Uma Aplicação Didática do Protocolo I2C em Sistemas de Comunicação

A Didactic Application of the I2C Protocol in Communication Systems

DOI:10.34117/bjdv7n10-001

Recebimento dos originais: 07/09/2021
Aceitação para publicação: 04/10/2021

Guilherme Araujo Machado do Nascimento
Graduando em Engenharia Mecatrônica
Universidade Federal de Santa Catarina
R. Dona Francisca, 8300 - Joinville - SC
E-mail: guilhermeamn1@gmail.com

Manoel Morais Lemos Neto
Graduando em Engenharia Mecatrônica
Universidade Federal de Santa Catarina
R. Dona Francisca, 8300 - Joinville - SC
E-mail: manoel.ctrl@hotmail.com

WyllianBezerra da Silva
Doutor em Engenharia Elétrica e Informática Industrial
Universidade Federal de Santa Catarina
R. Dona Francisca, 8300 - Joinville - SC
E-mail: wyllian.bs@ufsc.br

RESUMO
A fim de ampliar os conceitos abordados sobre sistemas de comunicação, este trabalho abordará uma aplicação didática do protocolo I2C, utilizando como hardware as placas Arduino e Raspberry Pi 3. Além disso, também serão implementados os métodos de detecção de erros checksum e CRC, bem como uma análise estatística dos resultados obtidos.

Palavras-chave: Protocolo I2C; Arduino; Raspberry Pi; checksum; CRC.

ABSTRACT
In order to extend the concepts covered about communication systems, this paper will address a didactic application of the I2C protocol, using as hardware the Arduino and Raspberry Pi 3 boards. In addition, the checksum and CRC error detection methods will also be implemented, as well as a statistical analysis of the results obtained.

Key-words: I2C Protocol; Arduino; Raspberry Pi; checksum; CRC.

1 INTRODUÇÃO

O I2C ou I²C (Inter-Integrated Circuit), foi criado em 1982 como um simples sistema de barramento interno para a construção de controles eletrônicos em vários chips Philips. Hoje, este protocolo se encontra amplamente difundido pelo mundo, implementado em mais de 1000 CIs (Circuitos Integrados) fabricados por mais de 50 empresas distintas. Devido à sua versatilidade, ele é usado em várias arquiteturas de controle tais como System Management Bus (SmBus), Power Bus (PMBus), Intelligent Platform Management Interface (IPMI), Advanced Telecom Computing Architecture (ATCA), entre outras (NPX Conductors, 2014).

O Protocolo I2C é utilizado principalmente para conectar periféricos de baixa velocidade e placas-mães, microcontroladores e afins (Romulo Camara, 2013). Para isso é necessário que haja suporte para esse protocolo, tanto pela unidade controladora quanto pelos periféricos, seja via software por meio da técnica bit-bang, a qual emula o protocolo bit a bit, ou via hardware no próprio SoC (System On Chip) ou utilizando CIs externos como o SC16IS750 (NPX Conductors, 2014).

Neste trabalho é apresentada uma maneira de conectar um Raspberry Pi (Raspberry Pi Ltd, 2019) a um Arduino UNO (Arduino, 2014) por meio deste protocolo, além de uma análise gráfica de dois métodos de detecção de erros: checksum e CRC (Cyclic Redundant Check). Com o auxílio destes, é possível verificar o impacto causado pelo ruído em um sistema de comunicação.

Este trabalho está estruturado da seguinte maneira: na próxima seção serão apresentados aspectos relacionados à fundamentação teórica acerca do protocolo I2C e as técnicas de detecção de erros. Na sequência, serão apresentadas as seções de materiais e métodos e resultados e discussão. Por fim, a última seção apresenta as conclusões e trabalhos futuros.

1.1 FUNDAMENTAÇÃO TEÓRICA

Esta seção aborda aspectos teóricos acerca do Protocolo I2C e as técnicas de detecção de erros checksum e CRC, amplamente empregadas em sistemas de comunicação de dados.

1.1.1 Protocolo I2C

O I2C é um dos protocolos de comunicação serial síncronos mais usado no mundo (M. P. Kumar e U. Rani. Nelakuditi, 2019). É muito adequado para comunicações entre

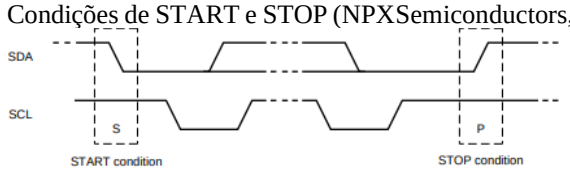
periféricos integrados para transferência de dados de baixa e média velocidade. Este módulo é extensamente utilizado em vários controladores, sensores e alguns outros circuitos integrados (Liu et al., 2019).

