Profibus

Tuane Sampaio Machado* Matheus Wilgen Gonçalves*
*Universidade Federal de Santa Catarina (UFSC)

***Resumo*—Replicar o protocolo de comunicação Profibus DP, utilizando dois microcontroladores Arduino Uno para simular a comunicação mestre-escravo, implementando o meio físico RS-485 por meio de um chip Max 485 e demais especificações do protocolo através de *software* e conexões físicas.**

***Palavras-chave*—Protocolos de comunicação, Detecção de erros, Sistemas de Automação, Propagação e Ondas.**

I. Introdução

O uso de automação em processos industriais é uma realidade em expansão graças aos inúmeros benefícios que ele traz para a indústria. Seu conceito se baseia na descentralização do controle dos processos produtivos, utilizando a tecnologia e a automação como forma de conectar todos os processos da operação. É claro que, para isso, se faz necessário ter a capacidade de comunicação entre os processos envolvidos de forma eficiente e segura.

Nesse cenário, a tecnologia da informação tornou-se determinante no desenvolvimento da tecnologia da automação. Para garantir que a descentralização funcione de fora eficiente, permitindo que seja possível replicar as técnicas de comunicação competentes em larga escala, se fez necessário o uso de mecanismos padronizados, abertos e transparentes que, com a evolução das técnicas de comunicação, se tornaram indispensáveis no conceito de automação atual.

Pensando na automação e na comunicação entre os processos envolvidos, é preciso memorar que, como as linguagens de cada um dos maquinários são diferentes, é necessário definir um conjunto de regras, ou seja, o modo como se dará a comunicação que atenda essa diversidade.

Sendo assim, foram criados protocolos, que são os padrões pelos quais é estabelecida a troca de dados. Basicamente, são códigos criados para que diferentes computadores possam "conversar", a fim de supervisionar e gerenciar uma série de processos diferentes. Atualmente existem diversas redes de comunicação disponíveis no mercado, dentre elas, uma das mais utilizadas é o protocolo Profibus.

Esse trabalho irá apresentar o protocolo de comunicação Profibus, suas tecnologias e o resultado da implementação do protocolo Profibus DP (*Decentralized Periphery*), utilizando dois microcontroladores Arduino Uno [1] para comunicação mestre-escravo e um chip Max 485 [2] para simular o meio fisíco desse protocolo.

II. Fundamentação Teórica

As redes industriais são uma forma de automação de indústria, que consistem em protocolos de comunicação usados para supervisionar e gerenciar processos com segurança e rapidez. Controlam a distribuição de dados que seguem em toda a planta industrial por meio de uma ágil e precisa troca de informações.

O Profibus (*Process Field Bus*) é um dos protocolos que fazem parte do grupo dos "fieldbuses" abertos e independentes de fornecedores (não-proprietários). Permitem a integração de equipamentos de diversos fabricantes em uma mesma rede, pois possuem como características a interoperabilidade (a capacidade de um sistema de se comunicar de forma transparente com outro sistema) e intercambialidade (possibilidade de utilização de um item, componente ou produto no lugar de outro(s), sem necessidade de adaptação ou ajustes para satisfazer os requisitos necessários) [3].

A. Tecnologias Profibus:

Como já mencionada, devido à grande expansão do uso do protocolo Profibus por redes industriais, foram criadas diversas tecnologias direcionadas para diversas aplicações para esse protocolo. Dessas, pode-se destacar três tipos principais que são comumente utilizadas pelo mercado atual: Profibus DP, Profibus PA e PROFINET.

1) Profibus DP: A tecnologia Profibus DP foi desenvolvido para operar com uma alta velocidade de comunicação, que pode chegar a 12Mb/s e com tempo de reação da ordem de um a cinco milisegundos. Sua conexão é de baixo custo e tem como objetivo principal fazer a conexão em equipamentos descentralizados de maneira acessível [4].

É utilizado na comunicação entre sistemas de controle de automação e seus respectivos I/O's distribuídos no nível de dispositivo. Pode ser usado para substituir a transmissão de sinal em 24 V em sistemas de automação de manufatura assim como para a transmissão de sinais de 4 a 20 mA ou HART em sistemas de automação de processo.

2) Profibus PA: Profibus PA é a solução Profibus que atende aos requisitos da automação de processos, onde se tem a conexão em processos com equipamentos de campo, tais como: transmissores de pressão, temperatura, conversores, posicionadores, etc. Possui uma característica adicional que é a transmissão intrinsecamente segura, o que faz com que ele possa ser usado em áreas classificadas, ou seja, ambientes onde existe o perigo de explosão. É indicado por controlar variáveis digitais em linhas de produção seriada ou células integradas de manufatura. Encontrado predominantemente nas indústrias de transformação, podendo ser usada em substituição ao padrão 4 a 20 mA [5].

O Profibus-PA foi desenvolvido em cooperação com os usuários da Indústria de Controle e Processo (NAMUR), satisfazendo as exigências especiais dessa área de aplicação:

- Perfil original da aplicação para a automação do processo e interoperabilidade dos equipamentos de campo dos diferentes fabricantes;

- Adição e remoção de estações de barramentos mesmo em áreas intrinsecamente seguras, sem influência para outras estações;
- Comunicação transparente através dos acopladores de segmento entre o barramento de automação do processo Profibus-PA e o barramento de automação industrial Profibus-DP;
- Alimentação e transmissão de dados sobre o mesmo par de fios baseado na tecnologia IEC 61158-2;
- Uso em áreas potencialmente explosivas com blindagem explosiva tipo "intrinsecamente segura" ou "sem segurança intrínseca".

Seu meio físico é definido pela norma IEC 61158-2. A transmissão síncrona, em conformidade à norma IEC 61158-2, possui uma taxa de transmissão definida em 31,25 Kbits/s.

3) Profinet: Essa rede é baseada em um padrão de comunicação Ethernet Industrial padronizado pelas normas IEC 61158-5 e IEC 61158-6 para conectar dispositivos de processo – sensores, atuadores e similares – aos sistemas de controle. Pode ser utilizado em aplicações em tempo real (rápidas) e em aplicações onde o tempo não é crítico, por exemplo, na conversão para rede Profibus DP.

Os cabos mais utilizados nas redes Profinet são coaxiais com 4 fios de cobre envoltos por uma blindagem, na Figura 1 pode-se ver como é a organização dos fios. Estes cabos suportam velocidades de 100 Mbps em uma distância de até 100 metros.

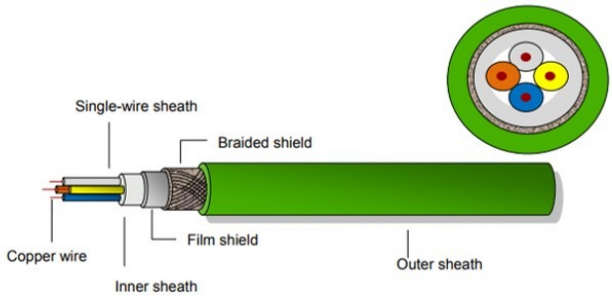

Figura 1. Cabo comumente utilizado nas redes Profinet.

