UNIVERSIDADE FEDERAL DO PARANÁ

ERIK NAYAN ONORIO DOS SANTOS

# OSCILOSCÓPIO DIGITAL DE BAIXO CUSTO, BASEADO EM KIT MICROCONTROLADOR STM32, COM TELA EM APLICATIVO *ANDROID* E ENVIO DE DADOS VIA REDE SEM FIO

CURITIBA

2019

ERIK NAYAN ONORIO DOS SANTOS

# OSCILOSCÓPIO DIGITAL DE BAIXO CUSTO, BASEADO EM KIT MICROCONTROLADOR STM32, COM TELA EM APLICATIVO *ANDROID* E ENVIO DE DADOS VIA REDE SEM FIO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Federal do Paraná como requisito à obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Luis Henrique Assumpção Lolis

CURITIBA

2019

**TERMO DE APROVAÇÃO**

**ERIK NAYAN ONORIO DOS SANTOS**

**OSCILOSCÓPIO DIGITAL DE BAIXO CUSTO, BASEADO EM KIT MICROCONTROLADOR STM32, COM TELA EM APLICATIVO *ANDROID* E ENVIO DE DADOS VIA REDE SEM FIO**

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Federal do Paraná como requisito à obtenção do grau de Engenheiro Eletricista, pela seguinte banca examinadora:

---

Prof. Dr. Luis Henrique Assumpção Lolis
Departamento de Engenharia Elétrica,
UFPR

---

Prof. Dr. Henri Frederico Eberspacher
Departamento de Engenharia Elétrica,
UFPR

---

Prof. Dr. Eduardo Gonçalves de Lima
Departamento de Engenharia Elétrica,
UFPR

Curitiba, 05 de Dezembro de 2019

*À memória de Magdalena Izaura Onorio.*

## AGRADECIMENTOS

Agradeço primeiramente a minha mãe, Katia, por sempre ter me dado o suporte necessário para que eu pudesse focar ao máximo em minha formação. Agradeço também a todo apoio que recebi de meu tio e de minha tia, Edison e Regina, os quais sempre estiveram do meu lado me ajudando a alcançar meus objetivos.

Agradeço também a todos os meus amigos, especialmente aqueles que foram meus colegas durante o curso técnico e na graduação, por todas as trocas de ideias, ensinamentos, dias de luta e dias de glória durante todos estes anos.

Agradeço aos professores que contribuíram para o meu crescimento, tanto pessoal quanto acadêmico, ao compartilharem seu conhecimento e suas experiências de vida. Obrigado especialmente aos professores Dr. Luis Henrique Assumpção Lolis e Dr. Ewaldo Luiz de Mattos Mehl, por todo o apoio, confiança e incentivo.

Finalmente, deixo os meus agradecimentos a todas as pessoas que passaram pelo meu caminho e contribuíram de alguma forma em minha jornada.

*"La pluralité des voix n'est pas une preuve qui vaille rien pour les vérités un peu malaisées à découvrir, à cause qu'il est bien plus vraisemblable qu'un homme seul les ait rencontrées que tout un peuple."*
*(René Descartes, Discours de la méthode)*

## RESUMO

A internet das coisas (*IoT*) está em forte expansão, assim como a consolidação do uso de aplicativos para *smartphone* está cada vez mais evidente, principalmente no Brasil. Em dissonância com estas tendências, pouco estímulo ao uso destas tecnologias é observado no ambiente universitário, nem mesmo em cursos de cunho científico, como da Engenharia Elétrica. Neste contexto, verifica-se a baixa disponibilidade e o alto custo de equipamentos de instrumentação, em específico os osciloscópios, os quais são fundamentais para realização de práticas de laboratório e concepção de protótipos. Assim sendo, o objetivo deste estudo é desenvolver um osciloscópio digital controlado por aplicativo *Android* via rede sem-fio. Utilizando um kit de desenvolvimento STM32 e um servidor local, este projeto propõe um conceito de osciloscópio de baixo custo, desempenho compatível com as exigências de disciplinas de laboratório na graduação e alta conectividade. Os resultados obtidos demonstram a funcionalidade do sistema, na medida em que formas de onda de frequências distintas puderam ser traçadas com ótima resolução na tela do aplicativo, alcançando também uma taxa de amostragem suficientemente alta para viabilizar o uso da solução no meio acadêmico.



**Palavras-chaves**: Osciloscópio. IoT. STM32. Sem fio. Aplicativo.

## ABSTRACT

The Internet of things (IoT) is booming, as well as the consolidation of mobile apps usage is increasingly evident, especially in Brazil. In dissonance with these trends, almost no incentive to the use of these technologies is observed in the university environment, not even in scientific courses, such as Electrical Engineering. In this context, the low availability and high cost of instrumentation equipment is noticeable, particularly for oscilloscopes, which are fundamental for carrying out laboratory practices and prototype design. Therefore, the purpose of this study is to develop an oscilloscope controlled by Android application via wireless network. This project proposes a low cost, convincing performance and high connectivity oscilloscope concept, employing an STM32 development kit and a local server. The results obtained show the functionality of the system, as waveforms of different frequencies could be charted with optimal resolution on the app screen, also reaching a sufficiently high sampling rate to enable the use of the solution in academia.



**Key-words**: Oscilloscope. IoT. STM32. Wireless. App.

## LISTA DE ILUSTRAÇÕES

## LISTA DE ABREVIATURAS E DE SIGLAS

**ACK** *Acknowledge*

**ADC** *Analog-to-Digital converter*

**AHB** *Advanced High-Performance Bus*

**API** *Application Programming Interface*

**AT** *Attention*

**DMA** *Direct memory access*

**DNS** *Domain Name System*

**DSO** *Digital Storage Oscilloscope*

**FTP** *File Transfer Protocol*

**GCC** *GNU Compiler Collection*

**GDB** *GNU Debugger*

**GPIO** *General Purpose Input/Output*

**HAL** *Hardware Abstraction Layer*

**HTML** *Hypertext Markup Language*

**HTTP** *Hypertext Transfer Protocol*

**I2C** *Inter-Integrated Circuit*

**IDE** *Integrated Development Environment*

**IP** *Internet Protocol*

**IWIN** *Inventek Systems Wireless Interoperability Network*

**IoT** *Internet of Things*

**LCD** *Liquid-Crystal Display*

**LED** *Light-Emitting Diode*

**MCU** *Microcontroller Unit*

**MSS** *Maximum Segment Size*

**MTU** *Maximum Transmission Unit*

**NFC** *Near Field Communication*

**PHP** *PHP: Hypertext Preprocessor*

**RF** *Radio frequency*

**SPI** *Serial Peripheral Interface*

**SRAM** *Static Random Access Memory*

**SSH** *Secure Shell*

**TCP** *Transmission Control Protocol*

**UART** *Universal asynchronous receiver transmitter*

**USART** *Universal synchronous asynchronous receiver transmitter*

**USB** *Universal Serial Bus*

**XML** *Extensible Markup Language*

**SUMÁRIO**

# 1 INTRODUÇÃO

A expansão da Internet das Coisas (*IoT*, do inglês *Internet of Things*) tem feito com que cada vez mais objetos conectados - principalmente a redes sem fio - façam parte do cotidiano das pessoas. Dentre os principais fatores que contribuem para esta expansão, destaca-se a diminuição do custo e do tamanho de sensores e microcontroladores, o que juntamente com o aumento da conectividade sem fio nos ambientes públicos e privados, proporciona um cenário ideal para que cada vez mais dispositivos "inteligentes" sejam utilizados nas mais variadas aplicações.

Paralelamente à expensão da IoT, pode-se verificar uma massiva popularização dos *smartphones* nos últimos anos, os quais vêm se tornando cada vez mais robustos em termos de conexão e capacidade de processamento. Isto proporciona um "casamento ideal" com os objetos conectados, já que aplicativos para *smartphone* (ou, comumente, *apps*), permitem a comunicação com objetos conectados de maneira fácil e interativa, atraindo assim a atenção de um grande público alvo, especialmente no Brasil. Segundo a 30ª Pesquisa Anual do FGVcia da FGV/EAESP (MEIRELLES, 2019), existem 230 milhões de *smartphones* em uso no país (número superior a 1 aparelho por habitante). Ao mesmo tempo, de acordo com o relatório *Spotlight on Consumer App Usage* da *App Annie* (ANNIE, 2017), o brasileiro usa cerca de 10 aplicativos a cada 24h, liderando a estatística mundial.

Analisando este contexto dentro das universidades e laboratórios acadêmicos do Brasil, é natural que estudantes e professores estejam cada vez mais conectados, beneficiando-se de aplicativos de celular para facilitar a realização das mais diversas tarefas. Contudo, mesmo em um ambiente como o do curso de Engenharia Elétrica, onde a tecnologia esta intrinsecamente presente, ainda há muito pouco incentivo à incorporação da *IoT* e dos aplicativos para *smartphone*, sobretudo no que diz respeito a integração com equipamentos e aparelhos didáticos, o que poderia favorecer o processo de aprendizado e impulsionar a realização de protótipos por parte dos alunos.

Logo, surge a ideia de desenvolver um sistema que seja proveitoso ao corpo discente e docente e que reúna as qualidades da *IoT* e dos aplicativos para *smartphone*: um osciloscópio digital monitorado por aplicativo android via rede sem fio.

## 1.1 PROBLEMA

O osciloscópio é uma ferramenta recorrente no arsenal dos estudantes e dos profissionais da área de Engenharia Elétrica e Eletrônica. Este instrumento possibilita que formas de onda de sinais eletrônicos sejam visualizadas, facilitando a compreensão do funcionamento do circuito e a identificação de eventuais problemas que possam estar ocorrendo no mesmo.

Todavia, é necessário frisar que tal aparelho costuma ter um custo elevado, dificultando o acesso a esse dispositivo fora de laboratórios universitários/industriais, fazendo com que muitos estudantes e hobistas não possuam este poderoso recurso de instrumentação para realização de seus projetos. Mesmo dentro das universidades, é comum que os osciloscópios sejam compartilhados por 2 ou mais alunos durante as atividades práticas, prejudicando o aprendizado com o equipamento. Além disso, as interfaces de controle dos osciloscópios costumam ser demasiadamente complexas, assim como a sua conectividade costuma ser limitada ou inexistente, o que diverge das tendências tecnológicas apresentadas anteriormente.

## 1.2 OBJETIVOS

### 1.2.1 Objetivo Geral

Implementar um osciloscópio digital monitorado através de um aplicativo para sistema *Android*, utilizando para tal o conversor analógico-digital e o módulo *Wi-Fi* presentes no kit de desenvolvimento B-L475E-IOT01A da *STMicroelectronics*, o qual enviará dados para um servidor configurado em uma máquina com sistema operacional *Linux*.

### 1.2.2 Objetivos Específicos

- Realizar a amostragem e conversão de sinais utilizando o ADC (Conversor analógico-digital) do microcontrolador STM32L475;
- Implementar de um cliente HTTP (*Hypertext Transfer Protocol*) utilizando o módulo *Wi-Fi* do kit B-L475E-IOT01A;

- Implementar um servidor HTTP em uma máquina equipada com sistema operacional *Linux*;
- Desenvolver um *script* do lado do servidor para tratar as requisições oriundas de clientes;
- Integrar uma biblioteca de gráficos *JavaScript* no servidor para exibição dos dados;
- Desenvolver um aplicativo *Android* para visualização dos sinais e pilotagem remota do sistema.

## 1.3 JUSTIFICATIVA

A proposta de um osciloscópio digital monitorado por aplicativo *Android* via rede sem fio visa fornecer uma alternativa para que alunos, pesquisadores e até mesmo os profissionais da área de sistemas embarcados possam usufruir de um equipamento eficiente, compacto, de custo reduzido e de fácil pilotagem através de uma interface simples e intuitiva para *smartphone*.

## 1.4 ESTRUTURA DO TRABALHO

O próximo capítulo deste estudo é uma revisão teórica de elementos essenciais para compreensão deste projeto. Em seguida, o terceiro capítulo do documento apresenta os materiais e métodos utilizados para a elaboração do trabalho. Depois, o quarto capítulo explica as estratégias e técnicas empregadas para o desenvolvimento do projeto. Logo após, o quinto capítulo apresenta os resultados obtidos ao final da realização deste trabalho. Por fim, uma conclusão é feita apontando os êxitos do projeto e as perspectivas de evoluções futuras.

# 2 REVISÃO BIBLIOGRÁFICA

## 2.1 PRINCÍPIOS DO OSCILOSCÓPIO

O osciloscópio é um instrumento de laboratório comumente usado para exibir e analisar formas de onda de sinais eletrônicos. Na prática, o dispositivo desenha um gráfico da tensão instantânea do sinal de entrada em função do tempo (ROUSE, 2005).

Um osciloscópio típico pode exibir formas de onda de corrente alternada (CA) ou corrente contínua (CC). Quanto a banda passante, os osciloscópios mais modernos podem chegar a medir sinais de frequências da ordem de 100 giga-hertz (GHz) (LECROY, 2019). A exibição costuma ser dividida nas chamadas divisões horizontais (hor/div) e verticais (vert/div). O tempo é exibido da esquerda para a direita na escala horizontal. A tensão instantânea aparece na escala vertical, normalmente acompanhada de uma linha de referência (0V) (ROUSE, 2005).

Em um osciloscópio padrão, controles para ajuste da exibição ou realização de medidas sobre o sinal costumam estar presentes. Entre os mais comuns, estão o ajuste da escala de tempo (eixo X), ajuste da escala de tensão (eixo Y), *trigger* (sincronização da tensão e do tempo, permitindo a visualização de um sinal fixo), cursores verticais e horizontais para medidas de tensão e frequência, *run/stop* da aquisição do sinal e mudança de acoplamento (CC ou CA).

Atualmente, ao contrário dos primeiros modelos de osciloscópio que utilizavam tubos de raios catódicos para exibir as formas de onda, os osciloscópios são dispositivos digitais complexos, compostos por vários módulos de software e hardware que trabalham juntos para capturar, processar, exibir e armazenar dados que representam os sinais de interesse do usuário.

Os osciloscópios digitais também são chamados de osciloscópio de armazenamento digital (DSO, do inglês *Digital Storage Oscilloscope*) ou osciloscópios de amostragem digital. Em sua forma mais simples, um osciloscópio digital conta com pelo menos um amplificador de entrada, um conversor analógico digital (ADC), uma fonte de *clock*, uma memória, um circuito para reconstrução da forma de onda e uma tela LCD (*Liquid-Crystal Display*) ou LED (*Light-Emitting Diode*). A Figura 1 apresenta um

dos modelos mais recentes de osciloscópio digital.

FIGURA 1 – OSCILOSCÓPIO DIGITAL LABMASTER 10 ZI-A 100 GHZ

FONTE: LeCroy (2019)

## 2.2 TEOREMA DA AMOSTRAGEM: DIGITALIZAÇÃO DO SINAL

O teorema da amostragem de Nyquist–Shannon é um elo fundamental entre sinais contínuos no tempo e sinais discretizados. Em linhas gerais, o teorema estabelece uma condição suficiente para que uma sequência de amostras discretas capture toda a informação de um sinal contínuo no tempo e de banda finita. Sendo assim, o teorema é um princípio a ser seguido pelos engenheiros durante o processo de digitalização de sinais analógicos, a fim de que a conversão analógico-digital (ADC) resulte em uma reprodução fiel do sinal de origem (NYQUIST, 1928) (SHANNON, 1949) .

Formalizando os conceitos do teorema, seja $x(t)$ a representação de um sinal contínuo no tempo e seja $X(f)$ sua Transformada de Fourier (OPPENHEIM; WILLSKY, 2010):

$$X(f) = \int_{-\infty}^{\infty} \mathrm{x(t)}\, \mathrm{e}^{-j2\pi f t}\, \mathrm{d}t \tag{2.1}$$

O sinal $x(t)$ é limitado em banda, B, se:

$$X(f) = 0 \quad \forall \quad |f| > B \tag{2.2}$$

A condição suficiente para a reconstrução adequada do sinal a partir das amostras feitas a uma taxa de amostragem uniforme $f_S$ é:

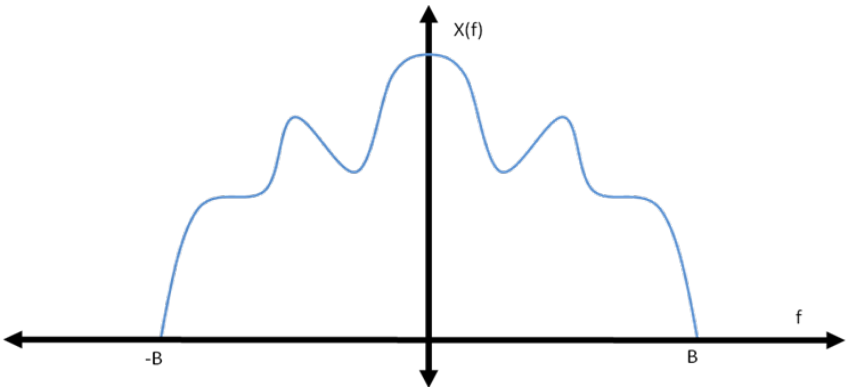
$$f_S > 2B \tag{2.3}$$

ou, em termos do período de amostragem:

$$T_S < \frac{1}{2B} \tag{2.4}$$

A Figura 2 apresenta um exemplo de espectro típico de um sinal limitado em banda B.

FIGURA 2 – ESPECTRO DE UM SINAL LIMITADO EM BANDA B.

