UNIVERSIDADE FEDERAL DO PAMPA
LICENCIATURA EM FÍSICA

DANIEL FONSECA CORRADINI FERRANDO

INSTRUMENTAÇÃO [SEMI-] AUTOMATIZADA, DAQ-DAC BASEADA EM PYTHON-ARDUINO

BAGÉ

2019

DANIEL FONSECA CORRADINI FERRANDO

INSTRUMENTAÇÃO [SEMI-] AUTOMATIZADA, DAQ-DAC BASEADA EM PYTHON-ARDUINO

Trabalho de Conclusão de Curso apresentado ao Curso de Licenciatura em Física da Universidade Federal do Pampa..

Orientador: Edson Massayuki Kakuno

BAGÉ

2019

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

F184i Ferrando Fonseca Corradini, Daniel
Instrumentação [Semi-] automatizada, DAQ-DAC Baseada em Python-Arduino / Daniel Fonseca Corradini Ferrando. – Bagé, 2019-
181p.

Trabalho de Conclusão de Curso(Graduação) - - Universidade Federal do Pampa, LICENCIATURA EM FÍSICA, 2019.
"Orientação: Edson Massayuki Kakuno"
.
1. Arduino. 2. Instrumentação física. 3. Python. I. Título.

DANIEL FONSECA CORRADINI FERRANDO

INSTRUMENTAÇÃO [SEMI-] AUTOMATIZADA, DAQ-DAC BASEADA EM PYTHON-ARDUINO

Trabalho de Conclusão de Curso apresentado ao Curso de Licenciatura em Física da Universidade Federal do Pampa.

Trabalho de Conclusão de Curso defendido e aprovado em: pela seguinte banca examinadora:

Edson Massayuki Kakuno
Orientador

Prof. dr. Bruno Silveira Neves
Unipampa

John Welvins Barros de Arraújo
FURG

Este trabalho é dedicado à Daniela Moro, que por algum motivo me aguenta.

“Few people have the wisdom to prefer
the criticism that would do them good,
to the praise that deceives them.”
(François de La Rochefoucauld,
Réflexions ou Sentences et Maximes Morales)

RESUMO

Este trabalho tem como objetivo a criação de uma instrumentação para laboratórios de ensino, tanto a nível médio como a nível superior, que se aproxime da aplicabilidade de laboratórios de pesquisa e laboratórios industriais. Nesse sentido, o projeto busca fornecer uma coleta de dados automatizada e tratamento estatístico. A instrumentação desenvolvida é controlada via *software*, com linguagem de alto nível *Python* e o controle do nível de dispositivo através da plataforma *Arduino*. Como as linguagens são inteligíveis e possuem uma comunidade expansiva de desenvolvedores na rede, o que facilita a utilização do instrumento desenvolvido. A proposta é criar módulos que possam permitir sua expansão ao longo do tempo por meio da criação de outros módulos. Inicialmente, é proposto um módulo de aquisição de dados composto por um conversor analógico digital (ADC), produzido pela *Analog Devices* (AD7794), que possui um estágio de pré-processamento de sinal com ganho ajustável via *software* e diversas entradas multiplexadas. O outro módulo desenvolvido foi um conversor digital analógico (*DAC*) que permite ao usuário conectá-lo a um circuito de potência, como um amplificador operacional, para alimentar circuitos ou acionar dispositivos elétricos. Conclui-se que essa instrumentação pode servir como base para a realização de experimentos automatizados no ensino de Física das escolas públicas brasileiras, trazendo às salas de aula a discussão da física teórica por trás de cada experimento desenvolvido. Ademais, pode permitir que instituições interessadas tenham acesso à uma instrumentação de qualidade e baixo custo, acessível e aberta a usuários de diferentes segmentos.

Palavras-chaves: Arduino. Instrumentação física. Python.

ABSTRACT

The objective of this work is the creation of an instrumentation for teaching laboratories, at medium and higher level, that approaches the applicability of research laboratories and industrial laboratories. In this sense, the project seeks to provide automated data collection and statistical treatment. The instrumentation developed is controlled via *software*, with high level language *Python* and device level control through the *Arduino* platform. As languages are intelligible and have an expansive community of developers on the network, which facilitates the use of the developed instrument. The purpose is to create modules that can allow their expansion over time through the creation of other modules. Initially, it is proposed a data acquisition module composed by a digital analog converter (ADC), produced by *Analog Devices* (AD7794), which has an adjustable gain signal preprocessing stage via *software*. and several multiplexed entries. The other module developed was a digital analog converter (*DAC*) that allows the user to connect it to a power circuit, such as an operational amplifier, to power circuits or drive electrical devices. It is concluded that this instrumentation can serve as a basis for conducting automated experiments in physics teaching in Brazilian public schools, bringing to the classroom the discussion of theoretical physics behind each experiment developed. In addition, it can allow interested institutions to have access to quality and low-cost instrumentation, accessible and open to users from different segments.

Key-words: Arduino. physics Instrumentation. Python.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E DE SIGLAS

ADC Conversor Analógico Digital

CI Circuito Integrado

DAC Conversor Digital Analógico

GUI Interface Grafica

OER *Open Educational Resources*

PCB Placa de Circuito Impresso

LISTA DE SÍMBOLOS

Ω	**Ohm**, Resistência Elétrica
V	**volts**, Potencial Elétrico
A	***Ampère***, Corrente Elétrica

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento tecnológico, ao longo das últimas décadas, tem proporcionado muitas facilidades a vida moderna e transformações no conhecimento e na forma como o adquirimos. Através das tecnologias e o uso globalizado da internet, pesquisadores e estudantes podem compartilhar experiencias e estudos de modo a ampliar a divulgação cientifica de fenômenos físicos de toda parte do mundo. Novas técnicas de pesquisa, como a meta-análise [1] e os estudos de grupos em redes sociais para análise de epidemias, por exemplo, foram desenvolvidas como consequência das mudanças tecnológicas e possibilitam acesso à novas fontes de informações até então não viáveis.

Nos laboratórios de ensino de física, nas escolas de ensino médio e em instituições de ensino superior (IES), os experimentos realizados, em sua maioria, ainda não responderam as mudanças tecnológicas. Ou seja, a aquisição de dados é geralmente feita manualmente (preenchendo uma tabela de dados impressa previamente) e os *kits* experimentais não correspondem as novidades que existem. Desta forma, eles geralmente são utilizados em outros ambientes, a maior parte deles de iniciativa privada. Devido a falta de recursos, ao alto custo dos equipamentos comerciais e uma série de outros fatores restringem à adesão destas tecnologias nas escolas de ensino médio e IES. A automatização (da aquisição de dados) permite uma série de experimentos que seriam praticamente inviáveis de se realizar com a coleta de dados manual, por exemplo a análise estatística de dados com conjuntos maiores que 100 pontos por passo.

No ensino médio das escolas públicas brasileiras, as diretrizes comumente são voltadas a atender os conteúdos solicitados nos currículos. Isso acaba por implicar em barreiras práticas no que tange à elaboração e ao uso de experimentos físicos em sala de aula. Quando são utilizados experimentos em aulas de física, eles acabam sendo na sua maioria, demonstrativos, ou para reforçar o conteúdo que já foi trabalhado pelo docente. Portanto, os experimentos acabam não sendo discutidos da forma que poderiam, pois não envolvem uma participação ativa dos alunos e um olhar exclusivo do professor e da escola para a sua utilização no ensino de física. De acordo com (GRASSELLI; GARDELLI, 2014) , a disciplina de Física é essencialmente apresentada de forma teórica, com foco em exercícios para fixação dos conceitos físicos e matemáticos. Assim, os alunos frequentemente apresentam dificuldades

[1] Uma meta-análise é uma análise estatística que combina os resultados de múltiplos estudos científicos. A meta-análise pode ser realizada quando há vários estudos científicos abordando a mesma questão, com cada estudo individual relatando medidas que se espera que tenham algum grau de erro.

em relacionar os conceitos com os fenômenos observados no seu dia a dia, o que pode causar problemas de entendimento da disciplina e afastá-los desta área. Mesmo existindo laboratórios experimentais, muitos ainda seguem um método tradicional, *id est*, o aluno segue um roteiro em que realiza algumas medidas e os resultados dessas medidas entram em uma sequência - quase mecânica - de cálculos no qual o resultado final converge para o resultado esperado da teoria e raramente existe uma oportunidade, tanto de tempo quanto de disponibilidade experimental, para que o aluno explore diferentes configurações. Desta forma, este trabalho propõe o desenvolvimento de uma instrumentação semi automatizado em que tanto o professor como o aluno possam ter mais flexibilidade no desenvolvimento dos experimentos.

Para isso, buscou-se desenvolver um sistema de aquisição de dados e uma fonte de tensão continua semi automatizada para uso geral em laboratórios de ensino do ensino médio e de graduação e, eventualmente, em laboratórios de pesquisa. O equipamento é composto por um *Arduino* [2], um conversor Analógico digital (ADC) e um conversor digital Analógico (DAC) que controla uma fonte de alimentação. Além disso, o equipamento terá uma interface escrita em Python [3] em que será possível fazer coleta de dados e visualização em tempo real.

O conjunto de características da plataforma descritas acima, pode permitir a elaboração de diversos experimentos controlados e/ou com coleta de dados via computador e com a adição de um mínimo de *hardware* externo. Com isso, pretende-se expandir as possibilidades de desenvolvimento de experimentos no ensino médio das escolas públicas brasileiras. Em busca de facilitar o uso da plataforma desenvolvida, foi escolhido utilizar a plataforma *Arduino* para o controle local dos dispositivos de aquisição de dados e da fonte de alimentação. A linguagem de programação Python foi usada para a comunicação em geral e permite assim a programação (controle) do experimento em si.

---

[2] Plataforma *open source* com entradas analógicas e digitais com capacidade de comunicação em diferentes protocolos como I^2C. Mais detalhes em http://www.Arduino.cc.

[3] Linguagem de programação de alto nível com diversas bibliotecas matemáticas, de análise numérica, interface gráfica, entre outras. Mais detalhes em http://www.python.org.

2 REVISÃO DA LITERATURA

Para o entendimento geral do projeto aqui desenvolvido, é importante separar a área de Física em dois grupos importantes para essa discussão: a Física Teórica e a Física Experimental. Por um lado, a Física Teórica busca explicar um fenômeno com conceitos matemáticos e a elaboração de teorias abstratas. Por outro lado, a Física Experimental é a categoria de disciplinas e sub-disciplinas no campo da Física que se preocupa com a observação de fenômenos físicos e experimentos. Esta última reagrupa todas as disciplinas da Física que estão preocupadas com a aquisição de dados, métodos de aquisição de dados, a conceitualização detalhada (além de simples experimentos mentais) e a realização de experimentos de laboratório.

De acordo com (GRASSELLI; GARDELLI, 2014), a utilização de experimentos torna-se essencial para o ensino, pois fomenta a racionalização da teoria vista pelos alunos ao aplicá-la para analisar e resolver problemas relacionados aos experimentos. Além disso, uma quantidade enorme de tecnologia moderna é baseada em pesquisas fundamentais da física. Ou seja, o avanço de várias outras ciências, incluindo a área médica e a engenharia, depende do desenvolvimento de novas técnicas e instrumentação de medição.

A importância do experimento no ensino já vem se consolidando há vários anos (SOUZA FILHO *et al.*, 2005). Conforme observado na pesquisa de Azevedo (2009), é possível constatar que de nove periódicos publicados e analisados no estudo, de 1978 a 2009, foram pulicados o total de 274 artigos. Conforme FIGURA 1, dos trabalhos apresentados, 60 deles direcionam-se à experimentos quantitativos e 54 à experimentos quantitativos com aparatos sofisticados. Nesse sentido, o experimento quantitativo sofisticado utiliza aparatos semelhantes ao que são utilizados nos laboratórios universitários (AZEVEDO *et al.*, 2009). O autor ainda questiona a falta de experimentos no ensino de física que buscam propor atividades problematizadoras para os alunos e de reconstruções histórias de experimentos.

FIGURA 1 – TABELA DE AZEVEDO ET AL., 2009

Periódicos	Categorias de análise						
	DS	Q	QS	P	RH	NE	TOTAL
Revista Brasileira de Ensino de Física (1979)	38	40	54	1	2		135
Caderno Brasileiro de Ensino de Física (1984)	86	15	-	4	-	-	105
Investigações em ensino de Ciências (1996)	-	-	-	-	-	1	1
Ciência & Ensino (1996)	1	-	-	-	-	-	1
Ciência & Educação (1998)	-	-	-	-	2	-	2
Ensaio (1999)	-	-	-	1	-	-	1
A Física na Escola (2000)	22	5	-	-	-	2	29
Alexandria (2008)	-	-	-	-	-	-	0
Revista Brasileira de Pesquisa em Educação em Ciências (2001)	-	-	-	-	-	-	0
Total de artigos por categoria	**147**	**60**	**54**	**6**	**4**	**3**	**274**

Legendas: Experimentos demonstrativos com aparatos de montagem simples (DS); Experimentos quantitativos com aparatos de montagem simples (Q); Experimentos quantitativos com aparatos sofisticados (QS); Experimentos problematizadores (P); Experimentos a partir de reconstruções de aparatos históricos (RH); Experimentos para portadores de necessidades especiais (NE)

FONTE: Azevedo *et al.*, (2009)

Legenda: Tabela apresentando os trabalhos publicados nas revistas separando-as em diferentes categorias.

Após realizar uma pequisa cursória em algumas revistas (Revista Brasileira de Ensino de Física, Revista Brasileira de Ensino de Ciência e Tecnologia, Revista Brasileira de Pesquisa em Educação em Ciências e o Caderno Brasileiro de Ensino de Física), verificou-se 34 publicações que utilizam a plataforma Arduino, como pode-se observar na Tabela 1. Destes trabalhos, a maioria busca desenvolver experimentos pontuais, *exempli gratia*, a determinação da constante de Planck (OLIVEIRA *et al.*, 2005), o movimento retilíneo, a queda livre (FETZNER FILHO, 2015), a força peso, a Lei de Hooke, a Foto-resistividade (MOURÃO, 2018), o gráfico da posição em tempo real (DWORAKOWSKI *et al.*, 2014), entre outros. Apesar de ter um grande desenvolvimento de experimentos utilizando microcontroladores e outros dispositivos eletrônicos, não foi possível encontrar nesta pesquisa cursória uma proposta de instrumentação que automatize / controle de experimentos e que, ao mesmo tempo, aproxime os equipamentos de laboratório e indústria, como o presente trabalho propõe.

TABELA 1 – RESULTADO DE PESQUISA CURSÓRIA

Revista Brasileira de Ensino de Física	Revista Brasileira de Ensino de Ciência e Tecnologia	Revista Brasileira de Pesquisa em Educação em Ciências	Caderno Brasileiro de Ensino de Física
22	1	1	10

FONTE: Autor (2019)

LEGENDA: Resultado de pesquisa cursória em quatro jornais de ensino de física que utilizam a plataforma Arduino

Nesse sentido, os trabalhos científicos que buscam envolver experimentos quantitativos sofisticados, ou seja, que se utilizam de aparatos experimentais aprimorados, *exempli Gratia*, os experimentos utilizados em laboratórios que fazem uso de computadores e de interfaces digitalizadoras, como placas de som AD/DA, portas seriais e USB, ainda são pouco desenvolvidos no Brasil. Um dos motivos para isso pode ser a falta de acessibilidade dos recursos disponíveis, o que acaba influenciado aspectos relacionados, por exemplo, à precisão dos dados recolhidos e, consequentemente, afastando o experimento construído de sua teoria física.

Para saber tendencias fora do Brasil, foi feito outra busca superficial na Web [1] através da busca por artigos que envolvem física e Arduino foram encontradas quatro revistas que possuíam artigos envolvendo física e microcontroladores ou Arduino essas revistas são: Springer(22 artigos no periodo de 2013 ao presente), AAPT Physics Education(26 artigos no periodo de 2013 ao presente), ieeexplore (105 artigos no periodo de 2010 ao presente) e a iopscience(61 artigos no periodo de 2012 ao presente). No total houve 214 artigos encontrados publicados no exterior uma diferença acentuada comparado ao 34 encontrado no Brasil e os 274 encontrado por Azevedo *et al.* (2009).

Para obter a visualização de dados de forma mais confortável, a Parallax para a Microsoft Office™desenvolveu um recurso comercial chamado PLX-DAQ, vide (DWORAKOWSKI *et al.*, 2014), que faz uma interface entre o Arduino e o Excel™. Há vários outros programas, comerciais e grátis, que fazem, além da coleta, a interface do Arduíno como o LabVIEW (NATIONAL INSTRUMENTS, 2019) e Matlab (RMAGTIBAY, 2013). Essas duas abordagens conseguem produzir GUIs, (*Graphical user interface*, interface gráfica para usuários) para experimentos, sendo eles *stand alone*, ou parte de um sistema de controle mais elaborado. Entretanto, ambos os programas requerem a aquisição de produtos comerciais. As alternativas abertas, *open source*, são o Guino (MADSHOBYE, 2014) e o DueGUI (COWASAKI, 2014), mas ambas estão limitadas pela própria estrutura do Arduíno, visto que a customização da interface gráfica é feita dentro do programa do IDE do Arduino, o que não permite o controle de outros dispositivos (diferente do Arduin) conectados ao computador (KOENKA; SÁIZ; C.HAUSER, 2014).

2.1 RECURSOS EDUCACIONAIS ABERTOS

Os Recursos Educacionais Abertos (OER, *Open Educational Resources*) são materiais de ensino e pesquisa de qualquer meio que esteja em domínio público ou tenha sido liberado sob uma licença aberta que permita acesso, uso, adaptação e redistribuição gratuitos. Desde 2002, os OER têm sido cada vez mais reconhecidos pela comunidade internacional como uma ferramenta inovadora que pode oferecer oportunidades de aprendizagem para alunos de diversos níveis de educação. Além disso, os OER são consistentes com o compromisso constitucional da UNESCO, que defende a “livre troca de ideias e conhecimento” (UNITED NATIONS EDUCATIONAL SCIENTIFIC AND CULTURAL ORGANIZATION, 2019). Ainda conforme a organização, os materiais de aprendizagem disponíveis gratuitamente para adaptação e

[1] *https://scholar.google.com/*

reutilização podem ampliar o acesso à aprendizagem e melhorar a qualidade com um custo menor:

> Emphasizing that the term Open Educational Resources (OER) was coined at UNESCO's 2002 Forum on Open Courseware and designates "teaching, learning and research materials in any medium, digital or otherwise, that reside in the public domain or have been released under an open license that permits no-cost access, use, adaptation and redistribution by others with no or limited restrictions. Open licensing is built within the existing framework of intellectual property rights as defined by relevant international conventions and respects the authorship of the work" (UNITED NATIONS EDUCATIONAL SCIENTIFIC AND CULTURAL ORGANIZATION, 2012)

.

O 1º Congresso Mundial sobre OER, em Paris, discutiu iniciativas de OER em andamento e buscou encorajar um compromisso com o princípio de que os produtos do trabalho financiado com fundos públicos em apoio à educação deveriam ter licenças abertas, além de apoiar esforços de capacitação, colaboração e pesquisa relacionados à OER (UNITED NATIONS EDUCATIONAL SCIENTIFIC AND CULTURAL ORGANIZATION, 2012). As licenças abertas para recursos tecnológicos podem ser divididas em dois grupos: *Hardware* e *Software*. São licenças que estão de acordo com a definição de Código aberto, portanto, permitem o uso, modificação e compartilhamento livre de *Hardware* e *Software* (OPENSOURCE.ORG, 2019).

Software livre refere-se à programas onde o código fonte está disponível para o público, para ser copiado, modificado e compartilhado. O código fonte pode ser visto como uma lista de instruções, em uma linguagem de programação, que uma pessoa pode ler e modificar. O *Software* livre contrasta com *Software* fechado, que não disponibiliza código fonte. Portanto, somente a pessoa ou a organização detentora do código consegue modificá-lo (RED HAT, 2019b). Este trabalho utilizará o Python, linguagem de fonte aberta [2], com licença livre aprovada pelo Open Source Initiative [3]. Ademais, adotará a licença livre da General Public License (GNU), v3.0.

Hardware livre se refere à uma prática legal e não é associado a nenhum objeto específico. Assim como no scope do o *software* o desenvolvedor disponibiliza seu código fonte, o desenvolvedor *Hardware* livre disponibiliza esquemáticos, desenhos, listas de componentes e documentação, além de ser feito, preferencialmente, em programas abertos, encorajando assim outros a reproduzirem, modificarem, aprenderem e melhorarem o desenvolvedor de *Hardware* livre. Em oposição, o desenvolvedor de *Hardware* fechado tem seu funcionamento ocluso, o que dificulta a possibilidade de modificação e aprendizado dele. Ademais, o *Hardware* é protegido de reprodução e redistribuição por patentes, salvo o caso em o detentor da patente permite o acesso. (RED HAT, 2019a).

Em relação ao projeto criado e apresentado nesse artigo, junto ao *software* desenvolvido, o *Hardware* terá uma licença livre, Cern ohl v1.2. Desta forma, o projeto como um todo

[2] disponível em https://github.com/python/cpython
[3] disponível em https://opensource.org/licenses/Python-2.0

poderá ser replicado e modificado por alunos, instituições e o público geral poderá estudá-lo, modificá-lo e compartilhá-lo de acordo com as licenças de uso. Além disso, o *Arduino* servirá como plataforma para o projeto, tendo em vista que ele é popularmente conhecido entre usuários e possui preço acessível.

2.2 ARDUINO

O *Arduino* é um projeto eletrônico de plataforma aberta (*open-source*), com a filosofia de fornecer um *Hardware* e um *software* fáceis de usar. Tem o objetivo de fornecer uma plataforma em que qualquer indivíduo possa produzir um projeto interativo. O *Hardware* é composto por uma placa eletrônica - placa *Arduino* - que contém um microcontrolador, responsável por realizar pequenas tarefas que são definidas pelo código criado no "software" *Arduino*, o qual os comandos são escritos sequencialmente (um comando ou comentário por linha). Sensores contactados à placa *Arduino* identificam o meio que o usuário quer trabalhar (luz, temperatura, umidade, etc.) e responde ao meio através de atuadores (motores, relés, lâmpadas, etc.). Esta plataforma vem sendo desenvolvida desde 2005, e, a partir de 2010, tem sido proposta como ferramenta de ensino nas áreas de exatas por vários pesquisadores, entre eles: (CAVALCANTE; TAVOLARO; MOLISANI, 2011), (SOUZA *et al.*, 2011), (H. CORDOVA, 2016), (AMORIM; DIAS; SOARES, 2015), (DIONISIO; SPALDING, 2017), (SILVEIRA; GIRARDI, 2017), (SILVA; SILVA; LIMA, 2014), (DWORAKOWSKI *et al.*, 2014). Alguns trabalhos também foram desenvolvidos em mestrados profissionais, como dos autores (H. CORDOVA, 2016), (SILVEIRA; GIRARDI, 2017) e (DWORAKOWSKI *et al.*, 2014). Contudo, outras áreas do conhecimento podem se beneficiar desta plataforma, como exemplo, áreas relacionadas à moda, que usam a variante *Lilypad* do *Arduino* (BUECHLEY; EISENBERG, 2008).

3 METODOLOGIA

Para o desenvolvimento do presente projeto, foram escritas duas bibliotecas: a primeira biblioteca *Arduino* foi escrita [1] para os circuitos integrados (doravante CI) AD7794, ADC (*Analog to Digital Converter*), e o AD5662, DAC (*Digital to Analog Converter*), que permite a medição de um sinal analógico através de uma eletrônica complementar; a segunda biblioteca permite o uso do sistema digital, *id est*, um sistema que varia somente entre tensão miníma ou a máxima ao controlar um sinal analógico, *id est*, um valor entre a tenção miníma até a tensão máxima. Para utilizar os CIs mencionados acima, foi projetado um circuito no Kicad [2] para a fabricação de uma placa de circuito impresso (doravante PCB). Tal placa facilita a montagem e utilização do sistema eletrônico, com a vantagem adicional de aumentar sua replicabilidade. O sistema tem as seguintes características:

- Entradas digitais, que permitem detectar uma variação de nível de potencial, que pode ser produzido por um sensor, por exemplo.
- Saída digital, que permite acionar um atuador, do tipo relé.
- Entrada analógica, que permite “ler” um sinal analógico, por exemplo, proveniente de um sensor.
- Saída analógica, que permite controlar a tensão de uma fonte de alimentação.
- Sistema desenvolvido de fácil reprodução, entendimento e manuseio.

O conjunto de características acima permite a elaboração de diversos experimentos controlados via computador, ou com coleta de dados via computador, e com a adição de um mínimo de *hardware* externo, possibilita expandir as possibilidades de desenvolvimento de experimentos. Para atender o último item das características, foi escolhido utilizar a plataforma *Arduino* para o controle local dos dispositivos de aquisição de dados e da fonte de alimentação, além da linguagem de programação Python para controle do sistema e programação do experimento.

3.1 INTERFACE PYTHON

A linguagem Python foi utilizada para compor o *Front-end*, a parte responsável por interagir com o usuário para a comunicação com o Arduíno, que acontece através de um protocolo de comunicação em que as funções que o programa pode executar estão em uma tabela de comandos. O *Arduino* recebe um *string* pela porta serial e executa essas funções

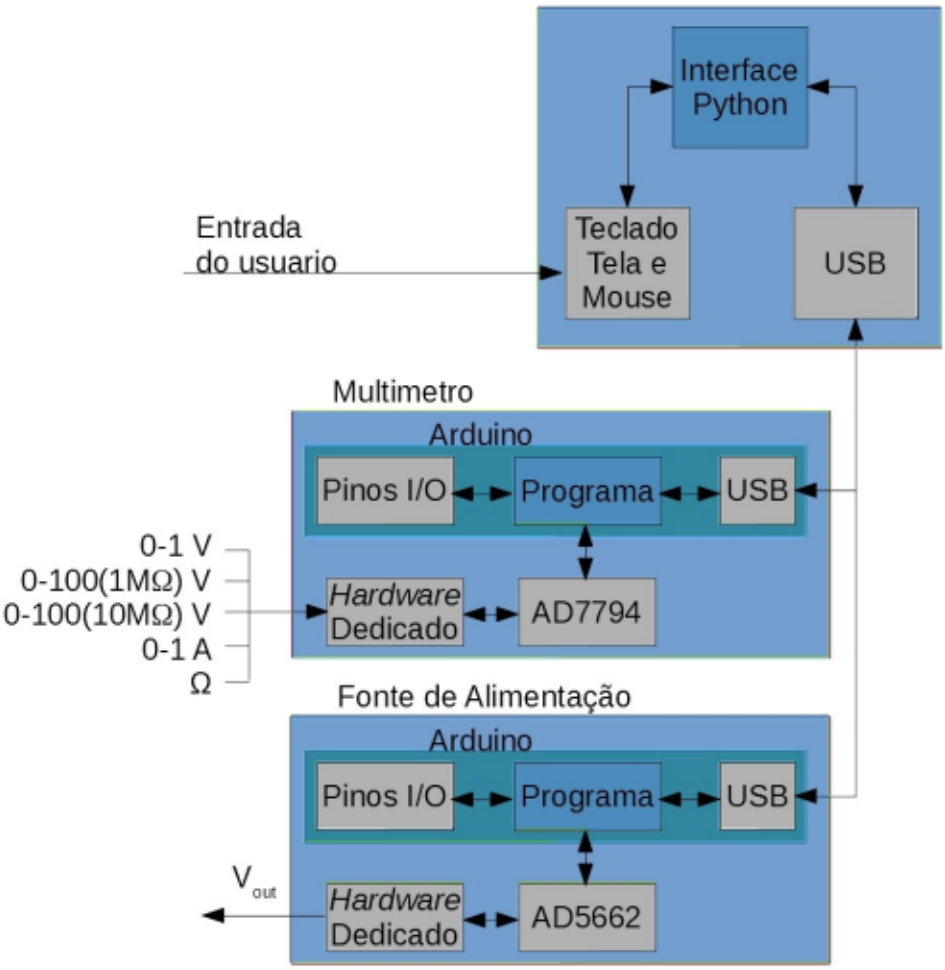
[1] Classe da linguagem c++ predefinida para interfacear/comunicar com um circuito integrado, neste caso.

[2] O Kicad é um conjunto de *software* gratuito para automação de *design* eletrônico (EDA). Facilita o desenho de esquemas para circuitos eletrônicos e sua conversão para projetos de PCB.

predefinidas. Dessas funções, três são para todos os tipos de *hardware*, ComTest, ID e ICID, para que o usuário, por meio do computador, consiga identificar as placas.

Todo o sistema pode ser automatizado através de um computador que roda o programa Python. A FIGURA 2 mostra um diagrama geral da instrumentação proposta. Nele, há dois módulos: um chamado de multímetro, e outro chamado de Módulo Fonte.

FIGURA 2 – ESQUEMA GERAL DO SISTEMA.

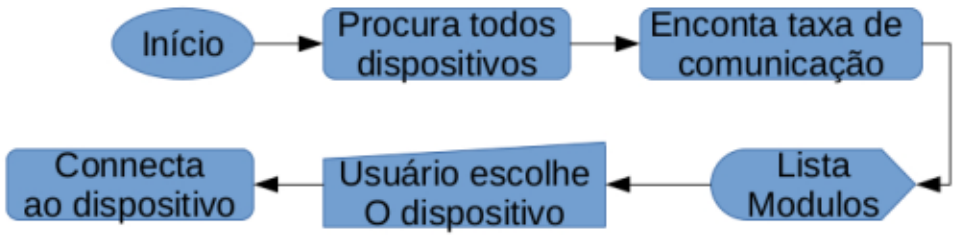


FONTE: Autor (2019)

Para implementar a comunicação entre o computador e o *Arduino* pela porta serial, o *Python* utiliza o módulo *Pyserial*, uma extensão de porta serial para *Win32*, *OSX*, *Linux*, *BSD*, *Jython* e *IronPython*. Para estabelecer comunicação com o dispositivo feito neste trabalho, foi criado um *script*, descrito pela FIGURA 3, que procura todos os os dispositivos na porta serial, acha a taxa de comunicação e lista os dispositivos, que estão com *firmware* criado neste trabalho. Para encontrar a taxa de comunicação, o *script* se conecta a todos os dispositivos na porta serial realiza uma varredura com as velocidade de comunicação padrões, e através de

um comando para que cada dispositivo responda com um *String* predefinido. O *script* listará todas as respostas que forem iguais a resposta padrão e, então, o usuário poderá escolher o dispositivo a ser utilizado. Este *script*, localizado no Apêndice A seção 1, é utilizado para mandar e receber dados para os dispositivos na porta serial.

FIGURA 3 – ESQUEMA GERAL DO SISTEMA.



FONTE: Autor (2019)

Para o *script* conectar-se com o *Arduino*, foi utilizado o SerialConnection presente em Apêndice A seção 1. Sua função é procurar todos dispositivos conectados na porta serial e listar os dispositivos para o usuário escolher. Caso haja apenas um dispositivo, o programa automaticamente o seleciona. Após a seleção, o programa encontra a taxa de comunicação com o *Arduino* mandando um comando de teste de comunicação e comparando a resposta com um *String* conhecido.

Deste modo, o *Arduino* terá que ser programado de acordo com a função do *Hardware* em que está inserido. Os comandos listados na TABELA 2 foram escritos em c++ e programados no *Arduino* para controlar qualquer módulo através desse padrão de comunicação. Ao receber um comando, o *Arduino* observa os primeiros quatro caracteres e procura o valor recebido na sua tabela de comandos. Caso o comando não seja encontrado, o dispositivo responde como "Comando não conhecido". Caso haja argumentos para uma função, eles se tornarão números entre 0 e 99, sempre em um espaço de dois caracteres.

TABELA 2 – LISTA DE COMANDOS UNIVERSAIS

Funções	Descrição	código em HEX
ComTest	Retorna um *String* de referencia	0x0
ID	Retorna a identificação da placa	0x1

FONTE: Autor (2019)

LEGENDA: Comandos presentes em todos os módulos.

Para o Python comunicar-se com o módulo do DAQ(Data acquisition)/Multimetro, os comandos listados na TABELA 3 foram implementados no *Arduino*. Para utilizar esses coman-

dos, basta mandar um *string* via porta serial, sendo que cada carácter e numero corresponde a um *byte*, como mostra o seguinte exemplo: "0006010106"(comando), onde 0006 é a função chamada (READCHANNEL), 01 é referente ao canal, que neste caso é o canal 2 (2^1), e 06 é o numero de pontos para integrar.

TABELA 3 – LISTA DE COMANDOS DO DAQ/MULTÍMETRO

Funções	Descrição	codigo	Argumentos
COMTEST	Retorna um *String* de referencia	0x0	Sem
ID	Retorna a identificação da placa	0x1	Sem
ICID	Lê a identificação do CI AD7794	0x2	Sem
UltimaMsg	Manda novamente o ultimo dado	0x3	Sem
TEMP	Retorna a temperatura medida pelo AD7794	0x4	Sem
SETGAIN	Ajusta o ganho para um canal do AD7794	0x5	Ganho(0-7), Canal, pontos para integrar
READCHANNEL	Lê um canal do AD7794	0x6	numero do pino
TOGGLE	Muda o estado do pino para alto ou baixo	0x7	numero do pino
CTRLIO	Ajusta o pino para estado alto ou baixo	0x8	numero do pino e estado desejado
READIO	Retornao estado do pino	0x9	numero do pino

FONTE: Autor (2019)

LEGENDA: Comandos presentes em do DAQ/Multimetro

Para o controlador do DAC, a TABELA 4 foi implementada no *Arduino*, seguindo a mesma lógica explicitada anteriormente.

TABELA 4 – LISTA DE COMANDOS DO DAC/FONTE

Funções	Descrição	codigo	Argumentos
COMTEST	Retorna um *String* de referencia	0x0	Sem
ID	Retorna a in dentificação da placa	0x1	Sem
OFF	Conecta a saída do DAC a um *pull-down* de $100k\Omega$	0x2	Sem
ON	Liga a saída do DAC de acordo com o programado	0x3	Sem
SETOUTPUT	programa a tensão de saída	0x4	Sem
OUTPUT	Programa a tensão, sem o *output*	0x5	Sem
TOGGLE	Retorna um *String* de referencia	0x7	numero do pino
CTRLIO	Retorna um *String* de referencia	0x8	numero do pino e estado desejado
READIO	Retorna um *String* de referencia	0x9	numero do pino

FONTE: Autor (2019)

LEGENDA: Comandos presentes em do DAC/Fonte

Com esta configuração, é possível fazer o levantamento de curva característica de diodo, onde varre-se a tensão do Módulo Fonte (desde valores negativos até valores positivos) e, um circuito série, (diodo + resistor), pode-se conectar (do módulo Multímetro) o amperímetro (para registro da corrente) e o voltímetro (no modo bipolar ou diferencial) para registrar a queda de tensão sobre o diodo. Atravez do *Python* pode-se programar uma varredura (incrementa tensão no DAC, lê corrente e tensão no DAQ/Multimetro e assim por diante) e obter automaticamente a curva IxA de um diodo. Alternativamente ambos os módulos (Fonte e Multímetro) podem funcionar no modo manual ou autônomo (*stand alone*). Portanto, está prevista a montagem destes módulos em gabinetes (caixas) independentes e a conexão ao computador será simplesmente pelo cabo USB.

Para controlar os dispositivos, há duas abordagens possíveis: sequencial ou paralelo. Para utilizar a interface no modo sequencial, basta importar o módulo de comunicação serial e o módulo contendo os comandos da placa a ser controlada dentro de um *script*, mandando os comandos para lê-los. No caso paralelo, importa-se no *script* o módulo *Glue* e, através dele, o

usuário deve conectar os dispositivos para comunicar. Nesta abordagem, o computador está sempre olhando para a porta serial para ver ser houve um dado na porta serial do Arduino, que é transparente ao usuário que executa seu *script* sem interferência. Os dados recebidos em um FIFO, *First In First Out*, para o objeto *python* do dispositivo, que pode ser o DAQ/Multimetro, fonte ou outro dispositivo semelhante . Para obter o dado, o usuário utiliza um wrapper, contido no *Glue*, para retirar o dado.

Para fazer um *datalog* dos dados, foi escrito um módulo com o nome *datalog*. Este módulo pode funcionar em dois modos: como editor de Excel ou CVS[3]. Como o editor de Excel do Python utiliza pacotes que não são compatíveis com a versão do Python anterior a 3.6, foi acrescentado ao datalog o CVS. As diferenças de funcionalidade entre Excel e CVS estão explicitadas a seguir:

- O Excel pode criar planilhas, enquanto o CVS precisa criar arquivos novos;
- O Excel automaticamente cria um arquivo caso ele não exista, enquanto, no CVS, o usuário precisa criar um arquivo dentro do Python.

Para a visualização dos dados, a interface terá capacidade de gerar gráficos em tempo real. Para isso, será utilizado PyQtGraph, um pacote gráfico cujo foco é gerar gráficos interativos.

3.2 LEITURA DE SINAL ANALÓGICO

O módulo Multímetro é composto por uma conversor A/D (ADC) da *Analog Devices* [4]. Em cada entrada diferencial foi inserido um filtro RC sugerido pelo fabricante (ANALOG DEVICES, INC, 2010, p. 6). Está previsto dois modos de operação: um automático via computador, em que é possível realizar coletas de dados e controle sincronizado com outros instrumentos (fonte de alimentação, por exemplo); outro modo de operação autônomo, que se assemelha muito a um multímetro convencional. Prevemos uma proteção de entrada utilizando dois reguladores TL431 contrapostos entre as entradas diferenciais. Cada entrada tem dois diodos de proteção, um conectado ao AVDD (fonte positiva analógica) e outro ao AVEE (terra analógica) . A corrente máxima que estes diodos podem suportar é de 10 mA (Analog, 2010, p.3).

O modulo utiliza um ADC da *Analog Devices* (AD7794), que possui integrado no chip uma tensão de referência e um amplificador de instrumentação (PGA) de entrada com ganho ajustável (de 1 a 128, ajustados em passos de 2^N). Esse amplificador permite, com um conjunto reduzido de componentes eletrônicos, configurar três entradas independentes de tensão do multímetro: uma sem atenuador de entrada de zero a 1,17 V (com ganho unitário) e duas com atenuadores x100, sendo que uma delas possui impedância de entrada total de $1,00 M\Omega$ (esta entrada permite conectar uma ponta de prova de osciloscópio atenuada x10 de $10 M\Omega$) e a

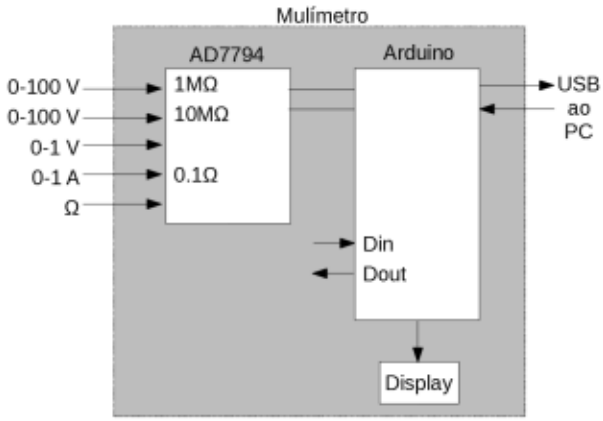
[3] Um arquivo de valores separados por vírgula (CSV) é um arquivo de texto delimitado que usa uma vírgula para separar valores.

[4] A Analog Devices, Inc., é uma empresa multinacional americana de semicondutores especializada em tecnologia de conversão de dados, processamento de sinais e gerenciamento de energia, com sede em Norwood, Massachusetts. Para mais informações, visite https://www.analog.com/.

outra de $10,0M\Omega$ de entrada, semelhante a entrada convencional de um multímetro portátil. Além disso, o multímetro tem uma entrada para medir resistência e outra entrada para medição de corrente.

Todas as entradas são diferenciais e não está prevista a possibilidade de medidas de tensão ou corrente alternada, pois julgou-se que raramente são realizadas medidas de tensão ou corrente alternadas em experimentos de laboratório didático, ou mesmo em laboratórios de pesquisa. É possível utilizar todas as cinco entradas para medidas sequenciais, *id est.*, não é necessário reconfigurar as conexões de entrada cada vez que se seleciona uma função diferente. Assim, o multímetro permite realizar de forma automatizada uma curva de tensão contra corrente, por exemplo. A FIGURA 4 mostra um diagrama geral da instrumentação proposta. Nela, há dois componentes: um Arduino e um conversor ADC da (AD7794). Este último disponibiliza seis entrada diferenciais. Optou-se por utilizar entradas diferenciais, bipolares, ao invés de entradas simples, unipolares, referenciadas ao terra. Desta forma, é possível realizar medidas não referenciadas ao terra, assegurando-se que nenhuma das entradas (Ain+ ou Ain-) ultrapasse o limite da fonte, i.e., entre 0 V e 4 V. Também foi implementado um isolador de sinais (Si8441 da Silicon Labs) e uma fonte chaviada isoladora, para que o terra do multímetro fique isolado do terra do sistema (Arduino ou Arduino + computador, via cabo USB).

FIGURA 4 – CIRCUITO SIMPLIFICADO DO MULTÍMETRO.



FONTE: Autor (2019)

Legenda: Esquema simplificado da circuito de potencia alta da fonte de alimentação

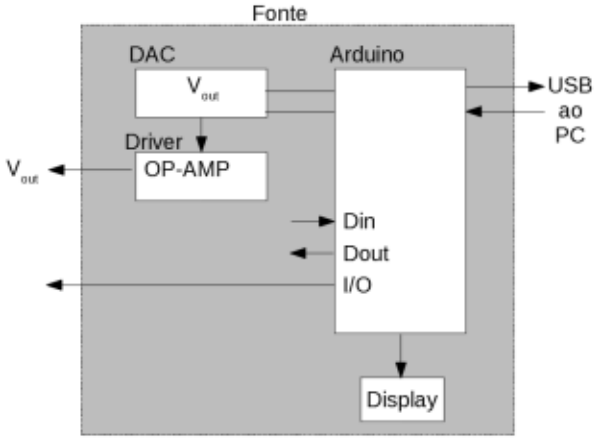
3.3 FONTE

O módulo fonte é basicamente o conversor digital analógico (DAC). Está pevisto, possui um estágio de saída *driver* composto por um amplificador operacional de potência (OPA 548 ou OPA 549), que também permite o controle de limite de corrente. Para o DAC, foram testados o CI AD5662, de 16 bits de resolução e com necessidade de tensão de referência externa, e o

DAC 8560, de 16 bits de resolução e com tensão de referência interna. Para o estágio de saída é previsto o OPA548, que é um amplificador operacional de potência, possibilitando uma saída simétrica de +/- 20 V @ 3 A. É possível operar o DAC como um gerador de áudio, em que a frequência máxima de geração está limitada pela velocidade ao qual o *Arduino* pode enviar dados da tensão ao DAC. O OPA548 tem limite de frequência de 1 MHz.

Como os DACs são de baixa potência no circuito final, o dispositivo será conectado a um amplificador operacional (OPAMP) de potência conforme a FIGURA 5, fará parte da cadeia da fonte de alimentação.

FIGURA 5 – CIRCUITO SIMPLIFICADO DA FONTE



FONTE: Autor (2019)

Legenda: esquemático simplificado do modulo da fonte

4 RESULTADOS E ANÁLISE DOS RESULTADOS

4.1 MULTÍMETRO

4.1.1 Hardware

Para que o multímetro tenha as especificações mencionadas no capítulo anterior e proteção de entrada adequada, foram testadas as combinações dos CIs LM431, TL431 e AP431 (sendo que o projeto é o mesmo e que a única diferença entre os CIs é de fabricantes). Como visto na TABELA 5, estes CIs começam a conduzir a partir de 1 V e, a 1.19 V, com uma corrente entre 78 e 68 nA. Visto que esses CIs estão na entrada do conversor analógico digital, a influência nas entradas foram testadas. A influencia provou-se indesejável (ler seção 4.1.1.3), já que, a partir de uma diferencia de potencial de 6V entre as entradas diferenciais, o CIs de proteção começavam a conduzir e, portanto, foi decidido excluí-los do projeto.

TABELA 5 – COMPARAÇÃO DE CIS DE PROTEÇÃO DE ENTRADA ANALÓGICA

Tensão (V)	TL431 e LM431	AP431 e AP431
0.99,4	0 nA	0 nA
1,000	7 nA	7 nA
1,100	23 nA	24 nA
1,199	78 nA	68 nA

FONTE: Autor (2019)

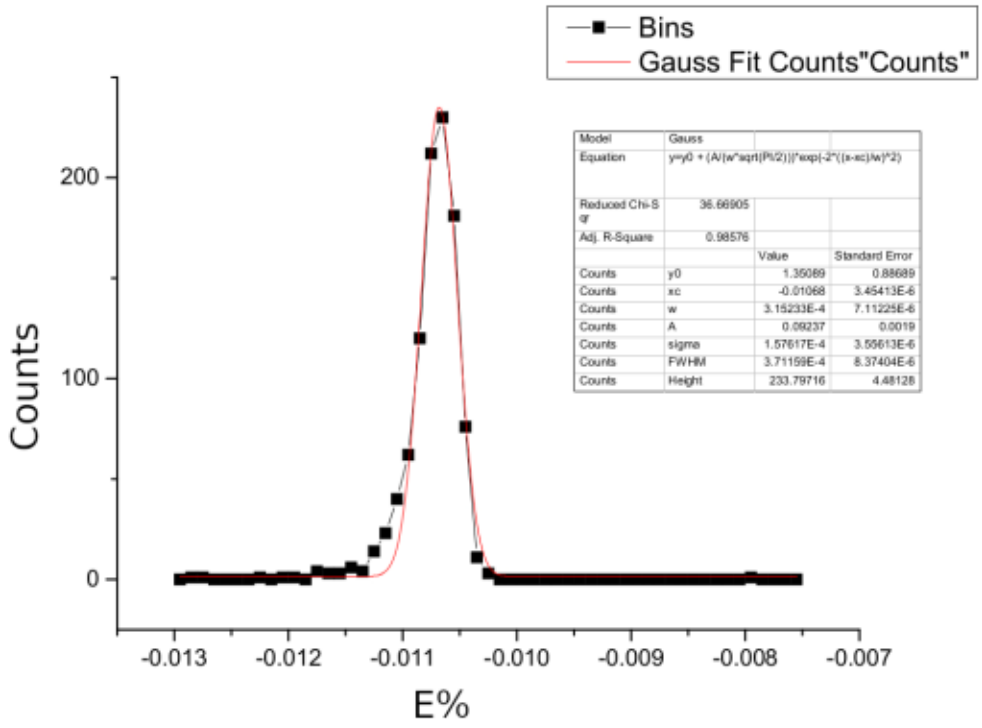
LEGENDA: Configurações dos testes dos CIs LM431, TL431 e AP431.

Além do circuito integrado responsável pela proteção de entrada, foi comparada a tensão lida pelo AD7794 com a lida por um multímetro de sete dígitos e meio de precisão (HP3457A). A FIGURA 6 foi obtida a partir da medição da entrada da placa do canal 1 para uma tensão fixa de $1V$. Foram coletados 1000 pontos, e, em seguida, gerou-se um histograma ajustado a uma curva Gaussiana para encontrar o erro percentual. Neste caso, o erro foi de $-0,0106\%$ em relação à referência (HP3457A). Portanto, podemos afirmar que o AD7794 está de acordo com sua especificação técnica.

O AD7794 é extremamente sensível a carga em suas entradas, principalmente a ruído de alta frequência. Inicialmente, foi planejado coletar dados aplicando uma tensão diretamente nas entradas. Entretanto, devido a sensibilidade aos ruídos de alta frequência, foi montado um filtro de polo simples para minimizar esse problema. Uma nova placa será projetada com as implementações que se fizeram necessários. Houve uma tentativa de isolar o circuito do AD7794, porém, a isolação da comunicação digital feita com opto-isolador não conseguiu

responder a frequência de comunicação miníma do SPI do Arduino, de $120khz$. Portanto, para a isolação de sinais digitais e terra utilizou-se um CI dedicado (Si8441), que internamente utiliza acoplamento RF. Para averiguar a resposta do AD7794, as medidas da FIGURA 6 foram realizadas por meio de uma referência alimentada por bateria, baseado no CI AD588, um gerador de tensão de referência.

FIGURA 6 – ERRO PERCENTUAL DA LEITURA



FONTE: Autor (2019)

Legenda: Entrada do AD7794 comparada com a referencia.

Foi conectada uma fonte e um multímetro nas entradas da módulo, para adequar a tensão da fonte com a entrada do AD7794, foi inserido atenuador de 10 e 1000 (ganhos 64 e 128). Esse divisor foi utilizado devido ao baixo limite de variação de tensão da fonte. Além disso, foi varrida as tenções e os ganhos para cada canal. Assim, a TABELA 6 foi obtida, por meio da leitura das entradas do AD7794, comparando os valores com a referência e realizando a diferença entre ambos. Para cada ganho, a fonte foi programada para variar entre 0V até o fundo de escala do AD7794. Ademais, a polaridade da fonte foi variada para cada ganho e canal. Todos os dados estão disponíveis no Apêndice H. A TABELA 6 sumariza os erros dos cinco canais utilizados e os respectivos ganhos, varrendo todos os ganhos possíveis. Para obter cada valor de erro, fixou-se um ganho e uma entrada, por exemplo ganho 1 e entrada canal 0 (Ain1+ e Ain1-), realizou-se uma varredura de -1.14 a $+1.14$ em 46 passos, sendo o ponto zero medido duas vezes e cada passo foi medido dez vezes, portanto cada valor de erro

corresponde a um ajuste gaussiano nos 480 valores de erros obtidos na varredura. Um valor de erro (Delta) corresponde a diferença entre o HP3459A e o AD7794. "Canal 0"até "Canal 5"corresponde ao nome nos registros de programação.