B. Interfaces seriais

Existem duas possibilidades de meios físicos para Profibus DP: Elétrico em padrão RS-485 e Fibra ótica.

1) Fibra ótica.: A fibra ótica pode ser utilizada pelo Profibus para aplicações em ambientes com alta interferência eletromagnética ou mesmo com o objetivo de aumentar o comprimento máximo com taxas de transmissão elevadas. Vários tipos de fibra estão disponíveis, com diferentes características, tais como, distância máxima, preço e aplicação. Os segmentos Profibus que utilizam fibra normalmente são em estrela ou em anel.

2) RS-485 - Interface serial escolhida.: O protocolo Profibus DP utiliza o padrão normalizado RS-485, esse padrão não define nenhum protocolo, apenas descreve a parte física da comunicação, como especificar os níveis de tensão de operação, número de dispositivos e distância máxima.

O padrão RS485 é a tecnologia de transmissão mais frequentemente encontrada no Profibus. Sua aplicação inclui todas as áreas nas quais uma alta taxa de transmissão, aliada a uma instalação simples e barata, se faz necessária. Um par trançado de cobre blindado com um único par condutor é o suficiente neste caso [6].

É confiável em ambientes ruidosos, pois como demonstra a Figura 2 a utilização estratégica do sinal diferencial permite uma filtragem eficiente dos ruídos captados pelo cabo ao longo de seu comprimento, tornando o padrão RS-485 robusto a interferências. Outra vantagem do sinal diferencial é sua imunidade a variação do potencial de terra dos diferentes dispositivos da rede. Pequenas variações no potencial de terra não prejudicam a comunicação.

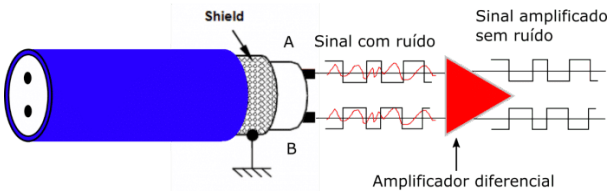

Figura 2. Amplificação de uma comunicação diferencial para filtrar ruído [7].

O padrão ainda pode operar em modo half-duplex utilizando um par de cabos padrão como é representado na Figura 3 ou em full-duplex utilizando dois pares de cabos, como mostra a Figura 4, sendo que em half-duplex somente é possível utilizar uma operação por vez no barramento, ou seja, as operações de envio ou recebimento de dados não podem ser realizadas no mesmo tempo. Já no modo full-duplex, é possível enviar e receber dados ao mesmo tempo.

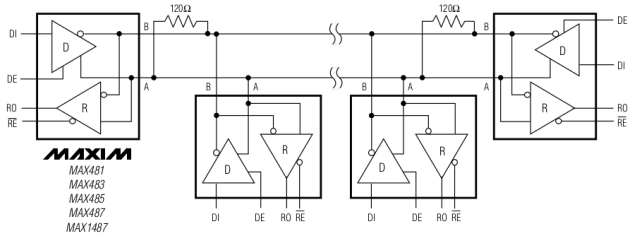

Figura 3. Configuração de comunicação em *half duplex* [2].

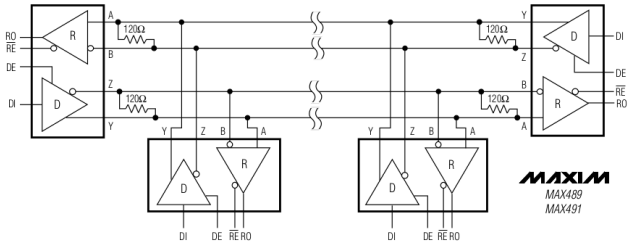

Figura 4. Configuração de comunicação em *full duplex* [2].

3) IEC 61158-2.: A IEC – Intenational Eletrotechical Committe, trata-se de uma organização não governamental

internacional para o desenvolvimento de padrões elétricos eletrônicos e relacionados, sendo responsável por muitos padrões internacionais.

Possui transmissão síncrona em conformidade à norma IEC 61158-2. Permite, além de segurança intrínseca, que os dispositivos de campo sejam energizados pelo próprio barramento. Assim, o PROFIBUS pode ser utilizado em áreas classificadas. As opções e limites do PROFIBUS com tecnologia de transmissão IEC61158-2 para uso em áreas potencialmente explosivas são definidos pelo modelo FISCO (Fieldbus Intrinsically Safe Concept) que é internacionalmente reconhecido como o modelo básico para barramentos em áreas classificadas. A transmissão é baseada nos seguintes princípios, e é frequentemente referida como H1:

- Cada segmento possui somente uma fonte de energia, a fonte de alimentação;
- Alimentação não é fornecida ao bus enquanto uma estação está enviando;
- Os dispositivos de campo consomem uma corrente básica constante quando em estado de repouso;
- Os dispositivos de campo agem como consumidores passivos de corrente (sink);
- Uma terminação passiva de linha é necessária, em ambos finais da linha principal do barramento;
- Topologia linear, árvore e estrela são permitidas;

No caso da modulação, supõe-se que uma corrente básica de pelo menos 10mA consumida por cada dispositivo no barramento. Através da energização do barramento, esta corrente alimenta os dispositivos de campo. Os sinais de comunicação são então gerados pelo dispositivo que os envia, por modulação de + /- 9mA, sobre a corrente básica.

A IEC 61158-2 também contempla um conjunto de instruções de instalações específicos utilizado em aplicações pelo protocolo Profibus.

Características:

- Transmissão de Dados: Digital, sincronizado a bit, código Manchester;
- Taxa de Transmissão: 31,25 kbits/s;
- Segurança de Dados: Preâmbulo, start e end limiter e FSC (frame check sequence);
- Cabos: Par trançado blindado;
- Alimentação: Via barramento ou externa (9-32 Vcc);
- Classe Proteção à Explosão: Segurança Intrínseca (Eex ia/ib) e invólucro (Eex d/m/p/q);
- Topologia: Linha ou árvore, ou combinadas;
- Número de Estações: Até 32 estações por segmento, máximo de 126;
- Distância Máxima sem repetidor: 1900 m (Cabo tipo A);
- Repetidores: Até 4 repetidores;

C. Métodos de detecção de erros

O objectivo da codificação de dados é a eficiência do transporte dos mesmos através de um meio específico. Cada tipo de meio de comunicação tem uma série de incapacidades que se traduzem em erros na transmissão de dados. Esses erros afetam a capacidade de transporte de dados e por isso devem ser detectados [8].

1) CRC: O CRC (*Cyclic Redundancy Check*) é uma técnica utilizada para detecção de erros na transmissão de dados digitais usada em redes LANs e WANs. É o método mais largamente empregado em sistemas de transmissão por dois motivos: 1) pela sua simplicidade de implementação, normalmente através de um hardware muito simples e 2) pela sua alta eficiência [9].