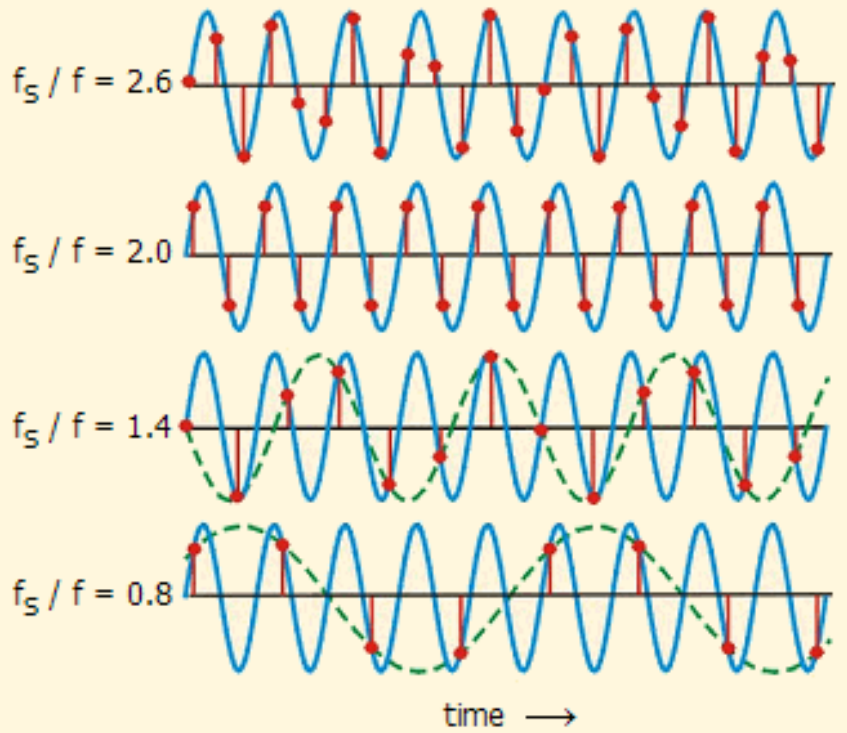



FONTE: Corbet (2019)

Se durante um processo de amostragem $f_S$ for menor do que $2B$, valor comumente chamado de Taxa de Nyquist, alguns dos componentes de frequência mais alta do sinal de entrada analógico não serão corretamente representados na saída digitalizada. Isso faz com que, durante o processo de reconstrução do sinal, componentes de frequências que não estavam presentes no sinal analógico original apareçam. Essa condição indesejável é uma forma de distorção chamada *aliasing* (SHANNON, 1949).

Para compreender melhor a influência da taxa de amostragem na qualidade da representação digital de um sinal analógico, a Figura 3 mostra um exemplo de amostragem de um sinal sinusoidal (azul claro) utilizando 4 frequências distintas (EFSTATHIOU, 2019). As linhas verdes tracejadas explicitam os casos onde ocorre *aliasing*, ou seja, quando $f_S < 2B$.

FIGURA 3 – AMOSTRAGEM DE UM SINAL EM 4 FREQUÊNCIAS $F_S$ DISTINTAS.




FONTE: Efstathiou (2019)

### 2.2.1 Erro e ruído de quantização

O erro de quantização é a caracterizado pela diferença entre o sinal analógico de entrada do ADC e o valor digital disponível mais próximo para cada amostra realizada (SHANNON, 1949). Este erro está diretamente relacionado com os níveis de decisão do sistema, os quais costumam progredir de forma linear, porém podem seguir outro padrão de evolução, como o logarítmico (COMPUTING, 2019). Para mensurar este erro, é necessário calcular o menor valor que o ADC consegue computar. Este valor é conhecido como *quantum* do ADC e define a resolução do conversor. A expressão a seguir exemplifica o cálculo do *quantum* para um ADC de N bits, com intervalo de operação entre $V_{max}$ e $V_{min}$:

$$V_q = \frac{(V_{max} - V_{min})}{2^N - 1} \quad (V) \tag{2.5}$$

Por consequência da aproximação feita pelo ADC, um ruído é introduzido no sinal amostrado, o chamado ruído de quantização (SHANNON, 1949). Quanto maior a resolução do conversor analógico-digital, menor o erro de quantização e menor o ruído

de quantização. A relação entre a resolução (em bits) e o ruído de quantização para um ADC ideal pode ser expressa como:

$$\frac{S}{N} = -20\,log(\frac{1}{2^N}) \quad (dB) \tag{2.6}$$

## 2.3 B-L475E-IOT01A DISCOVERY KIT

O kit de desenvolvimento conta com um MCU (*Microcontroller Unit*) da série *STM32L4 Ultra-low-power* da STMicroelectronics, o qual é baseado em um núcleo Arm Cortex-M4. Ele possui 1 Mbyte de memória *flash* e 128 Kbytes de memória SRAM (*Static Random Access Memory*) dentro do encapsulamento LQFP100. Além da presença dos tradicionais periféricos de comunicação SPI (*Serial Peripheral Interface*), I2C (*Inter-Integrated Circuit*) e USART (*Universal synchronous asynchronous receiver transmitter*) do microcontrolador, o kit se destaca pela presença dos seguintes módulos:

- *Bluetooth* V4.1 (SPBTLE-RF);
- RF (*Radio frequency*) programável de baixa potência Sub-GHz (868 ou 915 MHz) (SPSGRF-868 ou SPSGRF-915);
- *Wi-Fi* Inventek ISM43362-M3G-L44 (compatível 802.11 b/g/n);
- NFC (*Near Field Communication*) dinâmico baseado em M24SR com antena impressa.

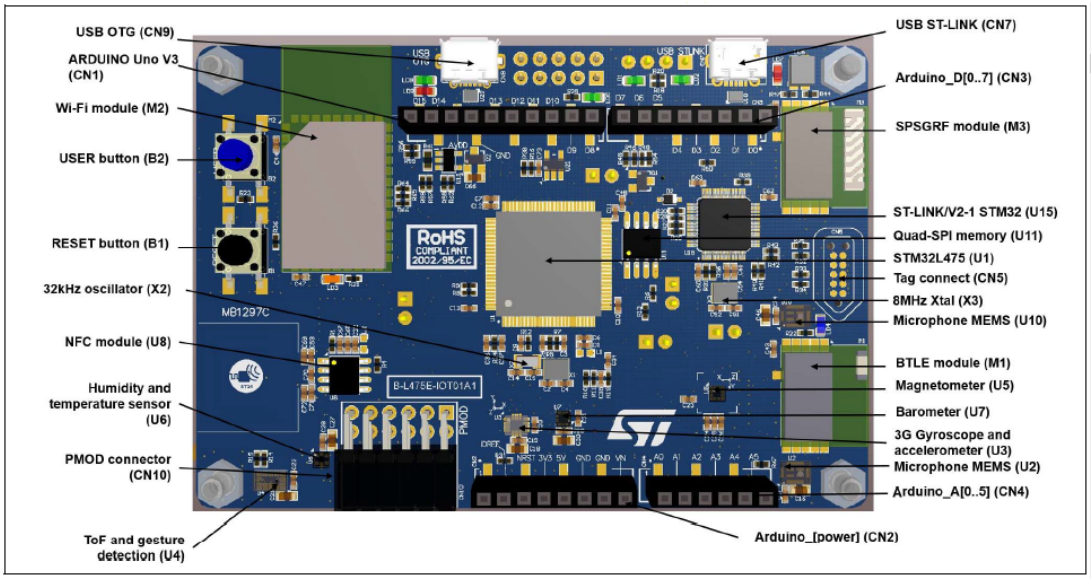
Por ser orientado ao desenvolvimento de aplicações IoT, ele também é equipado com uma alta gama de sensores integrados:

- 2 microfones digitais omnidirecionais (MP34DT01);
- Sensor capacitivo digital de umidade relativa e temperatura (HTS221);
- Magnetômetro de alto desempenho com 3 eixos (LIS3MDL);
- Acelerômetro e giroscópio 3D (LSM6DSL);
- Barômetro digital 260-1260 hPa absolutos (LPS22HB);

- Sensor *Time-of-Flight* e detecção de gestos (VL53L0X).

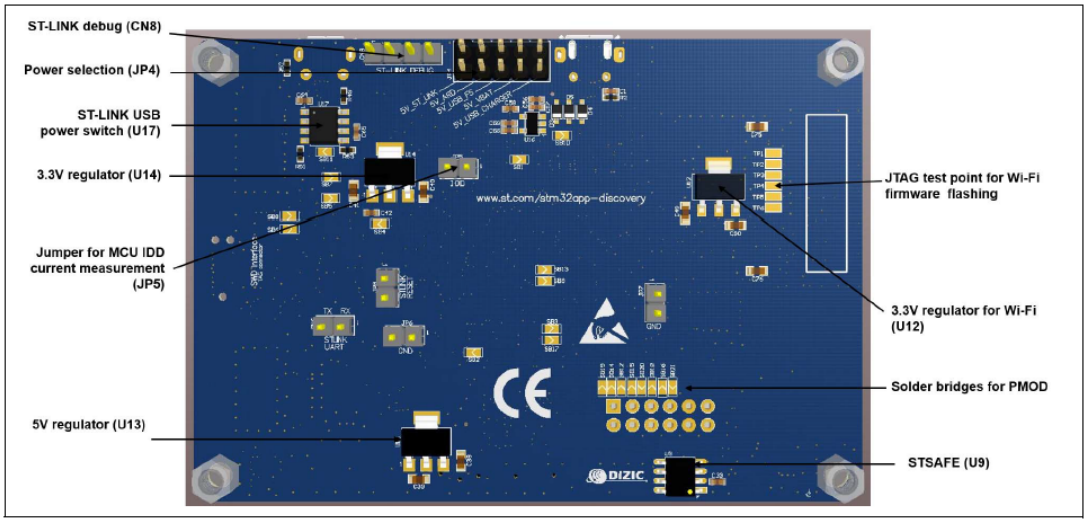
Ainda, alguns outros pontos que devem ser destacados são os conectores compatíveis com placas de extensão padrão Arduino Uno V3 e a presença de um *debugger/programmer* ST-LINK/V2-1 *onboard*, o que facilita a programação da placa e a tarefa de *debug* de código, utilizando para tal a interface micro-USB. A Figura 4 e a Figura 5 apresentam o diagrama de componentes da placa.

FIGURA 4 – VISÃO FRONTAL DO KIT B-L475E-IOT01A




FONTE: STMicroelectronics (2019b)

FIGURA 5 – VISÃO TRASEIRA DO KIT B-L475E-IOT01A




FONTE: STMicroelectronics (2019b)

### 2.3.1 Biblioteca HAL

Os *drivers* da biblioteca HAL (*Hardware Abstraction Layer*) oferecem uma maneira fácil de interagir com os periféricos dos microcontroladores da família STM32, diminuindo o tempo de implementação e desenvolvimento de projetos. Cada *driver* consiste em um conjunto de funções cobrindo os recursos mais comuns dos módulos periféricos. A biblioteca fornece uma API (*Application Programming Interface*) que garante alta portabilidade de código e, paralelamente, esconde a complexidade de programação "byte a byte" de registros do MCU. Entre os principais recursos das APIs da biblioteca HAL estão a inicialização e "*desinicialização*" de periféricos, gerenciamento de erros, interrupções e processos específicos de cada módulo (STMICROELECTRONICS, 2017).

### 2.3.2 ADC

O MCU STM32L475 presente no kit B-L475E-IOT01A é dotado de 3 conversores analógico-digital:

- O ADC1 e o ADC2 estão acoplados e podem operar no modo duplo (o ADC1 é o mestre);
- O ADC3 é controlado de forma independente.

Cada ADC consiste em um conversor analógico-digital de aproximação sucessiva de 12 bits, funcionando no intervalo de $0\,V$ a $3,3\,V$. Cada ADC possui até 20 canais multiplexados. A conversão A/D dos vários canais pode ser realizado no modo simples, contínuo, "scan" ou descontínuo. O resultado da conversão do ADC é armazenado em um registro de dados de 16 bits alinhado à esquerda ou à direita. Os ADCs são mapeados no barramento AHB (*Advanced High-Performance Bus*) para permitir o manuseio rápido dos dados. Os recursos de *watchdog* analógico permitem que a aplicação em questão detecte se a tensão de entrada aumenta fora dos limites (alto ou baixo) definidos pelo usuário. Um *oversampler* embutido no hardware permite melhorar o desempenho analógico ao mesmo tempo em que desvia a carga computacional da CPU. Também há disponível um modo de baixo consumo para operação em baixa frequência (STMICROELECTRONICS, 2018).

### 2.3.3 DMA

O controlador de acesso direto à memória (DMA) atua como "mestre" no barramento de dados e também como um periférico do sistema. O DMA é usado principalmente para realizar transferências programadas de dados entre periféricos mapeados na memória e/ou memórias, sob o controle indireto da CPU. O controlador DMA possui uma arquitetura de mestre do barramento AHB, o que proporciona uma manipulação dos dados mais rápida e eficiente. No kit B-L475E-IOT01A, existem duas instâncias de DMA disponíveis: DMA1 (7 canais) e DMA2 (7 canais). Cada canal é dedicado ao gerenciamento de solicitações de acesso à memória de um ou mais periféricos. Cada DMA inclui um "árbitro" para lidar com a prioridade entre as solicitações de DMA (STMICROELECTRONICS, 2018).

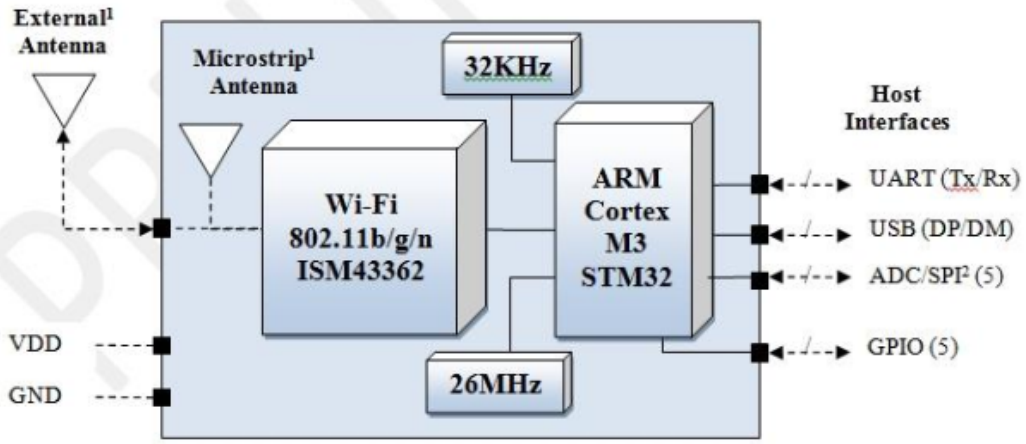
### 2.3.4 Módulo *Wi-Fi*

O módulo *Wi-Fi* Inventek ISM43362-M3G-L44 é um dispositivo de comunicação sem fio via serial. Ele possui interfaces UART (*Universal synchronous asynchronous receiver transmitter*), USB (*Universal Serial Bus*) e SPI que possibilitam a conexão com sistemas integrados. O módulo opera através de comandos IWIN (*Inventek Systems Wireless Interoperability Network*) AT (*Attention*) para estabelecer comunicação sem fio. Além disso, ele é compatível com os padrões de segurança de rede WEP-128, WPA-PSK(TKIP) e WPA2-PSK. A Figura 6 apresenta a visão externa do módulo e a Figura 7 detalha a sua arquitetura interna.

FIGURA 6 – MÓDULO *WI-FI* INVENTEK ISM43362-M3G-L44


FONTE: Inventek Systems (2017)

FIGURA 7 – ARQUITETURA DO MÓDULO *WI-FI*




FONTE: Inventek Systems (2017)

### 2.3.5 SPI

A interface SPI pode ser usada para se comunicar com dispositivos externos usando o protocolo SPI. O esquema de comunicação do SPI é selecionável por software, sendo o modo Motorola selecionado por padrão. O protocolo de interface periférica serial suporta a comunicação síncrona com dispositivos externos nos modos *half-duplex*, *full-duplex* e *simplex*. A interface pode ser configurada como *"master"* e, neste caso, fornece o relógio de comunicação ao dispositivo *"slave"* externo (STMICROELECTRONICS, 2018).

### 2.3.6 USART

O USART oferece um canal *full-duplex* de troca de dados flexível com equipamentos externos. Ele possui uma ampla variedade de taxas de transmissão, graças a disponibilidade de um gerador de taxas de transmissão programável. Além disso, efetuando o uso do DMA para obter uma configuração do tipo *multi-buffer*, é possível realizar comunicações de alta velocidade (STMICROELECTRONICS, 2018).

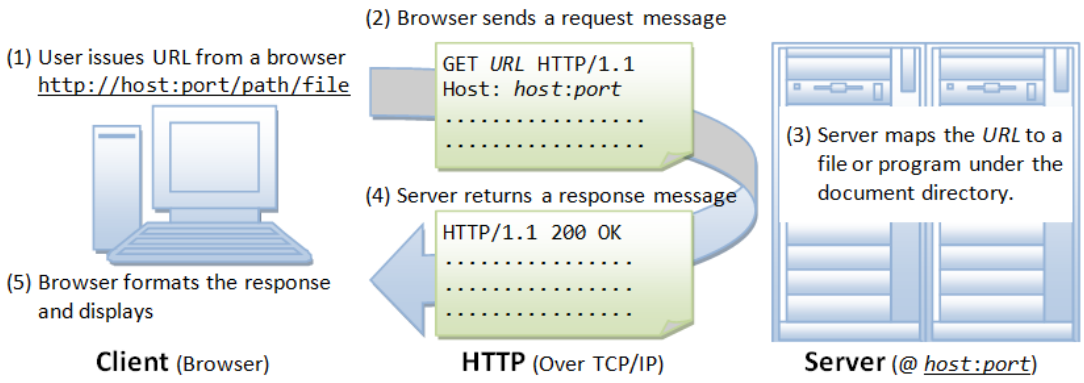
## 2.4 PROTOCOLO HTTP

O *Hypertext Transfer Protocol* (HTTP) é um protocolo de aplicação para sistemas distribuídos, colaborativos e sistemas de informação hipermídia. O HTTP representa a fundação da comunicação de dados para a *World Wide Web*, onde documentos

do tipo *hypertext* incluem *hyperlinks* para outros recursos que o usuário pode acessar facilmente, por exemplo com o clique de um mouse ou tocando a tela em um navegador Web (SOCIETY, 1994). O desenvolvimento do HTTP foi iniciado por Tim Berners-Lee no CERN em 1989, sendo que a primeira versão do protocolo tinha apenas um método, chamado GET, que solicitava uma página de um servidor e tinha como resposta a página HTML (*Hypertext Markup Language*) (SOUZA, 2019).