Ele é um barramento serial multimestre, ou seja, é capaz de possuir múltiplos mestres e é usado principalmente para conectar periféricos de baixa velocidade utilizando apenas duas linhas de transmissão, uma SDA (Serial Data Line) responsável pela transmissão de dados bidirecionalmente e uma SCL (Serial Clock Line) com o clock gerado pelo mestre. Além disso, é importante ressaltar que o mesmo trabalha com tensões entre +5 V e +3,3 V.

Cada dispositivo conectado ao barramento é um software endereçável por um endereço único e simples, representando a relação mestre/escravo estabelecida de modo permanente; mestres podem operar como transmissores-mestres ou como mestres-receptores (Irazabal e Blozis, 2003). Oprotocolo também é capaz de detectar colisões, algo comum para a topologia barramento com mais de um mestre, prevenindo que o sistema pare abruptamente.

O mestre é o dispositivo que inicia a transferência de dados no barramento e gera o clock que permite a comunicação. Nesse instante, qualquer dispositivo endereçado é considerado um escravo. As condições de START e STOP são sempre geradas pelo mestre, que funcionam da seguinte forma: Após o START o barramento é considerado ocupado, e algum tempo após STOP ter sido identificado o barramento é considerado disponível. A Figura 1 ilustra detalhes acerca deste processo.

Figura 1. Condições de START e STOP (NPXSemiconductors, 2014)



Todo byte colocado na linha de transmissão SDA, deve ter 8 bits de comprimento.

No entanto, não há restrição para quantos bytes que ela pode transmitir e cada byte deve ser acompanhado por um bit de reconhecimento. Os dados são transmitidos com o MSB (Most Significant Bit) primeiro. Se um escravo não puder receber ou transmitir um pacote de dados completo, pois estava realizando outra função, por exemplo, atendendo a solicitação de uma interrupção interna, ele poderá manter o SCL com sinal baixo, forçando um estado de espera no mestre. Assim, a transferência continua quando o

escravo estiver pronto para outro conjunto de dados e liberará o SCL (NPX Semiconductors, 2014).

1.1.2 Técnicas de Detecção de Erros

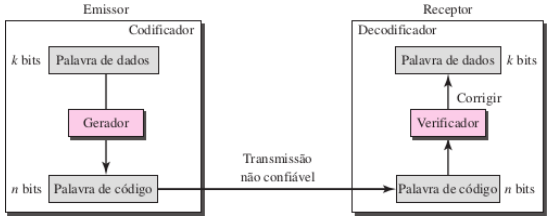
Sistemas de comunicação devem ser capazes de transmitir dados de maneira precisa, e para algumas situações é necessário que os dados enviados e recebidos sejam idênticos. No entanto, nem sempre isso ocorre, este fato se dá devido a uma série de fatores que podem alterar um ou mais bits de uma mensagem. Portanto, para esse tipo de situação é crucial a aplicação de mecanismos de detecção e correção de erros (Forouzan,2010).

A informação que chega devidamente no receptor é denominada transinformação, enquanto a parte perdida durante a transmissão é chamada de equivocação. Já a parte da informação que não tem ligação alguma com a fonte é chamada de dispersão (Rochol, 2012). São termos importantes que auxiliam no entendimento da composição da mensagem, e como a dispersão não possui relação com a mensagem original ela pode ser facilmente emulada.

O conceito mais importante na detecção e correção de erros é a redundância (Forouzan, 2010). Para que um sistema de comunicação seja capaz de detectar oucorrigir erros, é preciso enviar alguns bits extras redundantes junto com os dados. Esses bits redundantes são acrescentados pelo emissor e posteriormente retirados pelo receptor. Sua presença possibilita que o receptor detecte ou corrija bits corrompidos (Forouzan, 2010).

Em se tratando de código de blocos, as mensagens são divididas em blocos de tamanho igual a k, e cada um deles recebe então o nome de palavra de dados (datawords), que no geral é a mensagem gerada na fonte. Adicionando a estes blocos r bits de redundância serão obtidos novos com o valor n = k +r, os quais serão chamados de palavras de código (codeword), que será transmitida pelo canal de comunicação. A Figura 2 é a representação gráfica doque foi supracitado.

Figura 2. Estrutura do codificador e decodificador nacorreção de erros (Forouzan, 2010)



Para abordar a detecção de erros destas palavras serão abordadas duas técnicas, sendo elas: checksume CRC.

