CEEL - ISSN 2596-2221
Universidade Federal de Uberlândia
25 a 29 de novembro de 2019

IMPLEMENTAÇÃO DE UM DISPOSITIVO MEDIDOR DE VAZÃO COM COMUNICAÇÃO VIA RS485

Gabriel Felipe Vieira de Sousa*[1], Marcelo Barros de Almeida[1]

[1]FEELT – Universidade Federal de Uberlândia

***Resumo* – Sistemas de medição autônomos estão ganhando notoriedade em diversos setores industriais. Com o avanço de tecnologias de comunicação em redes, é possível monitorar e atuar equipamentos industriais através de variados protocolos de comunicação já existentes. Neste contexto, este trabalho almeja implementar um dispositivo embarcado responsável por realizar a contagem de litros de maneira autônoma utilizando um microcontrolador ARM Cortex M. Ademais, o dispositivo realizará comunicação com o mestre através de um barramento de rede do tipo RS485 e protocolo mestre-escravo. A verificação do desempenho da rede será analisada através de leituras e escritas cíclicas.**

***Palavras-Chave* – Comunicação de redes, embarcados, RS485, medidor industrial de vazão.**

IMPLEMENTATION OF A FLOW MEASUREMENT DEVICE WITH COMMUNICATION BASED ON RS485

***Abstract* - Autonomous measurement systems are gaining notoriety in many industrial sectors. With the advancement of communication technologies in networks, it is possible to oversee and operate industrial equipment through various communication protocols already existing. In this context, this work aims to implement an embedded device responsible for autonomously counting liters using a microcontroller ARM Cortex M through a network bus based on RS485 interface and a master-slave protocol. Verification of network performance will be analyzed by cyclic readings and writes.**

Keywords* - *Network communication, embedded systems, RS485 interface and industrial flow meter.

I. INTRODUÇÃO

Um crescente avanço das tecnologias da informação vem sendo presenciado nas últimas décadas, o que tem sido possível, em parte, graças ao desenvolvimento da microeletrônica [1] e das redes de comunicação. Dessa maneira, o setor de automação industrial vem se beneficiando de tal avanço, algo que pode ser notado através do surgimento de novas técnicas de implementações e funcionalidades que otimizam a produção industrial [2].

O aumento significativo do poder de processamento de computadores e microcontroladores, o surgimento de novos padrões de redes industriais com protocolos bem definidos e notavelmente eficientes, a ascensão de sistemas embarcados e implementação em *hardware* são algumas das diversas alternativas existentes para a resolução de problemas em processos industriais [3].

Dentre os métodos de comunicação entre dispositivos, o protocolo RS485 vem ganhando notoriedade por apresentar vantagens em relação ao RS232, que frequentemente é criticado por sua falta de imunidade a ruídos que interferem na comunicação e topologia ponto a ponto, enquanto que os sinais via RS485 são diferenciais, com a transmissão de dados feita sobre duas linhas, sendo uma com sinal positivo e outra com sinal negativo, de modo que o receptor do RS485 realize a comparação entre a diferença de tensão das linhas [4]. Diferente do RS232, o RS485 permite a criação de uma topologia em barramento, com vários dispositivos compartilhando o mesmo meio físico. Vale lembrar que protocolos como CAN, Modbus e Profibus DP utilizam RS485 como meio físico, evidenciando as mesmas qualidades.

Mediante a grande capacidade dos atuais microcontroladores, torna-se viável a implementação de sistemas embarcados autônomos que realizam tarefas específicas e utilizam protocolos de comunicação para se comunicar com mestres diversos. Neste contexto, o presente trabalho almeja implementar um dispositivo capaz de mensurar a quantidade total de litros desejada pelo operador. O sistema será controlado via protocolo implementado sobre um meio físico baseado em RS485.

II. DISPOSITIVO INTEGRADOR DE VAZÃO

O dispositivo a ser implementado tem por finalidade aplicações em pequenas indústrias que necessitam contabilizar, com exatidão, uma determinada quantidade de litros a ser definida pelo operador. O sistema fará a aquisição de dados de pulsos emitidos por um sensor de

*gabrielfelipe@ufu.br

vazão e converterá o valor em litros totais, a partir do momento em que o usuário deseja iniciar a operação. O término do processo se dá no instante em que a quantidade total de litros atinja o valor informado pelo usuário. O sistema pode ser operado por painel físico do dispositivo ou ainda, remotamente, através de comandos.

