**SISTEMA DE AQUISIÇÃO DE DADOS COM COMUNICAÇÃO MODBUS**

**MODBUS COMMUNICATION DATA ACQUISITION SYSTEM**

Wericks da Silva Costa\*

Rúben Christian Barbosa\*\*

**RESUMO**

A agroindústria converte produtos primários em subprodutos para abastecer a população. Voltados principalmente para a produção de alimentos para o consumidor final, esses subprodutos são produzidos gradativamente andando lado a lado com a tecnologia, visando a melhoria no processo de produção. Ao contrário do setor industrial, o agronegócio lida com matérias-primas com base na perecibilidade, heterogeneidade e sazonalidade. O propósito do trabalho foi desenvolver um dispositivo de baixo custo capaz de atender as necessidades da agroindústria e utilizar do protocolo Modbus industrial através de um microcontrolador Arduino. Um dispositivo implantado na área de embalagem de banana, realizara a leitura do tipo de fruta a ser embalada e o operador que esta embalando. Essa aplicação envia dados para o mestre da rede utilizando o protocolo Modbus TCP/IP onde serão tratados as informações coletadas através do leitor RFID, onde o mestre da rede fará o controle de produção afim de fornecer informações essenciais como peso, tipo de fruta, data, hora e operador para o datalloger. A uma interface de hardware será através de um microcontrolador Arduino, onde realiza a leitura do cartão RFID. O funcionamento foi comprovado com testes, e os resultados foram satisfatórios, a comunicação com a rede Modbus, a comunicação entre os *shields* e a leitura do RFID, sendo então transcritos no presente trabalho. Com o sistema em pleno funcionamento, hoje se encontra em estado de melhorias para redução dos *shields*.

**Palavras-chave:** Modbus TCP/IP. Microcontrolador. RFID.

**ABSTRACT**

Agroindustry converts primary products into by-products to supply the population. Aimed primarily at the production of food for the final consumer, these by-products are produced gradually walking side by side with technology, aiming at improving the production process. Unlike the industrial sector, agribusiness deals with raw materials based on perishability, heterogeneity and seasonality. The purpose of the work was to develop a low-cost device capable of meeting the needs of the agroindustry and using the industrial Modbus protocol through an Arduino microcontroller. A device implanted in the banana packaging area had read the type of fruit to be packed and the operator packing. This application sends data to the network master using the Modbus TCP/IP protocol where the information collected through the RFID reader will be treated, where the network master will control the production in order to provide essential information such as weight, fruit type, date, time and operator for the datalloger. The hardware interface will be through an Arduino microcontroller, where it performs the reading of the RFID card. The functioning was proven with tests, and the results were satisfactory, the communication with the Modbus network, the communication between the shields and the rfid reading, being then transcribed in the present work. With the system in full operation, today it is in a state of improvement to reduce shields.

**Keywords:** Modbus TCP/IP. Microcontroller. RFID. Agroindustry.

1. **Introdução**

A agroindústria vem tomando seu lugar no mundo inteiro, com sistemas tecnológicos para controle de produção, e cultivo dos produtos para oferecer o melhor da fruta (ASSESSORIA DE COMUNICAÇÃO, 2019). Em Linhares, no Norte do Espírito Santo, é uma das cidades que mais se destaca pela quantidade de produtos, seja eles de banana, coco, mamão, café e entre outros, com o crescimento dessas produtoras, identificamos um problema com relação a separar os tipos de produtos embalados, de forma a auxiliar o sistema de controle de estoque.

Com a necessidade de controle do produto embalado e o operador que está trabalhando na linha de produção, para alimentar o sistema de estoque, notou-se que a melhor solução é utilizar um sistema de rádio frequência (*Radio-Frequency Identification* - RFID).

Altamente usado na aviação, rádio frequência ou identificação RFID fornece informações sobre posições, altura, velocidade e identificação de aeronaves. A origem da tecnologia se da na Segunda Guerra Mundial. Embora na década de 1930 os radares fossem capazes de determinar a presença de aeronaves de longe, não se sabe se esses sinais eram aeronaves aliadas ou inimigas. O que os alemães descobriram é que quando eles viram a aeronave, retornando a base, o sinal que refletia e mudava no radar. Este método foi o embrião do que mais tarde seria convertido em RFID. Após a guerra, nos anos 1960 e 1970, com a progressão das tecnologias de radiofrequência, os cientistas dos Estados Unidos, Japão e da Europa começaram a primeira busca pelo uso da tecnologia para objetos diferentes de identificação remota (MARINA MARTHA, 2014).

O sistema de rádio frequência (Radio-Frequency Identification - RFID), utiliza de sinais de rádio que é lido por um leitor RFID que recebe as informações que vem de um micro chip, uma sequência numérica gravada nele. A onda sonora refletida é convertida em informações digitais e interpretada pelo sistema.

O trabalho objetivou desenvolver um sistema de aquisição de dados e *datalloger* para o processo de embalagem de bananas. Para desenvolver o sistema, uma plataforma microcontrolada com uso de um meio de comunicação do sistema, tendo em vista a segurança dessa comunicação e que é uma linguagem Modbus padronizada para equipamentos de automação industrial. Criando assim um dispositivo de baixo custo, que fará a leitura da fruta embalada na linha de produção e o operador que está embalando a fruta, onde todas as informações serão levadas como por exemplo, para uma interface homem máquina, onde pode ser configurado o *datalloger*.