1.1.3 Técnica CRC

Baseado na teoria de códigos de correção de erros cíclicos, o CRC altera o tamanho da mensagem acrescentado bits de redundância para que posteriormente no decodificador seja possível identificar e corrigir os erros. Ele se mostra bastante efetivo na detecção de erros em rajadas. Graças aessa característica, é utilizado em redes LANs (Local Area Networks) e WANs (Wide Area Networks) (Forouzan, 2010).

Antes de apresentar o funcionamento do algoritmo é importante explicitar alguns termos e suas respectivas representações:

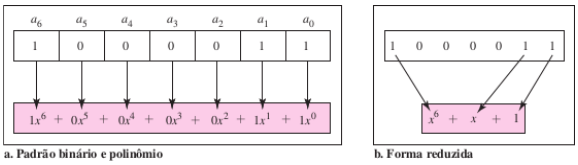
- Palavra de dados: d(x);
- Palavra de código: c(x);
- Síndrome: s(x);
- Erro: e(x);
- Gerador: g(x).

De acordo com Forouzan (2010), o algoritmo do CRC pode ser descrito da seguinte forma: no codificador, a palavra de dados tem k bits e a palavra de código tem n bits. Logo, o tamanho da palavra de dados é aumentado adicionando zeros aos n – k bits ao lado direito da palavra. O resultado de n bits alimenta o gerador. O gerador usa um divisor de tamanho n – k + 1, predefinido e estabelecido por ambas as partes. O gerador divide a palavra de dados aumentada pelo divisor (divisão de módulo 2). O quociente da divisão é descartado; o resto (r2r1r0) é anexado à palavra de dados para criar a palavra de código. O decodificador recebe a palavra de código possivelmente corrompida. Uma cópia de todos os n bits é alimentada no verificador, que é uma réplica dogerador. O resto produzido pelo verificador é uma síndrome de n – k bits que alimenta o analisador lógico de decisão. O analisador tem uma função simples. Se os bits de síndrome forem todos zeros, osn – k + 1 bits mais à esquerda da palavra de código serão aceitos como palavras de dados (interpretado como não sendo um erro); caso contrário, os n – k +1 serão descartados, devido à presença de erro.

A operação de divisão é realizada utilizando a operação XOR (OU exclusivo) representada simbolicamente por $\oplus$. O mesmo processo descrito pode ser melhorado com

a utilização de polinômios, pois dessa forma é possível criar uma simplificação significativa da palavra de código. Este processo requer a conversão das palavras escritas na forma de zeros e uns para polinômios. Considera-se todos os bits da palavra de dados como coeficientes de um polinômio, o que permite eliminar todos os termos atrelados aos bits de valor zero. Quanto aos de valor um, estes representarão um valor do tipo x^i, em que i é o grau do polinômio, cujos valores são decrescentes da esquerda para a direita da palavra de dados, e o maior grau possível é dado pelo número de bits contidos nela, menos um. A Figura 3 demonstra um exemplo apresentado em (Forouzan, 2010).

Figura 3. Representação polinomial de uma palavrabinária (Forouzan, 2010).



O exemplo ilustra como pode ser realizada a representação polinomial de uma palavra binária, fazendo a conversão do número para o seu equivalente em polinômio.

1.1.4 Checksum

Assim como o CRC, o checksum também se baseia no conceito de redundância.

Seu conceito é simples e consiste no envio de pacotes de palavras de mesmo tamanho com o valor de sua soma de todas as palavras. No receptor será recalculada essa soma e será efetuada uma comparação com o valor da soma recebida, caso sejam iguais, não há presença de erro, em caso de discrepância entre os valores, presume-se que ocorreu um erro. O método pode também ser usado para verificação na transmissão de imagens, como uma forma de pré-classificação das imagens recebidas pelo receptor da comunicação em aplicações de reconhecimento de imagem e visão computacional (Fedorov, 2012).

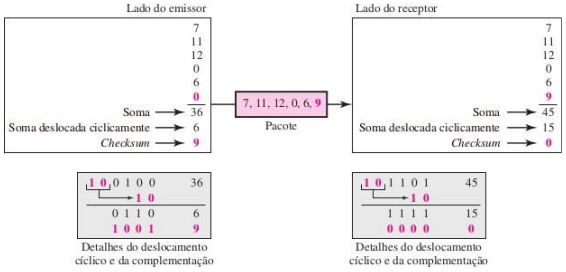
Segundo (Forouzan, 2010) é possível facilitar o trabalho do receptor da seguinte maneira: se for enviado o complemento da soma, denominada checksum. O receptor pode somar todos os números recebidos (inclusive a soma de verificação).