O protocolo HTTP funciona segundo o padrão solicitação/resposta. Após estabelecida uma conexão entre um cliente e o servidor, o cliente envia uma solicitação, a qual é processada pelo servidor que envia uma resposta indicando o sucesso ou falha da requisição (MDN CONTRIBUTORS, 2019). A Figura 8 detalha esse fluxo em um exemplo de comunicação entre cliente (navegador) e um servidor Web.

FIGURA 8 – EXEMPLO DE FUNCIONAMENTO DO PROTOCOLO HTTP.



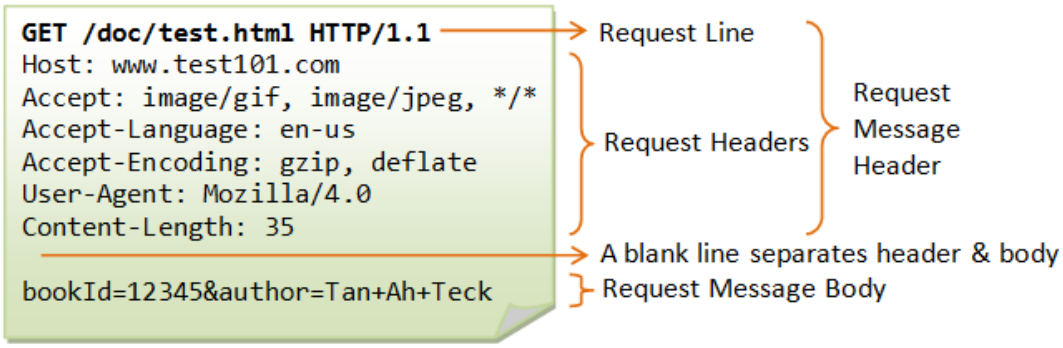

FONTE: Hock-Chuan (2009)

As solicitações HTTP consistem dos seguintes elementos, conforme mostrado na Figura 10:

- Um método HTTP que define qual operação o cliente deseja realizar. Tipicamente, um cliente que pegar um recurso (usando o método GET) ou enviar dados (usando o método POST), embora outras operações também sejam possíveis;
- O caminho do recurso a ser buscado, composto pela URL do recurso sem os elementos que são de contexto ("http://", por exemplo), o domínio e a porta TCP (*Transmission Control Protocol*), que pode ser ocultada no caso da porta padrão 80;

- A versão do protocolo HTTP;
- Cabeçalhos opcionais que contém informações adicionais para os servidores;
- Um corpo de dados opcional para métodos que contém o recurso requisitado ou dados a ser enviados.

FIGURA 9 – EXEMPLO DE SOLICITAÇÃO HTTP.

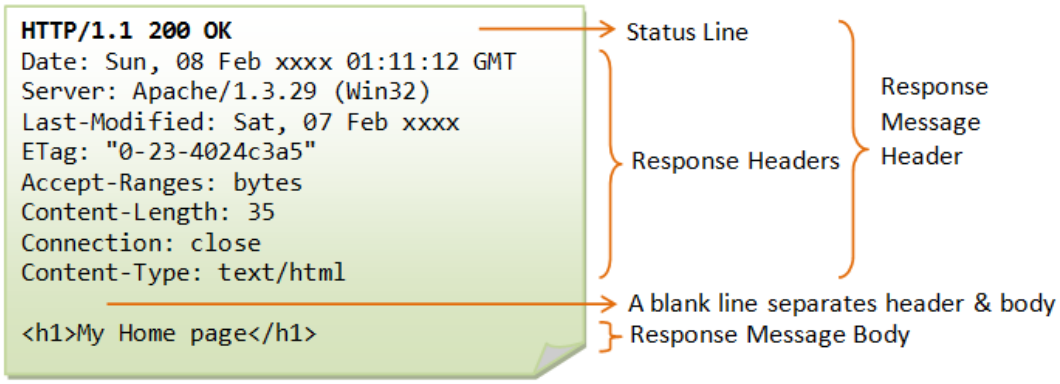



FONTE: Hock-Chuan (2009)

As respostas HTTP consistem dos elementos a seguir, conforme a Figura 10:

- A versão do protocolo HTTP que elas seguem;
- Um código de status, indicando se a requisição foi bem sucedida ou teve falha (200, por exemplo);
- Uma mensagem de status, uma pequena descrição informal sobre o código de status (OK, por exemplo);
- Cabeçalhos HTTP, como aqueles das requisições;
- Opcionalmente, um corpo com dados do recurso requisitado.

FIGURA 10 – EXEMPLO DE RESPOSTA HTTP.




FONTE: Hock-Chuan (2009)

## 2.5 PROTOCOLO TCP

O TCP é um dos principais protocolos de Internet, permitindo o estabelecimento de uma comunicação confiável, ordenada e efetuando a verificação de erros no fluxo de *bytes* entre aplicações na rede IP (*Internet Protocol*). As principais aplicações de Internet como *World Wide Web*, *email*, administração remota e transferência de arquivos dependem do protocolo TCP para o seu funcionamento (TANENBAUM; WETHERALL, 2012).

O gerenciamento das conexões do TCP é feito através de *sockets*, os quais possuem um número de *socket* formado pelo endereço IP do dispositivo e uma porta de comunicação local. Para habilitar o serviço TCP, é necessário que uma conexão seja estabelecida entre um *socket* em uma máquina e um *socket* em outra máquina. Por padrão, no caso do protocolo HTTP, a porta utilizada pelo TCP é a 80 (TANENBAUM; WETHERALL, 2012).

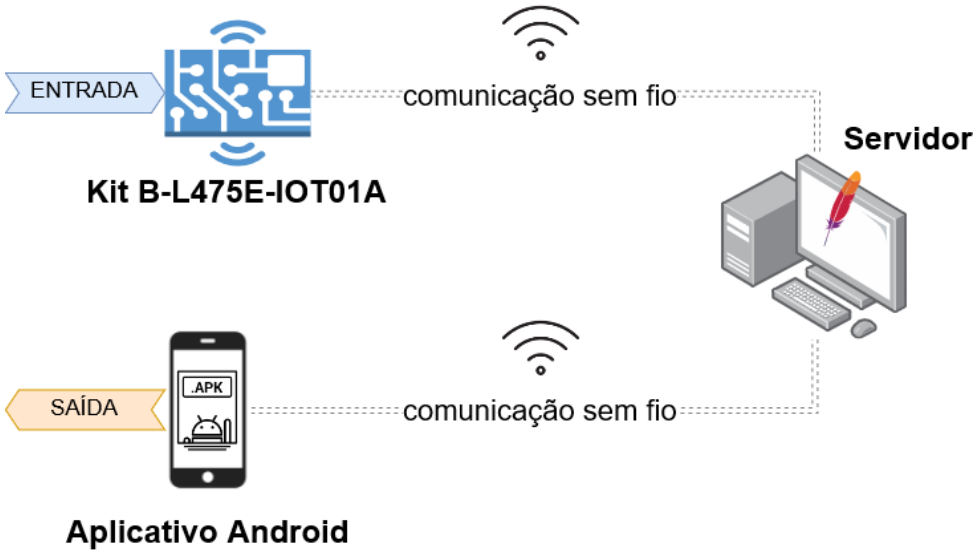
# 3 MATERIAIS E MÉTODOS

A metodologia de desenvolvimento do projeto deu-se através do processo de programação e *debug* repetitivo, juntamente com a validação através da verificação das formas de onda no aplicativo/*website*, contando também com o auxílio de um gerador de funções e um osciloscópio para verificação do funcionamento do osciloscópio com sinais reais no laboratório.

## 3.1 ESTRUTURA DO PROJETO

A estrutura deste trabalho é composta por 3 elementos principais: servidor, kit de desenvolvimento e aplicativo do usuário, conforme exemplificado na Figura 11.

FIGURA 11 – ESTRUTURA ELEMENTAR DO PROJETO




FONTE: o Autor (2019)

O funcionamento desejado do sistema segue as etapas enumeradas a seguir:

1. O sinal eletrônico é aplicado em uma entrada analógica da placa B-L475E-IOT01A;
2. "N" amostras do sinal são tomadas e armazenadas;
3. As amostras são enviadas para um servidor via rede sem fio;

4. O servidor recebe os dados e atualiza o gráfico de sinal;
5. O usuário visualiza a forma de onda através do aplicativo.

## 3.2 SOFTWARES E FERRAMENTAS

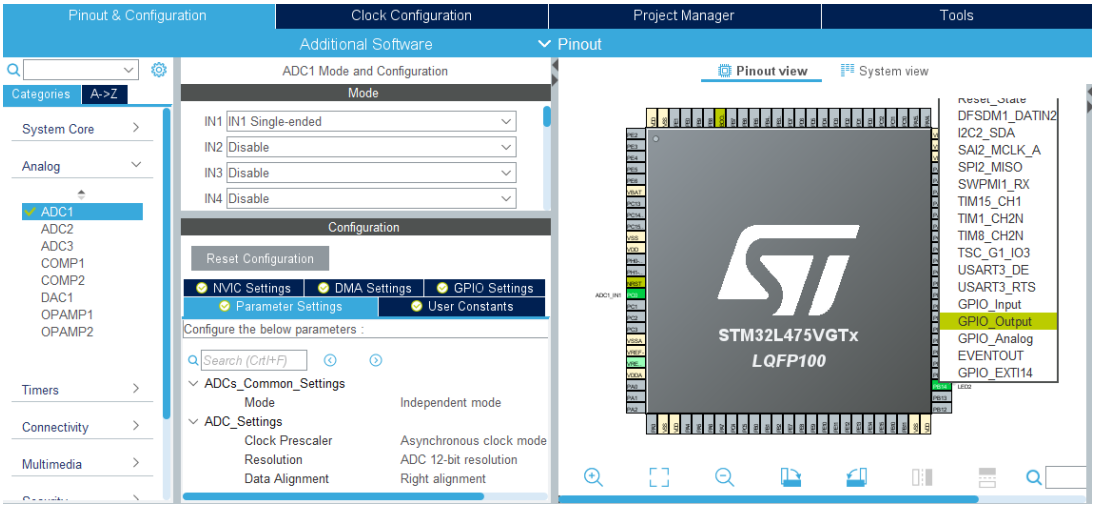
Nesta seção são apresentados os softwares, ferramentas e demais recursos utilizados para o desenvolvimento do projeto.

### 3.2.1 STM32CubeIDE

STM32CubeIDE é uma ferramenta de desenvolvimento multifuncional *all-in-one*, que faz parte do ecossistema de software STM32Cube desenvolvido pela STMicroelectronics. Trata-se de uma plataforma avançada de desenvolvimento C/C++ com recursos de configuração de periféricos, geração de código, compilação e depuração para microcontroladores e microprocessadores STM32. Ele é baseado na estrutura ECLIPSE e na cadeia de ferramentas GCC (*GNU Compiler Collection*) para a compilação e no GDB (*GNU Debugger*) para depuração.

A IDE (*Integrated Development Environment*) reúne todas as funcionalidades presentes no antigo STM32CubeMX, diminuindo o tempo necessário para construção do projeto e aumentando a produtividade. Após a seleção de um MCU STM32 ou da seleção de um kit, o usuário pode selecionar quais periféricos deseja habilitar, bem como o modo de funcionamentos das entradas e saídas (GPIOs, do inglês *General Purpose Inputs/Outputs*) e a configuração de *clocks* do sistema. Em seguida, o projeto é criado e um código de inicialização é gerado, economizando um precioso tempo que seria dedicado a realizar a configuração dos módulos do MCU. Além disso, a qualquer momento durante o desenvolvimento, o usuário pode retornar à aba de e configuração dos periféricos ou *middleware* e regenerar o código de inicialização sem afetar o código do usuário. A Figura 12 apresenta esta interface.

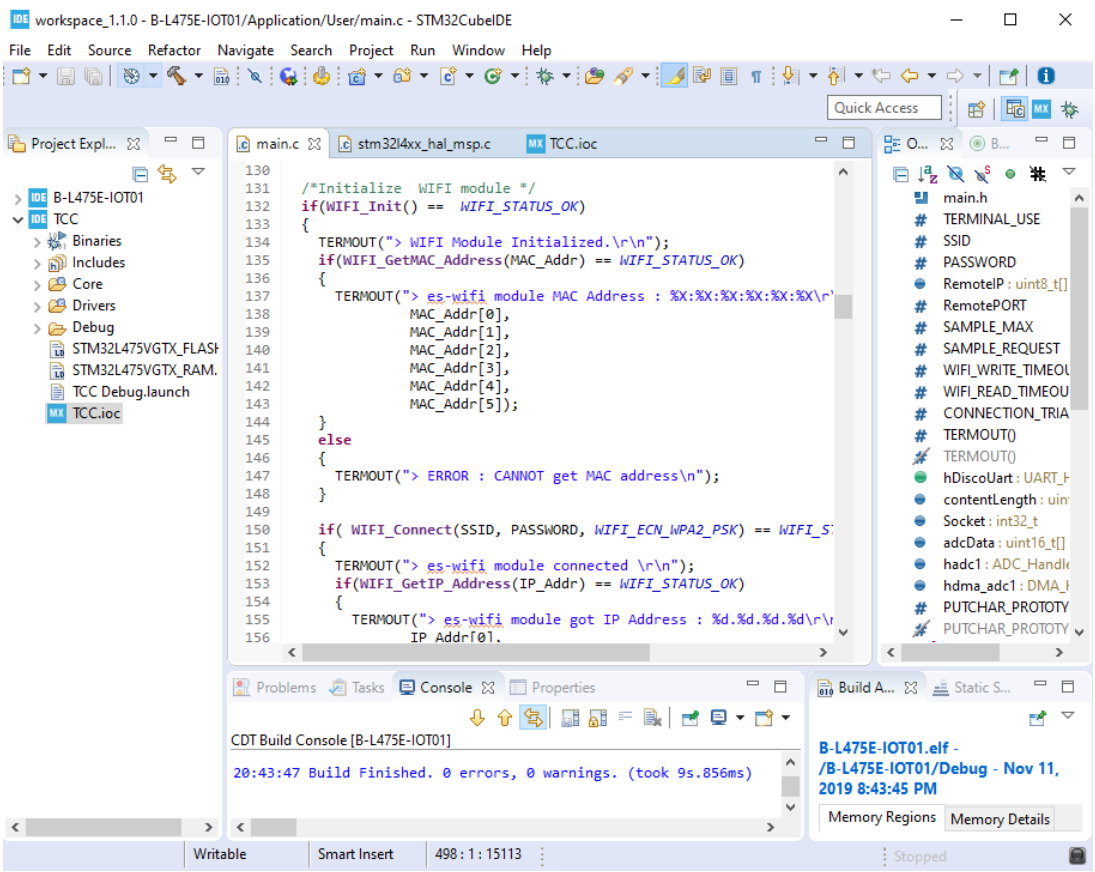
FIGURA 12 – ABA DE CONFIGURAÇÃO DE PINOS E PERIFÉRICOS NO STM32CUBEIDE.




FONTE: o Autor(2019)

O STM32CubeIDE inclui também analisadores de compilação e empilhamento, os quais fornecem ao usuário informações úteis sobre o status do projeto e os requisitos de memória. A Figura 13 mostra o ambiente de compilação e desenvolvimento.

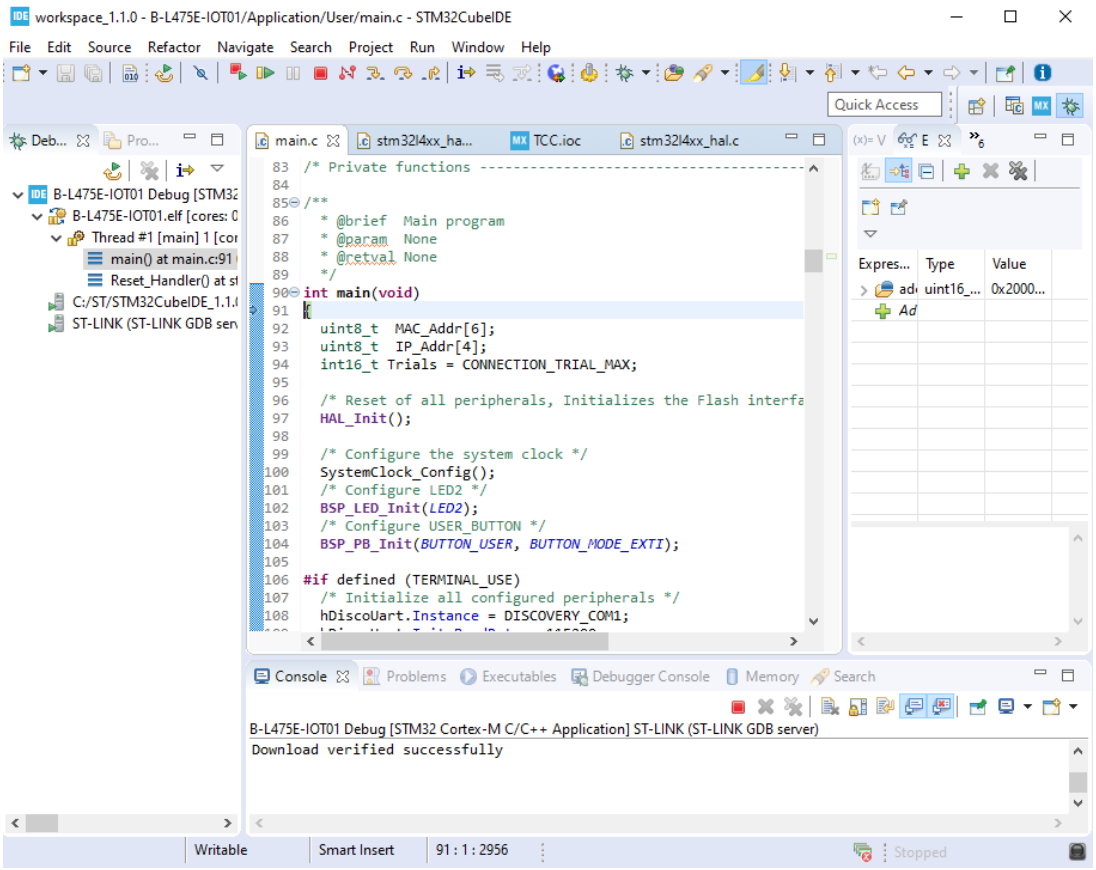
FIGURA 13 – VISTA DO COMPILADOR NO STM32CUBEIDE.