O dispositivo deve ser capaz de medir com precisão fluídos de baixa viscosidade, como água, glicerina, combustíveis e ar, desde que a vazão de saída seja constante. Com o intuito de desenvolver um dispositivo de baixo custo e com alta capacidade de processamento, foi viável utilizar a plataforma de hardware baseada em micontroladores ARM.

A. STM32 e Geração de Código

As plataformas STM32 são produzidas pela empresa *STMicroeletronics* ST, baseados em microcontroladores de diversas famílias ARM, como ARMv6-M, ARMv7-M e, mais recentemente, ARMv8-M, todas com arquitetura de 32 bits. Este trabalho utilizou o STM32F103C8T6 como principal controlador. Baseado no núcleo ARM *Cortex* M3 (ARMv7-M), este microcontrolador apresenta uma boa relação entre desempenho e custo baixo, com frequência de operação é de 72 Mhz. O microcontrolador conta com memórias embarcadas de alta velocidade (memória *flash* de 64 *Kbytes* e SRAM de 20 *Kbytes*). Além disso, o dispositivo oferece dois ADC's de 12 bits, três temporizadores de uso geral de 16 bits mais um temporizador PWM, bem como interfaces de comunicação padrão e avançadas (dois I2C's e SPI's, três interfaces USART's, um USB e um CAN) [5].

Para a implementação do *firmware* será utilizado o software STM32CubeMX, cuja funcionalidade é definir a função de cada pino do microcontrolador, configuração de clock, interrupções, entre outras coisas. Ao final desse processo, um esqueleto básico de código é gerado, com as configurações de partida dos periféricos selecionados.

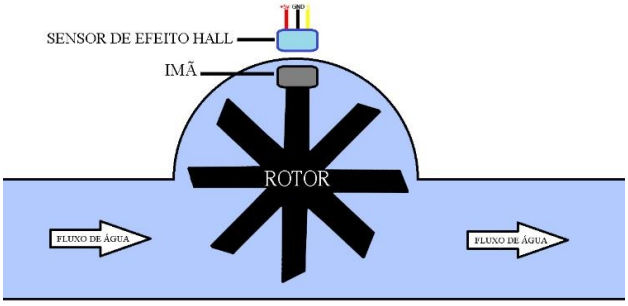
B. Periféricos de Hardware

Além do microcontrolador, faz-se necessário conhecer cada componente fundamental ao projeto e seu respectivo funcionamento.

1) *Sensor de Vazão*

Com o propósito de realizar uma medição precisa, será adotado o sensor de fluxo YF S-201. O dispositivo contém internamente um rotor com hélices, o qual é responsável por realizar emissão de pulsos que é diretamente proporcional à quantidade de fluido passado por ele [5]. Na Figura 1 é ilustrado o princípio de funcionamento do dispositivo.

Figura 1: Representação esquemática do funcionamento do sensor de fluxo.

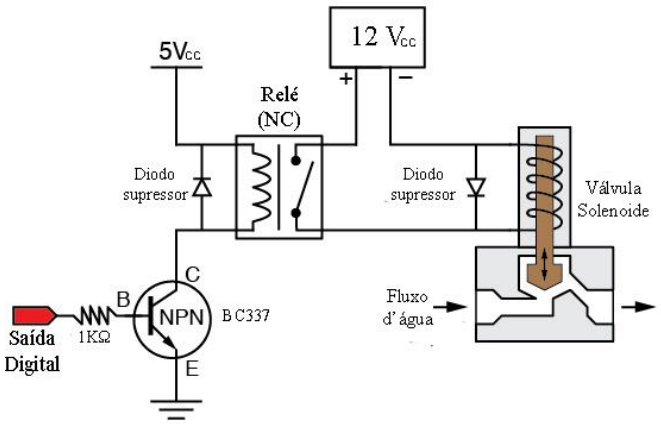


Devido à presença de um sensor de efeito Hall magnético, a cada revolução do rotor é gerado um pulso elétrico. Usando um algoritmo de conversão o qual será explanado na sessão IV, é possível determinar a vazão e a quantidade total de litros através dos pulsos [5]. Além disso, deve se atentar à frequência de operação, que deve ser compreendida entre 16 Hz e 90 Hz.