O Modbus TCP/IP é utilizada para comunicação utilizando redes, conectando pela porta 502. O Modbus TCP não requer um cálculo de *checksum* (usado para verificar a integridade dos dados transmitidos através de um canal de counicação Modbus), isso porque o TCP/IP já garante a integridade dos dados. Em 1999, o Modbus TCP/IP surgiu para lidar com a nova realidade dos protocolos Ethernet em aplicações industriais. O TCP/IP usa a internet para transportar os dados dos dispositivos, e as instruções do protocolo Modbus vêm através do TCP/IP (ALTUS, 2021).

Para validar o projeto, foram realizadas pesquisas dos componentes que irão se comunicar, pesquisar a fundo o funcionamento do sistema de rádio frequência (*Radio-Frequency Identification* - RFID), um protocolo Modbus utilizado e que seja seguro para transmitir todas as informações coletadas e os *shields* (componentes que irão complementar o arduino, comunicação Ethernet e leitor de USB).

1. **Referencial terórico**

**2.1- Agroindústria**

A agroindústria e a instalação da indústria no campo, para a transformação de matérias primas em subprodutos. A agropecuária proveniente da silvicultura e da agricultura tem como objetivo transformar as matérias primas, tornando mais longo sua disponibilidade, trazendo maior prazo de consumo, adicionando valor ao alimento, mantendo a naturalidade do alimento. Permitindo aos produtos rurais a começarem a ser beneficiados ainda nas fazendas e aos agricultores e pecuaristas deixarem de ser apenas fornecedores de matérias-primas para ganhar outros espaços no comércio nacional e internacional (ANDREA OLIVEIRA, 2021).

**2.2- Rastreamento de informações de produção**

Um dos processos básicos de rastreabilidade é a nomeação do tipo de produto. Este registro identifica a data, hora, produto e operador. Se ocorrer uma falha, essas informações são importantes para a conferência como por exemplo o peso total embalado para um determinado cliente.

Além de ser utilizado para abastecer o sistema de controle de estoque/produto, e da produtividade dos operadores, podendo usar as informações para melhorias e gratificações, além de servir como documento para auditoria.

## 2.3- Plataforma microcontrolada para prototipia Arduino

O Arduino (Figura 01-A) é um microcontrolador que contém suporte para entrada/saída dados já embutido, com linguagem de programação padrão baseado em C/C++ e programado pela IDE (*Integrated development environment*) ambiente integrado de desenvolvimento que é o *software* utilizado para escrever os programas para as placas de Arduino.

Utilizaram (MARINA MARTHA, 2014) o Arduino pela velocidade que se consegue desenvolver algo e pela simplicidade, isso é possível devido a modularidade do Arduino, da farta documentação, exemplos e da grande quantidade de módulos disponíveis para se conectar ao Arduino. É usado por ser um dispositivo de baixo custo e fácil integração em sistemas industriais através da internet das coisas.

Utilizando a comunicação Ethernet através do Shield W 5100 (Figura 01-B), realizou-se a comunicação Modbus TCP/IP, uma das comunicação mais utilizadas na automação industrial, e de fácil integração além de transmitir as informações a longa distância, podendo utilizar estrutura Ethernet existente, mais rápido e fácil de fazer diagnósticos com a redução de custo com estrutura.

|  |
| --- |
|  |
| (A) |
|  |
| (B) |

**Figura 1:** (A) Arduino Uno; (B) Shield Ethernet W5100

**Fonte:** FlipFlop (2014)

## 2.4- Protocolos de comunicação industriais

Modbus RS485: É um dos mais antigos protocolos de rede industrial (fundado em 1979). No entanto, geralmente é fácil se adaptar às mais diversas aplicações, sua velocidade de transmissão pode variar de 100Kbs a 35Mbs dependendo da distância. Muito econômico, principalmente porque não há necessidade de comprar software ou chips. Tem somente um mestre na rede, porém possibilita ter mais de um escravo (AUTOMACAO E CARTOONS, 2018).

Modbus RS232: No RS232 disponibiliza apenas a comunicação ponto a ponto, ou seja, um mestre e um escravo na rede. Envia dados na forma de uma série temporal de bits. É um padrão para comunicação entre o terminal de dados e o equipamento de terminação do circuito de dados. Sua velocidade de transmissão pode variar de 100Kbs a 35Mbs dependendo da distância (AUTOMACAO E CARTOONS, 2018).

CANopen: Pode operar sem host e é considerada uma rede multi-mestre. É adequado para projetos que requerem comunicações mais complexas e são econômicos (SPECS, 2015). CANopen foi desenvolvido como uma rede embarcada padronizada com recursos de configuração altamente flexíveis. Ele foi projetado originalmente para sistemas de controle de máquina orientados por movimento, como sistemas de manuseio. Sua velocidade de transmissão pode ser ajustada em níveis especificados na área de 10 kbit/s até 1 Mbit/s. Utiliza de uma saída DB9 para comunicar.