FONTE: STMicroelectronics (2019a)

O STM32CubeIDE também inclui recursos de depuração avançados, incluindo visualizações dos registros principais da CPU, memórias e registros periféricos, bem como monitoramento de variáveis ao vivo e analisador de falhas. A Figura 14 mostra a visão do ambiente de *debug*.

FIGURA 14 – VISTA DO *DEBUGGER* NO STM32CUBEIDE.




FONTE: STMicroelectronics (2019a)

### 3.2.2 MobaXterm

O MobaXterm é uma suíte de ferramentas para computação remota. Ele é especialmente orientado à programadores, administradores de TI e usuários que precisam lidar com máquinas remotas de maneira mais simples e agradável (MOBATEK, 2019).

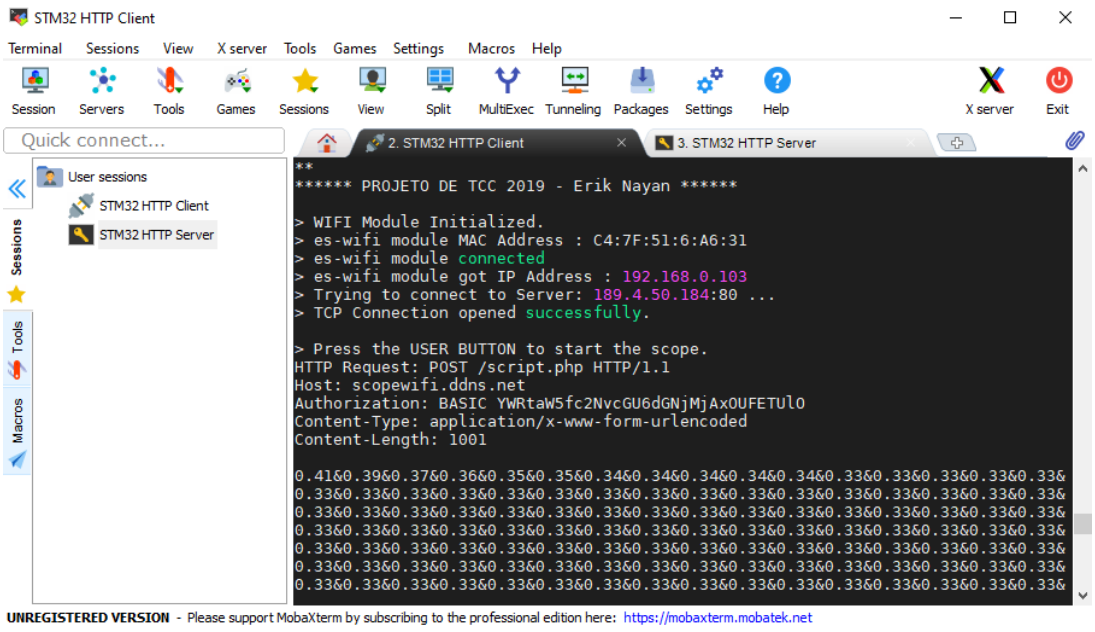
O MobaXterm reúne todos os mais importantes protocolos para operações entre redes, como SSH (*Secure Shell*) e FTP (*File Transfer Protocol*), possibilitando também o envio de comandos Unix (*bash, ls, cat, sed, grep, awk, rsync, ...*) a partir do sistema operacional Windows. Outras vantagens do MobaXterm são o terminal Serial,

o redirecionamento de monitor (do inglês *X11-forwarding*) e o Web browser integrado (MOBATEK, 2019).

Deste modo, considerando todas as funcionalidades e recursos disponibilizados, pode-se afirmar que o MobaXterm é uma ferramenta preciosa e extremamente útil para o desenvolvimento de aplicações envolvendo redes, servidores e páginas Web.

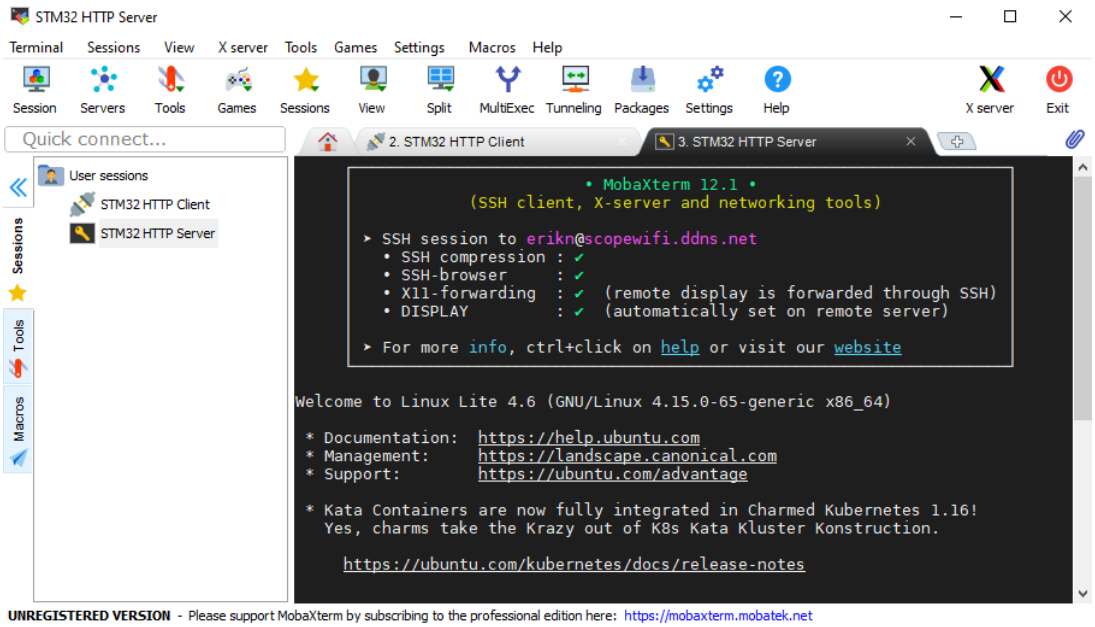
Ao longo do projeto, os recursos de terminal Serial e SSH foram amplamente utilizados, especialmente para realizar o *debug* de código e para conectar-se remotamente ao servidor HTTP. A Figura 15 e a Figura 16 apresentam as interfaces do MobaXterm citadas.

FIGURA 15 – SESSÃO DE TERMINAL SERIAL NO MOBAXTERM.




FONTE: o Autor (2019)

FIGURA 16 – SESSÃO DE TERMINAL SSH NO MOBAXTERM.




FONTE: o Autor (2019)

### 3.2.3 No-IP

O No-IP é um serviço gratuito de DNS (*Domain Name System*) dinâmico, que aponta um nome de *host* para um endereço IP dinâmico (NO-IP, 2019). No caso do projeto, isso permite que o acesso ao servidor através de seu nome (scopewifi.ddns.net) se mantenha estável, mesmo quando ocorre uma atualização do endereço IP da rede doméstica na qual ele está instalado. Para tal, o cliente No-IP verifica a rede constantemente e, ao perceber que o IP dinâmico do *host* em questão foi alterado, automaticamente realiza a atualização do endereço em sua tabela de DNS, garantindo que o acesso ao servidor se mantenha normal.

### 3.2.4 Apache

O Apache HTTP Server foi lançado em 1995 e é o servidor Web mais popular da Internet desde abril de 1996. O projeto é um esforço colaborativo de desenvolvimento de software, o qual busca implementar um servidor HTTP de código-fonte robusto, de nível comercial e de distribuição gratuita. O projeto é gerenciado em conjunto por um grupo de voluntários espalhados por todo o mundo, usando a Internet e a Web para se comunicar, planejar e desenvolver o servidor e a documentação do mesmo. Além

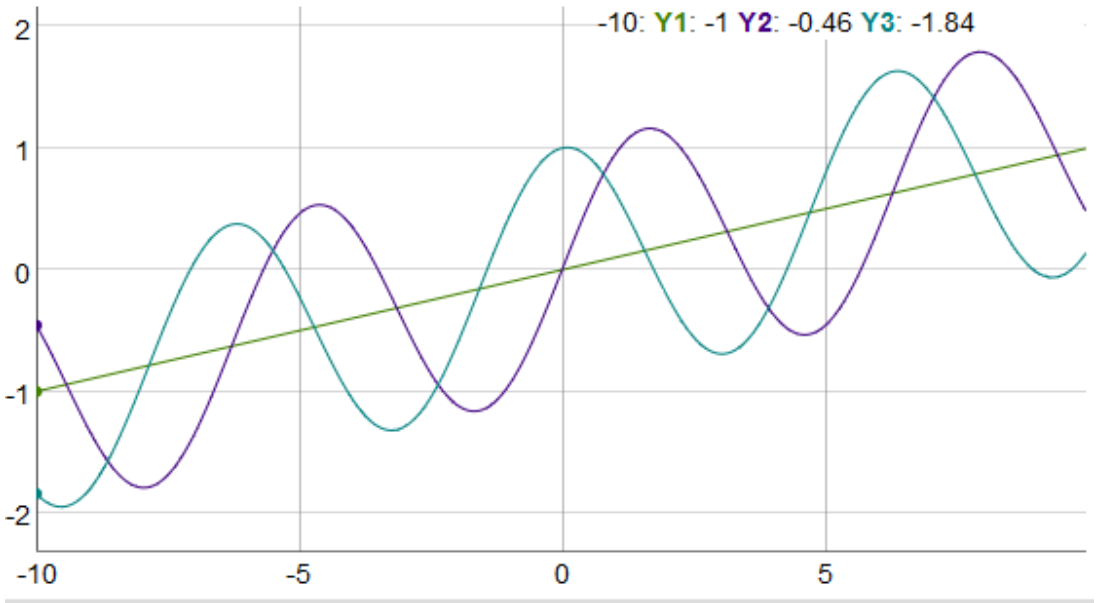
disso, um número expressivo de usuários contribuíram e ainda contribuem com ideias, códigos e documentação para o projeto (FOUNDATION, 2019).

### 3.2.5 Dygraphs

Dygraphs é uma biblioteca gráfica gratuita em JavaScript, que permite traçar gráficos e interpretar extensos conjuntos de dados de maneira rápida e flexível. A biblioteca é capaz de plotar milhares de pontos sem travamentos e oferece alta customização através de diversas opções para personalizar o comportamento do gráfico e também a maneira de como os dados são exibidos. Outra característica importante é a interatividade, pois o Dygraphs conta com recursos de *zoom*, *pan* e *mouseover*, os quais são habilitados por padrão na interface (DYGRAPHS, 2019).

Diversos exemplos de aplicação são fornecidos pelos desenvolvedores em sua página web, assim como tutoriais e projetos realizados pela comunidade, o que facilita o ciclo de desenvolvimento utilizando o Dygraphs. A Figura 17 apresenta uma demonstração de plotagem utilizando a biblioteca.

FIGURA 17 – EXEMPLO DE PLOTAGEM DE FUNÇÕES MATEMÁTICAS COM DYGRAPHS




FONTE: Dygraphs (2019)

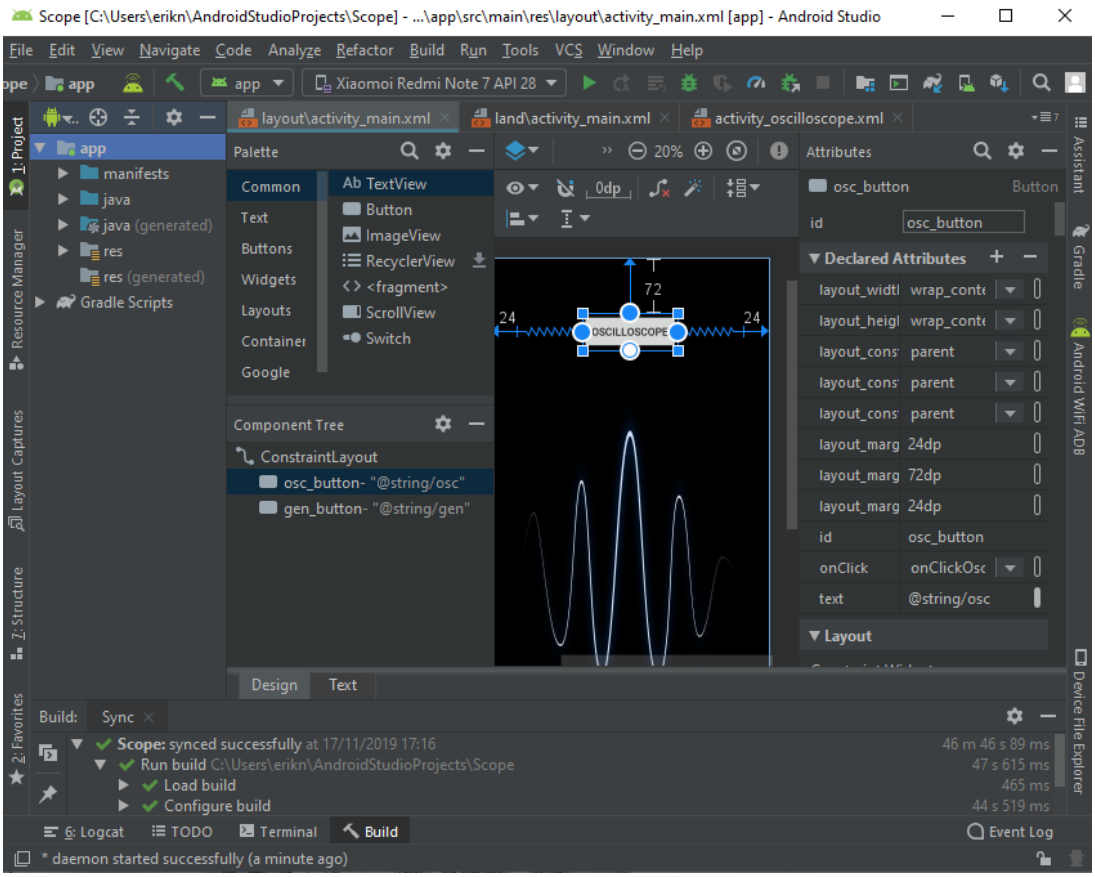
### 3.2.6 Android Studio

O Android Studio é um ambiente de desenvolvimento integrado (IDE, na sigla em inglês) oficial para o desenvolvimento de *apps* Android e é baseado no IntelliJ IDEA. Além do editor de código, o Android Studio possui vários outros recursos para expandir as possibilidades durante a criação de aplicativos, sendo os principais:

- Sistema de compilação flexível baseado em Gradle;
- Emulador de sistema Android totalmente personalizável;
- Aplicação de alterações de código e recursos ao aplicativo em execução sem reiniciá-lo;
- Integração com GitHub facilitando o controle de versão;
- *Frameworks* e ferramentas de teste (DEVELOPERS, 2019a).

A interface do Android Studio é bastante amigável, pois permite alternar rapidamente entre o editor de códigos e a visualização do *layout* do aplicativo. Ainda sobre a parte gráfica, é possível construí-la tanto através de código XML (*Extensible Markup Language*) quanto através de uma GUI interativa onde os recursos podem ser adicionados de maneira rápida e fácil, arrastando o recurso desejado da palheta de opções para dentro da área do *app*. A Figura 18 apresenta o ambiente de trabalho do Android Studio.

FIGURA 18 – AMBIENTE DE DESENVOLVIMENTO DO ANDROID STUDIO.




FONTE: o Autor (2019)

## 3.3 LINGUAGENS DE PROGRAMAÇÃO

Nesta seção são citadas as linguagens de programação exigidas no desenvolvimento de códigos deste projeto.

### 3.3.1 Linguagem C

C é uma linguagem de programação compilada de propósito geral, estruturada, imperativa, procedural e padronizada. Ela foi desenvolvida inicialmente por Dennis Ritche em 1972 na empresa AT&T Bell Labs para desenvolvimento do sistema operacional Unix. Entre os principais recursos da linguagem C estão o acesso de baixo nível a memória, o conjunto simples de palavras-chave e o estilo simples e tangível. Essas e outras características fazem da linguagem C uma das mais populares até

hoje, principalmente para a programação de sistemas embarcados de tempo real e microcontroladores em geral (GEEKSFORGEEKS, 2019).

### 3.3.2 Linguagem PHP

PHP é uma linguagem de *script* de uso geral de código aberto amplamente utilizada. Ela é especialmente adequada para o desenvolvimento Web e pode ser incorporada ao HTML. A sua sintaxe baseia-se em C, Java e Perl. O principal objetivo da linguagem é permitir que os desenvolvedores Web escrevam páginas geradas de maneira dinâmica e desenvolvam *scripts* inteligentes para lidar com o fluxo de dados (GROUP, 2019).

### 3.3.3 Linguagem HTML

HTML é uma das linguagens mais utilizadas no desenvolvimento de *websites*. Tim Berners-Lee, o mesmo criador do HTTP, foi quem criou o HTML. O HTML ficou bastante conhecido quando começou a ser utilizado para formar a rede pública daquela época (anos 90), o que se tornaria mais tarde a internet que conhecemos hoje (EIS, 2011).

O HTML pode ser considerado a linguagem base da internet. O fato de ser uma linguagem de fácil entendimento tanto para seres humanos quanto para as máquinas, faz com que essa linguagem seja extremamente popular e prática para o desenvolvimento de páginas Web.