2) *Válvula solenoide*

Com o objetivo de interromper a passagem do fluido, deve ser utilizada uma válvula solenoide normalmente fechada ou NC (NC: *normally closed*) de 12 V. Visto que o microcontrolador utilizado opera com tensão de 3,3 V e o relé utilizado é acionado através de um sinal de 5 V, também é necessário um transistor para acionamento do relé [6]. Adicionalmente, um relé de 12 V fornece a tensão necessária para controle da válvula, permitindo que o comando enviado pelo microcontrolador seja efetivado. Na Figura 2 é possível visualizar o circuito utilizado.

Figura 2: Esquemático de hardware utilizado para energizar a válvula solenoide.



A válvula utilizada possui algumas especificações de uso, como pressão de trabalho entre 0,02 a 0,8 MPa e temperatura máxima do fluido de 100 ºC.

3) *Display Alfanumérico*

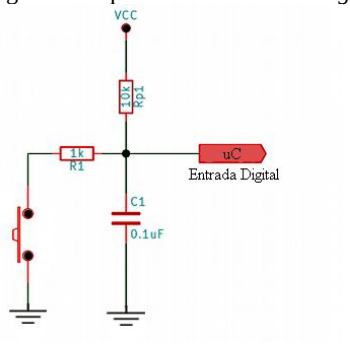
Será utilizado ainda um visor LCD 16x2, responsável por informar ao usuário que estiver próximo ao painel de comando a quantidade total de litros que já foram

contabilizados bem como a configuração inicial do processo.

4) Botões de acionamento não retentivos

Com o objetivo de realizar operações através de um painel físico, serão utilizados três botões do tipo *push-buttons*. Dois são usados para reduzir e aumentar a quantidade de litros a serem contabilizados e o terceiro serve para iniciar o processo. Visto que será realizado o tratamento de interrupções, se faz necessário a utilização de uma técnica de *debouncing*, como mostra a Figura 3, para evitar repiques no acionamento [7].

Figura 3: Esquemático de *debouncing*.



A constante de tempo (τ) é calculada por:

$$\tau = R \times C \quad (1)$$

Onde,

R - Resistência utilizada (R_{p1})

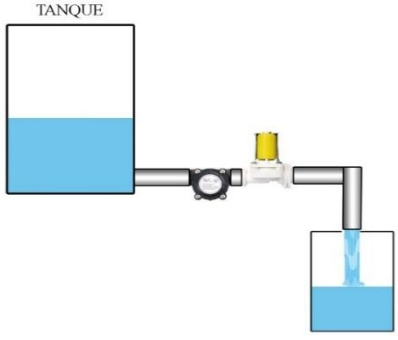
C - Capacitância (C_1)

Os valores adotados para R e C devem ser suficientes para reduzir problemas de repiques mecânicos. A constante τ resulta no tempo necessário para carregar o capacitor de 0 a 63,2% da tensão aplicada no capacitor, que no limite, corresponde a, aproximadamente, 2/3 da tensão de alimentação (3,3V).

C. Protótipo do Tanque

Com o intuito de realizar ensaios experimentais, foi adotado um tanque aberto, conforme mostra a Figura 4. As dimensões do tanque correspondem a 35,0 cm de altura por 19,0cm x 17,5cm de base, o que resulta em 9,975 litros.

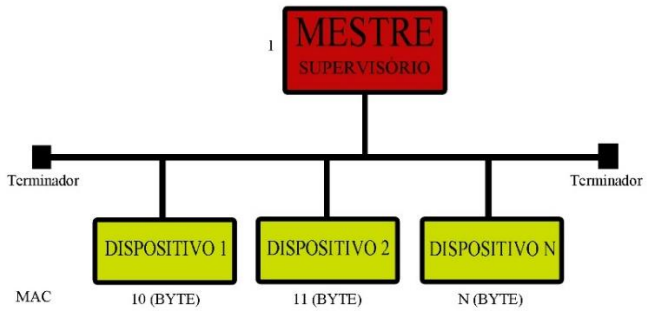
Figura 4: Representação esquemática da interconexão do sensor YF S-201 e da válvula solenoide com tanque.