Se o resultado for zero, supõe-se que não existem erros; caso contrário, um erro foi detectado. Além disso, é necessário o uso da aritmética de complemento de um. Nesta aritmética, é possível representar números sem sinal entre 0 e $2^n - 1$, usando apenas n bits. Se o número tiver mais de n bits, os bits extras mais à esquerda precisam ser

adicionados aos n bits mais à direita (wrapping). Na aritmética de complemento um, um número negativo pode ser representado invertendo-se todos os bits (transformando 0 em 1 e 1em 0). Isso é o mesmo que subtrair o número por 2^n –

1- A Figura 4 é um exemplo do funcionamento deste algoritmo, demonstrando como o checksum é calculado e enviado para o receptor.

Figura 4. Exemplo prático do algoritmo *checksum* (Forouzan, 2010)

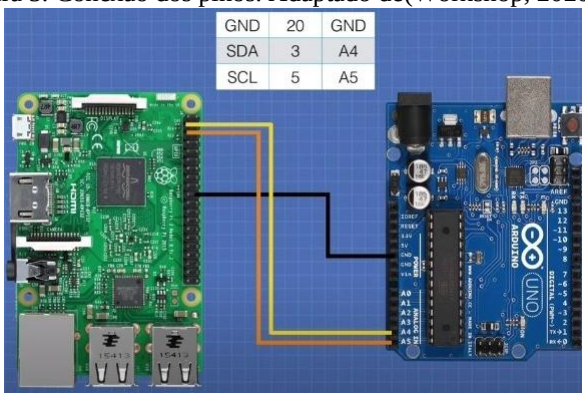


Apesar de sua simplicidade, o checksum é considerado um dos métodos menos caros de detecção de erros (Saxena e McCluskey, 1990), por isso ainda é amplamente utilizado na Internet. No entanto, aos poucos ele vem sendo substituído pelo CRC. Pois seu modelo tradicional conta com um número pequeno de bits (16) (Forouzan, 2010). Ademais, ele também é incapaz de detectar, por exemplo, quando uma palavra é incrementada e outra é decrementada pelo mesmo valor, os dois erros não serão detectados, visto que a soma e o checksum permanecerão idênticos. Da mesma forma que, se os valores de várias palavras forem incrementados, mas se a mudança total for múltipla de 65535, a soma e o checksum serão alterados, o que significa que erros não serão detectados (Forouzan, 2010).

2 MATERIAL E MÉTODOS

Os hardwares utilizados neste trabalho foram: Raspberry Pi 3 (Raspberry Pi Ltd, 2019), Arduino UNO (Arduino, 2014), 3 jumpers macho-fêmea e uma fonte de alimentação de 5 V e 2,5 A. Quanto aossoftwares, foram utilizados o sistema Raspberry Pi OS (Raspberry Pi OS, 2021), o framework PyQT5 (The Qt Company, 2021) e a Arduino IDE (Arduino IDE, 2021). A conexão entre os hardwares foi realizada conforme o esquemático ilustrado na Figura 5. O GND (ground) de ambos foram conectados um ao outro, a saída SDA do Raspberry foi conectada à entrada analógica A4 do Arduino e a saída SCL conectada à entrada A5 do Arduino.

Figura 5. Conexão dos pinos. Adaptado de(Workshop, 2020)



A Figura 6 ilustra a montagem física utilizada para o experimento, de acordo com o esquemático.

Figura 6. Montagem dos hardwares



Para realizar a comunicação, basta o Raspberry estar conectado à fonte de 5 V, o Arduino conectado ao notebook e os dispositivos conectados entre si conforme mostrado.

2.1 IMPLEMENTAÇÕES

O dispositivo Raspberry Pi foi configurado como mestre durante toda a execução, enquanto o Arduino atuou como escravo. Principalmente em razão da diferença no suporte de tensão dos dois componentes,visto que, o Raspberry funciona a +3,3 V se o Arduino enviasse uma tensão de +5 V, o miniPC poderia ser danificado.

