MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

GUSTAVO BATISTA DE CASTRO MATOS
MELQUÍADES RODRIGUES MACIEL

TELEMETRIA PARA SISTEMAS HÍDRICOS

GOIÂNIA - GO
2019

sistema de bibliotecas ufg

EMC

UFG

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC nº 1204/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG):

Nome completo dos autores: Melquíades Rodrigues Maciel,
Gustavo Batista de Castro Matos

Título do trabalho: Telemetria para sistemas hídricos

2. Informações de acesso ao documento:

Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF do TCCG.

Melquíades Rodrigues Maciel — **Melquíades Rodrigues Maciel**

Gustavo Batista de Castro Matos — **Gustavo Batista de Castro Matos**

Ciente e de acordo:

Marcelo Stehling de Castro

Marcelo Stehling de Castro

Data: 13/12/2019

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Versão abril de 2018

[2] As assinaturas devem ser originais sendo assinadas no próprio documento, imagens coladas não serão aceitas.

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

GUSTAVO BATISTA DE CASTRO MATOS
MELQUÍADES RODRIGUES MACIEL

TELEMETRIA PARA SISTEMAS HÍDRICOS

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Goiás como um dos pré-requisitos para a obtenção do grau de Bacharel em Engenharia Elétrica, sob a orientação do Prof. Dr. Marcelo Stehling de Castro.

GOIÂNIA – GO

2019

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Batista de Castro Matos, Gustavo
Telemetria para sistemas hídricos [manuscrito] / Gustavo Batista de Castro Matos, Melquíades Rodrigues Maciel. - 2019.
xvi, 16 f.: il.

Orientador: Prof. Dr. Marcelo Stehling de Castro.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Engenharia Elétrica, Goiânia, 2019.
Bibliografia. Apêndice.
Inclui fotografias, gráfico, tabelas, algoritmos.

1. Arduino. 2. CCK4400. 3. CCK6700E. 4. FS300A. 5. Mega2560. I. Rodrigues Maciel, Melquíades. II. Castro, Marcelo Stehling de , orient. III. Título.

CDU 621.3

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO
COORDENAÇÃO DE ESTÁGIO E PROJETO FINAL

ATA DE AVALIAÇÃO DE PROJETO FINAL

CURSO

(X) Eng. Elétrica ( ) Eng. Mecânica ( ) Eng. de Computação
( ) Projeto Final 1 ( ) Projeto Final II

AVALIAÇÃO DE PROJETO FINAL

Título do projeto: Telemetria para sistemas hídricos

BANCA AVALIADORA

Membro 1: Marcelo Stehling de Castro

Membro 2: Sérgio Granato de Araújo

Membro 3: Gustavo Dias de Oliveira

ESTUDANTES

Matrícula	Nome
201408061	Gustavo Batista de Castro Matos
201407769	Melquíades Rodrigues Maciel

NOTAS

Matrícula	Membro 1: Marcelo Stehling de Castro				Membro 2: Sérgio Granato de Araújo				Membro 3: Gustavo Dias de Oliveira				Média
	NPT	NTE	NAA	NF	NPT	NTE	NAA	NF	NPT	NTE	NAA	NF	
201408061	10	10	10	10	10	10	10	10	10	10	10	10	10
201407769	10	10	10	10	10	10	10	10	10	10	10	10	10

NPT – Nota plano de trabalho; NTE – Nota do trabalho escrito; NAA – Nota de apresentação e arguição

Para Eng. Elétrica, Mecânica e PFC2 da Eng. Da Computação: NF = 0,1 x NPT + 0,45xNTE + 0,45 x NAA

Para PFC1 da Eng. Da Computação: NF = 0,3 x NPT + 0,7 x NAA

Goiânia, 09 de dezembro de 2019.

Marcelo Stehling de Castro

Marcelo Stehling de Castro

Sérgio Granato de Araújo

Gustavo Dias de Oliveira

Telemetria para sistemas hídricos

Melquíades Rodrigues Maciel[1], Gustavo Batista de Castro Matos[2], Marcelo Stehling de Castro[3]

Universidade Federal de Goiás (UFG) – Escola de Engenharia Elétrica, Computação e de Computação - (EMC) Goiânia, Goiás, Brasil 74605-010, e-mails: melquiades@discente.ufg.br[1], gustavo_matos@discente.ufg.br[2], mcastro@ufg.br[3].

***Resumo* — Com o intuito de suprir a necessidade de realizar monitoramento do consumo de água, o projeto descrito neste artigo foi idealizado em conjunto com o Prof. Dr. Marcelo S. de Castro e como parte do programa UFG sustentável, para ser incorporado dentro da Universidade Federal de Goiás. Foram testadas soluções comercias e desenvolvido um protótipo baseado no Arduino mega, utilizando uma arquitetura simplificada no projeto, na construção, na utilização e na manutenção do sistema. O protótipo foi desenvolvido em um curto período de tempo, com componentes facilmente encontrados no mercado. A premissa desse projeto é oferecer uma solução mais simples, barata e de fácil utilização em comparação a opções comerciais existentes, com a implementação do servidor utilizando docker. Os resultados dos testes e a coleta de dados comprovaram a viabilidade da solução proposta perante as soluções comerciais.**

***Palavras Chaves* — Arduino, CCK4400, CCK6700E, FS300A, Mega2560, monitoramento, programação, YF-S201.**

***Abstract* — In order to meet the need to monitor water consumption, the project described in this article was conceived in conjunction with Prof. Dr. Marcelo S. de Castro and as part of the sustainable UFG program to be incorporated within the Federal University of Goiás. Commercial solutions were tested and a prototype based on the Arduino mega was developed using a simplified architecture in the design, construction, use and system maintenance. The prototype was developed in a short time, with components easily found on the market. The premise of this project is to offer a simpler, cheaper, and easier-to-use solution than existing business options by deploying the server using a docker. Test results and data collection proved the viability of the proposed solution against commercial solutions.**

***Key Words* — Arduino, CCK4400, CCK6700E, FS300A, Mega2560, monitoring, programming, YF-S201.**

I. INTRODUÇÃO

O desenvolvimento sustentável tem se tornado algo essencial ao mundo moderno, podendo ser definido como: o desenvolvimento que satisfaz as necessidades do presente sem comprometer a capacidade das gerações futuras de satisfazerem suas próprias necessidades.

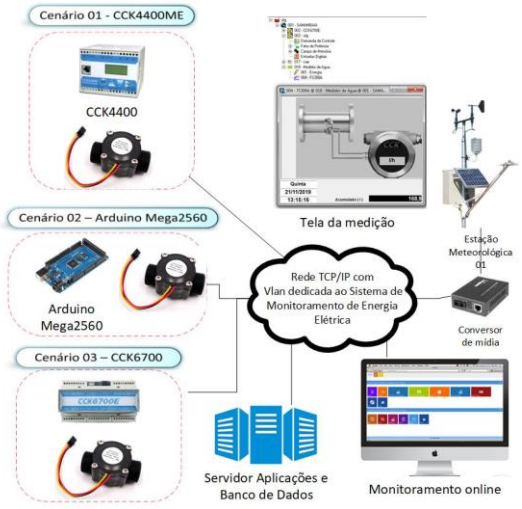
Com o crescimento da população mundial, o consumo dos recursos naturais tem aumentado de maneira desenfreada, ligando um alerta para o risco de escassez destes recursos em um futuro não muito distante. Diante deste risco, surgiu a necessidade inevitável de uma busca por soluções sustentáveis.

O aumento do desempenho de processos e a automação são fundamentais para se atingir metas de sustentabilidade e aumentar a economia. Neste contexto a medição do consumo de recursos como água, luz, gás, se torna indispensável para o controle e gerenciamento destas metas. Uma das maneiras mais modernas e eficientes de se realizar medições hoje em dia é a telemetria, que será abordada neste trabalho, com foco em sistemas hídricos [2].

A telemetria é um sistema que vem sendo utilizado há mais de 45 anos para monitoramento de sistemas hídricos, e que, com a evolução da tecnologia, fez sua utilização deixar as fabricas para ser utilizadas em ambientes mais comuns como nossas casas.

Empresas como a CCK oferecem produtos prontos para esta função como o CCK6700E e o CCK4400ME, que são capazes de realizar esses tipos de medições através de sensores de fluxo como o FS300A e o YF-S201. Além destas opções comerciais também pode-se adotar opção mais caseira, com projetos sendo executados de maneiras mais simples e direta, substituindo os aparelhos da CCK pelo Arduino por exemplo. Com isso em mente, iniciou-se o desenvolvimento desse projeto para a telemetria do consumo de água da UFG como parte do programa UFG sustentável. A Fig. 1 apresenta o esquemático do projeto que será apresentado neste trabalho.

Fig. 1. Esquemático.



Fonte: Autores

Este projeto é uma iniciativa que visa o uso eficiente de água a partir de novas tecnologias para a gestão do consumo, aplicadas a faculdade. O projeto compreende o diagnóstico preciso do uso da água, a partir de monitoramento remoto do consumo e a avaliação da possibilidade da aplicação da instalação do projeto piloto para as unidades consumidoras da UFG. Segundo o SEINFRA, aproximadamente 80% das tubulações presentes na UFG são de ¾”, outros 10% são de 1” e os últimos 10% são de 1 ½”, o que torna a escolha dos sensores FS300A e YF-S201 apropriadas para estas aplicações.

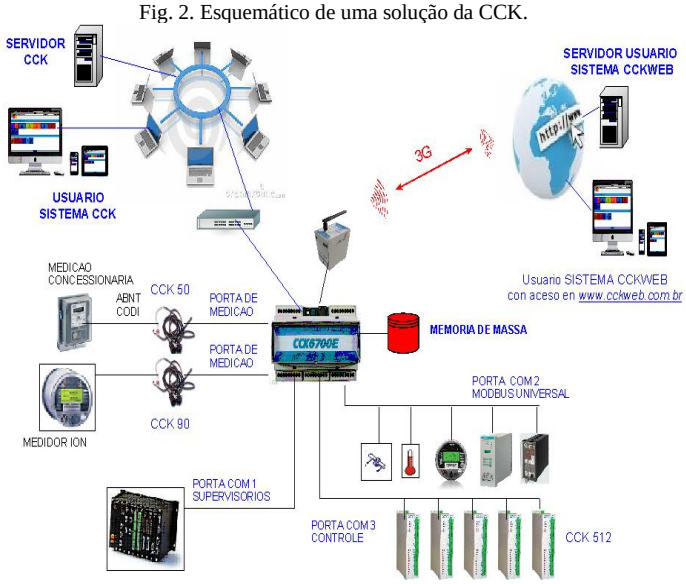
Realizou-se um estudo para utilização das opções comerciais já existentes dentro da universidade associada a um projeto desenvolvido em torno do Arduino, que possui um baixo custo e suas possibilidades para ampliação do projeto em questão.