### 3.3.4 Linguagem JAVA

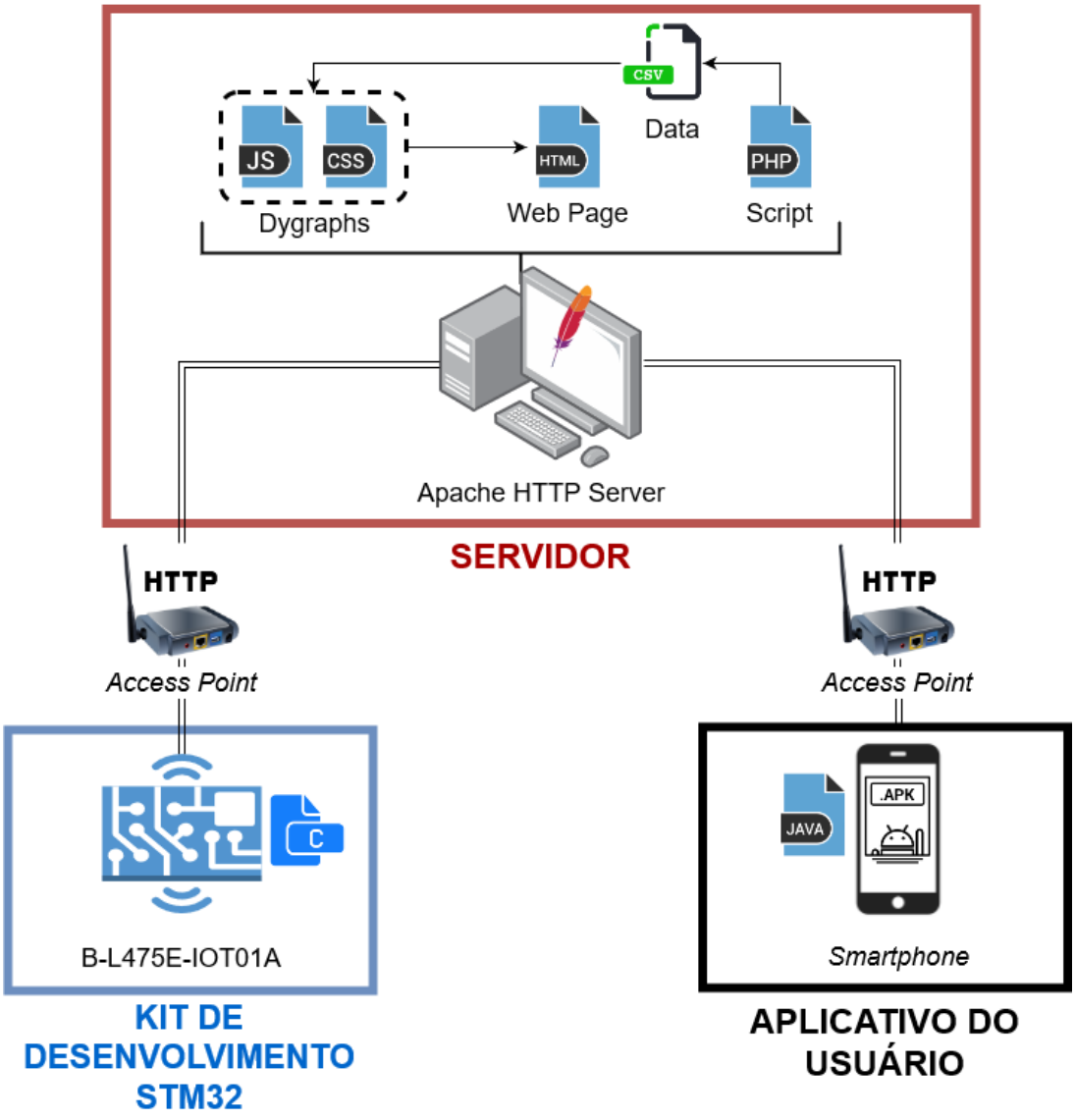
O Java é uma das mais populares linguagens de programação, sendo desenvolvido pela Sun Microsystems (atualmente pertencente a Oracle). O Java incorpora muitos dos recursos principais de linguagens mais antigas como C e C++, ao mesmo tempo em que aperfeiçoa algumas de suas desvantagens. Entre as principais características do Java, pode-se citar a facilidade de compreensão e aprendizado, juntamente com o fato de ser orientada a objetos e ser concebida visando ser uma linguagem independente da plataforma em que esteja rodando, através do uso de máquinas virtuais (CONDER; DARCEY, 2010).

Uma das principais vantagens do Java é a possibilidade de desenvolver aplicações nativas para Android, dado a proximidade e compatibilidade entre as APIs. Seguindo esta metodologia, aplicativos são criados ao mixar os recursos gráficos e bibliotecas específicas do Android com as classes padrão do Java (CORDEIRO, 2017).

# 4 DESENVOLVIMENTO

Para compreender em detalhes a estrutura completa do projeto e seus componentes, um esquemático indicando os elementos do sistema e as linguagens de programação envolvidas é apresentado pela Figura 19.

FIGURA 19 – ESTRUTURA GLOBAL DO SISTEMA PROPOSTO

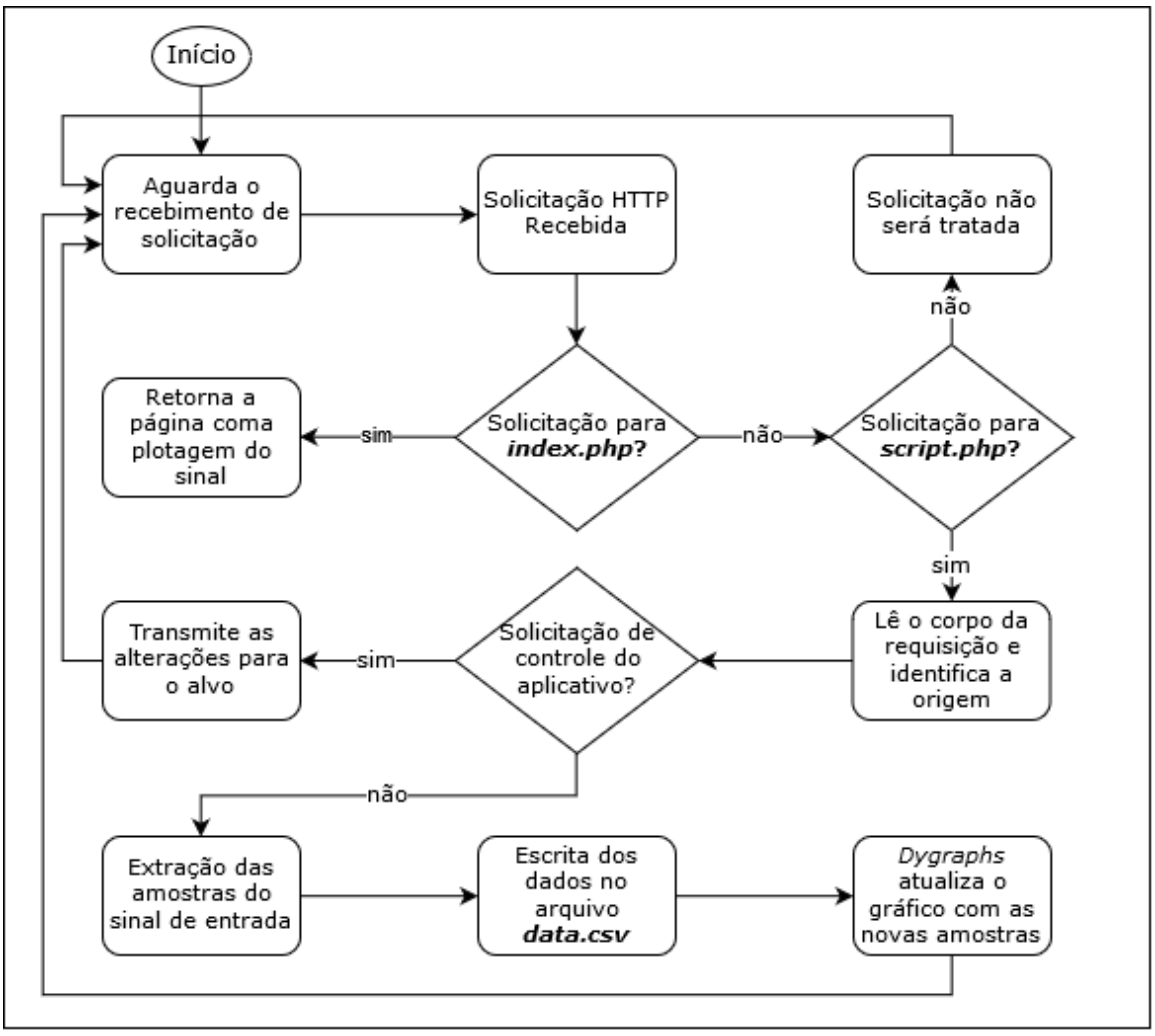



FONTE: o Autor (2019)

## 4.1 PARTE DO SERVIDOR

A Figura 20 apresenta o fluxograma de funcionamento do servidor, mostrando o papel de cada item da parte do servidor na estrutura global do sistema.

FIGURA 20 – FLUXOGRAMA SIMPLIFICADO DO FUNCIONAMENTO DO SERVIDOR




FONTE: o Autor (2019)

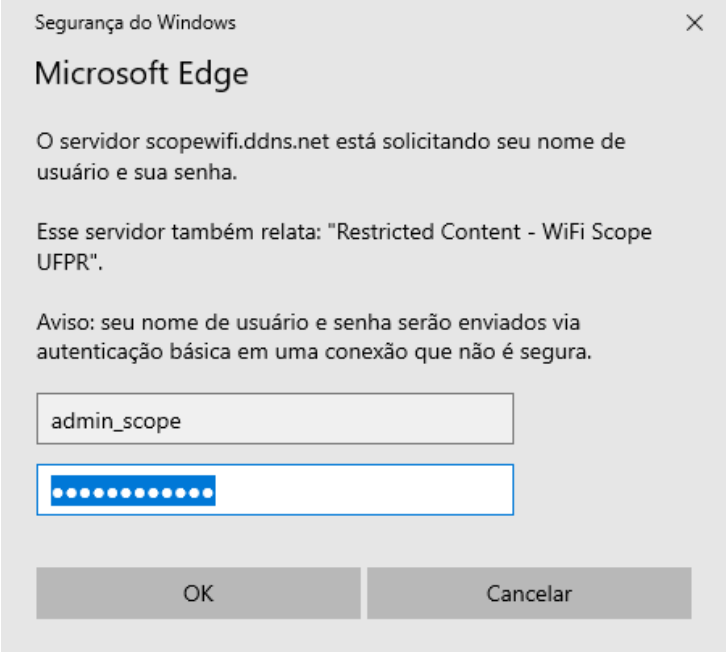
### 4.1.1 Configurações

A instalação do servidor foi feita de maneira extremamente simples, através de comandos no terminal Linux (EBRAHIM, 2017). No entanto, algumas configuração extras precisaram ser realizadas, a fim de viabilizar o funcionamento do projeto e assegurar o acesso aos dados de maneira mais segura.

#### 4.1.1.1 Autenticação

Como uma primeira camada de segurança, foi implementada a autenticação HTTP no servidor, de modo a restringir o acesso ao diretório principal do projeto. Dois usuários foram criados, *admin_scope* e *user_scope*, os quais foram adicionados a um arquivo de senhas. Feito isso, a demanda de autenticação foi inserida no diretório do *host* virtual do projeto (ELLINGWOOD, 2015). A Figura 21 mostra o pedido de autenticação dada uma tentativa de acesso ao *Website*.

FIGURA 21 – PEDIDO DE AUTENTICAÇÃO AO ACESSAR O SERVIDOR




FONTE: o Autor (2019)

#### 4.1.1.2 *Timeout*

A diretiva de *Timeout* define a quantidade de tempo que o servidor Apache aguardará por três coisas:

- O tempo total necessário para receber uma solicitação GET;
- A quantidade de tempo entre o recebimento de pacotes TCP em uma solicitação POST ou PUT;
- A quantidade de tempo entre os ACKs (*Acknowledges* nas transmissões de pacotes TCP em respostas (DOWNEY, 2019).

O valor padrão do *Timeout* é 300 segundos, o que costuma ser muito mais do que o necessário na maioria das situações. Porém, para o projeto do osciloscópio, esse tempo foi aumentado para 3600 segundos com a intenção de garantir que não hajam desconexões indesejadas causadas por eventuais falhas na conexão.

#### 4.1.1.3 *Keep-Alive*

O recurso de conexão persistente *Keep-Alive* fornece sessões HTTP de longa duração que permitem o envio de várias solicitações pela mesma conexão TCP. Isso permite reduzir de maneira significativa o tempo de latência nas trocas de dados pela rede. A partir da versão 1.1 do HTTP, o recurso *Keep-Alive* é habilitado por padrão, a menos que especificado de outra forma (DOWNEY, 2019).

#### 4.1.1.4 *KeepAliveTimeout*

A diretiva *KeepAliveTimeout* representa o número de segundos que o Apache aguardará uma solicitação subsequente antes de fechar a conexão. Depois que uma solicitação é recebida, o tempo de espera especificado pela diretiva *Timeout* se aplica. Quanto maior o tempo do *KeepAliveTimeout*, mais processos do servidor serão mantidos ocupados aguardando conexões com clientes inativos (DOWNEY, 2019).

Assim como o que foi feito para a diretiva *Timeout*, o valor de *KeepAliveTimeout* foi alterado para 3600 segundos, de maneira a também permitir que a aquisição do osciloscópio seja pausada e retomada em um tempo limite de 1 hora.

#### 4.1.1.5 *MaxKeepAliveRequests*

A diretiva *MaxKeepAliveRequests* limita o número de solicitações permitidas por conexão quando *KeepAlive* está ativado. O valor padrão desta diretiva é 100 (DOWNEY, 2019). Entretanto, este valor foi redefinido para "0", de modo que não há limite para o número de solicitações. Isto possibilita o envio constante de requisições HTTP pela mesma conexão entre o kit B-L475E-IOT01A e o servidor.

#### 4.1.1.6 Cliente No-IP

Além das configurações adicionais realizadas, um cliente do serviço No-IP foi integrado, para que fosse possível manter atualizado o IP do servidor na tabela de

DDNS dinâmica, mantendo o acesso através do nome de *host* disponível.

### 4.1.2 Página principal

A página principal do projeto se resume ao gráfico de exibição das formas de onda, com a tensão (V) sendo representada no eixo Y e a escala de tempo (s) no eixo X. Para exibir o gráfico, a biblioteca Dygraphs é incluída no arquivo da página. A biblioteca permite personalizar a exibição e busca os dados para plotagem em um arquivo de extensão CSV.

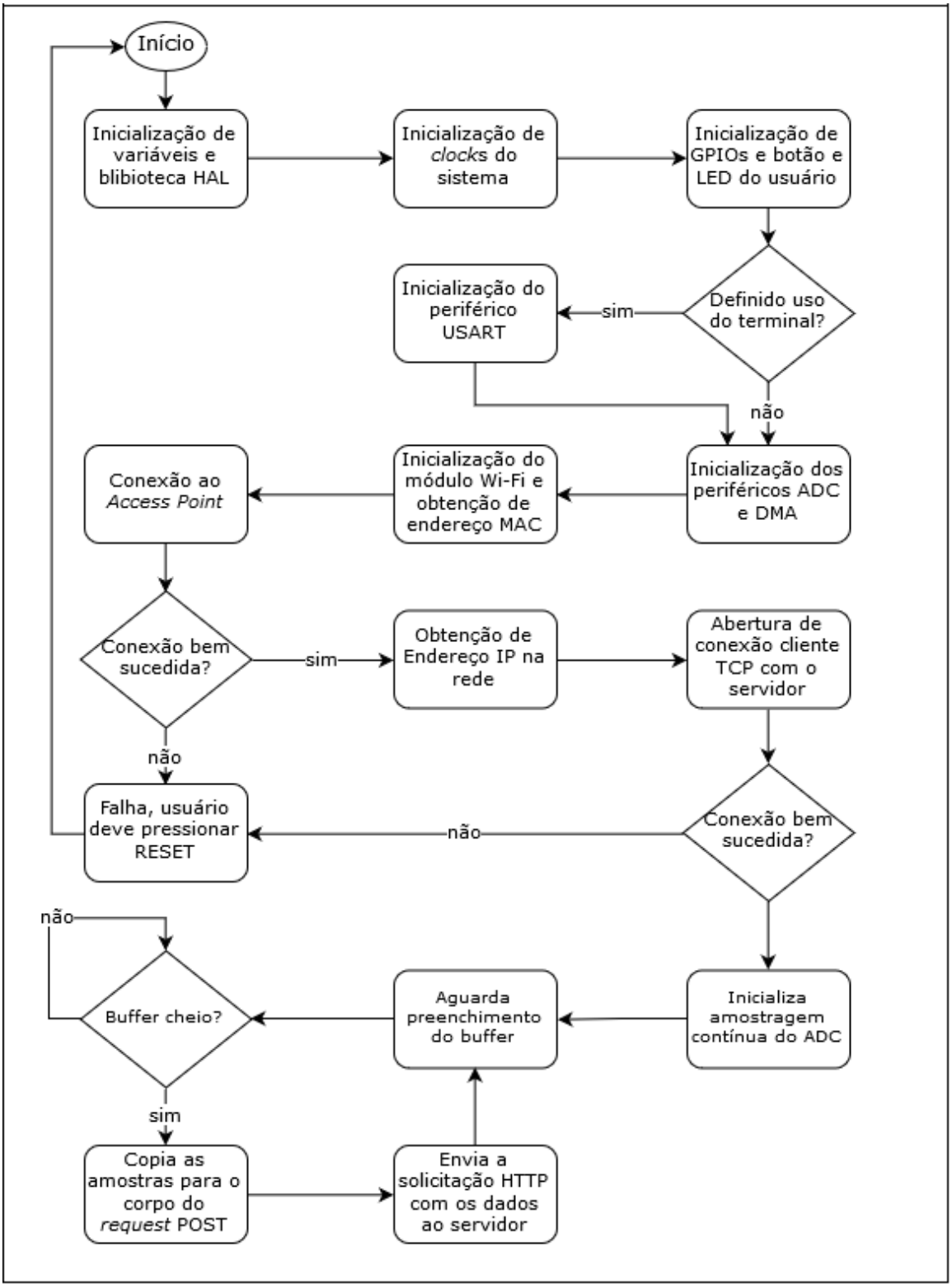
### 4.1.3 Script PHP

Este *script* em linguagem PHP tem o objetivo de tratar as solicitações HTTP feitas ao servidor, extraindo os dados do corpo da requisição e realizando o tratamento adequado. No caso de requisições oriundas do kit B-L475E-IOT01A, o *script* recupera as amostras e atualiza os valores presentes no arquivo CSV lido pela biblioteca gráfica. No caso de requisições oriundas do aplicativo, o *script* repassa as modificações necessárias para ajustar a exibição da forma de onda na tela.