III. PROTOCOLO DE COMUNICAÇÃO

A visão geral do protocolo implementado consiste em um mestre (ou supervisório) e um barramento onde localizam-se os dispositivos escravos. Como mencionado, o meio físico será RS485, amplamente usado em outros protocolos industriais como *Profibus* DP/PA, *Modbus* e *CAN*. A Figura 5 descreve a estrutura de conexão a ser utilizada.

Figura 5: Diagrama de blocos da estrutura de comunicação em rede.



A priori, é necessário conhecer as funções da camada de enlace nessa aplicação. Suas atribuições consistem em realizar a delimitação de dados, permitir o acesso ao meio (*media access control – MAC)*, promover uma entrega de dados confiável, realizar a detecção de erros, bem como o controle de acesso [8]. É possível notar, na Figura 5, que cada dispositivo contém um endereço no barramento de um byte (0 a 255), devendo esse endereço ser único para cada dispositivo. Essa topologia é conhecida também como *multidrop*. A detecção de erros será realizada através do mecanismo de checagem de redundância cíclica (CRC), de 2 Bytes, seguindo o polinômio empregado no Modbus (CRC16-ANSI) [9]. A delimitação do quadro será definida pela lógica de *Byte Stuffing* do Protocolo de Ponto a Ponto (PPP) [10].

Esta estrutura garante que, caso exista uma repetição durante a transmissão, um byte especial conhecido como caractere de escape (ESC) será então adicionado na estrutura do quadro, antes do caractere especial. Outros protocolos, como o HDLC (*High-Level Data Link Control)*, utilizam técnicas parecidas para comunicação em links seriais [8].

Para a interconexão *multitrop* entre o mestre e os dispositivos, será utilizado um *transceiver* USB/RS485, que será capaz de interligar o computador na rede RS485 via porta USB. Por conseguinte, será usado um segundo *transceiver* que converterá RS485 para TTL ou CMOS, sendo esse dispositivo acoplado no microcontrolador.

A seguir, será apresentado o protocolo e os comandos a serem implementados.

Visando tratar de forma geral os dispositivos na rede, o protocolo implementado tem uma série de comandos que permitem a descoberta do dispositivo e de suas capacidades, sem que o mestre seja alimentado previamente com as características do escravo, como é feito em redes do tipo

Profibus DP através dos arquivos de configuração conhecidos como GSD [11].

A. Versão do Protocolo

Assim como encontrado no protocolo IPv4, o primeiro comando consiste em determinar a versão utilizada do protocolo. Neste registro é armazenado qual a versão da interface do dispositivo será implementada.

B. Identificação do Dispositivo

Este comando será responsável por realizar a identificação do nome do dispositivo, sua versão de *hardware*, informações sobre o fabricante bem como a quantidade de pontos a serem monitorados.

C. Descrição dos Pontos

Atrelados ao dispositivo escravo, sensores, válvulas e registros serão monitorados via rede. Os periféricos a serem monitorados e/ou atuados são denominados de "*pontos*".

O intervalo de registro para a descrição dos pontos é usado para fornecer informações sobre um ponto específico, como nome do ponto, acesso a direitos, tipo de dados e unidade. As unidades seguem o padrão *Hart*, variando de 0 a 253. Quanto ao suporte de dados, a Tabela I apresenta os tipos de dados suportados bem como seu respectivo tamanho.

Tabela I: Tipos de dados suportados

Tipo	Acrônimo	Representação
0x00	B	*Byte*, 8 bits sem sinal
0x01	b	*Byte*, 8 bits com sinal
0x02	S	*Short*, 16 bits sem sinal
0x02	s	*Short*, 16 bits com sinal
0x04	L	*Long*, 32 bits sem sinal
0x05	l	*Long*, 32 bits com sinal
0x06	Q	*Long long*, 64 bits sem sinal
0x07	Q	*Long long*, 64 bits com sinal
0x08	F	*Float*, IEE 754 precisão única, 4 bytes
0x09	D	*Double*, IEE 754 dupla precisão, 8 bytes
0x0A	a	*Array* de caracteres (*string*), tamanho variável

Para o presente dispositivo, serão monitorados os pontos de quantidade total de litros, status de operação (ON/OFF) e *setup* de operação (quantidade de litros a serem contabilizados). A Tabela II descreve as características de cada ponto.