O codificador e decodificador para o CRC pode ser implementado tanto em hardware quanto em software (Panda e Kumar, 2014). Neste trabalho, as implementações foram realizadas em software. No Arduino, por ter suporte à Linguagem C++, duas classes foram criadas para representar os métodos, com seus respectivos nomes. A classe CRC possui como atributos a codeword, o gerador e as variáveis n, k e r, além de métodos

para calcular divisão entre binários, operação lógica XOR e um método que representa a decodificação. A classe checksum possui como atributos o somatório dos valores do pacote, o tamanho da palavra e o valor do checksum, além de métodos que calculam a soma do pacote recebido, convertem binário para decimal e vice-versa, calculam o complemento de um número binário, somam dois binários e um método que representa o receptor da comunicação.

O código cíclico CRC foi dividido em duas partes: uma para o codificador e também mestre, Raspberry Pi 3 (Raspberry Pi Ltd, 2019), e outra para o decodificador e escravo, Arduino UNO (Arduino, 2014). A coleta de dados necessária para o processo de codificação pode ser resumida da seguinte maneira: O usuário insere por meio da interface a dataword e o gerador desejado, e de acordo com o modo escolhido dentre os disponíveis: CRC, CRC-8 ou CRC-10. Para o caso dos três últimos, um geradorpadrão é automaticamente preenchido na interface para facilitar o processo.

Contudo o usuário permanece com a liberdade de alterá-lo, mas é importante salientar que o uso de um gerador inadequado afeta significativamente na eficiência do algoritmo. Portanto é aconselhável não alterá-los de maneira arbitrária, a não ser é claro, para fins de testes. Este gerador obedece ao padrão determinado pelos polinômios descritos na Tabela 1.

Tabela 1. Polinômios-padrão. Adaptado de (Forouzan, 2010)

Nome	Polinômio
CRC-8	$X^8 + x^2 + x + 1$
CRC-10	$X^{10} + x^9 + x^5 + x^4 + x^2 + 1$
CRC-16	$X^{16} + x^{12} + x^5 + 1$
CRC-32	$X^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Por falta de suporte nos hardwares utilizados para o experimento, o CRC-16 e CRC-32 não puderam ser implementados. Por meio da interface, há a possibilidade de injetar ruído na transmissão, bem como o número de bits que serão afetados.

No processo de codificação, primeiramente o tamanho da palavra de dados é calculado, assim como a quantidade de bits de redundância. Este último, é calculado utilizando a expressão a seguir (Forouzan, 2010).

$$r = n - k, \qquad (1)$$

em que r a quantidade de bits de redundância, n o tamanho da palavra de código e k tamanho da palavrade dados. Após os cálculos, são adicionados r zeros na dataword. Esta nova palavra com os bits de paridade adicionados é divida pelo gerador utilizando-se palavras de n – k + 1 bits. O resto dessadivisão é anexado à palavra de dados, criando a palavra de código, a qual será enviada para o decodificador por meio da comunicação com o protocolo I2C. Caso a opção de inserir erros tenha sido selecionada, uma função é chamada para gerar uma palavra composta por zeros e uma quantidade especificada pelo usuário de uns, gerados em posições aleatórias. A operação XOR é realizada entre a codeword correta e esta palavra gerada, para transmitir os erros gerados para a palavra de código. Esta então é finalmente enviada ao decodificador.

No decodificador, nesse caso o Arduino, a codeword é recebida assim como o gerador, devido ao fato de que o gerador deve ser o mesmo para os dois comunicadores. O mesmo cálculo das variáveis n, k er feitas no codificador, são também feitos aqui. A palavra de código recebida na comunicação I2C é dividida pelo gerador e o resto dessa divisão, tambémchamado de síndrome, é analisado. Caso o usuário tenha selecionado a opção de transmissão com ruído, haverá bits uns na síndrome e a dataword será descartada. Caso contrário, a síndrome será compostaapenas de bits zeros, e a dataword será aceita.

Para o método de detecção checksum, o código também foi dividido em duas partes, com o emissor sendo o dispositivo Raspberry Pi 3 (Raspberry Pi Ltd, 2019) e o receptor, o Arduino. Na interface, o usuário insere os números que compõem o pacote que deseja ser enviado e o checksum é inicializado com zero. O somatório destes valores é calculado, convertido para binário e então é feita a soma deslocada ciclicamente, utilizando palavras de quatrobits. Como a implementação foi feita com string, as últimas quatro posições são a parte superior da soma,e o restante das posições são a parte inferior.

Este valor é então passado por uma função que retorna o complemento de um número binário, sendo este retorno o valor do checksum. Por último, o dispositivo Raspberry envia o pacote inserido pelo usuário com o checksum para o Arduino.