## 2.5- Protocolos de comunicação baseados em Ethernet

EtherCAT: Um dos maiores benefícios é que ele tem excelente velocidade de comunicação e recursos de empacotamento de dados. Portanto, apresenta uma série de vantagens na automação industrial (SPECS, 2015). Tem como base a Ethernet, oferecendo uma gama de vantagens importantes para a automação industrial, entre elas, a velocidade de comunicação e fácil integração a rede corporativa da indústria. Para comunicação, utiliza-se o Ethernet e a velocidade de sua transmissão chega a 2x100Mbit/s (Fast Ethernet, Full-Duplex).

Modbus TCP/IP: É um dos mais recentes protocolos de rede industrial do mercado. Ele fornece uma variedade de soluções multifuncionais incríveis para sua fábrica. Além disso, é muito rápido na comunicação e transferência de informações. Sem falar que ele pode ser facilmente conectado ao dispositivo. Para aplicações em uma arquitetura com meio Ethernet físico, o Modbus possui o protocolo de comunicação TCP que pode atingir velocidades de 100Mbps a 10Gbps. Este modelo é utilizado há muitos anos em equipamentos automatizados e funciona através de uma estrutura de aplicação que distribui tarefas entre prestadores de serviços "servidores" e "clientes" que necessitam desta informação (ALTUS, 2021).

ControlNet: É um protocolo de rede industrial aberto e é gerenciado pela anteriormente O*pen* ODVA(*DeviceNet Vendors Association*). O ControlNet tem uma taxa de transferência de dados de 5 Mbps e está na faixa média das três redes populares com EtherNet/IP variando de 10 Mbps a 1 Gbps e DeviceNet variando de 125 a 500 Kbps. Foi projetado para fornecer controle confiável de alta velocidade e transferência de dados de Entrada/Saída usando a programação que define a lógica para temporização específica na rede (CÉSAR CASSIOLATO, 2010).

# Metodologia

## 3.1- Sistema

Para o dispositivo, será necessário a comunicação Ethernet, para utilizar o protocolo Modbus TCP/IP, entrada USB para o leitor de de rádio frequência (*Radio-Frequency Identification* - RFID) e alimentação do 5 VDC.

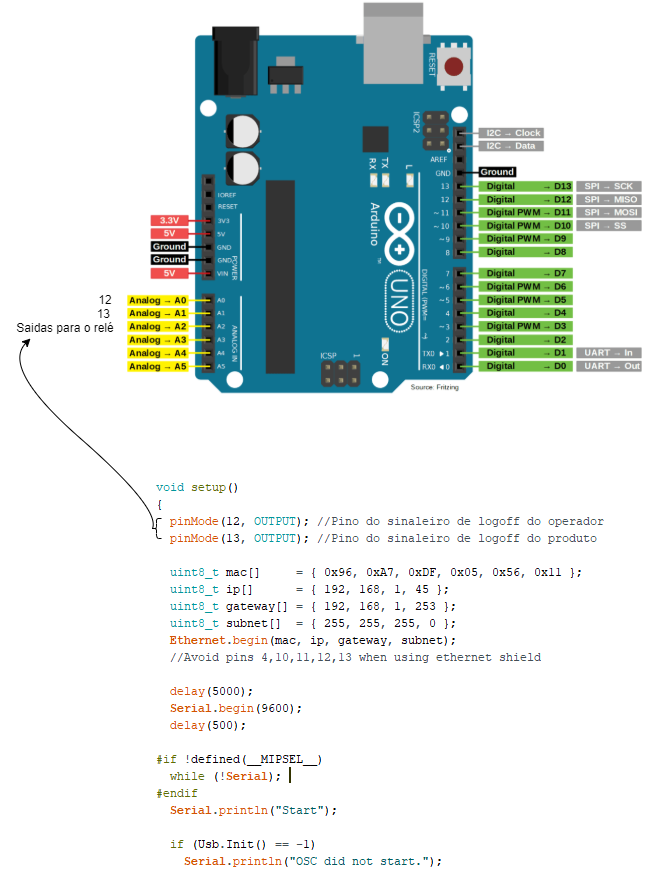
Definiu-se o Arduino UNO para controlar o sistema de identificação, para alimentar todo o sistema, utilizou-se de um regulador de tensão *step-down* (placa responsável pela redução da tensão de entrada) para entrada 24 VDC com saída 5 VDC.

Foi elaborado uma prova de conceito para testes preliminares em campo, para ser analisado os possíveis erros que não são detectados em bancada.

Para indicação de produto ou operador logado, fizemos a conversão das entradas analógicas para saídas digitais (Figura 02), utilizando dois relés para acionar dois sinaleiros 24 VDC.