## 4.2 PARTE DO KIT DE DESENVOLVIMENTO

A Figura 22 apresenta o algoritmo principal programado no microcontrolador STM32L475, o qual realiza as tarefas de aquisição de amostras do sinal de interesse e envio delas ao servidor via rede sem fio.

FIGURA 22 – FLUXOGRAMA SIMPLIFICADO DO FUNCIONAMENTO DO MICROCONTROLADOR

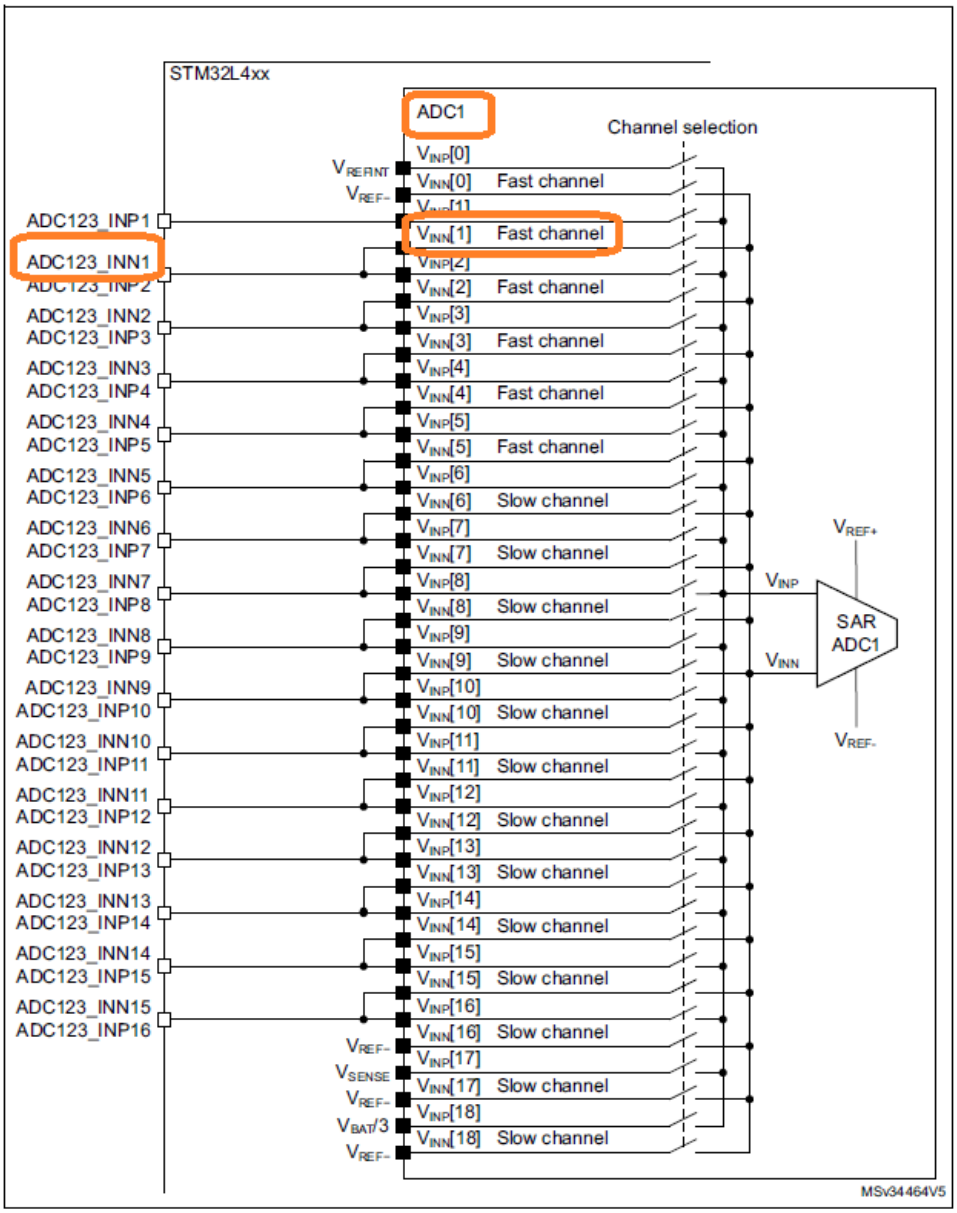



FONTE: o Autor (2019)

### 4.2.1 Configuração do ADC e DMA

Com o objetivo de obter a maior taxa de amostragem possível com o ADC disponível, o pino de entrada/saída escolhido como porta de entrada do osciloscópio foi o A5(PC0), que está internamente multiplexado a entrada IN1 do ADC1, que corresponde a um canal rápido (*fast channel*), conforme mostra a Figura 23. Além disso, a fonte de *clock* selecionada para o ADC foi o *clock* principal do sistema de $80\,MHz$.

FIGURA 23 – DIAGRAMA DOS CANAIS DO ADC1 DO STM32L475.

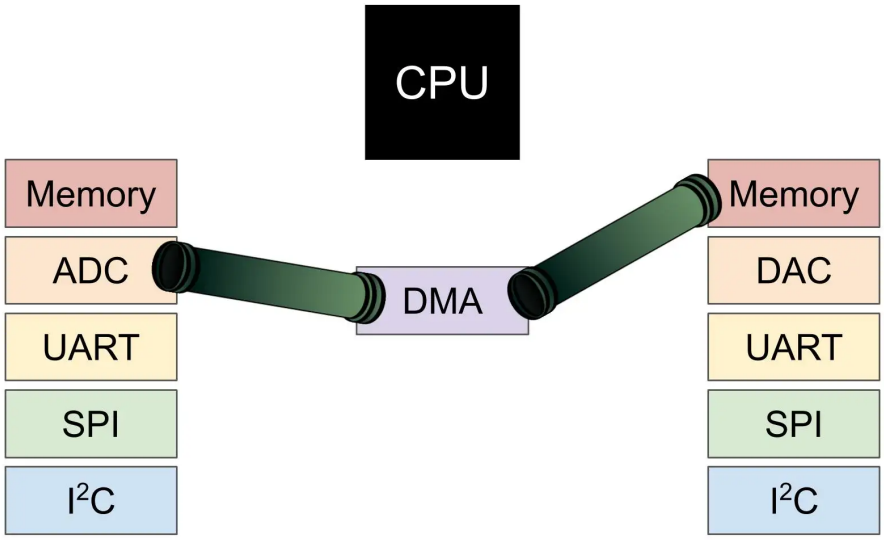



FONTE: o Autor (2019)

Decidiu-se operar o ADC em dois modos: simples e contínuo. No modo simples, o ADC realiza uma única aquisição por *trigger*. Já em modo contínuo, o ADC opera ininterruptamente, preenchendo um *buffer* com auxílio do DMA em segundo plano. Os dois modos foram utilizados e testados durante o desenvolvimento do projeto.

A utilização do DMA em conjunto com o ADC possui duas vantagens principais: a maior velocidade de escrita na memória e também o *offloading* da CPU, ou seja, o fato da CPU ficar livre para executar outras ações, sem ser carregada para realizar o processamento e armazenamento dos dados provenientes do ADC na memória. A Figura 24 ilustra esta técnica.

FIGURA 24 – FUNCIONAMENTO DO ADC EM CONJUNTO COM O DMA




FONTE: Hymel (2019)

### 4.2.2 Transmissão de dados

A transmissão de dados é feita seguindo o protocolo HTTP, conforme já discutido no Capítulo 2.4 deste documento. O cabeçalho das solicitações do projeto é apresentado a seguir:

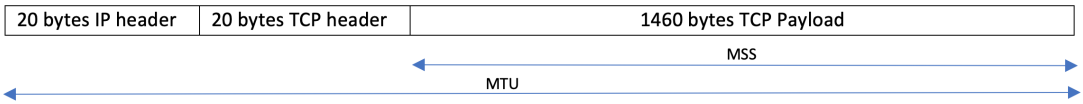
```
POST /script.php HTTP/1.1\r\n
Host: scopewifi.ddns.net\r\n
Authorization: BASIC YWRtaW5fc2NvcGU6dGNjMjAxOUFETUlO\r\n
Content-Type: application/x-www-form-urlencoded\r\n
```

```
Content-Length: X
```

Pode-se observar que o usuário e senha de autenticação são codificados no padrão de Base64 (BASE64ORG, 2019). Além disso, o tamanho de conteúdo (*Content-Length*) é variável de acordo com a quantidade de amostras enviadas no pacote.

Uma restrição importante para a transmissão dos dados é o tamanho do *payload* do protocolo TCP/IP, também conhecido como *Maximum Segment Size* (MSS), ou seja, o tamanho disponível para envio de dados em cada pacote. Por padrão, este espaço é de 1460 bytes no protocolo TCP/IP, que conta com mais 20 bytes de cabeçalho TCP e outros 20 bytes de cabeçalho IP, totalizando 1500 bytes equivalentes ao *Maximum Transmission Unit* (MTU) do protocolo (KUMAR, 2019). Considerando isto e o fato de que cada amostra ocupa 5 bytes (4 bytes do valor + 1 byte do separador), é possível enviar pouco mais de 250 amostras por pacote. A Figura 25 apresenta a composição de *frame* TCP/IP e as suas divisões.

FIGURA 25 – QUADRO TCP: MTU E MSS




FONTE: Kumar (2019)

Sendo assim, foi desenvolvido um algoritmo para viabilizar o envio de amostras de uma mesma sequência em pacotes distintos, as quais são reagrupadas posteriormente pelo *script* do servidor. Desta maneira, é possível utilizar um número superior de amostras do que pode-se enviar em um único pacote TCP.

## 4.3 PARTE DO APLICATIVO *ANDROID*

O aplicativo *Android* foi projetado para servir como tela do osciloscópio, fornecendo ao mesmo tempo a possibilidade de enviar comandos para ajuste e controle do osciloscópio via rede sem fio.

Para reproduzir o sinal na tela de um *smartphone*, optou-se pelo uso da classe *WebView* do *Android*, a qual permite exibir conteúdo Web como sendo parte do layout

da tela em questão. Desta forma, é possível reproduzir o gráfico do sinal gerado pelo *Dygraphs* e integrá-lo ao aplicativo.

Para o envio de comandos, foi utilizada a biblioteca *Volley*, que permite o envio de solicitações HTTP de maneira fácil e direta (DEVELOPERS, 2019b). Com o auxílio desta biblioteca, é possível enviar comandos do aplicativo para o servidor, aonde o *script* em PHP é capaz de extrair os dados e efetuar aos modificações necessárias.

Inicialmente, três controles foram previstos na interface:

- *Run/Stop* - pausar ou continuar a aquisição em tempo real;
- *Refresh* - permite atualizar a página caso necessário;
- *Time/Div* - este botão permite modificar a escala horizontal do gráfico (tempo), através de um *knob*, buscando relembrar o controle de um osciloscópio clássico de laboratório (MENOZZI, 2017).

Por fim, cabe citar o uso da classe *WebViewDatabase*, que possibilita o armazenamentos das credenciais de autentição HTTP, sendo necessária para estabelecer a conexão com o servidor (DEVELOPERS, 2019c).

# 5 RESULTADOS

## 5.1 DESEMPENHO DO SERVIDOR

Para avaliar o desempenho do servidor, foi utilizado o Apache Bench, uma ferramenta para teste e *benchmarking* frequentemente utilizada por desenvolvedores Web. Ele permite a simulação de testes de carga a fim de avaliar o desempenho de servidores. Sendo executado diretamente no terminal de comando, nele é possível definir quantas requisições são desejadas e quantas delas podem ser feitas simultaneamente durante o teste (TARTARI, 2018). A seguir estão listados os principais resultados obtidos no teste:

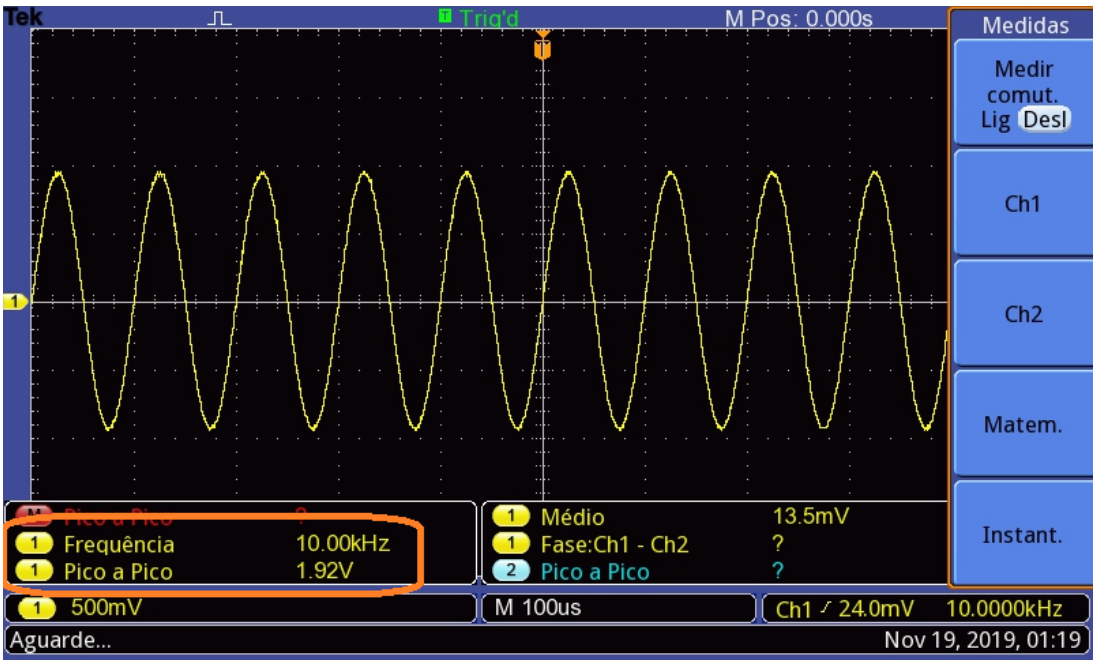
- Software do servidor: Apache/2.4.29;
- *Hostname* do servidor: scopewifi.ddns.net;
- Porta do servidor: 80;
- Tempo total do teste: 0.139 segundos;
- Solicitações completadas: 100;
- Solicitações com falha: 0;
- Total de dados transferidos: 71800 bytes;
- Total de dados HTML: 46500 bytes;
- Solicitações médias por segundo: 720,34;
- Tempo médio por solicitação: 13,882 ms;

## 5.2 TAXA DE AMOSTRAGEM PRÁTICA

De maneira a verificar a taxa de amostragem real do ADC do STM32L475, duas estratégias distintas foram empregadas, de acordo com o modo de funcionamento do conversor. Para ambos os modos de funcionamento, um *buffer* foi preenchido com 1000

amostras. Como sinal de entrada, um sinal senoidal de 10kHz foi utilizado, conforme mostra a Figura 26.

FIGURA 26 – SINAL DE ENTRADA SENOIDAL DE 10KHZ.




FONTE: o Autor (2019)

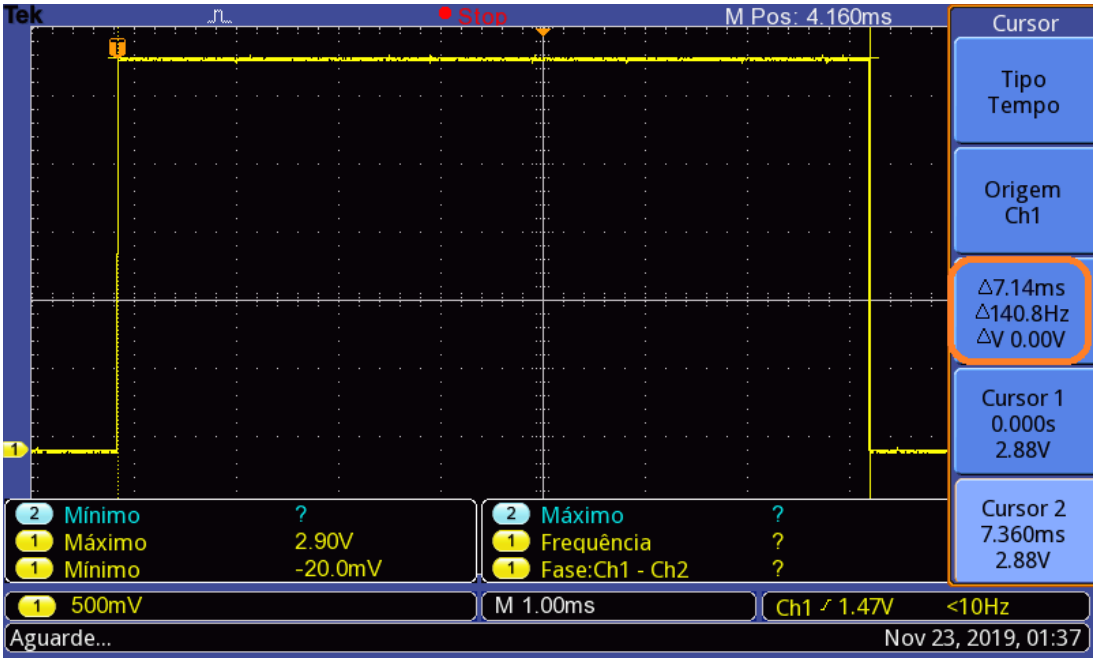
### 5.2.1 Modo de aquisição simples

Para obter a taxa de amostragem neste modo, o LED2 do kit é ativado antes do *loop* que realiza "*SAMPLE_MAX*" amostras em modo simples (neste caso, 1000 amostras), sendo desativado assim que o *loop* é completado, conforme o trecho de código C a seguir:

```
HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_SET);
for (int i=0; i<SAMPLE_MAX; i++){
    /* ADC READ */
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, 1);
    data[i] = HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
}
HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
```

Utilizando um osciloscópio de bancada para visualizar o sinal sobre LED2, obteve-se a forma de onda apresentada na Figura 27. Ao dividir o tempo de aquisição obtido de $7,14ms$ por pelo total de amostras (1000), tem-se $7,14\,\mu s$ como período de amostragem, o que é equivalente a $\mathbf{f_S = 140k\,Hz}$. Esta taxa de amostragem obtida é pequena, visto que osciloscópios digitais modernos possuem taxas de amostragem da ordem de $f_S = 1\,GHz$.