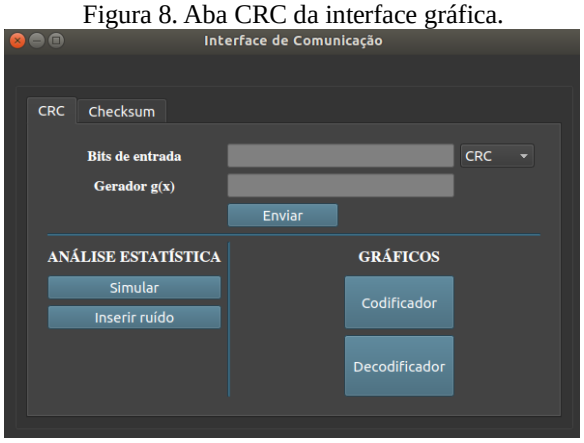
No lado do receptor, o pacote é recebido, sendo os primeiros números o pacote original digitado pelo usuário, e o último dígito o checksum. O processo realizado aqui é semelhante ao do emissor: a soma dos dígitos é calculada, convertida para binário, a soma deslocada ciclicamente é realizada e o complemento desta é calculado, da mesma forma

explicada anteriormente, e armazenado no checksum. A diferença aqui é a verificação do valor do checksum, a fim de analisar se há erros na transmissão. Caso este valor tenha dado diferente de zero, significa que os dados foram corrompidos.

2.2 INTERFACE GRÁFICA

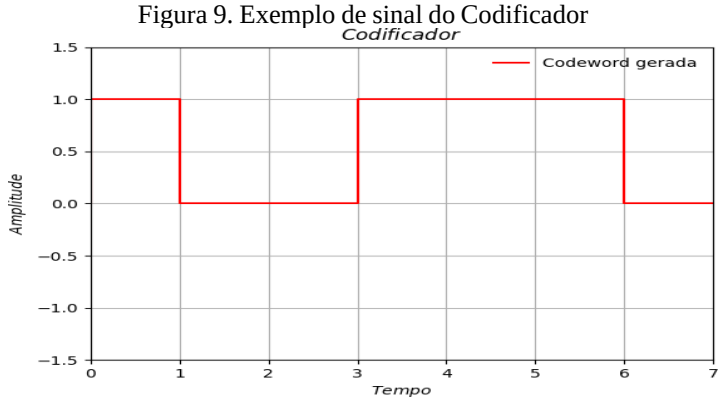
Com o auxílio do framework PyQt5 (The Qt Company, 2021), foi desenvolvida uma interface gráfica com aspecto amigável para o usuário, conforme ilustra a Figura 8.

Figura 8. Aba CRC da interface gráfica.



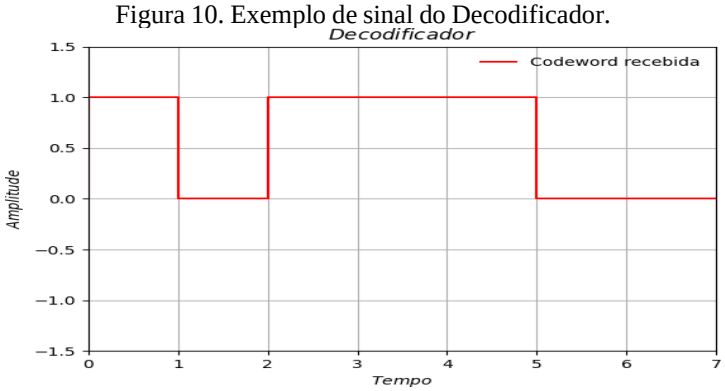
Para o CRC, há três modos que podem ser selecionados: CRC, CRC-8 e CRC-10. O usuário deve selecionar o modo desejado e entrar com os bits de entrada, assim como o gerador. Para o CRC-8 e CRC-10, ao serem selecionados o gerador é automaticamente preenchido com um gerador padrão, conforme a Tabela 1. Na área de **Análise Estatística** há a opção de **Inserir ruído**, que ao ser clicada aparece um espaço para inserir a quantidade de erros desejada. Esta opção irá gerar erros em posições aleatórias da codeword, antes de ser enviada ao receptor. Há também a opção de **Simular**, conforme será detalhado na seção Resultados. Na área de **Gráficos**, estão disponíveis as opções de visualizar o gráfico tanto do codificador quanto do decodificador. Para o CRC estes gráficos representam a palavra de código no emissor e no receptor, respectivamente. Na Figura 9, é possível ver o gráfico de um exemplo de codeword gerada no codificador.