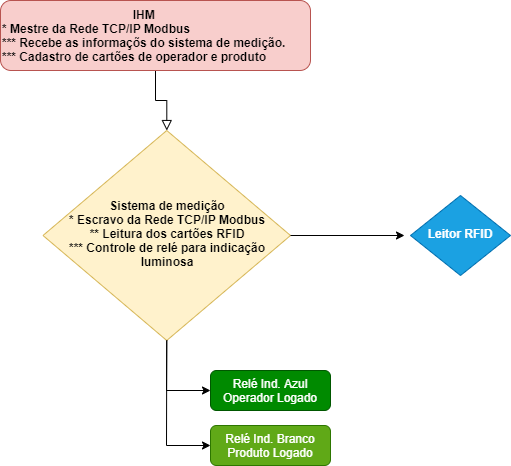
O sistema coleta as informações do leitor RFID, dividindo a leitura em três *strings* (sequências de caracteres) e envia para o mestre da rede, assim que recebido a informação, o mestre da rede junta as três sequências de caracteres em apenas uma, essa foi a forma de trocar as informações entre os disposistivos. A IHM irá verificar se os caracteres lidos se encontram em sua base de dados, caso seja encontrado, envia um *bit* (sinal de comunicação) para o dispositivo de leitura RFID acionando assim o relé de operador ou produto logado.

A estrutura da rede (Figura 03) foi separada por dois tipos de comunicações durante a implementação, tirando todo risco de erro na rede, o sistema de leitura RFID foi utilizado a comunicação Modbus TCP/IP enquanto a comunicação do CLP para controle de peso das balanças, ser feita pelo Modbus RS485.



**Figura 02:** Esquema de ligação relé

**Fonte:** Projeto DoceBela (2021)



**Figura 03:** Esquema de ligação Modbus TCP/IP e relés

**Fonte:** Drawio (2021)

## 3.2- Comunicação Modbus TCP/IP

Para comunicação do sistema de rádio frequência (*Radio-Frequency Identification* - RFID) com os dispositivos de automação já existes ou novos, foi definido um protocolo de comunicação Modbus TCP/IP devido a facilidade de instalação e pelo fato de não afetar a estrutura de um sistema existente.

## 3.3- Programação do Arduíno

A IDE do Arduíno (ambiente de programação) é um espaço de programação amigável, dispondo de bibliotecas integradas já na IDE e também possibilitando adicionar novas, além de ser uma plataforma de fácil programação.

Será utilizado para tratar a leitura coletada através do leitor RFID, enviar essas informações para um mestre da rede.

É necessário instalar as bibliotecas para a camunicação Modbus TCP/IP e o leitor de rádio frequência (*Radio-Frequency Identification* - RFID). Foi utilizado a versão 1.8.14 do Arduíno e as bibliotecas *Ethernet.h*, *SPI.h* e Mudbus.h responsáveis pela comunicação Modbus e também as bibliotecas *Hidboot.h* e *Usbhub.h* responsáveis pelo funcionamento do leitor pela porta USB.

## 3.4- Teste do sistema

Para validar primeiro na bancada, o dispositivo foi ligado a uma IHM, antes de lêr cartão contendo os caracteres, foi feito o teste de comunicação Modbus TCP/IP, conectando a IHM e o dispositivo de leitura a um *switch* mandando um *bit* para ativar o relé de operador e o relé de produto. Após essa confirmação, foi feito o teste lendo o cartão RFID e mostrando na tela da IHM o número lido pelo dispositivo. Deixando a conexão entre o mestre e o escravo ligada durande dois dias direto para analisar se haveria a queda na comunicação ou desse algum erro.

# Resultados e discussões

## 4.1- Desenvolvimento do sistema de medição

O desenvolvimento foi feito em três etapas: O primeiro foi feito o teste com todos os shields que formam o sistema de medição para receber as informações do leitor RFID. Após a validação dessa etapa, foi feito o teste usando o protocolo de comunicação Modbus TCP/IP para comunicar o sistema de medição com o mestre da rede, a fim de passar e receber todas as informações coletadas. Por fim, o teste de comunicação com o mestre da rede, enviando informações da IHM para o Arduino, para validar a comunicação entre eles.

O programa do arduino para comunicar o Modbus TCP/IP além da biblioteca de comunicação, foi configurado um IP e um *Gateway* para a IHM comunicar com o dispositivo. Além disso foi necessário a utilização de um *shield* Ethernet W5100.

Utilizando um leitor RFID de bancada, que funciona como um teclado, o arduino faz a leitura e transmite para a IHM os números lidos do cartão, porém dividido em três partes e utilizando a variável tipo INT (inteiro), antes de dividir em três partes, ele fornece a informação em formato *String,* que teve que ser convertido para enviar na rede. A IHM recebe as três variáveis do tipo INT, converte novamente para *String* e logo após junta as três variáveis e pesquisa no banco de dados se o cartão apresentado contendo aqueles números lidos está cadastrado. Caso esteja, ela envia um *bit* para o Arduino confirmando o operador/produto.

## 4.2- Testes de bancada

Em bancada, os testes de comunicação entre os *Shields* foram feitos, e logo após o teste com o leitor RFID de bancada, mandando apenas imprimir e mostrar no monitor serial do arduino, após o sucesso na leitura, foi feito a comunicação Modbus TCP/IP entre uma IHM e o Arduino, o testes de comunicação foram realizados através de envio de *bit* da IHM para o Arduino e do Arduino para a IHM, nos testes inicias, foi feito a tentativa de envio de uma *String* inteira, contendo dez caracteres, porém a IHM não conseguiu pegar esse valor, foi necessário dividir a essa *String* em três partes e removendo o zero inicial, já que o Arduino não conseguia dividir contendo o zero inicial, foi removido no próprio configurador do leitor. Assim a IHM foi capaz de receber as informações enviadas pelo microcontrolador.