Tabela II: Descrição dos pontos a serem monitorados.

Ponto	Variável	Tipo	Direitos	Unidade
0	SETUP	0x02	Leitura e escrita (RW)	41 (Litros)
1	STATUS	0x00	Leitura e escrita (RW)	251 (Sem unidade)
2	LITROS	0x08	Apenas leitura (RO)	41 (Litros)

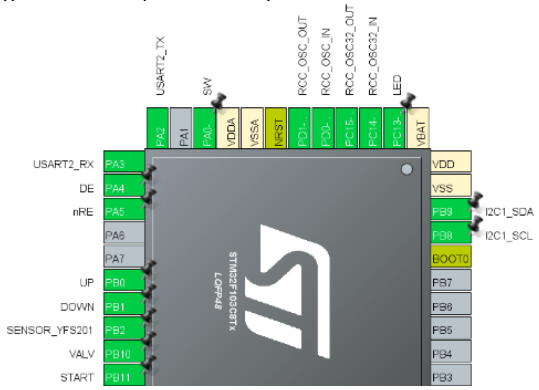
D. Funções de Leitura e Escrita nos Pontos

Com o objetivo de obter o valor atual em cada ponto ou modificar o seu valor, quando permitido, o mestre pode usar funções específicas para leitura e escrita. Dessa maneira, a função de escrita permitirá ao usuário definir a quantidade de litros a serem contabilizados e a iniciar o processo de operação. A implementação das funções não é trivial. O protocolo permite o envio de bytes via serial e valores acima de um byte devem ser divididos. Por exemplo, um dado de 16 bits deve ser enviado em dois bytes, ocupando 2 posições no *payload (*parte de dados). Já para a leitura de pontos do tipo *float*, deve se atentar que tal dado é do tipo *little endian*, enquanto que o protocolo é *big endian*, ou seja, o byte mais significativo ocupará a primeira posição. Portanto é necessário implementar funções responsáveis por realizar o envio de um dado *float* de acordo com as especificações do protocolo.

IV. IMPLEMENTAÇÃO DO FIRMWARE

A implementação do dispositivo foi feita inicialmente a partir da definição dos pinos no software STM32CubeMX, como mostrado na Figura 6.

Figura 6: Definição e *set* dos pinos utilizados na STM32.



Os pinos de UP (*PB0)*, DOWN (*PB1)* e START (*PB11)* foram definidos em modo de interrupção externa com detecção de borda de decida. Não foi preciso configurá-los como p*ull-up* no *software* devido à ligação do hardware com técnica de *debouncing*.

Para a utilização do display LCD foi viável utilizar o protocolo serial I2C, que utiliza apenas 2 fios, sendo eles SCL (PD8) e SDA (PD9). O SCL (*Clock Line)* é usado para sincronizar a transferência de dados pelo barramento I2C,

enquanto que o SDA (*Data Line)* realiza a transferência de dados uma vez sincronizados [12]. É preciso atentar-se à velocidade de *clock* do I2C, sendo que a faixa ideal para esse periférico foi de 50KHz à 80KHz. Pull-ups internos foram colocados nas linhas do I2C, evitando componentes internos.

Os pinos de PA2 ao PA5 são destinados à sincronização, transmissão e recepção de dados, via RS485.

A. Algoritmo de contabilização dos litros.

Com o objetivo de capturar os pulsos emitidos pelo sensor YF S-201, o algoritmo de contagem de litros se baseia em utilizar um temporizador, cujo a fonte de *clock* seja provinda do sensor. Dessa maneira, se faz necessário realizar a configuração do temporizador de modo a selecionar a fonte do relógio como *External Clock Mode 1* e a fonte do *trigger* como TI1_ED, cuja função é sincronizar a fonte externa de *clock*. Ademais, é necessário utilizar o modo de captura de entrada *(Input Capture Mode)* oferecido por temporizadores de uso geral e avançados que permitem calcular a frequência de sinais externos aplicados a cada um dos 4 canais oferecidos por esses temporizadores de maneira independente [13]. O canal utilizado em modo de captura de entrada foi o canal 1.