II. Equipamentos e Materiais Utilizados

Nesta seção serão apresentados os equipamentos e materiais utilizados no projeto.

A Fig. 2 ilustra através de esquema simplificado, os equipamentos e tecnologias usados neste projeto[13].

Fig. 2. Esquemático de uma solução da CCK.



Fonte: sistema-cckweb-de-telemetria3.ppt

A. CCK6700E

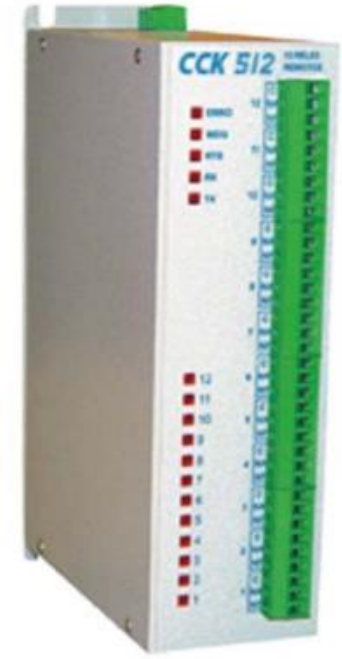
O gerenciador de energia multifuncional CCK6700E foi escolhido para esse projeto por já ser utilizado para medições de consumo de energia dentro da UFG e possuir toda uma infraestrutura pronta. Além do consumo de energia, o CCK6700E possui diversas portas de I/O, o que possibilita aproveitar suas portas que não estejam sendo usadas. Foi utilizado o borne ENT PULSO P2. A Fig. 3 apresenta o equipamento mencionado[12].

Fig. 3. CCK 6700E

Fonte: http://www.cck.com.br/produto.php?nm=CCK6700E

O CCK6700E necessita de outros equipamentos como o CCK512 ou o CCK632, ilustrados na Fig. 4 e na Fig. 5, para realizar a leitura de sinais, sendo o CCK512 responsável por leituras de sinais analógicos e o CCK532 por leitura de sinais digitais. Eles possuem, respectivamente, 12 e 32 portas para leitura.

Fig. 4. CCK512



Fonte: catalogoCCK512-MODBUS.pdf

Fig. 5. CCK632

Fonte: catalogoCCK632-MBUS_N2.pdf

B. CCK4400ME

O transdutor de energia CCK4400ME, mostrado na Fig. 6, é um equipamento utilizado para medição de consumo de energia com duas entradas de pulsos e uma saída RJ45[11].

Em termos mais simples, o CCK4400 é uma versão mais simples e menos robusta em comparação ao CCK6700E e consequentemente mais barato.

O CCK4400ME realiza apenas o monitoramento da rede em que está instalado e já que ele possui entradas de pulsos e é comumente utilizado nas instalações da UFG, será material de estudo para comparação neste projeto.

Fig. 6. CCK4400ME

Fonte: catalogoCCK4400ME.pdf

C. ARDUINO MEGA 2560

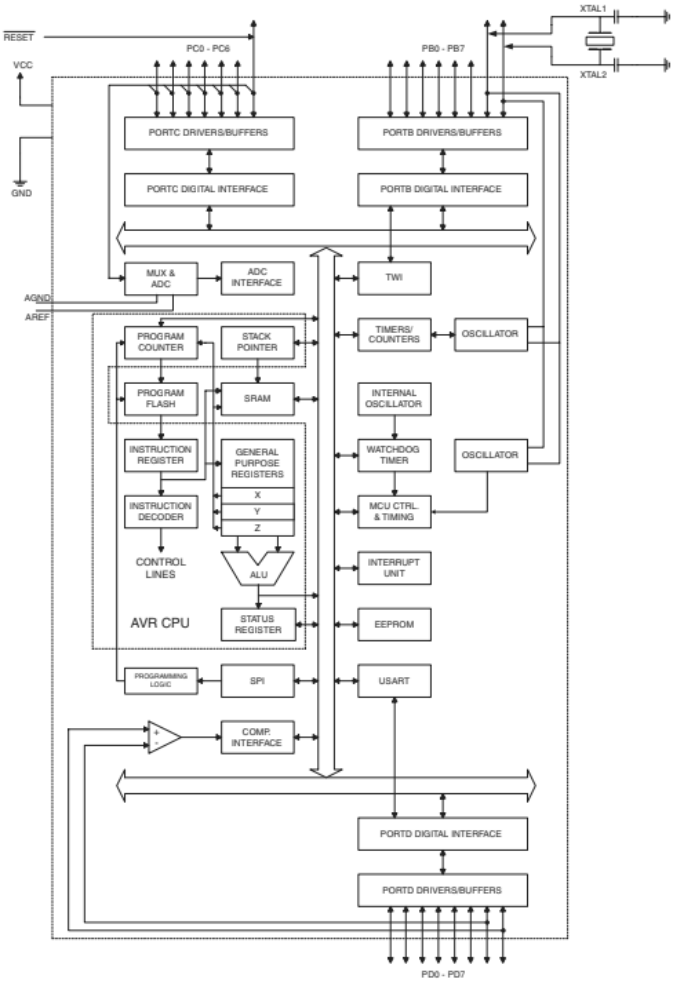
O primeiro Arduino foi lançado na Itália no ano de 2005, sendo um sistema composto por um microcontrolador ATmega8 com 8 Kbytes de memória flash, 1 Kbyte de memória RAM, 512 bytes de EEPROM, 14 pinos digitais, 6 pinos analógicos e 3 saídas PWM (*pulse width modulation*) [10].

O projeto foi evoluindo e novas versões foram sendo lançadas, adicionando novos recursos e melhorando os já existentes. Como pode-se ver na Fig. 7 e na Fig. 8, a complexidade de sua arquitetura, funcionalidades e capacidades foram amplamente expandidas com o passar do tempo.

O Arduino é comumente utilizado em projetos DIY (*Do it yourself*) devido sua modularidade, a grande quantidade de informação que você consegue encontrar na internet e seu preço relativamente baixo se comparado a soluções embarcadas.

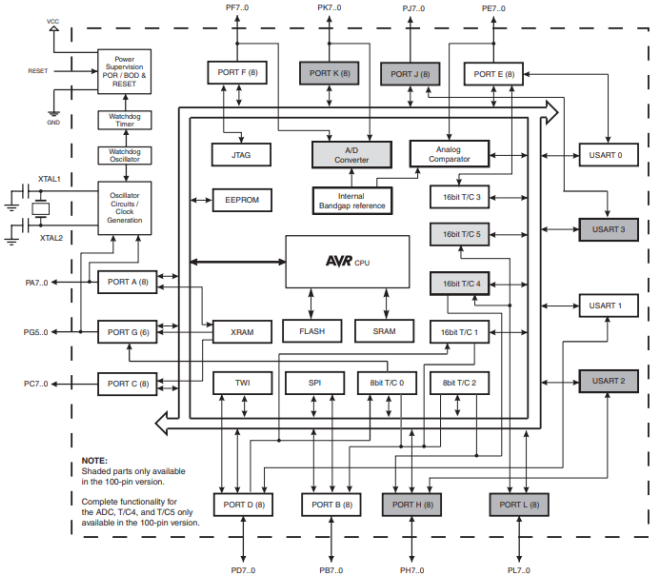
A versatilidade que essa plataforma oferece, permite que o seu usuário modifique sua funcionalidade de acordo com suas necessidades e utilizar uma linguagem de programação mais difundida foi um fator crucial na escolha para este projeto.

Fig. 7. Arquitetura Arduino ATmega8



Fonte: Experimentos com o Arduino, Silveira, 2013, p. 18.

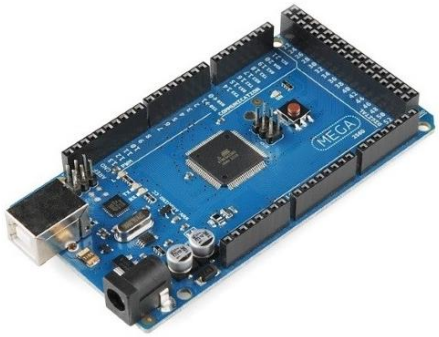
Fig. 8. Arquitetura Arduino ATmega2560



Fonte: https://pdf1.alldatasheet.com/datasheet-pdf/view/174756/ATMEL/ATMEGA1280.html

A alimentação do Arduino Mega 2560, conforme mostrado na Fig. 9, pode ser feita de duas maneiras, através de fonte externa ou através da porta Serial com um USB e pode trabalhar em uma faixa de tensão de 7V a 20V, porém, é recomendado que utilize uma fonte de alimentação entre 7V e 12V, sendo que acima de 12V o controlador de tensão da placa começa esquentar em excesso, necessitando de refrigeração forçada.

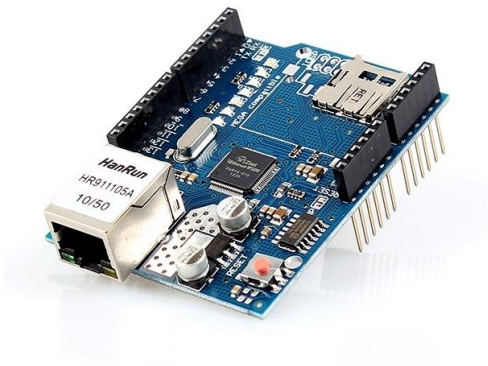
Fig. 9. Arduino mega2560



Fonte: https://uploads.flipeflop.com/2017/07/1AC04-1.jpg

O Arduino se comunicaria com o servidor e guardaria seus os dados através do modulo Ethernet Shield W5100. Esse modulo permite sua conexão à rede (IP) por um conector RJ45 pelos protocolos TCP ou UDP e poderia utilizaria um cartão micro-SD para armazenar os arquivos que vão servir na rede.

Fig. 10. Ethernet Shield W5100



Fonte: https://uploads.filipeflop.com/2017/07/1000701-1.jpg

D.SENSOR FS300A E YF-S201

O sensor FS300A é um sensor de fluxo de água que é composto principalmente pelo seu corpo de plástico, um rotor e um sensor de efeito hall.

Quando a água atravessar o sensor, o rotor girara e através da sua velocidade a parte eletrônica calculara o fluxo e emitira pulsos.

A quantidade de litros é dada pela equação a seguir fornecida pelo datasheet do componente sendo Q a quantidade de pulsos e L a quantidade de litros por minuto.

$$L = Q/5{,}5 \quad (1)$$

O sensor FS300A é instalado na entrada de água na qual se quer monitorar o fluxo.

Fig. 11. Sensor FS300A

Fonte:https://www.openhacks.com/uploadsproductos/g1__water_flow_sensor_-_wiki.pdf

O outro sensor utilizado foi o YF-S201 que se baseia no mesmo efeito do FS300A para realizar a medição de fluxo, o efeito hall.