Nos códigos de blocos, uma mensagem é dividida em blocos, cada um deles com k bits, denominados palavras de dados (datawords). Adiciona-se r bits redundantes a cada bloco para obter o comprimento n = k + r. Os blocos resultantes de n bits são chamados palavras de código (codewords) [10].

É possível criar uma combinação de 2^k palavras de dados e criar uma combinação de 2^n palavras de código. Sendo $n > k$, o número de palavras de código possíveis é maior que o número de palavras de dados. No processo de codificação de blocos a mesma palavra de dados é sempre codificada como a mesma palavra de código. Isso significa que temos $2^n - 2^k$ palavras de código não usadas.

Na técnica CRC, é adicionando n − k bits 0s a palavra de dados e o resultado será utilizado para alimentar o gerador. O gerador usa um divisor de tamanho n − k + 1. A definição de quais serão os n − k + 1 bits utilizados como divisor será explicado posteriormente. O quociente da divisão binária é descartado; o resto ($r_2\ r_1\ r_0$) é anexado à palavra de dados para criar a palavra de código.

Utilizado um exemplo, conforme o da Tabela I para exemplificar o funcionamento explicado acima. Para o código CRC com n=7 e k=4, tem-se um divisor de tamanha três e a palavra de dados aumentada de tamanho sete.

Tabela I
CÓDIGO CRC DE C(7, 4)

Palavras de dados	Palavras de código	Palavras de dados	Palavras de código
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

Para esse exemplo, a Figura 5 mostra um projeto possível para o codificador e decodificador [10].

No decodificador uma cópia de todos os n bits é alimentada no verificador (réplica do gerador). O resto produzido pelo verificador é uma síndrome de n − k (3, nesse caso) bits que alimenta o analisador lógico de decisão. O analisador verifica se os bits de síndrome forem todos 0s, em caso afirmativo, são aceitos os 4 bits mais à esquerda da palavra de código como palavras de dados (interpretado como não sendo um erro); caso contrário, os 4 bits são descartados (erro).

- Codificador

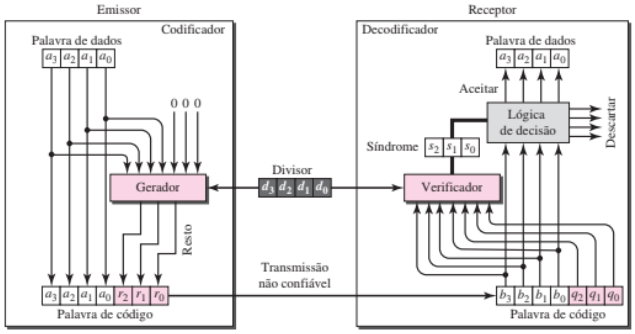

Figura 5. Codificador e decodificador CRC

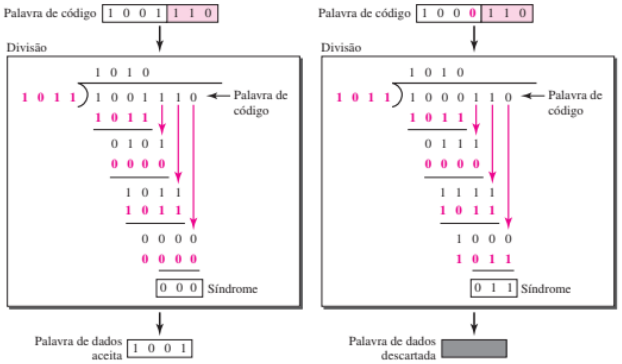

Figura 7. Divisões sucessivas no decodificador CRC para dois casos

No codificador é incrementado a palavra de dados com um número n − k de bits 0s, conforme exemplificado na Figura 5 e, divide a palavra de dados resultante pelo divisor.

A divisão realizada pelo codificador é binária de módulo 2. Esse tipo de divisão se distingue da binária simples pelo fato de a subtração ser de módulo 2. Dessa forma, a divisão longa em módulo 2, consiste em emparelhar o divisor ao dividendo, realizando as operações XOR e processando a "queda" dos bits de forma sequencial, até que divisor apresente grau maior que o dividendo, fator que implicará no termino da operação e na obtenção do resto da divisão, como pode ser observado na Figura 6.

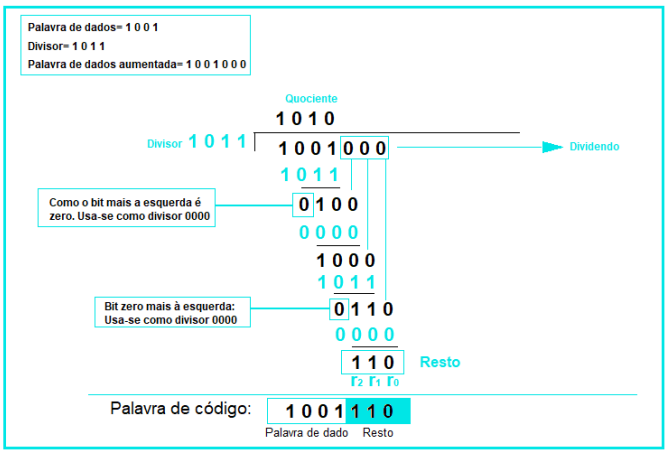

Figura 6. Divisão no codificador CRC

- Decodificador

A palavra de código pode ser corrompida durante sua transmissão. O decodificador realiza o mesmo processo de divisões sucessivas do codificador. O resto da divisão é a síndrome. Se a síndrome for formada completamente por 0s, não existem erros; a palavra de dados é separada da palavra de código recebida e aceita. Caso contrário, tudo é descartado. A Figura 7 expõe os dois casos, respectivamente.

- Divisor

No exemplo anterior, o divisor escolhido foi 1011. A escolha do divisor envolve uma certa dose de álgebra abstrata. Prestando atenção nos passos tomados no exemplo, percebe-se a aplicação repetidamente de operações XOR entre o divisor e parte do dividendo. Além disso, o divisor tem n – k + 1 bits que são predefinidos ou, então, é formado completamente por 0s caso o bit mais à esquerda dos bits de dados aumentados que faziam parte da operação XOR fosse zero. Por fim, uma analise criteriosa mostra que são necessários apenas n – k bits do divisor na operação XOR, sendo que, o bit mais à esquerda não é necessário, pois o resultado da operação é sempre 0.

- Resto

Para o exemplo utilizado, o resto tem um comprimento de 3 bits (n – k bits em termos genéricos). Nesse caso, devem ser usados três registradores (dispositivos de armazenamento de um único bit) para armazenar esses bits.