TABELA 6 – ERRO DE MEDIDA DO AD7794 (DELTA)

Ganho	Canal 0 Ain 1	Canal 1 Ain 2	Canal 2 Ain 3	Canal 3 Ain 4	Canal 5 Ain 6
1	-3.1E-7	-5.9E-7	-6.1E-7	-1.2E-6	6.2E-7
2	2.6-7	-2.7E-7	-8.1E-7	-3.1E-7	1.0E-6
4	-9.1E-19	5.3E-5	-9.1E-19	2.6E-4	5.3-5
8	-6.2E-19	1.3E-5	-6.2E-19	1.3E-5	1.3E-5
16	6.7E-6	-1.3E-6	-6.7E-6	1.1E-5	1.1E-5
32	3.2E-6	4.4E-6	3.2E-6	3.6E-6	3.7E-6
64	1.5E-5	1.5E-5	5.4E-6	2.0E-5	1.5E-5
128	2.1E-6	3.9E-6	2.1E-6	6.2E-5	

FONTE: Autor (2019)

LEGENDA: Comandos presentes no DAC/Fonte

O esquemático do circuito foi realizado no programa *Kicad*, programa *open source software suite* de automação de design eletrônico. Foi decidido deixar um espaço dedicado para o Arduino Nano para deixar o módulo mais compacto e mais fácil de manusear. Os pinos I/O estão disponíveis nos *pin headers* (J1 e J5), em ambos os lados do Arduino, enquanto os pinos *SPI* ficaram dedicados à comunicação com o ADC.

Para visualizar os dados dos módulos, foram utilizados dois *displays*, um OLED ssd1306 de 128x64 (pixeis) e um LCD de 20x4 (caracteres por linhas). O Apêndice E.3 contem instruções em formato de manual para se usar o OLED. As linhas 143 à 173, No Apêndice E.1, são para selecionar, através de um *encoder* com botão, um canal e o habilitar para automaticamente fazer medidas. De acordo com as necessidades do projeto, foi criado um arquivo de *font* customizado, conforme Apêndice E, que permite destacar alguns números para indicar ao usuário qual canal está selecionado.

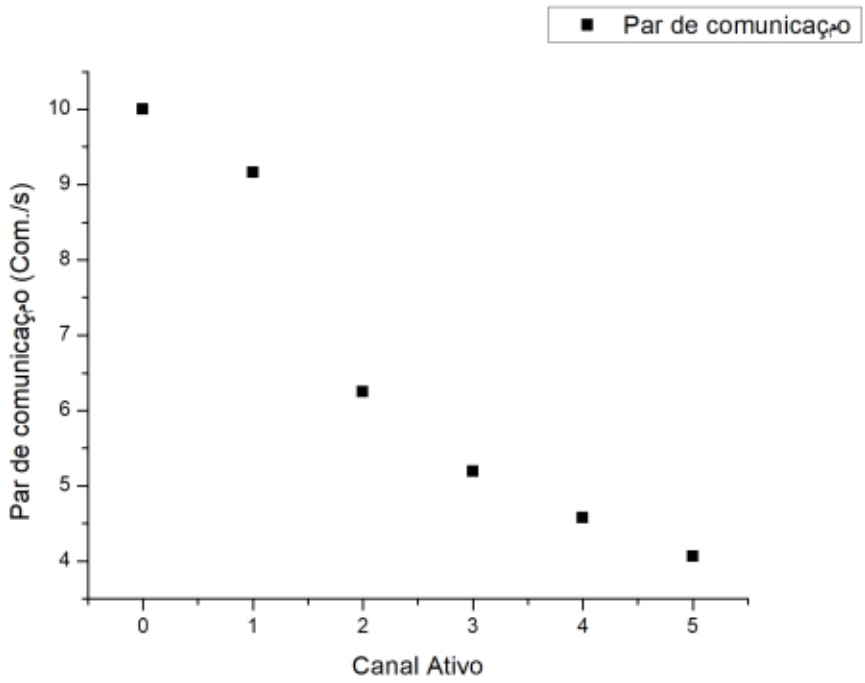
Devido ao espaço da memória do Arduino ocupada pelo OLED *display* e a necessidade de manter todos os pixeis dos caracteres na memória do Arduino, decidiu-se trocar o *display* pelo LCD. O código do *display* do LCD está presente na versão atual do módulo e, mesmo demandando menos memória, o número de canais ativos e mostrados nos *displays* afeta a velocidade de comunicação. Quando um canal está ativado, o módulo utiliza o AD7794 com frequência de leitura de 50 amostras por segundo para medi-lo. Para testar a velocidade de comunicação (tempo de latência) foi utilizado o *script* abaixo. Esse script conecta-se ao dispositivo, pede o *string ComTest* ao longo de 60 segundos e incrementa um valor para cada comando e resposta.

```
1    from SerialConnect import SerialConnection as SC
2    from  AD7794Codes import *
```

```
3       import time
4
5       DAQ = SC()
6       index = 0
7       dt = 0
8       start = time.time()
9       while dt <= 60:
10          DAQ.SendData(COMTEST)
11          a = DAQ.ReadData
12          index = index + 1
13          dt = time.time() - start
14      print(index, dt)
15      DAQ.Disconnect()
```

Depois de medir o números de comunicação total por minuto (Banda), cada ponto foi dividido pelo tempo levado para fazer o teste, conforme mostra a FIGURA 7.

FIGURA 7 – LATÊNCIA DA COMUNICAÇÃO



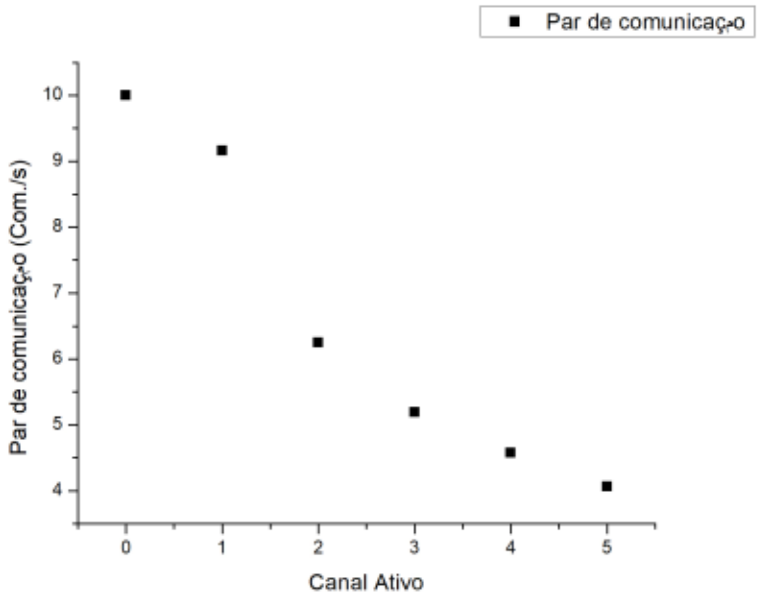
FONTE: Autor (2019)

Legenda: Par de comunicação por segundo com canais ativados e atualizando.

A FIGURA 8 foi gerada da mesma forma que a anterior, porém, o módulo retornava uma medida de um canal analógico. Desta forma, COMTEST foi substituido por READCHANNEL +

CHANNELS[0] + GAINS[0]. Nota-se que há uma queda na velocidade de comunicação a partir do segundo canal ativado.

FIGURA 8 – LATÊNCIA DA COMUNICAÇÃO COM MEDIÇÃO



FONTE: Autor (2019)

Legenda: Par de comunicação por segundo com canais ativados e atualizando.

As entradas do AD7794 possuem um filtro passa baixa do tipo *Single-Pole Filter* (SPF) para cada entrada e um filtro diferencial, isto é, um capacitor entre os pares diferenciais. Para encontrar a frequência de corte de $-3dB.$, a Equação 4,1 foi utilizada:

$$f_{cm}(Hz) = \frac{1}{2\pi R C_{CM}} \tag{4.1}$$

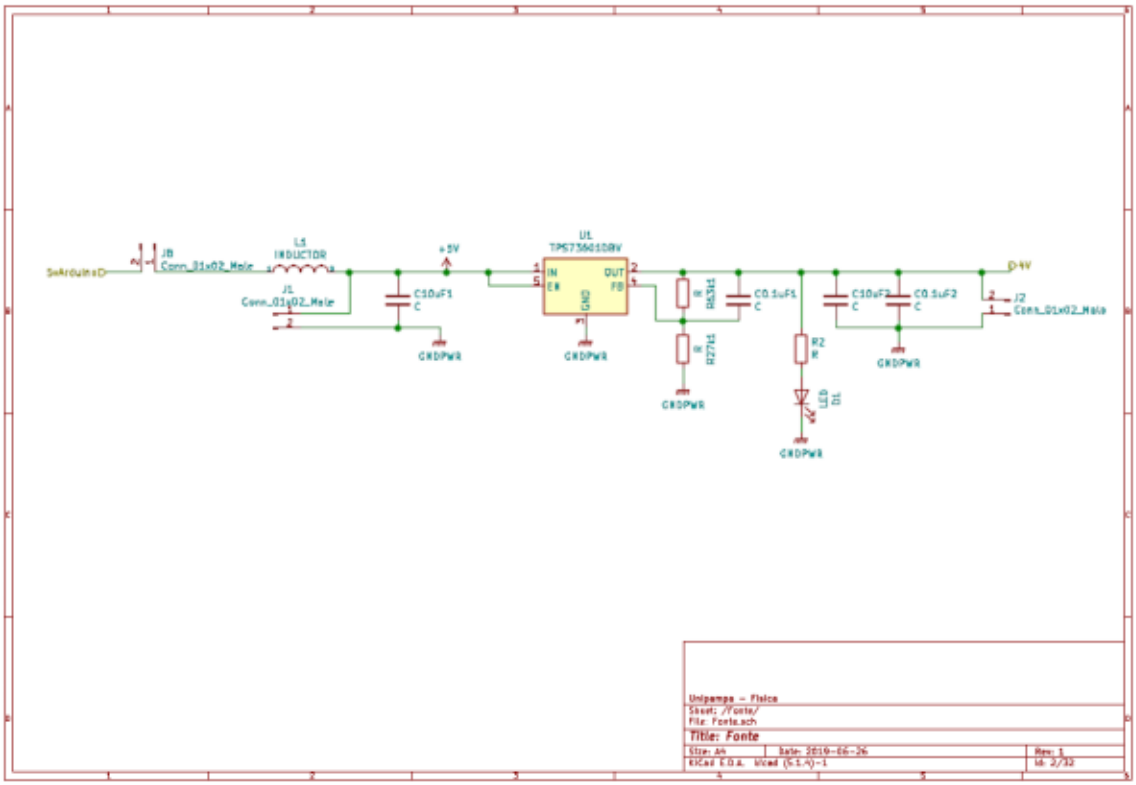
Como o ADC mede o diferencial entre duas entradas independentes, foi implementado um filtro de modo comum em conjunto com o SPF para melhorar o *common mode rejection*. Para encontrar a frequência de corte desse filtro, a a Equação 4,2 foi usada:

$$f_{diff}(Hz) = \frac{1}{2\pi R_{tot} C_{CM}} \tag{4.2}$$

4.1.1.1 Fonte do Multímetro/DAQ

A fonte Analógica do módulo pode ser alimentada pelos $5\,V$ do Arduino J8 da FIGURA 9, ou por uma fonte J1 externa de até $5.5\,V$. Para simplificar a montagem e diminuir o custo e o número de componentes e para alimentar a faixa de tensão permitida pela fonte do AD7794, o CI TPS73601 foi trocado por um diodo zener de $3.9V/500mW$.

FIGURA 9 – FONTE ANALÓGICA DO AD7794

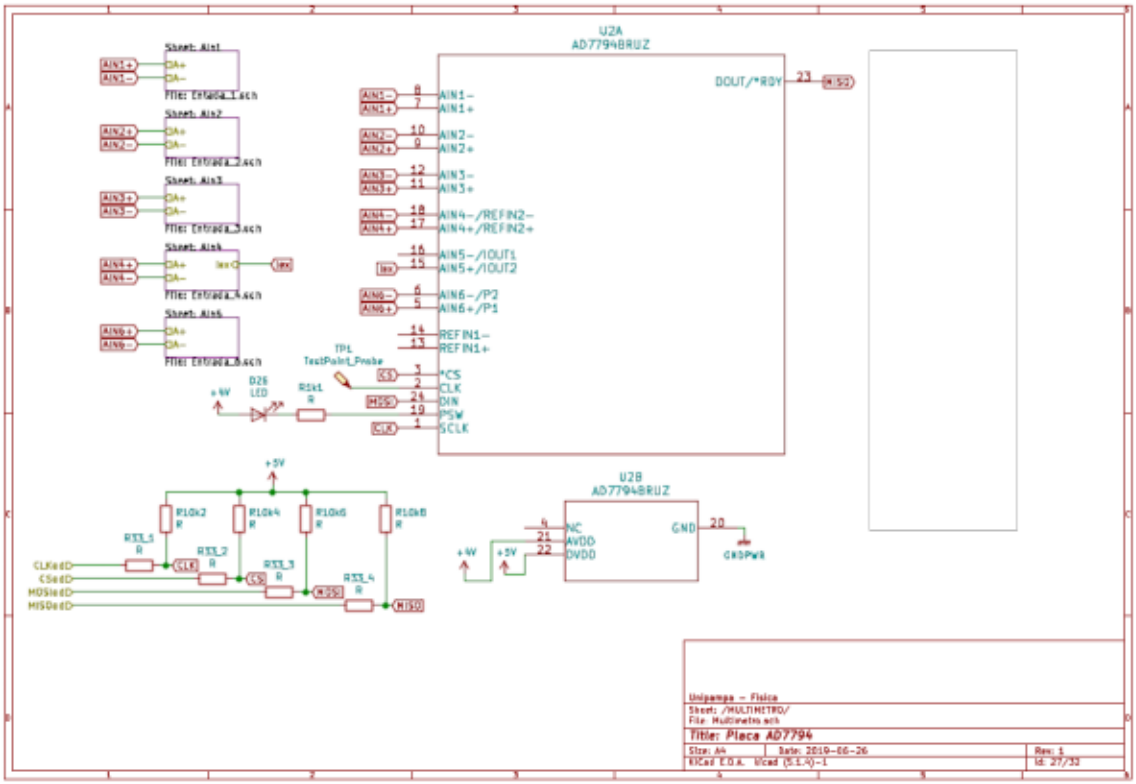


FONTE: Autor (2019)
LEGENDA: Fonte Analógica do AD7794.

4.1.1.2 Circuito AD7794

O ADC AD7794 é alimentado pelos 5 V do Arduino digital ou pela fonte Analógica descrita anteriormente. O ADC possui um LED para indicar erro e cinco entradas diferenciais. Para comunicação SPI, foram colocados *Pull-ups* e resistores em série, afim de proteger os componentes da placa, conforme mostra a FIGURA 10. Cada uma das seis entradas diferenciais dos quais utilizamos cinco - explicitadas a seguir - contém um circuito para proteção, um filtro e um divisor resistivo.

FIGURA 10 – DIAGRAMA DE CANECÕES DO AD7794

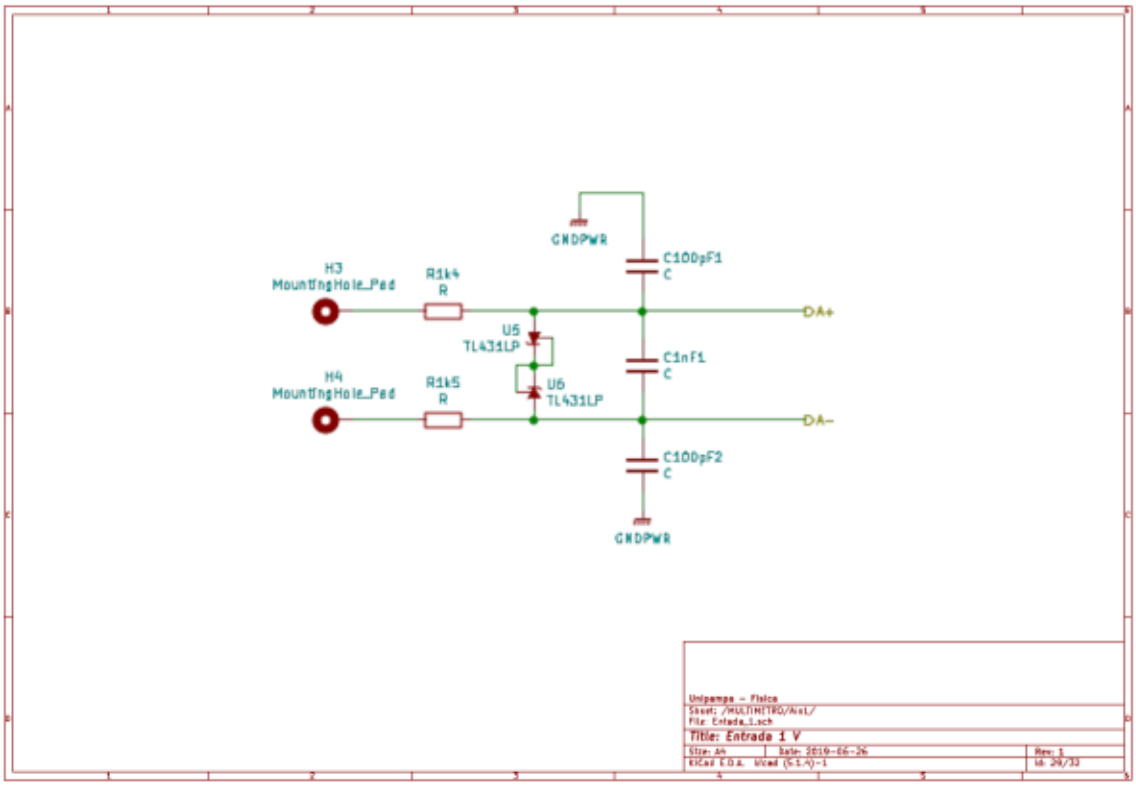


FONTE: Autor (2019)
LEGENDA: Circuito geral do ADC, AD7794.

4.1.1.3 Entrada 1V

A entrada é direta, sem atenuação, em que no ganho 1 aceita tenções entre -1.17 a $+1.17$. Os resistores e capacitores formam um filtro passa baixo, tipo single pole. O mesmo é implementado nas outras entradas. Os dois TL431 formam a proteção de entrada.

FIGURA 11 – ENTRADA 1 V

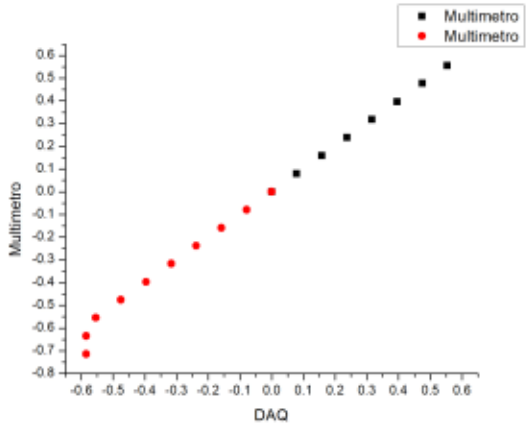


FONTE: Autor (2019)
LEGENDA: Circuito da entrada de $1V$.

Inicialmente, foi planejado utilizar o CI TL431 para proteger as entradas do AD7794, porém, quando a entrada foi testada, a proteção começou a atuar partir de $5.5V$. Como o AD7794 possui uma proteção de entrada que começa a atuar a partir de $0.3V$ de Vcc, o AD7794 começou a conduzir afim de se proteger, como observado na FIGURA 12. Quando o ADC está neste estado, a corrente não deve passar de $10mA$, visto que ele não é projetado para suportar um valor maior que este em modo contínuo. Entretanto, a proteção do AD7794 suporta um pico de corrente de $100mA$. Como esta entrada possui somente um resistor de $1k\Omega$, o usuário deverá tomar cuidado, já que sob estas circunstancias é provável que o limite seja excedido.

Como visto na FIGURA 11, as entradas diferenciais estavam flutuando. Nos ganhos acima de 4 as conversões do AD7794 não eram confiáveis, porque as entradas estavam excedendo o limite de tensão que o ADC conseguia ler. Assim, apesar da diferença entre as duas entradas ser menor que $\frac{1.17}{ganho}$, as tensões poderiam ser superiores as que eram permitidas pelo ADC e isso influenciava a acurácia e precisão das medidas. Para solucionar este problema, foi criado um divisor resistivo por $\frac{AV_{DD}}{2}$ partindo da fonte analógica de $4V$ e conectado em serie a um resistor de $1M\Omega$, que por sua vez, está conectado ao resistor R1k5, (Ain(-)) para *bias* a entrada.

FIGURA 12 – INFLUENCIA DA PROTEÇÃO DE TENSÃO



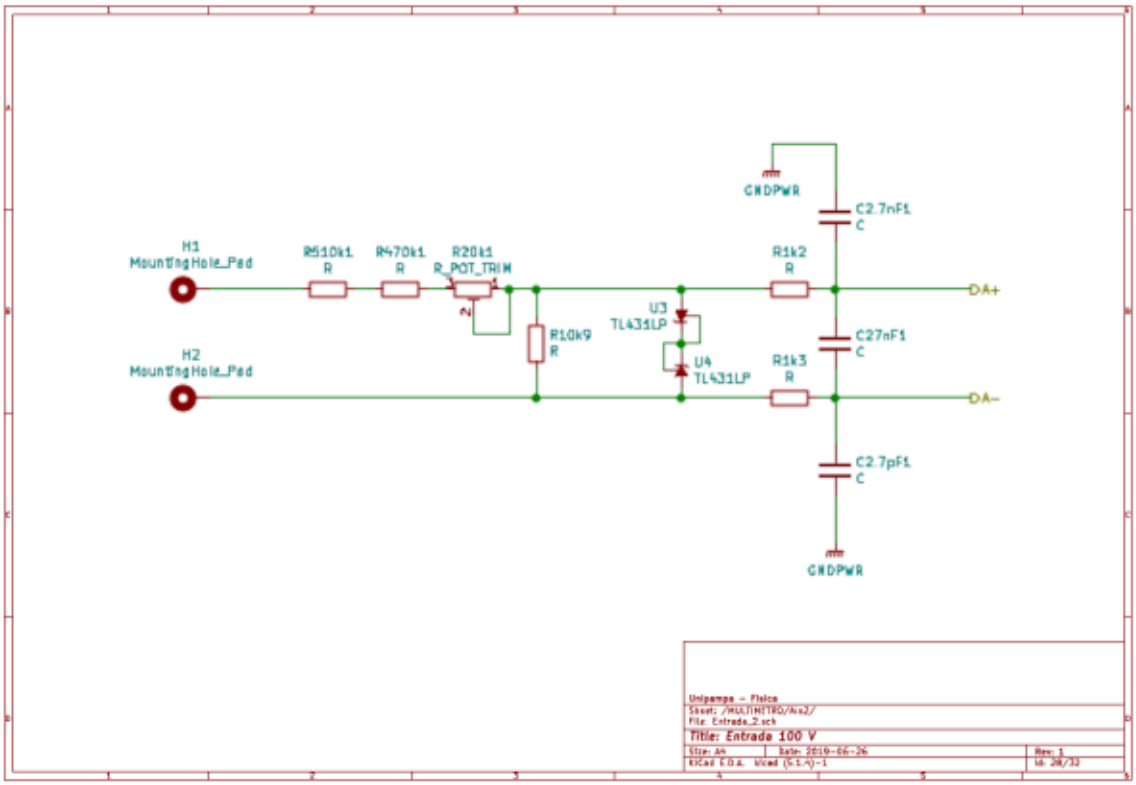
FONTE: Autor (2019)

Legenda: influencia da proteção do canal 0.

4.1.1.4 Entrada 100 V com Impedância de $1M\Omega$

A entrada possui um divisor de tensão resistivo com fator de atenuação de x100, com um ajuste (trimmer de $50k\Omega$) e resistores de 0.1% de precisão e baixo coeficiente de temperatura (25 ppm). A impedância total da entrada é de $1M\Omega$, o qual permite utilizar uma ponta de prova de osciloscópio padrão, atenuado x100, permitindo medições de até $1000V$.

FIGURA 13 – ENTRADA 100 V COM IMPEDÂNCIA DE $1M\Omega$



FONTE: Autor (2019)
LEGENDA: Circuito da entrada de $100V/1M\Omega$.

O circuito de entrada descrito pela FIGURA 13, da mesma forma que aquele da FIGURA 11, também apresentou flutuação nas entradas diferenciais. Para solucionar este problema, foi criado um divisor resistivo por $\frac{AV_{DD}}{2}$ partindo da fonte analógica de $4V$ e conectado em serie a um resistor de $1M\Omega$, que por sua vez, está conectado ao resistor R1k5, (Ain(-)) para *bias* a entrada.Visto que há resistores de valores altos, na ordem de $1M\Omega$, o usuário não precisa preocupar-se com a corrente que passa pelo ADC.

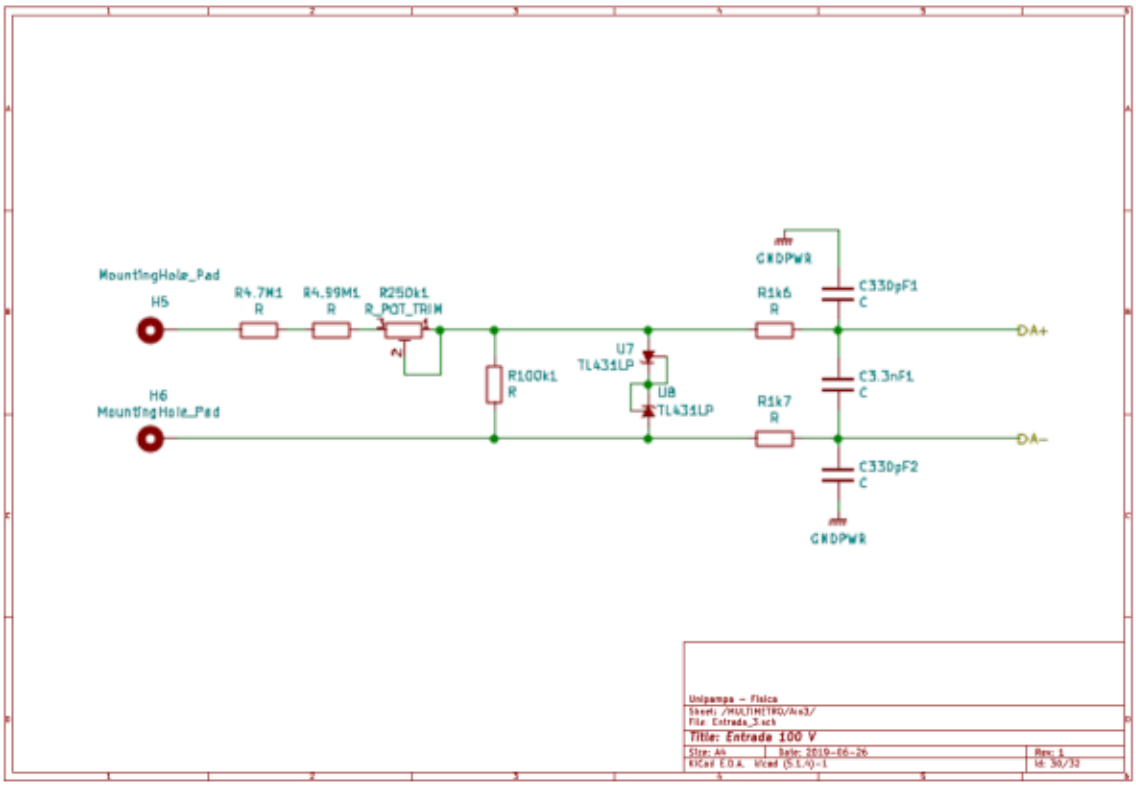
Essa entrada foi projetada para usar uma ponta de prova de osciloscópio que possui atenuadores de 1 e 10. Desta forma, o canal se torna mais flexível, pois o usuário terá a possibilidade de escolher o atenuador da ponta de prova de acordo com suas necessidades.

4.1.1.5 Entrada 100V com Impedância de $10M\Omega$

A entrada possui um divisor resistivo de tenção com fator x100. A sensibilidade da entrada pode ser incrementada (x1 a x128) internamente no AD7794. Possui um ajuste fino (trimmer de $200k\Omega$) e resistores de 0.1% e 1% com baixo coeficiente de temperatura (25 ppm a 2500ppm). A impedância total de entrada é de $10M\Omega$, que corresponde ao padrão de multímetros digitais portáteis.

A FIGURA 14, da mesma forma que a FIGURA 11 e a FIGURA 13, também apresentou flutuação nas entradas diferenciais. Para solucionar este problema, foi criado um divisor resistivo por $\frac{AV_{DD}}{2}$ partindo da fonte analógica de $4V$ e conectado em serie a um resistor de $1M\Omega$, que por sua vez, está conectado ao resistor R1k3, (Ain(-)) para *bias* a entrada. Da mesma forma, como há resistores de valores altos, na ordem de $1M\Omega$, o usuário não precisa preocupar-se com a corrente que passa pelo ADC. Essa entrada não possui a opção de uso de atenuador. Deste modo, a carga sobre as entradas diferenciais é menor.

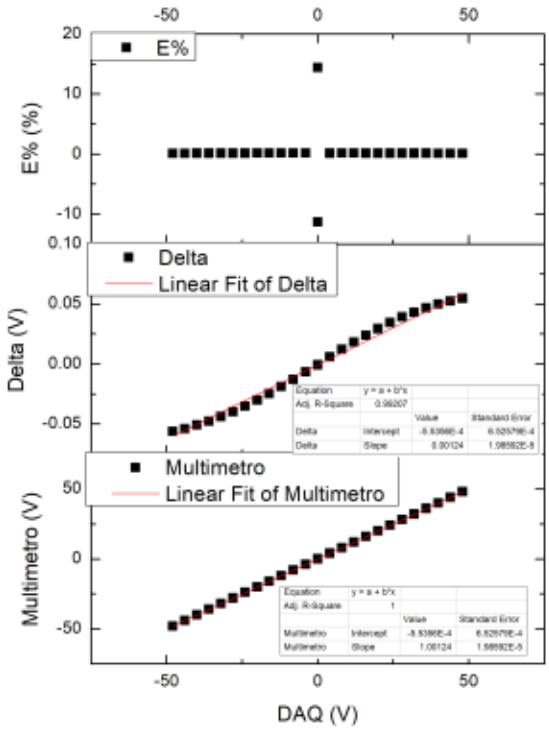
FIGURA 14 – ENTRADA 100 V COM IMPEDÂNCIA DE 10$M\Omega$



FONTE: Autor (2019)
LEGENDA: Circuito da entrada de $100V/10M\Omega$.

Para testar a entrada atenuada e sua influencia na medida do AD7794, foi variasse a tensão de -50 a $50V$, porque não havia uma fonte confiável que variava entre 0 e $120V$ para testar a entrada até o *Fullscale* (vide FIGURA 15). O ajuste linear da diferença entre a medida e a referência resultou em um componente angular de $0,002$ e um *offset* de aproximadamente 0. Portanto, não há necessidade de fazer uma correção via *software*.

FIGURA 15 – INFLUENCIA DO ATENUADOR DE X100 COM IMPEDÂNCIA DE $10M\Omega$



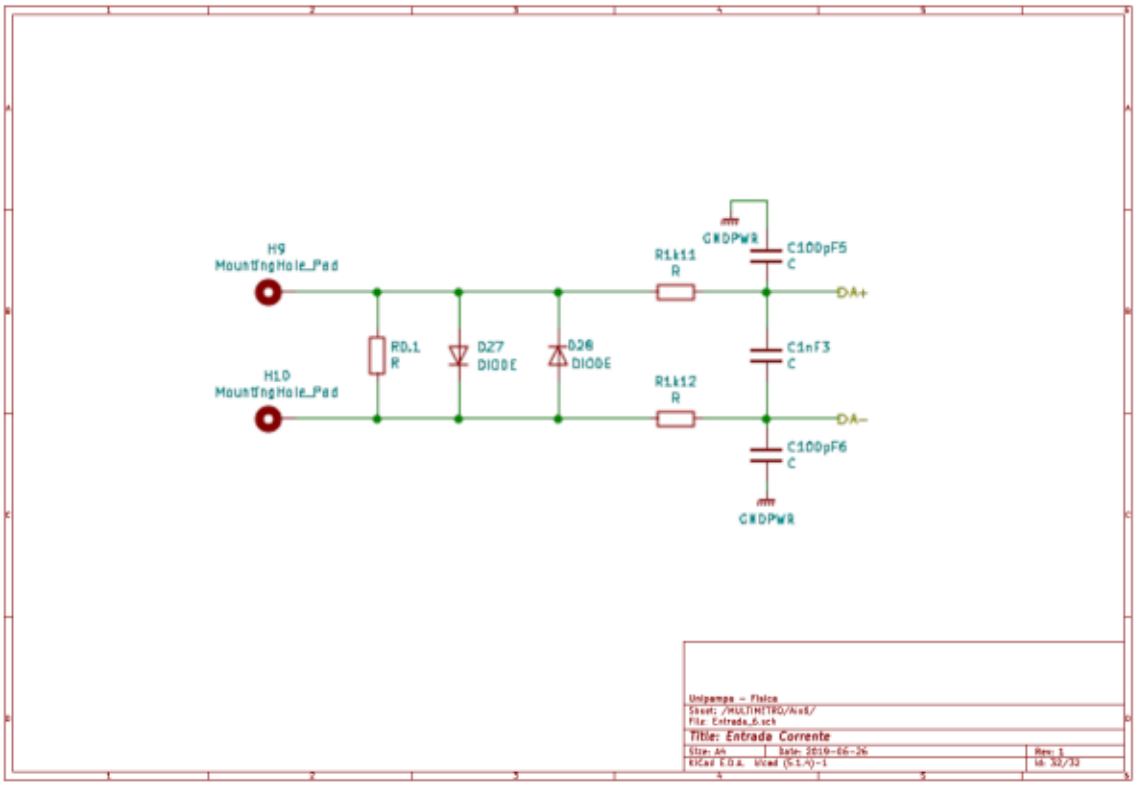
FONTE: Autor (2019)

Legenda: influência do circuito de proteção na entrada x100.

4.1.1.6 Entrada de Corrente

Sua entrada possui dois diodos para proteger as entradas e um *shunt* de 0.1Ω de $1W$. O AD7794 mede a diferença de potência em cima do *shunt* e através da relação de Ohm a corrente é calculada. Para medir o potencial, o módulo usa um ganho de 64 com taxa de conversão de $4Hz$ quando mede a corrente com escala automática para mostrar no *display*. O limite de corrente, definido por projeto é de no máximo 1 *Ampère* ($-1A$ a $+1A$), portanto o AD7794 inicia com ganho 64 indo em a ganhos menores, menor sensibilidade de corrente.

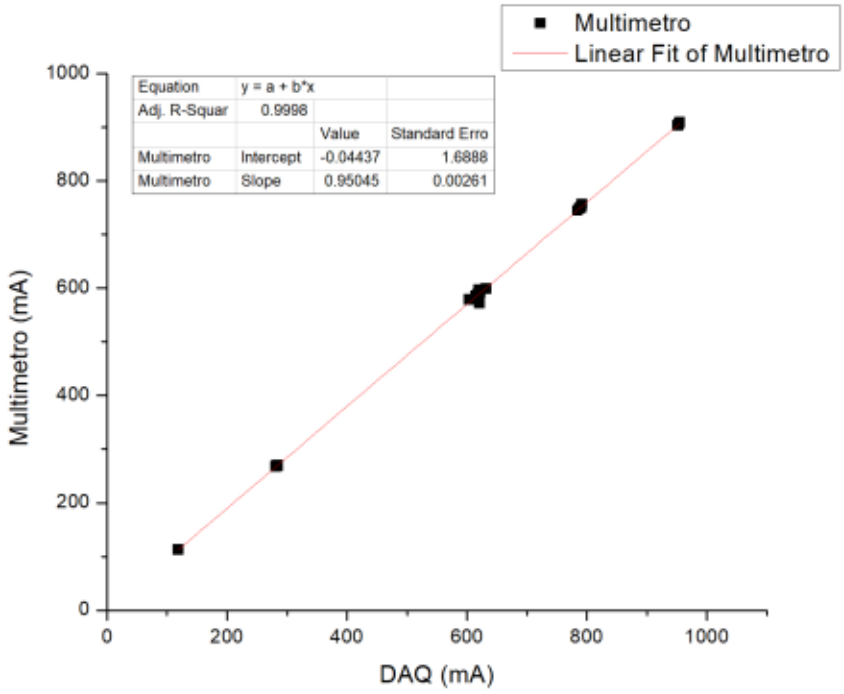
FIGURA 16 – CIRCUITO DA ENTRADA DE CORRENTE



FONTE: Autor (2019)
LEGENDA: A corrente é determinada atravez da queda de tenão no R0,1 e limitada a 1A.

Para caracterizar o amperímetro, os dados foram coletados manualmente através do *display*. A FIGURA 17 mostra a comparação entre a conversão feita pelo módulo e a referência. Como esta entrada tem um erro máximo de 50mA dentro do projetado, foi implementada uma correção baseada na linearização encontrada.

FIGURA 17 – CORRENTE



FONTE: Autor (2019)

Legenda: Gráfico de calibração da corrente para ganho 8.

4.1.1.7 Entrada do Ohmímetro

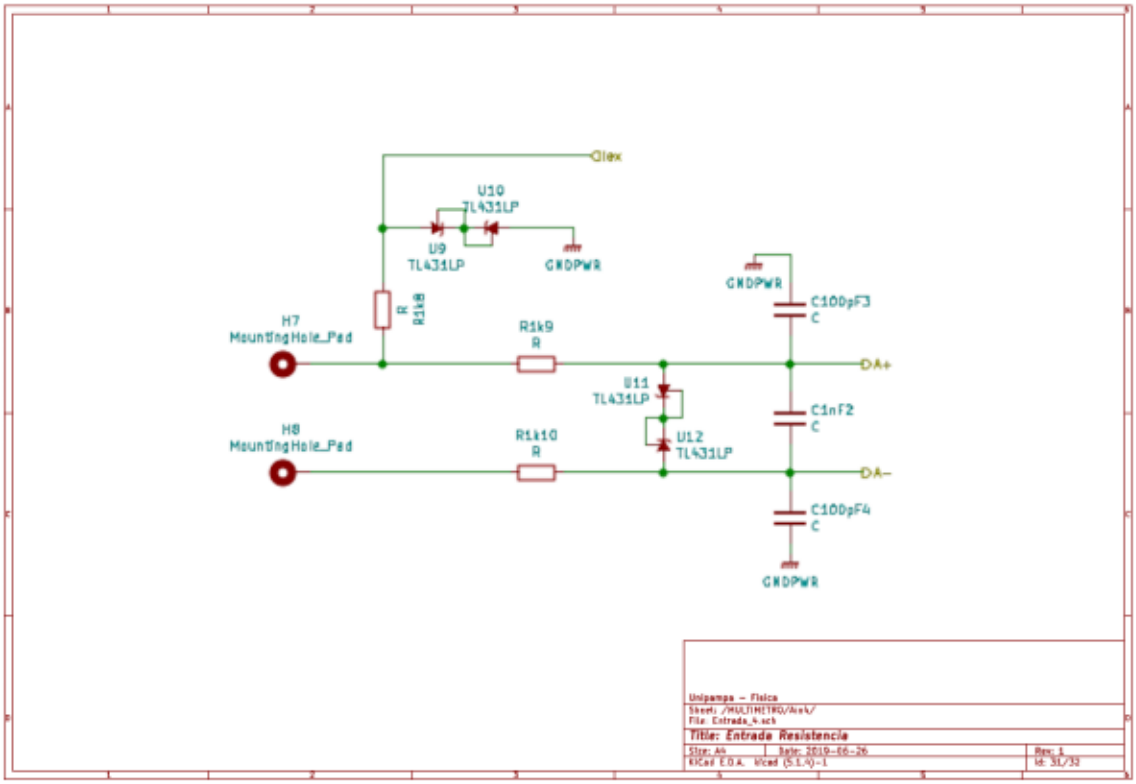
Para medir a resistência é necessário gerar uma corrente conhecida que passa pelo resistor a ser medido. O AD7794 mede uma queda de potencial no resistor e através da relação de Ohm o Arduino calcula o valor da resistência. O ohmímetro é projetado para gerar uma corrente de $10\mu A$, $210\mu A$ e $1mA$. Assim, seria possível medir a resistência de até $1k\Omega$, $4.5k\Omega$ e $100k\Omega$.

Apesar de conseguir medir a diferença de tensão entre as entradas diferencias no ohmímetro, não foi possível gerar uma corrente utilizando a fontes de corrente programável (*excitation current sources*) programáveis dentro do AD7794. Quando o registro é lido com sua configuração *default* após reiniciar o ADC (que deveria ser de valor 0x00), o *Arduino* recebe o valor 0xFF, mesmo conseguindo escrever e ler outros registros. Este comportamento foi observado em dois AD7794. Isso parece mostrar que, para ajustar este registro, há um procedimento específico que deve ser feito para conseguir acessá-lo. Este procedimento não está evidente no *datasheet*[1], visto que houve vários procedimentos para o uso do ADC que não estavam presentes no *datasheet*, mas sim em um *Frequently Asked Questions (FAQ)* publicado pela *Analog Devices* a respeito desse ADC. Isso sustenta a hipótese de que há um procedimento não evidente no material disponibilizado pelo fabricante do conversor. Caso o usuário necessite medir desistência com maior acurácia e precisão, pode montar um circuito

[1] Disponível em: https://www.analog.com/media/en/technical-documentation/data-sheets/AD7794_7795.pdf

especifico, por exemplo, o *flowmeter* do datasheet que determina a resistência pelo método *ratiometric*.

FIGURA 18 – ENTRADA DE RESISTÊNCIA

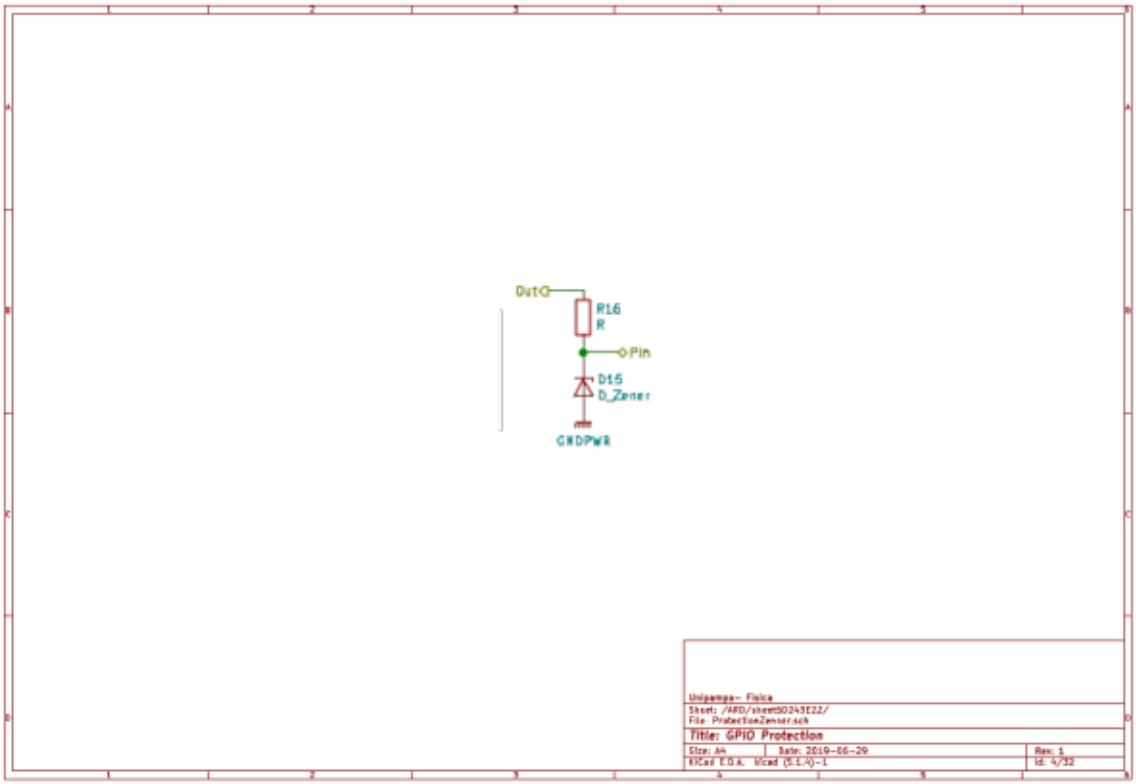


FONTE: Autor (2019)
LEGENDA: Diagrama do circuito de entrada para a medição de resistencia.

4.1.1.8 Proteção das Portas Digitais

Para proteger o Arduino de circuitos externos, foi previsto na PCB uma proteção de tensão, via um diodo tipo Zener, e de corrente, através de um resistor em série. Assim, a tensão sobre o pino digital não pode exceder os $5.1V$ que o Arduino suporta, e o resistor em série limita a corrente que pode passar pelo Zener, protegendo-o, assim como o Arduino, que não pode fornecer mais de $50mA$.

FIGURA 19 – PROTEÇÃO DAS PORTAS DIGITAIS DO ARDUINO.



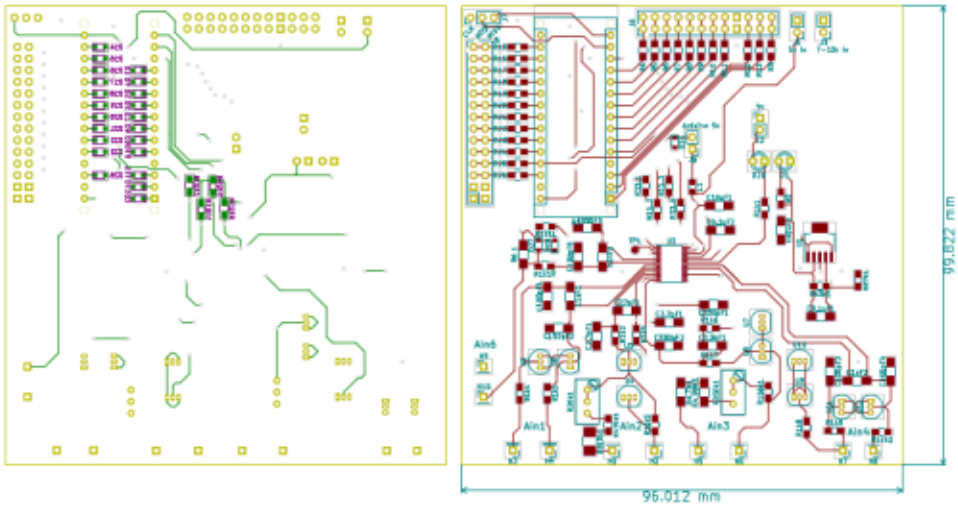
FONTE: Autor (2019)
LEGENDA: Diagrama de proteção das saídas digitais, disponível na PCB como opcional.

4.1.1.9 *Printed Circuit Board (PCB)*

Para o desenvolvimento deste trabalho, também foram feitas *PCBs*. A placa na FIGURA 20 possui duas camadas (face frontal e traseira) tentou-se concentrar as trilhas na face frontal, sendo que algumas trilhas foram traçadas na face traseira. A areá de cobre não ocupada em ambas as faces foram traçadas como se fosse um plano de terra. As placas foram fabricadas na China pela empresa JLCPCB[2], a baixo custo (2 USD por cinco placas). Todavia, devido a distância, houve grande latência, desde a chegada das peças no Brasil após sua produção, até sua entrega aos pesquisadores para a continuação do trabalho.

[2] Para mais informações sobre a empresa e seus serviços, visite: https://jlcpcb.com/. Acesso em: 16 nov. 2019.

FIGURA 20 – PLACA CIRCUITO IMPRESSO DA PLACA AD7794



FONTE: Autor (2019)

Legenda: *Layout* do modulo multímetro, Versão 1.0.

4.2 FONTE

4.2.1 *Hardware*

Para desenvolver o módulo Fonte, foi criada uma biblioteca para o *Arduino*, disponível no Apêndice C. Para visualizar e modificar a saída do módulo, foi utilizado um *OLED* SSD1306, controlado através da biblioteca *SSD1306AsciiWire* [3]. Um botão que possui um *encoder* foi utilizado para controlar o dispositivo, sem o uso de um computador, para a contagem dos pulsos saindo do *encoder* para ajustar a tensão do DAC, pode-se utilizar o *encoder* ou um computador, porém, ambos utilizam memórias diferentes e, portanto, seus valores são independentes.