Com a comunicação funcionando, foram feitos testes de leitura simulando o dia a dia da fazenda, feito a leitura no sistema de medição, a IHM recebeu a informação e como o operador e o produto lido já estavam cadastrados, mostrou na tela da IHM o nome do operador e o produto lido, a IHM mandou para o Arduino dois sinais para informar os cartões apresentados estão cadastrados na base de dados. Acionando assim dois indicadores luminosos. Apresentando um cartão em branco, o operador e produto é deslogado automaticamente possibilitando passar um novo usuário. Vários cartões foram lidos e não ocorreu nenhum erro durante o teste.

Para validação da comunicação entre o Arduino e IHM, o sistema ficou comunicando, sem desligar durante cinco dias consecutivos e testes de leitura foram realizados durante esse período de comunicação.

## 4.3- Operação

Após o sucesso de todos os testes de bancada, foram instalados seis dispositivos de leitura RFID na fazenda Doce Bela (Local), utilizando cabos Ethernet CAT5 e um *switch* para ligar todos os seis sistemas de medição RFID distribuídos no galpão, separando a comunicação do mestre da rede com os dispositivos pela rede Modbus TCP/IP e a comunicação do mestre da rede via RS485 com o CLP (Controlador Lógico Programável) (Figura 04-A).

Localizado dentro dos quadros (Figura 04-B) toda a leitura era feita aproximando o cartão por fora do quadro de comando que abrigava o sistema (Figura 05) localizados nas linhas de produções e saída dos engaços dos cachos de banana.

Aproximadamente trinta metros de distância, onde se localizava cinco dispositivos, a IHM fazia a comunicação sem nenhum erro de rede, porém tinha um ponto de aproximadamente setenta metros de distância onde se localizava o dispositivo responsável pela contagem de engaços na entrada do galpão, o sistema a priori funcionou perfeitamente, se passou um dia após a integração do sistema de medição com o mestre da rede, e começou aparecer erros na comunicação Modbus TCP/IP onde a IHM mostrava o dispositivo que estava com erro, acusando o sistema seis, que era responsável pela contagem dos engaços, com um dispositivo com erro, o sistema ficava instável e afetava os demais dispositivos.

Foram revisados os códigos de programação tanto do Arduino, quando da IHM. E não foi encontrado nada que pudesse ocasionar o erro. Foi decidido trocar o cabo Ethernet CAT5 para um Ethernet CAT6 blindado, pois foi notado que na eletrocalha disponibilizada para cabos de comunicação, passava também uma rede trifásica. Antes de trocar toda estrutura da rede Modbus TCP/IP, foram feitos testes com oitenta metros de cabo blindado, ligado no *switch* e no dispositivo de medição. O teste foi um sucesso, já que durante cinco horas não teve queda na comunicação.

Após a troca de toda a estrutura de rede ser feita, foi novamente ligado o sistema e ficou durante três dias sem erro algum e funcionando corretamente.

O sistema permaneceu funcionando durante seis meses e começou a apresentar instabilidade mais uma vez, então foram adaptados botões individuais em cada dispositivo para que quando tivesse o erro, o operador pudesse reiniciar o sistema manualmente, e a comunicação ser restaurada.

Passaram mais três meses e a instabilidade aumentou e o reset já não restaurava a comunicação. Todos os dispositivos foram removidos e substituídos por IHM’s para deixar o sistema funcionando no cliente. Voltaram pra bancada para analisar de perto os erros que estavam ocorrendo e foi descoberto que a biblioteca de comunicação já estava desatualizada. Foram feitas as alterações e incluindo o *watchdog* utilizado para verificar a todo momento os testes de comunicação Modbus e caso viesse a falhar, o sistema reiniciaria automáticamente a comunicação Modbus.

## 4.4- Dispositivos do mercado

Ao aparecer a demanda do dispositivo de leitura, buscamos empresas que já trabalham com esse tipo de dispositvo, e ao cotar percebemos que o custo era elevado, e o cliente recuou na hora da proposta. O valor do sistema já pronto ficou na faixa de R$14.000,00 enquanto o nosso sistema mesmo utilizando 20 HH (hora homem) de dois desenvolvedores, não chegou a ½ do valor de mercado (Tabela 01).