Fig. 12. Sensor YF-S201.

Fonte: https://produto.mercadolivre.com.br/MLB-725932987-sensor-de-fluxovazo-agua-12-yf-s201-arduino-efeito-hall-_JM?quantity=1

A quantidade de litros é dada pela equação a seguir fornecida pelo datasheet do componente sendo Q a quantidade de pulsos e L a quantidade de litros por minuto.

$$L = Q/7 \quad (2)$$

Foram realizados testes para se aferir se as equações dos sensores correspondem aos valores reais e foi constatado que havia uma discrepância grande com os valores lidos pelo sensor e medido, algo em torno de 15%.

Com isso em mente foram realizadas 10 medidas para construção de uma nova equação com mais precisão do que a fornecida pelo fabricante.

O método utilizado consiste no Arduino contando o tempo e a quantidade de pulsos total que houve durante a medição. A Fig. 13 a seguir mostra o como foi feita a medição e as tabelas I e II os valores obtidos.

Fig. 13. Teste e calibração dos sensores

Fonte: Autores

TABELA I
RESULTADO DAS MEDIÇÕES YF-S201 E FS300A

LITROS	TEMPO (s)	PULSOS YF-S201	PULSOS FS300A
2,5	19	947	677
2,8	26	1022	723
2,7	47	978	667
2,4	21	908	645
2,35	21	873	623
2,45	47	869	594
4,4	44	1578	1136
7,6	55	2647	1973
9,7	77	3429	2509
17,6	146	5890	4554

Fonte: Autores

TABELA II
RESULTADO DAS MEDIÇÕES YF-S201 E FS300A

Litros/s	YF-S201 Hz	FS300A Hz
0,1316	49,8421	35,6316
0,1077	39,3077	27,8077
0,0574	20,8085	14,1915
0,1143	43,2381	30,7143
0,1119	41,5714	29,6667
0,0521	18,4894	12,6383
0,1000	35,8636	25,8182
0,1382	48,1273	35,8727
0,1260	44,5325	32,5844
0,1205	40,3425	31,1918

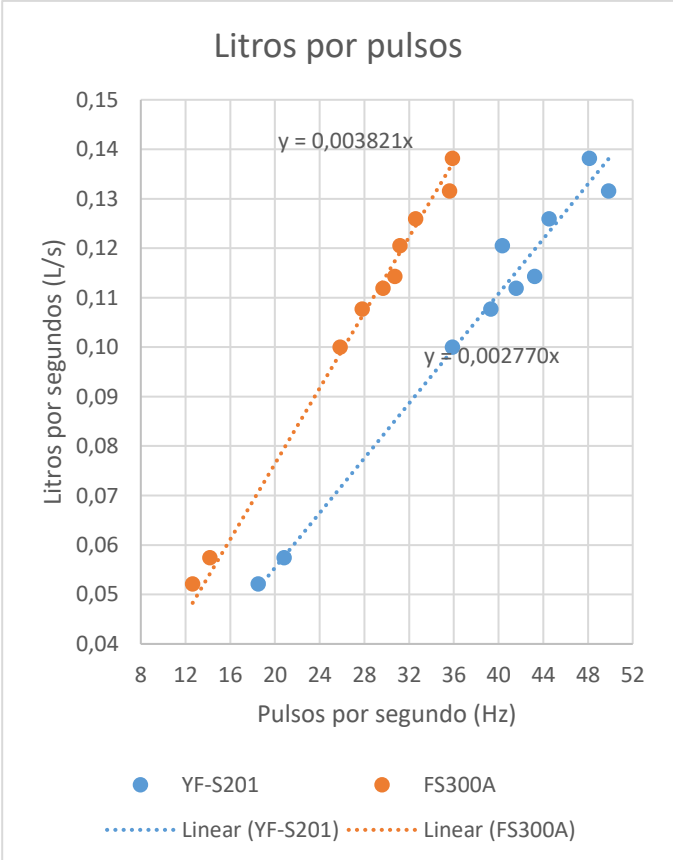
Fonte: Autores

Com esses valores foram traçados uma nova reta para os sensores YF-S201 e FS300A. Essas novas retas que podem ser conferidas na Fig. 14 a seguir apresentam as seguintes equações sendo L litros por segundo e Q a quantidade de pulsos:

$$\text{YF-S201: } L = 0{,}0027770 * Q \quad (3)$$

$$\text{FS300A: } L = 0{,}003821 * Q \quad (4)$$

Fig. 14. Gráfico litros por pulsos dos sensores YF-S201 e FS300A



Fonte: Autores

Por apresentar menor dispersão, o sensor FS300A será o sensor utilizado para os experimentos. Utilizando a nova reta, as medidas digitais e o valor medido apresentaram discrepâncias de aproximada de 1,5% como pode ser observado na tabela III a seguir.

TABELA III
RESULTADO DAS MEDIÇÕES DO FS300A

Valor medido		Erro %
Digital	Analógico	
2,632	2,6	1,230769
4,875	4,95	1,515152
7,658	7,8	1,820513
10,505	10,3	1,990291
12,873	12,65	1,762846
14,931	15,15	1,445545
17,439	17,65	1,195467
21,325	21,55	1,044084
	Média:	**1,500583**

Fonte: Autores

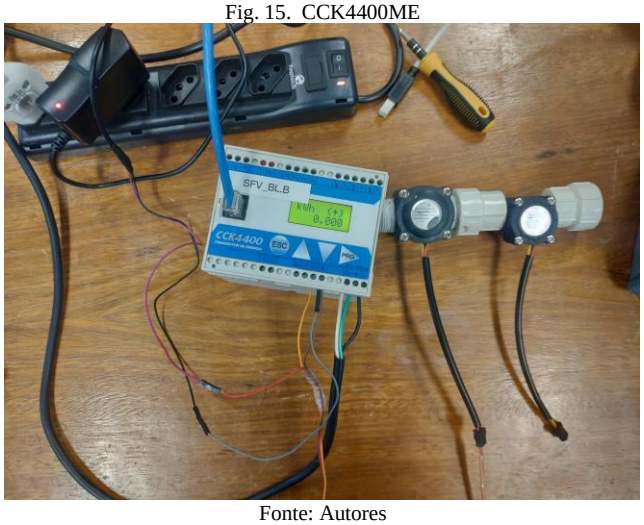
III. PROTOTIPO CONSTRUIDO

A seguir serão apresentados os detalhes do protótipo construído e sua integração com o sistema de monitoramento existente.

A. CCK4400ME

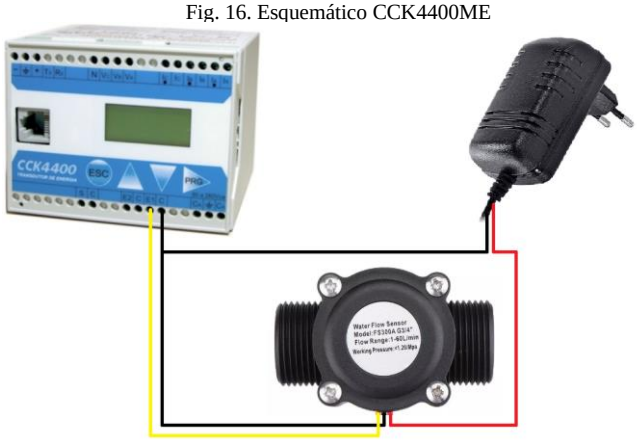
O protótipo construído utilizando o CCK4400ME pode ser observado nas Fig. 15 e Fig. 16 a seguir. A montagem com o CCK4400ME necessita de alimentação externa para o sensor, pois sua saída de 4V não é o suficiente para o funcionamento do mesmo. A montagem consiste em ligar o fio de sinal (Amarelo) em E1 com a referência (Preto) conectado no C do CCK e no sensor. Para os testes com esse equipamento o sensor foi alimentado com uma fonte de 9 Volts.

Fig. 15. CCK4400ME



Fonte: Autores

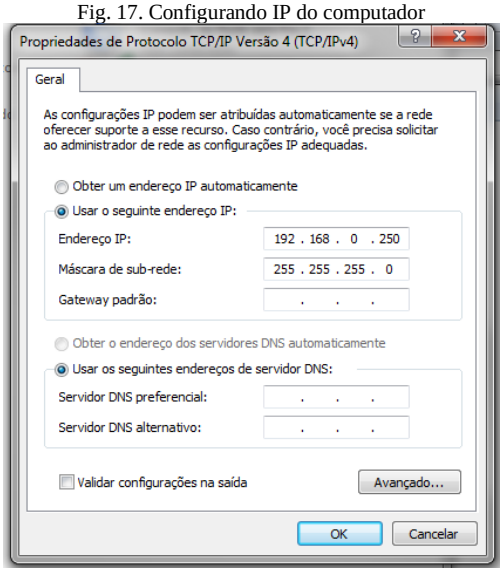
Fig. 16. Esquemático CCK4400ME



Fonte: Autores

Após a montagem do protótipo, foi necessário realizar diversas configurações para que ele funcione corretamente. A primeira coisa a se fazer para que o equipamento comunique com o computador é definir um IP fixo para comunicação via RJ45.

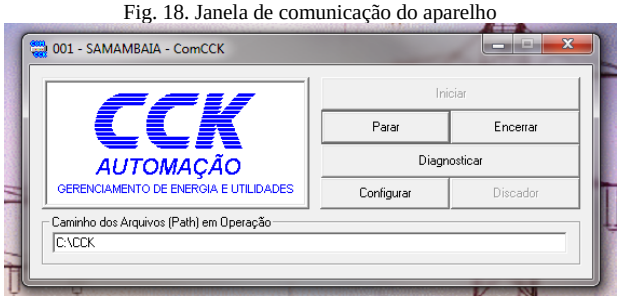
Fig. 17. Configurando IP do computador



Fonte: Autores

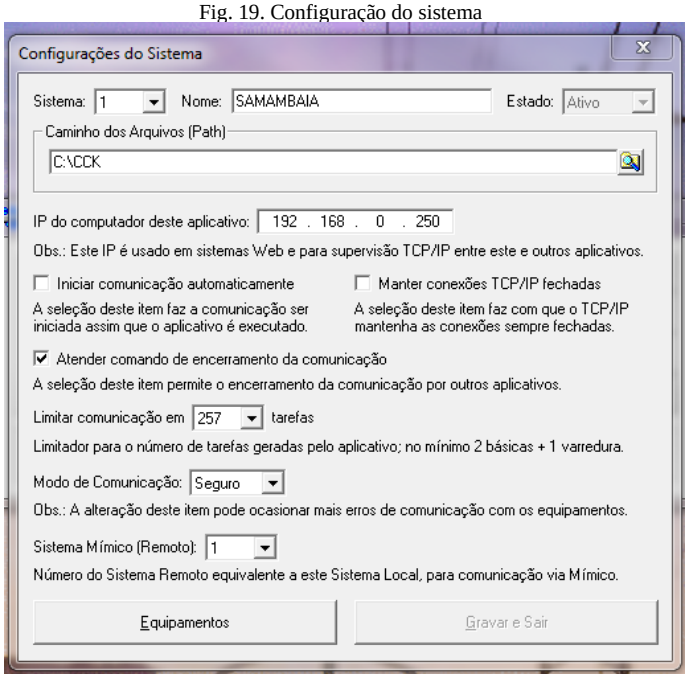
Após definir o IP, deve-se inicializar o programa CCK gerente, ir na aba Supervisão e em seguida em comunicação. Abrirá uma janela para a comunicação com o aparelho no qual deverá entrar com um usuário e senha e, em seguida, realizar a configuração colocando o mesmo IP utilizado no computador.