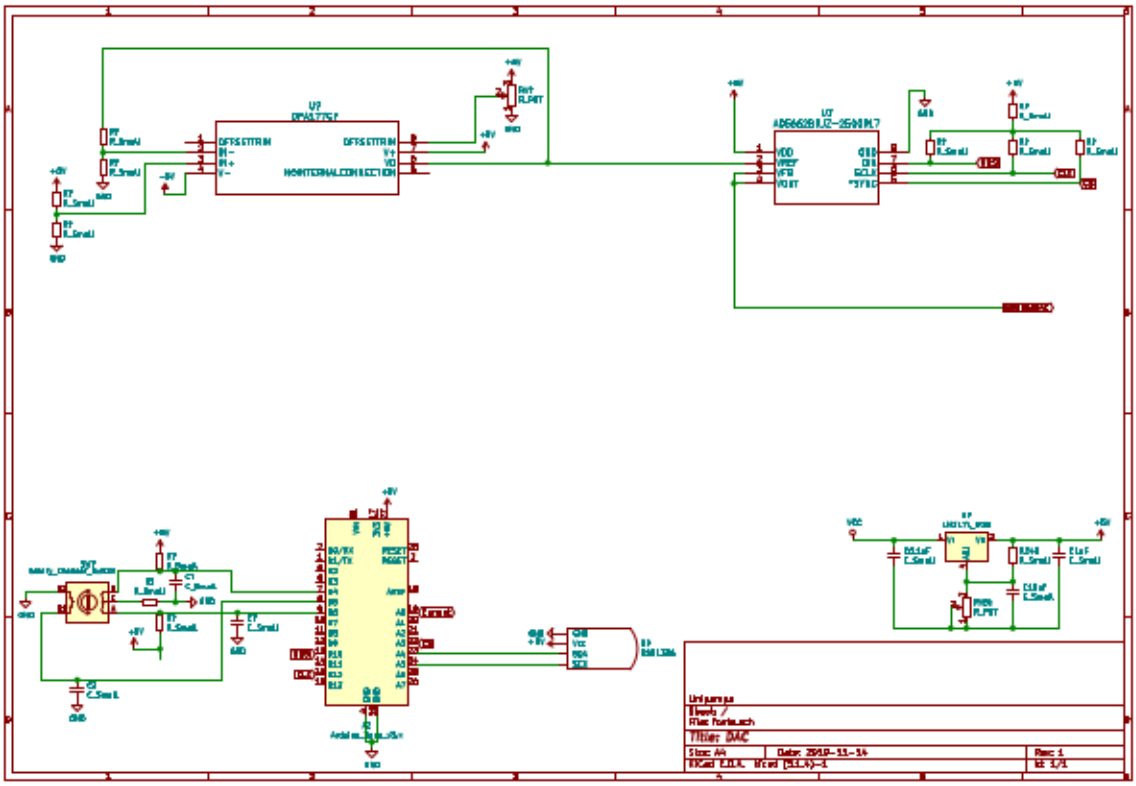
Para caracterizar a saída de tensão do DAC5662 quanto a sua estabilidade e precisão, foi utilizado um multímetro de precisão ($7\frac{1}{2}$ dígitos) e o Python, que se comunica com o multímetro através da interface *The Hewlett-Packard Interface Bus* (HPIB) para realizar a coleta de dados. Com os dados, compara-se a diferença da tensão teórica, utilizando a equação 4.3, onde a tensão de saída é v_{out}, V_{ref} é a tensão de referência, n é o numero de bits do DAC e entrada digital é um código binário de n bits que é utilizado para ajustar V_{out}.

$$V_{out} = \frac{V_{ref}}{2^n - 1} entrada\ digital \tag{4.3}$$

Para gerar a tensão de saída, foi utilizado o ADC AD5662, com uma referência de $4V$, através de um amplificador operacional (OP-AMP) OPA17FP, alimentado por $5V$, através de uma fonte externa. O DAC utiliza um padrão de comunicação compatível com SPI, todavia, foi criado uma biblioteca para comunicar e controlar o DAC através do *hardware* SPI do Arduino, ou através de um SPI *bit bang*. Deste modo, o usuário poderá conectar o DAC em qualquer pino digital do microcontrolador.

[3] disponível em: https://github.com/greiman/SSD1306Ascii

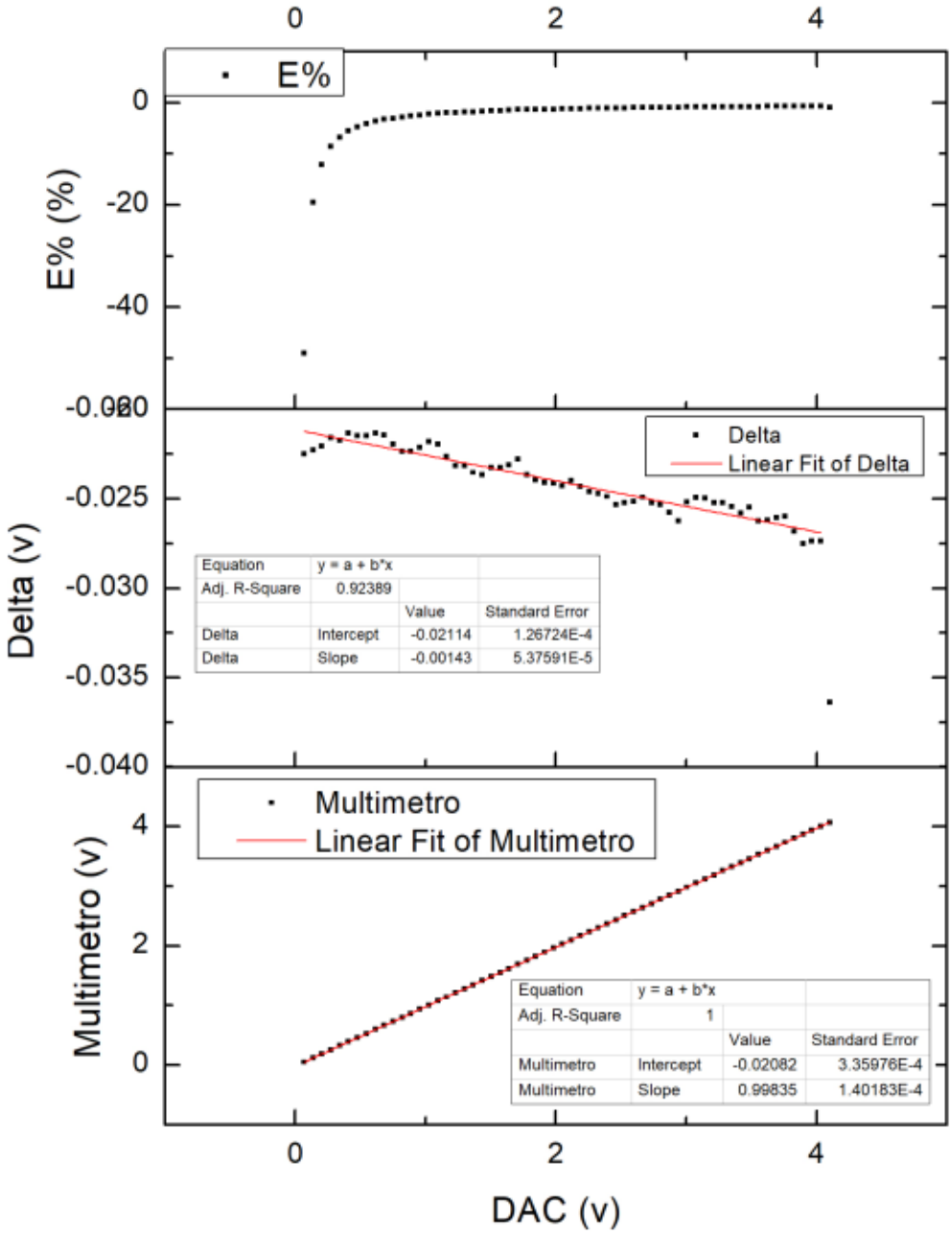
FIGURA 21 – ESQUEMÁTICO DO DAC



FONTE: Autor (2019)

Para averiguar a saída do DAC foi feito um programa, utilizando o *software* desenvolvido neste trabalho, para ajustar a saída do DAC que foi medida e comparar com a do multímetro HP3457A As especificações do AD5662 garante 12 bits de acurácia de $\frac{12log(2)}{log(10)} = 3.5$ digitos , conforme mostra a FIGURA 22. Como observado, a diferença entre o DAC e o multímetro está dentro do esperado e não é necessário fazer uma correção via *software*.

FIGURA 22 – CIRCUITO SIMPLIFICADO DA FONTE



FONTE: Autor (2019)

Legenda: Saida do DAC comparada com a do HP3457A.

4.3 *SOFTWARE*

4.3.1 *Arduino-Comunicação*

Para comunicar-se com o computador, o Arduino tem um *interrupt* [4] em sua entrada serial, onde ele monta um *string* carácter a carácter, com tamanho máximo de 200 caracteres. Porém, a avaliação desse *string* é executada dentro de um *loop* infinito. Portanto, a interpretação do string pode demorar, dependendo da demanda posta sobre o Arduíno. Quando o *string* é avaliado, o programa procura "/n". Caso seja encontrado, o string será interpretado e comparado com a tabela de comandos. A seguir, encontra-se um exemplo de comunicação. Somente a última linha dessa comunicação é considerada válida para interpretação e será explicitada posteriormente.

0
00
000
0006
00060
000601
0006010
00060102
000601020
0006010200/
0006010200/n

Fonte: Autor (2019)

A partir do acréscimo da tela LCD ao Arduino, a comunicação provou-se instável e causava problemas ao controlar os demais funcionamentos do microcontrolador, pois o *Arduino* não diferenciava o final de um comando com inicio de outro. Para solucionar esta situação, foi acrescentado dois filtros na parte de comunicação, além do descrito anteriormente. Deste modo, o Arduino passou a procurar um *string* de início de comunicação (DANI) e separar os dados entre este início e o *string* "/n". Com esses dados separados, ele consegue selecionar os primeiros 12 caracteres, truncando os demais. Após o filtro da comunicação, o string volta a ser interpretado como anteriormente.

D
DA
DAN
DANI
DANI0
DANI00

[4] Uma interrupção é um sinal para a CPU começar imediatamente a executar código diferente, *Interrupt service routine*, e a CPU se lembrará da localização da próxima instrução que ela executaria armazenando esse endereço de instrução em um registro ou local de memória e, então, pulará diretamente para o código designado pelo programador. Disponível em: https://www.embeddedrelated.com/showarticle/469.php. Acesso em: 16 nov. 2018

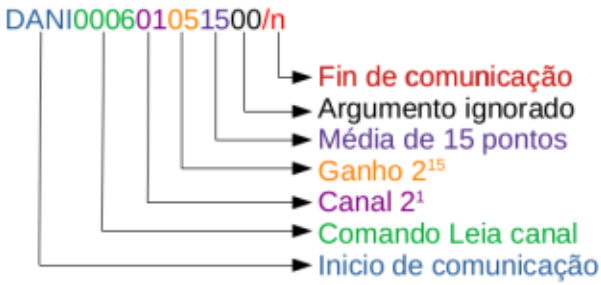
DANI000
DANI0006
DANI00060
DANI000601
DANI0006010
DANI00060102
DANI000601020
DANI0006010200
DANI0006010200/
DANI0006010200/n

Fonte: Autor (2019)

Legenda: Exemplo de *string* válido para comunicação (2019)

A interpretação da comunicação é feita através da função IN do código do multímetro/DAQ, descrito na seção E.1(linhas 82-218). Inicialmente, os quatro primeiros caracteres referenciam o comando/função e os demais são argumentos das funções. Os comandos passam por um *case statment* e os argumentos são tratados de acordo com o comando, *id est*, pode ser por transformação em número, retornado pela serial ou por qualquer ação desejada, como descritos na TABELA 2, na TABELA 3 e na TABELA 4. A FIGURA 23 serve como exemplo de comunicação.

FIGURA 23 – ESTRUTURA DE COMANDO



FONTE: Autor (2019)

Legenda: Exemplo de Comando para o Multímetro/DAQ

4.3.2 *Python-Comunicação*

Para comunicar-se com o Arduíno, utilizou-se a linguagem *Python* com o pacote *Pyserial*, um módulo que faz o acesso para as portas seriais para *Windows*, *Linux* e *OSX*. Foi escrito um módulo de *Python* A.1 em que o usuário importará seu código. Tal modulo é responsável por fazer a conexão entre o *Arduino* e o computador do usuário.

O código *SerialConnection*, do Apêndice A.1, descreve uma função que procura dispositivos conectados na porta serial, real ou virtual, através da porta USB. Além disso, lista

os dispositivos - caso seja encontrado mais do que um - e retorna um objeto [5]. Para comunicar com o dispositivo encontrado, é preciso encontrar a velocidade de comunicação. Para isso, o módulo automaticamente varre através das velocidades seriais comuns e encontra o *baudrate*.

Para que o *Python* se conecte ao dispositivo, será necessário importar o módulo de comunicação com a classe *SerialConnection*. Quando uma instância da classe é declarada, ela automaticamente varre todos os dispositivos classificado como *COM port*, sendo eles virtuais ou não. Enquanto varre os dispositivos, o *Python* se comunica com eles e pede um *string* padrão. Caso recebido o *script*, ele pede a identificação do módulo afim de listá-los para o usuário selecionar.

O *script* abaixo mostra como conectar o *Arduino* ao dispositivo e controlá-lo. Há quatro comandos que podem ser utilizados para interagir com o dispositivo: *SendData*, *ReadData*, *SendRead* e *WatchPort*. O *SendData(d)* manda um *string* para o dispositivo, de acordo com o protocolo e *ReadData* retorna os dados no *buffer* do porta serial. A função *SendRead(d)* é uma junção dos dois comandos anteriores, ela manda um comando ao módulo e retorna a resposta. Caso a função não receba retorno, é assumido que houve um erro e o comando é reenviado, até receber um retorno do módulo. *WatchPort* foi criado somente para ser utilizado com *multithreading* e *multiprocess*. Este comando precisa ser um *thread* independente do processo principal.

```
1   import Kaki
2   from AD7794Codes import *
3   # cria instancia e excluino a porta COM3
4   DAQ = Kaki.SerialConnection(EXCLUDE=['COM3'])
5   # Mostra a classe criada para o usuario
6   print(DAQ)
7   # Lista de dispositivos para excluir da busca
8   print(DAQ.EXCLUDE)
9   # Porta que o dispositivo esta conectado
10  print(DAQ.PORTS)
11  # Manda um comando, le um valor e mostra para o usuario
12  print(DAQ.SendRead(TEMP))
13  # Manda comando para Toggle um pino digital 5
14  DAQ.SendData(TOGGLEIO+"5")
```

Ao executar o código, o usuário é questionado sobre qual placa deseja conectar. Depois que o programa estabelece a conexão, ele realiza as seguintes operações: escreve o objeto, mostra as portas seriais - que podem ser ignoradas caso o usuário deseje conectar-se com outras placas -, escreve a porta serial que o dispositivo está conectado e a resposta que

[5] Nome ou caminho completo do dispositivo, por ex. /dev/ttyUSB0. Esta é também a informação retornada como primeiro elemento quando acessada pelo índice.

recebe do módulo. O(s) primeiro(s) item(ns) da resposta são colocados em uma lista com a(s) mensagem(ns) recebida(s) pelo computador, e, o último item da lista é o tempo em segundos, desde dia 01 de janeiro de 1970, como um número de ponto flutuante em que o *Python* recebeu a comunicação.

```
Select Device
0 - 7794, COM14
C to cancel
Number: 0
<Kaki.SerialConnection object at 0x04108190>
['COM3', 'COM14']
['COM14']
['25.84', 1573999375.920371]
```

Para comunicar-se com o dispositivo utilizando *multithreading*, é necessário utilizar a classe *Glue* contida no mesmo arquivo que a classe *SerialConnection*. Ao contrario da forma anterior, o *buffer* da porta serial está sendo constantemente lido em um processo independente, rodando a função *WatchPort*. Pela existência do *GIL* (*Global Interpreter Lock* [6]) é necessário utilizar um *FIFO*. Para acessar esta memória, o usuário utiliza a função *Read(Nome)*. Se o *FIFO* estiver vazio, a função retorna um *bool* falso, do contrario, é retornada a mensagem recebida pelo módulo. Outra diferença entre as duas formas de comunicação é que, além de escolher a placa a ser conectada, o usuário a nomeia, com o objetivo de facilitar o uso da ferramenta. Portanto, as duas formas de interagir com o módulo são através dos comandos *Send* e *Read* e ambas necessitam do nome dado para a placa. Abaixo, há um exemplo de leitura de temperatura com o módulo multímetro.

```
1 from Kaki import Glue
2 din = False
3 if __name__ == '__main__':
4     glue = Glue()
5     glue.Send('y', TEMP)
6     while din == False:
7         din = glue.Read('y')
8     print(din)
```

E, abaixo, o resultado do código acima.

[6] um *mutex* que protege o acesso a objetos *Python*, impedindo que vários *threads* executem *bytecodes* do *Python* de uma só vez. Esse bloqueio é necessário principalmente porque o gerenciamento de memória do *Python* não é seguro para *threads*. (No entanto, como o *GIL* existe, outros recursos passaram a depender das garantias que ele impõe).

```
Select Device
0 - 7794, COM14
C to cancel
Number: 0
What do you want to name the board? y
['25.55', 1574004670.3461304]
```

Inicialmente, na primeira forma de comunicação havia somente dois comandos (*ReadData* e *SendData*), porém, foi observado que o *Python* falhava ao ler a primeira comunicação. Nesse sentido, foi criada a função *SendRead* para facilitar o uso do dispositivo e lidar com falha de comunicação. Desta observação, nasceu a ideia de criar uma função que constantemente 'olha' para o *buffer* e salva os dados no momento que entram.

Se o *Python* estiver sempre olhando para a porta serial, o programa efetivamente pararia nesta etapa, sem o usuário poder interagir com o código numa interface qualquer. Desta forma, foi implementado o código utilizando *threads*. Como esta implementação é mais complexa, foi criada a classe *Glue* para facilitar o uso. Uma vantagem de utilizar esta implementação é a possibilidade de criar uma interface gráfica, como através do *pyqt5*, possibilitando uma maior facilidade de manuseio do dispositivo. Visto que existe esta possibilidade, as funções e classes foram escritas de tal maneira a facilmente implementar um pacote de interface gráfico no trabalho.

Para visualização dos dados em tempo real, basta chamar um pacote de gráfico do *Python* de modo que ele permita a animação com *pyqtgraph* ou *matplotlib*. Inicialmente, utilizou-se o pacote de gráfico *pyqtgraph*, pois é um um pacote recente e recomendado pela comunidade *Python* para gráficos em tempo real. Apesar de conseguir utilizar este pacote em uma versão mais antiga do projeto *pyqtgragh*, o trabalho aqui desenvolvido não teve tempo suficiente para usufruir do uso de uma ferramenta tão sofisticada. Se a utilizasse, o programa ocuparia menos recursos do computador, já que o pacote *matplotlib* é antigo e não optimizado. Abaixo, segue um exemplo de leitura de temperatura do CI AD7794. A FIGURA 24 apresenta o resultado obtido durante a execução do programa.

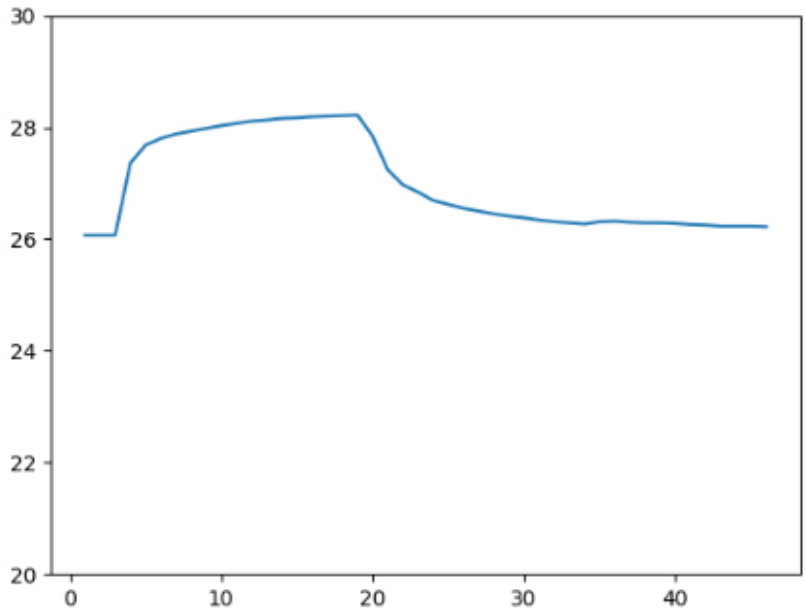
```
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import style
from  AD7794Codes import *
from SerialConnect import
    ↪  SerialConnection as SC
DAQ = SC()
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)

plt.ylim(0,100)
xs = []
ys = []
def animate(i):
    DAQ.SendData("00040000")
    y = DAQ.ReadData
    ys.append(float(y[0]))
    xs.append(len(ys))

    ax1.clear()
    ax1.set_ylim([20,30])
    ax1.plot(xs, ys)

ani = animation.FuncAnimation(fig,
    ↪  animate, interval=1000, blit=False)
plt.show()
```

FIGURA 24 – GRAFICO GERADO EM TEMPO REAL



FONTE: Autor (2019)

Legenda: Temperatura do AD7794 sendo aquecido pelo dedo do autor.

Para atender a necessidade de salvar os dados, foi escrito um módulo *Python* para isso. Neste módulo, há duas formas de salvar dados: em arquivo *CVS* ou *Excel*. Como o módulo *openpyxl* não é possível de se instalar nas versões anteriores à 3.6, o *script* automaticamente detecta se *openpyxl* está instalado e, caso não esteja, o *script* avisa o usuário. O mesmo é feito para salvar em *CVS*, entretanto, não é obrigatório e está presente somente para controlar erro. Para salvar dados utilizando o *CVS* é necessário criar o arquivo e, se já existe um arquivo com o mesmo nome desejado, o original será apagado. As funções presentes na classe *DataLogCVS* são *Create* e *Save*. São funções do tipo *variadic* [7] , cujo primeiro argumento obrigatório é o nome do arquivo e os outros são os *n* dados que são salvos. O delimitador para este é "" para facilitar a leitura. A seguir, há um exemplo de medição de temperatura salva em um arquivo *CVS*.

[7] Funções que permitem um número de argumentos não determinados.

```
1  import Kaki
2  from DataLog import DataLogCVS
3  from AD7794Codes import *
4
5  # cria um arquivo para salvar dados e
   ↪  cabeçario
6  DataLogCVS.Create("Exemplo",
   ↪  "temperatura")
7  # cria instancia
8  DAQ = Kaki.SerialConnection()
9  for i in range(10):
10     # Manda um comando, le um valor e
       ↪  mostra para o usuario
11     a = DAQ.SendRead(TEMP)
12     # Salva o dado no arquivo
13     DataLogCVS.Save("Exemplo", a[0])
```

Para salvar os dados em um arquivo *Excel*, há apenas uma função: Save. A função é do tipo *variadic* de dois argumentos obrigatórios, *id est*, o nome do arquivo e o nome da planilha que deseja salvar os dados. Caso não exista o arquivo em questão, ele é automaticamente criado, porém, se existe um arquivo, todos os dados serão preservados e os novos dados serão acrescentados a partir da última coluna que contém dados. A seguir, há um exemplo utilizando esta classe e realizando a mesma operação que o exemplo anterior.

```
1  import Kaki
2  from DataLog import DataLogExcel
3  from AD7794Codes import *
4
5  # cria cabeçario
6  DataLogExcel.Save("Exemplo", "Sheet1",
   ↪  "Temperatura")
7  # cria instancia
8  DAQ = Kaki.SerialConnection()
9  for i in range(10):
10     # Manda um comando, le um valor e
       ↪  mostra para o usuario
11     a = DAQ.SendRead(TEMP)
12     # Salva o dado no arquivo
13     DataLogExcel.Save("Exemplo",
       ↪  "Sheet1", a[0])
```

5 CONCLUSÕES FINAIS

O presente trabalho teve como objetivo desenvolver instrumentos de medidas que se aproximam da aplicabilidade de instrumentos de laboratório de pesquisa, com capacidade de salvar dados e de ser controlado por um computador. Especificamente, foram desenvolvidos dois módulos, sendo um deles capaz de gerar tensão e outro capaz de medir tensão, corrente e resistência. Neste trabalho, foi elaborada uma interface *Python-Arduino* que permite salvar os dados, controlar os dispositivos e plotar um gráfico em tempo real, além de direcioná-la de forma a facilitar a implementação de uma interface gráfica.

Além disso, foi desenvolvido um programa que controla um *DAC* através da interface Python-Arduino, em que pode-se criar um circuito de potência específico, de acordo com o desejo do usuário, para servir como fonte programável. O módulo Multímetro/DAQ consegue medir tensões nas entradas de $1V$ e duas entradas de $100V$ com impedâncias de $1M\Omega$ e $10M\Omega$, com medição de corrente, dentro das especificações do AD7794 que equivalem a um multímetro de 6 dígitos.

Durante o desenvolvimento deste trabalho, também aconteceram algumas modificações. A primeira delas relacionou-se com a realização das medições com o AD7794, pois não era esperada a necessidade de isolá-lo do *Arduino*, tanto no que se referia à sua comunicação, quanto à sua alimentação. Consequentemente, o computador que o alimentava também foi isolado do AD7784. Para buscar solucionar o problema, realizou-se a revisão detalhada da biblioteca desenvolvida para controlar o AD7794 e a verificação das conexões elétricas de todos os equipamentos utilizados. Entretanto, ambas as tentativas não solucionaram o problema, já que o empecilho se encontrava, de fato, nas entradas, que precisavam ser *biased*.

Outra modificação relacionou-se ao entendimento de como comunicar-se com o AD7794, pois muitas informações fundamentais não se encontravam no *datasheet*. A respeito da fonte mencionada no tópico 4.2, a interface de controle foi implementada em outro conversor *DAC*, o AD5781 - disponível no Apêndice F, além do AD5662. Ao comparar o código dos dois dispositivos, é possível observar que, com pequenas mudanças, é viável a implementação desse controle em diversos dispositivos que possuem finalidades semelhantes.

Devido a acurácia e precisão do conversor analógico digital e sua habilidade de controlar uma fonte, é possível realizar experimentos ou rotinas de calibração, como de sensores de temperatura. Nesse sentido, por ter baixo custo em comparação com um instrumento comercial [1] e por ser de natureza aberta, o projeto pode possibilitar que um curso ou escola com poucos recursos financeiros disponíveis consiga ter condições para utilizar aparatos automatizados.

Acredita-se que esse trabalho possa servir como base para a realização de expe-

[1] Por exemplo: o modulo *MyDAQ* custa 371 USD e necessita do software Labview a um custo de 3150 USD e o modulo multimetro/DAQ tem um custo de 100 USD.

rimentos automatizados futuramente. No que tange ao ensino de Física, os experimentos desenvolvidos em sala de aula também podem ser automatizados. Isso implica que a discussão da física teórica por trás de cada experimento desenvolvido pode ser trabalhada nas escolas públicas brasileiras, sem resumir-se unicamente à aquisição de dados.

REFERÊNCIAS

AMORIM, Helio Salim do; DIAS, Marco Adriano; SOARES, Vitorvani. Sensores digitais de temperatura com tecnologia one-wire: Um exemplo de aplicação didática na área de condução térmica. **Revista Brasileira de Ensino de Física**, v. 37, n. 4, 2015. DOI: 10.1590/S1806-11173742009. Endereço: http://dx.doi.org/10.1590/S1806-11173742009. Acesso em: 7 jun. 2019.

ANALOG DEVICES, INC. **AD779x Instrumentation Converters Frequently Asked Questions**. [*s.l.*], 2010. Endereço: https://www.analog.com/media/en/technical-documentation/frequently-asked-questions/AD779x_FAQ_Instru_Conv.pdf. Acesso em: 7 jun. 2019.

AZEVEDO, Hernani Luiz *et al.* O USO DO EXPERIMENTO NO ENSINO DA FÍSICA: TENDÊNCIAS A PARTIR DO LEVANTAMENTO DOS ARTIGOS EM PERIÓDICOS DA ÁREA NO BRASIL. **Encontro Nacional de pesquisa em Educação em Ciências**, 2009.

BUECHLEY, Leah; EISENBERG, Michael. The LilyPad Arduino: Towards Wearable Engineering for Everyone. In: 2. v. 7. Endereço: https://ieeexplore.ieee.org/document/4487082. Acesso em: 22 jun. 2019.

CAVALCANTE, Marisa Almeida; TAVOLARO, Cristiane Rodrigues Caetano; MOLISANI, Elio. Física com Arduino para iniciantes. **Revista Brasileira de Ensino de Física**, v. 33, n. 4, 2011.

COWASAKI, Darren. **DUEGUI**. [*s.l.*: s.n.], 2014. Endereço: https://github.com/ghlawrence2000/DUEGUI. Acesso em: 7 jun. 2019.

DIONISIO, Guilherme; SPALDING, Luiz Eduardo Schardong. Visualização da forma de onda e conteúdo harmônico da corrente elétrica alternada em eletrodomésticos. **Revista Brasileira de Ensino de Física**, v. 39, n. 1, 2017. DOI: 10.1590/1806-9126-RBEF-2016-0121. Endereço: http://dx.doi.org/10.1590/1806-9126-RBEF-2016-0121. Acesso em: 7 jun. 2019.

DWORAKOWSKI, Luiz Antonio *et al.* Uso da plataforma Arduino e do software PLX-DAQ para construção de gráficos de movimento em tempo real. **Revista INNOVER**, v. 1, n. 4, 2014.

FETZNER FILHO, Gilberto. Experimentos de baixo custo para o ensino de Física em Nível Médio usando a placa Arduino-UNO, 2015.

GRASSELLI, Erasmo Carlos; GARDELLI, Daniel. O ENSINO DA FÍSICA PELA EXPERIMENTAÇÃO NO ENSINO MÉDIO: DA TEORIA À PRÁTICA. **Cadernos PDE**, v. 2, 2014.

H. CORDOVA, A. C. Tort. Medida de g com a placa Arduino em um experimento simples de queda livre. **Revista Brasileira de Ensino de Física**, v. 38, n. 2, 2016. DOI: 10.1590/1806-9126-RBEF-2015-0012. Endereço: http://dx.doi.org/10.1590/1806-9126-RBEF-2015-0012. Acesso em: 7 jun. 2019.

KOENKA, Israel Joel; SÁIZ, Jorge; C.HAUSER, Peter. Instrumentino:An open-source modular Python framework for controlling Arduino based experimental instruments. **Computer Physics Communication**, v. 185, n. 10, 2014. Endereço: https://www.sciencedirect.com/science/article/pii/S0010465514002112?via%3Dihub.

MADSHOBYE. **Guino**. [*s.l.*: s.n.], 2014. Endereço: https://github.com/madshobye/guino. Acesso em: 7 jun. 2019.

MOURÃO, Oséias. **Arduino & Ensino de Física Automação de práticas experimentais**. Tianguá: Clube dos Autores, 2018.

NATIONAL INSTRUMENTS. **Arduino™ Compatible Compiler for LabVIEW by Aledyne-TSXperts**. [*s.l.*: s.n.], 2019. Endereço: https://www.tsxperts.com/arduino-compatible-compiler-for-labview/. Acesso em: 7 jun. 2019.

OLIVEIRA, Ivanor Nunes de *et al.* AUTOMATIZAÇÃO DE LABORATÓRIOS DIDÁTICOS DE FÍSICA GERAL - CONSTRUÇÃO DE UMA MAQUETE EXPERIMENTAL AUTOMATIZADA PARA A DETERMINAÇÃO DA CONSTANTE DE PLANCK. **Experiências em Ensino de Ciências**, v. 13, n. 1, 2005.

OPENSOURCE.ORG. **Licenses & Standards**. [*s.l.*: s.n.]. Endereço: https://opensource.org/licenses. Acesso em: 7 jun. 2019.

RED HAT, Inc. **What is open hardware?** [*s.l.*: s.n.]. Endereço: https://opensource.com/resources/what-open-hardware. Acesso em: 7 jun. 2019.

RED HAT, Inc. **What is open source?** [*s.l.*: s.n.]. Endereço: https://opensource.com/resources/what-open-source. Acesso em: 7 jun. 2019.

RMAGTIBAY. **Arduino I/O-Matlab Basic Tutorial**. [*s.l.*: s.n.], 2013. Endereço: https://www.instructables.com/id/Arduino-IO-Matlab-basic-tutorial/. Acesso em: 7 jun. 2019.

SILVA, Janicleide T. da; SILVA, Josiane T. da; LIMA, Gustavo F. de. Controle E Monitoramento de Nível Utilizando Plataforma Open Source Arduino. **Revista INNOVER**, v. 1, n. 4, 2014.

SILVEIRA, Sérgio; GIRARDI, Mauricio. Desenvolvimento de um kit experimental com Arduino para o ensino de Física Moderna no Ensino Médio. **Revista Brasileira de Ensino de Física**, v. 39, n. 4, 2017. DOI: 10.1590/1806-9126-rbef-2016-0287. Endereço: http://dx.doi.org/10.1590/1806-9126-rbef-2016-0287. Acesso em: 7 jun. 2019.

SOUZA FILHO, Moacir Pereira de *et al.* TENDÊNCIAS DA PESQUISA EM ENSINO DE FÍSICA EM PUBLICAÇÕES E EVENTOS RECENTES. **V Encontro Nacional de pesquisa em Educação em Ciências**, 2005.

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

UNITED NATIONS EDUCATIONAL SCIENTIFIC AND CULTURAL ORGANIZATION. **2012 PARIS OER DECLARATION**. [*s.l.*: s.n.], 2012. Endereço: http://www.unesco.org/new/fileadmin/MULTIMEDIA/HQ/CI/WPFD2009/English_Declaration.html. Acesso em: 7 jun. 2019.

UNITED NATIONS EDUCATIONAL SCIENTIFIC AND CULTURAL ORGANIZATION. **Open Educational Resources (OER)**. [*s.l.*: s.n.], 2019. Endereço: https://en.unesco.org/themes/building-knowledge-societies/oer. Acesso em: 7 jun. 2019.

APÊNDICES

APÊNDICE A – PROGRAMA PYTHON

O codigo principal ainda esta sendo escrita.

A.1 KAKI

Este código busca os Arduínos conectados no computador e lista-os, retornando o dispositivo selecionado.

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Fri Aug 23 12:15:04 2019
4
5  @author: Daniel
6  """
7  import queue
8  import threading
9  from operator import methodcaller
10 import serial
11 from serial.tools import list_ports
12 import time
13
14
15 class SerialConnection:
16     BaudRates = [9600]  # , 14400, 19200, 28800, 38400, 57600, 115200]
17     CommTest = "sKIAjfrcfvmo2CbzOsogvkR9dm7ZOaSN"
18     Device = ""
19     EMsg = ""
20     Conn = False
21     PORTS = []
22     BAUDS = []
23     ID = []
24     EXCLUDE = []
25     SELECTION = ''
26
27     def __init__(self, **kwargs):
28         for name, value in kwargs.items():
29             if name.upper() == "EXCLUDE":
30                 if isinstance(value, list):
31                     for i in value:
32                         self.EXCLUDE.append(i)
```

```
33
34          self.Find()
35          if len(self.PORTS) > 0:
36              self.Select()
37
38      def WatchPort(self, q, error):
39          time.sleep(1)
40          data = []
41          while True:
42              while self.Device.in_waiting:
43                  try:
44
                        ↪  data.append(self.Device.readline().decode('utf-8').rstrip("\n\r"))
45                      data.append(time.time())
46                      q.put(data)
47                      data = []
48                  except Exception as error:
49                      self.Conn = False
50                      self.HandleError(error)
51
52      def HandleError(self, error):
53          # self.Error = True
54          # print("{}. line:{}".format(error, error.__traceback__.tb_lineno))
55          self.EMsg = "{}. line:{}".format(error, error.__traceback__.tb_lineno)
56          # return  # self.EMsg
57          # print("{}. line:{}".format(error, error.__traceback__.tb_lineno))
58
59      def SendData(self, cmd):
60          set_temp1 = 'Dani' + str(cmd) + '\r\n'
61          self.Device.write(set_temp1.encode())
62          self.Device.flushInput()
63
64      def ReadData(self):
65          data = []
66          while not self.Device.in_waiting:
67              pass
68          while self.Device.in_waiting:
69              try:
70                  d = self.Device.readline().decode('utf-8').rstrip("\n\r")
71                  d = [x.strip() for x in d.split("\t")]
72                  for i in d:
73                      data.append(i)
```

```
74                  data.append(time.time())
75              except Exception as error:
76                  self.HandleError(error)
77          return data
78
79      def SendRead(self, d):
80          data = []
81          while self.Device.in_waiting == 0:
82              self.SendData(d)
83              time.sleep(0.5)
84          while self.Device.in_waiting:
85              try:
86                  d = self.Device.readline().decode('utf-8').rstrip("\n\r")
87                  d = [x.strip() for x in d.split("\t")]
88                  for i in d:
89                      data.append(i)
90                  data.append(time.time())
91              except Exception as error:
92                  self.HandleError(error)
93          return data
94
95      def Disconnect(self):
96          self.Device.close()
97          self.Conn = False
98
99      def Connect(self):
100         self.Device = serial.Serial(self.PORTS[self.SELECTION],
            ↪  baudrate=self.BAUDS[self.SELECTION], timeout=4)
101         self.Device.flushInput()
102         self.Device.flushOutput()
103         self.SendRead("0000")
104         self.Conn = True
105
106     def Find(self):
107
108         devices = [p.device for p in serial.tools.list_ports.comports()]
109         while len(devices) == 0:
110             print('please connect device')
111             if input("Try to connect? ").upper() == "Y":
112                 devices = [p.device for p in serial.tools.list_ports.comports()]
113             else:
114                 break
```

```
115
116         for i in list(set(devices) - set(self.EXCLUDE)):
117             self.FindBaud(i)
118
119     def FindBaud(self, COMPORT):
120         for i in self.BaudRates:
121             device = serial.Serial(COMPORT, baudrate=i, timeout=4)
122             self.Device = device
123             device.flushInput()
124             device.flushOutput()
125             for j in range(10):
126                 din = self.SendRead("0000")
127
128                 if len(din) != 0:
129                     if din[0] == self.CommTest:
130                         self.BAUDS.append(i)
131                         self.PORTS.append(COMPORT)
132                         self.ID.append(self.SendRead("0001")[0])
133                         self.Disconnect()
134                         return
135             self.Disconnect()
136
137         pass
138
139     def Select(self):
140         print("Select Device")
141         for i in range(len(self.PORTS)):
142             print("{} - {}, {}".format(i, self.ID[i], self.PORTS[i]))
143         print("C to cancel")
144         usr = input("Number: ").upper()
145
146         try:
147             if usr == "C":
148                 return
149             else:
150                 if int(usr) > len(self.PORTS):
151                     raise ValueError
152                 else:
153                     self.SELECTION = int(usr)
154                     self.EXCLUDE.append(self.PORTS[self.SELECTION])
155                     self.Connect()
156         except ValueError:
```

```
157             print("Invalid Option!")
158             self.Select()
159
160
161 class Glue:
162     board = {}
163     Devices = []  # devices with established Connection
164     Names = []  # Name devices report to have.
165     Threads = []  # Threads connected to the devices
166     q = []
167     errorq = []
168
169     def __init__(self):
170         self.Connect()
171
172     def ls(self):
173         return list(self.board.keys())
174
175     def Connect(self):
176         self.Devices.append(SerialConnection())
177         self.Names.append(self.Devices[-1].ID)
178         self.board[input("What do you want to name the board? ").upper()] =
        ↪  len(self.Devices)-1
179         self.q.append(queue.Queue())
180         self.errorq.append(queue.Queue())
181         self.Threads.append(threading.Thread(target=methodcaller("WatchPort",
        ↪  self.q[-1], self.errorq[-1]),
182                                              args=(self.Devices[-1],)))
183         self.Threads[-1].start()
184
185     def Send(self, Board_Name, com):
186         try:
187             self.Devices[self.board[Board_Name.upper()]].SendData(com)
188         except:
189             self.Devices[self.board[Board_Name.upper()]].Conn = False
190             print("Device {} not connected".format(Board_Name))
191
192     def Read(self, Board_Name):
193         try:
194             return self.q[self.board[Board_Name.upper()]].get(timeout=1)
195         except queue.Empty:
196             return False
```

A.2 LISTA DE COMANDOS PARA CONTROLAR O AD7794

Este código contem os comandos principais do modulo Multímetro/DAQ

```
COMTEST = "0000"
ID = "0001"
ICID = "0002"
NAN = "0003"
TEMP = "0004"
SETGAIN = "0005"
READCHANNEL = "0006"
TOGGLEIO = "0007"
CTRLIO = "0008"
READIO = "0009"
SPS = "00010"

CH0 = "00"
CH1 = "01"
CH2 = "02"
CH3 = "03"
CH4 = "04"
CH5 = "05"
CH6 = "06"
CHANNELS = [CH0, CH1, CH2, CH3, CH4, CH5, CH6]
G1 = "00"
G2 = "01"
G4 = "02"
G8 = "03"
G16 = "04"
G32 = "05"
G64 = "06"
G128 = "07"
GAINS = [G1, G2, G4, G8, G16, G32, G64, G128]
```

A.3 DATA LOGGER

Este código contem as classe responsável por salvar os dados

```
# -*- coding: utf-8 -*-
"""
Created on Fri Aug 23 12:15:04 2019

```

```
5   @author: Daniel
6   """
7   try:
8       from openpyxl import Workbook, load_workbook
9       XL = True
10  except ImportError:
11      XL = False
12
13  try:
14      import csv
15      CVS = True
16  except ImportError:
17      CVS = False
18
19
20  class DataLogCVS:
21      def __init__(self):
22          if not CVS:
23              print("CVS not installed")
24              input("press enter to continue")
25
26      @staticmethod
27      def Create(FileName, *args):
28          fileName = FileName + ".cvs"
29          data = []
30          for index in range(len(args)):
31              data.append(args[index])
32          with open(fileName, mode='w') as a:
33              b = csv.writer(a, delimiter="\t", quotechar='"',
                ↪  quoting=csv.QUOTE_MINIMAL)
34              b.writerow(data)
35
36      @staticmethod
37      def Save(FileName, *args):
38          fileName = FileName + ".cvs"
39          data = []
40          for index in range(len(args)):
41              data.append(args[index])
42          for index in range(len(args)):
43              with open(fileName, mode='a') as a:
44                  b = csv.writer(a, delimiter="\t", quotechar='"',
                    ↪  quoting=csv.QUOTE_MINIMAL)
```

```
45                  b.writerow(data)
46
47
48  class DataLogExcel:
49      def __init__(self):
50          if not XL:
51              print("openpyxl not installed")
52              input("press enter to continue")
53
54      @staticmethod
55      def Save(FileName, SheetName, *args, **kwargs):
56          FileName = FileName + ".xlsx"
57          try:
58              wb = load_workbook(FileName)
59          except FileNotFoundError:
60              wb = Workbook()
61          if SheetName not in wb.sheetnames:
62              print("Creating new Sheet")
63              wb.create_sheet(SheetName)
64
65          sheet = wb[SheetName]
66
67          r = sheet.max_row + 1
68          for i in range(len((args))):
69              sheet.cell(row=r, column=i + 1).value = args[i]
70          # Save the file
71          wb.save(FileName)
```

APÊNDICE B – BIBLIOTECA ARDUINO PARA O AD7794

B.1 HEADER

Biblioteca Arduino responsavel por controlar o AD7794.

```
1  #ifndef AD779X_H
2  #define AD779X_H
3
4  #include <Arduino.h>
5  #include <stdarg.h>
6
7  #pragma region communication register
8
9  // communication register options
10 #define WEN      0B0 << 7 // --1 bit
11 #define R       0B1 << 6 // --1 bit
12 #define W       0B0 << 6 // --1 bit
13 #define CREAD    0B1 << 2 // --1 bit
14 #define SREAD    0B0 << 2 // --1 bit
15
16 // communication register Selection
17 #define COM      0x0 << 3 // --8  bits  W
18 #define STATUS    0x0 << 3 // --8  bits  R
19 #define MODE    0x1 << 3 // --16 bits  R/W
20 #define CONFIG    0x2 << 3 // --16 bits  R/W
21 #define DATA    0x3 << 3 // --16/24 bits  R
22 #define ID      0x4 << 3 // --8  bits  R
23 #define IO      0x5 << 3 // --8  bits  R/W
24 #define OFFSET    0x6 << 3 // --16 bits  R/W
25 #define FULL_SCALE  0x7 << 3 // --16 bits  R/W
26
27 #pragma endregion communication register
28
29 #pragma region Status Register
30 #define RDY 0 // LOW IF DATA READY
31 #define ADDRDY 7 //
32 #define ERR 1 // HIGH IF ERROR: OVERRANGE, UNDERRANGE
33 #define AD92 1 // IS 92
34 #define AD93 0 // IS 93
35
```

```
36  #pragma endregion Status Register
37
38  // Operation Options
39  #pragma region Mode Register
40  #pragma region Operating Modes
41  #define Cont 0x0 << 13
42  #define Single 0x1 << 13
43  #define Idle   0x2 << 13
44  #define PowerDown 0x3 << 13
45  #define IntZeroScaleCal 0x4 << 13
46  #define IntFullScaleCal 0x5 << 13
47  #define SysZeroScaleCal 0x6 << 13
48  #define SysFullScaleCal 0x7 << 13
49  #pragma endregion Operating Modes
50  #pragma region Power Switch Control Bit
51  #define PSWClose 0x1 << 11
52  #define PSWOpen  0x0 << 11
53  #pragma endregion Power Switch Control Bit
54  #pragma region Instrumentation Amplifier Common-Mode Bit
55  #define AMPCMSet 0x1 << 9
56  #define AMPCMClear 0x0 << 9
57  #pragma endregion Instrumentation Amplifier Common-Mode Bit
58  #pragma region ADC Clock Source
59  #define Int64 0x0 << 6
60  #define Int64AtCLKPin 0x1<< 6
61  #define Ext64 0x2 << 6
62  #define ExtCLK 0x3 << 6
63  #pragma endregion ADC Clock Source
64  #pragma region CHOP Bit
65  #define CHOPEnable 0x0 << 4
66  #define CHOPDisable 0x1 << 4
67  #pragma endregion CHOP Bit
68  #pragma region Filter Update Rate
69  #define F470    0x0 << 0
70  #define F242    0x1 << 0
71  #define F123    0x2 << 0
72  #define F62       0x3 << 0
73  #define F50      0x4 << 0
74  #define F39      0x5 << 0
75  #define F33      0x6 << 0
76  #define F19      0x7 << 0
77  #define F16_80db  0x8 << 0
```

```
78  #define F16_65db  0x9 << 0
79  #define F12       0xA << 0
80  #define F10       0xB << 0
81  #define F8       0xC << 0
82  #define F6       0xD << 0
83  #define F4       0xE << 0
84  #pragma endregion Filter Update Rate
85  #pragma endregion Mode Register
86
87  #pragma region CONFIGURATION REGISTER
88  #pragma region Bias Voltage Generator bit
89  #define VBIASDisable 0x0 << 14
90  #define VBIASA1      0x1 << 14
91  #define VBIASA2      0x2 << 14
92  #define VBIASA3      0x3 << 14
93  #pragma endregion Bias Voltage Generator bit
94  #pragma region Burnout Current Bit
95  #define BOEnable  0x1 << 13
96  #define BODisable 0x0 << 13
97  #pragma endregion Burnout Current Bit
98  #pragma region Unipolar/Bipolar Bit
99  #define Uipolar 0x1 << 12
100 #define Bipolar 0x0 << 12
101 #pragma endregion Unipolar/Bipolar Bit
102 #pragma region Boost Bit
103 #define BoostOn  0x1 << 11
104 #define BoostOff 0x0 << 11
105 #pragma endregion Boost Bit
106 #pragma region Gain Select
107 #define G1   0x0 << 8
108 #define G2   0x1 << 8
109 #define G4   0x2 << 8
110 #define G8   0x3 << 8
111 #define G16  0x4 << 8
112 #define G32  0x5 << 8
113 #define G64  0x6 << 8
114 #define G128 0x7 << 8
115 #pragma endregion Gain Select
116 #pragma region Reference Select
117 #define REFSEL1       0x0 << 6
118 #define REFSEL2       0x1 << 6
119 #define REFSELInternal 0x2 << 6
```

```
120 #define REFSELReserved 0x3 << 6
121 #pragma endregion Reference Select
122 #pragma region Reference Detect Function Bit
123 #define Ref_DetOff 0x0 << 5
124 #define Ref_DetOn 0x1 << 5
125 #pragma endregion Reference Detect Function Bit
126 #pragma region buffer mode operation Bit
127 #define BUFOff 0x0 << 4
128 #define BUFOn  0x1 << 4
129 #pragma endregion buffer mode operation Bit
130 #pragma region Channel Select
131 #define CH1 0x0
132 #define CH2 0x1
133 #define CH3 0x2
134 #define CH4 0x3
135 #define CH5 0x4
136 #define CH6 0x5
137 #define TempSensor 0x6
138 #define VddMonitor 0x7
139 #define CH1Neg 0x8
140 #pragma endregion Channel Select
141 #pragma endregion CONFIGURATION REGISTER
142
143 #pragma region IO Register
144 #pragma region IO Enable Bit
145 #define IODis 0x0 << 6
146 #define IOEn 0x1 << 6
147 #pragma endregion IO Enable Bit
148 #pragma region IO Pin State (High/Low)
149 #define IOP2L_P1L  0B00 << 4
150 #define IOP2L_P1H  0B01 << 4
151 #define IOP2H_P1L  0B10 << 4
152 #define IOP2H_P1H  0B11 << 4
153 #pragma endregion IO Pin State (High/Low)
154 #pragma region Direction of Current Sources
155 #define IEXCDIR_11_22  0x0 << 2
156 #define IEXCDIR_12_21  0x1 << 2
157 #define IEXCDIR_11_21  0x2 << 2
158 #define IEXCDIR_12_22  0x3 << 2
159 #pragma endregion Direction of Current Sources
160 #pragma region Value of Excitation Current
161 #define IEXCDisable 0x0
```

```
162 #define IEXC10uA  0x1
163 #define IEXC210uA  0x2
164 #define IEXC1mA    0x3
165 #pragma endregion Value of Excitation Current
166 #pragma endregion IO Register
167
168 class AD779X
169 {
170   public:
171   //AD779X(int ChipSelect, int MISOPin);
172   int CS;
173   int MISO;
174   int ContRead = false;
175   int Gain = 128;
176
177   void SetSPI(int ChipSelect, int MISOPin);
178   void Reset(void);
179   void SetContinuousRead();
180   void SPIWrite(byte x);
181   byte SPIRead(byte x);
182   void Configure(int ComReg, int value);  // Configure register
183
184   float Temperature(void); // configures to read once and returns conversion
185   byte GetID(void); // returns ID register
186   int32_t GetOffSetRegister(void); // returns offset register value
187   int32_t GetFullScaleRegister(void); // returns full scale register
188   int32_t ContinuousRead(void); // reads data without using Chip select pin
189   int32_t SingleRead(void); // reads data using Chip select pin
190   double ConvertToVolts(int32_t data);
191   double Volts( int Gain, int channel, int cont);
192   protected:
193   private:
194 };
195
196 #endif
```

B.2 CPP

Declaração da classe do AD7794

```
1 // add all functions here
2 #include <SPI.h>
```

```
3  #include <Arduino.h>
4  #include <stdarg.h>
5  #include <AD779X.h>
6
7
8  #define NOPSingle 0xFF
9  #define NOPCont    0x00
10
11
12 void AD779X::SetSPI(int ChipSelect, int MISOPin)
13 {
14   AD779X::CS = ChipSelect;
15   AD779X::MISO = MISOPin;
16   pinMode(AD779X::CS, OUTPUT);
17   digitalWrite(AD779X::CS, HIGH);
18   pinMode(AD779X::MISO, INPUT);
19   SPI.beginTransaction(SPISettings(500000, MSBFIRST, SPI_MODE0)); // SPI setup
     ↪  SPI communication
20   delay(200);           // SPI delay for initialization
21   SPI.begin();          // SPI Module 'wake up'
22 }
23
24 void AD779X::Reset(void)
25 {
26   SPIWrite(0xFF);
27   SPIWrite(0xFF);
28   SPIWrite(0xFF);
29   SPIWrite(0xFF);
30   delay(10);
31 }
32
33 void AD779X::SetContinuousRead(void)
34 {
35   digitalWrite(AD779X::CS, 0);
36   SPIWrite(0x5C);
37 }
38
39 void AD779X::SPIWrite(byte x)
40 {
41   SPI.transfer(x);
42 }
43
```

```
44 byte AD779X::SPIRead(byte x)
45 {
46   return SPI.transfer(x);
47 }
48
49 void AD779X::Configure(int ComReg, int value)
50 {
51   AD779X::ContRead = ((ComReg) & 0b100) >> 2;
52   if (ComReg = CONFIG)
53   {
54     AD779X::Gain = pow(2,(value & (0x7 << 8)) >> 8);
55   }
56   switch (ComReg)
57   {
58     case IO:
59     break;
60     default:
61     SPIWrite(ComReg);
62     SPIWrite(value >> 8);
63     SPIWrite(value & 0xff );
64     break;
65   }
66 }
67
68 float AD779X::Temperature(void)
69 {
70   digitalWrite(AD779X::CS, 0);
71   Configure(MODE, Single | PSWOpen | AMPCMSet | Int64 | CHOPEnable | F4);
72   Configure(CONFIG, VBIASDisable | BODisable | Bipolar | BoostOff | G1
73   | REFSELInternal | Ref_DetOff | BUFOn | TempSensor);
74   float temp = (SingleRead() - 0x800000)*(1.17/8388608);
75   temp = temp/(0.81/1000)-273;
76   digitalWrite(AD779X::CS, 1);
77   return temp;
78 }
79
80 byte AD779X::GetID(void)
81 {
82   digitalWrite(AD779X::CS, 0);
83   SPIWrite(0b01100000);
84   byte reply = SPIRead(NOPSingle);
85   digitalWrite(AD779X::CS, 0);
```

```
86    return reply;
87  }
88
89  int32_t AD779X::GetOffSetRegister(void)
90  {
91    digitalWrite(8, 0);
92    SPIWrite(R | OFFSET | SREAD );
93    int32_t data;
94    for (int i=0; i<3; i++)
95    {
96      data |= SPIRead(0xFF);
97      if (i == 0 | i == 1)
98      { data <<= 8;}
99    }
100   digitalWrite(8, 1);
101   return data;
102 }
103
104 int32_t AD779X::GetFullScaleRegister(void)
105 {
106   digitalWrite(8, 0);
107   SPIWrite(R | FULL_SCALE | SREAD );
108   int32_t data;
109   for (int i=0; i<3; i++)
110   {
111     data |= SPIRead(0xFF);
112     if (i == 0 | i == 1)
113     { data <<= 8;}
114   }
115   digitalWrite(8, 1);
116   return data;
117 }
118
119 int32_t AD779X::ContinuousRead(void)
120 {
121   while(digitalRead(AD779X::MISO)==1){;;}
122   int32_t data;
123   for (int i=0; i<1; i++)
124   {
125     data |= SPIRead(NOPSingle);
126     if (i == 0 | i == 1)
127     { data <<= 8;}
```

```
128   }
129   //float conv = ( ((data - 0x800000)/ 0x800000)*(1.17/AD779X::Gain) );
130   return data;
131 }
132
133 int32_t AD779X::SingleRead(void)
134 {
135   while(digitalRead(AD779X::MISO)==1){;;}
136   SPIWrite(0x58);
137   int32_t data;
138   for (int i=0; i<3; i++)
139   {
140     data |= SPIRead(NOPSingle);
141     if (i == 0 | i == 1)
142     { data <<= 8;}
143   }
144   return data;
145 }
146
147 double AD779X::ConvertToVolts(int32_t data)
148 {
149   double v = ((double)(data-8388608)*1.17)/(8388608*AD779X::Gain);
150   return v;
151 }
152
153 double AD779X::Volts(int Gain, int channel, int cont)
154 {
155   digitalWrite(AD779X::CS, 0);
156   //Configure(MODE | SREAD, Cont | PSWOpen | AMPCMSet | Int64 | CHOPEnable |
      ↪  F4);
157   Configure(CONFIG, VBIASDisable | BODisable | Bipolar | BoostOff | Gain
158   | REFSELInternal | Ref_DetOff | BUFOn | channel );
159   //Configure(W | MODE | SREAD, Single | PSWOpen | AMPCMSet | Int64 | CHOPEnable
      ↪  | F4);
160   double V;
161
162   if (cont == SREAD)
163   {
164     V = ConvertToVolts(SingleRead());
165     digitalWrite(AD779X::CS, 1);
166   }
167   if (cont == CREAD)
```

```
168     {
169       Configure(MODE | SREAD, Cont | PSWOpen | AMPCMSet | Int64 | CHOPEnable |
          ↪ F4);
170       V = ConvertToVolts(ContinuousRead());
171     }
172     return V;
173   }
```

APÊNDICE C – BIBLIOTECA ARDUINO PARA AD5662

C.1 HEADER

Declaração da classe da biblioteca Arduino responsável por controlar o DAC AD5662.

