somente quem possui a chave privada do par poderá descriptografar o conteúdo.

Esta criptografia consiste num algoritmo que utiliza dois números primos p e q, sendo que no processo de decifrar a mensagem é necessário a determinação destes. A chave pública contém o produto de p e q, sendo necessário portanto a fatoração do número que representa este produto para a determinação da mensagem (CASTRO, 2014). Porém, pelo fato dos números primos do algoritmo possuirem uma quantidade elevada de *bits*, é computacionalmente complexo realizar a quebra da criptografia.

APÊNDICE C – Detalhamento técnico da roda de desbalanceamento



Roda de desbalanceamento

Projetista: Helder Sales	Código: N/A
Setor: N/A	Material: PLA (Ácido Polilático)
	Massa: 44 g
	Escala: 1:4 A4