Fig. 18. Janela de comunicação do aparelho



Fonte: Autores

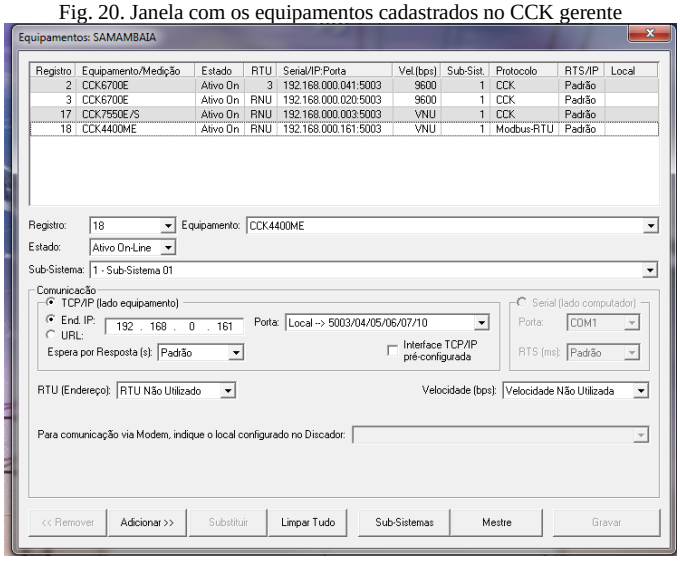
Fig. 19. Configuração do sistema



Fonte: Autores

Em seguida, deve-se adicionar o aparelho em questão selecionando um número de registro, o equipamento utilizado, o estado do equipamento, o subsistema a que ele pertence e atribuindo um endereço de rede.

Fig. 20. Janela com os equipamentos cadastrados no CCK gerente

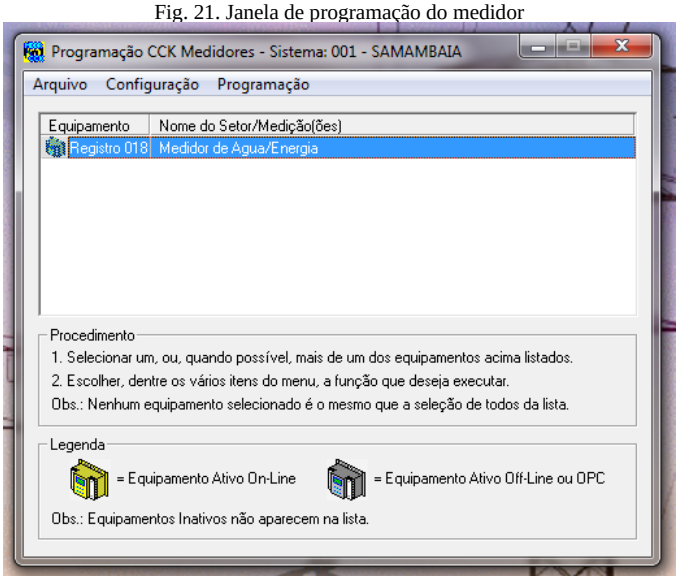


Fonte: Autores

Confirmando que há conexão entre o computador e o CCK4400ME, retorna-se ao CCK gerente e agora segue-se para a aba programação. Em programação selecionou-se a opção de CCK medidores.

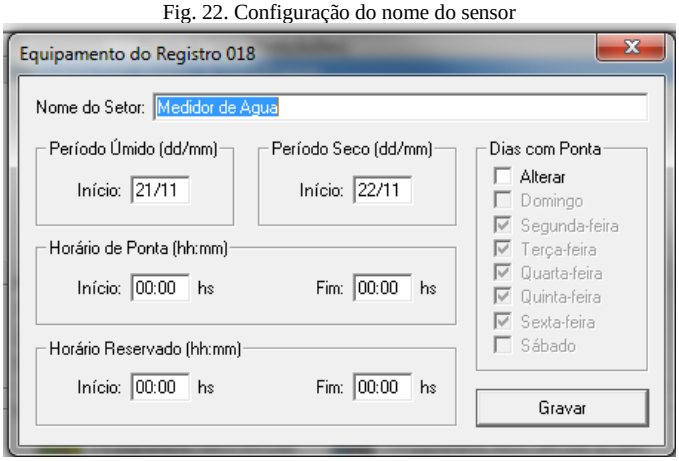
Após entrar com login e senha novamente, tem-se uma janela exibindo o aparelho e opções de configuração como nome do medidor, tempo de envio entre medições, e o que está sendo contado nas entradas de pulsos, assim como a razão entre as grandezas para realizar as medições.

Fig. 21. Janela de programação do medidor



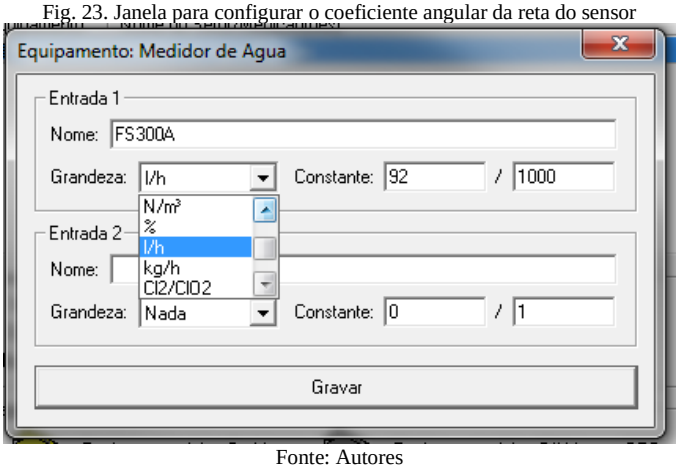
Fonte: Autores

Fig. 22. Configuração do nome do sensor



Fonte: Autores

Fig. 23. Janela para configurar o coeficiente angular da reta do sensor

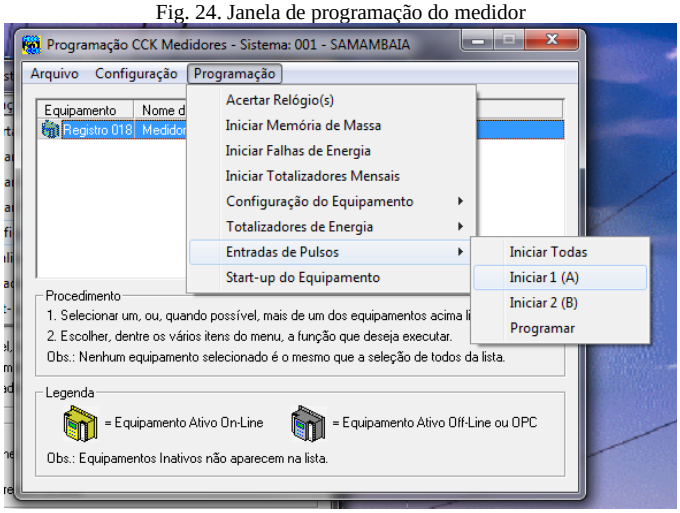


Fonte: Autores

Depois de realizadas as devidas configurações, é necessário enviá-las para o CCK e fazer uma leitura do equipamento para confirmar se todas as configurações foram aplicadas corretamente. Confirmando que todas as configurações foram aplicadas corretamente, pode-se iniciar a contagem de pulsos nas portas do CCK.

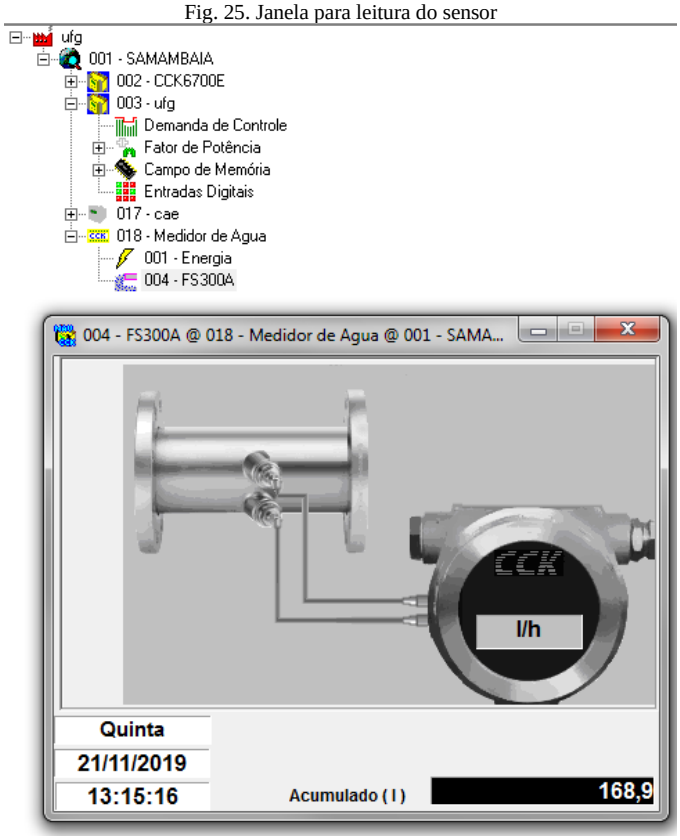
Após o início da contagem é possível conferir a leitura do consumo em tempo real.