```
1  #ifndef AD5662_H
2  #define AD5662_H
3
4
5  class AD5662
6  {
7    public:
8    /*****************************************************************************/
9    /* AD5662 Power Modes                                                        */
10   /*****************************************************************************/
11   int AD5662_NORM       = 0x0;
12   int AD5662_1K_2_GND   = 0x1 ;
13   int AD5662_100K_2_GND = 0x2;
14   int AD5662_TRISTATE   = 0x3;
15   long int AD5662_FULLSCALE  = 0xFFFF;
16
17   AD5662(int CS, float VoltageReference, int ComType){AD5662_CS = CS;
     ↪  AD5662_VoltageReference = VoltageReference; AD5662_COM = ComType;};
18   void ad5662_BitBang(int data, int clk){AD5662_DATA = data; AD5662_CLK = clk;
     ↪  AD5662_COM = 2;};
19   void ad5662_Init(void);
20   void ad5662_Write(int mode, unsigned long data);
21   void ad5662_SetVoltage(float vOut);
22   protected:
23   private:
24   int AD5662_COM = 1;
25   int AD5662_CS = 10;
26   int AD5662_DATA;
27   int AD5662_CLK;
28   float AD5662_VoltageReference = 2.5;
29
30 };
31
32 #endif
```

C.2 CPP

Funções da classe AD5662.

```
#include <SPI.h>
#include <Wire.h>

void AD5662::ad5662_Init(void)
{
  pinMode(AD5662::AD5662_CS, OUTPUT);
  digitalWrite(AD5662::AD5662_CS, HIGH);
  /* Setup SPI Interface */

  switch (AD5662_COM)
  {
    case 0:
    Wire.begin();
    break;
    case 1:
    SPI.beginTransaction(SPISettings(10000000, MSBFIRST, SPI_MODE2)); // SPI
    ↪  setup SPI communication
    SPI.begin();
    break;
    case 2:
    pinMode(AD5662::AD5662_DATA, OUTPUT);
    pinMode(AD5662::AD5662_CLK, OUTPUT);
    digitalWrite(AD5662::AD5662_DATA, LOW);
    digitalWrite(AD5662::AD5662_CLK, LOW);
    break;
    default:
    break;
  }

}

void AD5662::ad5662_Write(int mode, unsigned long data)
{
  unsigned char dataBuf[3]   = { mode,           // dataBuf[0] MSB
    (data & 0x0000FF00) >> 8,  // dataBuf[1] MID
  (data & 0x000000FF) >> 0}; // dataBuf[2] LSB

  digitalWrite(AD5662::AD5662_CS, LOW);
```

```
38
39    #pragma region Communication
40    switch (AD5662_COM)
41    {
42      case 0:
43      Wire.write(dataBuf[0]);
44      Wire.write(dataBuf[1]);
45      Wire.write(dataBuf[2]);
46      break;
47      case 1:
48      SPI.transfer(dataBuf[0]);
49      SPI.transfer(dataBuf[1]);
50      SPI.transfer(dataBuf[2]);
51      break;
52      case 2:
53      for (int i = 0; i<=2; i++)
54      {
55        for (int j=7; j>=0; j--)
56        {
57          digitalWrite(AD5662::AD5662_DATA, (dataBuf[i] >> j) & 1);
58          digitalWrite(AD5662::AD5662_CLK, HIGH);
59          digitalWrite(AD5662::AD5662_CLK, LOW);
60        }
61      }
62      break;
63      default:
64      break;
65    }
66    #pragma endregion Communication
67
68    digitalWrite(AD5662::AD5662_CS, HIGH);
69  }
70
71  void AD5662::ad5662_SetVoltage(float vOut)
72  {
73    int16_t dacCode = 0;
74    if (vOut <= AD5662::AD5662_VoltageReference)
75    {
76      dacCode  = (int16_t)((float)(vOut / AD5662::AD5662_VoltageReference) *
       ↪  AD5662::AD5662_FULLSCALE);
77      ad5662_Write(AD5662_NORM, dacCode);
78    }
```

```
79
80  }
```

APÊNDICE D – EXEMPLOS DO USO DA INTERFACE PYTHON

D.1 ENVIANDO COMANDO DE LEITURA PARA O MODULO MULTÍMETRO/DAQ

Conecta ao modulo multímetro e lé o canal 1 com ganho 8 faz uma media de 4 amostras e mostra para o usuário.

```
1  import Kaki
2  from AD7794Codes import *
3  # cria instancia e excluino a porta COM3
4  DAQ = Kaki.SerialConnection(EXCLUDE=['COM3'])
5  # Manda um comando, le um valor e mostra para o usuario
6  print(DAQ.SendRead(READCHANNEL + CH1 + G8 + "4"))
```

D.2 CONECTA-SE A 3 MÓDULOS MULTÍMETRO/DAQ

Conecta a tres modulos multímetros.

```
1  import Kaki
2  from AD7794Codes import *
3  ignore = []
4  DAQ1 = Kaki.SerialConnection(EXCLUDE=ignore)
5  ignore = DAQ1.EXCLUDE
6  DAQ2 = Kaki.SerialConnection(EXCLUDE=ignore)
7  ignore = DAQ2.EXCLUDE
8  DAQ3 = Kaki.SerialConnection(EXCLUDE=ignore)
```

D.3 CONECTA-SE AO DAC E AJUSTA A TENSÃO

Conecta ao DAC e manda ajustar uma tensão, ambos os DACs neste trabalho é compatível.

```
1  import Kaki
2  DAC = Kaki.SerialConnection(EXCLUDE=[])
3  print(DAC.SendData("0006"+"0.7500000"))
```

APÊNDICE E – PROGRAMA DO DAQ/MULTIMETRO

E.1 ARQUIVO INO

Arquivo ino do Arduino, é análogo ao Main de um programa C/C++. Esta Implementação esta configurada para funcionar com um LCD de 20x4 e mostra o valor lido em todos os canais.

```
1  // Visual Micro is in vMicro>General>Tutorial Mode
2  //
3  /*
4      Name:       MultimetroV2.ino
5      Created:  10/15/2019 5:04:54 PM
6      Author:     DESKTOP-03RESN7\danie
7  */
8
9  // Define User Types below here or use a .h file
10 //
11 #include "InterpMulti.h"
12 InterpMulti SIn;
13 #include <LiquidCrystal.h>
14 const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
15 LiquidCrystal Display(rs, en, d4, d5, d6, d7);
16 //#include "OLED_DISPLAY.h"
17 //OLED_DISPLAY Display;
18
19 // Define Function Prototypes that use User Types below here or use a .h file
20 //
21
22
23 // Define Functions below here or use other .ino or cpp files
24 //
25
26
27 // Define Variables below here or use other .ino or cpp files
28 //
29 String inputString = "";         // a String to hold incoming data
30 bool stringComplete = false;  // whether the string is complete
31 // For proper delays
32 unsigned long Time = millis();
33 // For Button
```

```
34  // rotary button
35  #define CLK_PIN  A2
36  #define DATA_PIN A3
37  #define BUTTON A4
38  static int state=0,counter=0;
39
40
41  // The setup() function runs once each time the micro-controller starts
42  void setup()
43  {
44  #pragma region Serial
45    //Initialize serial and wait for port to open:
46    Serial.begin(9600);
47    while (!Serial) {
48      ; // wait for serial port to connect. Needed for native USB
49    }
50
51    // reserve 200 bytes for the inputString:
52    inputString.reserve(200);
53    SIn.Setup(10, 12);
54    SIn.ChannelOn[0] = true;
55    SIn.ChannelOn[1] = true;
56    SIn.ChannelOn[2] = true;
57    SIn.ChannelOn[3] = true;
58    SIn.ChannelOn[4] = true;
59    //SIn.debug(true);
60  #pragma endregion Serial
61
62    Display.begin(20, 4);
63
64  #pragma region Rotary Button
65    pinMode(CLK_PIN, INPUT);
66    pinMode(DATA_PIN, INPUT);
67    pinMode(BUTTON, INPUT);
68    attachInterrupt(digitalPinToInterrupt(BUTTON), Select, FALLING );
69  #pragma endregion Rotary Button
70  }
71
72  // Add the main program code into the continuous loop() function
73  void loop()
74  {
75    Interpret();
```

```
76    if (millis() >= (Time+1000))
77    {
78      SIn.ReadActive();
79      UpdateScreen();
80      Time = millis();//
81    }
82  }
83
84  void UpdateScreen(){
85    Display.clear();
86    for (int i = 0; i<=sizeof(SIn.ChannelOn); i++)
87    {
88      if (SIn.ChannelOn[i])
89      {
90        Serial.print(i); Serial.println(SIn.Unit[i][0]);
91        double data = SIn.GainValueSPS[i][1];
92        int multiplier = 0;
93        while((data < 1 && data>-1 ) && multiplier<3){data = data*1000; multiplier
          ↪  += 1;}
94        if (data < 0)
95        {
96          if (i>=4){Display.setCursor(11, i-4);}
97          else     {Display.setCursor(0, i);}
98
99          Display.print(data ,6-multiplier*2);
100
101         if (i>=4){Display.setCursor(18, i-4); }
102         else     {Display.setCursor(8, i);}
103
104         Display.print(SIn.Unit[i][multiplier]);
105       }
106       else
107       {
108         if (i>=4){Display.setCursor(11, i-4);}
109         else     {Display.setCursor(0, i);}
110         Display.print("+");
111         Display.print(data ,6-multiplier*2);
112         if (i>=4){Display.setCursor(18, i-4); }
113         else     {Display.setCursor(8, i);}
114
115         Display.print(SIn.Unit[i][multiplier]);
116       }
```

```
117
118         SIn.ChannelRead[i] = false;
119       }
120     }
121   }
122
123
124   void Interpret(){
125     if (stringComplete) {
126       SIn.IN( inputString );
127       inputString = "";
128       stringComplete = false;
129     }
130   }
131
132   void serialEvent() {
133     while (Serial.available()) {
134       char inChar = (char)Serial.read();
135       inputString += inChar;
136       if (inChar == '\n') {
137         inputString.remove( inputString.length()-2,2 );// remove the newline
138         if (inputString.length()>0){stringComplete = true;} // make sure data is
          ↪  larger than 4 and effectivly delete data
139       }
140     }
141   }
142
143   void RButton(){
144     /*
145     state=(state<<1) | digitalRead(CLK_PIN) | 0xe000;
146
147     if (state==0xf000){
148       state=0x0000;
149       if(digitalRead(DATA_PIN)){
150         if (counter >= sizeof(SIn.ChannelRead)-1)
151         {
152           counter = 0;
153         }
154         else
155         {
156           counter++;
157         }
```

```
158         Display.UpdateScreen(counter);
159       }
160       else {
161         if (counter <= 0)
162         {
163           counter = sizeof(SIn.ChannelRead)-1;
164         }
165         else
166         {
167           counter--;
168         }
169         Display.UpdateScreen(counter);
170       }
171     }
172     */
173 }
174
175 void Select(){
176   SIn.ChannelOn[counter] = !SIn.ChannelOn[counter];
177 }
```

E.2 INTERPRETADOR

Classe que interpreta o *string* mandado ao Arduino pelo Python

```
1  #ifndef InterpMulti_H
2  #define InterpMulti_H
3
4
5  class InterpMulti
6  {
7    public:
8    char LastMessage[200] = "";
9    bool DB = false;
10   bool ADC_Error;
11   int ADC_ID = 0;
12   bool ChannelOn[5]= {false};
13   bool ChannelRead[5]= {false};
14   double GainValueSPS[5][3]={
15     {0,0,0},
16     {0,0,0},
17     {0,0,0},
```

```
18      {0,0,0},
19      {0,0,0}
20    };
21    String Unit[5][4]= {
22      {"V", "mV", "uV", "pV"},
23      {"V", "mV", "uV", "pV"},
24      {"V", "mV", "uV", "pV"},
25      {"A", "mA", "uA", "pA"},
26      {"Ohm", "mOhm", "uOhm", "pOhm"}
27    };
28    int CurrentGain = 0, CurrentChannel = 0;
29
30    //InterpMulti(int ChipSelect, int MISOPin);
31    void Setup(int ChipSelect, int MISOPin);
32    void IN(String SerialIn);
33    void debug (bool db);
34    void ReadActive();
35
36    protected:
37    private:
38    void ADCID(void);
39  };
40
41  #endif
42
43
44  // add all functions here
45  #include <SPI.h>
46  #include <Arduino.h>
47  #include <stdarg.h>
48  #include <AD779X.h>
49  AD779X XADC;
50
51  // Command Codes
52  #define COMTEST 0x0
53  #define GETID 0x1
54  #define ICID 0x2
55  #define TEMP 0x4
56  #define SETGAIN 0x5
57  #define READ 0x6
58
59  // Return Codes
```

```
60  #define InitCom "sKIAjfrcfumo2CbzOsogvkR9dm7ZOaSN"
61  #define id 0x7794  // Multimeter ID
62
63
64  void InterpMulti::Setup(int ChipSelect, int MISOPin)
65  {
66    XADC.SetSPI(10, 12);
67    XADC.Reset();
68    InterpMulti::ADC_ID = XADC.GetID();
69    if (InterpMulti::ADC_ID != 79)
70    {
71      InterpMulti::ADC_Error = true;
72      //Serial.println(InterpMulti::ADC_Error);
73      //Serial.println("didnt work");
74    }
75    else
76    {
77      InterpMulti::ADC_Error = false;
78      //Serial.println(InterpMulti::ADC_Error);
79    }
80  }
81
82  void InterpMulti::IN(String SerialIn)
83  {
84
85    String Command = SerialIn.substring(0,4); // find command number
86    String value = SerialIn.substring(4, SerialIn.length());
87    switch(Command.toInt()){
88
89  // works
90  #pragma region ComTest: 0x0
91      case COMTEST:
92      Serial.println(InitCom);
93      break;
94  #pragma endregion ComTest: 0x0
95
96  // works
97  #pragma region ID: 0X1
98      case GETID:
99      Serial.println(id,HEX);
100     break;
101 #pragma endregion ID: 0X1
```

```
102
103 // works
104 #pragma region ICID: 0X2
105
106     case ICID:
107       Serial.println(XADC.GetID());
108
109 #pragma endregion ICID: 0X2
110
111 // works, remove
112 #pragma region reprint Last message
113
114     case 0x3:
115       Serial.println(InterpMulti::LastMessage);
116     break;
117
118 #pragma endregion Toggle Pin 0x3
119
120 // works
121 #pragma region TEMP 0x4
122
123     case TEMP:
124       Serial.println(XADC.Temperature());
125     break;
126
127 #pragma endregion TEMP 0x4
128
129 // works
130 #pragma region SETGAIN 0x5
131     case SETGAIN:
132     {
133       GainValueSPS[SerialIn.substring(4, 6).toInt()][0] = (
        ↪  SerialIn.substring(6, SerialIn.length())).toInt();
134     }
135     break;
136 #pragma endregion SETGAIN 0x5
137
138
139 // working
140 #pragma region READCHANNEL 0x6
141     case READ:
142     {
```

```
143         InterpMulti::LastMessage[8] = {0};
144         int Gain = ((SerialIn.substring(6, SerialIn.length())).toInt());
145         int Chan = SerialIn.substring(4, 6).toInt();
146         //int integration = ((SerialIn.substring(8, SerialIn.length())).toInt());
147         int integration = 10;
148         InterpMulti::GainValueSPS[Chan][0] = Gain;
149         double V = 0;
150         for (int i = 0; i<= integration-1; i++)
151         {
152           V += (double)XADC.Volts(Gain << 8, Chan, 0, F4);
153         }
154
155         V = V/integration;
156
157
158         Serial.print(V,6);
159         Serial.print("\t");
160         Serial.print(millis());
161         Serial.print("\t");
162         Serial.println(int(pow(2,Gain)));
163         static char outstr[20];
164         dtostrf(V, 8, 6, outstr);
165         strcpy(InterpMulti::LastMessage, outstr);
166         strcpy(InterpMulti::LastMessage, "/t");
167         strcpy(InterpMulti::LastMessage, millis());
168         strcpy(InterpMulti::LastMessage, "\t");
169         strcpy(InterpMulti::LastMessage, int(pow(2,Gain)));
170
171         InterpMulti::GainValueSPS[Chan][1] = V;
172         InterpMulti::GainValueSPS[Chan][0] = Gain;
173         InterpMulti::ChannelRead[((SerialIn.substring(4,
          ↪  SerialIn.length())).toInt())] = true;
174         InterpMulti::ChannelOn[((SerialIn.substring(4,
          ↪  SerialIn.length())).toInt())] = true;
175       }
176       break;
177   #pragma endregion READCHANNEL 0x6
178
179   // works
180   #pragma region Toggle IO 0x7
181       case 0x7:
182       {
```

```
183         digitalWrite((SerialIn.substring(4, 6)).toInt(),
            ↪  !digitalRead((SerialIn.substring(4, 6)).toInt()));
184       }
185       break;
186   #pragma endregion Toggle IO 0x7
187
188   // works
189   #pragma region ctrl IO 0x8
190       case 0x8:
191       {
192         Serial.println(SerialIn.substring(6, SerialIn.length()).toInt());
193         digitalWrite((SerialIn.substring(4, 6)).toInt(),
194              (SerialIn.substring(6, SerialIn.length()).toInt())
195             );
196       }
197
198       break;
199   #pragma endregion ctrl IO 0x8
200
201   #pragma region READ LED 0x9
202       case 0x9:
203       Serial.println(digitalRead((SerialIn.substring(4, 6)).toInt()));
204       break;
205   #pragma endregion READ LED
206
207
208       default:
209       Serial.print(Command); Serial.println("Unknown Command");
210       break;
211     }
212     if (InterpMulti::DB){
213       Serial.println("-----------------------");
214       Serial.print("SerialIn: ");Serial.println(SerialIn);
215       Serial.print("command: ");Serial.println(Command.toInt());
216       Serial.print("Value: ");Serial.println(value.toInt());
217       Serial.println("-----------------------");
218     }
219     //InterpMulti::ReadActive();
220   }
221
222   void InterpMulti::debug(bool db)
223   {
```

```
224   InterpMulti::DB=db;
225
226 }
227
228 void InterpMulti::ReadActive()
229 {
230   int lim = sizeof(ChannelRead);
231   for (int i = 0; i < lim;i++)
232   {
233     if (!InterpMulti::ChannelRead[i] && InterpMulti::ChannelOn[i])
234     {
235       int Chan = i;
236       int Gain = InterpMulti::GainValueSPS[Chan][0];
237       double V = (double)XADC.Volts(Gain << 8, Chan, 0, F50);
238       /*
239       while (V >= 0xFFFFF)
240       {
241         Gain = Gain-1;
242         if (Gain>=0)
243         {
244           V = (double)XADC.Volts(Gain << 8, Chan, 0, F50);
245         }
246         else{Serial.println("Overflow"); break;}
247
248       }
249       */
250       InterpMulti::GainValueSPS[Chan][1] = V;
251       InterpMulti::GainValueSPS[Chan][0] = Gain;
252       InterpMulti::ChannelRead[i] = false;
253     }
254   }
255
256 }
257
258 void InterpMulti::ADCID(void)
259 {
260   /*
261   InterpMulti::ADC_ID = (int)XADC.GetID();
262   */
263 }
```

E.3 DISPLAY OLED

Classe e funções responsáveis por atualizar os dados na OLED

```
#ifndef OLED_DISPLAY_H
#define OLED_DISPLAY_H


class OLED_DISPLAY
{
  public:
    void SetUp(void);
    void UpdateScreen(int counter);

  protected:
  private:
    String Unit[7]= {
      "V",
      "V",
      "V",
      "A",
      "Ohm",
      "",
      "",
    };
};


#endif

#include <Wire.h>
#include "SSD1306AsciiWire.h"
SSD1306AsciiWire oled;

#define I2C_ADDRESS 0x3C
// Define proper RST_PIN if required.
#define RST_PIN -1

void OLED_DISPLAY::SetUp(void)
{
  Wire.begin();
  Wire.setClock(400000L);
```

```
39
40    #if RST_PIN >= 0
41    oled.begin(&Adafruit128x64, I2C_ADDRESS, RST_PIN);
42    #else // RST_PIN >= 0
43    oled.begin(&Adafruit128x64, I2C_ADDRESS);
44    #endif // RST_PIN >= 0
45
46    oled.setFont(Multi);
47  }
48
49  void OLED_DISPLAY::UpdateScreen(int counter)
50  {
51    //write values to screen
52    oled.clear();
53    for (int i=0; i < sizeof(SIn.ChannelRead)-1;i++)
54    {
55
56      if (counter == i)
57      {
58        oled.print("      "); oled.print(char(0x80 + i*2+1)); oled.print(" ");
59      }
60      else
61      {
62        oled.print("      "); oled.print(char(0x80 + i*2)); oled.print(" ");
63      }
64
65
66      if (SIn.ChannelRead[i] && SIn.ChannelOn[i] && (i!=4))
67      {
68        oled.print(SIn.GainValueSPS[i][1],6);oled.print(" ");
69        oled.print(SIn.Unit[i]);oled.print(" ");
70        SIn.ChannelRead[i] = false;
71      }
72
73      if (SIn.ChannelOn[i] == false)
74      {
75        for (int i = 0; i<=6;i++)
76        {
77          oled.print("--");
78        }
79      }
80
```

```
81      // for last value
82      if (i == 0)
83      {
84        if (counter == 4)
85        {
86          oled.print(char(0x80 + 9));
87        }
88        else
89        {
90          oled.print(char(0x80 + 8));
91        }
92
93        if (SIn.ChannelRead[i] && SIn.ChannelOn[i] && (i==4))
94        {
95          oled.print(SIn.GainValueSPS[i][1],6);oled.print(" ");
96          oled.print(SIn.Unit[i]);oled.print(" ");
97          SIn.ChannelRead[i] = false;
98        }
99      }
100     oled.println("");
101   }
102 }
```

E.4 FONT CUSTUMIZADO PARA A OLED

Aqui é declarado o Font que o trabalho utiliza para mostar os dados na OLED

```
1  /*
2   *
3   * new Font
4   *
5   * created with FontCreator
6   * written by F. Maximilian Thiele
7   *
8   * http://www.apetech.de/fontCreator
9   * me@apetech.de
10  *
11  * File Name           : Daniel.h
12  * Date                : 06.10.2019
13  * Font size in bytes  : 7912
14  * Font width          : 10
15  * Font height         : 13
```

```
16   * Font first char     : 32
17   * Font last char      : 138
18   * Font used chars     : 106
19   *
20   * The font data are defined as
21   *
22   * struct _FONT_ {
23   *     uint16_t   font_Size_in_Bytes_over_all_included_Size_it_self;
24   *     uint8_t    font_Width_in_Pixel_for_fixed_drawing;
25   *     uint8_t    font_Height_in_Pixel_for_all_characters;
26   *     unit8_t    font_First_Char;
27   *     uint8_t    font_Char_Count;
28   *
29   *     uint8_t    font_Char_Widths[font_Last_Char - font_First_Char +1];
30   *                  // for each character the separate width in pixels,
31   *                  // characters < 128 have an implicit virtual right empty row
32   *
33   *     uint8_t    font_data[];
34   *                  // bit field of all characters
35   */
36
37
38   GLCDFONTDECL(Multi) = {
39       0x1E, 0xE8, // size
40       0x0A, // width
41       0x0D, // height
42       0x20, // first char
43       0x6A, // char count
44
45       // char widths
46       0x00, 0x01, 0x03, 0x06, 0x05, 0x09, 0x09, 0x01,
47       0x03, 0x03, 0x05, 0x07, 0x02, 0x03, 0x01, 0x03,
48       0x05, 0x03, 0x05, 0x05, 0x06, 0x05, 0x05, 0x05,
49       0x05, 0x05, 0x01, 0x02, 0x06, 0x06, 0x06, 0x04,
50       0x0A, 0x08, 0x07, 0x07, 0x08, 0x07, 0x06, 0x08,
51       0x08, 0x03, 0x04, 0x08, 0x07, 0x0B, 0x08, 0x08,
52       0x06, 0x08, 0x08, 0x05, 0x07, 0x08, 0x07, 0x0B,
53       0x08, 0x08, 0x07, 0x03, 0x03, 0x02, 0x06, 0x06,
54       0x02, 0x05, 0x05, 0x04, 0x06, 0x04, 0x04, 0x05,
55       0x06, 0x03, 0x02, 0x06, 0x03, 0x09, 0x06, 0x05,
56       0x06, 0x06, 0x04, 0x04, 0x03, 0x06, 0x06, 0x09,
57       0x05, 0x07, 0x05, 0x03, 0x01, 0x03, 0x05, 0x06,
```

```
58      0x09, 0x09, 0x09, 0x09, 0x09, 0x09, 0x09, 0x09,
59      0x09, 0x09,
60
61      // font data
62      0xFE, 0x10, // 33
63      0x0E, 0x00, 0x0E, 0x00, 0x00, 0x00, // 34
64      0xD0, 0x78, 0x56, 0x50, 0xF8, 0x56, 0x18, 0x00, 0x00, 0x18, 0x00, 0x00, //
        ↪  35
65      0x1C, 0x12, 0xFF, 0x22, 0xC4, 0x08, 0x10, 0x38, 0x10, 0x08, // 36
66      0x0C, 0x12, 0x8C, 0x40, 0x30, 0x08, 0x86, 0x40, 0x80, 0x00, 0x10, 0x08,
        ↪  0x00, 0x00, 0x00, 0x08, 0x10, 0x08, // 37
67      0xC0, 0x20, 0x3C, 0xD2, 0x8A, 0x46, 0x30, 0x10, 0x00, 0x08, 0x10, 0x10,
        ↪  0x10, 0x08, 0x10, 0x10, 0x10, 0x08, // 38
68      0x0E, 0x00, // 39
69      0xF0, 0x0C, 0x02, 0x08, 0x30, 0x40, // 40
70      0x02, 0x0C, 0xF0, 0x40, 0x30, 0x08, // 41
71      0x14, 0x08, 0x3E, 0x08, 0x14, 0x00, 0x00, 0x00, 0x00, 0x00, // 42
72      0x20, 0x20, 0x20, 0xFC, 0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0x08, 0x00,
        ↪  0x00, 0x00, // 43
73      0x00, 0x00, 0x50, 0x30, // 44
74      0x80, 0x80, 0x80, 0x00, 0x00, 0x00, // 45
75      0x00, 0x10, // 46
76      0x00, 0xF8, 0x06, 0x18, 0x00, 0x00, // 47
77      0xFC, 0x02, 0x02, 0x02, 0xFC, 0x08, 0x10, 0x10, 0x10, 0x08, // 48
78      0x04, 0xFE, 0x00, 0x10, 0x18, 0x10, // 49
79      0x04, 0x02, 0x82, 0x42, 0x3C, 0x10, 0x18, 0x10, 0x10, 0x18, // 50
80      0x04, 0x02, 0x22, 0x32, 0xCC, 0x10, 0x10, 0x10, 0x10, 0x08, // 51
81      0xC0, 0xA0, 0x98, 0x84, 0xFE, 0x80, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, //
        ↪  52
82      0x00, 0x0C, 0x0A, 0x12, 0xE2, 0x10, 0x10, 0x10, 0x08, 0x00, // 53
83      0xF0, 0x28, 0x14, 0x12, 0xE2, 0x08, 0x10, 0x10, 0x10, 0x08, // 54
84      0x04, 0x02, 0x02, 0xF2, 0x0E, 0x00, 0x00, 0x18, 0x00, 0x00, // 55
85      0x8C, 0x52, 0x22, 0x52, 0x8C, 0x08, 0x10, 0x10, 0x10, 0x08, // 56
86      0x3C, 0x42, 0x42, 0xC2, 0x7C, 0x10, 0x10, 0x08, 0x00, 0x00, // 57
87      0x10, 0x10, // 58
88      0x10, 0x00, 0x50, 0x30, // 59
89      0x20, 0x50, 0x50, 0x50, 0x88, 0x88, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //
        ↪  60
90      0x50, 0x50, 0x50, 0x50, 0x50, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //
        ↪  61
91      0x88, 0x88, 0x50, 0x50, 0x50, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //
        ↪  62
```

```
92      0x0C, 0xC2, 0x22, 0x1C, 0x00, 0x10, 0x00, 0x00, // 63
93      0xF0, 0x08, 0xE4, 0x12, 0x0A, 0x0A, 0x92, 0x7A, 0x04, 0xF8, 0x18, 0x20,
        ↪  0x48, 0x90, 0x90, 0x88, 0x88, 0x90, 0x48, 0x20, // 64
94      0x00, 0x80, 0x78, 0x46, 0x58, 0x60, 0x80, 0x00, 0x10, 0x18, 0x10, 0x00,
        ↪  0x00, 0x10, 0x18, 0x10, // 65
95      0x02, 0xFE, 0x22, 0x22, 0x22, 0x3C, 0xC0, 0x10, 0x18, 0x10, 0x10, 0x10,
        ↪  0x10, 0x08, // 66
96      0xF8, 0x04, 0x02, 0x02, 0x02, 0x04, 0x8E, 0x00, 0x08, 0x10, 0x10, 0x10,
        ↪  0x08, 0x00, // 67
97      0x02, 0xFE, 0x02, 0x02, 0x02, 0x02, 0x04, 0xF8, 0x10, 0x18, 0x10, 0x10,
        ↪  0x10, 0x10, 0x08, 0x00, // 68
98      0x02, 0xFE, 0x22, 0x22, 0x72, 0x06, 0x00, 0x10, 0x18, 0x10, 0x10, 0x10,
        ↪  0x10, 0x08, // 69
99      0x02, 0xFE, 0x22, 0x22, 0x72, 0x06, 0x10, 0x18, 0x10, 0x00, 0x00, 0x00, //
        ↪  70
100     0xF8, 0x04, 0x02, 0x02, 0x02, 0x24, 0xEE, 0x20, 0x00, 0x08, 0x10, 0x10,
        ↪  0x10, 0x10, 0x08, 0x00, // 71
101     0x02, 0xFE, 0x22, 0x20, 0x20, 0x22, 0xFE, 0x02, 0x10, 0x18, 0x10, 0x00,
        ↪  0x00, 0x10, 0x18, 0x10, // 72
102     0x02, 0xFE, 0x02, 0x10, 0x18, 0x10, // 73
103     0x00, 0x02, 0xFE, 0x02, 0x18, 0x10, 0x08, 0x00, // 74
104     0x02, 0xFE, 0x22, 0x50, 0x88, 0x06, 0x02, 0x02, 0x10, 0x18, 0x10, 0x00,
        ↪  0x00, 0x18, 0x10, 0x10, // 75
105     0x02, 0xFE, 0x02, 0x00, 0x00, 0x00, 0x00, 0x10, 0x18, 0x10, 0x10, 0x10,
        ↪  0x10, 0x08, // 76
106     0x02, 0xFE, 0x06, 0x18, 0xE0, 0x00, 0xE0, 0x18, 0x06, 0xFE, 0x02, 0x10,
        ↪  0x18, 0x10, 0x00, 0x00, 0x18, 0x00, 0x00, 0x10, 0x18, 0x10, // 77
107     0x02, 0xFE, 0x08, 0x10, 0x60, 0x82, 0xFE, 0x02, 0x10, 0x18, 0x10, 0x00,
        ↪  0x00, 0x00, 0x18, 0x00, // 78
108     0xF8, 0x04, 0x02, 0x02, 0x02, 0x02, 0x04, 0xF8, 0x00, 0x08, 0x10, 0x10,
        ↪  0x10, 0x10, 0x08, 0x00, // 79
109     0x02, 0xFE, 0x22, 0x22, 0x22, 0x1C, 0x10, 0x18, 0x10, 0x00, 0x00, 0x00, //
        ↪  80
110     0xF8, 0x04, 0x02, 0x02, 0x02, 0x02, 0x04, 0xF8, 0x00, 0x08, 0x10, 0x10,
        ↪  0x30, 0x50, 0x48, 0x40, // 81
111     0x02, 0xFE, 0x22, 0x22, 0xE2, 0x1C, 0x00, 0x00, 0x10, 0x18, 0x10, 0x00,
        ↪  0x00, 0x08, 0x10, 0x10, // 82
112     0x0C, 0x12, 0x22, 0x22, 0xC6, 0x18, 0x10, 0x10, 0x10, 0x08, // 83
113     0x06, 0x02, 0x02, 0xFE, 0x02, 0x02, 0x06, 0x00, 0x00, 0x10, 0x18, 0x10,
        ↪  0x00, 0x00, // 84
114     0x02, 0xFE, 0x02, 0x00, 0x00, 0x02, 0xFE, 0x02, 0x00, 0x08, 0x10, 0x10,
        ↪  0x10, 0x10, 0x08, 0x00, // 85
```

```
115     0x02, 0x1E, 0xE2, 0x00, 0xE2, 0x1E, 0x02, 0x00, 0x00, 0x00, 0x18, 0x00,
        ↪  0x00, 0x00, // 86
116     0x02, 0x1E, 0xE2, 0x80, 0x62, 0x1E, 0xE2, 0x80, 0x72, 0x0E, 0x02, 0x00,
        ↪  0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, // 87
117     0x02, 0x06, 0x8A, 0x70, 0x70, 0x8A, 0x06, 0x02, 0x10, 0x18, 0x10, 0x00,
        ↪  0x00, 0x10, 0x18, 0x10, // 88
118     0x02, 0x06, 0x3A, 0xC0, 0x30, 0x0A, 0x06, 0x02, 0x00, 0x00, 0x10, 0x18,
        ↪  0x10, 0x00, 0x00, 0x00, // 89
119     0x00, 0x86, 0x42, 0x32, 0x0A, 0x06, 0x82, 0x10, 0x18, 0x10, 0x10, 0x10,
        ↪  0x10, 0x18, // 90
120     0xFE, 0x02, 0x02, 0x78, 0x40, 0x40, // 91
121     0x06, 0xF8, 0x00, 0x00, 0x00, 0x18, // 92
122     0x02, 0xFE, 0x40, 0x78, // 93
123     0x10, 0x0C, 0x02, 0x02, 0x0C, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //
        ↪  94
124     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, //
        ↪  95
125     0x04, 0x08, 0x00, 0x00, // 96
126     0xA0, 0x50, 0x50, 0xE0, 0x00, 0x08, 0x10, 0x10, 0x18, 0x10, // 97
127     0xFE, 0x20, 0x10, 0x10, 0xE0, 0x08, 0x10, 0x10, 0x10, 0x08, // 98
128     0xE0, 0x10, 0x10, 0x30, 0x08, 0x10, 0x10, 0x08, // 99
129     0xE0, 0x10, 0x10, 0x22, 0xFE, 0x00, 0x08, 0x10, 0x10, 0x08, 0x18, 0x10, //
        ↪  100
130     0xE0, 0x50, 0x50, 0x60, 0x08, 0x10, 0x10, 0x08, // 101
131     0x10, 0xFC, 0x12, 0x02, 0x10, 0x18, 0x10, 0x00, // 102
132     0x60, 0x90, 0x90, 0x70, 0x10, 0x68, 0x90, 0x90, 0x90, 0x60, // 103
133     0x02, 0xFE, 0x20, 0x10, 0xE0, 0x00, 0x10, 0x18, 0x10, 0x00, 0x18, 0x10, //
        ↪  104
134     0x10, 0xF2, 0x00, 0x10, 0x18, 0x10, // 105
135     0x10, 0xF2, 0x80, 0x78, // 106
136     0x02, 0xFE, 0x40, 0xB0, 0x10, 0x00, 0x10, 0x18, 0x10, 0x00, 0x18, 0x10, //
        ↪  107
137     0x02, 0xFE, 0x00, 0x10, 0x18, 0x10, // 108
138     0x10, 0xF0, 0x20, 0x10, 0xE0, 0x20, 0x10, 0xE0, 0x00, 0x10, 0x18, 0x10,
        ↪  0x00, 0x18, 0x10, 0x00, 0x18, 0x10, // 109
139     0x10, 0xF0, 0x20, 0x10, 0xE0, 0x00, 0x10, 0x18, 0x10, 0x00, 0x18, 0x10, //
        ↪  110
140     0xE0, 0x10, 0x10, 0x10, 0xE0, 0x08, 0x10, 0x10, 0x10, 0x08, // 111
141     0x10, 0xF0, 0x20, 0x10, 0x10, 0xE0, 0x80, 0xF8, 0x88, 0x10, 0x10, 0x08, //
        ↪  112
142     0xE0, 0x10, 0x10, 0x20, 0xF0, 0x00, 0x08, 0x10, 0x10, 0x88, 0xF8, 0x80, //
        ↪  113
```

```
143     0x10, 0xF0, 0x20, 0x10, 0x10, 0x18, 0x10, 0x00, // 114
144     0x60, 0x50, 0x90, 0xB0, 0x18, 0x10, 0x10, 0x08, // 115
145     0x10, 0xFC, 0x10, 0x00, 0x18, 0x10, // 116
146     0x10, 0xF0, 0x00, 0x10, 0xF0, 0x00, 0x00, 0x08, 0x10, 0x10, 0x18, 0x10, //
        ↪  117
147     0x30, 0xD0, 0x00, 0xC0, 0x30, 0x10, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, //
        ↪  118
148     0x10, 0xF0, 0x00, 0x90, 0xF0, 0x10, 0xC0, 0x30, 0x10, 0x00, 0x00, 0x18,
        ↪  0x00, 0x00, 0x18, 0x00, 0x00, 0x00, // 119
149     0x10, 0x30, 0xC0, 0x30, 0x10, 0x10, 0x18, 0x00, 0x18, 0x10, // 120
150     0x10, 0x70, 0x90, 0x00, 0xD0, 0x30, 0x10, 0x80, 0x80, 0x48, 0x30, 0x08,
        ↪  0x00, 0x00, // 121
151     0x30, 0x10, 0xD0, 0x30, 0x10, 0x10, 0x18, 0x10, 0x10, 0x18, // 122
152     0x40, 0xBC, 0x02, 0x00, 0x38, 0x40, // 123
153     0xFE, 0x78, // 124
154     0x02, 0xBC, 0x40, 0x40, 0x38, 0x00, // 125
155     0x80, 0x40, 0xC0, 0x80, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, // 126
156     0xFC, 0x04, 0x04, 0x04, 0x04, 0xFC, 0x18, 0x10, 0x10, 0x10, 0x10, 0x18, //
        ↪  127
157     0xFF, 0x01, 0xF9, 0x05, 0x05, 0x05, 0xF9, 0x01, 0xFF, 0xF8, 0x80, 0x98,
        ↪  0xA0, 0xA0, 0xA0, 0x98, 0x80, 0xF8, // 0x80  0
158     0xFF, 0xFF, 0x07, 0xFB, 0xFB, 0xFB, 0x07, 0xFF, 0xFF, 0xF8, 0xF8, 0xE0,
        ↪  0xD8, 0xD8, 0xD8, 0xE0, 0xF8, 0xF8, // 0x81  0 sel
159     0xFF, 0x01, 0x01, 0x09, 0xFD, 0x01, 0x01, 0x01, 0xFF, 0xF8, 0x80, 0x80,
        ↪  0xA0, 0xB8, 0xA0, 0x80, 0x80, 0xF8, // 82  1
160     0xFF, 0xFF, 0xFF, 0xF7, 0x03, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0xF8, 0xF8,
        ↪  0xD8, 0xC0, 0xD8, 0xF8, 0xF8, 0xF8, // 83  1 sel
161     0xFF, 0x01, 0x09, 0x05, 0x05, 0x85, 0x79, 0x01, 0xFF, 0xF8, 0x80, 0xA0,
        ↪  0xB0, 0xA8, 0xA0, 0xB0, 0x80, 0xF8, // 84  2
162     0xFF, 0xFF, 0xF7, 0xFB, 0xFB, 0x7B, 0x87, 0xFF, 0xFF, 0xF8, 0xF8, 0xD8,
        ↪  0xC8, 0xD0, 0xD8, 0xC8, 0xF8, 0xF8, // 85  2 sel
163     0xFF, 0x01, 0x09, 0x05, 0x45, 0x65, 0x99, 0x01, 0xFF, 0xF8, 0x80, 0x90,
        ↪  0xA0, 0xA0, 0xA0, 0x98, 0x80, 0xF8, // 86  3
164     0xFF, 0xFF, 0xF7, 0xFB, 0xBB, 0x9B, 0x67, 0xFF, 0xFF, 0xF8, 0xF8, 0xE8,
        ↪  0xD8, 0xD8, 0xD8, 0xE0, 0xF8, 0xF8, // 87  3sel
165     0xFF, 0x01, 0x81, 0x41, 0x31, 0x09, 0xFD, 0x01, 0xFF, 0xF8, 0x80, 0x88,
        ↪  0x88, 0x88, 0x88, 0xB8, 0x80, 0xF8, // 88  4
166     0xFF, 0xFF, 0x7F, 0xBF, 0xCF, 0xF7, 0x03, 0xFF, 0xFF, 0xF8, 0xF8, 0xF0,
        ↪  0xF0, 0xF0, 0xF0, 0xC0, 0xF8, 0xF8 // 89  4 sel
167
168 };
```

APÊNDICE F – PROGRAMA DO DAC/FONTE AD5662

F.1 ARQUIVO INO

Arquivo ino do Arduino, é análogo ao Main de um programa C/C++. Esta Implementação esta configurada para funcionar com um um oled de 128x32 e mostra o valor lido em todos os canais.

```
1  // Visual Micro is in vMicro>General>Tutorial Mode
2  //
3  /*
4      Name:       MultimetroV2.ino
5      Created:  10/15/2019 5:04:54 PM
6      Author:     DESKTOP-03RESN7\danie
7  */
8
9  // Define User Types below here or use a .h file
10 //
11 #include "InterpMulti.h"
12 InterpMulti SIn;
13 //#include <LiquidCrystal.h>
14 //const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
15 //LiquidCrystal Display(rs, en, d4, d5, d6, d7);
16 #include "OLED_DISPLAY.h"
17 OLED_DISPLAY Display;
18 #include <Encoder.h>
19 Encoder myEnc(6, 5);
20 // Define Function Prototypes that use User Types below here or use a .h file
21 //
22
23
24 // Define Functions below here or use other .ino or cpp files
25 //
26
27
28 // Define Variables below here or use other .ino or cpp files
29 //
30 String inputString = "";         // a String to hold incoming data
31 bool stringComplete = false;  // whether the string is complete
32 // For proper delays
33 unsigned long Time = millis();
```

```
34  long oldPosition  = -999; // FOR ROTARY
35
36
37
38  // The setup() function runs once each time the micro-controller starts
39  void setup()
40  {
41  #pragma region Serial
42    //Initialize serial and wait for port to open:
43    Serial.begin(9600);
44    while (!Serial) {
45      ; // wait for serial port to connect. Needed for native USB
46    }
47
48    // reserve 200 bytes for the inputString:
49    inputString.reserve(200);
50    SIn.Setup(10, 12, 2.5);
51    //SIn.debug(true);
52  #pragma endregion Serial
53
54    //analogReference(INTERNAL);
55    Display.SetUp();
56
57
58  }
59
60  // Add the main program code into the continuous loop() function
61  void loop()
62  {
63    Interpret();
64    if (millis() >= (Time+3000))
65    {
66      Time = millis();//
67      Display.UpdateScreen();
68    }
69    RButton();
70  }
71
72  void Interpret(){
73    if (stringComplete) {
74      SIn.IN( inputString );
75      inputString = "";
```

```
76      stringComplete = false;
77    }
78  }
79
80  void serialEvent() {
81    while (Serial.available()) {
82      char inChar = (char)Serial.read();
83      inputString += inChar;
84      if (inChar == '\n') {
85        if (inputString.indexOf('Dani') != -1)
86        {
87          inputString.remove( 0, inputString.indexOf('Dani')+1); // remove the
            ↪ string start
88          inputString.remove( inputString.length()-2,2 ); // remove the newline
89          inputString.remove( 12, inputString.length() ); // remove anything
            ↪ larger than 12 characters
90          if (inputString.length()>0){stringComplete = true;} // make sure data is
            ↪ larger than 4 and effectivly delete data
91        }
92        else{inputString = "";}
93      }
94    }
95  }
96
97  void RButton(){
98    long newPosition = myEnc.read()*100;
99    if (newPosition != oldPosition) {
100       SIn.Vout = (4.0912/65536)*oldPosition;
101       if (newPosition >= XDAC.AD5662_FULLSCALE)
102       {
103         oldPosition = XDAC.AD5662_FULLSCALE;
104       }
105       else if (SIn.Vout<0)
106       {
107         oldPosition = 0;
108       }
109       else{oldPosition = newPosition;}
110       XDAC.ad5662_Write(XDAC.AD5662_NORM, oldPosition);
111       Display.UpdateScreen();
112   }
113 }
```

F.2 INTERPRETADOR PARA AD5662

Declaração da Classe do AD5662 que interpreta o *string* mandado ao Arduino pelo Python.