- Polinomial

Uma cadeia de bits pode ser melhor representada de forma polinomial. Dessa forma, as palavras de código (sequências de bits), também podem ser interpretadas e representadas por polinômios constituídos por potências de x e com coeficientes 0 e 1, de acordo com a posição dos bits da cadeia em questão [9].

Os códigos polinomiais se baseiam no tratamento de strings de bits como representações de polinômios com coeficientes 0 e 1. Um quadro de k bits é considerado a lista de coeficientes para um polinômio com k termos, variando desde x^{k-1} até a x^0. Dizemos que tal polinômio é de grau $k-1$. O bit de mais alta ordem (mais à esquerda) é o coeficiente x^{k-1}; o bit seguinte é o coeficiente de x^{k-2} e assim por diante.

Dessa forma, uma mensagem binária de n bits, pode ser convertida em um polinômio de ordem n – 1, como pode ser observado no quadro de exemplos da Tabela II:

Tabela II
CÓDIGO CRC DE C(7, 4)

$M(x) = 10101 = 1.\ x^4 + 0.\ x^3 + 1.\ x^2 + 0.\ x^1 + 1.\ x^0$
$M(x) = 1100101 = 1.\ x^6 + 1.\ x^5 + 0.\ x^4 + 0.\ x^3 + 1.\ x^2 + 0.\ x^1 + 1.\ x^0$
$M(x) = 1011100 = 1.\ x^6 + 0.\ x^5 + 1.\ x^4 + 1.\ x^3 + 1.\ x^2 + 0.\ x^1 + 0.\ x^0$

- CRC usando polinômios:

Com o uso da representação polinomial para bits, encontrar a palavra de código se torna uma tarefa mais simples. Basicamente, citando o exemplo usado anteriormente, a palavra de dados binária 1001 e torna o polinômio $x^3 + 1$ e o divisor 1011 torna-se o polinômio $x^3 + x + 1$. Para encontrar a palavra de dados aumentada na forma polinomial, primeiro observamos que na forma binário são acrescidos três zeros a palavra de dados, nesse caso tem-se, 1001000. Aplicando os conceitos visto anteriormente, a palavra de dados aumentada polinomial segue:

$$1 \cdot x^6 + 0 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 \quad (1)$$

$$x^6 + x^3 \quad (2)$$

Na divisão, inicialmente será dividido o primeiro termo do dividendo, x^6, pelo primeiro termo do divisor, x^3. Em seguida, multiplica-se x^3 pelo divisor e subtraí o resultado do dividendo. O resultado será x^3, com grau maior que o grau do divisor; continua a divisão até que o grau do resto seja menor que o grau do divisor. A Figura 8 ilustra a divisão explicada acima.

Em uma representação polinomial, normalmente o divisor é conhecido como polinômio gerador $G(x)$. A maior ou menor eficiência do CRC depende fortemente da escolha do polinômio gerador $G(x)$. Isso levou os organismos de padronização a definir G(x) para determinados sistemas de comunicação de dados.

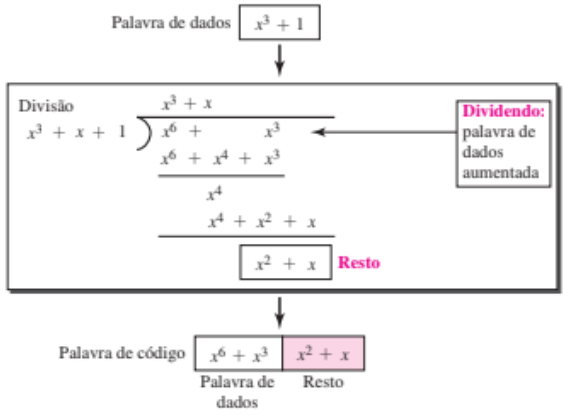

Figura 8. Divisão CRC usando polinômios

2) Checksum: O método de detecção de erros *checksum* é muito utilizado em diferentes protocolos na internet. O método baseia-se em uma operação aritmética entre os valores recebidos e os compara ao valor de *checksum* para verificação se foi recebido um pacote de dados coeso [10]. O *checksum* utiliza o complemento de um para manter o número de bits consistente em um sistema. Como exemplo pode-se ver na Figura 9 uma demonstração da troca de pacote entre um emissor e um receptor, no final da comunicação se o valor de *checksum* no receptor for igual a zero significa que não houve perda de dados.

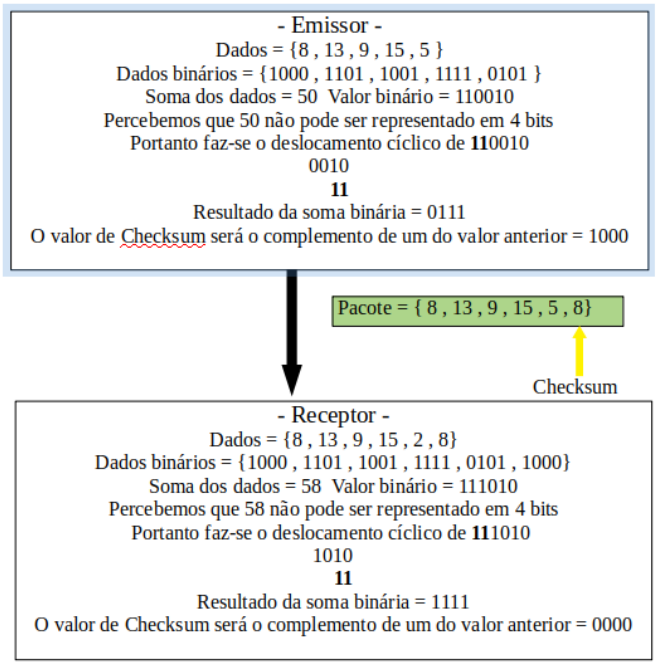

Figura 9. Diagrama do processo de detecção do erro com o método *Checksum*.

D. Sistema multimestre

O Profibus é um sistema multimestre que permite a operação conjunta de equipamentos, controladores terminais de engenharia ou visualização, com seus respectivos periféricos. Os Dispositivos Mestres determinam a comunicação de dados em um barramento enquanto os escravos somente respondem às solicitações dos mestres. Essa comunicação é realizada enquanto o dispositivo mestre possui o direito de acesso ao barramento (token). O token é um mecanismo de arbitragem que deve ser implementado para evitar possíveis colisões no barramento quando mais de uma estação deseja transmitir uma mensagem. Seu funcionamento está exemplificado na Figura 10 [10].

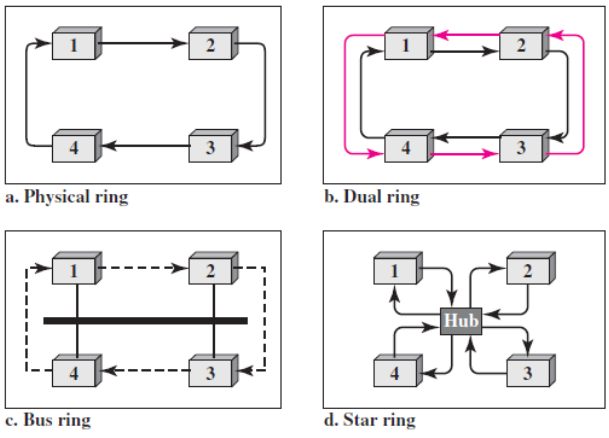

Figura 10. Colisão do primeiro bit no CSMA/CD.