Fig. 24. Janela de programação do medidor



Fonte: Autores

Fig. 25. Janela para leitura do sensor



Fonte: Autores

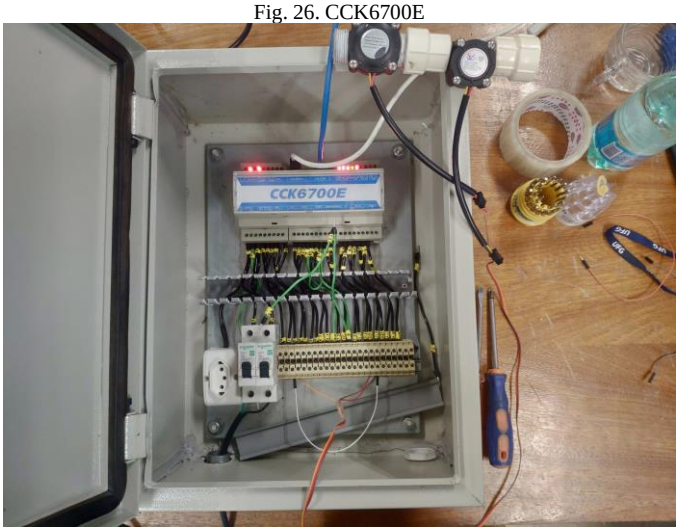
Para os testes o CCK4400ME foi configurado para enviar os dados coletados ao CCK gerente a cada 15 minutos. Por não possuir memória de massa dedicada a contagem de pulsos, se houver queda de energia durante esse tempo os dados colhidos pelo aparelho serão perdidos.

B. CCK6700E

O protótipo construído utilizando o CCK6700E pode ser observado na Fig. 26 e na Fig. 27. A montagem com o CCK6700E não necessita de alimentação externa para o sensor, pois sua saída de 5V é o suficiente para o seu funcionamento.

A montagem consiste em ligar o fio de sinal (Amarelo) em P2 com a referência (Preto) conectado no T do CCK e no sensor.

Fig. 26. CCK6700E



Fonte: Autores

Fig. 27. Esquemático CCK6700E

Fonte: Autores

Após a montagem deste protótipo, foi realizado o mesmo passo a passo do CCK4400ME para a devida configuração do aparelho. Depois de aplicadas as configurações, pode-se observar que o CCK6700E apresenta uma janela do consumo de água diferente do CCK4400ME, porem com a mesma funcionalidade.

Fig. 28. Janela para leitura do sensor

Fonte: Autores

C. ARDUINO MEGA2560

No cenário com Arduino Mega2560, foi montado um protótipo acrescido com o Sensor FS300A e o módulo Ethernet Shield W5100, que permite a comunicação com a rede através da porta Ethernet RJ45, além de um servidor em back-end para processar e armazenar os dados enviados pelo Arduino em um banco de dados MySql.

Nas Fig. 29 e Fig. 30 pode ser observado o protótipo construído utilizando o Arduino Mega2560 juntamente ao Ethernet Shield W5100 e o Sensor FS-300A.

Fig. 29. Protótipo Arduino Mega2560

Fonte: Autores

Fig. 30. Protótipo Arduino Mega2560

Fonte: Autores

O Ethernet Shield W5100 quando conectado à rede, fornece um endereço de IP compatível com os protocolos TCP e UPD. No protótipo construído foi vinculado um IP Address manualmente para o Ethernet Shield através da programação do Arduino afim de tornar o mesmo estático na rede para possíveis acessos futuros como pode ser observado na Fig. 31 a seguir.

Fig. 31. Código para definir um IP para o Arduino

```
/***************************************
 * Endereço do servidor arduino na rede *
 ***************************************/
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,25);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255, 255, 255, 0);

EthernetClient client;
```

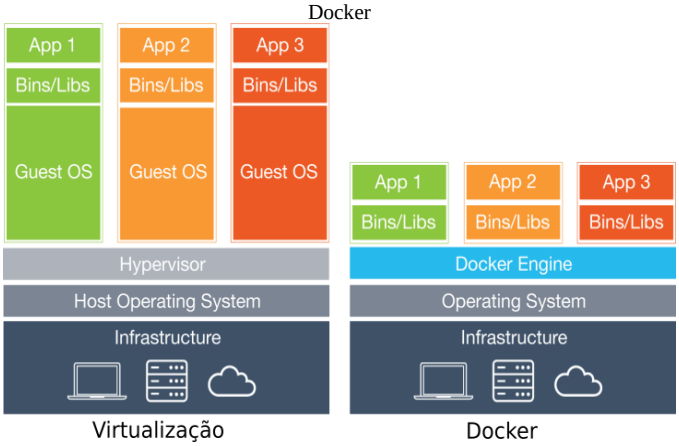
Fonte: Autores

Para efetuar a persistência dos dados lidos pelo sensor, assim como permitir futuras consultas, foi desenvolvido um servidor utilizando o conceito moderno de micro serviços, que se baseia em separar cada serviço em um ambiente próprio exclusivo, onde cada serviço funciona de maneira independente, porém pode se comunicar com outros serviços.

O objetivo de utilizar o conceito de micros serviços é o de obter maior estabilidade para todo o sistema, possibilidade de utilização de várias linguagens de programação, uma vez que cada serviço é responsável apenas por si próprio, além de maior facilidade em manutenções futuras.

Para utilizar os micro serviços foi utilizada a ferramenta chamada Docker, que permite que em uma máquina ou host sejam criados vários ambientes isolados, esses ambientes são chamados popularmente de “Containers”, os mesmos funcionam de maneira semelhante à uma máquina virtual, porém possui apenas o necessário para que o serviço vinculado a este ambiente possa funcionar, não necessitando de arquivos de um sistema operacional completo em cada ambiente.

Fig. 32. Comparação entre ambientes criados com máquinas virtuais e com Docker



Fonte: https://docker-unleashed.readthedocs.io/aula1.html

No protótipo criado, além do servidor do Arduino que já funciona de maneira independente através do módulo Ethernet Shield W5100, foram criados também 3 containers para realização dos demais serviços necessários para o sistema funcionar.

O primeiro container é responsável pela execução do PHP, linguagem de programação que foi utilizada no desenvolvimento do servidor, o qual funciona como uma API (Application Programming Interface), tanto para persistência, tanto para consulta dos dados medidos pelo sensor.

O segundo container mantém o MySql, banco de dados utilizado para armazenamento dos dados, seu papel é apenas armazenar informações que são cadastradas, no nosso caso, através do PHP [4].

O terceiro container mantém um servidor em Node JS, outra linguagem de programação, desta vez para oferecer o ambiente que sustenta a interface visual que pode ser acessada via navegador, a qual exibe para o usuário os dados armazenados, através de gráficos separados por dia ou mês, proporcionando uma fácil leitura.

Todo o código desenvolvido para funcionamento do protótipo, começando pelo que é executado pelo Arduino foi escrito em C++ [3].

Assim como é feito com o CCK4400 e o CCK6700, a contagem dos pulsos enviados pelo sensor e a contagem do tempo, foram realizadas utilizando-se as interrupções através do pino digital do Arduino, o qual está ligado o sensor e a interrupção por tempo.

Com a quantidade de pulsos por segundo é possível fazer a conversão para litros por segundo através da equação (4), utilizando o coeficiente angular da reta do sensor, obtendo assim a vazão do tempo em específico que é acrescentada a cada segundo afim de obter o montante de litros passados pelo sensor, o mesmo é salvo no cartão de memória que é conectado ao Ethernet Shield W5100 a cada minuto, quando a contagem da quantidade de litros é zerada com intuito de desocupar a memória do Arduino.

Fig. 33. Armazenamento de dados no cartão SD

```
void recordData() {
  boolean firstLine = true;

  if (SD.exists(fileName)) {
    firstLine = false;
  }

  dataFile = SD.open(fileName, FILE_WRITE);
  if ( ! dataFile) {
    Serial.println("ERRO AO ACESSAR O SD...");
    return;
  }

  if ( ! firstLine) {
    dataFile.println(",");
  }

  dataFile.print("{\"minute\":\"");
  dataFile.print(minutes);
  dataFile.print("\",\"liters\":\"");
  dataFile.print(liters);
  dataFile.print("\"}");
  dataFile.close();

  liters = 0;
}
```

Fonte: Autores

Fig. 34. Calculo do tempo e vazão

```
void calculateData () {
  seconds++;
  if(seconds == 60) {
    minutes++;
    seconds = 0;
  }
  if(pulses > 0) {
    flowRate = pulses * coefficient;
    liters = liters + flowRate;
  }
  pulses = 0;
}
```

Fonte: Autores

Os dados são salvos em um arquivo com o formato **TXT** no cartão SD, porém escritos em formato **JSON** a fim de facilitar o envio para o servidor futuramente. Com os dados já salvos no SD, a cada 5 minutos é feita uma tentativa de envio dos dados para o servidor, utilizando o protocolo HTTP com o método POST, caso o envio seja concluído com sucesso, os dados do SD são apagados, caso contrário, continuaram sendo incrementados no cartão SD e haverá uma nova tentativa de envio a cada minuto.

Fig. 35. Lógica para armazenamento e envio de dados

```
void loop() {
    if (seconds == 0 && minutes > 0) {

      recordData();

      if (
        minutes >= sendPeriod
        && sendData(getJsonData())
      )
      {
        removeData();
        minutes = 0;
      }

      delay(1000);
    }
}
```

Fonte: Autores

Fig. 36. Envio de dados para o servidor

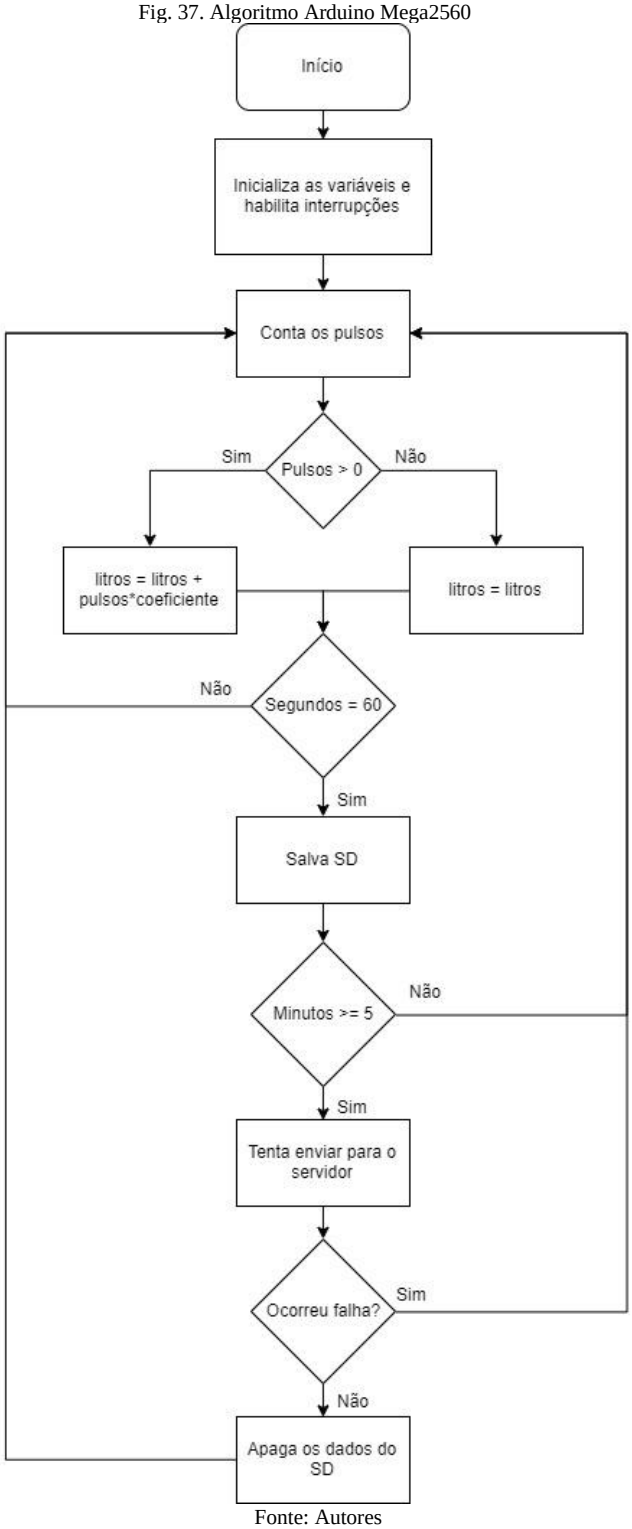
```
boolean sendData (String data) {
  if( ! client.connect(serverIp, serverPort)) {
    return false;
  }
  client.connect(serverIp, serverPort);
  client.println("POST /sensor_data HTTP/1.1");
  client.print("Host: ");
  client.print(serverIp);
  client.print(":");
  client.println(serverPort);
  client.println("Content-Type: application/json");
  client.print("Content-Length: ");
  client.println(data.length());
  client.println("Connection: keep-alive");
  client.println();
  client.println(data);

  return true;
}
```

Fonte: Autores

A algoritmo utilizado é detalhado na Fig. 37.