```
#ifndef InterpMulti_H
#define InterpMulti_H


class InterpMulti
{
  public:
  double Vout = 0;
  char LastMessage[200] = "";
  bool DB = false;
  bool ADC_Error;

  //InterpMulti(int ChipSelect, int MISOPin);
  void Setup(int ChipSelect, int MISOPin, float Vref);
  void IN(String SerialIn);
  void debug (bool db);
  void ReadActive();

  protected:
  private:
  void ADCID(void);
};

#endif


#include <SPI.h>
#include <Arduino.h>
#include <stdarg.h>
#include "AD5662.h"
AD5662 XDAC(A3, 3.32, 2);

// Command Codes
#define COMTEST 0x0
#define GETID 0x1
#define ICID 0x2
#define TEMP 0x4
```

```
38 #define SETGAIN 0x5
39 #define READ 0x6
40
41 // Return Codes
42 #define InitCom "sKIAjfrcfumo2CbzOsogvkR9dm7ZOaSN"
43 #define id 0x5662  // Multimeter ID
44
45 void InterpMulti::Setup(int DATAPin, int CLKPin, float Vref)
46 {
47   XDAC.ad5662_BitBang(DATAPin, CLKPin, Vref);
48   XDAC.ad5662_Init();
49 }
50
51 void InterpMulti::IN(String SerialIn)
52 {
53   String Command = SerialIn.substring(0,4); // find command number
54   String value = SerialIn.substring(4, SerialIn.length());
55   switch(Command.toInt()){
56
57 // works
58 #pragma region ComTest: 0x0
59     case COMTEST:
60     {
61       Serial.println(InitCom);
62     }
63     break;
64 #pragma endregion  ComTest: 0x0
65
66 // works
67 #pragma region ID: 0X1
68     case GETID:
69     {
70       Serial.println(id,HEX);
71     }
72     break;
73 #pragma endregion ID: 0X1
74
75 // works
76 #pragma region ICID: 0X2
77
78 case 2:
79 Serial.println(0x5781, HEX);
```

```
80  break;
81
82  #pragma endregion ICID: 0X2
83
84  // works
85  #pragma region OFF: 0X3
86
87      case 3:
88        XDAC.ad5662_Write(XDAC.AD5662_100K_2_GND, 0);
89      break;
90
91  #pragma endregion OFF: 0X3
92
93  // works
94  #pragma region On: 0x4
95      case 0x4:
96        XDAC.ad5662_Write(XDAC.AD5662_NORM, InterpMulti::Vout);
97      break;
98  #pragma endregion On: 0x4
99
100 // works
101 #pragma region SetOutPut: 0X5
102     case 0x5:
103     InterpMulti::Vout = SerialIn.substring(4, SerialIn.length()).toFloat();
104     break;
105 #pragma endregion SetOutPut: 0X5
106
107 // works
108 #pragma region Output: 0X6
109     case 0x6:
110     {
111       InterpMulti::Vout = SerialIn.substring(4, SerialIn.length()).toFloat();
112       InterpMulti::Vout = -0.02082 + 0.99835 * InterpMulti::Vout;
113
114       int code = (int)((InterpMulti::Vout/4.0912)*65536);
115       if (InterpMulti::Vout>=4.06336)
116       {
117         code = XDAC.AD5662_FULLSCALE;
118       }
119       else if (InterpMulti::Vout<=0)
120       {
121         code = 0;
```

```
122         }
123         XDAC.ad5662_Write(XDAC.AD5662_NORM, code);
124       }
125       break;
126   #pragma endregion Output: 0X6
127
128   // works
129   #pragma region Toggle IO 0x7
130       case 0x7:
131       {
132         digitalWrite((SerialIn.substring(4, 6)).toInt(),
          ↪  !digitalRead((SerialIn.substring(4, 6)).toInt()));
133       }
134       break;
135   #pragma endregion Toggle IO 0x7
136
137   // works
138   #pragma region ctrl IO 0x8
139       case 0x8:
140       {
141         Serial.println(SerialIn.substring(6, SerialIn.length()).toInt());
142         digitalWrite((SerialIn.substring(4, 6)).toInt(),
143              (SerialIn.substring(6, SerialIn.length()).toInt())
144             );
145       }
146
147       break;
148   #pragma endregion ctrl IO 0x8
149
150   #pragma region READ IO 0x9
151       case 0x9:
152       Serial.println(digitalRead((SerialIn.substring(4, 6)).toInt()));
153       break;
154   #pragma endregion READ IO
155
156       default:
157       Serial.print(Command); Serial.println("Unknown Command");
158       break;
159
160     if (InterpMulti::DB){
161       Serial.println("------------------------");
162       Serial.print("SerialIn: ");Serial.println(SerialIn);
```

```
163       Serial.print("command: ");Serial.println(Command.toInt());
164       Serial.print("Value: ");Serial.println(value.toInt());
165       Serial.println("-----------------------");
166       }
167     }
168     //InterpMulti::ReadActive();
169   }
170   void InterpMulti::debug(bool db)
171   {
172     InterpMulti::DB=db;
173
174   }
175   void InterpMulti::ADCID(void)
176   {
177     /*
178     InterpMulti::ADC_ID = (int)XADC.GetID();
179     */
180   }
```

APÊNDICE G – PROGRAMA DO DAC/FONTE AD5781

G.1 ARQUIVO INO

Arquivo ino do Arduino, é análogo ao Main de um programa C/C++. Esta Implementação não esta configurada para funcionar com um Display.

```
1  // Visual Micro is in vMicro>General>Tutorial Mode
2  //
3  /*
4      Name:       MultimetroV2.ino
5      Created:  10/15/2019 5:04:54 PM
6      Author:     DESKTOP-03RESN7\danie
7  */
8
9  // Define User Types below here or use a .h file
10 //
11 #include "InterpMulti.h"
12 InterpMulti SIn;
13 //#include <LiquidCrystal.h>
14 //const int rs = 7, en = 6, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
15 //LiquidCrystal Display(rs, en, d4, d5, d6, d7);
16 #include "OLED_DISPLAY.h"
17 OLED_DISPLAY Display;
18 #include <Encoder.h>
19 Encoder myEnc(6, 5);
20 // Define Function Prototypes that use User Types below here or use a .h file
21 //
22
23
24 // Define Functions below here or use other .ino or cpp files
25 //
26
27
28 // Define Variables below here or use other .ino or cpp files
29 //
30 String inputString = "";         // a String to hold incoming data
31 bool stringComplete = false;  // whether the string is complete
32 // For proper delays
33 unsigned long Time = millis();
34 long oldPosition  = -999; // FOR ROTARY
```

```
35
36
37
38  // The setup() function runs once each time the micro-controller starts
39  void setup()
40  {
41  #pragma region Serial
42    //Initialize serial and wait for port to open:
43    Serial.begin(9600);
44    while (!Serial) {
45      ; // wait for serial port to connect. Needed for native USB
46    }
47
48    // reserve 200 bytes for the inputString:
49    inputString.reserve(200);
50    SIn.Setup(10, 12, 2.5);
51    //SIn.debug(true);
52  #pragma endregion Serial
53
54    //analogReference(INTERNAL);
55    Display.SetUp();
56
57
58  }
59
60  // Add the main program code into the continuous loop() function
61  void loop()
62  {
63    Interpret();
64    if (millis() >= (Time+3000))
65    {
66      Time = millis();//
67      Display.UpdateScreen();
68    }
69    RButton();
70  }
71
72  void Interpret(){
73    if (stringComplete) {
74      SIn.IN( inputString );
75      inputString = "";
76      stringComplete = false;
```

```
77    }
78  }
79
80  void serialEvent() {
81    while (Serial.available()) {
82      char inChar = (char)Serial.read();
83      inputString += inChar;
84      if (inChar == '\n') {
85        if (inputString.indexOf('Dani') != -1)
86        {
87          inputString.remove( 0, inputString.indexOf('Dani')+1); // remove the
            ↪ string start
88          inputString.remove( inputString.length()-2,2 ); // remove the newline
89          inputString.remove( 12, inputString.length() ); // remove anything
            ↪ larger than 12 characters
90          if (inputString.length()>0){stringComplete = true;} // make sure data is
            ↪ larger than 4 and effectivly delete data
91        }
92        else{inputString = "";}
93      }
94    }
95  }
96
97  void RButton(){
98    long newPosition = myEnc.read()*100;
99    if (newPosition != oldPosition) {
100       SIn.Vout = (4.0912/65536)*oldPosition;
101       if (newPosition >= XDAC.AD5662_FULLSCALE)
102       {
103         oldPosition = XDAC.AD5662_FULLSCALE;
104       }
105       else if (SIn.Vout<0)
106       {
107         oldPosition = 0;
108       }
109       else{oldPosition = newPosition;}
110       XDAC.ad5662_Write(XDAC.AD5662_NORM, oldPosition);
111       Display.UpdateScreen();
112   }
113 }
```

G.2 INTERPRETADOR PARA O AD5781

Declaração da Classe do AD5662 que interpreta o *string* mandado ao Arduino pelo Python.

```
#ifndef InterpMulti_H
#define InterpMulti_H


class InterpMulti
{
  public:
  int32_t Vout = 0;
  bool OutputOn = false;
  char LastMessage[200] = "";
  bool DB = false;
  bool ADC_Error;

  //InterpMulti(int ChipSelect, int MISOPin);
  void Setup();
  void IN(String SerialIn);
  void debug (bool db);
  void ReadActive();

  protected:
  private:
  void ADCID(void);
};

#endif


// add all functions here
#include <SPI.h>
#include <Arduino.h>
#include <stdarg.h>
#include "AD5781.h"
AD5781 XDAC(7, 8, 9);

// Command Codes
#define COMTEST 0x0
#define GETID 0x1
```

```
38  #define ICID 0x2
39  #define READ 0x6
40
41  // Return Codes
42  #define InitCom "sKIAjfrcfumo2CbzOsogvkR9dm7ZOaSN"
43  #define id 0x5781  // Multimeter ID
44  #define WRITE 0x100000
45
46  void InterpMulti::Setup()
47  {
48    //XDAC.AD5781_Write(1);
49    XDAC.AD5781_Init();
50  }
51
52  void InterpMulti::IN(String SerialIn)
53  {
54
55    String Command = SerialIn.substring(0,4); // find command number
56    String value = SerialIn.substring(4, SerialIn.length());
57    switch(Command.toInt()){
58
59  // works
60  #pragma region ComTest: 0x0
61      case COMTEST:
62      {
63        Serial.println(InitCom);
64      }
65      break;
66  #pragma endregion ComTest: 0x0
67
68  // works
69  #pragma region ID: 0X1
70      case GETID:
71      {
72        Serial.println(id,HEX);
73      }
74      break;
75  #pragma endregion ID: 0X1
76
77  // works
78  #pragma region ICID: 0X2
79
```

```
80      case ICID:
81        Serial.println(0x5781, HEX);
82      break;
83
84  #pragma endregion ICID: 0X2
85
86  // works
87  #pragma region Off: 0x3
88      case 0x3:
89        XDAC.AD5781_Write(0x200015);
90      break;
91  #pragma endregion Off: 0x3
92
93  // works
94  #pragma region SetOutput
95
96      case 0x4:
97        InterpMulti::Vout = 0x80000 + (SerialIn.substring(4,
          ↪  SerialIn.length()).toFloat() * 262143 / 5);
98      break;
99
100 #pragma endregion SetOutput
101
102 // works To turn on output
103 #pragma region OutputSetVoltage
104
105   case 0x5:
106   {
107     XDAC.AD5781_Write( WRITE | InterpMulti::Vout);
108     if (! InterpMulti::OutputOn)
109     {
110       XDAC.AD5781_Write(0x200010);
111     }
112   }
113   break;
114
115 #pragma endregion OutputSetVoltage
116
117 // works
118 #pragma region Output: 0X6
119     case 0x6:
120     {
```