FIGURA 27 – TEMPO DE AQUISIÇÃO DE 1000 AMOSTRAS - ADC EM MODO SIMPLES




FONTE: o Autor (2019)

### 5.2.2 Modo de aquisição contínua com DMA

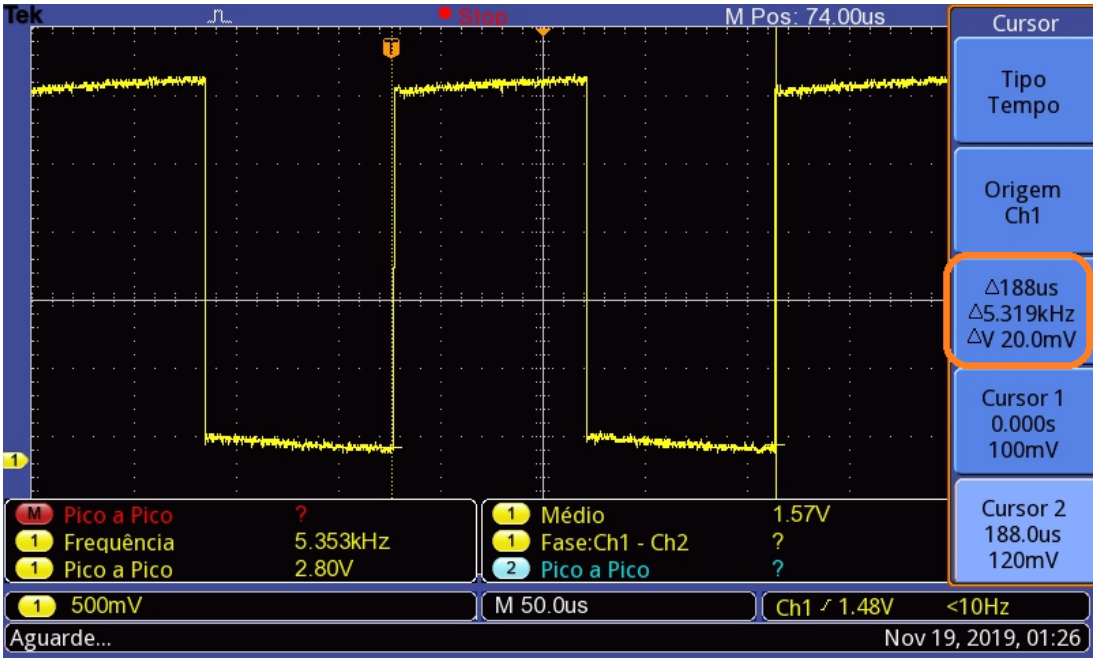
Para medir na prática a taxa de amostragem neste modo, foi tirado proveito das funções chamadas pelas interrupções do DMA quando o *buffer* está parcialmente preenchido e quando ele está completamente preenchido, segundo o trecho de código a seguir:

```
// Called when first half of buffer is filled
void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc) {
  HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_SET);
}
```

```
// Called when buffer is completely filled
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
  HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
}
```

A mesma estratégia de ativação/desativação do LED2 foi empregada, porém neste caso é preciso considerar um período completo da forma de onda, já que quando o *buffer* está preenchido pela metade o LED2 é ativado, e quando ele está completamente preenchido o LED2 é desativado. A Figura 28 mostra o tempo de aquisição obtido. Pode-se verificar que as 1000 amostras foram realizadas dentro de um intervalo de $188\mu s$, ou seja, 1 amostra a cada $0,188\mu s$, o que é equivalente a $\mathbf{5,33\ Msps}$, exatamente a taxa de conversão exposta no *datasheet* do MCU (STMICROELECTRONICS, 2018).

FIGURA 28 – TEMPO DE AQUISIÇÃO DE 1000 AMOSTRAS - ADC EM MODO CONTÍNUO COM DMA




FONTE: o Autor (2019)

Consequentemente, o uso do ADC em conjunto com o DMA provou-se muito mais vantajoso, não só pela taxa de amostragem superior, mas também pelo fato de não sobrecarregar a CPU ao realizar a aquisição de amostras em segundo plano. Outro ponto que deve ser citado é que a banda de frequências comumente estudada nas

práticas de laboratórios de eletrônica costuma ser da ordem de algumas centenas de kilo-hertz (kHz). Sendo assim, a taxa de amostragem obtida em modo contínuo com auxílio do DMA é largamente suficiente para a utilização do sistema na graduação.

## 5.3 VALOR DO *QUANTUM* E RUÍDO DE QUANTIZAÇÃO PRÁTICOS

Por meio da expressão (2.5), foi possível calcular o valor do *quantum* do osciloscópio, ou seja, o menor valor analógico que ele pode tratar. Considerando $V_max = 3,3\,V$, $V_min = 0\,V$ e $N = 12\,bits$:

$$V_q = \frac{(V_{max} - V_{min})}{2^N - 1} = \frac{(3,3 - 0)}{2^{16} - 1} = \frac{3,3}{4095} \approx 0,8\,mV \tag{5.1}$$

Da mesma maneira, servindo-se da equação Ecuación 2.6, o ruído de quantização prático foi obtido:

$$\frac{S}{N} = -20\,log(\frac{1}{2^N}) = 20\,log(\frac{1}{2^{16}}) \approx -96\,dB \tag{5.2}$$

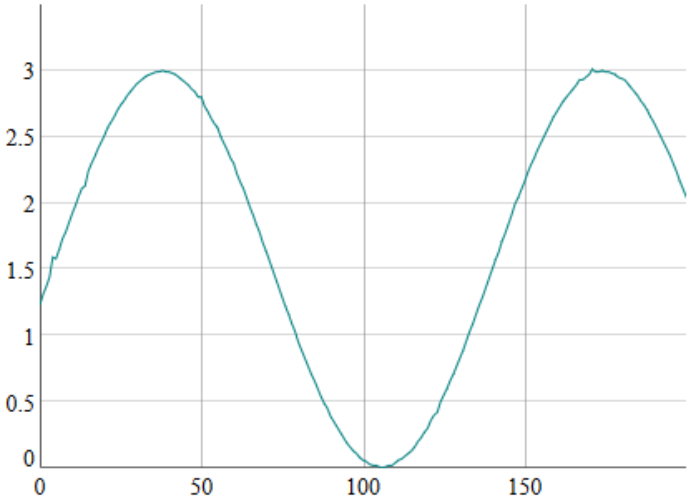
## 5.4 VISUALIZAÇÕES DE FORMAS DE ONDA

Nesta seção são apresentados os resultados práticos de visualização de formas de onda com sinais reais, para os dois modos de aquisição do ADC abordados. Para todos os resultados apresentados, o sinal de entrada foi uma onda senoidal, $3\,V_{pp}$ com $1,5\,V$ de *offset* (de modo a obter um sinal totalmente compreendido entre $3,3\,V$ e $0\,V$. A frequência do sinal foi variada de maneira a permitir a avaliação da resolução do osciloscópio em toda a banda de funcionamento, com o intuito de identificar possíveis distorções ou outros erros na exibição das formas de onda.

### 5.4.1 Modo de aquisição simples

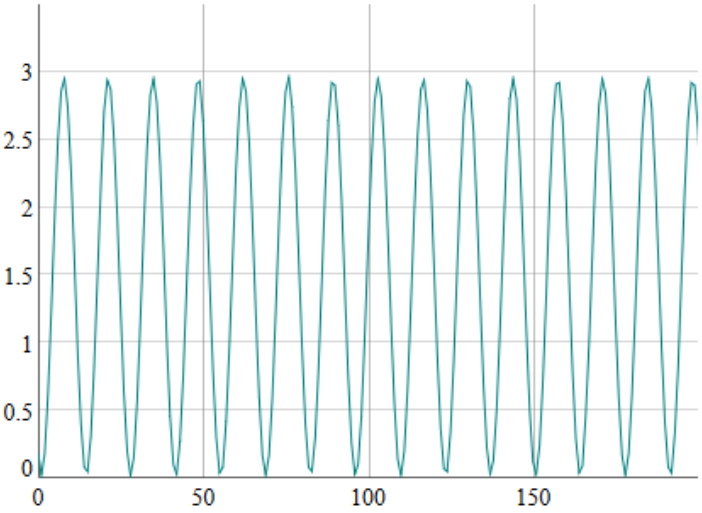
Neste modo o ADC opera com uma frequência de amostragem real $f_S = 140\,kHz$. Os resultados apresentados na Figura 29 e Figura 30 mostram a forma de onda reproduzida pelo osciloscópio para sinais de entrada de $f_e = 1\,kHz$ e $f_e = 10\,kHz$, respectivamente. O sistema demonstrou um desempenho satisfatório para um total de 200 amostras por quadro, com apenas algumas pequenas distorções podendo ser vistas.

FIGURA 29 – FORMA DE ONDA PARA UM SINAL DE ENTRADA DE 1KHZ




FONTE: o Autor (2019)

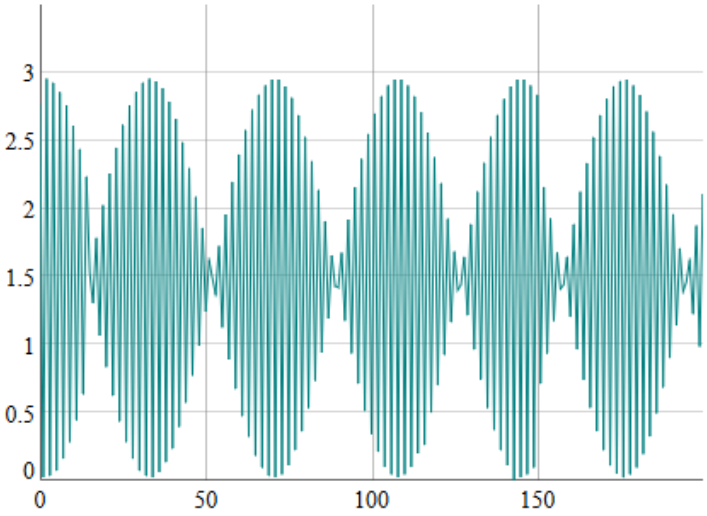
FIGURA 30 – FORMA DE ONDA PARA UM SINAL DE ENTRADA DE 10KHZ




FONTE: o Autor (2019)

Em seguida, a frequência de entrada foi aproximada ao limite do critério de Nyquist (NYQUIST, 1928), com $f_e = 70\,kHz$. Neste cenário, é possível verificar uma distorção importante no sinal de saída, conforme mostra a Figura 31.

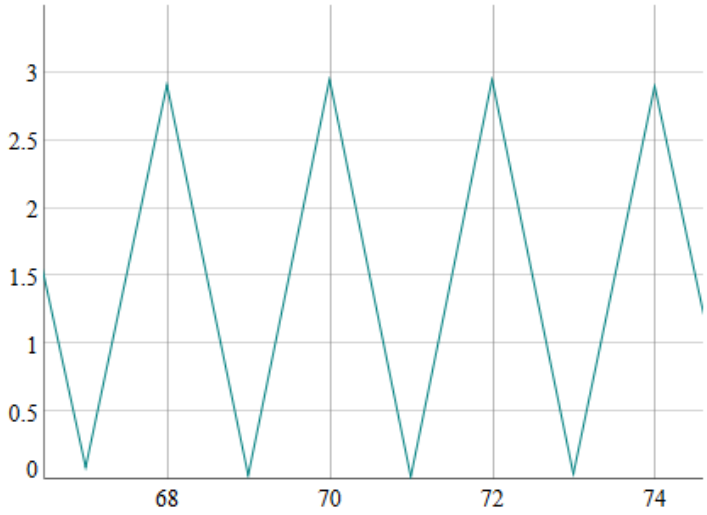
FIGURA 31 – FORMA DE ONDA PARA UM SINAL DE ENTRADA DE 70KHZ




FONTE: o Autor (2019)

Através do recurso de *zoom*, é possível verificar que somente 2 pontos da onda estão sendo amostrados por período, o que acaba criando uma reprodução em forma triangular do sinal de origem, efeito apresentado na Figura 32.

FIGURA 32 – FORMA DE ONDA COM *ZOOM* PARA UM SINAL DE ENTRADA DE 70KHZ



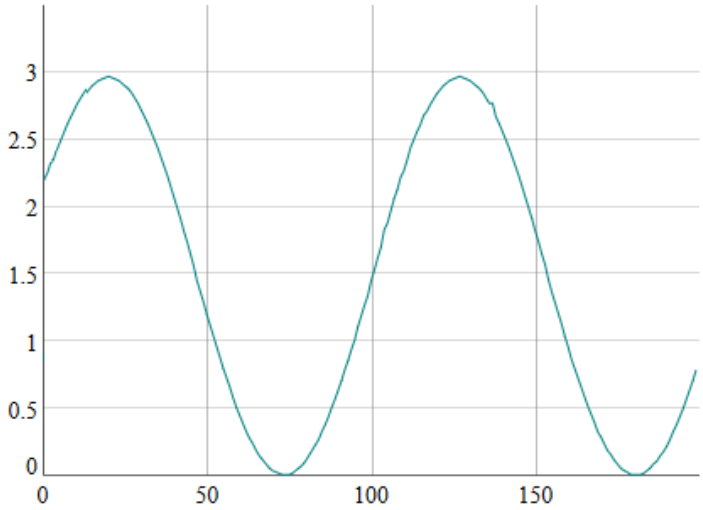

FONTE: o Autor (2019)

### 5.4.2 Modo de aquisição contínua com DMA

Neste modo o ADC opera com uma frequência de amostragem real $f_S = 5,33\,MHz$. Os resultados apresentados na Figura 33 e Figura 34 mostram a forma

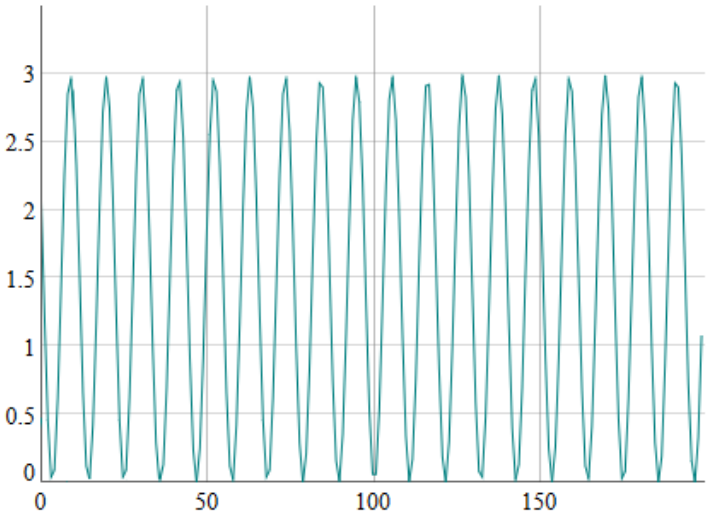
de onda reproduzida pelo osciloscópio para sinais de entrada de $f_e = 50\,kHz$ e $f_e = 500\,kHz$, respectivamente. O sistema demonstrou novamente um desempenho satisfatório para um total de 200 amostras por quadro. Cabe citar que, para esta nova frequência de amostragem muito mais rápida, é necessário ajustar a escala de tempo para que sinais mais lentos possam ser visualizados por completo (pelo menos 1 período).

FIGURA 33 – FORMA DE ONDA PARA UM SINAL DE ENTRADA DE 50KHZ




FONTE: o Autor (2019)

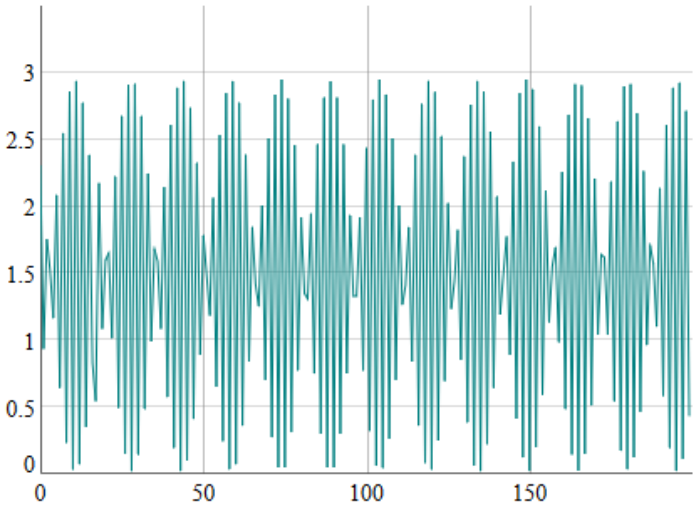
FIGURA 34 – FORMA DE ONDA PARA UM SINAL DE ENTRADA DE 500KHZ




FONTE: o Autor (2019)

Posteriormente, a frequência de entrada foi aproximada ao limite do critério de Nyquist (NYQUIST, 1928), sendo agora $f_e = 2,5\,MHz$. Foi possível verificar novamente uma distorção significativa no sinal de saída, conforme mostra a Figura 35.

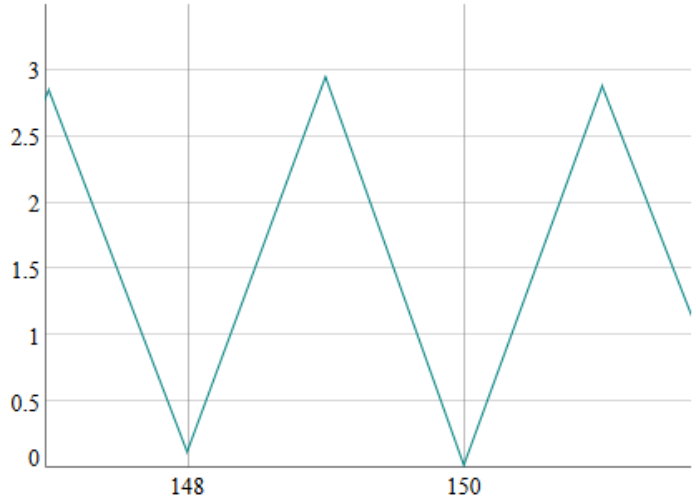
FIGURA 35 – FORMA DE ONDA PARA UM SINAL DE ENTRADA DE 2,5MHZ




FONTE: o Autor (2019)

Utilizando mais uma vez o recurso de *zoom*, foi possível verificar que somente 2 pontos da onda estão sendo amostrados por período, como mostra a Figura 36 .