Existe um dispositivo que realiza a leitura do RFID e trata as informações para enviar via RS232, onde o mestre da rede consegue acessar (VIA ONDA RFID). A diferença do sistema de medição transcrito no presente trabalho, é o custo-benefício. Levando em consideração a utilização de um microcontrolador e leitor RFID de mesa. Outro diferencial é a utilização da comunicação Modbus TCP/IP onde a implementação do mesmo, não afeta a estrutura da rede, ao contrário do RS232 que na maioria dos dispositivos é limitado a uma entrada.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cotação realizada dia 29/01/2020 - MARCHI INDUSTRIA/VIA ONDA | | | | |
| Quantidade | Produto | Descrição | Valor und | Valor Total |
| 1 UND | MI-ID40 | Leitor RFID UHF | R$6.500,00 | R$6.500,00 |
| 2 UND | Antena UHF | Antena RFID | R$451,00 | R$902,00 |
| 7 UND | Leitor Fixo RFID UHF | Leitor RFID de mesa | R$900,00 | R$6.300,00 |
| 2 UND | Cabo RFID | Cabo de coaxial para antena | R$67,55 | R$135,10 |
| TOTAL : R$13.837,10 | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cotação realizada dia 10/02/2020 – MPA Soluções de Engenharia | | | | |
| Quantidade | Produto | Descrição | Valor und | Valor Total |
| 7 UND | W5100 | Shield Ethernet para arduino | R$60,00 | R$420,00 |
| 7 UND | USB Host Arduino | Shield USB para arduino | R$90,00 | R$665,00 |
| 7 UND | Arduino Uno | Microcontrolador | R$35,00 | R$245,00 |
| 7 UND | Leitor Fixo RFID 13.56Mhz | Leitor RFID de mesa | R$158,00 | R$1106,00 |
| 50 UND | Cartão RFID13.56Mhz | Cartão para identificação RFID | R$2,49 | R$124,50 |
| 40 H | Hora Homem | Horas para desenvolvimento | R$100,00 | R$4.000,00 |
| TOTAL : R$6.560,50 | | | | |

**Tabela 01:** Levantamento de custo Equipamento de mercado – Equipamento desenvolvido.

**Fonte:** Projeto DoceBela (2020)

|  |
| --- |
|  |
| (A) |
|  |
| (B) |

**Figura 04: (**A) Comunicação Modbus RS485 (B) Comunicação Modbus TCP/IP dos dispositivos

**Fonte:** Projeto DoceBela (2020)

|  |
| --- |
|  |

**Figura 05:** Leitor RFID na linha de embalagem

**Fonte:** Projeto DoceBela (2020)

# Considerações Finais

Para alcançar a comunicação e integrar com o sistema de automação existente, foi necessário a utilização de um microcontrolador confiável e com grande flexibilidade com a utilização do *Shield* Ethernet e do *Shield* USB, características encontradas na plataforma de prototipia Arduino UNO, para tratar os dados coletatos de um leitor RFID. Além de entregar o que é necessário, o custo benefício é muito bom pois é composto de componentes de fácil acesso.

A comunicação com a rede modbus TCP/IP, a leitura através do microcontrolador e a comunicação entre os *shields*, foram alcançados utilizando um protótipo na área do cliente, vivenciando realmente a rotina dos operadores utilizando o sistema. Apesar dos erros apresentados, todos foram corrigidos e atualizados, buscando melhorias para trazer mais confiança para o projeto em questão.

Melhorias na biblioteca foram feitas para que a comunicação com o mestre da rede não se perca durante a rotina de operação. Os testes realizados durante nove meses, foram essenciais para a validação do projeto.

A melhoria planejada é a redução da quantidade de *shields* utilizados no projeto, para reduzir ainda mais o tempo de resposta do dispositio e reduzir a quantidade de bibliotecas usadas.

# Referências

ALTUS. Conhecendo os protocolos Modbus TCP, EtherNet/IP e PROFINET. *Página da Altus Sistemas de Automação S.A.*, 2021. Disponivel em: <https://www.altus.com.br/post/411/conhecendo-os-protocolos-modbus-tcp-2c-ethernet-2fip-e-profinet>.

ANDREA OLIVEIRA. Agroindústria. *Página da Industria Rural,* 2021. Disponivel em: <https://www.industriarural.com.br/agroindustria/quais-sao-os-tipos-de-agroindustria>.

ASSESSORIA DE COMUNICAÇÃO. Agroindústria de Linhares. *Página do Site de Linhares.,* 2019. Disponivel em: <https://www.sitedelinhares.com.br/noticias/geral/agroindustrias-de-linhares-poderao-comercializar-seus-produtos-em-todo-o-estado>.

AUTOMACAO E CARTOONS. Protocolo Modbus. *Página da Automacao e Cartoons,* 2018. Disponivel em: <https://automacaoecartoons.com/2018/11/23/protocolo-modbus/>.

CÉSAR CASSIOLATO. Redes Industriais. *Página da Smar,* 2010. Disponivel em: <https://www.smar.com/brasil/artigo-tecnico/redes-industriais>.

FLIP FLOP. Arduino. *Página do Flip Flop.*, 2021. Disponivel em: <https://www.filipeflop.com/?s=Arduino+&post\_type=post>.

MARINA MARTHA. Historia RFID. *Página do Cabtec,* 2014. Disponivel em: <https://cabtec.com.br/historia-rfid>.

NI. The Modbus Protocol. *Página da Ni.,* 2021. Disponivel em: <https://www.ni.com/pt-br/innovations/white-papers/14/the-modbus-protocol-in-depth.html>.

SIMPLY. *Página da Simply Modbus,* 2020. Disponivel em: <https://simplymodbus.ca/>.