```
121         float v = SerialIn.substring(4, SerialIn.length()).toFloat();
122         InterpMulti::Vout = 0x80000 + (v * 262143 / 5);
123         XDAC.AD5781_Write( WRITE | InterpMulti::Vout);
124         if ( !InterpMulti::OutputOn)
125         {
126           XDAC.AD5781_Write(0x200010);
127         }
128       }
129       break;
130
131   #pragma endregion Output: 0X6
132
133   // works
134   #pragma region Toggle IO 0x7
135       case 0x7:
136       {
137         digitalWrite((SerialIn.substring(4, 6)).toInt(),
          ↪  !digitalRead((SerialIn.substring(4, 6)).toInt()));
138       }
139       break;
140   #pragma endregion Toggle IO 0x7
141
142   // works
143   #pragma region ctrl IO 0x8
144       case 0x8:
145       {
146         Serial.println(SerialIn.substring(6, SerialIn.length()).toInt());
147         digitalWrite((SerialIn.substring(4, 6)).toInt(),
148               (SerialIn.substring(6, SerialIn.length()).toInt())
149             );
150       }
151
152       break;
153   #pragma endregion ctrl IO 0x8
154
155   #pragma region READ GPIB 0x9
156       case 0x9:
157       Serial.println(digitalRead((SerialIn.substring(4, 6)).toInt()));
158       break;
159   #pragma endregion READ GPIB
160
161
```

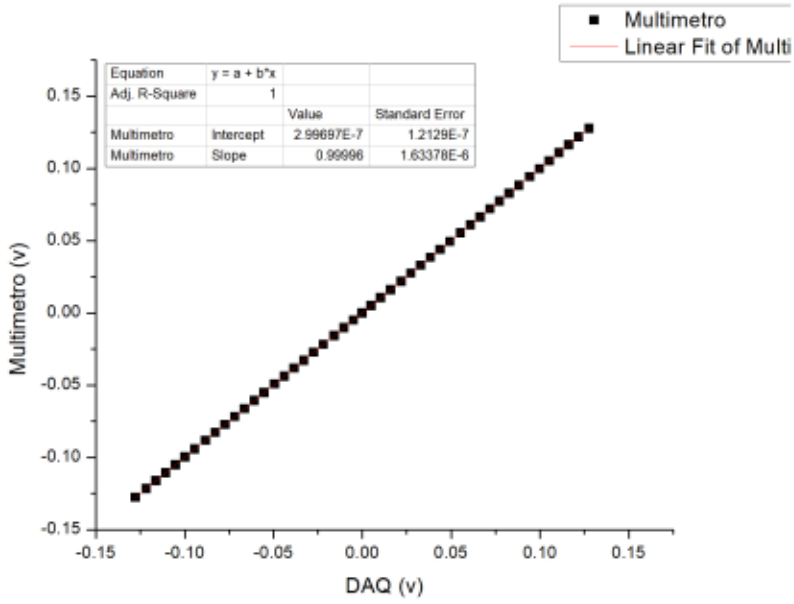
```
162       default:
163       Serial.print(Command); Serial.println("Unknown Command");
164       break;
165     }
166     if (InterpMulti::DB){
167       Serial.println("------------------------");
168       Serial.print("SerialIn: ");Serial.println(SerialIn);
169       Serial.print("command: ");Serial.println(Command.toInt());
170       Serial.print("Value: ");Serial.println(value.toInt());
171       Serial.println("------------------------");
172     }
173     //InterpMulti::ReadActive();
174   }
175
176   void InterpMulti::debug(bool db)
177   {
178     InterpMulti::DB=db;
179
180   }
181
182
183   void InterpMulti::ADCID(void)
184   {
185     /*
186     InterpMulti::ADC_ID = (int)XADC.GetID();
187     */
188   }
```

APÊNDICE H – GRÁFICOS DAS ENTRADAS DO AD7794

H.1 SEM ATENUADOR

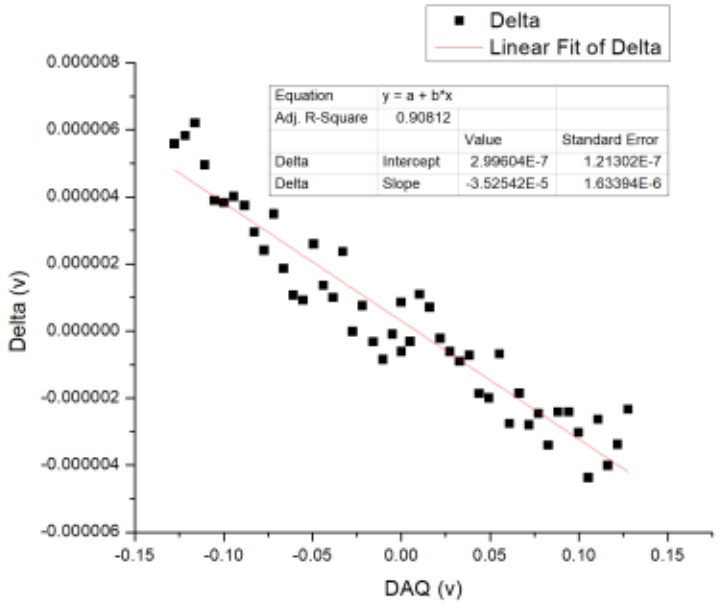
H.1.1 Ain 1

FIGURA 25 – CH0 - GANHO 1- AD7794 CONTRA REFERENCIA



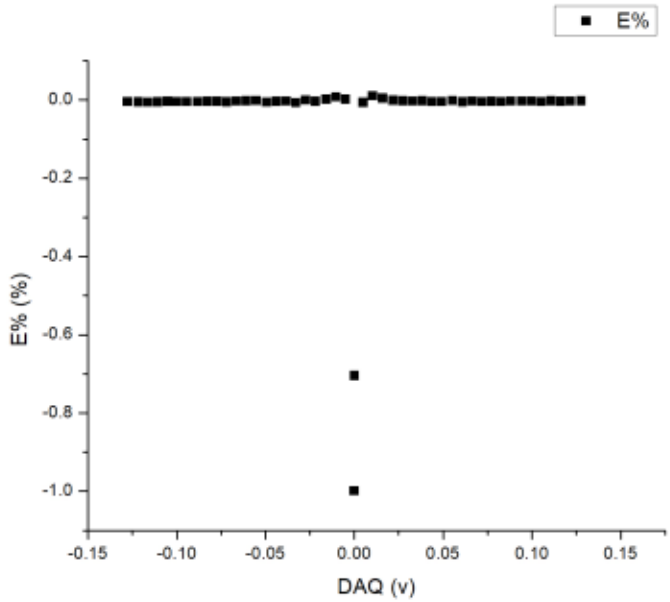
FONTE: Autor (2019)

FIGURA 26 – CH0 - GANHO 1- DIFERENÇA ENTRE AD7794 E REFERENCIA



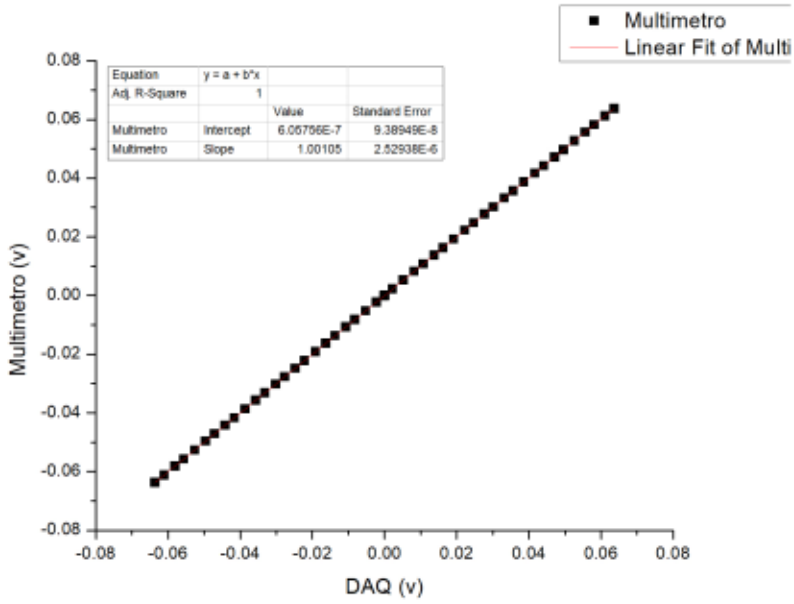
FONTE: Autor (2019)

FIGURA 27 – CH0 - GANHO 1- ERRO PERCENTUAL



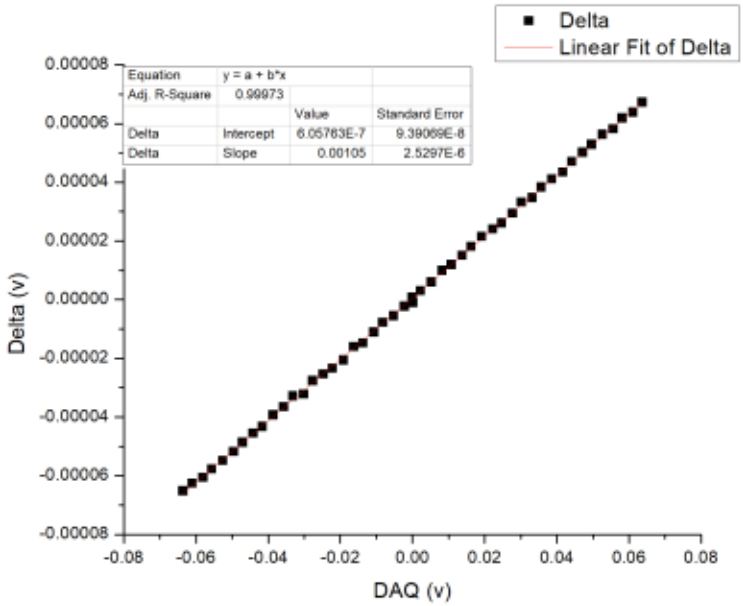
FONTE: Autor (2019)

FIGURA 28 – CH0 - GANHO 2- AD7794 CONTRA REFERENCIA



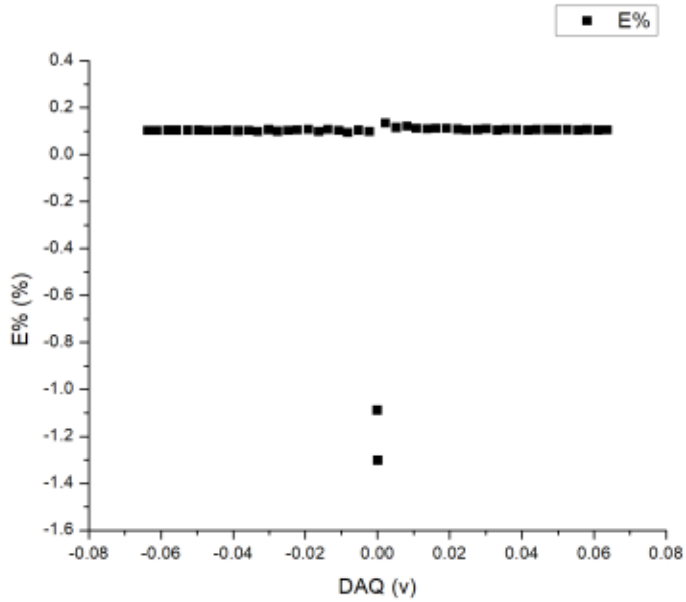
FONTE: Autor (2019)

FIGURA 29 – CH0 - GANHO 2- DIFERENÇA ENTRE AD7794 E REFERENCIA



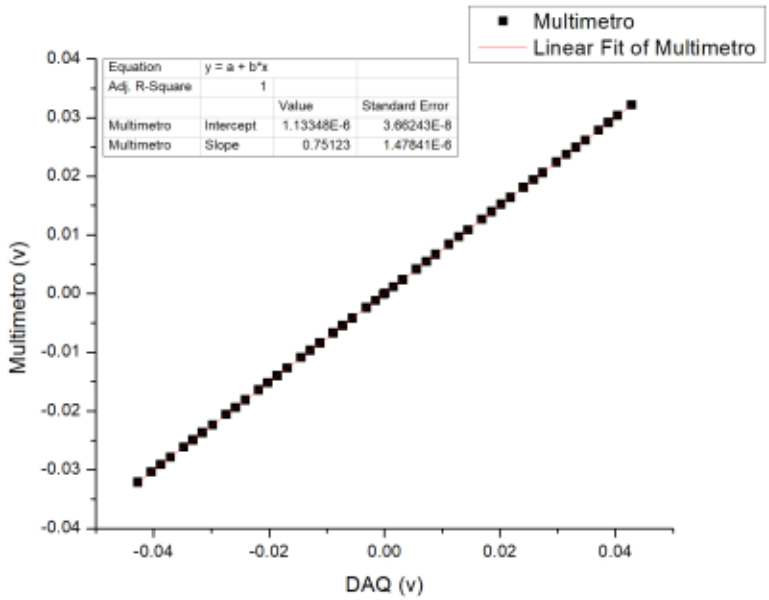
FONTE: Autor (2019)

FIGURA 30 – CH0 - GANHO 2- ERRO PERCENTUAL



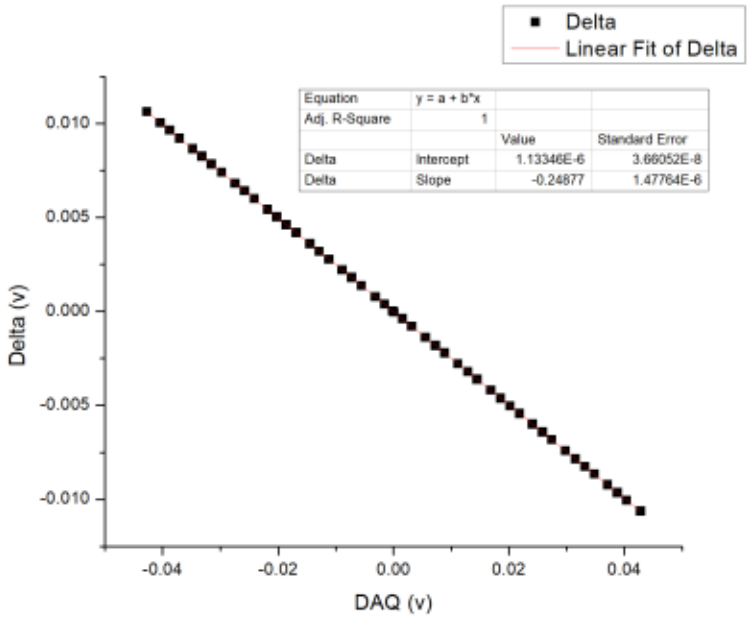
FONTE: Autor (2019)

FIGURA 31 – CH0 - GANHO 4- AD7794 CONTRA REFERENCIA



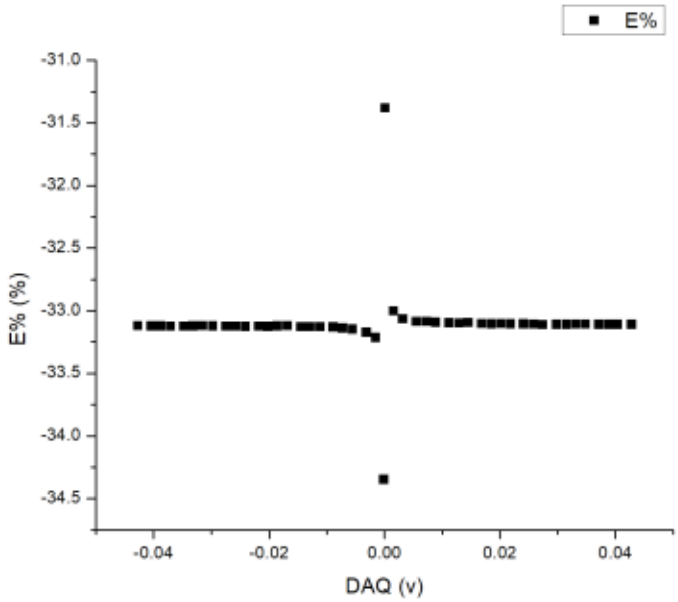
FONTE: Autor (2019)

FIGURA 32 – CH0 - GANHO 4- DIFERENÇA ENTRE AD7794 E REFERENCIA



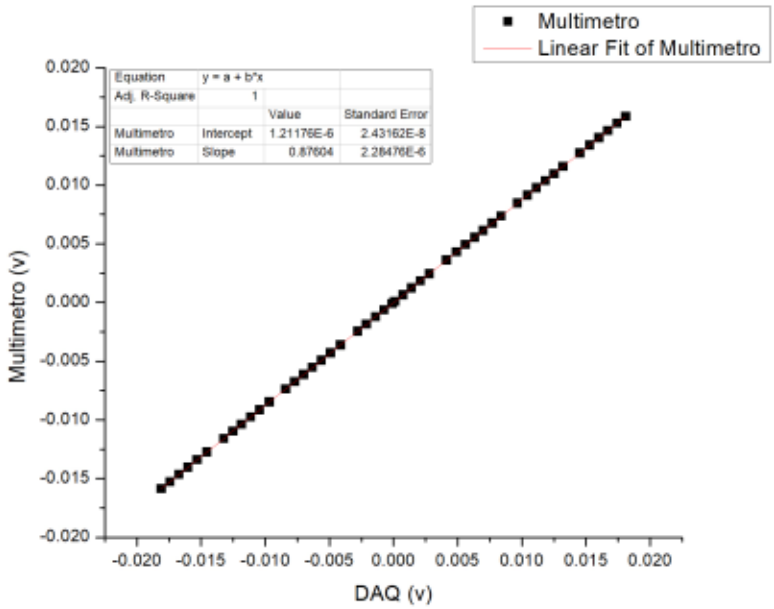
FONTE: Autor (2019)

FIGURA 33 – CH0 - GANHO 4- ERRO PERCENTUAL



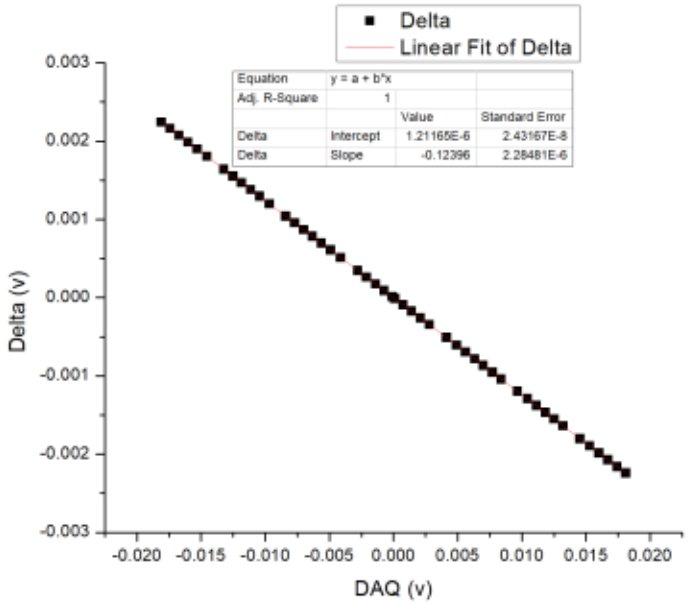
FONTE: Autor (2019)

FIGURA 34 – CH0 - GANHO 8- AD7794 CONTRA REFERENCIA



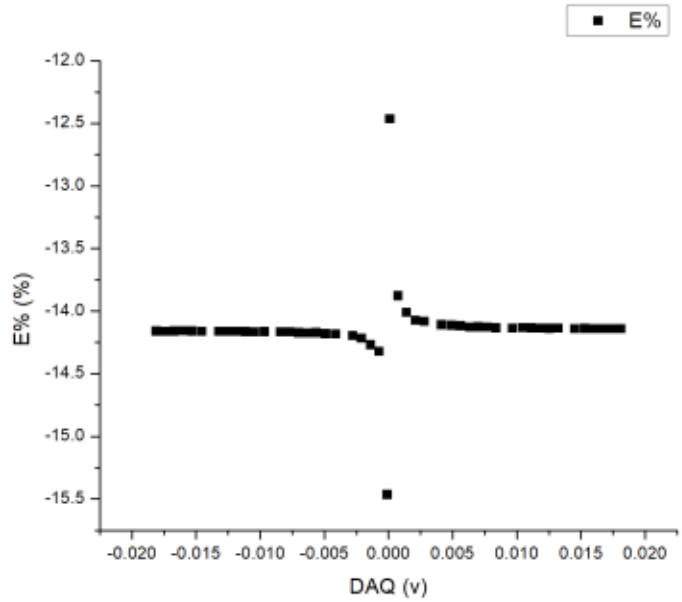
FONTE: Autor (2019)

FIGURA 35 – CH0 - GANHO 8- DIFERENÇA ENTRE AD7794 E REFERENCIA



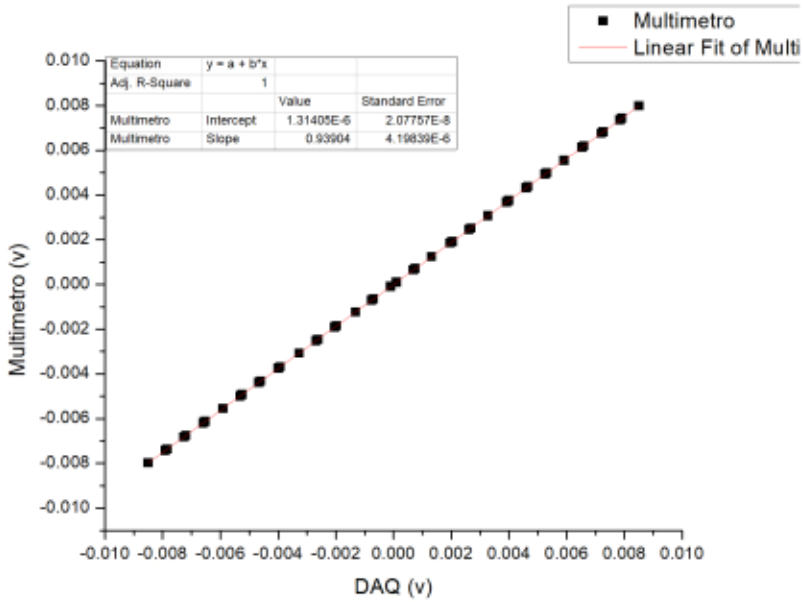
FONTE: Autor (2019)

FIGURA 36 – CH0 - GANHO 8- ERRO PERCENTUAL



FONTE: Autor (2019)

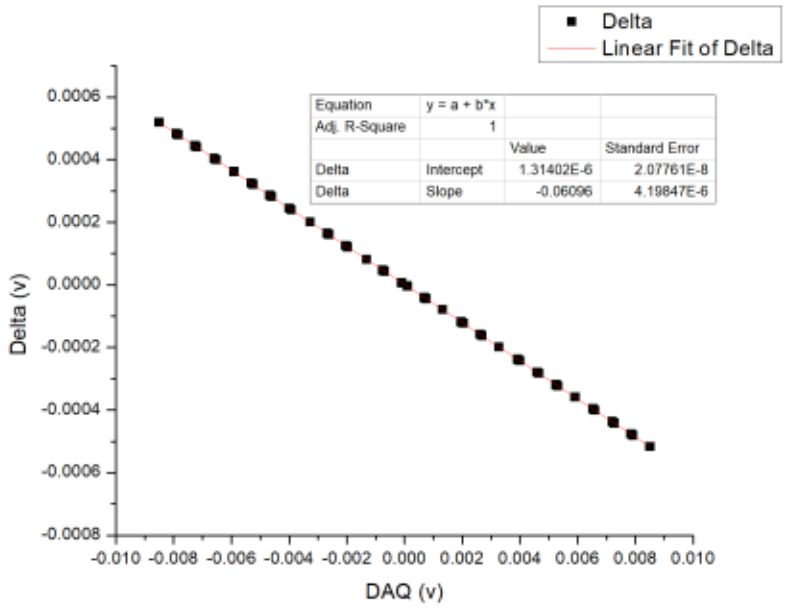
FIGURA 37 – CH0 - GANHO 16- AD7794 CONTRA REFERENCIA



FONTE: Autor (2019)

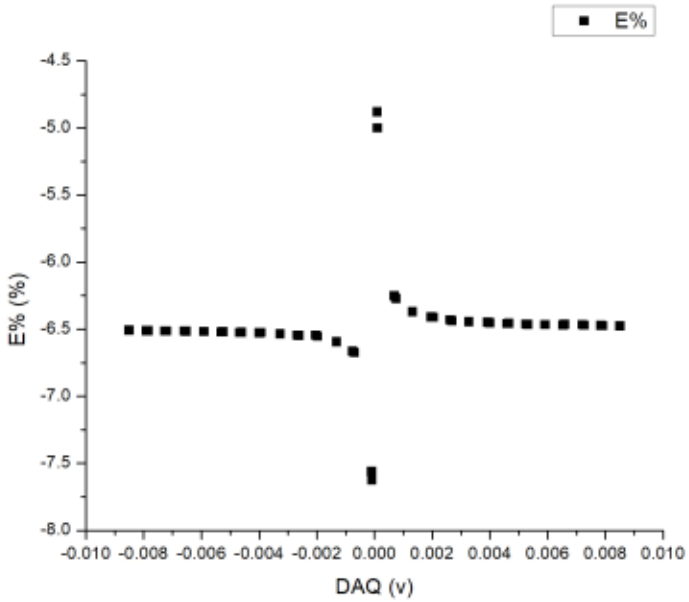
H.1.2 Ain 2

FIGURA 38 – CH0 - GANHO 16- DIFERENÇA ENTRE AD7794 E REFERENCIA



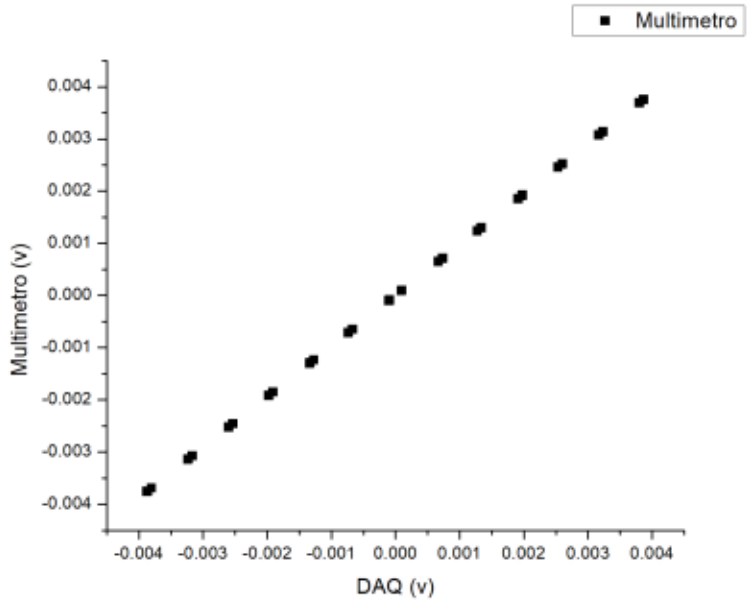
FONTE: Autor (2019)

FIGURA 39 – CH0 - GANHO 16- ERRO PERCENTUAL



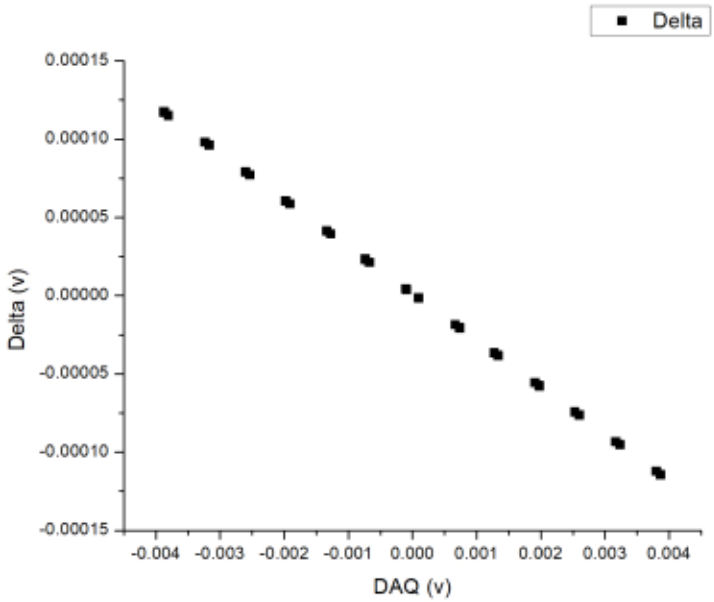
FONTE: Autor (2019)

FIGURA 40 – CH0 - GANHO 32- AD7794 CONTRA REFERENCIA



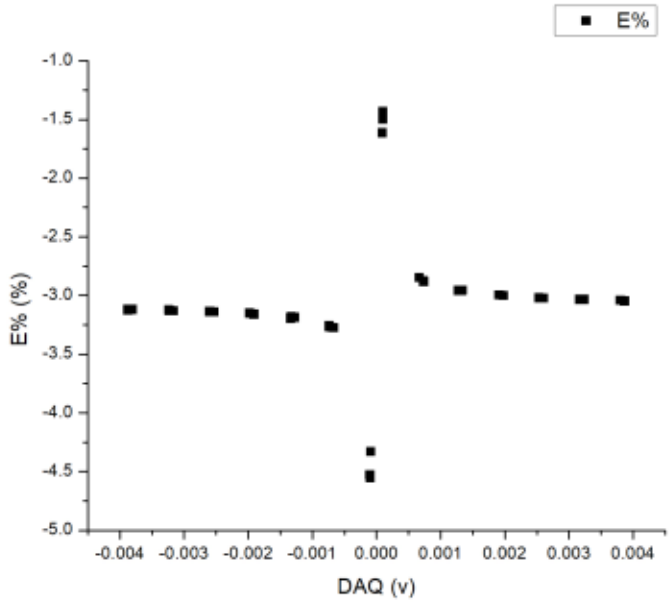
FONTE: Autor (2019)

FIGURA 41 – CH0 - GANHO 32- DIFERENÇA ENTRE AD7794 E REFERENCIA



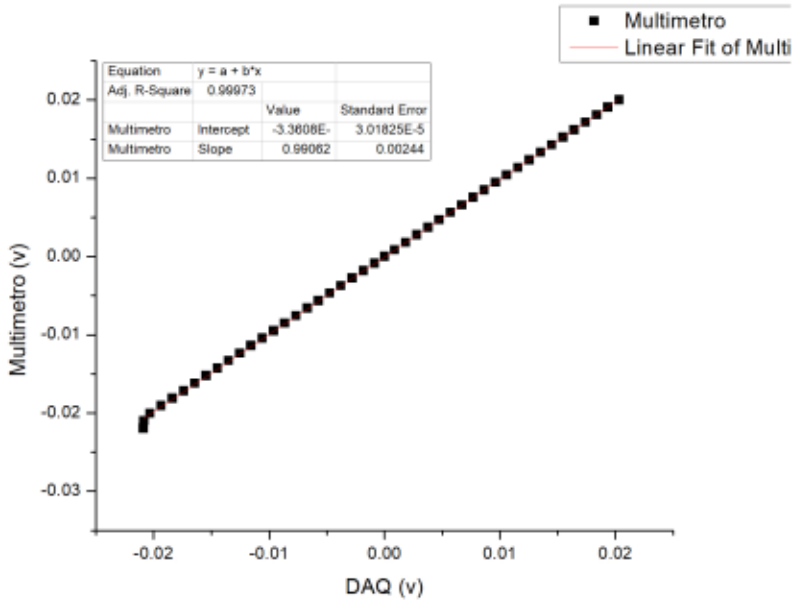
FONTE: Autor (2019)

FIGURA 42 – CH0 - GANHO 32- ERRO PERCENTUAL



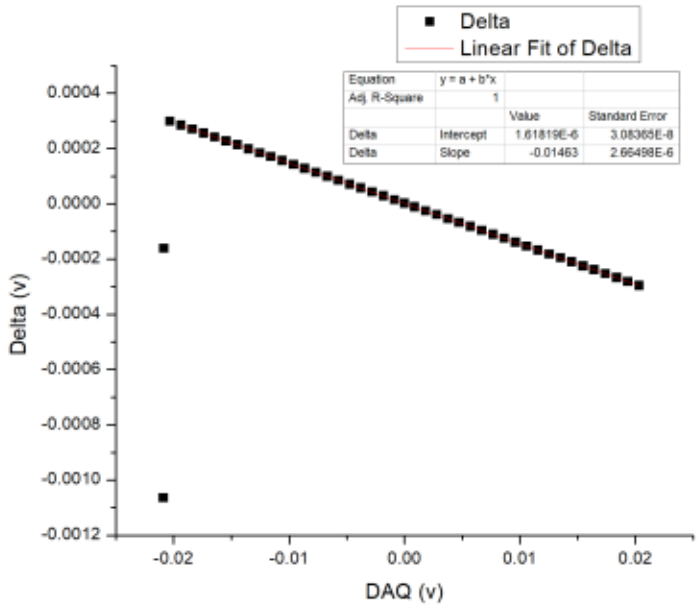
FONTE: Autor (2019)

FIGURA 43 – CH0 - GANHO 64- AD7794 CONTRA REFERENCIA



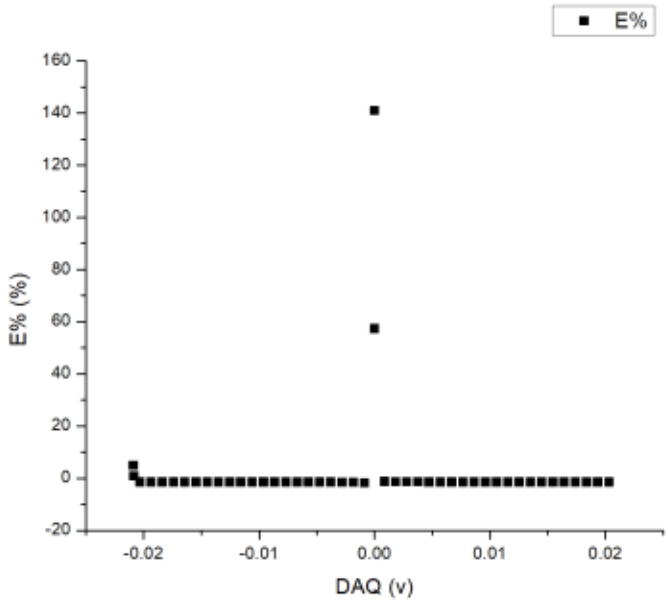
FONTE: Autor (2019)

FIGURA 44 – CH0 - GANHO 64- DIFERENÇA ENTRE AD7794 E REFERENCIA



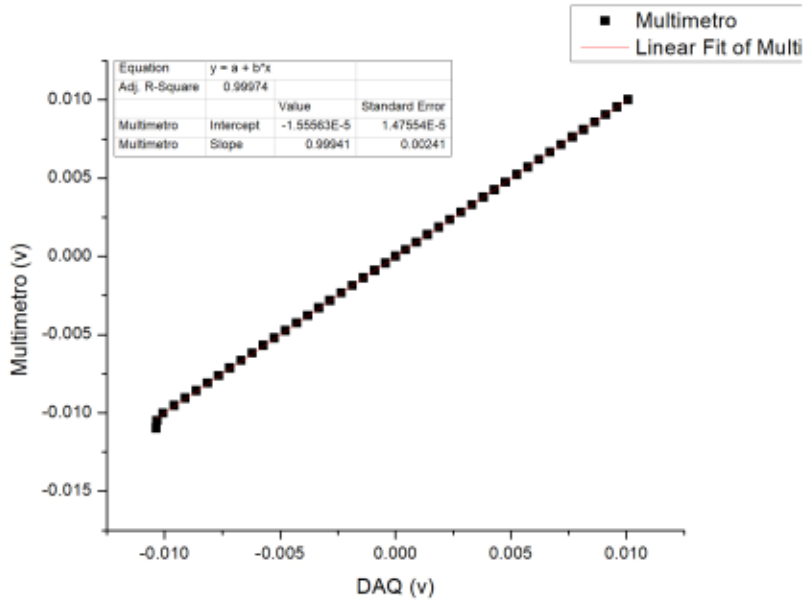
FONTE: Autor (2019)

FIGURA 45 – CH0 - GANHO 64- ERRO PERCENTUAL



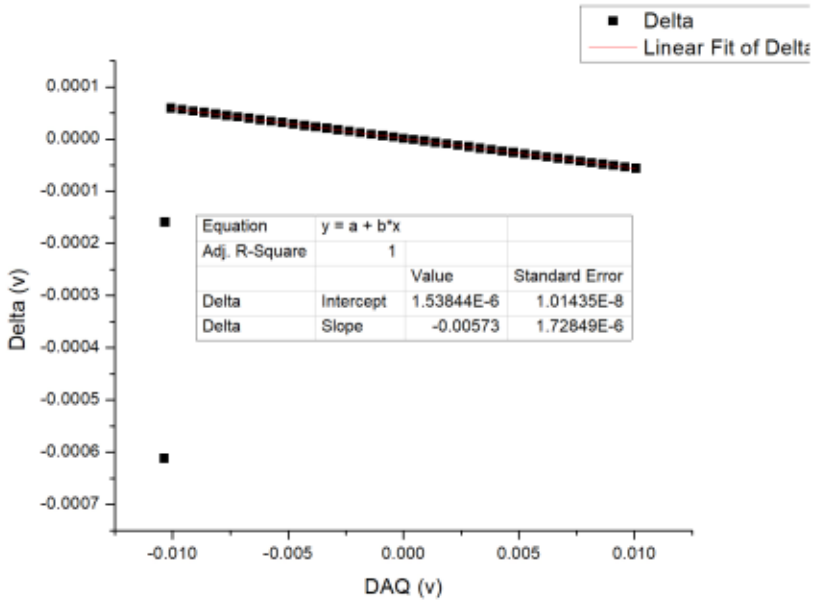
FONTE: Autor (2019)

FIGURA 46 – CH0 - GANHO 128- AD7794 CONTRA REFERENCIA



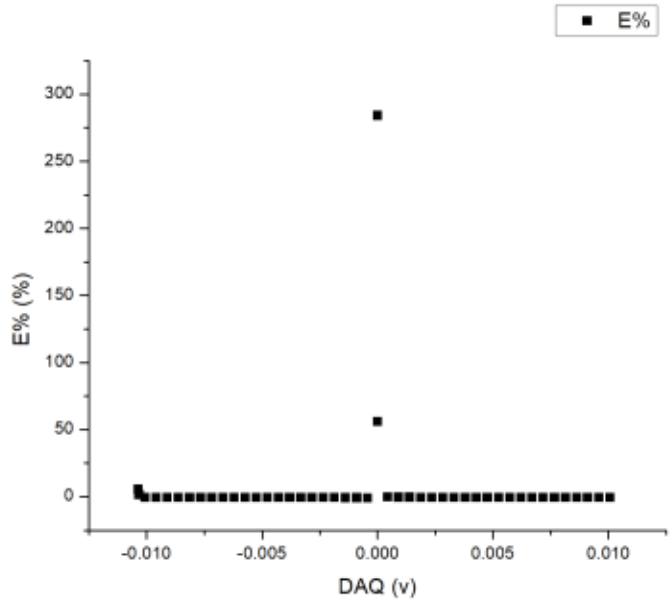
FONTE: Autor (2019)

FIGURA 47 – CH0 - GANHO 128- DIFERENÇA ENTRE AD7794 E REFERENCIA



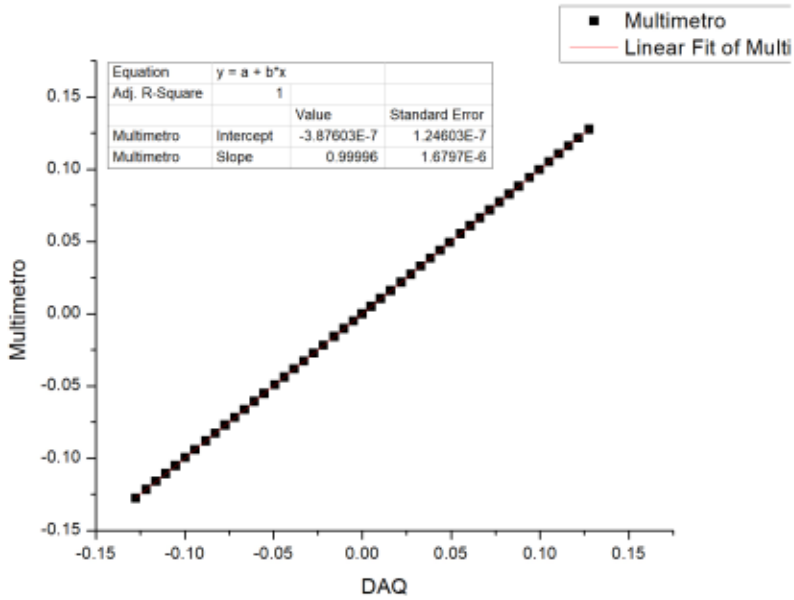
FONTE: Autor (2019)

FIGURA 48 – CH0 - GANHO 128- ERRO PERCENTUAL



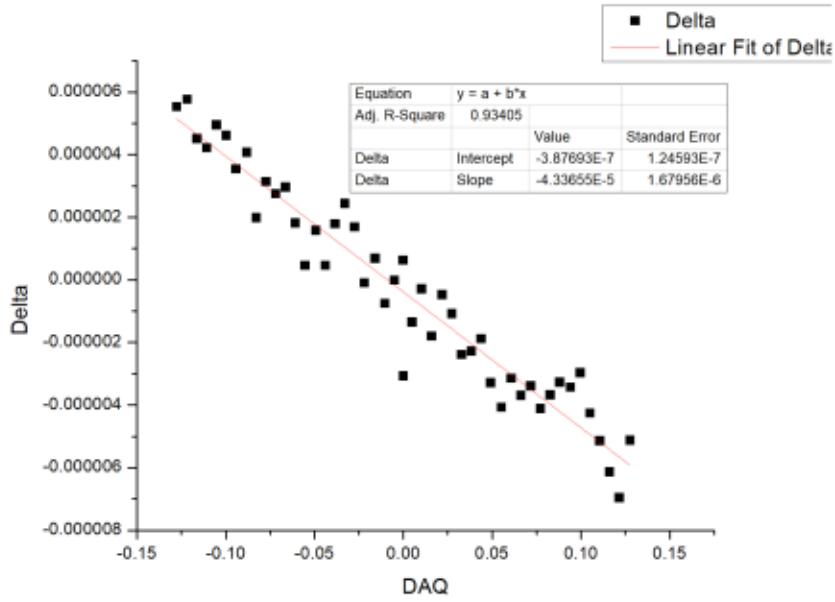
FONTE: Autor (2019)

FIGURA 49 – CH1 - GANHO 1- AD7794 CONTRA REFERENCIA



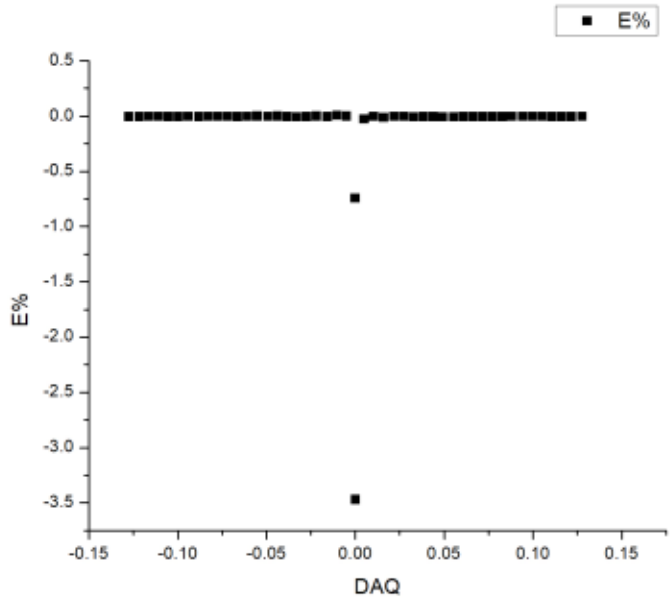
FONTE: Autor (2019)

FIGURA 50 – CH1 - GANHO 1- DIFERENÇA ENTRE AD7794 E REFERENCIA



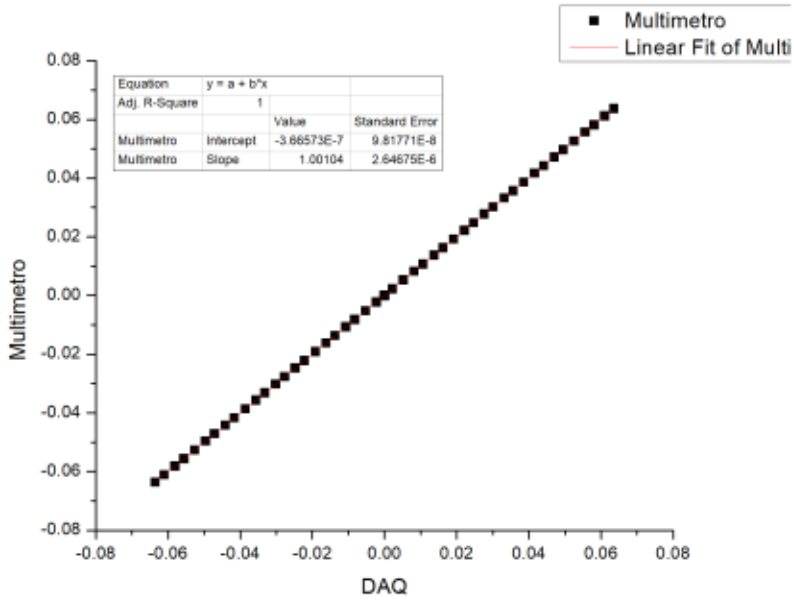
FONTE: Autor (2019)

FIGURA 51 – CH1 - GANHO 1- ERRO PERCENTUAL



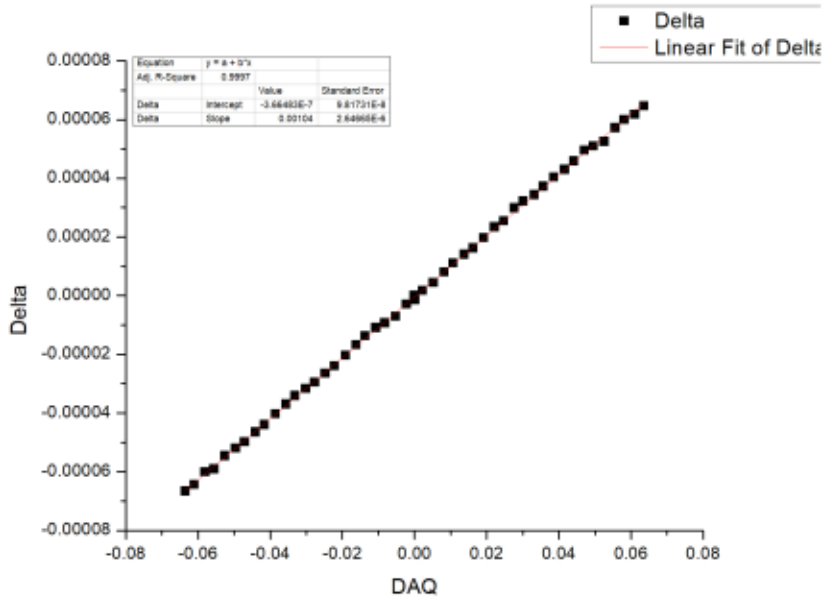
FONTE: Autor (2019)