Os mestres são chamados de estações ativas no barramento.

Já os Dispositivos Escravos são dispositivos de periferia como, válvulas, módulos de I/O, posicionadores, transmissores etc. Esses periféricos não possuem direito de acesso ao barramento, e somente enviam ou reconhecem alguma informação do mestre quando for solicitado.

1) Troca de mensagens no Profibus: A troca de mensagens no Profibus acontece em ciclos e cada pacote de dados é conhecido como mensagem ou frame. Cada frame de requisição de dados ou de envio de dados a uma estação mestre Profibus está associado a um frame de confirmação ou resposta de uma estação mestre ou escrava [11].

2) Sistema DP: Cada sistema DP pode conter três tipos diferentes de dispositivos:

- Mestre DP Classe 1:

Esse dispositivo é um controlador central que troca informações com as estações escravas dentro de um ciclo de mensagens especificado. O dispositivo mestre mais comum é o controlador lógico programável (CLP) [12].

- Mestre DP Classe 2:

Os mestres DP Classe 2, são os programadores, dispositivos de configuração ou sistemas de supervisão, como por exemplo, o Simatic PDM. São utilizados para a configuração da rede, ou para os propósitos de operação e monitoria [12].

- Escravo DP:

Um escravo DP é um dispositivo periférico (dispositivos de E/S, inversor de frequência, IHM, válvula) que coleta informação de entrada e/ou atua sobre o processo com informações derivadas da própria rede. A quantidade de informação de entrada e saída depende no tipo de dispositivo. O PROFIBUS permite até 246 bytes de entrada e 246 bytes de saída [12].

E. Comunicação no Profibus DP

A comunicação no Profibus DP funciona com trocas de caracteres. Os caracteres no Profibus DP são compostos por 11 bits (1 start bit + 8 bits de dados + 1 bit de paridade + 1 stop bit).

1) Estrutura de uma carácter: A transmissão binária ocorre em codificação de linha NRZ Unipolar (*Non Return to Zero Unipolar*). A sincronização e a detecção de erros ocorre pela adição dos bits de start, stop e paridade, com a adição do bit de paridade pode-se detectar até 50% dos erros da transmissão [13]. Na Figura 11 é representado a estrutura de um caracter no Profibus, note que o dado binário é invertido sendo transmitido seguindo a ordem do bit LSB ao MSB.

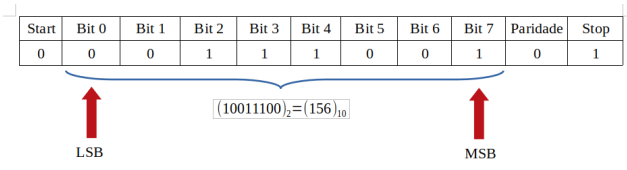

Figura 11. Estrutura de um carácter transmitido no Profibus.

2) Estrutura dos frames no Profibus: Na especificação do Profibus é apresentado 4 tipos de frames principais que possuem caracteres específicos para informar o início da comunicação:

- **SD1** = $(10)_{16}$ ou $(16)_{10}$
- **SD2** = $(68)_{16}$ ou $(104)_{10}$
- **SD3** = $(A2)_{16}$ ou $(162)_{10}$
- **SD4** = $(DC)_{16}$ ou $(220)_{10}$

Após o carácter de início temos outros caracteres que podem compor um frame:

- **DA**: Byte de endereço de destino.
- **SA**: Byte de endereço de fonte.
- **FC**: Byte de controle, identifica a função do frame. Conforme a configuração dos bits é possível identificar o tipo de estação que o enviou. Contém também a natureza das informações que o frame carrega, i.e., se é resposta de resposta, confirmação ou pedido.
- **FCS**: Byte de checagem, neste byte é disponibilizado para a detecção de erros por *checksum* [13]. É feita a soma de todos os bytes contidos no pacote ignorando o *overflow*.
- **LE**: Byte de comprimento, assume valores entre 4 e 249.
- **LEr**: Byte de comprimento repetido.
- **DU**: Campo para os dados, este pode variar entre 1 a 244 bytes.
- **ED**: Byte finalizador, tem seu valor fixo à $(16)_{16}$ ou $(22)_{10}$
- **SC**: Frame de resposta curta, tem seu valor fixo em $(5)_{16}$ ou $(5)_{10}$

Esses diferentes caracteres são agrupados de acordo com o carácter inicial, desta forma é formado um frame completo como pode ser visualizado na Figura 12-a). Na Figura 12-b) pode-se ver como cada byte é transferido de acordo com o *Clock* e bits de início e fim.

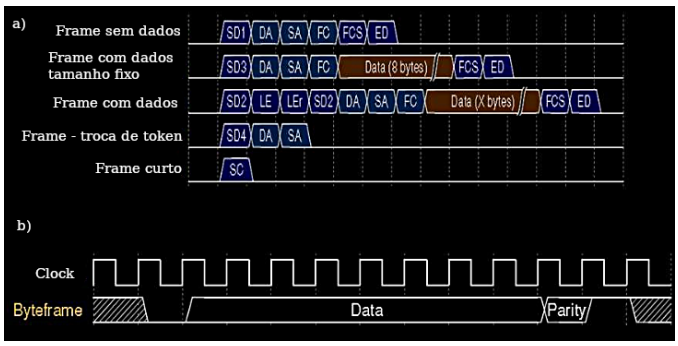

Figura 12. Estrutura de um carácter transmitido no Profibus (adaptado de [11].

III. Montagem do Experimento

Para a parte física da comunicação mestre-escravo conforme os padrões especificados pelo protocolo Profibus DP, foi utilizado dois microcontroladores Arduino Uno [1], dois módulos MAX-RS485 [2], fios jumper trançados (traçados

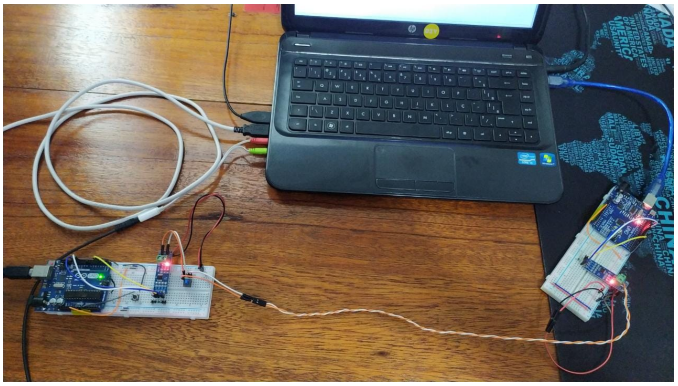

Figura 13. Imagem do experimento construído.

manualmente) para fazer o ligamento entre os módulos. A Figura 13 mostra o circuito montado para o projeto.