SPECS. *Página do Modbus,* 2015. Disponivel em: <https://modbus.org/specs.php>.

VIA ONDA RFID. Leitores-2. *Página da Via onda RFID.* Disponivel em: <https://www.viaondarfid.com.br/leitores-2/>.

APÊNDICE I

Código Fonte Arduino

//Bibliotecas para o funcionamento do Modbus TCP

#include <Ethernet.h>

#include <SPI.h>

#include "Mudbus.h"

//Bibliotecas para o funcionamento do Leitor RFID pela porta USB

#include <hidboot.h>

#include <usbhub.h>

// Satisfy the IDE, which needs to see the include statment in the ino too.

#ifdef dobogusinclude

#include <spi4teensy3.h>

#endif

#include<stdio.h>

#include <stdlib.h>

Mudbus Mb;

//Function codes 1(read coils), 3(read registers), 5(write coil), 6(write register)

//signed int Mb.R[0 to 125] and bool Mb.C[0 to 128] MB\_N\_R MB\_N\_C

//Port 502 (defined in Mudbus.h) MB\_PORT

char Tag\_vet[20];

int j = 0;

long int Id\_rfid = 0; //Valor completo lido pelo leitor rfid (9 digitos)

String Id\_rfid\_string = "0"; //String contendo o valor completo lido pelo rfid (9 chars)

String Id\_rfid\_string\_1 = "0"; //YYY.XXX.XXX - String contendo os três primeiros números da direita para a esquerda

String Id\_rfid\_string\_2 = "0"; //XXX.YYY.XXX - String contendo os três números do meio

String Id\_rfid\_string\_3 = "0"; //XXX.XXX.YYY - String contendo os três últimos números da direita para a esquerda

int Id\_rfid\_1 = 0; // Tranformação da primeira string em um número inteiro

int Id\_rfid\_2 = 0; // Tranformação da segunda string em um número inteiro

int Id\_rfid\_3 = 0; // Tranformação da terceira string em um número inteiro

int Log\_OP; // INT para ascender sinaleiro de logoff do operador

int Log\_PROD; // INT para ascender sinaleiro de logoff do produto

int Limpa; // INT para apagar a leitura do RFID

unsigned long t0;

unsigned long t1;

unsigned long t;

int est=1;

class KbdRptParser : public KeyboardReportParser

{

void PrintKey(uint8\_t mod, uint8\_t key);

protected:

void OnControlKeysChanged(uint8\_t before, uint8\_t after);

void OnKeyDown (uint8\_t mod, uint8\_t key);

void OnKeyUp (uint8\_t mod, uint8\_t key);

void OnKeyPressed(uint8\_t key);

};

void KbdRptParser::PrintKey(uint8\_t m, uint8\_t key)

{

MODIFIERKEYS mod;

\*((uint8\_t\*)&mod) = m;

};

void KbdRptParser::OnKeyDown(uint8\_t mod, uint8\_t key)

{

uint8\_t c = OemToAscii(mod, key);

if (c)

OnKeyPressed(c);

}

void KbdRptParser::OnControlKeysChanged(uint8\_t before, uint8\_t after) {

MODIFIERKEYS beforeMod;

\*((uint8\_t\*)&beforeMod) = before;

MODIFIERKEYS afterMod;

\*((uint8\_t\*)&afterMod) = after;

}

void KbdRptParser::OnKeyUp(uint8\_t mod, uint8\_t key)

{

PrintKey(mod, key);

}

void KbdRptParser::OnKeyPressed(uint8\_t key)

{

Tag\_vet[j] = (char)key; //atribuição do vetor de char (String)

j++;

if (j >= 10) {

long int Tag\_num = strtol (Tag\_vet,NULL,10); //converter a string de numeros em inteiro

Id\_rfid=Tag\_num;

Id\_rfid\_string=String(Id\_rfid);

Id\_rfid\_string\_1 = String(Id\_rfid\_string[0])+String(Id\_rfid\_string[1])+String(Id\_rfid\_string[2]);

Id\_rfid\_string\_2 = String(Id\_rfid\_string[3])+String(Id\_rfid\_string[4])+String(Id\_rfid\_string[5]);

Id\_rfid\_string\_3 = String(Id\_rfid\_string[6])+String(Id\_rfid\_string[7])+String(Id\_rfid\_string[8]);

Id\_rfid\_1 = Id\_rfid\_string\_1.toInt();

Id\_rfid\_2 = Id\_rfid\_string\_2.toInt();

Id\_rfid\_3 = Id\_rfid\_string\_3.toInt();

est=0;

t0=millis();

digitalWrite(A0, HIGH);

digitalWrite(A1, HIGH);

//TESTE LEITURA RFID e CODIFICAÇÃO

//Serial.println(Tag\_num); //Tag do cartão RFID (inteiro) 10 digitos

Serial.println();

//Serial.println(Id\_rfid\_string);

Serial.println(Id\_rfid\_1);

Serial.println(Id\_rfid\_2);

Serial.println(Id\_rfid\_3);

j = 0;

}

};

USB Usb;

//USBHub Hub(&Usb);