FIGURA 52 – CH1 - GANHO 2- AD7794 CONTRA REFERENCIA



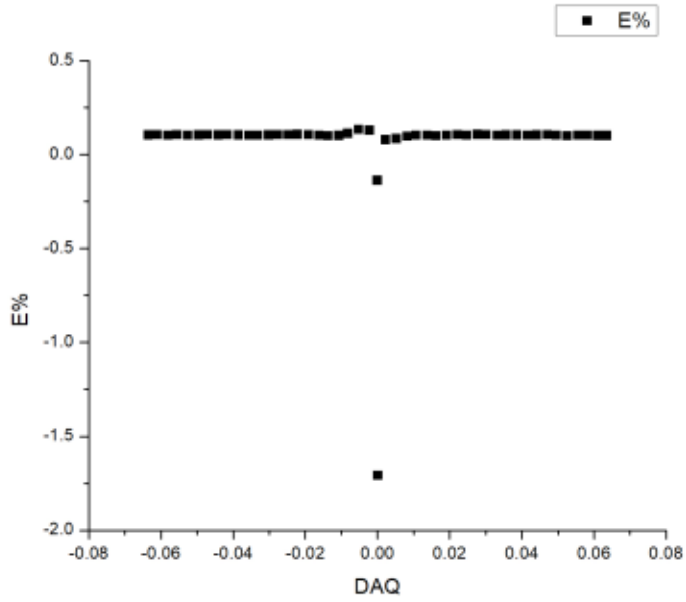
FONTE: Autor (2019)

FIGURA 53 – CH1 - GANHO 2- DIFERENÇA ENTRE AD7794 E REFERENCIA



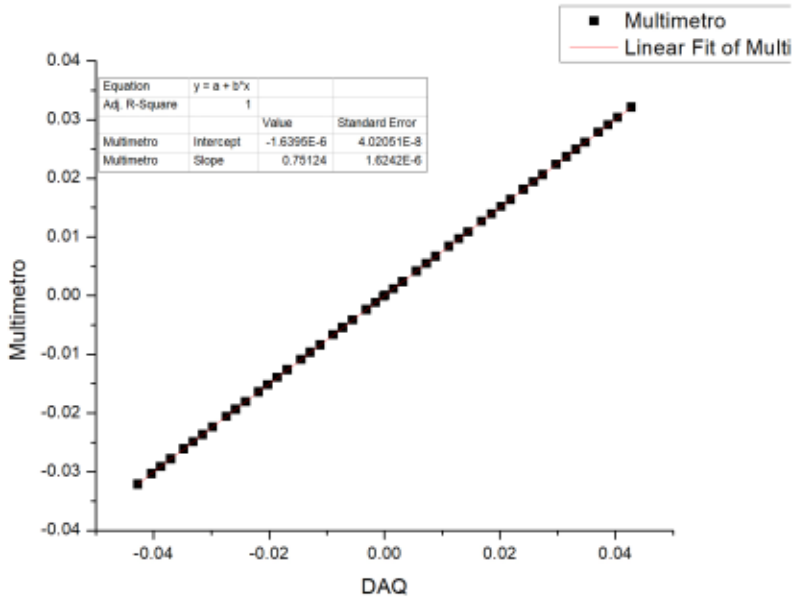
FONTE: Autor (2019)

FIGURA 54 – CH1 - GANHO 2- ERRO PERCENTUAL



FONTE: Autor (2019)

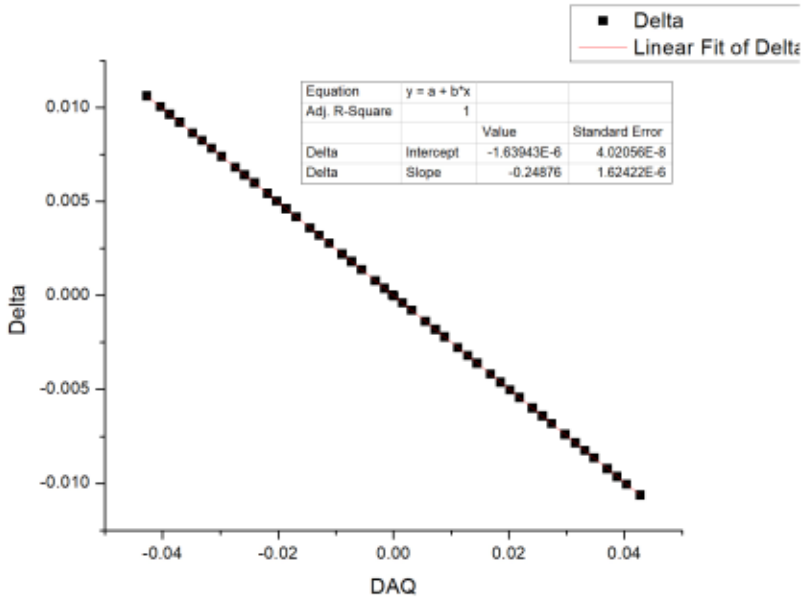
FIGURA 55 – CH1 - GANHO 4- AD7794 CONTRA REFERENCIA



FONTE: Autor (2019)

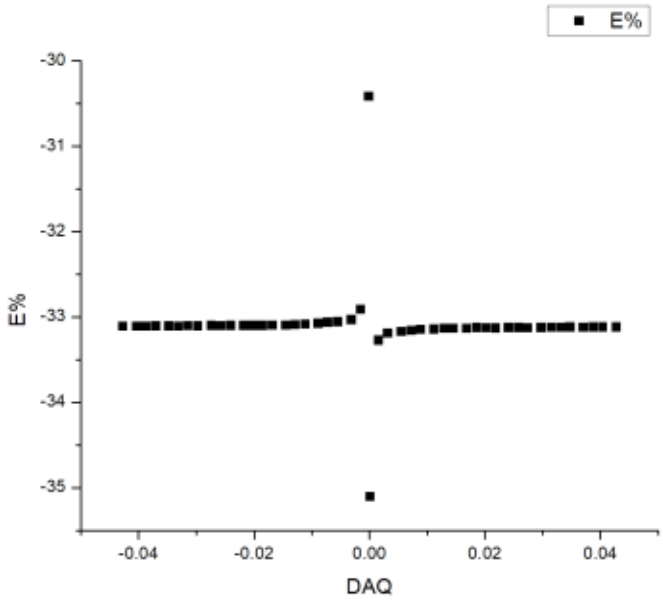
H.1.3 Ain 3

FIGURA 56 – CH1 - GANHO 4- DIFERENÇA ENTRE AD7794 E REFERENCIA



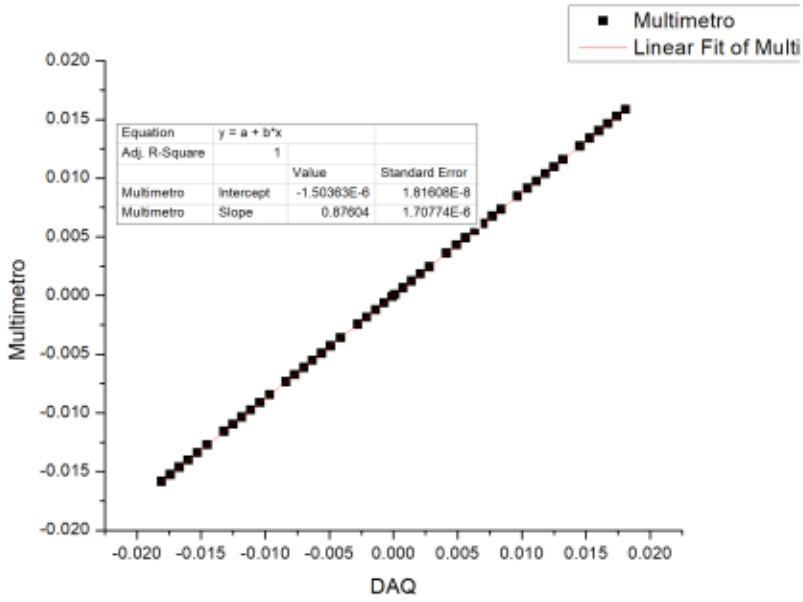
FONTE: Autor (2019)

FIGURA 57 – CH1 - GANHO 4- ERRO PERCENTUAL



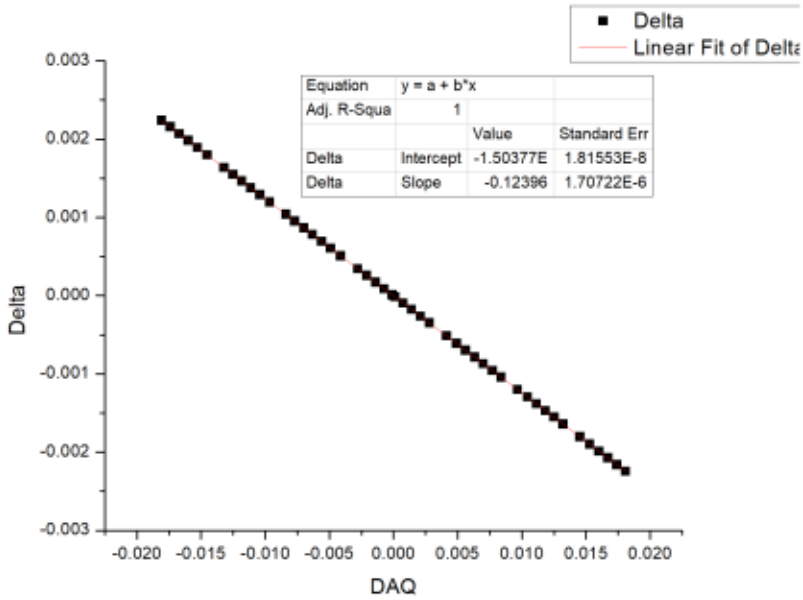
FONTE: Autor (2019)

FIGURA 58 – CH1 - GANHO 8- AD7794 CONTRA REFERENCIA



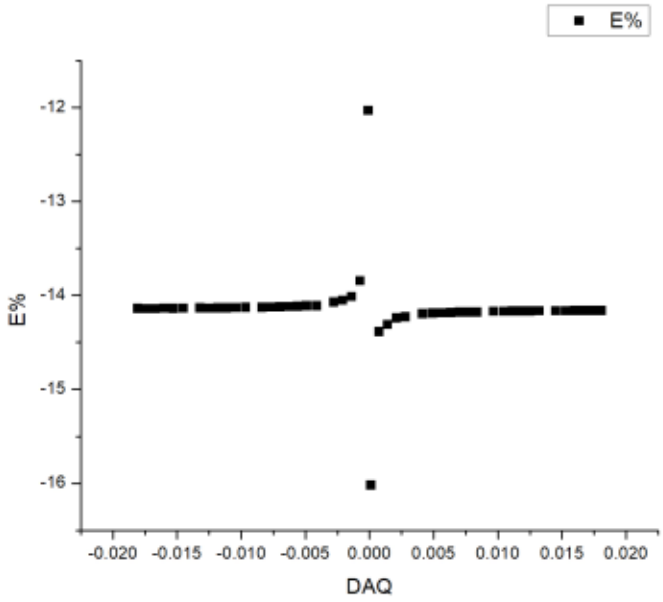
FONTE: Autor (2019)

FIGURA 59 – CH1 - GANHO 8- DIFERENÇA ENTRE AD7794 E REFERENCIA



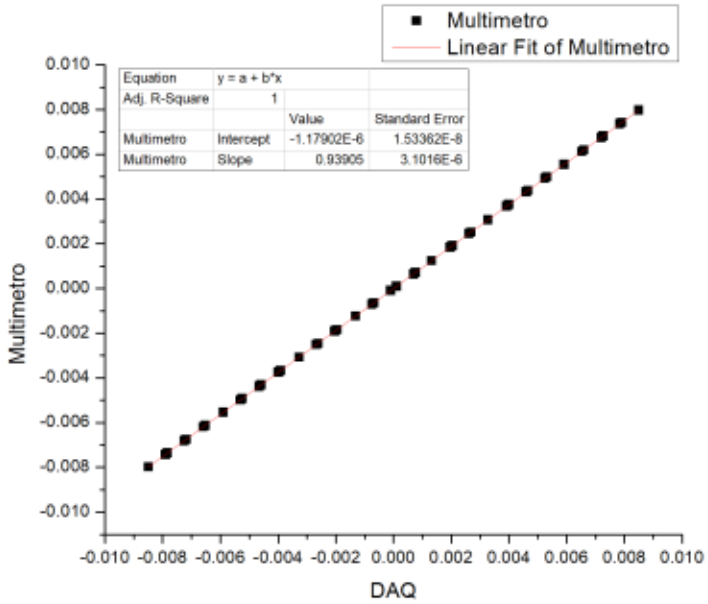
FONTE: Autor (2019)

FIGURA 60 – CH1 - GANHO 8- ERRO PERCENTUAL



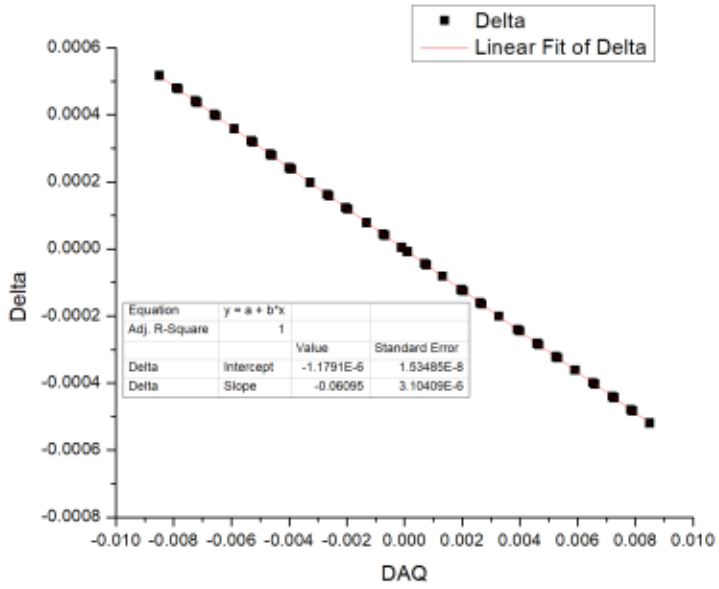
FONTE: Autor (2019)

FIGURA 61 – CH1 - GANHO 16- AD7794 CONTRA REFERENCIA



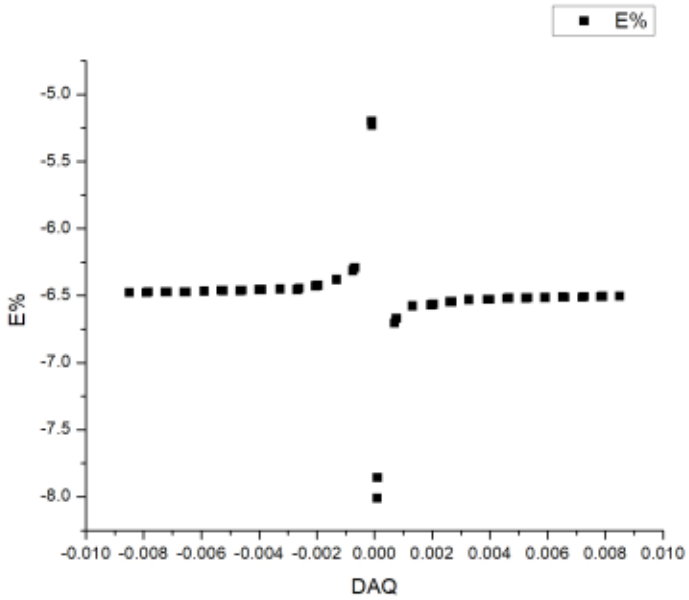
FONTE: Autor (2019)

FIGURA 62 – CH1 - GANHO 16- DIFERENÇA ENTRE AD7794 E REFERENCIA



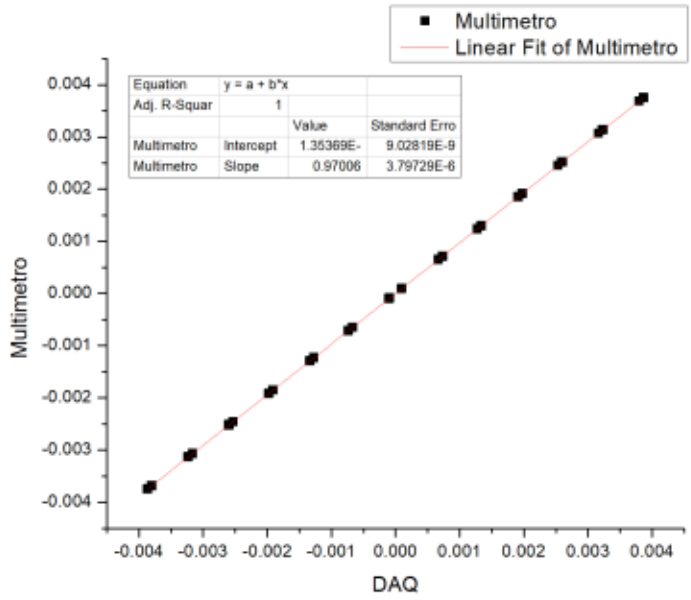
FONTE: Autor (2019)

FIGURA 63 – CH1 - GANHO 16- ERRO PERCENTUAL



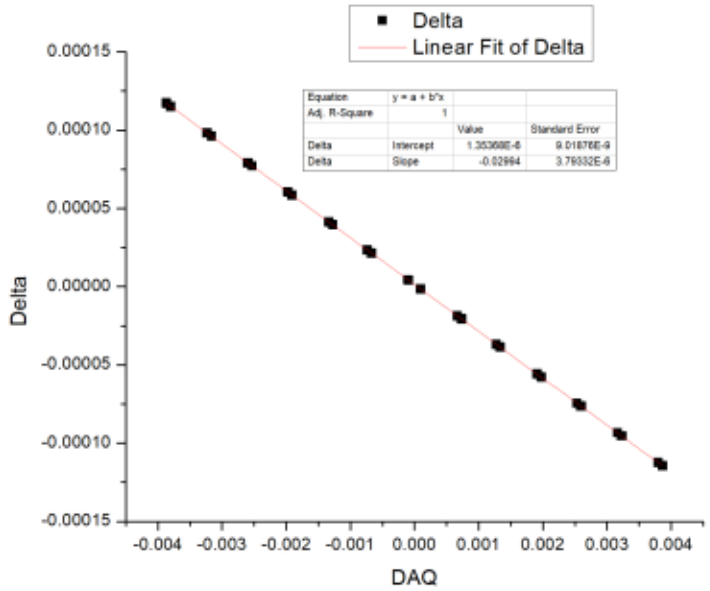
FONTE: Autor (2019)

FIGURA 64 – CH1 - GANHO 32- AD7794 CONTRA REFERENCIA



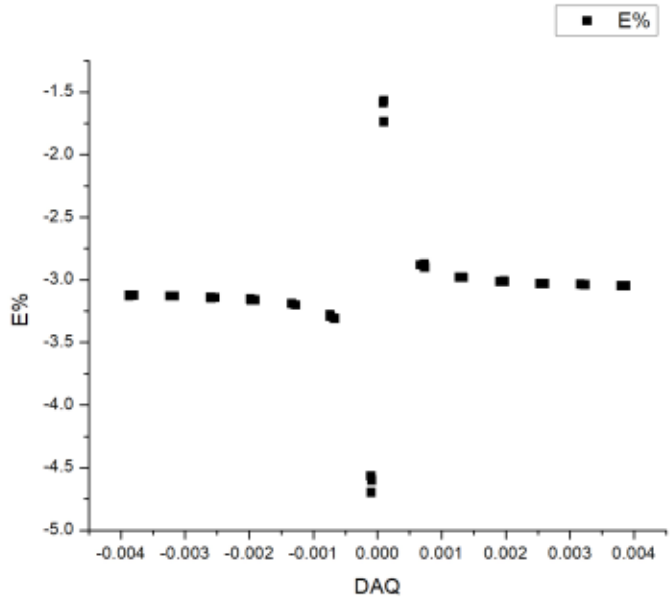
FONTE: Autor (2019)

FIGURA 65 – CH1 - GANHO 32- DIFERENÇA ENTRE AD7794 E REFERENCIA



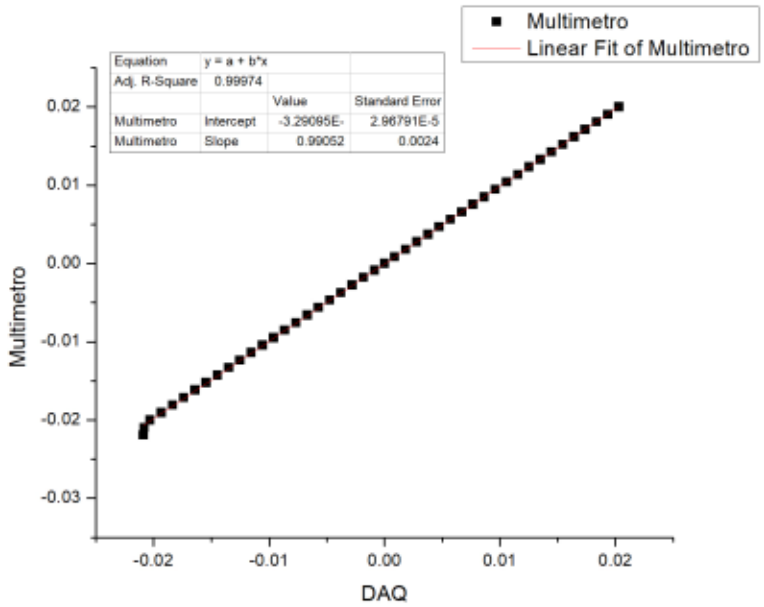
FONTE: Autor (2019)

FIGURA 66 – CH1 - GANHO 32- ERRO PERCENTUAL



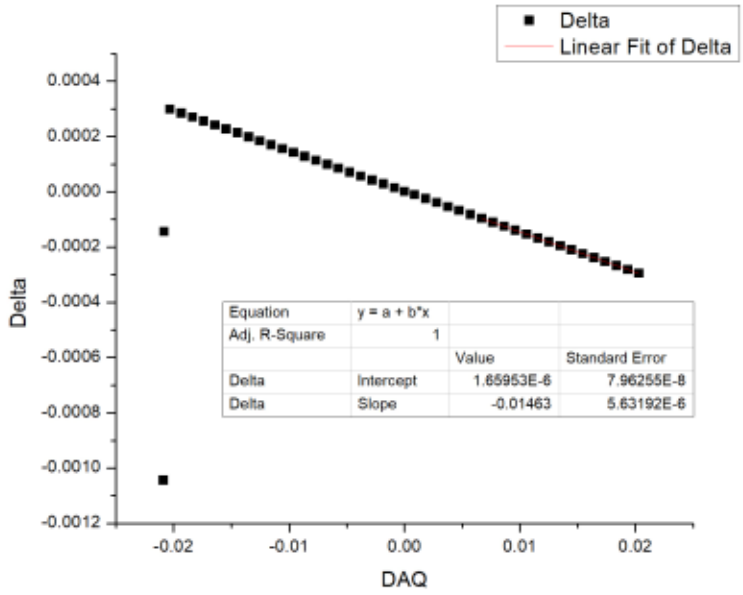
FONTE: Autor (2019)

FIGURA 67 – CH1 - GANHO 64- AD7794 CONTRA REFERENCIA



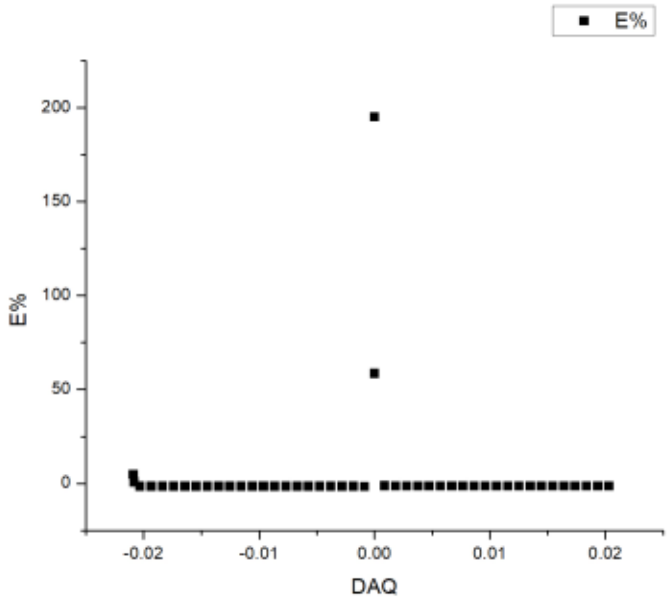
FONTE: Autor (2019)

FIGURA 68 – CH1 - GANHO 64- DIFERENÇA ENTRE AD7794 E REFERENCIA



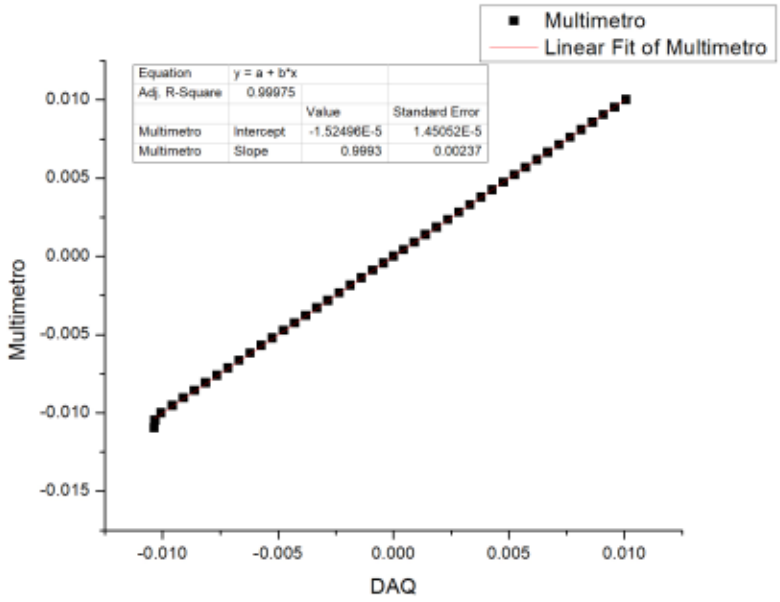
FONTE: Autor (2019)

FIGURA 69 – CH1 - GANHO 64- ERRO PERCENTUAL



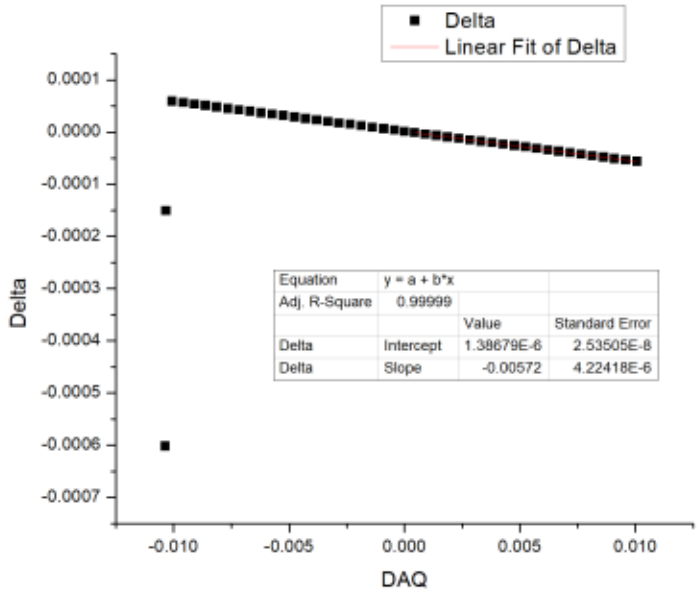
FONTE: Autor (2019)

FIGURA 70 – CH1 - GANHO 128- AD7794 CONTRA REFERENCIA



FONTE: Autor (2019)

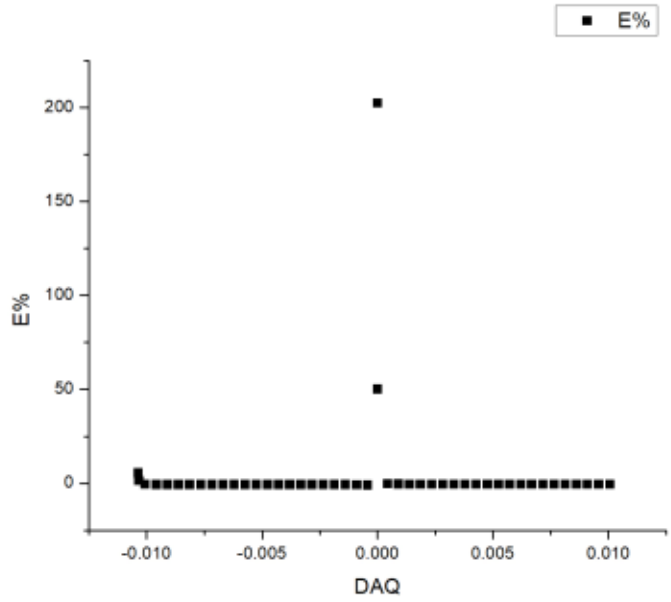
FIGURA 71 – CH1 - GANHO 128- DIFERENÇA ENTRE AD7794 E REFERENCIA



FONTE: Autor (2019)

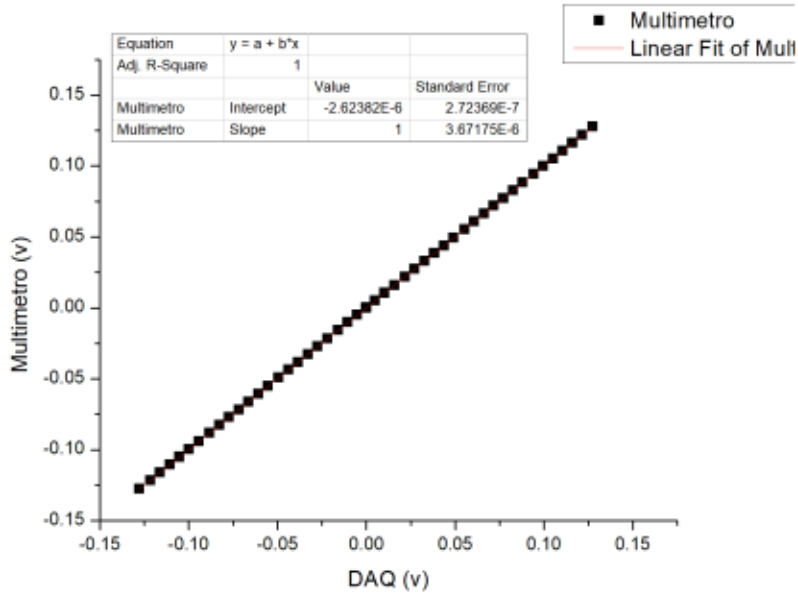
H.1.4 Ain 4

FIGURA 72 – CH1 - GANHO 128- ERRO PERCENTUAL



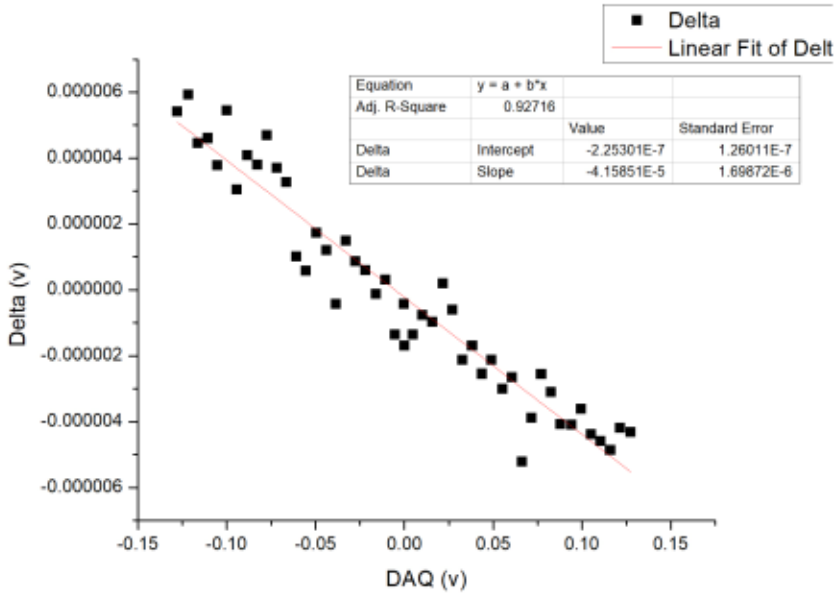
FONTE: Autor (2019)

FIGURA 73 – CH2 - GANHO 1- AD7794 CONTRA REFERENCIA



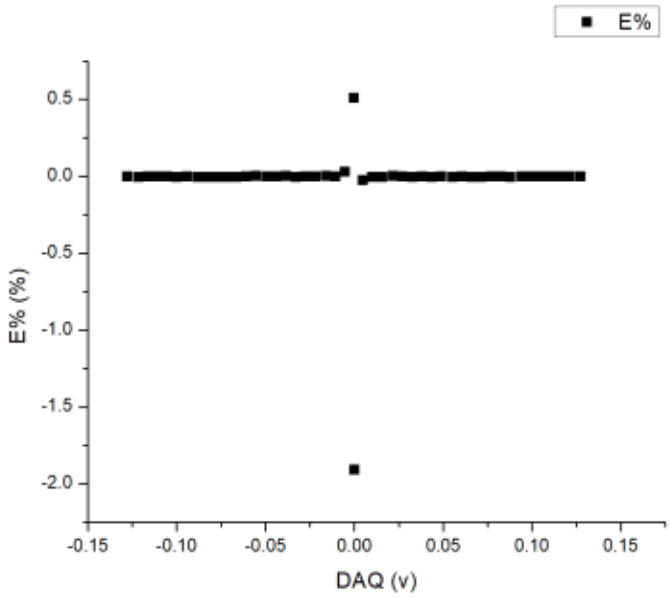
FONTE: Autor (2019)

FIGURA 74 – CH2 - GANHO 1- DIFERENÇA ENTRE AD7794 E REFERENCIA



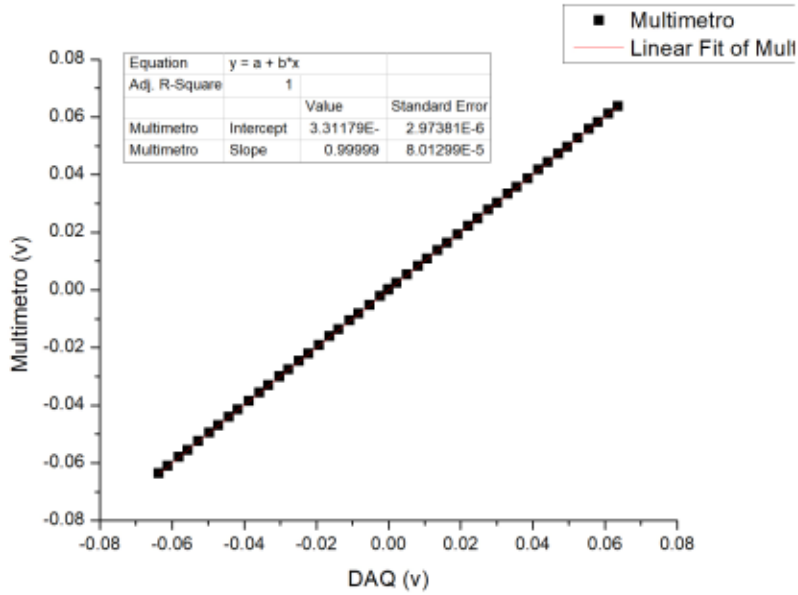
FONTE: Autor (2019)

FIGURA 75 – CH2 - GANHO 1- ERRO PERCENTUAL



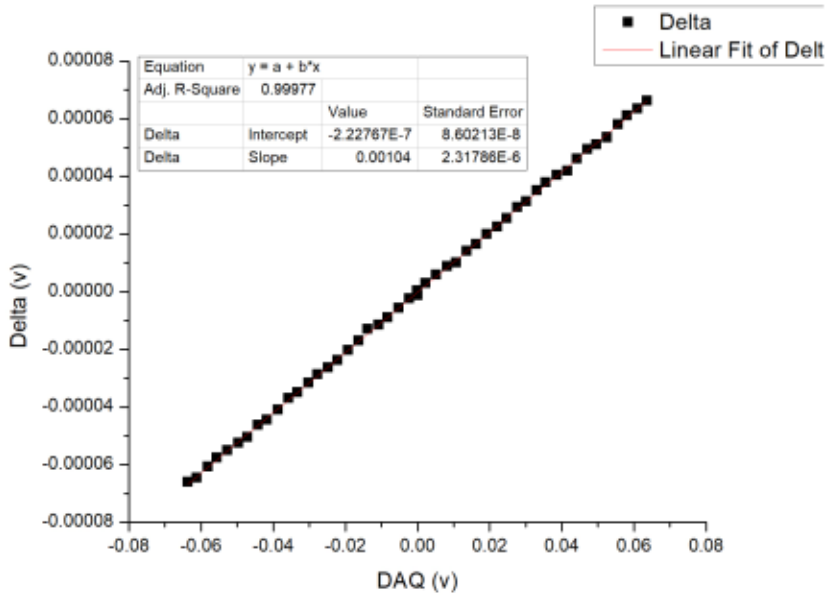
FONTE: Autor (2019)

FIGURA 76 – CH2 - GANHO 2- AD7794 CONTRA REFERENCIA



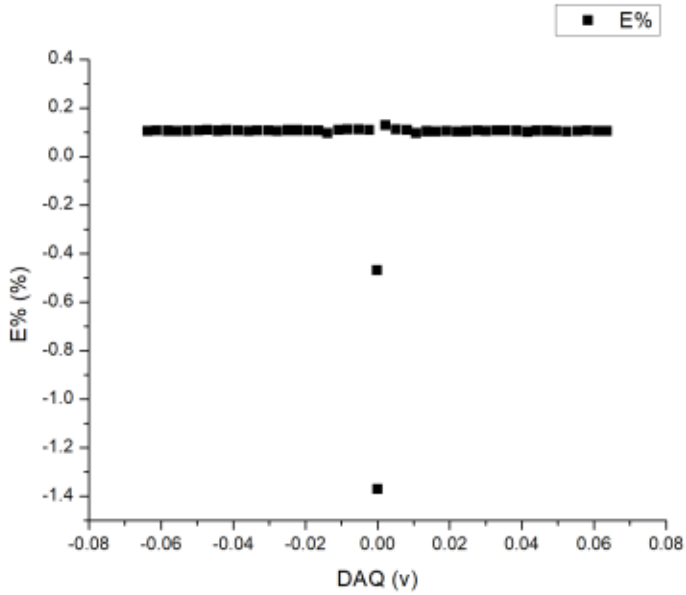
FONTE: Autor (2019)

FIGURA 77 – CH2 - GANHO 2- DIFERENÇA ENTRE AD7794 E REFERENCIA



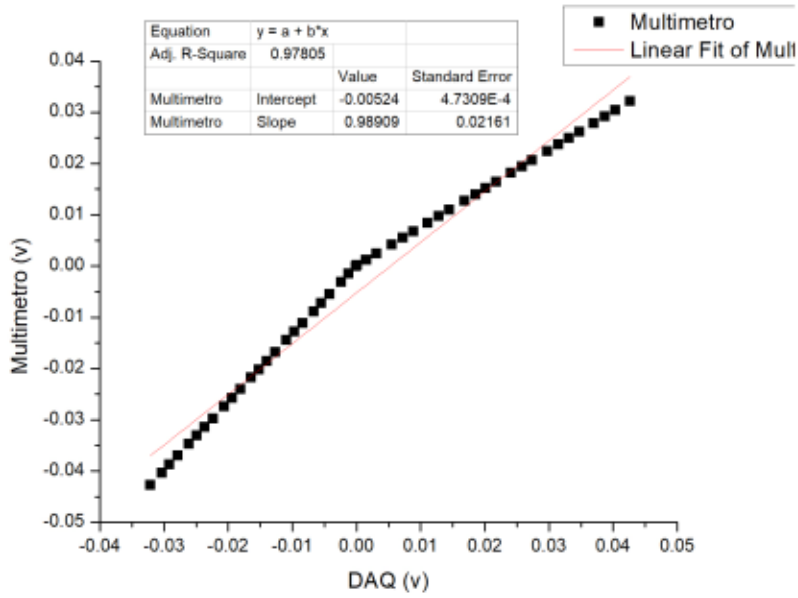
FONTE: Autor (2019)

FIGURA 78 – CH2 - GANHO 2- ERRO PERCENTUAL



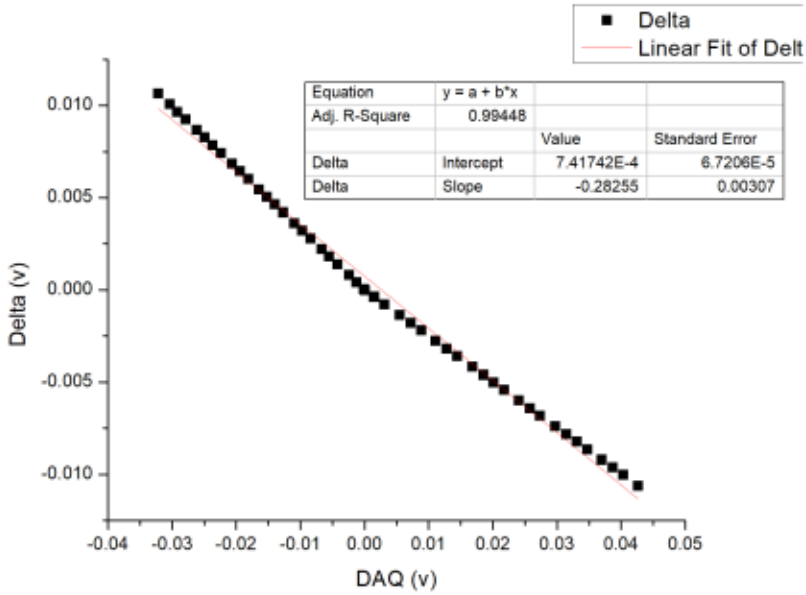
FONTE: Autor (2019)

FIGURA 79 – CH2 - GANHO 4- AD7794 CONTRA REFERENCIA



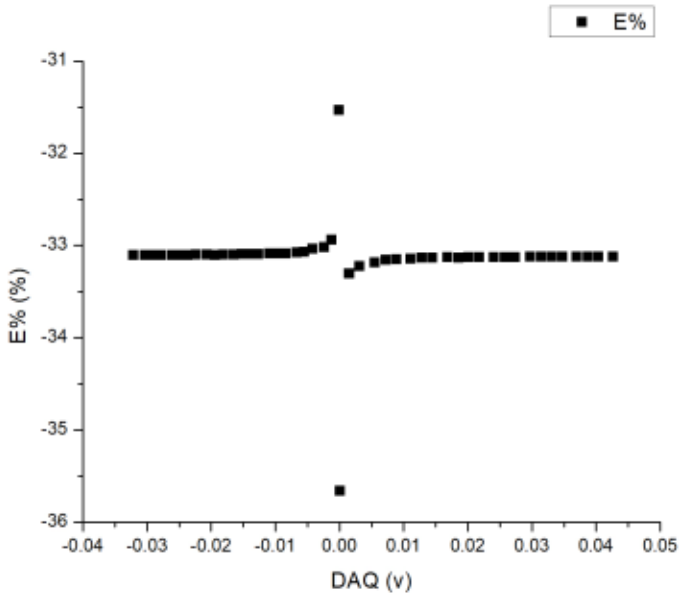
FONTE: Autor (2019)

FIGURA 80 – CH2 - GANHO 4- DIFERENÇA ENTRE AD7794 E REFERENCIA



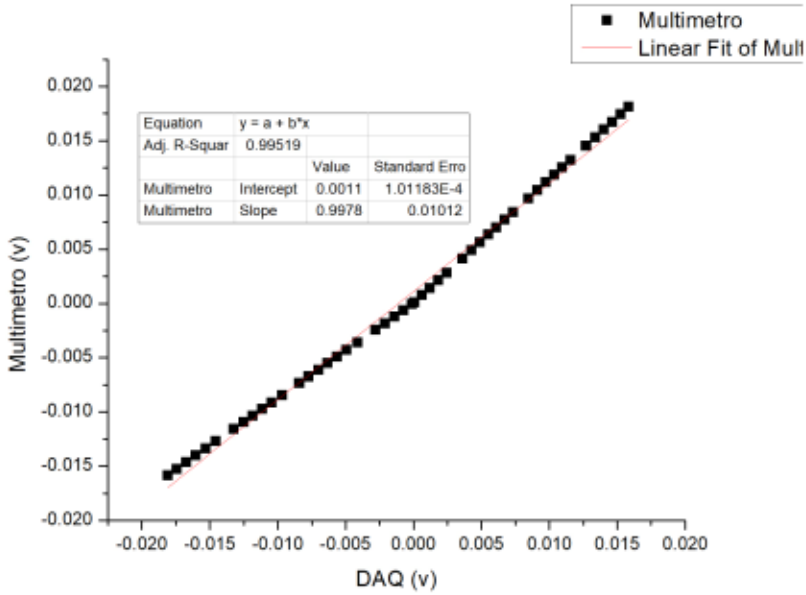
FONTE: Autor (2019)

FIGURA 81 – CH2 - GANHO 4- ERRO PERCENTUAL



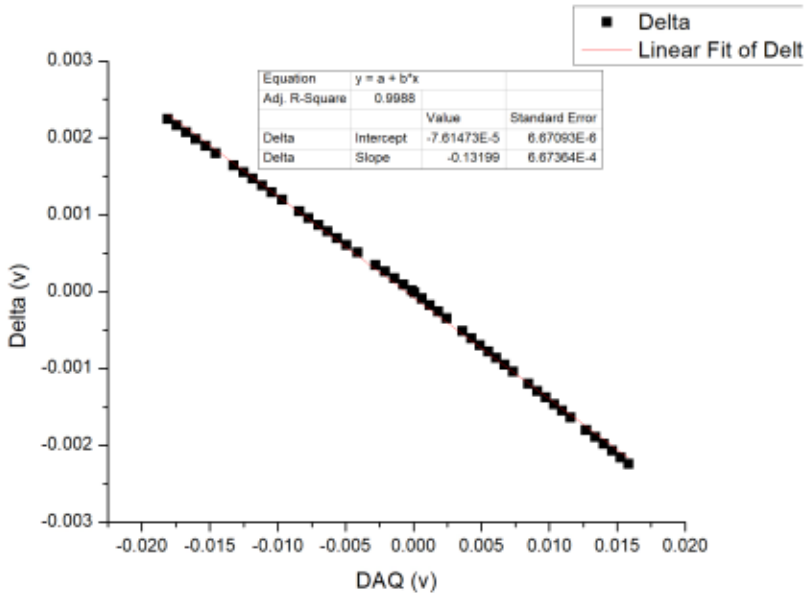
FONTE: Autor (2019)

FIGURA 82 – CH2 - GANHO 8- AD7794 CONTRA REFERENCIA



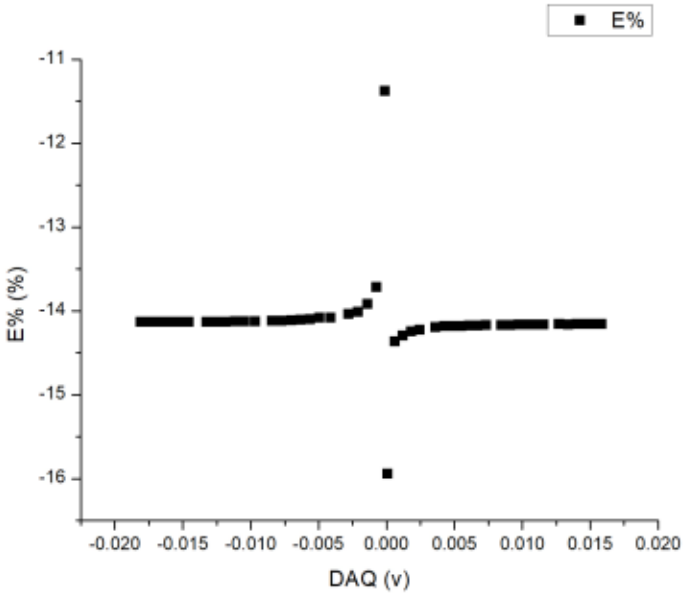
FONTE: Autor (2019)

FIGURA 83 – CH2 - GANHO 8- DIFERENÇA ENTRE AD7794 E REFERENCIA



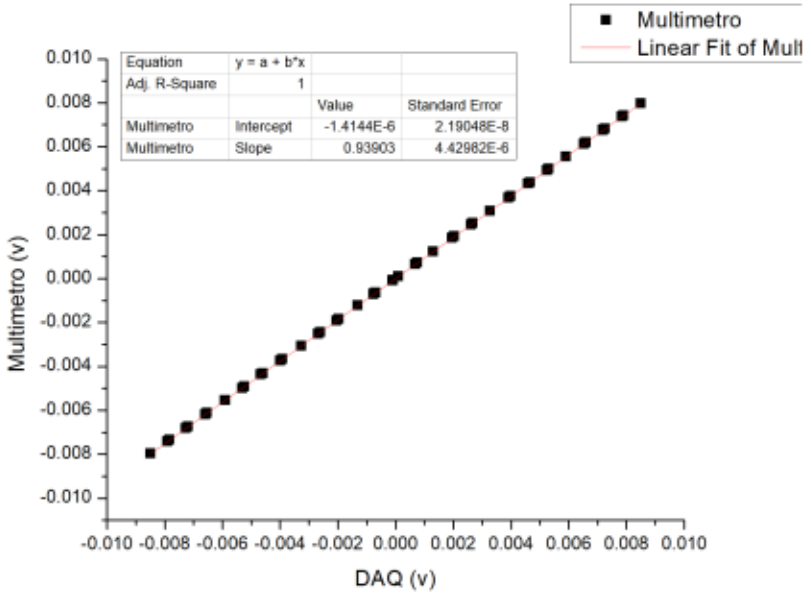
FONTE: Autor (2019)

FIGURA 84 – CH2 - GANHO 8- ERRO PERCENTUAL



FONTE: Autor (2019)

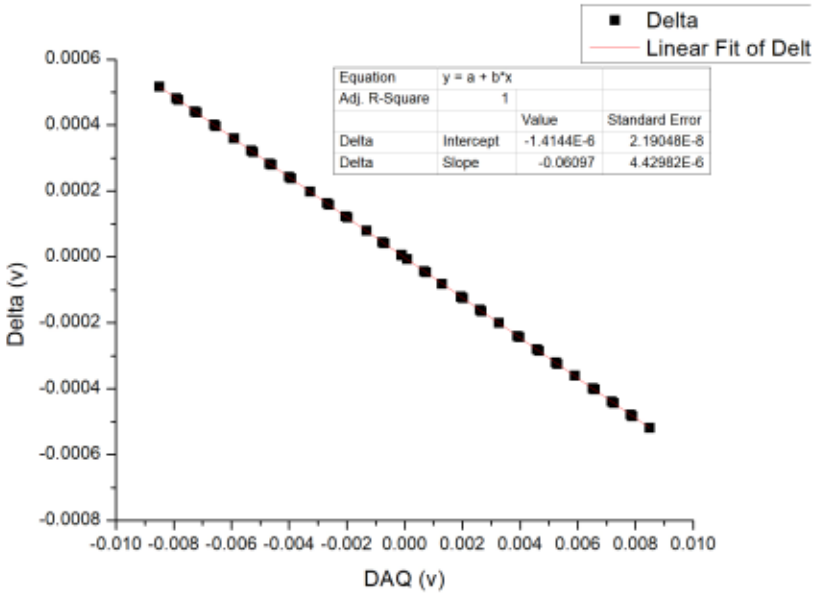
FIGURA 85 – CH2 - GANHO 16- AD7794 CONTRA REFERENCIA



FONTE: Autor (2019)

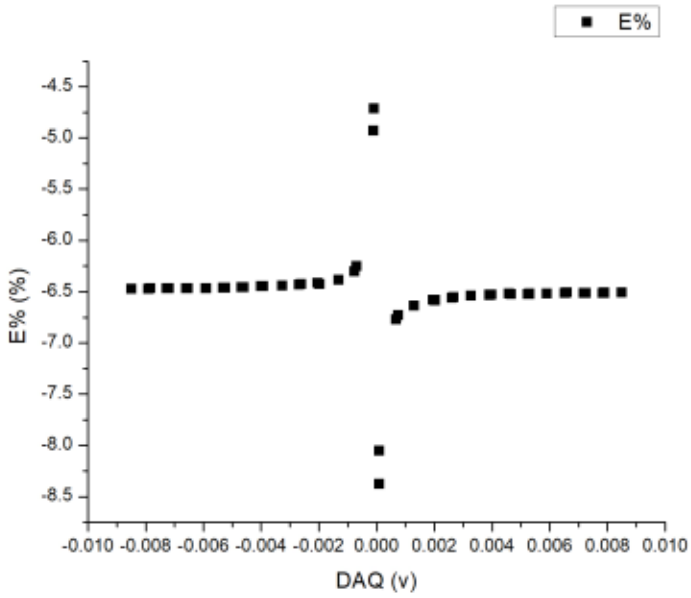
H.1.5 Ain 6

FIGURA 86 – CH2 - GANHO 16- DIFERENÇA ENTRE AD7794 E REFERENCIA



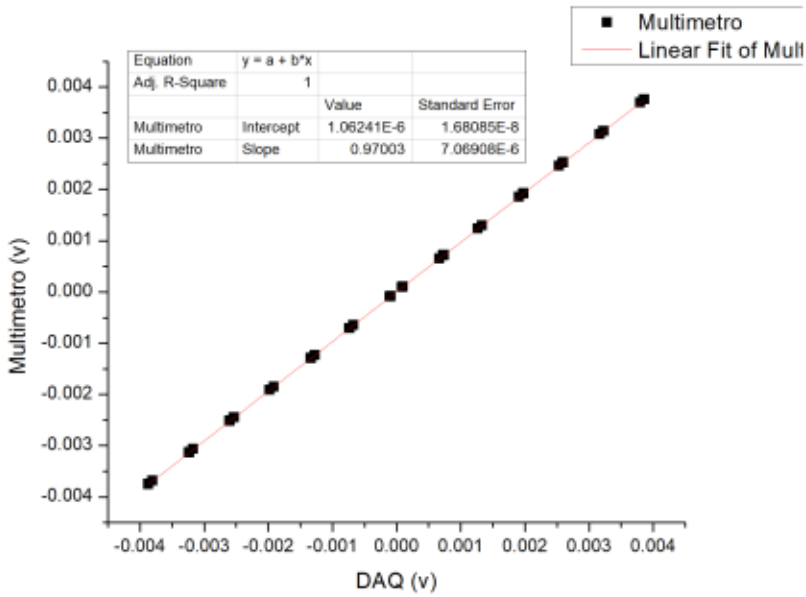
FONTE: Autor (2019)

FIGURA 87 – CH2 - GANHO 16- ERRO PERCENTUAL



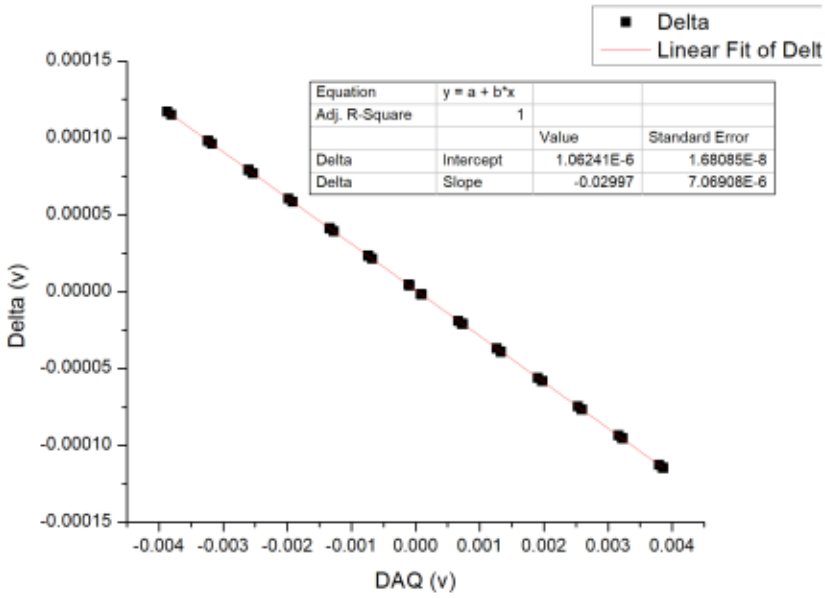
FONTE: Autor (2019)

FIGURA 88 – CH2 - GANHO 32- AD7794 CONTRA REFERENCIA



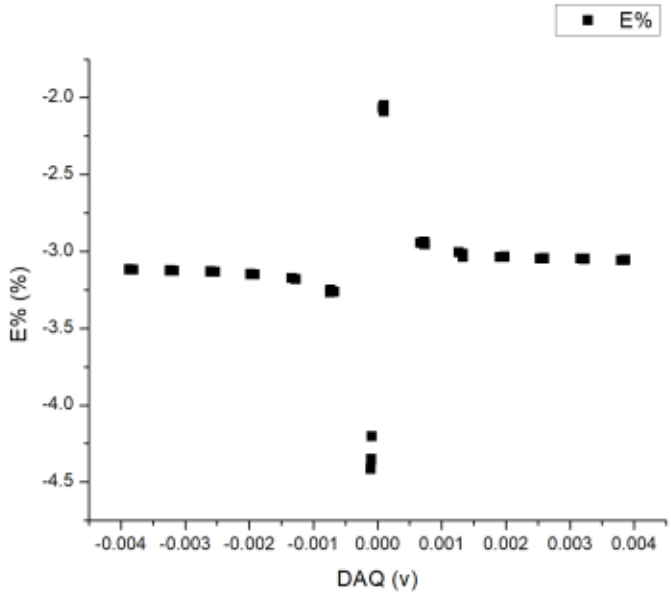
FONTE: Autor (2019)

FIGURA 89 – CH2 - GANHO 32- DIFERENÇA ENTRE AD7794 E REFERENCIA



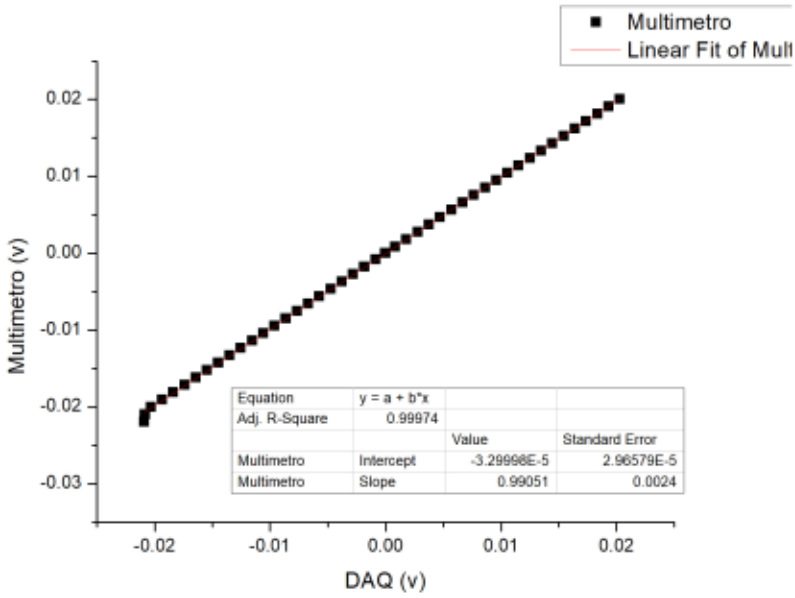
FONTE: Autor (2019)

FIGURA 90 – CH2 - GANHO 32- ERRO PERCENTUAL



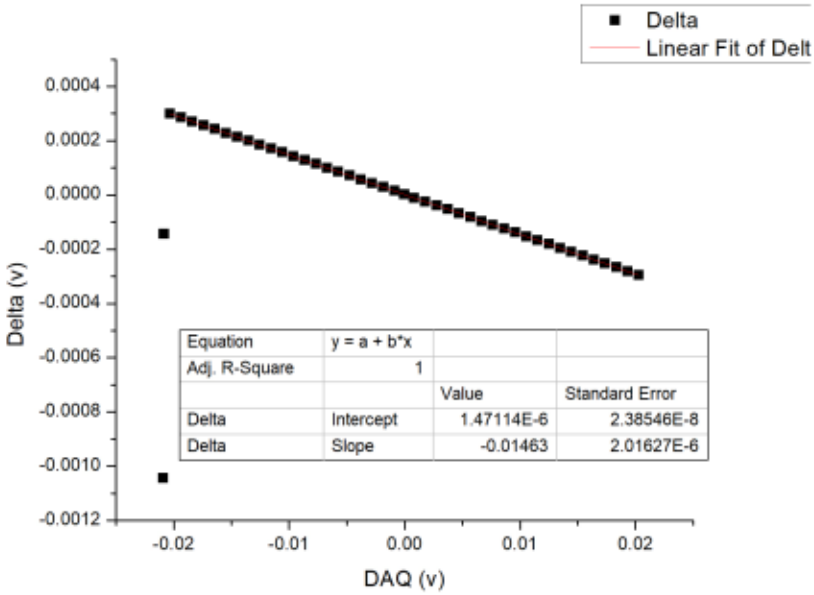
FONTE: Autor (2019)

FIGURA 91 – CH2 - GANHO 64- AD7794 CONTRA REFERENCIA



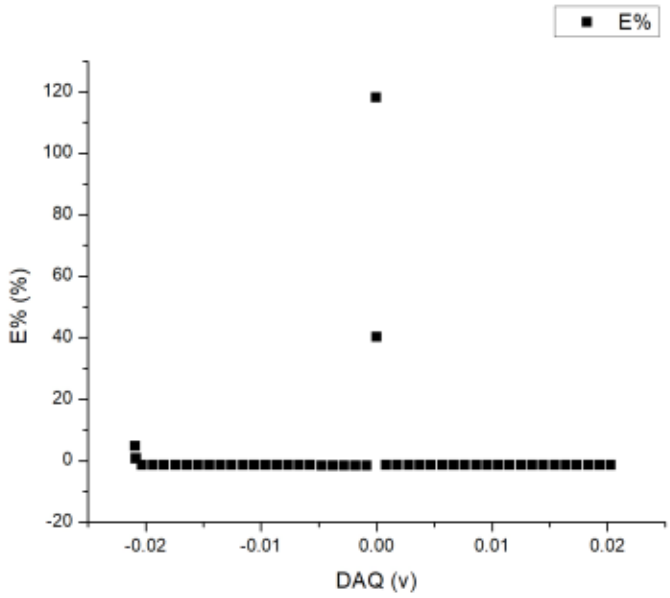
FONTE: Autor (2019)

FIGURA 92 – CH2 - GANHO 64- DIFERENÇA ENTRE AD7794 E REFERENCIA



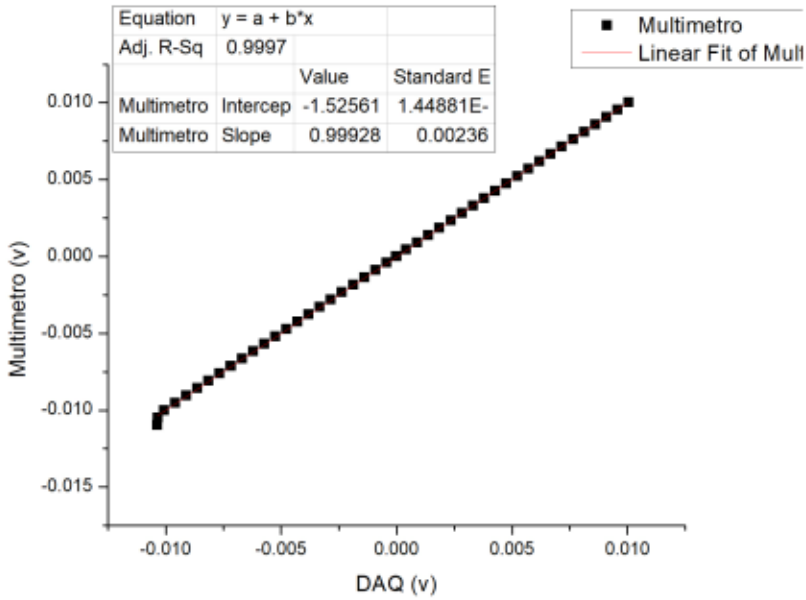
FONTE: Autor (2019)

FIGURA 93 – CH2 - GANHO 64- ERRO PERCENTUAL



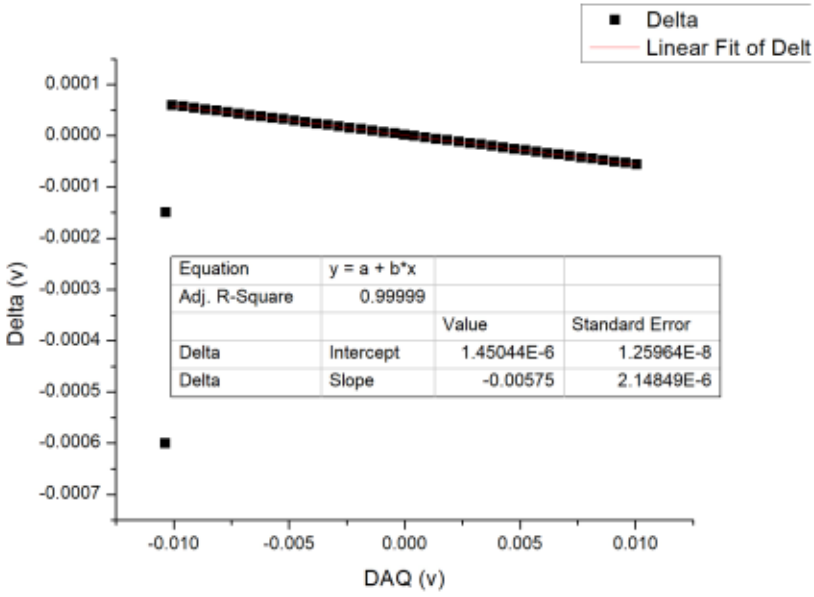
FONTE: Autor (2019)

FIGURA 94 – CH2 - GANHO 128- AD7794 CONTRA REFERENCIA



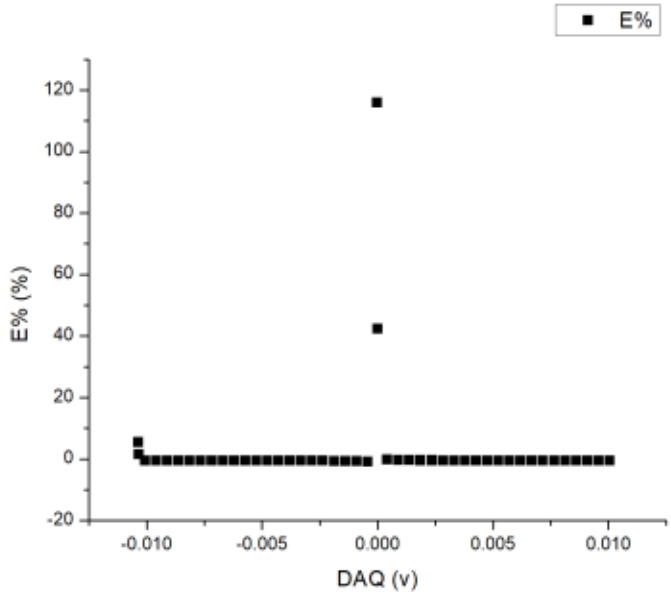
FONTE: Autor (2019)

FIGURA 95 – CH2 - GANHO 128- DIFERENÇA ENTRE AD7794 E REFERENCIA



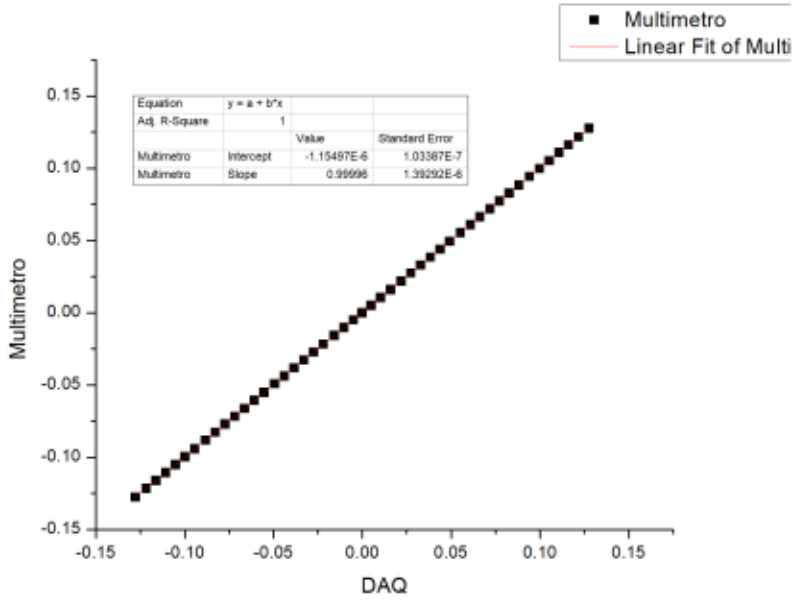
FONTE: Autor (2019)

FIGURA 96 – CH2 - GANHO 128- ERRO PERCENTUAL