A conexão entre os módulos está exemplificado na Figura 14 de forma mais didática.

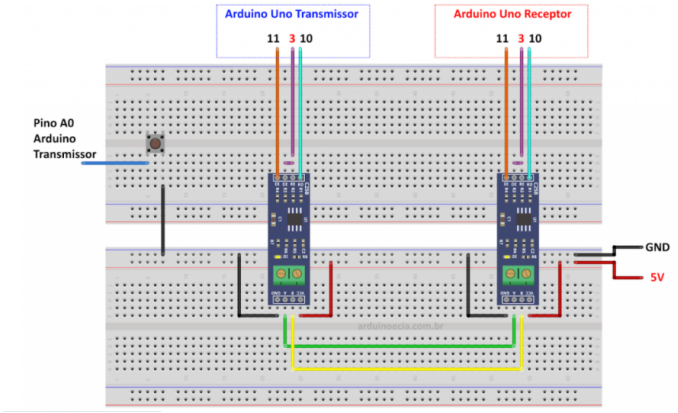

Figura 14. Circuito comunicação dos módulos.

A parte de programação dos microcontroladores foi implementada através da IDE do Arduino. Existem diversas estruturas de frame possíveis na rede Profibus, para o trabalho foi escolhida a estrutura de frame com dados de tamanho fixo. A Figura 15 abaixo ilustra a estrutura desse tipo de frame:

SD3	DA	SA	FC	Data(8 bytes)	FCS	CRC	ED

Figura 15. Estrutura frame com dados de tamanho fixo.

O que significa cada campo do frame da Figura 15 está explicado na seção de fundamentação teórica. Os valores utilizados no programa para inicializar cada campo são: SD3 = B10100010; DA = B00000010; SA = B00000001; FC = B00001010; Dado = B00000001 (valor será recebido da interface gráfica posteriormente); FCS = B00000000 (inicializado como zero); CRC = B00000000 (inicializado como zero); ED = B00010110;

Foi desenvolvida uma interface gráfica utilizando a IDE Processing [14]. A interface permite a inserção do campo de dado do frame, essa entrada é enviada para o código do transmissor na IDE do Arduino, que organiza o dado recebido na organização do frame e trata o frame utilizando detectores de erro CRC e Checksum. O transmissor envia o frame para o receptor que novamente trata os dados do frame utilizando detectores de erro CRC e Checksum e, então, envia os dados para a código da interface que mostra-os na tela.

Por ultimo, um botão foi adicionado ao circuito para simular um ruído. Basicamente, quando o ruido é acionado as detecções de erro CRC e Checksum no receptor retornam que há um erro no dado recebido do transmissor.

IV. Resultados

Foi realizada a comunicação entre os dois microcontroladores, conforme descrita no tópico montagem do experimento. Os resultados obtidos foram conforme esperado pois, com o fio de tamanho 50 cm não houve grades interferências.

Para o codificador (transmissor), a forma de onda gerada para o primeiro byte enviado está ilustrada na Figura 16.

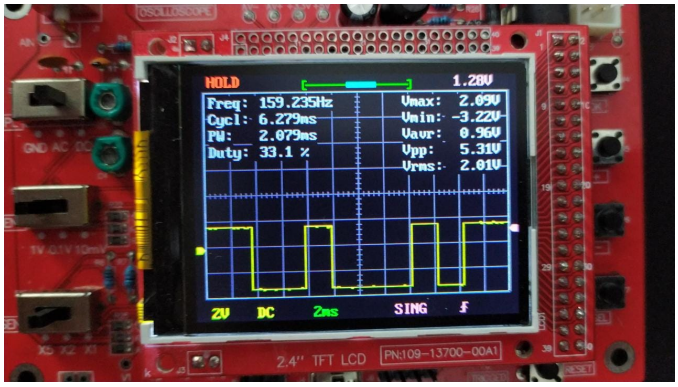

Figura 16. Forma de onda codificador origem sem ruído.

É possível visualizar que a codificação de linha utilizada nessas transmissões é a NRZ Unipolar, pois o valor sendo transmitido na Figura 17 é 162, i.e. o valor para o início do frame SD3 no Profibus. A representação na forma binária é de 10100010, nessa transmissão é enviado do LSB (bit menos significativo) ao MSB (bit menos significativo), portanto a sequência binária representada na Figura 17 é 01000101.

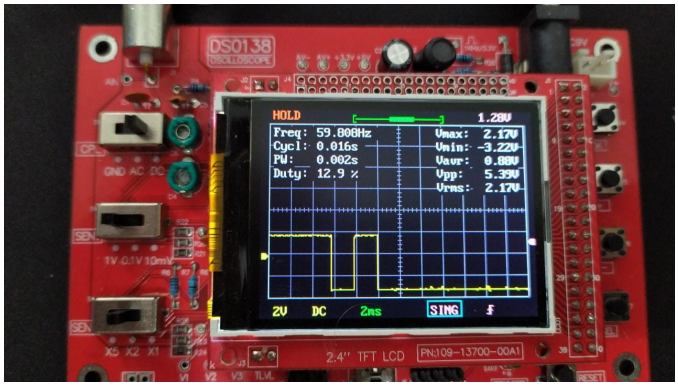

Figura 17. Forma de onda decodificador destino com ruído adicionado.

As Figura 18 e 17 ilustram a forma de onda obtida quando é adicionado um ruído na transmissão do frame para o decodificador (receptor).

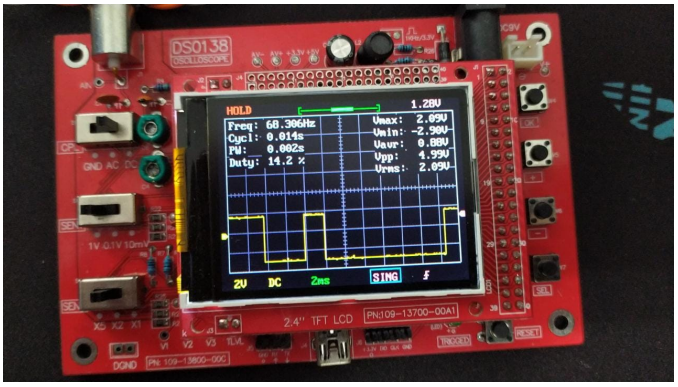

Figura 18. Forma de onda decodificador destino com ruído adicionado.

Quando o ruído é adicionado, as funções de detecção de erro CRC e Checksum retornam erro no dado recebido, conforme esperado.

Figura 19. Print de tela mostrando a interface gráfica executando.

A interface gráfica como mostra a Figura 19 se mostrou uma ferramenta útil para interagir com o sistema, nela é possível analisar de maneira rápida se os bytes em um frame do Profibus estão de acordo com o esperado.