Fig. 37. Algoritmo Arduino Mega2560



Fonte: Autores

Uma vez que os dados são enviados pelo Arduino, os mesmos são agora tratados no servidor, construído em PHP, utilizando um microframework chamado “Lumen”, que é derivado de outro framework chamado “Laravel”, porém o Lumen é apropriado para construção de API’s dispensando todo o código desnecessário de sistemas totalmente web [6].

O Lumen foi utilizado com intuito de facilitar as requisições tanto de envio quanto de consulta de dados, assim como as relações com o banco de dados, o mesmo já possui recursos que possibilitam de maneira simples e rápida, endereçar end-points a controladores específicos, manipular toda relação com o banco de dados, dês da criação das tabelas, até a persistência e consulta dos dados, além da possibilidade de implementar um sistema de autenticação facilmente no futuro.

Como o Arduino não conhece o horário em que está trabalhando, nele são contados apenas a quantidade de minutos antes do envio para o servidor, e o servidor por sua vez tem a responsabilidade de fazer o processo reverso, para poder vincular cada dado recebido ao horário real em que o mesmo foi gravado, para assim salvar corretamente no banco de dados.

Fig. 38. Persistindo os dados recebidos do Arduino

```php
public function add(Request $request): Response
{
    if ($request->input( key: "initializing")) {
        $sensorData = new SensorData();
        $sensorData->date_time = date( format: 'Y-m-d H:i:s');
        $sensorData->liters = 0;
        $sensorData->average_flow = 0;
        $sensorData->save();
    } elseif ($request->input( key: 'data')) {
        foreach ($request->input( key: 'data') as $data) {
            $sensorData = new SensorData();
            $sensorData->date_time = date(
                format: 'Y-m-d H:i:s',
                strtotime(
                    time: "-" . (
                        count($request->input( key: 'data')) - $data['minute']
                    ) . " minutes"
                )
            );
            $sensorData->liters = $data['liters'];
            $sensorData->average_flow = $data['liters'] / 60;
            $sensorData->save();
        }
    }
    return new Response( content: "Ok",  status: 200);
}
```

Fonte: Autores

Ainda no servidor foi criado um end-point da API para consulta de todos os dados cadastrados no banco de dados para que possa ser consumido por aplicativos ou serviços externos de terceiros.

Nesse end-point é possível passar por parâmetro datas de início e ou fim, para que o resultado seja limitado e os dados retornados nele são puramente análogos ao seu estado no banco de dados, sem sofrer qualquer tipo de alteração ou manipulação.

Por fim, foi desenvolvida uma interface gráfica para consulta dos dados por meio da web, utilizando um framework baseado na linguagem de programação Javascript, chamado “Vue Js”, que permitiu de maneira simples, consumir a API criada anteriormente, consultar os dados salvos no banco de dados e exibi-los em formato de gráficos, facilitando seu entendimento[5].

Foram disponibilizados na interface 2 gráficos. Um dos gráficos informa a somatória do consumo hora a hora, para cada dia do mês, assim como o total consumido referente a cada dia. Outro gráfico informa a somatória geral de consumo em cada mês do ano, assim também como o total consumido no ano.

Fig. 39. Consumindo a API via Vue Js

```
mounted() {
  axios.get('http://192.168.1.44:8000/sensor_data/month/actual')
    .then(response => {
      this.labelCharts = response.data.label;
      this.dayleChart.allData = response.data.data;
      this.dayleChart.hours = response.data.hours;
      this.dayleChart.days = response.data.days;
      this.dayleChart.activeIndex = response.data.active;
      this.dayleChart.totals = response.data.totals;
      this.initDayleChart(this.dayleChart.activeIndex);
    });
  axios.get('http://192.168.1.44:8000/sensor_data/year/actual')
    .then(response => {
      this.annualChart.allData = response.data.data;
      this.annualChart.months = response.data.months;
      this.annualChart.total = response.data.total;
      this.initAnnualChart();
    });
}
```

Fonte: Autores

IV. RESULTADOS

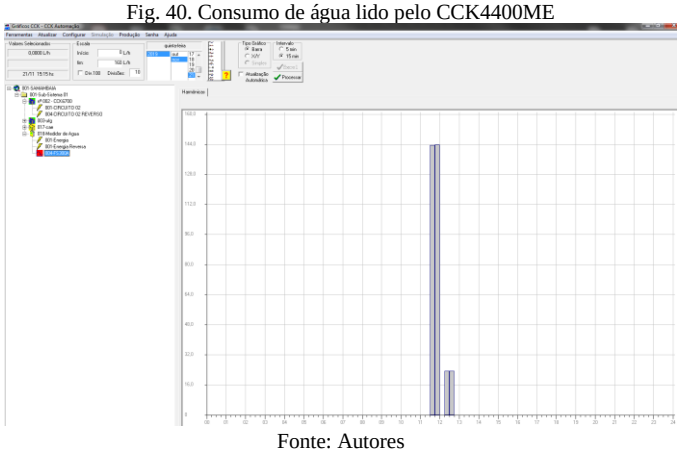
Serão apresentados os resultados das medições realizadas utilizando-se o CCK4400ME, o CCK6700E e o ARDUINO MEGA2560.

A. CCK4400ME

O CCK4400ME mediu corretamente os pulsos e realizou o envio de seus dados com sucesso para o programa CCK gerente, porem como ele não apresenta memória de massa para armazenamento da contagem de pulsos, se houver uma perca de energia entre os envios, estes dados seriam perdidos.

Como pode-se observar na Fig. 40 a seguir, os valores registrados pelo CCK4400ME da leitura do sensor em seu banco de dados.

Fig. 40. Consumo de água lido pelo CCK4400ME



Fonte: Autores

B. CCK6700E

Apesar de estar configurado da mesma maneira que o CCK4400ME, o CCK6700E não conseguiu contar os pulsos, sendo que, as vezes iniciava com valores aleatórios em sua memória, mesmo quando era zerado, contando de maneira decrescente ou crescente. Isto mostra que mesmo com porta para contar pulsos, o equipamento não é capaz de realizar contagem sem outros CCK's como o CCK512 e CCK632 mencionados anteriormente. Mesmo não contando corretamente, o CCK6700E possui memória de massa para a contagem de pulsos, mostrando os mesmos valores presentes em sua contagem antes de interromper a sua alimentação.

C. ARDUINO MEGA2560

O Arduino Mega2560 executou a leitura dos dados corretamente, permitindo enorme liberdade para o tratamento dos mesmos, bem como fazer o armazenamento sem apresentar inconsistências, assim como seu envio para o servidor e sucessivamente para o banco de dados.

Como os dados são sempre armazenados primeiramente no cartão SD, caso haja qualquer tipo de falha na comunicação com o servidor os dados não serão perdidos, haverá sempre uma nova tentativa de enviá-los, onde somente serão apagados após o êxito no envio.

Uma vez que os dados foram recebidos pelo servidor, todos dados referentes ao intervalo entre as requisições foram salvos no banco de dados de forma ordenada com êxito, permitindo facilmente exportação para utilização em outros cenários.

Fig. 41. Dados armazenados no Banco de Dados

Servidor: arduino.mysql | Banco de dados: arduino | Tabela: sensor_data | Dados | Consulta*

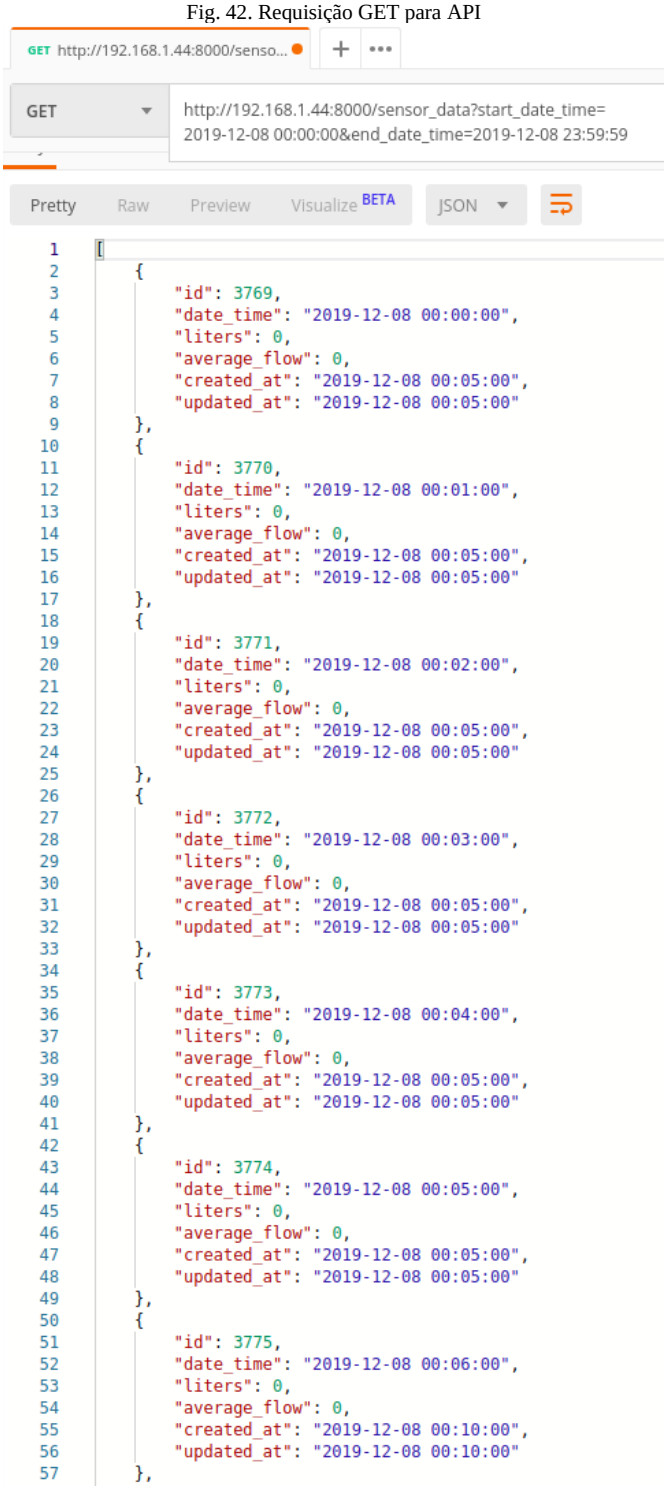
arduino.sensor_data: 2.160 registros totais (aproximadamente), 720 registros que combinam com o filtro