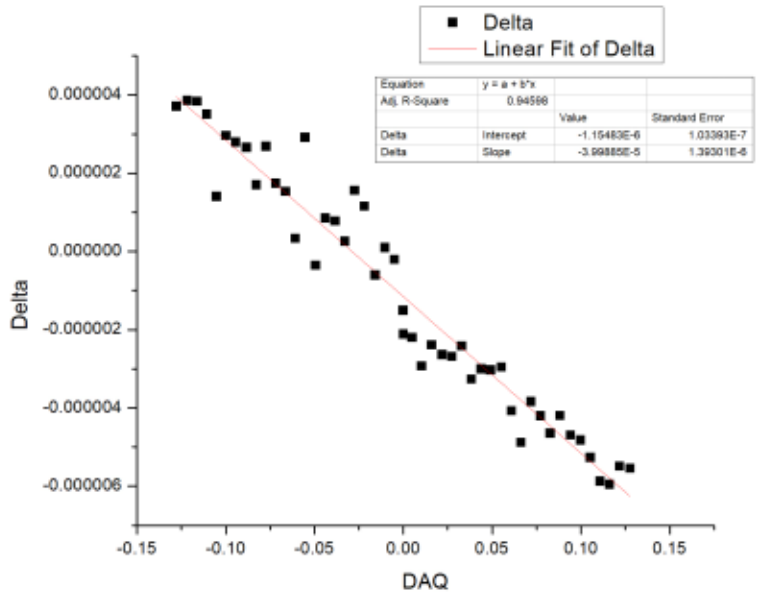
FONTE: Autor (2019)

FIGURA 97 – CH3 - GANHO 1- AD7794 CONTRA REFERENCIA



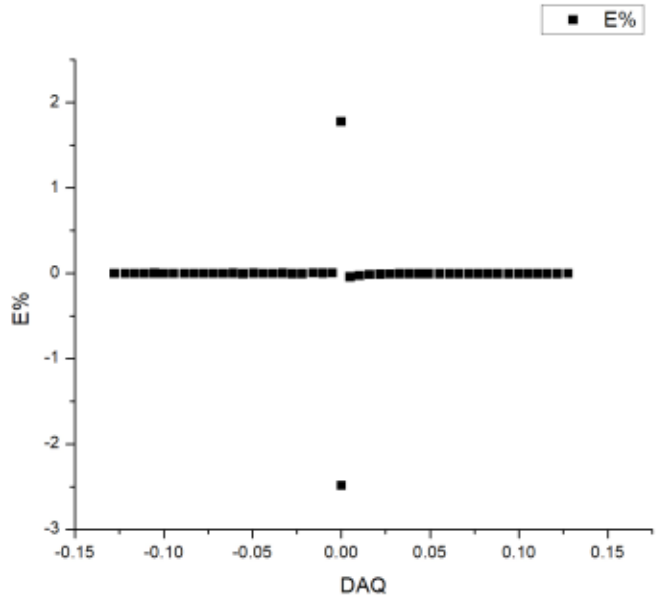
FONTE: Autor (2019)

FIGURA 98 – CH3 - GANHO 1- DIFERENÇA ENTRE AD7794 E REFERENCIA



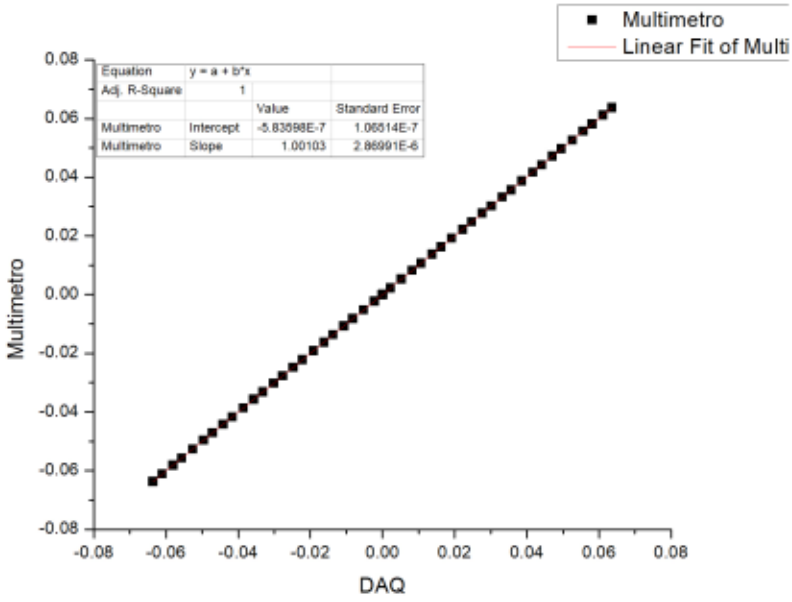
FONTE: Autor (2019)

FIGURA 99 – CH3 - GANHO 1- ERRO PERCENTUAL



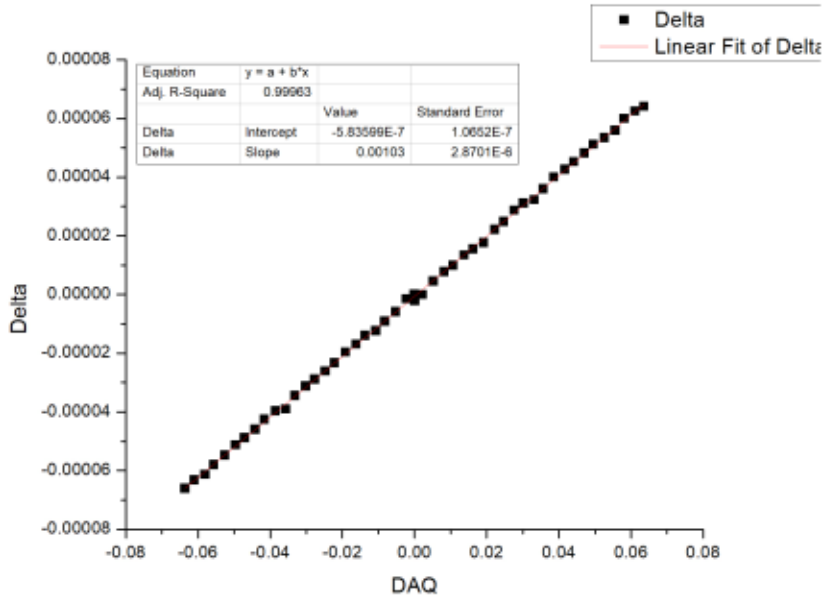
FONTE: Autor (2019)

FIGURA 100 – CH3 - GANHO 2- AD7794 CONTRA REFERENCIA



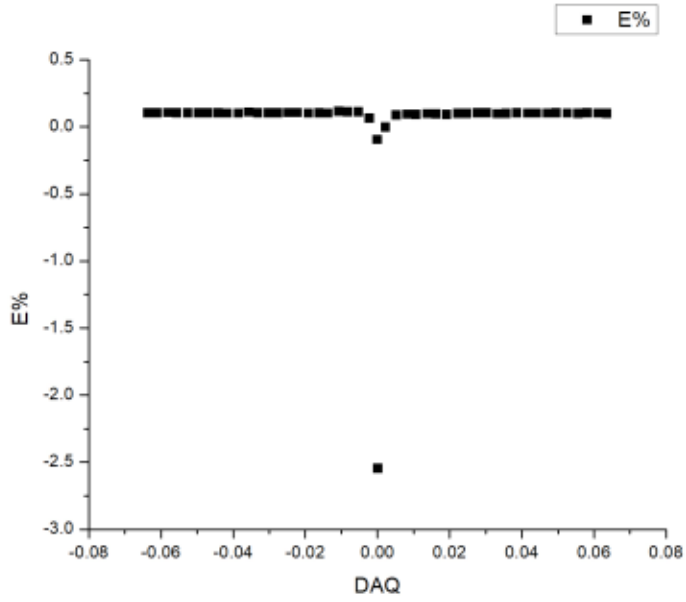
FONTE: Autor (2019)

FIGURA 101 – CH3 - GANHO 2- DIFERENÇA ENTRE AD7794 E REFERENCIA



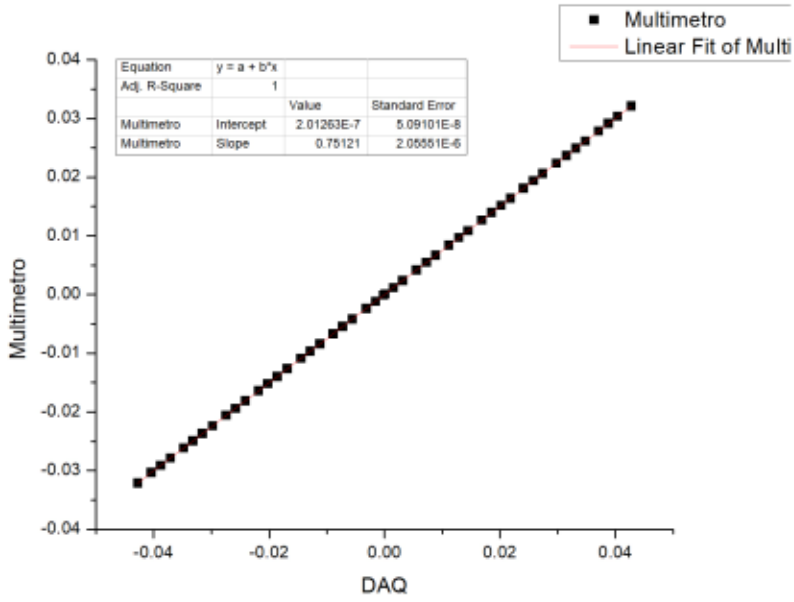
FONTE: Autor (2019)

FIGURA 102 – CH3 - GANHO 2- ERRO PERCENTUAL



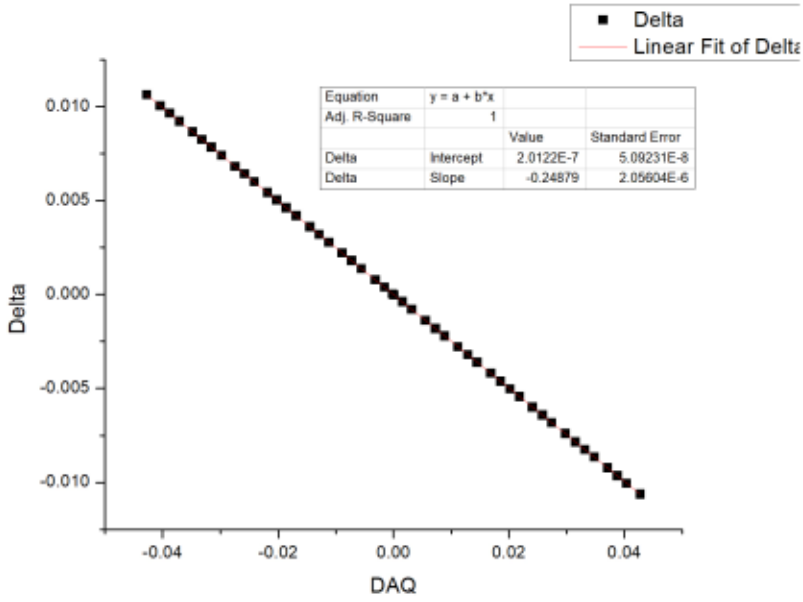
FONTE: Autor (2019)

FIGURA 103 – CH3 - GANHO 4- AD7794 CONTRA REFERENCIA



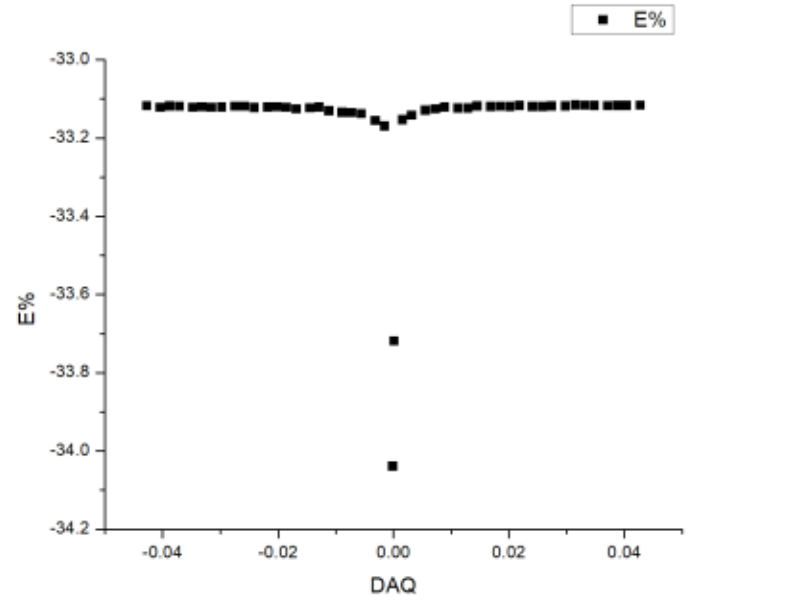
FONTE: Autor (2019)

FIGURA 104 – CH3 - GANHO 4- DIFERENÇA ENTRE AD7794 E REFERENCIA



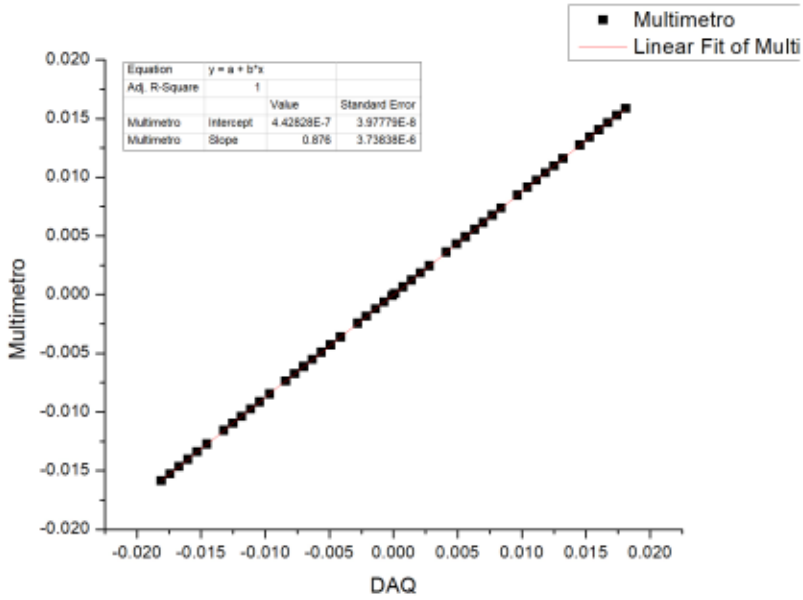
FONTE: Autor (2019)

FIGURA 105 – CH3 - GANHO 4- ERRO PERCENTUAL



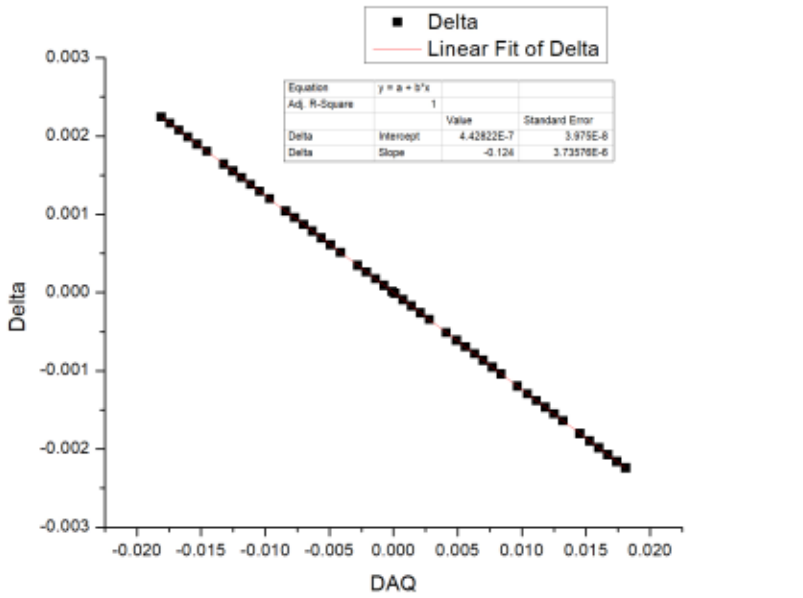
FONTE: Autor (2019)

FIGURA 106 – CH3 - GANHO 8- AD7794 CONTRA REFERENCIA



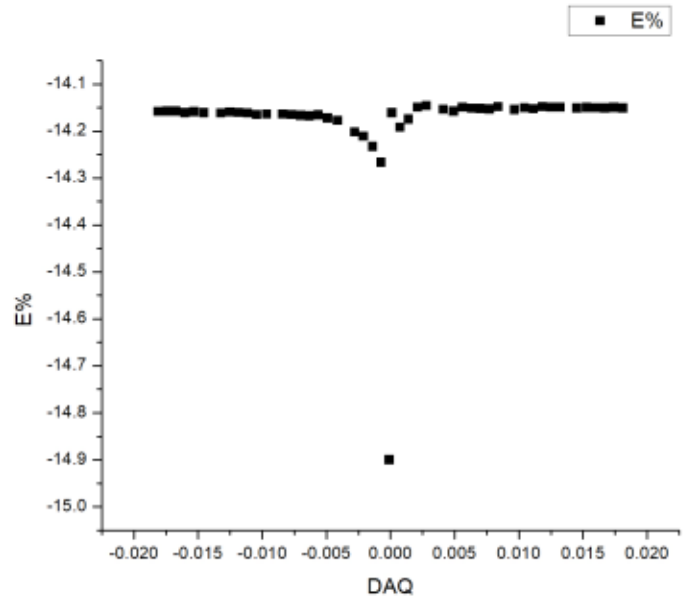
FONTE: Autor (2019)

FIGURA 107 – CH3 - GANHO 8- DIFERENÇA ENTRE AD7794 E REFERENCIA



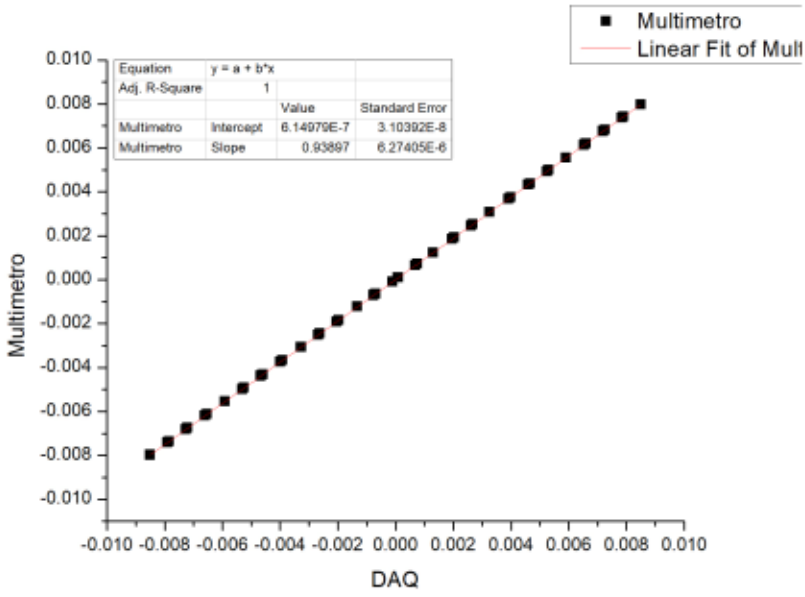
FONTE: Autor (2019)

FIGURA 108 – CH3 - GANHO 8- ERRO PERCENTUAL



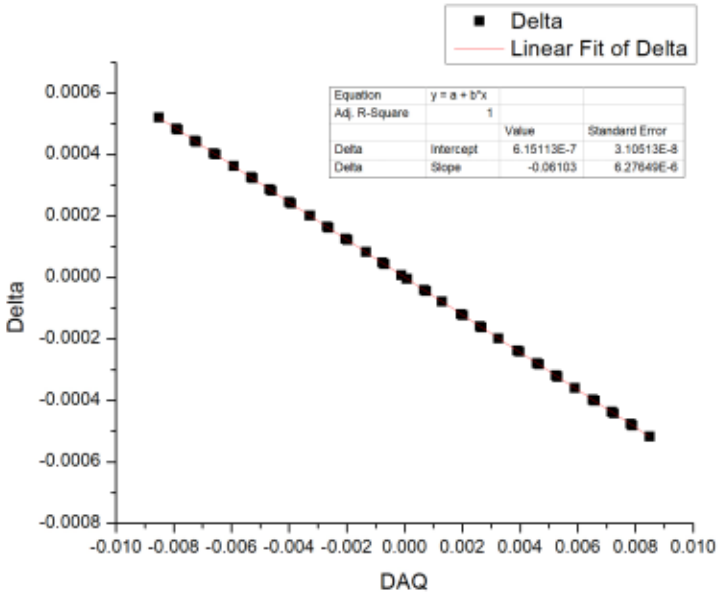
FONTE: Autor (2019)

FIGURA 109 – CH3 - GANHO 16- AD7794 CONTRA REFERENCIA



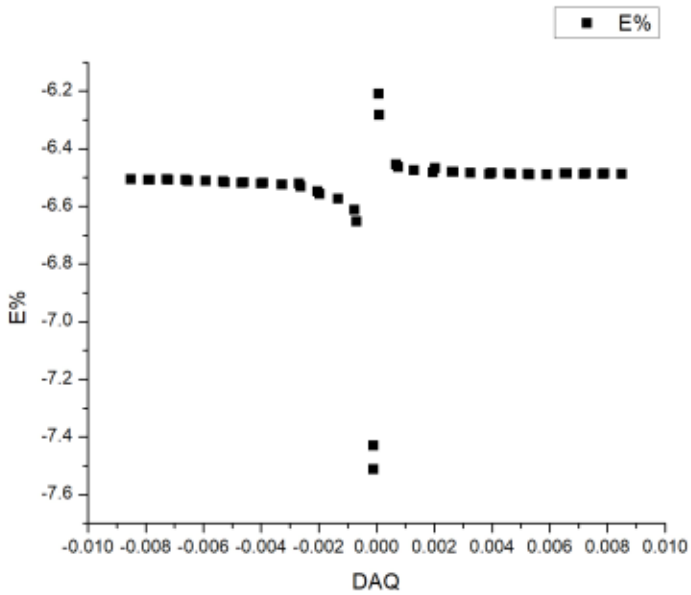
FONTE: Autor (2019)

FIGURA 110 – CH3 - GANHO 16- DIFERENÇA ENTRE AD7794 E REFERENCIA



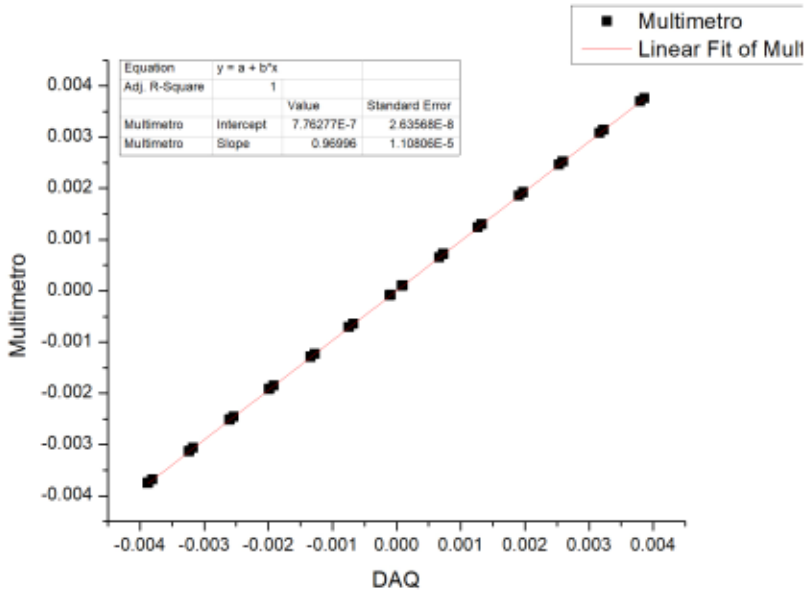
FONTE: Autor (2019)

FIGURA 111 – CH3 - GANHO 16- ERRO PERCENTUAL



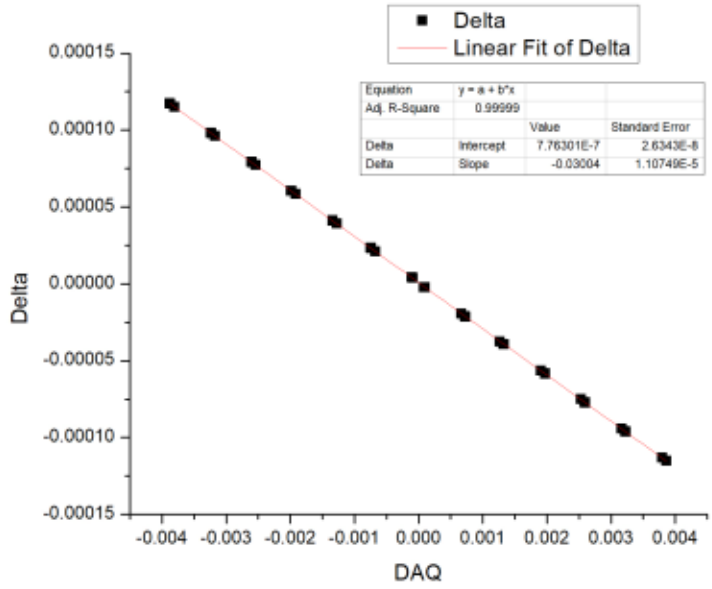
FONTE: Autor (2019)

FIGURA 112 – CH3 - GANHO 32- AD7794 CONTRA REFERENCIA



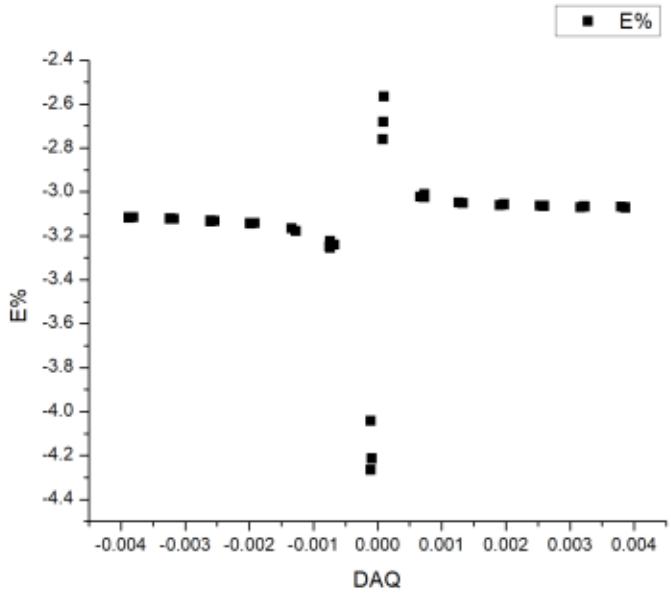
FONTE: Autor (2019)

FIGURA 113 – CH3 - GANHO 32- DIFERENÇA ENTRE AD7794 E REFERENCIA



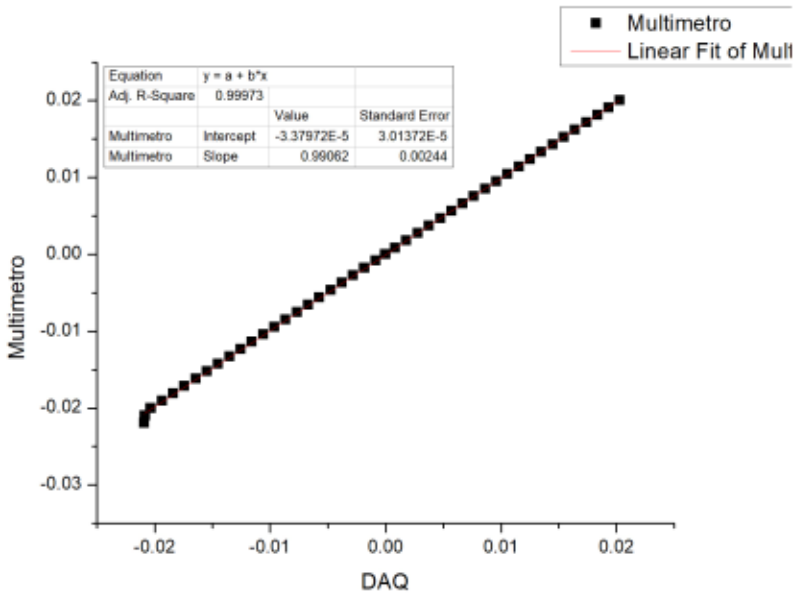
FONTE: Autor (2019)

FIGURA 114 – CH3 - GANHO 32- ERRO PERCENTUAL



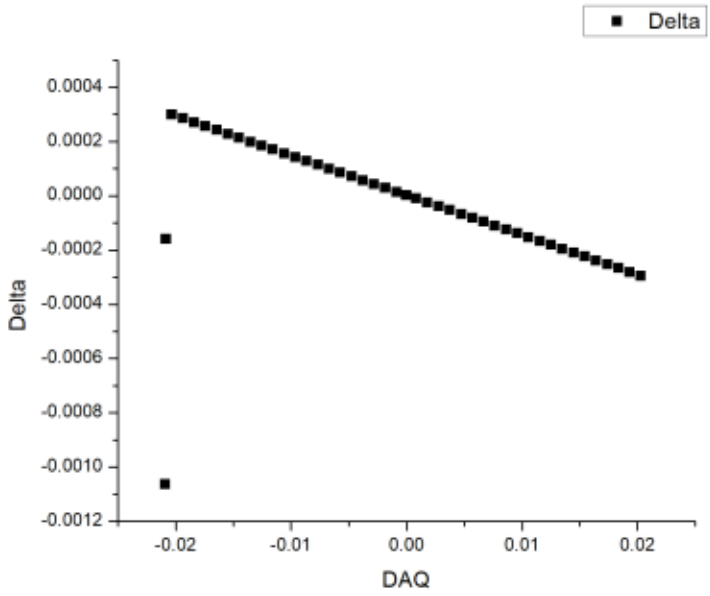
FONTE: Autor (2019)

FIGURA 115 – CH3 - GANHO 64- AD7794 CONTRA REFERENCIA



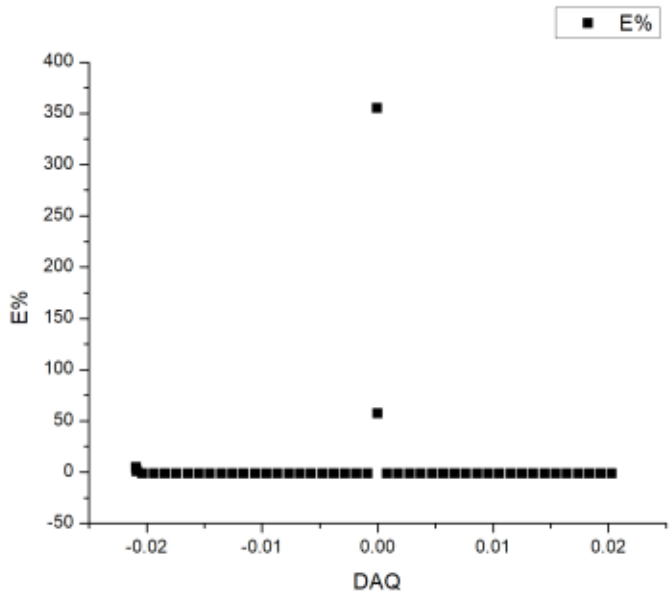
FONTE: Autor (2019)

FIGURA 116 – CH3 - GANHO 64- DIFERENÇA ENTRE AD7794 E REFERENCIA



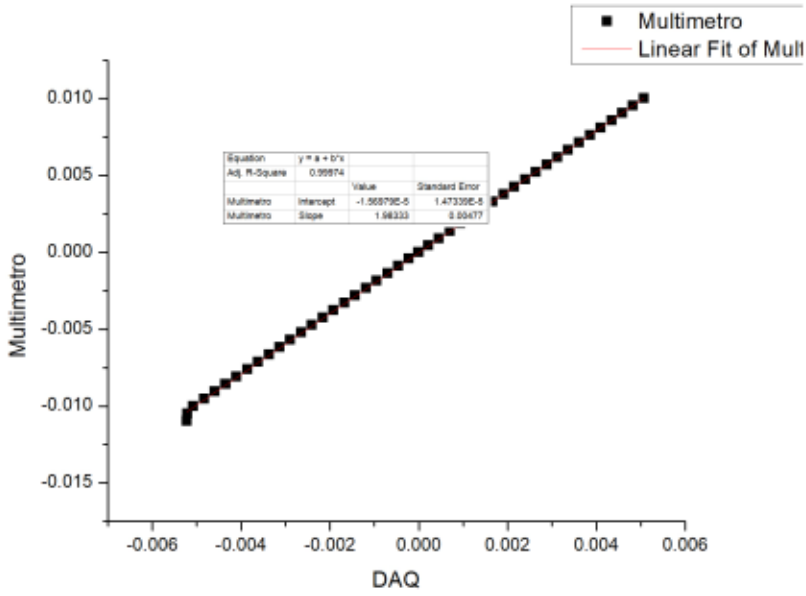
FONTE: Autor (2019)

FIGURA 117 – CH3 - GANHO 64- ERRO PERCENTUAL



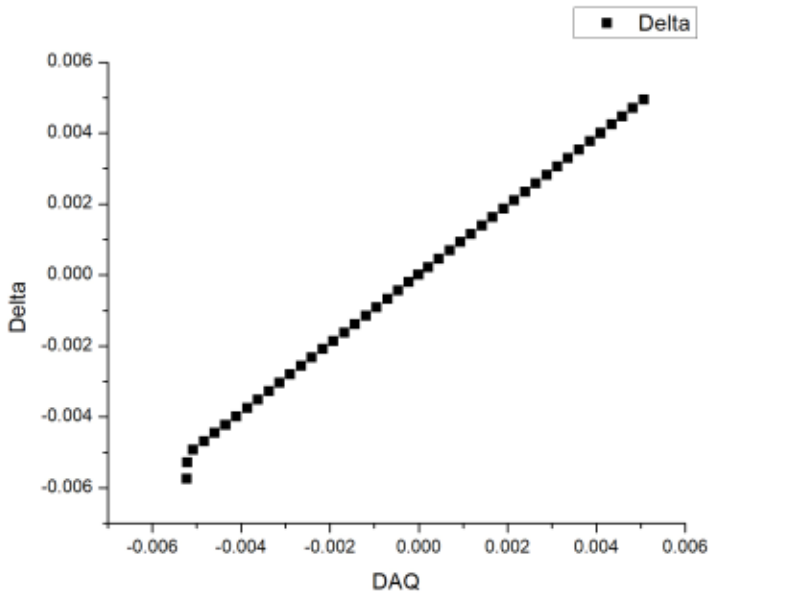
FONTE: Autor (2019)

FIGURA 118 – CH3 - GANHO 128- AD7794 CONTRA REFERENCIA



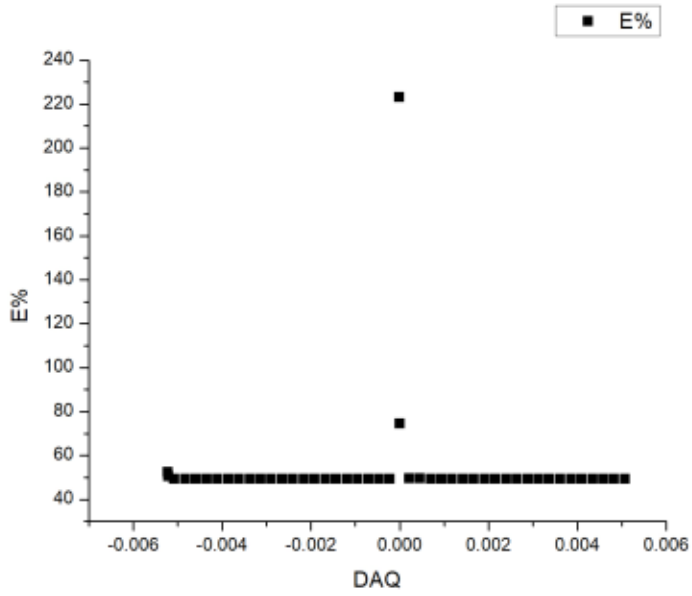
FONTE: Autor (2019)

FIGURA 119 – CH3 - GANHO 128- DIFERENÇA ENTRE AD7794 E REFERENCIA



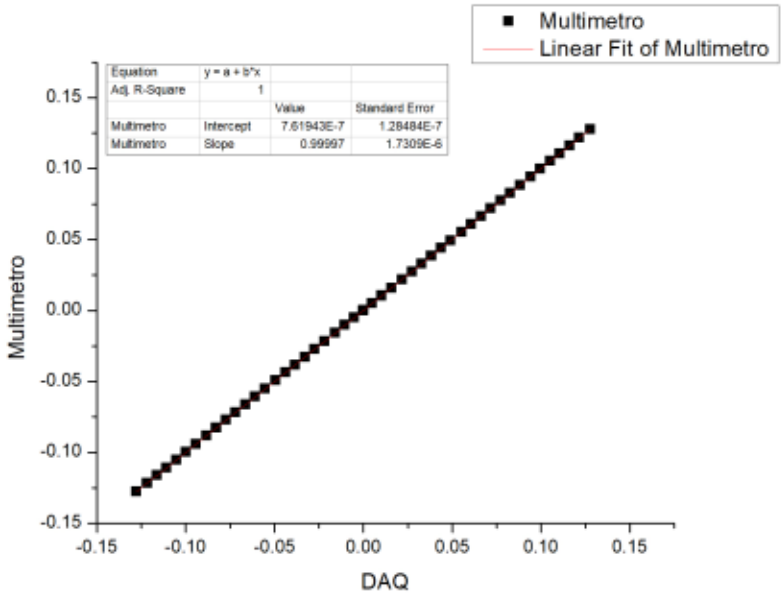
FONTE: Autor (2019)

FIGURA 120 – CH3 - GANHO 128- ERRO PERCENTUAL



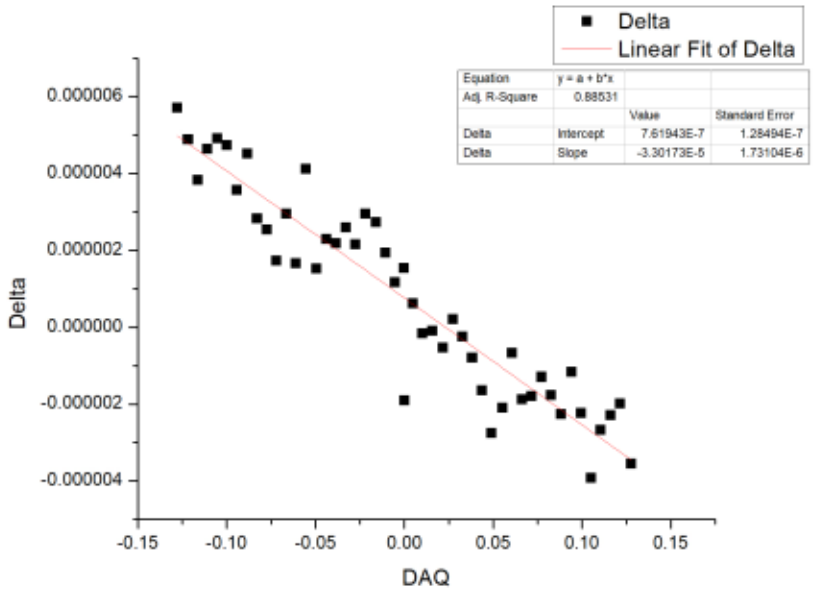
FONTE: Autor (2019)

FIGURA 121 – CH5 - CH5 - GANHO 1- AD7794 CONTRA REFERENCIA



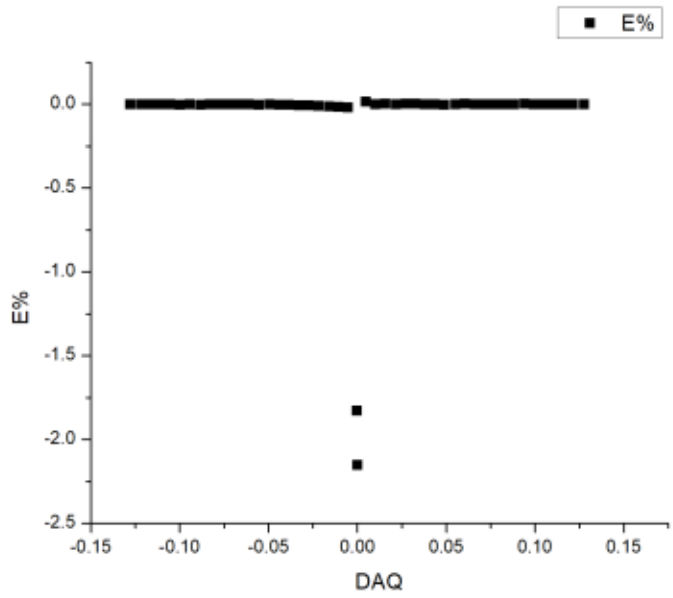
FONTE: Autor (2019)

FIGURA 122 – CH5 - GANHO 1- DIFERENÇA ENTRE AD7794 E REFERENCIA



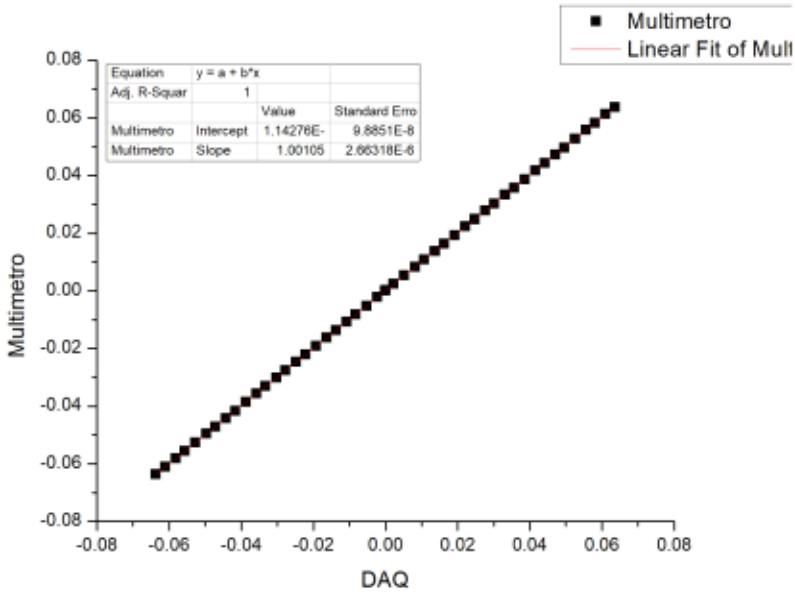
FONTE: Autor (2019)

FIGURA 123 – CH5 - GANHO 1- ERRO PERCENTUAL



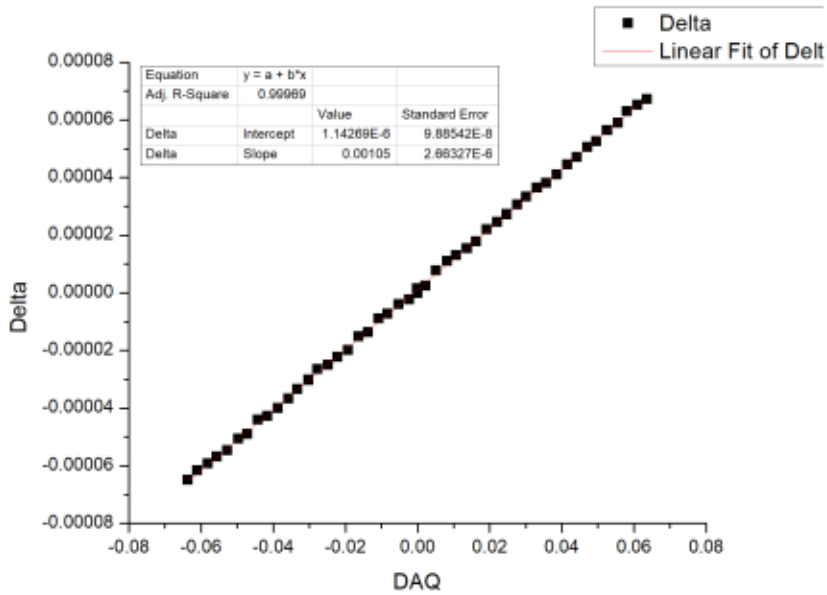
FONTE: Autor (2019)

FIGURA 124 – CH5 - GANHO 2- AD7794 CONTRA REFERENCIA



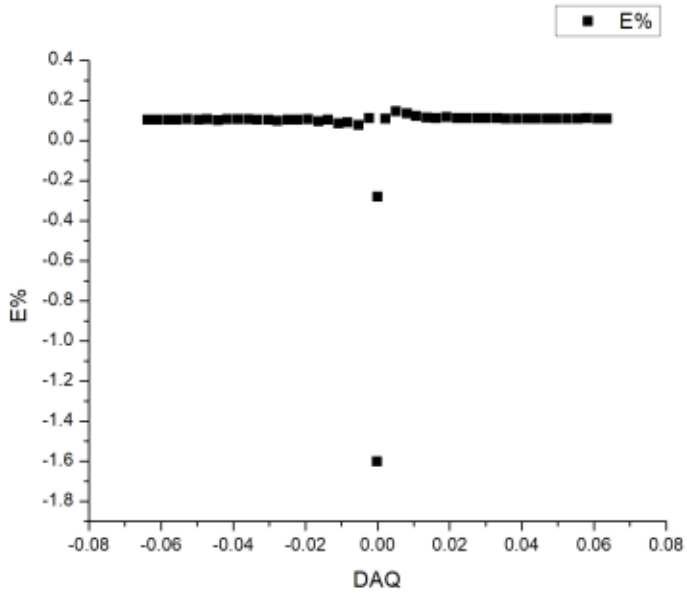
FONTE: Autor (2019)

FIGURA 125 – CH5 - GANHO 2- DIFERENÇA ENTRE AD7794 E REFERENCIA



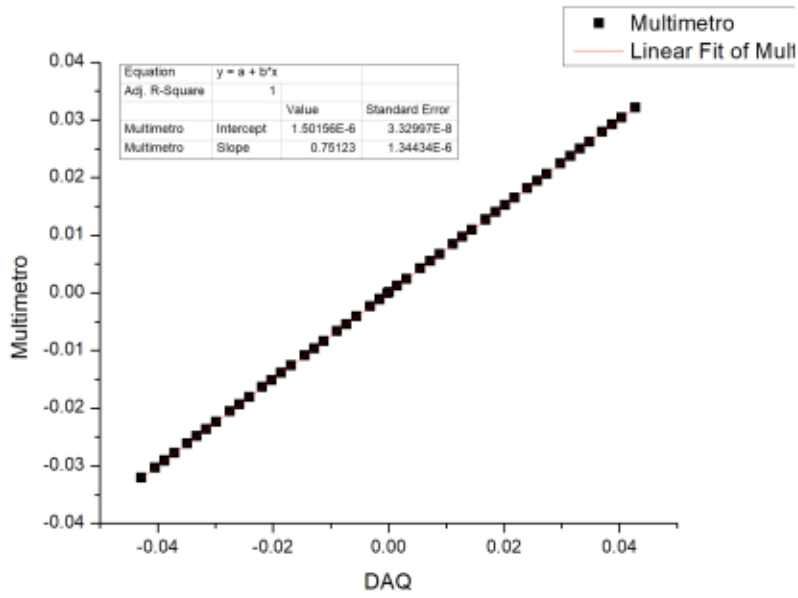
FONTE: Autor (2019)

FIGURA 126 – CH5 - GANHO 2- ERRO PERCENTUAL



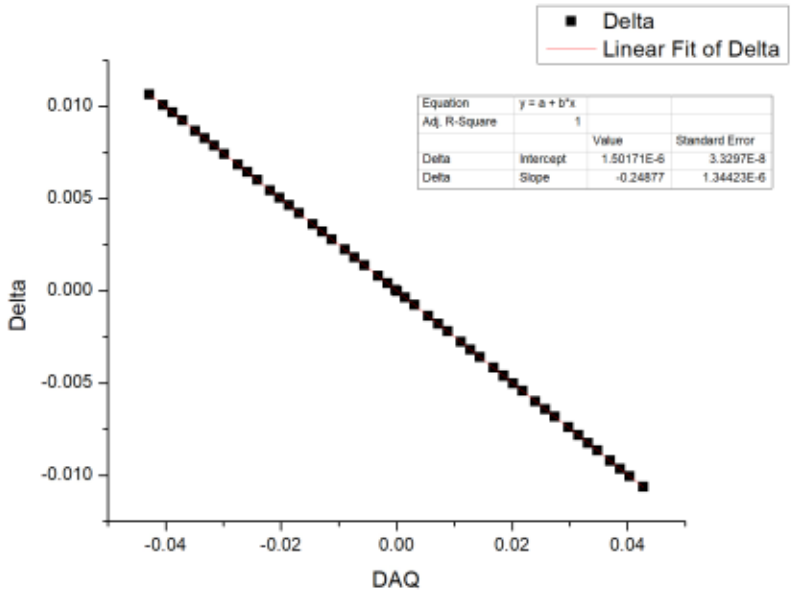
FONTE: Autor (2019)

FIGURA 127 – CH5 - GANHO 4- AD7794 CONTRA REFERENCIA



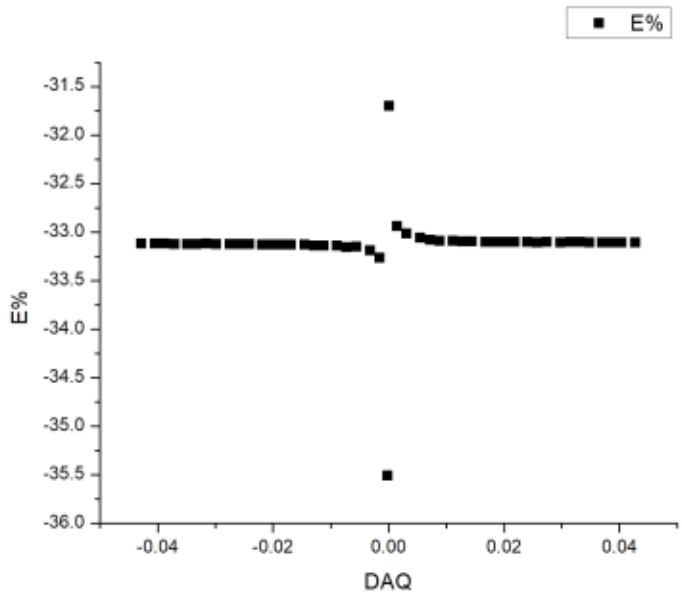
FONTE: Autor (2019)

FIGURA 128 – CH5 - GANHO 4- DIFERENÇA ENTRE AD7794 E REFERENCIA



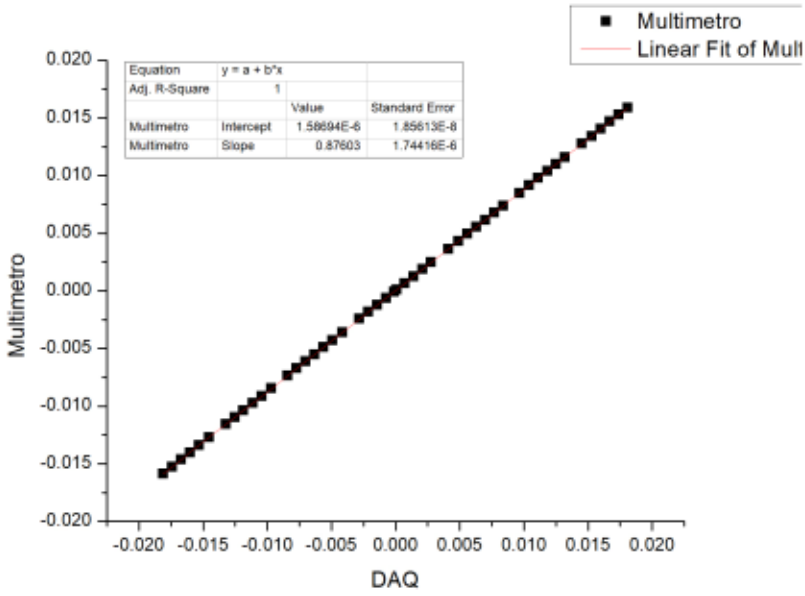
FONTE: Autor (2019)

FIGURA 129 – CH5 - GANHO 4- ERRO PERCENTUAL



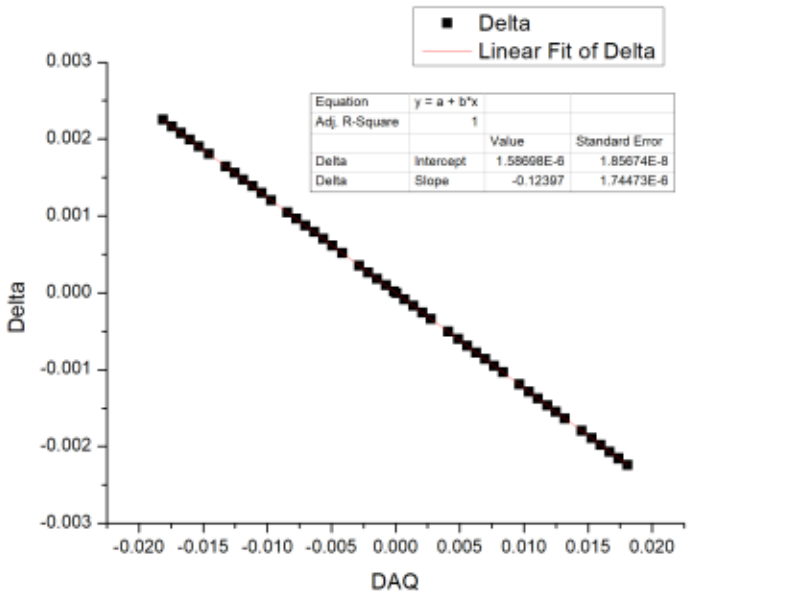
FONTE: Autor (2019)

FIGURA 130 – CH5 - GANHO 8- AD7794 CONTRA REFERENCIA



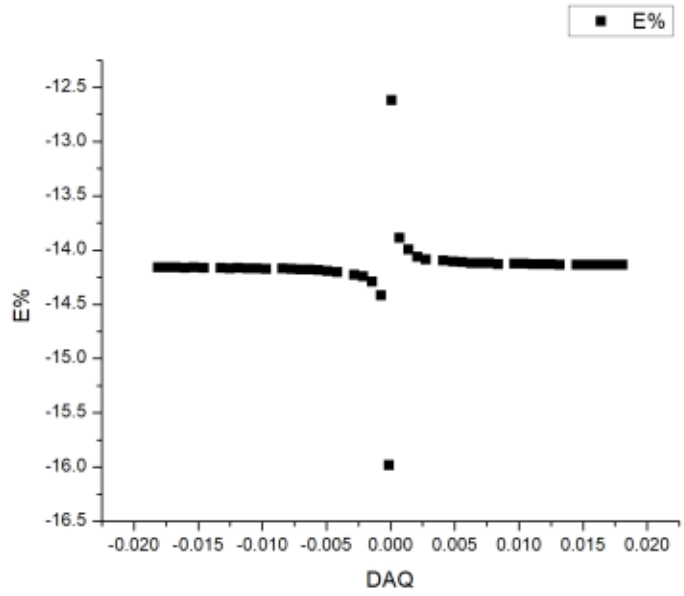
FONTE: Autor (2019)

FIGURA 131 – CH5 - GANHO 8- DIFERENÇA ENTRE AD7794 E REFERENCIA



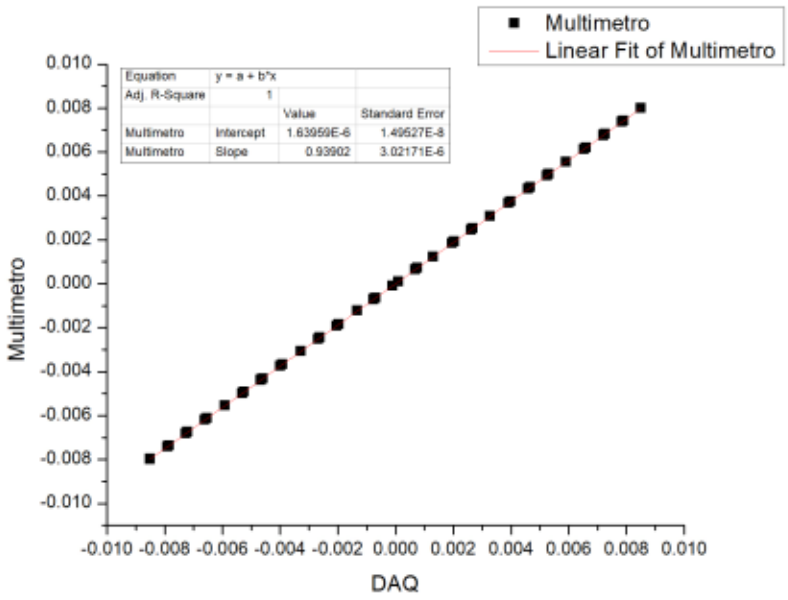
FONTE: Autor (2019)

FIGURA 132 – CH5 - GANHO 8- ERRO PERCENTUAL



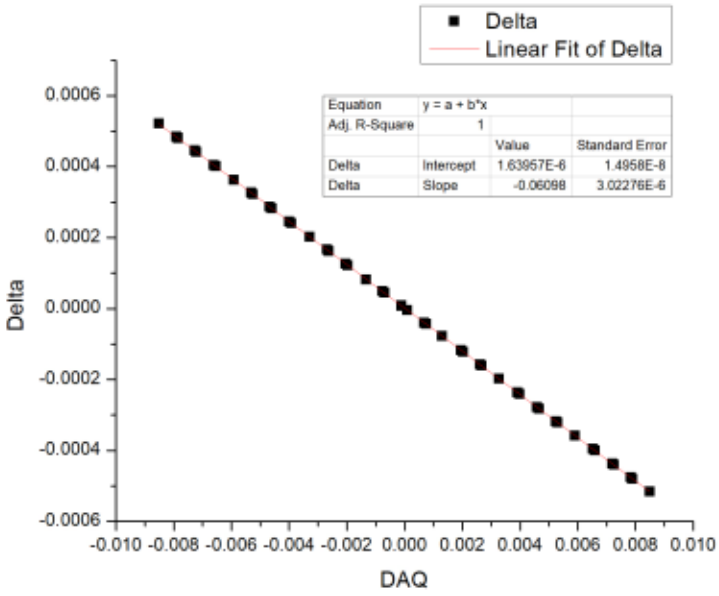
FONTE: Autor (2019)

FIGURA 133 – CH5 - GANHO 16- AD7794 CONTRA REFERENCIA



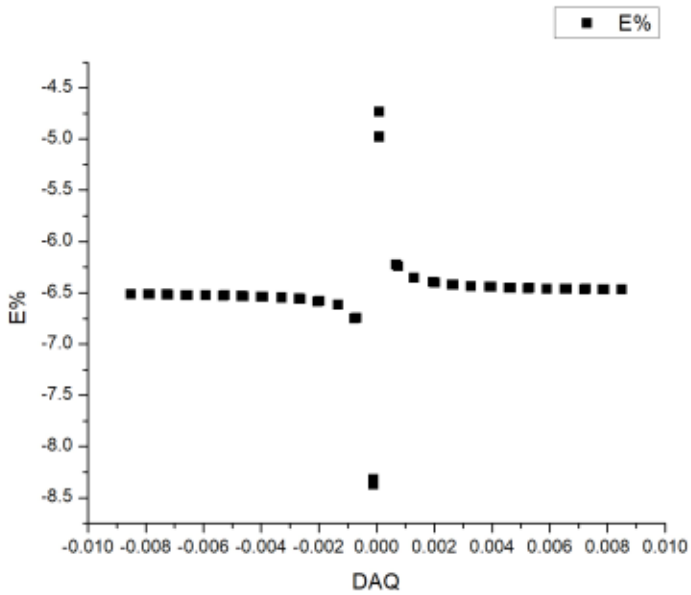
FONTE: Autor (2019)

FIGURA 134 – CH5 - GANHO 16- DIFERENÇA ENTRE AD7794 E REFERENCIA



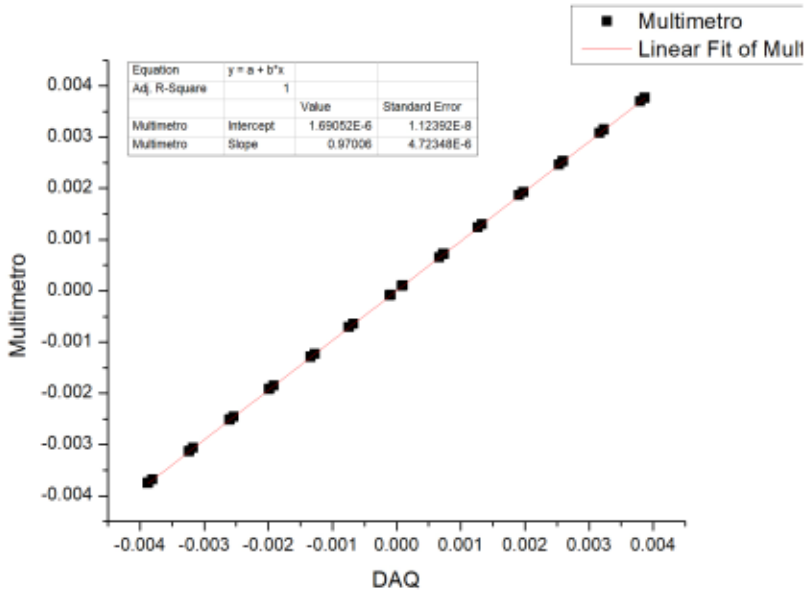
FONTE: Autor (2019)

FIGURA 135 – CH5 - GANHO 16- ERRO PERCENTUAL



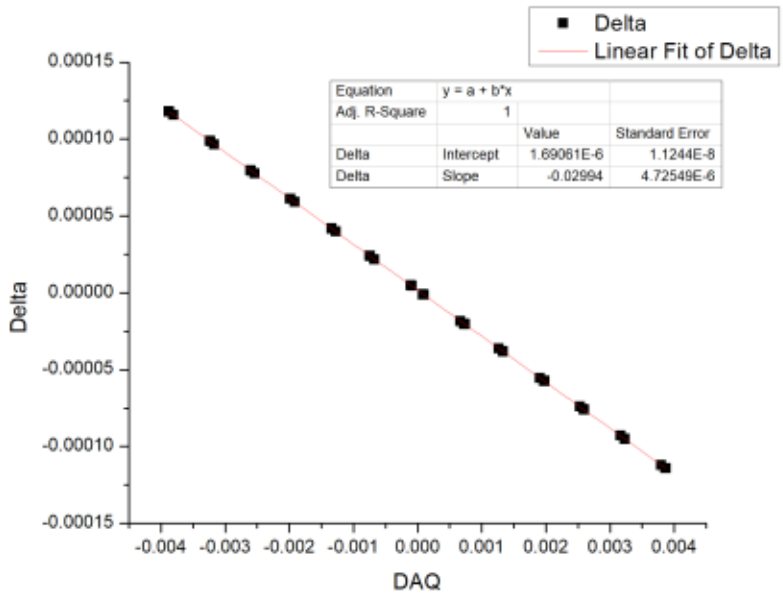
FONTE: Autor (2019)

FIGURA 136 – CH5 - GANHO 32- AD7794 CONTRA REFERENCIA



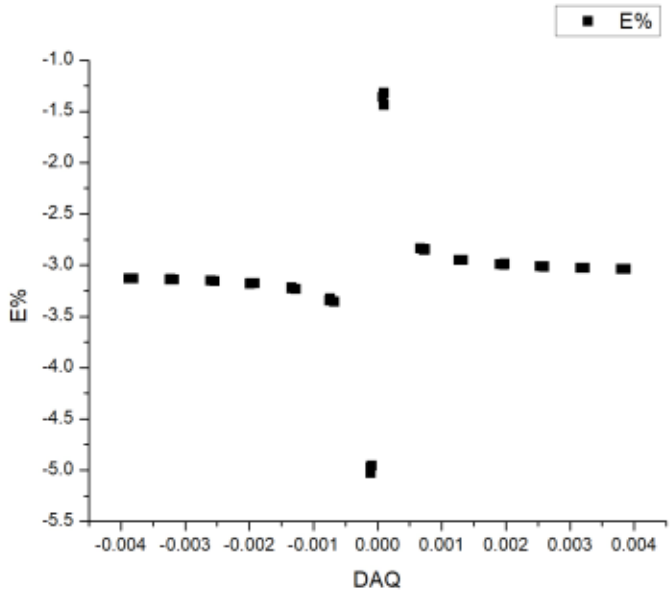
FONTE: Autor (2019)

FIGURA 137 – CH5 - GANHO 32- DIFERENÇA ENTRE AD7794 E REFERENCIA



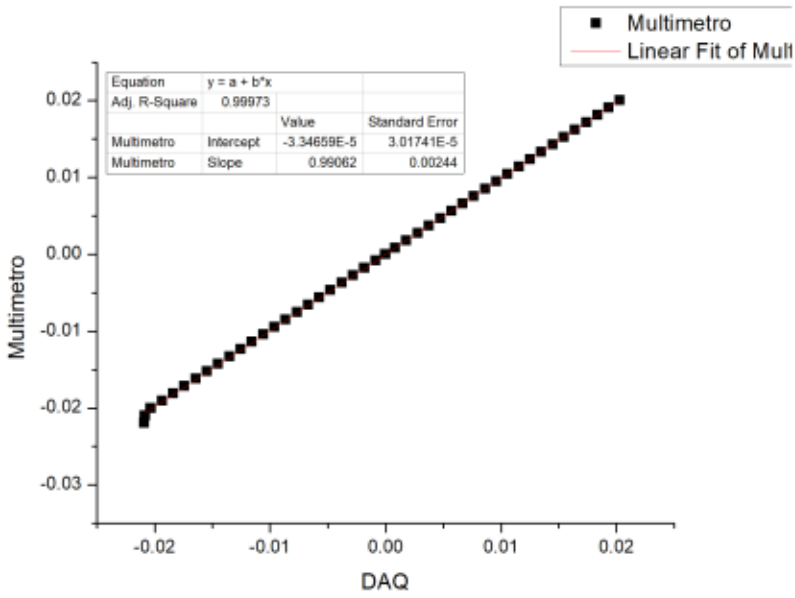
FONTE: Autor (2019)

FIGURA 138 – CH5 - GANHO 32- ERRO PERCENTUAL



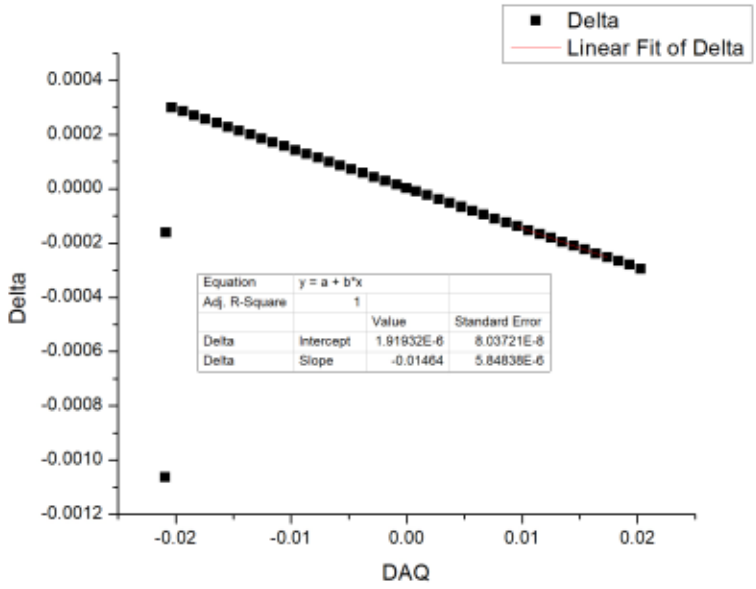
FONTE: Autor (2019)

FIGURA 139 – CH5 - GANHO 64- AD7794 CONTRA REFERENCIA



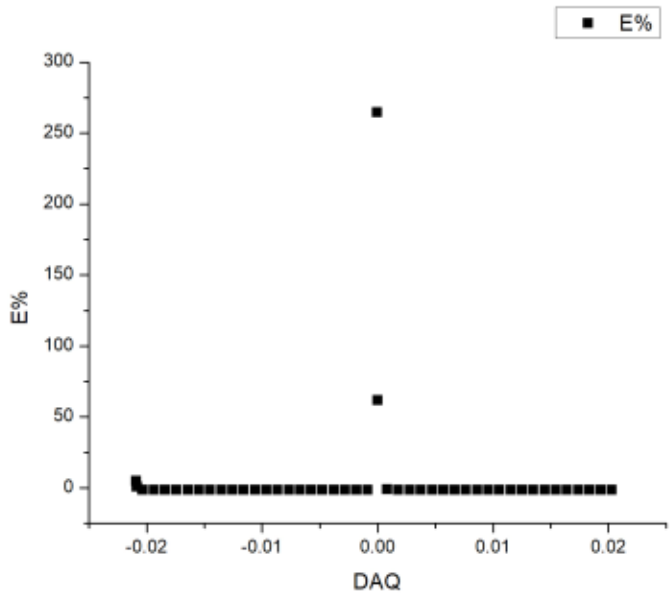
FONTE: Autor (2019)

FIGURA 140 – CH5 - GANHO 64- DIFERENÇA ENTRE AD7794 E REFERENCIA



FONTE: Autor (2019)

FIGURA 141 – CH5 - GANHO 64- ERRO PERCENTUAL



FONTE: Autor (2019)