HIDBoot<USB\_HID\_PROTOCOL\_KEYBOARD> HidKeyboard(&Usb);

KbdRptParser Prs;

void setup()

{

pinMode(A0, OUTPUT); //Pino do sinaleiro de log do operador

pinMode(A1, OUTPUT); //Pino do sinaleiro de log do produto

uint8\_t mac[] = { 0x97, 0xA8, 0xD0, 0x06, 0x57, 0x12 };

uint8\_t ip[] = { 192, 168, 2, 209 };

uint8\_t gateway[] = { 192, 168, 1, 1 };

uint8\_t subnet[] = { 255, 255, 0, 0 };

Ethernet.begin(mac, ip, gateway, subnet);

//Avoid pins 4,10,11,12,13 when using ethernet shield

// delay(5000);

Serial.begin(9600);

// delay(500);

#if !defined(\_\_MIPSEL\_\_)

while (!Serial); // Wait for serial port to connect - used on Leonardo, Teensy and other boards with built-in USB CDC serial connection

#endif

Serial.println("Start");

if (Usb.Init() == -1)

Serial.println("OSC did not start.");

// delay(500);

HidKeyboard.SetReportParser(0, &Prs);

}

void loop()

{

Usb.Task();

Mb.Run();

Mb.R[0] = Id\_rfid\_1;

Mb.R[1] = Id\_rfid\_2;

Mb.R[2] = Id\_rfid\_3;

Log\_OP = Mb.R[3];

Log\_PROD = Mb.R[4];

Limpa = Mb.R[5];

digitalWrite(A1, Mb.C[4]);

if(est==0){

t1=millis();

t=t1-t0;

if(t>100){

digitalWrite(A0, LOW);

digitalWrite(A1, LOW);

}

if(t>200){

digitalWrite(A0, HIGH);

digitalWrite(A1, HIGH);

est=1;

}

}

if(Limpa == 1){

Id\_rfid\_1 = 0;

Id\_rfid\_2 = 0;

Id\_rfid\_3 = 0;

Mb.R[5]= 0;

}

if(Log\_OP == 1 and est==1){

digitalWrite(A0, LOW);

}

if(Log\_OP == 0 and est==1){

digitalWrite(A0, HIGH);

}

if(Log\_PROD == 1 and est==1){

digitalWrite(A1, LOW);

}

if(Log\_PROD == 0 and est==1){

digitalWrite(A1, HIGH);

}

}

APÊNDICE II

Código Fonte IHM (RFID)

void RFID1\_TAG3\_ValueChange(System.Object sender, Core.Api.DataSource.ValueChangedEventArgs e)

{

if(RFID1\_TAG3.Value != 0 && RFIDDesabilitado.Value == false){

//Busca informações do arduíno e limpa os campos4

RFID1StoreTag1.Value = RFIDTag\_Num2Text(RFID1\_TAG1.Value);

RFID1StoreTag2.Value = RFIDTag\_Num2Text(RFID1\_TAG2.Value);

RFID1StoreTag3.Value = RFIDTag\_Num2Text(RFID1\_TAG3.Value);

RFID1\_LIMPA.Value = 1;

//BLINK ARDUINO -> RFID LIDO

// Busca operador no banco de dados

RFID1TagNum.Value = "0"+RFID1StoreTag1.Value+RFID1StoreTag2.Value+RFID1StoreTag3.Value;

Globals.TagsOperador.LoadRecipe(RFID1TagNum.Value,false);

// Encontrou operador

if(TagNome.Value.ToString() != ""){

if(RFID1Operador.Value != ""){

if(RFID1Operador.Value != TagNome.Value.ToString()){

DLContLoginOp.Value = RFID1Operador.Value;

DLContLoginBal.Value = 1;

DLContLoginStatus.Value = 0;

Globals.ControleLogin.Log();

}

}

RFID1Operador.Value = TagNome.Value.ToString();

// Logado como operador

RFID1\_LOG\_OP.Value = 1;

B1\_LOGADO.Value = 1;

DLContLoginOp.Value= RFID1Operador.Value;

DLContLoginBal.Value = 1;

DLContLoginStatus.Value = 1;

Globals.ControleLogin.Log();

// Não encontrou operador

}else{

// Busca projeto no banco de dados

Globals.TagsProjeto.LoadRecipe(RFID1TagNum.Value,false);

// Encontrou projeto

if(TagProjID.Value.ToString() != ""){

RFID1Produto.Value = TagProjID.Value.ToString()+TagProjEq.Value.ToString();

// Logado como projeto

RFID1\_LOG\_PROD.Value = 1;

// Não encontrou a tag

}else{

if(RFID1Operador.Value != ""){

DLContLoginOp.Value = RFID1Operador.Value;

DLContLoginBal.Value = 1;

DLContLoginStatus.Value = 0;

Globals.ControleLogin.Log(); }

// Não está logado

RFID1Operador.Value = "";

RFID1Produto.Value = "";

// Variáveis Arduino

RFID1\_LOG\_OP.Value = 0;

RFID1\_LOG\_PROD.Value = 0;

// BLINK ARDUINO -> NÃO ENCONTRADO