Vale ressaltar que o método utilizado para fazer a contagem de pulsos pode ser considerado uma alternativa otimizada em relação aos métodos encontrados para a plataforma Arduíno, que consistem em criar uma taxa de verificação do GPIO (*General Purpose Input/Outpu)* na qual as bordas são detectadas por *pooling/sampling* [5], e não por *hardware*, o que resulta em um maior consumo de processamento.

Por conseguinte, foi necessário utilizar a função do HAL (*Hardware Abstraction Layer*) da STM32 *HAL_TIM_GET_COUNTER(_HANDLE__)*, responsável por armazenar o valor obtido dos pulsos.

É sabido que o sensor YF-S201 pode medir no máximo 30 litros de água por minuto, o que equivale a 450 pulsos/min. Então:

$$L_r = \frac{Pulsos}{F_{calibração}} \qquad (3)$$

Onde,

$F_{calibração}$ - Fator de Calibração.

L_r - Taxa de litros.

O fator de calibração é obtido através da relação da quantidade de pulsos por segundo, que resulta em 7,5. O número total de litros d'água que fluiu através do sensor pode ser obtido dividindo L_{rate} por 60.

A partida do sistema ocorre através do botão *START* que é responsável por acionar o relé. No laço principal é feito a comparação entre a quantidade total de litros e a quantidade de litros desejada. Quando ambas variáveis são iguais, os valores de pulsos são zerados, o relé é fechado e então pode-se realizar uma nova operação.

B. Implementação das funções de rede

Os quadros de requisição e resposta possuem 3 divisões (*Header, Payload* e *Trailer)* e seguem o modelo descrito na Tabela III, no qual o campo de dados (*Payload)* varia de acordo com a requisição feita. O detalhamento dos quadros é apresentado na documentação do protocolo.

Tabela III: Descrição da estrutura do quadro de requisição.

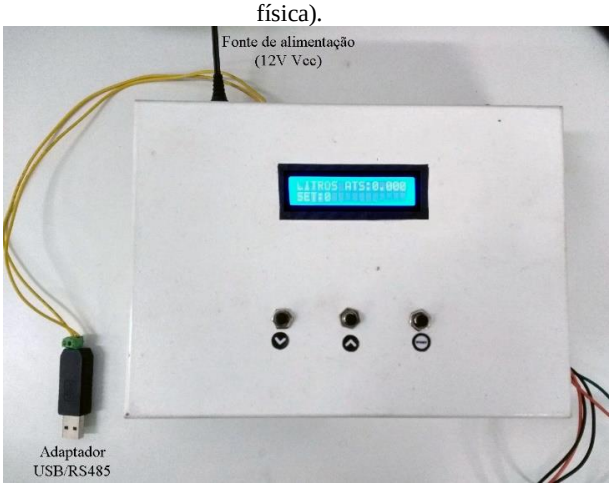
Tamanho	Descrição	
B	Endereço de destino (DST)	HEADER
B	Endereço de origem (SCR)	
B	Endereço de registro (REG)	
B	Tamanho do pacote de dados (SIZE)	
nB	Dados (PLD)	PAYLOAD
2B	CRC	TRAILER

As funções da rede foram implementadas seguindo os critérios apresentados na sessão III.

V. RESULTADOS

Após a implementação e testes do firmware, foi desenvolvido o protótipo ilustrado na Figura 7.

Figura 7: Protótipo do dispositivo medidor de litros (interface física).



Como ilustra a Figura 12, o display LCD exibe informações de litros já contabilizados (primeira linha) e o *setup* inicial de litros (segunda linha). Os botões da interface não apresentaram repiques no acionamento e respondem prontamente aos comandos. Os processadores ARM do cortex M possuem um temporizador *systick* utilizado para temporizações do sistema. Dessa maneira, a taxa de atualização do display é de 500ms e foi realizada através de uma função acionada pelo *systick*. Ademais, foi utilizado uma fonte de 12 V juntamente com um conversor de tensão para realizar a alimentação de todos os circuitos do sistema.