V. Conclusão

Apesar da quantidade de exigências que devem ser obedecidas na implementação de um protocolo industrial, o trabalho validou a possibilidade de se replicar a comunicação mestre-escravo do protocolo Profibus DP, por meio de dois microcontroladores, dois módulos MAX-RS485 e fios trançados. Com os resultados, é possível estender a comunicação para quantos mestres e escravos foram desejados e aplicar a comunicação, conforme foi desenvolvida no trabalho, em projetos de implementação que envolvam o protocolo Profibus DP.

Apêndice A
Códigos Utilizados

Nesta seção é mostrado os códigos utilizados nos dois Arduinos UNO e o código para realizar a interface gráfica no software *Processing*.

- Receptor.ino:

```
//-------Bibliotecas_utilizadas------------//

#include <SoftwareSerial.h>
#include <util/crc16.h>

//
    ------------------------------------------

//------------Mapeamento_de_hardware
    -----------

#define Pino_RS485_RX    10 //Pinos de
    comunicacao serial do modulo RS485
#define Pino_RS485_TX    11

#define SSerialTxControl 3 //Pino de controle
    transmissao/recepcao

#define Pin13LED         13 //Define led 13
    para mostrar atividade na comunicacao

//----------------------------------------
//---------------Configuracao------------//

#define RS485Transmit    HIGH
#define RS485Receive     LOW

SoftwareSerial RS485Serial(Pino_RS485_RX,
    Pino_RS485_TX); //Cria a serial por
    sofware para conexao com modulo RS485

int incomingByte = 0;      // vari vel para o
    dado recebido.
byte ED =   B00010110;     //ED para finalizar
    o Pacote.
byte data[8];              //Vetor para
    armazenar dados recebidos.
int i = 0;
char prefixo[8][5] = {"SD3:" , "DA: " , "SA: "
     , "FC: " , "DU: " , "FCS:" , "CRC:" , "ED
    : "};

//----------------------------------------
//----------------------------------------
//--------Prototipo de Funcoes---------//

bool Func_Checksum() //Retorna 1 caso erro
{                    //Retorna 0 caso checksum
     correto
  byte soma = 0;
  for (int i = 0; i < sizeof data / sizeof
      data[0]; i++) {
    if (i != 6) { //se estiver na posi  o
       diferente do byte de CRC.
      soma += data[i];   //soma os valores.
    }
    //Serial.println(soma);
  }
  soma = ~soma; // Complemento de 1 do
     CHECKSUM realizado
  if(soma != 0){
    Serial.println("ERRO NO CHECKSUM");
    return true;
  }
  Serial.println("CHECKSUM CORRETO");
  return false;
}

bool Func_CRC() // Retorna 0 caso CRC correto
```

```
{               // Retorna 1 caso CRC
    incorreto
  uint8_t crc = 0, i;
  for (i = 0; i < sizeof data / sizeof data
      [0]; i++){
    if(i != 6){ // Passa pela posicao 6 que
        a do CRC.
      crc = _crc_ibutton_update(crc, data[i]);
          // a funcao _crc_ibutton_update usa
          o polinomio x^8 + x^5 + x^4 + 1
    }
  }
  crc = _crc_ibutton_update(crc, data[6]); //
     Checa por ultimo com o resto que esta na
      posicao 6.
  if(crc == 0){
    Serial.println("CRC CORRETO");
    return false;
  }
  Serial.println("ERRO NO CRC");
  return true;
}

//----------------------------------------

void setup()
{
  //Inicializa a serial do Arduino
  Serial.begin(9600);
  Serial.println("Modulo Receptor");
  Serial.println("Aguardando dados...");

  pinMode(Pin13LED, OUTPUT);
  pinMode(SSerialTxControl, OUTPUT);

  digitalWrite(SSerialTxControl, RS485Receive)
      ; //Coloca o modulo RS485 em modo de
      recepcao

  RS485Serial.begin(480); //Inicializa a
      serial do modulo RS485
}

void loop()
{
  if (RS485Serial.available() > 0)
  {
      incomingByte = RS485Serial.read(); //
          l   do buffer o dado recebido:

      Serial.print(prefixo[i]); // responde
          com o dado recebido:
      Serial.println(incomingByte);
      data[i] = incomingByte;
      i++;

      if(incomingByte == ED){
        Serial.println("Pacote Recebido");
        Serial.println("Teste para deteccao de
            erros");
        Func_Checksum();
        Func_CRC();
        i = 0;
      }
  }

}
```

- Transmissor.ino:

```
//-------Bibliotecas_utilizadas-----------//

#include <SoftwareSerial.h>
#include <util/crc16.h>

//----------------------------------------
//-------Mapeamento_de_Hardware-----------//

//Pinos de comunicacao serial do modulo RS485
#define Pino_RS485_RX    10
#define Pino_RS485_TX    11

//Pino de controle transmissao/recepcao
#define SSerialTxControl 3

//Define led 13 para mostrar atividade na
    comunicacao
#define Pin13LED         13

//----------------------------------------
//----------------Configuracoes----------//

SoftwareSerial RS485Serial(Pino_RS485_RX,
    Pino_RS485_TX); //Cria a serial por
    sofware para conexao com modulo RS485

#define RS485Transmit    HIGH
#define RS485Receive     LOW

byte SD3 =  B10100010;    //Inicio de Frame =
    162 (decimal).
byte DA =   B00000010;    //Endere o do
    Receptor = 2 (decimal).
byte SA =   B00000001;    //Endere o do
    Transmissor = 1 (decimal).
byte FC =   B00001010;    //Valor de FC (Frame
     check) para Diagnose dos Sinais de dados.
byte Dado = B00000001;    //Dado = 1 (decimal)
    .
byte FCS =  B00000000;    //inicializado em 0,
     antes da logica do CHECKSUM.
byte CRC =  B00000000;    //inicializado em 0,
     antes da logica do CRC.
byte ED =   B00010110;    //ED = 22 (decimal)
    para finalizar o Pacote.

byte data[] = { SD3, DA, SA, FC, Dado, FCS,
    CRC, ED };

//----------------------------------------
//--------Prototipo de Funcoes---------//

void Func_Checksum()
{
  data[5] = 0;
  for (int i = 0; i < sizeof data / sizeof
      data[0]; i++) {
    if (i != 5 && i != 6) { //se estiver na
        p o s i   o diferente de FCS (Checksum)
        e CRC.
      data[5] += data[i];   //soma os valores.
    }
    //Serial.println(data[5]);
  }
  data[5] = ~data[5]; // Complemento de 1 do
      CHECKSUM realizado
}

void Func_CRC()
```

```
{
  uint8_t crc = 0, i;
  for (i = 0; i < sizeof data / sizeof data
      [0]; i++)
    if (i != 6) {
      crc = _crc_ibutton_update(crc, data[i]);
           // a funcao _crc_ibutton_update usa
           o polinomio x^8 + x^5 + x^4 + 1
    }
  data[6] = crc;  // Precisa ser 0 para que o
      dado esteja coerente
                  // ou o valor do resto para
                      enviar em conjunto com a
                       Palavra de dados
}

//----------------------------------------
//

void setup()
{
  Serial.begin(9600); //Inicializa a serial
      padr o do Arduino
  Serial.println("Modulo Transmissor");
  Serial.println("Pressione o botao para
      enviar os dados...");

  pinMode(Pin13LED, OUTPUT);
  pinMode(SSerialTxControl, OUTPUT);

  RS485Serial.begin(480); //Inicializa a
      serial do modulo RS485

  pinMode(A0, INPUT_PULLUP); //Seta o pino A0
      como entrada e habilita o pull up

  //  Serial.println(data[6]);
  //  Serial.println(_crc_ibutton_update(0,
      SD3));
  //  Serial.println(data[5]);
  //  delay(1000);
}

void loop()
{
    if (Serial.available() > 0) //Se algo for
        recebido pela serial
    {
      Dado = Serial.read();
      data[4] = Dado;
      Func_Checksum();
      Func_CRC();
      //Serial.println("aqui");

    }
//      delay(5000);
//      Serial.write(data[4]);

    int  valor = digitalRead(A0); //Verifica
        se o botao foi pressionado
    if (valor == 0)
    {
      Serial.println("Botao pressionado.
          Enviando dados!");

      //Habilita o modulo para transmissao
      digitalWrite(SSerialTxControl,
          RS485Transmit);
      digitalWrite(Pin13LED, HIGH); //Liga o
          led 13 para mostrar que vai
          transmitir
      delay(10);
      //Envia a string
      for(int i = 0; i < 8; i++){
        RS485Serial.write(data[i]);
        delay(1000);
      }

      digitalWrite(Pin13LED, LOW);

      digitalWrite(SSerialTxControl,
          RS485Receive); //Desabilita o modulo
           para transmissao
      while (digitalRead(A0) == 0)
      {
        delay(50);
      }
    }
}
```