id	date_time	liters	average_flow	created_at	updated_at
3.769	2019-12-08 00:00:00	0,0000	0,00000000	2019-12-08 00:05:00	2019-12-08 00:05:00
3.770	2019-12-08 00:01:00	0,0000	0,00000000	2019-12-08 00:05:00	2019-12-08 00:05:00
3.771	2019-12-08 00:02:00	0,0000	0,00000000	2019-12-08 00:05:00	2019-12-08 00:05:00
3.772	2019-12-08 00:03:00	0,0000	0,00000000	2019-12-08 00:05:00	2019-12-08 00:05:00
3.773	2019-12-08 00:04:00	0,0000	0,00000000	2019-12-08 00:05:00	2019-12-08 00:05:00
3.774	2019-12-08 00:05:00	0,0000	0,00000000	2019-12-08 00:05:00	2019-12-08 00:05:00
3.775	2019-12-08 00:06:00	0,0000	0,00000000	2019-12-08 00:10:00	2019-12-08 00:10:00
3.776	2019-12-08 00:07:00	0,0000	0,00000000	2019-12-08 00:10:00	2019-12-08 00:10:00
3.777	2019-12-08 00:08:00	0,0000	0,00000000	2019-12-08 00:10:00	2019-12-08 00:10:00
3.778	2019-12-08 00:09:00	0,0000	0,00000000	2019-12-08 00:10:00	2019-12-08 00:10:00
3.779	2019-12-08 00:10:00	0,0000	0,00000000	2019-12-08 00:10:00	2019-12-08 00:10:00
3.780	2019-12-08 00:11:00	0,0000	0,00000000	2019-12-08 00:15:00	2019-12-08 00:15:00
3.781	2019-12-08 00:12:00	0,0000	0,00000000	2019-12-08 00:15:00	2019-12-08 00:15:00
3.782	2019-12-08 00:13:00	0,0000	0,00000000	2019-12-08 00:15:00	2019-12-08 00:15:00
3.783	2019-12-08 00:14:00	0,0000	0,00000000	2019-12-08 00:15:00	2019-12-08 00:15:00
3.784	2019-12-08 00:15:00	0,0000	0,00000000	2019-12-08 00:15:00	2019-12-08 00:15:00
3.785	2019-12-08 00:16:00	0,0000	0,00000000	2019-12-08 00:20:00	2019-12-08 00:20:00
3.786	2019-12-08 00:17:00	0,0000	0,00000000	2019-12-08 00:20:00	2019-12-08 00:20:00
3.787	2019-12-08 00:18:00	0,0000	0,00000000	2019-12-08 00:20:00	2019-12-08 00:20:00
3.788	2019-12-08 00:19:00	0,0000	0,00000000	2019-12-08 00:20:00	2019-12-08 00:20:00
3.789	2019-12-08 00:20:00	0,0000	0,00000000	2019-12-08 00:20:00	2019-12-08 00:20:00
3.790	2019-12-08 00:21:00	0,0000	0,00000000	2019-12-08 00:25:00	2019-12-08 00:25:00
3.791	2019-12-08 00:22:00	0,0000	0,00000000	2019-12-08 00:25:00	2019-12-08 00:25:00
3.792	2019-12-08 00:23:00	0,0000	0,00000000	2019-12-08 00:25:00	2019-12-08 00:25:00
3.793	2019-12-08 00:24:00	0,0000	0,00000000	2019-12-08 00:25:00	2019-12-08 00:25:00
3.794	2019-12-08 00:25:00	0,0000	0,00000000	2019-12-08 00:25:00	2019-12-08 00:25:00
3.795	2019-12-08 00:26:00	0,0000	0,00000000	2019-12-08 00:30:00	2019-12-08 00:30:00
3.796	2019-12-08 00:27:00	0,0000	0,00000000	2019-12-08 00:30:00	2019-12-08 00:30:00
3.797	2019-12-08 00:28:00	0,0000	0,00000000	2019-12-08 00:30:00	2019-12-08 00:30:00
3.798	2019-12-08 00:29:00	0,0000	0,00000000	2019-12-08 00:30:00	2019-12-08 00:30:00
3.799	2019-12-08 00:30:00	0,0000	0,00000000	2019-12-08 00:30:00	2019-12-08 00:30:00
3.800	2019-12-08 00:31:00	0,0000	0,00000000	2019-12-08 00:35:00	2019-12-08 00:35:00
3.801	2019-12-08 00:32:00	0,0000	0,00000000	2019-12-08 00:35:00	2019-12-08 00:35:00
3.802	2019-12-08 00:33:00	0,0000	0,00000000	2019-12-08 00:35:00	2019-12-08 00:35:00
3.803	2019-12-08 00:34:00	0,0000	0,00000000	2019-12-08 00:35:00	2019-12-08 00:35:00
3.804	2019-12-08 00:35:00	0,0000	0,00000000	2019-12-08 00:35:00	2019-12-08 00:35:00
3.805	2019-12-08 00:36:00	0,0000	0,00000000	2019-12-08 00:40:00	2019-12-08 00:40:00
3.806	2019-12-08 00:37:00	0,0000	0,00000000	2019-12-08 00:40:00	2019-12-08 00:40:00
3.807	2019-12-08 00:38:00	0,0000	0,00000000	2019-12-08 00:40:00	2019-12-08 00:40:00
3.808	2019-12-08 00:39:00	0,0000	0,00000000	2019-12-08 00:40:00	2019-12-08 00:40:00

Fonte: Autores

A API criada permite facilmente a consulta dos dados através de uma requisição utilizando o método GET do servidor HTTP retornando corretamente os dados para serem utilizados por qualquer outro aplicativo, ou até mesmo persistência em outro servidor.

Fig. 42. Requisição GET para API



Fonte: Autores

Por fim, a interface gráfica satisfaz o esperado, fornecendo de maneira agrupada e organizada os dados salvos, para possível entendimento do usuário final. Por se tratar de uma interface web, a mesma pode ser acessada sem dificuldade por quaisquer dispositivos.

O servidor foi hospedado em uma máquina local, porém foi exposto através de um DNS, permitindo assim acesso de qualquer equipamento com acesso à internet através do endereço https://hidrometro-arduino.ddns.net.

Fig. 43. Interface gráfica Web para consulta dos dados

Fonte: Autores

V. CONCLUSÃO

Com todos os experimentos e testes realizados, pode-se observar que o erro na leitura apresentado pelo sensor se mostra aceitável para pequenos consumidores. Havia outros tipos de medidores de vazão que seriam mais recomendados como os magnéticos, ultrassônicos e deprimogênios como o Tubo de Venturi [1].

Uma dificuldade enfrentada neste projeto foi a documentação precária fornecida pela CCK sobre seus produtos, dificultando o manuseio para a instalação e configuração. Apesar de ter sido possível configurar e realizar com sucesso medições no CCK4400ME, não se obteve o mesmo sucesso com o CCK6700E, que apesar de possuir uma entrada para contagem de pulsos, não realizou uma medição correta, mostrando ser necessário equipamentos auxiliares para realizar estas medições.

Os custos das soluções que foram discutidas são apresentados na Tabela 4

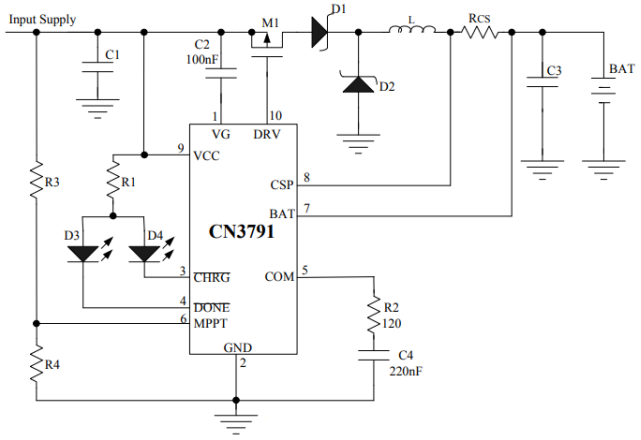
TABELA IV
SENSOR FS300A NO CCK4400ME

Componente	Custo(R$)
Arduino Mega2560	R$ 56,00
Ethernet Shield W5100	R$ 34,68
Sensor FS300A	R$ 59,31
Cartão micro-SD 16 GB	R$ 15,01
Fonte DC 9V	R$ 14,99
Total Arduino Mega2560:	**R$ 179,99**
CCK4400ME	R$ 1665,00
Fonte DC 9V	R$ 14,99
Sensor FS300A	R$ 59,31
Total CCK4400ME:	**R$ 1739,30**
CCK6700E	R$ 1,00
CCK632	R$ 1,00
Sensor FS300A	R$ 59,31
Total CCK6700E	**R$ 61,31**

Fonte: Autores

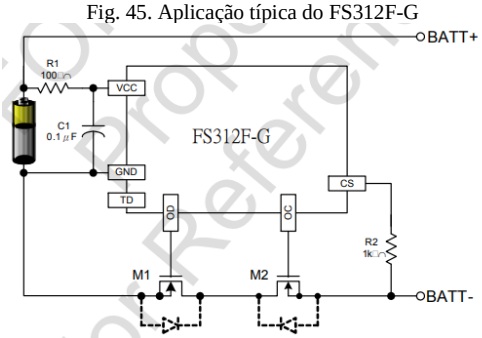
Um problema enfrentado pelo CCK4400ME é que ele não realiza contagem do consumo de água sem energia, algo que pode ser contornado no Arduino com o circuito proposto na Fig. 47, utilizando as aplicações típicas dos CI e fazendo as devidas adaptações.