Figura 9. Exemplo de sinal do Codificador



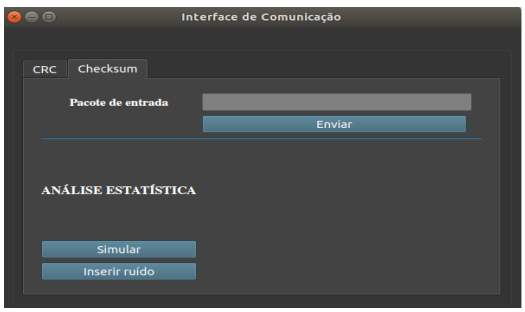
Já a Figura 10 ilustra o gráfico da codeword recebidapelo decodificador. O exemplo demonstra um caso em que há erros durante a transmissão. Vale observarque se não houver erros durante a transmissão, estes gráficos serão idênticos.

Figura 10. Exemplo de sinal do Decodificador.



Para o modo checksum, a interface oferece espaço para o usuário entrar com o pacote que deseja transmitir, conforme ilustra a Figura 11. Nesta implementação, a transmissão recebe pacotes de cinco dígitos.

Figura 11. Aba checksum da interface gráfica.



Na área de **Análise Estatística** há a opção de **Simular**, a qual será explicada mais detalhadamente na Seção Resultados.

Assim como no outro modo, há a opção **Inserir ruído**, a qual para esse modo irá gerar um valor aleatório entre 1 e 10 para somar ao *checksum* previamente calculado, antes de enviá-lo ao receptor.

3 RESULTADOS E DISCUSSÃO

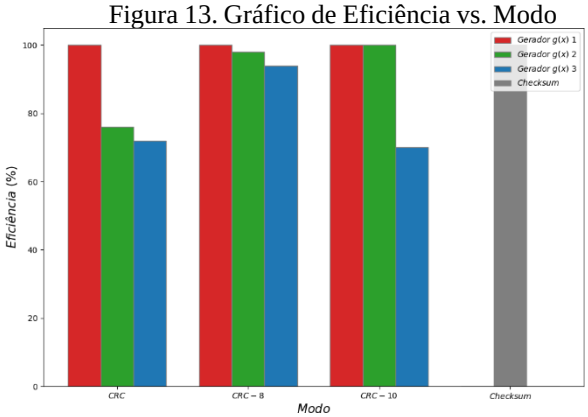
Um modo de simulação foi implementado para melhorar a visualização dos resultados dos métodos de detecção de erros. Na interface há a opção de **Simular** (vide Figura 11), a qual gera cem datawords aleatórias para o modo CRC escolhido. Destas, 50% utiliza dois erros injetados e os outros 50% sem erro. O gerador deve ser fornecido pelo usuário, de acordo com o CRC escolhido. Por meio desse procedimento é possível analisar a porcentagem de sucesso na detecção dos erros no receptor na implementação deste trabalho. Para o checksum o processo é semelhante, porém dessa vez um dígito aleatório é somado ao pacote. Dessa forma, cem pacotes são gerados aleatoriamente, com metade deles possuindo um erro injetado. Os dados necessários para análise estatística são gravados pelo Arduino em um arquivo de texto. Um código na Linguagem Python é responsável por ler estes arquivos gerados pelo Arduino, extrair a quantidade de erros detectados e calcular a eficiência do gerador, para o caso do CRC e da implementação, para o caso do checksum, conforme Equação (2) a seguir.

$$eficiência = \frac{erros\ detectados}{erros\ injetados} \quad (2)$$

Por fim, outro script é responsável pelo plot do gráfico, contendo os resultados da simulação tanto das versões disponíveis do CRC quanto do checksum. A Figura 13 representa a saída de uma simulação de exemplo, ilustrando os resultados obtidos de eficiência na detecção de erros utilizando três geradores diferentes para cada um dos métodos CRC, CRC-8 e CRC-10. Para a realização dos testes, os geradores utilizados para o CRC foram 1011, 1001 e 0001. Para o CRC-8 os geradores utilizados foram 100000111, 100000001 e 000000001. Para o último modo, os geradores 1100011011, 0100000011 e 0100000000 foram escolhidos. Para os casos do CRC-8 e CRC-10 o primeiro gerador foi escolhido, de acordo com a Tabela 1. Para os outros geradores a escolha foi feita baseada nos critérios de eficiência de um polinômio gerador, os quais são: ter pelo menos

dois termos; o coeficiente do termo x^0 deve ser 1; o polinômio não deve ser divisível por $x^t + 1$, sendo tentre 2 e n - 1; o polinômio deve ter o fator x + 1 (Forouzan, 2010).