- Interface Grafica.pde:

```
import processing.serial.*; //Importa a
    biblioteca para abrir uma comunica  o
    Serial
Serial myPort, myPort2; //Inst ncia a
    biblioteca para a comunia  o Serial
import controlP5.*;

PImage fundo;     //objeto de imagem para
    Background.

// Para textbox-----//
String text, text2;
ControlP5 cp5;
Textarea myTextarea;

//=======================

int valor_recebido; //Cria uma vari vel para
    armazenas o valor recebido pela serial.
int valor_recebido2;
void setup()
{
  PFont font = createFont("GFSArtemisia-Bold"
      ,20);

  String portName = Serial.list()[0];
  String portName2 = Serial.list()[33];
  myPort2 = new Serial(this, portName2, 9600);
  myPort = new Serial(this, portName, 9600);

  size(1000, 600); //Define o tamanho da tela

  //Carrega imagem de fundo.
  fundo = loadImage("imagens/background.png");
  //
      ================================================


  cp5 = new ControlP5(this);
  cp5.addTextfield("Dado").setPosition
      (100,230).setSize(200,40).setFont(font);
  cp5.addButton("Enviar").setPosition(100,
      300).setSize(200, 40).setFont(font);
  myTextarea = cp5.addTextarea("txt")
              .setPosition(550,150)
              .setSize(300,400)
```

```
                .setFont(font)
                .setLineHeight(14)
                .setColor(color(128))
                .setColorBackground(color
                    (255,255,255))
                .setColorForeground(color
                    (25,25,112));
                ;
  myTextarea.setLineHeight(25); // configura o
       espa o entre linhas

}

void draw()
{
  background(fundo); //Atualiza a imagem de
      fundo (backgroud) da interface

  if (myPort2.available() > 0) //Se algo for
      recebido pela serial
  {
    text2 = myPort2.readStringUntil('\n');
    myTextarea.append(text2);
    println(text2);
  }

}

void Enviar() { //  Funcao chamada quando o
    botao Enviar    pressiondado
  println();    //  deve ter o mesmo nome do
      bot o.
  print("this is the text you typed :");
  text = cp5.get(Textfield.class, "Dado").
      getText();
  println(Integer.parseInt(text));
  myPort.write((byte)Integer.parseInt(text));
  text += '\n';
  myTextarea.append("Dado enviado: ");
  myTextarea.append(text);
  print(text);
}
```

REFERÊNCIAS

[1] Farnell. Arduino uno r3. [Online]. Available: https://partineh.com/wp-content/uploads/2018/07/1522237550_arduino-uno-r3.pdf

[2] maximintegrated. Max485. [Online]. Available: https://www.maximintegrated.com/en/products/interface/transceivers/MAX485.html

[3] L. F. Medina. Automação e supervisão de uma mini-caldeira elétrica utilizando protocolo profibus. [Online]. Available: http://www.tcc.sc.usp.br/tce/disponiveis/18/180500/tce-18112011-104624/?&lang=br

[4] C. M. Freitas. Fundamentos e aplicações do protocolo profibus. [Online]. Available: https://www.embarcados.com.br/fundamentos-protocolo-profibus/

[5] C. Cassiolato. Uma visão de profibus, desde a instalação até a configuração básica. [Online]. Available: https://www.profibus.org.br/artigo-tecnico-detalhe.php?id=uma-visao-de-profibus-desde-a-instalacao-ate-a-\configuracao-basica-parte-5

[6] ——. Profibus: O meio físico. [Online]. Available: https://www.vivaceinstruments.com.br/en/article/profibus-o-meio-fisico

[7] C. M. Freitas. Redes de comunicação em rs-485. [Online]. Available: https://www.embarcados.com.br/redes-de-comunicacao-em-rs-485/

[8] R. Barbosa. Detecção de erros de comunicação de dados crc. [Online]. Available: https://paginas.fe.up.pt/~ee06166/documentos/Deteccao_de_erros.pdf

[9] J. Rochol, ***Livro**: Comunicação de Dados*, 1st ed. Porto Alegre: Bookman, novembro 2011, vol. 22.

[10] B. A. Fourozan, ***Livro**: COMUNICAÇÃO DE DADOS E REDES DE COMPUTADORES*, 4th ed. Porto Alegre: AMGH, outubro 2010.

[11] C. Cassiolato. Profibus: Serviços de transmissão e frames. [Online]. Available: https://www.vivaceinstruments.com.br/pt/artigo-detalhe?id=profibus-servicos-de-transmissao-e-frames

[12] ——. O que é profibus? [Online]. Available: https://www.smar.com/brasil/profibushttps://www.smar.com/brasil/profibus

[13] M. Felser, "Quality of profibus installations," *ResearchGate*, 01 2006.

[14] Processing.org. Processing. [Online]. Available: https://processing.org/