FIGURA 36 – FORMA DE ONDA COM *ZOOM* PARA UM SINAL DE ENTRADA DE 2,5MHZ

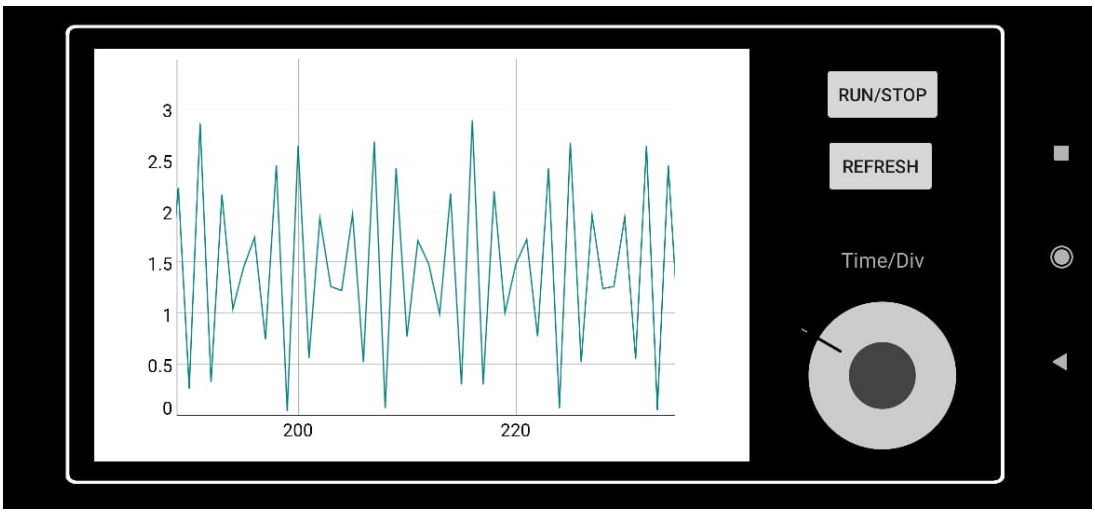



FONTE: o Autor (2019)

## 5.5 APLICATIVO *ANDROID*

O aplicativo *Android* mostrou ser capaz de cumprir seu papel, exibindo as formas de ondas através da *WebView* e enviando os comandos previstos para o servidor através da biblioteca HTTP *Volley*. A Figura 37 apresenta a interface do aplicativo, onde é possível visualizar a tela do osciloscópio à esquerda e os botões de controle à direita.

FIGURA 37 – TELA DO OSCILOSCÓPIO NO APLICATIVO




FONTE: o Autor (2019)

# 6 CONCLUSÃO

## 6.1 CONCLUSÃO DO TRABALHO DESENVOLVIDO

Conforme o avanço da tecnologia, faz-se necessário analisar o seu impacto na sociedade, principalmente nos hábitos de cada uma das pessoas. Com o advento da *Internet of things* e a excepcional facilidade de acesso aos *smartphones*, os aplicativos para celular têm se tornado uma ferramenta de uso constante no dia a dia de milhares de pessoas, facilitando a comunicação, o entretenimento e o acesso à informação.

Na esfera acadêmica, é imprescindível repensar a relação de aluno e professores com estas tecnologias. Dentro deste propósito, o projeto do osciloscópio digital monitorado por aplicativo mostrou-se muito relevante ao aliar a irreprimível tendência do uso de *smartphones* com um equipamento primordial para o ensino e desenvolvimento da eletrônica.

Dado o fato de que os dados de amostragem são enviados e disponibilizados em um servidor centralizado, múltiplas conexões podem ser feitas simultaneamente, viabilizando assim não só o uso individual, mas também o uso pedagógico durante aulas práticas. Um exemplo desta aplicação seria o professor conectar-se como "*admin*"do sistema, controlando os sinais de entrada com o hardware em mãos, e os alunos se conectarem através de sessões de *usuário*, podendo acompanhar as medidas e oscilações das formas de onda.

Obviamente, a performance do protótipo desenvolvido não pode ser comparada àquela obtida por equipamentos mais robustos e sofisticados, mesmo sendo extremamente satisfatória tendo em vista a relação custo/benefício. Porém, para fins acadêmicos, pode-se assegurar a viabilidade do sistema, cujo potencial oportuniza seu aperfeiçoamento.

O kit B-L475E-IOT01A demonstrou toda a sua capacidade para o desenvolvimento de projetos de *IoT*, ao dispor dos mais variados tipos de sensores e módulos de comunicação integrados. Aliando isto ao baixo consumo e vasto repertório de periféricos do microcontrolador STM32, foi possível implementar com sucesso as funcionalidades de hardware necessárias ao trabalho em questão.

O monitoramento dos gráficos de sinais via aplicativo *Android*, com ajuda da biblioteca *Dygraphs*, também se mostrou simples e efetivo, permitindo ao usuário controlar a escala de tempo de visualização bem como definir as configurações de conexão ao servidor.

Finalmente, o desenvolvimento deste trabalho foi extremamente satisfatório para o autor, visto que este pôde aprimorar conhecimentos obtidos anteriormente, aprender novos conceitos, técnicas e linguagens, além de aprofundar seu interesse pela *IoT*, microcontroladores e redes de computadores.

## 6.2 TRABALHOS FUTUROS

Este projeto definitivamente deixa aberta a porta para aprimoramentos e adaptações de sua proposta, visto que se trata essencialmente de um PoC (do inglês, *Proof of Concept*).

Em termos de hardware, é sugerida a concepção de uma placa de circuito impresso dedicada ao projeto, contendo somente o hardware necessário a aplicação. Isto diminuiria as dimensões físicas e o consumo do circuito como um todo, além de garantir um melhor isolamento dos sinais.

Ainda no escopo do hardware, um circuito de adaptação de potência ampliaria a gama de tensões mensuráveis pelo osciloscópio, fazendo a conversão de níveis de tensão mais elevados para o intervalo digital do ADC (0V à 3,3V). A adição de conectores fêmea do tipo BNC também poderia aumentar a compatibilidade do osciloscópio com os padrões de cabo normalmente utilizados em laboratório.

No aspecto de software, inúmeras funcionalidades podem ser adicionadas ao aplicativo, de forma a aumentar ainda mais a interatividade com o osciloscópio. Com o intuito de realizar medidas personalizadas, o recurso de cursores pode ser adicionado, bem como um algoritmo de *trigger* para estabilizar a visualização do sinal na tela. No que diz respeito a parte visual, as possibilidades são infinitas, podendo-se personalizar a interface de acordo com as necessidades do usuário.

# REFERÊNCIAS

ANNIE, App. **Spotlight on Consumer App Usage**. 2017. Disponível em: <http://files.appannie.com.s3.amazonaws.com/reports/1705_Report_Consumer_App_Usage_EN.pdf>. Acesso em: 9 set. 2019. Citado 1 vez na página 15.

BASE64ORG. **Base64 Decode and Encode - Online**. 2019. Disponível em: <https://www.base64decode.org/>. Acesso em: 19 out. 2019. Citado 1 vez na página 49.

COMPUTING, Measuring. **Analog to Digital Conversion**. 2019. Disponível em: <https://www.mccdaq.com/handbook/chapt_2.aspx>. Acesso em: 17 nov. 2019. Citado 1 vez na página 21.

CONDER, Shane; DARCEY, Lauren. **Learn Java for Android Development: Introduction to Java**. 2010. Disponível em: <https://code.tutsplus.com/tutorials/learn-java-for-android-development-introduction-to-java--mobile-2604>. Acesso em: 15 nov. 2019. Citado 1 vez na página 39.

CORBET, Chris. **MATLAB EQ: Background on Equalization**. 2019. Disponível em: <https://cnx.org/contents/kF_AG3Rc@1/MATLAB-EQ-Background-on-Equalization>. Acesso em: 10 nov. 2019. Citado 0 vez na página 20.

CORDEIRO, Fillipe. **[Guia] Android para Desenvolvedor Java | AndroidPro**. 2017. Disponível em: <https://www.androidpro.com.br/blog/carreira/desenvolvedor-java/>. Acesso em: 15 nov. 2019. Citado 1 vez na página 40.

DEVELOPERS, Google. **Conheça o Android Studio | Android Developers**. 2019. Disponível em: <https://developer.android.com/studio/intro>. Acesso em: 15 nov. 2019. Citado 1 vez na página 37.

_____. **Web-based content**. 2019. Disponível em: <https://developer.android.com/training/volley>. Acesso em: 7 nov. 2019. Citado 1 vez na página 50.

_____. **WebViewDatabase**. 2019. Disponível em: <https://developer.android.com/reference/android/webkit/WebView>. Acesso em: 12 nov. 2019. Citado 1 vez na página 50.

DOWNEY, Tim. **Timeout and Keep Alive Directives**. Disponível em: <https://users.cs.fiu.edu/~downeyt/webdev/timeout>. Acesso em: 25 out. 2019. Citado 4 vezes nas páginas 43, 44.

DYGRAPHS. **Dygraphs Tutorial**. 2019. Disponível em: <http://dygraphs.com/tutorial.html>. Acesso em: 29 set. 2019. Citado 1 vez na página 36.

EBRAHIM, Mokhtar. **Install, Configure, and Troubleshoot Linux Web Server (Apache)**. 2017. Disponível em: <https://likegeeks.com/linux-web-server/>. Acesso em: 23 out. 2019. Citado 1 vez na página 42.

EFSTATHIOU, Prof. C. E. **Nyquist - Shannon Signal Sampling Theorem**. 2019. Disponível em: <http://195.134.76.37/applets/AppletNyquist/Appl_Nyquist2.html>. Acesso em: 12 nov. 2019. Citado 1 vezes nas páginas 20, 21.

EIS, Diego. **O básico: O que é HTML?**: 2011. Disponível em: <https://tableless.com.br/o-que-html-basico/>. Acesso em: 15 nov. 2019. Citado 1 vez na página 39.

ELLINGWOOD, Justin. **How To Set Up Password Authentication with Apache on Ubuntu 14.04**. 2015. Disponível em: <https://www.digitalocean.com/community/tutorials/how-to-set-up-password-authentication-with-apache-on-ubuntu-14-04>. Acesso em: 30 out. 2019. Citado 1 vez na página 43.

FOUNDATION, The Apache Software. **About the Apache HTTP Server Project**. 2019. Disponível em: <https://httpd.apache.org/ABOUT_APACHE.html>. Acesso em: 27 out. 2019. Citado 1 vez na página 36.

GEEKSFORGEEKS. **C Language Introduction**. 2019. Disponível em: <https://www.geeksforgeeks.org/c-language-set-1-introduction/>. Acesso em: 12 nov. 2019. Citado 1 vez na página 39.

GROUP, The PHP. **PHP Manual**. 2019. Disponível em: <https://www.php.net/manual/en/preface.php>. Acesso em: 5 out. 2019. Citado 1 vez na página 39.

HOCK-CHUAN, Chua. **In Introduction to HTTP Basics**. 2009. Disponível em: <https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html>. Acesso em: 16 out. 2019. Citado 0 vezes nas páginas 27–29.

HYMEL, Shawn. **Getting Started with STM32 - Working with ADC and DMA**. 2019. Disponível em: <https://www.digikey.com/en/maker/projects/getting-started-with-stm32-working-with-adc-and-dma/f5009db3a3ed4370acaf545a3370c30c>. Acesso em: 17 out. 2019. Citado 0 vez na página 48.

INVENTEK SYSTEMS. **ISM43362-M3G-L44 Product Specification**. 2017. Disponível em: <https://www.inventeksys.com/wp-content/uploads/ISM43362_M3G_L44_Functional_Spec.pdf>. Acesso em: 2 nov. 2019. Citado 0 vezes nas páginas 25, 26.

NO-IP. **No-IP about us**. 2019. Disponível em: <https://www.noip.com/about>. Acesso em: 17 out. 2019. Citado 1 vez na página 35.

KUMAR, Shashank Suresh. **How TCP segment size can affect application traffic flow**. 2019. Disponível em: <https://medium.com/walmartlabs/how-tcp-segment-size-can-affect-application-traffic-flow-7bbceed5816e>. Acesso em: 18 out. 2019. Citado 1 vez na página 49.

LECROY, Teledyne. **Teledyne LeCroy - 100 GHz Real-Time Oscilloscope**. 2019. Disponível em: <https://teledynelecroy.com/100ghz/>. Acesso em: 17 nov. 2019. Citado 1 vezes nas páginas 18, 19.

MDN CONTRIBUTORS. **Uma visão geral do HTTP - HTTP | MDN**. 2019. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Acesso em: 13 nov. 2019. Citado 1 vez na página 27.

MEIRELLES, Prof. Fernando S. **30ª Pesquisa Anual do FGVcia da FGV/EAESP, 2019**. 2019. Disponível em: <https://eaesp.fgv.br/sites/eaesp.fgv.br/files/noticias2019fgvcia_2019.pdf>. Acesso em: 9 set. 2019. Citado 1 vez na página 15.

MENOZZI, Beppi. **Rotary Knob Selector**. 2017. Disponível em: <https://github.com/BeppiMenozzi/Knob>. Acesso em: 27 out. 2019. Citado 1 vez na página 50.

MOBATEK. **MobaXterm free Xserver and tabbed SSH client for Windows**. 2019. Disponível em: <https://mobaxterm.mobatek.net/>. Acesso em: 9 set. 2019. Citado 2 vezes nas páginas 33, 34.

NYQUIST, Harry. Certain topics in telegraph transmission theory. In: 2. TRANS. AIEE. [S.l.]: IEEE, abr. 1928. v. 90, p. 617–644. Feb 2002 Archived 2013-09-26 at the Wayback Machine. Citado 3 vezes nas páginas 19, 56, 59.

OPPENHEIM, Alan V.; WILLSKY, Alan S. **Sinais e Sistemas**. 2. ed. São Paulo, Brasil: Pearson Prentice Hall, 2010. p. 305–307. Citado 1 vez na página 19.

ROUSE, Margaret. **What is oscilloscope? - Definition from WhatIs.com**. 2005. Disponível em: <https://whatis.techtarget.com/definition/oscilloscope>. Acesso em: 17 nov. 2019. Citado 2 vez na página 18.

SHANNON, Claude E. Communication in the presence of noise. In: 2. PROCEEDINGS of the Institute of Radio Engineers. [S.l.]: IEEE, jan. 1949. v. 86, p. 10–21. Archived 2010-02-08 at the Wayback Machine. Citado 4 vezes nas páginas 19–21.

SOCIETY, The Internet. **Hypertext Transfer Protocol – HTTP/1.1**. 1994. Disponível em: <https://tools.ietf.org/html/rfc2616>. Acesso em: 13 nov. 2019. Citado 1 vez na página 27.

SOUZA, Ivan de. **HTTP: entenda o que é, para que serve e como funciona**. 2019. Disponível em: <https://rockcontent.com/blog/http/>. Acesso em: 13 nov. 2019. Citado 1 vez na página 27.

STMICROELECTRONICS. **Description of STM32L4/L4+ HAL and Low-layer drivers - en.DM00173145.pdf**. 2017. Disponível em: <https://www.st.com/content/ccc/resource/technical/document/user_manual/63/a8/8f/e3/ca/a1/4c/84/DM00173145.pdf/files/DM00173145.pdf/jcr:content/translations/en.DM00173145.pdf>. Acesso em: 1 out. 2019. Citado 1 vez na página 24.

_____. **RM0351 - Reference manual - STM32L4x5 and STM32L4x6 advanced Arm®-based 32-bit MCUs**. 2018. Disponível em: <https://www.st.com/content/ccc/resource/technical/document/reference_manual/02/35/09/0c/4f/f7/40/03/DM00083560.pdf/files/DM00083560.pdf/jcr:content/translations/en.DM00083560.pdf>. Acesso em: 13 set. 2019. Citado 5 vezes nas páginas 24–26, 54.

_____. **STM32CubeIDE - Integrated Development Environment for STM32**. 2019. Disponível em: <https://www.st.com/en/development-tools/stm32cubeide.html>. Acesso em: 18 set. 2019. Citado 0 vezes nas páginas 32, 33.

_____. **UM2153 - User manual - Discovery kit for IoT node, multi-channel communication with STM32L4**. 2019. Disponível em: <https://www.st.com/content/ccc/resource/technical/document/user_manual/group0/b1/b8/7a/f2/f7/8d/4b/

6b/DM00347848/files/DM00347848.pdf/jcr:content/translations/en.DM00347848.pdf>. Acesso em: 13 set. 2019. Citado 0 vez na página 23.

TANENBAUM, Andrew S.; WETHERALL, David J. **Computer Networks**. 5. ed. São Paulo, Brasil: Pearson Prentice Hall, 2012. p. 552–560. Citado 2 vez na página 29.

TARTARI, Tiago. **Usando Apache Bench para Teste de Carga e Análise de Performance**. 2018. Disponível em: <https://medium.com/@tiago.tartari/usando-apache-bench-para-teste-de-carga-e-an%5C%C3%5C%A1lise-de-performance-7dc92321de84>. Acesso em: 5 nov. 2019. Citado 1 vez na página 51.

# APÊNDICES

# APÊNDICE A – REPOSITÓRIO DOS CÓDIGOS DESENVOLVIDOS

De modo a permitir o acesso a integridade dos códigos do projeto, porém sem estender demasiadamente este documento, optou-se por disponibilizar todos os arquivos em um repositório *GitHub* permanente. Isto garante o acesso a todos os arquivos necessários a reprodução do sistema, além de aumentar as possibilidades de divulgação do mesmo. Outra vantagem é que quaisquer alterações futuras serão automaticamente espelhadas no repositório. O acesso pode ser feito pelo *link* a seguir:

`https://github.com/eriknayan/Scope-WiFi-UFPR`