Figura 13. Gráfico de Eficiência vs. Modo



Como o propósito deste trabalho é testar a eficiência de diferentes geradores, alguns desses critérios foram propositalmente descumpridos para o segundo e terceiro gerador de cada caso, como por exemplo, alguns possuem menos de dois termos, outros possuem o coeficiente do termo x nulo ou então não possuem o fator x + 1.

Para o caso do checksum, o gráfico serve apenas para ilustrar a eficiência da implementação do método, visto que não há um gerador interferindo na detecção de erros, mostrando quantos erros foram detectados em relação aos pacotes enviados, que como demonstra o gráfico foi de 100% dos casos.

4 CONCLUSÃO

A aplicação do protocolo I2C assim como visto na teoria, mostrou-se de fácil implementação, o que contribuiu para uma aplicação prática e didática dos métodos de detecção de erros: CRC e *checksum*. Além disso, por meio das simulações feitas e dos gráficos gerados, ficou nítida a importância de geradores adequados para o CRC, que para cada aplicação possuem suas vantagens e desvantagens. Quanto ao *checksum*, os dados obtidos foram exatamente os esperados, com uma precisão igual ou aproximadamente 100%, ou seja, dos 50 erros injetados foi possível captar todos ou praticamente todos. Com estes métodos de detecção de erros implementados, é possível garantir uma linha de transmissão com maior confiabilidade para o usuário, não apenas para o protocolo I2C, como para todos os outros. Para futuros trabalhos, a implementação polinomial desses métodos pode ser estudada, visto que neste trabalho foram abordados apenas com palavras binárias. Com a implementação em forma de polinômios pode ser estudada também a possibilidade de se implementar os métodos de CRC-16 e CRC-32, devido a

menor quantidade de caracteres que será enviada pela comunicação, fazendo com que os hardwares utilizados no projeto talvez suportem a quantidade de bits transportadas.

REFERÊNCIAS

Arduino. "Arduino UNO". https://datasheet.octopart.com/A000066-Arduino-dat asheet-38879526.pdf, 2014.

Arduino IDE. "Software Arduino". https://www.arduino.cc/en/software, 2021.

A. S. Panda e G. L. Kumar. "Comparison of serial data-input crc and parallel data-input crc design for crc-8 atm hec employing mlfsr". 2014 International Conference on Electronics and Communication Systems (ICECS), 2014.

B. A. Forouzan. "Comunicação de Dados e Redes deComputadores", 4th ed. McGraw-Hill, 2010.

C. Liu, Q. Meng, T. Liao, X. Bao, e C. Xu. "A flexible hardware architecture for slave device of i2c bus". 2019 International Conference on Electronic Engineering and Informatics (EEI), 2019.

D. Fedorov. "Information technology of image recognition using checksums". Proceedings of International Conference on Modern Problem of Radio Engineering, Telecommunications and Computer Science, 2012.

D. Workshop. "I2c with arduino and raspberry pi - two methods". Youtube. [Online].https://www.youtube.com/watch?v=me7mhrRbspk&t =1366s, 2020.

J. Irazabal e S. Blozis. "I2C MANUAL", 24th ed. https://www.nxp.com/docs/en/application-note/AN10216.pdf, 2003.

J. Rochol. "Comunicação de Dados", 22nd ed. Bookman, 2012.

M. P. Kumar e U. Rani. Nelakuditi. "Iot and i2c protocol based m-health medication assistive system for elderly people". 2019 IEEE 16th India Council International Conference (INDICON), 2019.

N. Saxena and E. McCluskey. "Analysis of checksums, extended-precision checksums, and cyclic redundancy checks". IEEE Transactions on Computers, 1990.

NXP Semiconductors. "I2C-bus specification anduser manual", 6th ed. https://www.nxp.com/docs/en/user-guide/UM10204. pdf, 2014.

Raspberry Pi Ltd. "Raspberry Pi Compute Module3+". https://www.raspberrypi.org/documentation/hardware /computemodule/datasheets/rpi_DATA_CM3plus_1p 0.pdf, 2019.

Raspberry Pi OS. "Raspberry Pi OS". https://www.raspberrypi.org/software/, 2021.

Romulo Camara. “Protocolo I2C”. UNIVASF, http://www.univasf.edu.br/~romulo.camara/novo/wp-content/uploads/2013/11/Barramento-e-Protocolo-I2 C.pdf, 2013.

The Qt Company. “PyQt5 Reference Guide”. https://www.riverbankcomputing.com/static/Docs/PyQt5/, 2021.