A contagem de litros se mostrou precisa operando tanto com vazão constante quanto com vazão variável. Durante os testes houve um pequeno *offset* na medição, o que foi corrigido com o ajuste fino do fator de calibração. A Figura

8 apresenta o sistema em funcionamento com o sensor e a válvula acoplados à saída do tanque.

Figura 8: Protótipo do dispositivo medidor de litros acoplado ao tanque.

Com o objetivo de verificar a robustez da rede, foram realizados testes de leitura e escrita cíclicas. Os testes realizados demonstraram que o sistema opera de modo satisfatório quando submetido às requisições cujo *timeout* seja igual ou superior a 500 ms.

VI. CONCLUSÕES

Este artigo apresentou o projeto de *firmware* de um dispositivo capaz de contabilizar litros. A interface de comando não apresentou problemas de *software* durante a execução de tarefas. Com base nos resultados obtidos de leitura e escrita cíclica o dispositivo apresentou um bom desempenho quando operado por rede.

Futuros trabalhos consistirão em utilizar como dispositivo mestre um CLP (Controlador Lógico Programável), afim de fazer a integração do processo com outros dispositivos industriais no mesmo barramento.

AGRADECIMENTOS

Os autores agradecem ao Laboratório de Automação, Sistemas Eletrônicos e Controle (LASEC) pelo apoio ao desenvolvimento do projeto.

REFERÊNCIAS

[1] A. d. S. Soares, "A automação e o terceiro mundo," *Administração Empresarial,* 2008.

[2] B. B. d. Oliveira, P. d. F. Monteiro e S. L. M. d. Queiroga, "Aplicabilidade dos Microcontroladores em Inovações Tecnológicas," *Congresso Norte Nordeste de Pesquisa e Inovação,* 19 Outubro 2012.

[3] C. Neves, L. Duarte, N. Viana e V. F. d. Lucena, "OS DEZ MAIORES DESAFIOS DA AUTOMAÇÃO INDUSTRIAL: AS," *II Congresso de Pesquisa e Inovação da Rede Norte Nordeste de Educação Tecnológica,* 2007.

[4] A. Glória, F. Cercas e N. Souto, "Comparison of communication protocols for low cost Internet of Things devices," *South Eastern European Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM),* pp. 1-6, 2017.

[5] R. R. Iyengar, "THE WATER FLOW MONITORING MODULE," *International Journal of Engineering Research and General Science* , vol. Volume 4, nº Issue 3, pp. 106-113, Maio-Junho de 2016.

[6] G. Recktenwald, "Wiring of the Solenoid Valves," 2018. [Online]. Available: http://web.cecs.pdx.edu/~eas199/B/howto/fishtank/wiring/solenoid_wiring.html. [Acesso em 21 Junho 2019].

[7] J. Bernardo, "O Efeito Boucing e as técnicas para Deboucing," EletronWorld, 21 junho 2016. [Online]. Available: http://eletronworld.com.br/eletronica/efeito-bounce/. [Acesso em 22 Junho 2018].

[8] K. W. R. James F. Kurose, Redes de Computadores e a Internet, São Paulo: Pearson, 2010.

[9] Modbus Tools, "An example of a C language function performing Modbus CRC16 generation.," 2019. [Online]. Available: https://www.modbustools.com/modbus_crc16.html. [Acesso em 05 Agosto 2019].

[10] W. SIMPSON, The Point-to-Point Protocol (PPP), 1994.

[11] Profinet University, "PROFINET GSD File Basics," 2018. [Online]. Available: https://profinetuniversity.com/profinet-basics/profinet-gsd-file-basics/. [Acesso em 13 Julho 2019].

[12] J.-M. Irazabal e S. Blozis, "I²C Licensing Information," 2003.

[13] C. Noviello, Mastering STM32, Leanpub, 2016.

[14] STMicroeletronics, "Microcontrollers & Microprocessors: STM32F103C8 (Datasheet)," ST, 2019. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html. [Acesso em 24 Julho 2019].