Fig. 44. Aplicação típica do CN3791



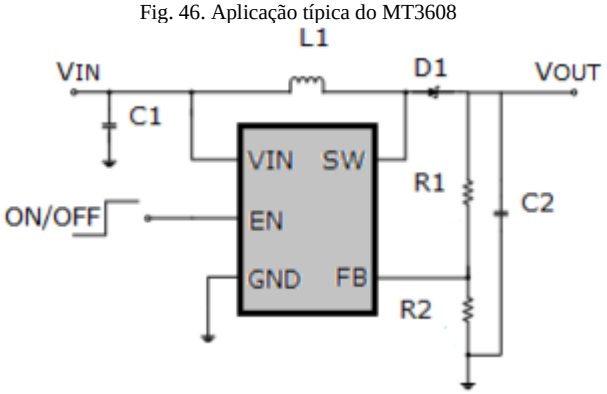
Fonte: http://www.consonance-elec.com/pdf/datasheet/DSE-CN3791.pdf

Fig. 45. Aplicação típica do FS312F-G



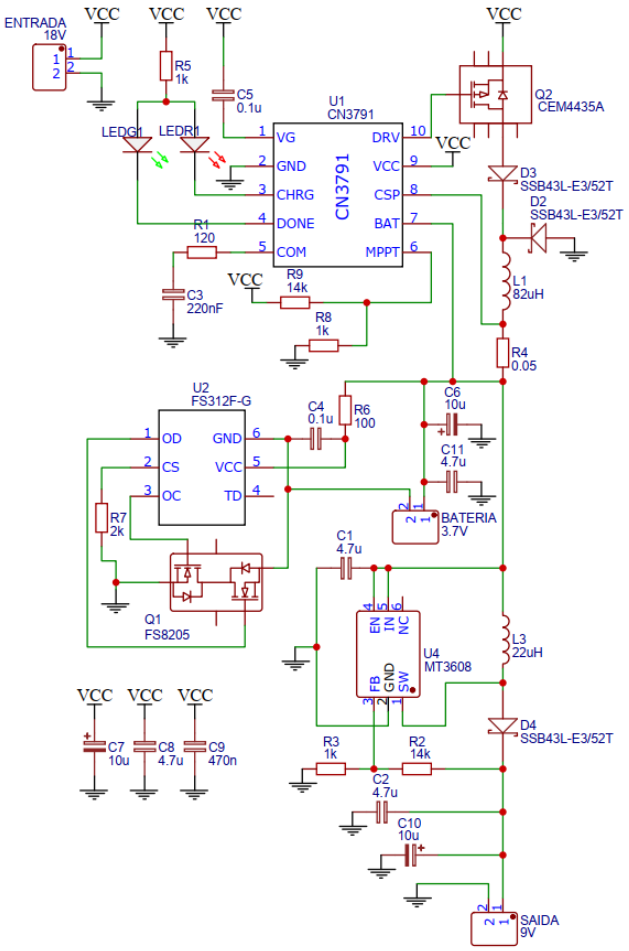
Fonte: https://www.ic-fortune.com/upload/Download/FS312F-G-DS-12_EN.pdf

Fig. 46. Aplicação típica do MT3608



Fonte: https://www.olimex.com/Products/Breadboarding/BB-PWR-3608/resources/MT3608.pdf

Fig. 47. Circuito para recarga de bateria LiPO e alimentação do Arduino Mega2560



Fonte: Autores

Esse circuito foi projetado para resolver o problema de não realizar medição quando não houver energia, realizando a alimentação do Arduino através de uma bateria de LiPO e recarregá-la quando houver energia, oferecendo uma tensão de saída constante e estável.

Há 3 CI neste circuito de desempenham papeis específicos para que o circuito atinja estes objetivos.

O primeiro CI é o CN3791 que é responsável pela recarga da bateria, oferecendo o método de carregamento de tensão e

corrente constantes. Este método de carregamento é o indicado para baterias de LiPO [7].

O segundo CI é o FS312F-G que é responsável pela proteção da bateria, protegendo-a contra curto circuito, sobre tensão e baixa tensão [8].

O terceiro CI é o MT3608 que é um step-up, responsável por elevar a tensão e manter uma saída estável. Este circuito pode ser alimentado tanto com uma fonte de 18V, como uma placa fotovoltaica de 10W com tensão em máxima potência de 18V. Na tabela V é mostrado o orçamento para a construção desta solução [9].

TABELA V
CUSTO DO CIRCUITO PARA RECARGA DE BATERIA LIPO E ALIMENTAÇÃO DO ARDUINO MEGA2560

Componente	Quantidade	Preço (un)	Total(R$)
CN3791	1	1,92	1,92
CEM4435A	1	0,73	0,73
MT3608	1	0,33	0,33
FS312-G	1	0,40	0,40
FS8205	1	0,61	0,61
Diodo SSB43L	3	1,22	3,67
Led verde	1	0,43	0,43
Led vermelho	1	0,07	0,07
Resistor 0,05Ω	1	0,21	0,21
Resistor 100Ω	1	0,10	0,10
Resistor 120Ω	1	0,10	0,10
Resistor 1kΩ	5	0,10	0,50
Resistor 2kΩ	1	0,10	0,10
Resistor 13kΩ	2	0,10	0,20
Capacitor 100nF	2	0,10	0,20
Capacitor 220nF	1	0,10	0,10
Capacitor 470nF	1	0,10	0,10
Capacitor 4,7µF	4	0,10	0,40
Capacitor 10µF	3	0,10	0,30
Indutor 22µF	1	0,07	0,07
Indutor 82µF	1	0,16	0,16
Borne KRE 2 entradas	3	2,50	7,50
Placa virgem 15x15	1	8,00	8,00
Bateria LiPO 4000mah	1	68,97	68,97
Total do circuito projetado:			**R$ 95,18**
Fonte DC 18V	1	25,33	R$ 25,33
Total com fonte DC de 18V:			**R$ 120,51**
Placa Fotovoltaica 10W	1	R$ 98,00	R$ 98,00
Total com placa fotovoltaica de 10W:			**R$ 193,18**

Fonte: Autores

Empregando essas soluções, foi satisfatória a elaboração de um projeto robusto e funcional, seguindo as diretrizes iniciais da busca por operabilidade e economia. Comparando o projeto feito com a solução oferecida pela CCK, o sistema exibe vantagens para telemetria de sistemas hídricos, aonde sua vantagem estaria em seu menor custo, tamanho, modularidade.

Citando tais vantagens, o projeto proposto se impõe como mais vantajoso que o equipamento utilizado e totalmente viável para realizar as medições em sistemas hídricos.

VI. TRABALHOS FUTUROS

É sugerido a implementação do projeto em um prédio da UFG, para averiguar a robustez do sistema operando a longo prazo.

Realizar modificações no servidor implementado no projeto, para permitir a utilização de vários pontos de coleta, discernimento de qual Arduino ele está recebendo os dados.

A coleta em vários prédios permitirá analises do consumo de água de acordo com as estações do ano, umidade e temperatura para traçar um perfil de consumo correlacionando esses dados.

Realizar a troca do modulo de comunicação utilizado neste projeto para um modulo wi-fi, permitindo medições em regiões mais afastadas.

Realizar aprimoramentos no software do servidor para detecção de vazamentos e consumo anormais de água em determinados períodos, gerando alarmes e relatórios permitindo uma intervenção imediata.

Adicionar um modulo para tratamento de dados com correção de erros.

APÊNDICE

Todo o código desenvolvido para este projeto pode ser obtido em: https://github.com/melquiadesr?tab=repositories

REFÊRENCIAS BIBLIOGRÁFICAS

[1] SILVA, L. D. *Desenvolvimento de um sistema de medição de vazão com o uso da eletrônica microprocessada*. 2014. 93f. *Trabalho de conclusão de curso* – Universidade Estadual de Londrina.

[2] SALOMÃO, A. *Desenvolvimento de um Módulo ZigBee para o Monitoramento Remoto do Consumo de Água em Instalações Prediais Aeroportuárias.* 2009. 164f. *Dissertação de mestrado* – Instituto Tecnológico de Aeronáutica.

[3] DOCUMENTAÇÃO DE REFERÊNCIA DA LINGUAGEM ARDUINO. ARDUINO. Disponível em: <www.arduino.cc/reference/pt>. Acesso em: 03 nov 2019.

[4] MYSQL DOCUMENTATION. MYSQL. Disponível em: <dev.mysql.com/doc>. Acesso em: 29 nov 2019.

[5] VUE.JS DOCUMENTATION. VUE.JS. Disponível em: <vuejs.org/v2/guide>. Acesso em: 29 nov 2019.

[6] LUMEN DOCUMENTATION. LUMEN. Disponível em: <lumen.laravel.com/docs/6.x>. Acesso em: 30 nov 2019.

[7] CONSONANCE. Standalone Li-ion Battery Charger IC With Photovoltaic Cell MPPT Function.

[8] FORTUNE. One Cell Lithium-ion/Polymer Baterry Protection IC. 2014.

[9] AEROSEMI. MT3608.

[10] ALEXANDRE, A. S. Experimentos com o Arduino: 2. Ed. Revista do Arduino, 2013.

[11] CCK4400ME. CCK. Disponível em: <http://www.cck.com.br/produtos/produto.php?nmprod=CCK4400ME&modoCatalogo=1>. Acesso em: 27 nov 2019.

[12] CCK6700E. CCK. Disponível em: < http://www.cck.com.br/catalogo.php?nm=CCK6700E>. Acesso em: 27 nov 2019.

[13] CCK. sistema-cckweb-de-telemetria3.ppt.

Marcelo S. de Castro graduou-se em Engenharia Elétrica pela Universidade Federal de Juiz de Fora (1992), com mestrado em Engenharia Elétrica pela Universidade Estadual de Campinas (1995) e doutorado em Engenharia Elétrica pela UnB (2010). Professor Associado da Universidade Federal de Goiás, tendo ingressado em 1996. Possui experiência na área de engenharia de redes, computação paralela e distribuída, comunicações óticas e tecnologias alternativas de última milha (BPL, ZigBee, Wi-Fi). Desenvolve pesquisas em temas que incluem redes de comunicação (5G, Gigabit Wi-Fi), Smart Grids, Smart Cities, Smart Campus, tecnologia da informação e comunicação e gestão aplicadas a projetos de redes de telecomunicações, projetos de automação usando Plataforma Arduino e educação em engenharia.

Gustavo B. C. Matos Nasceu em Goiania-GO, Brasil, em 1996. Cursa atualmente Engenharia Elétrica na Universidade Federal de Goiás, onde encontra-se no 9º semestre da graduação. Possui experiência em projetos de circuitos elétricos analógicos .

Melquíades R. Maciel Graduando de Engenharia Elétrica na Universidade Federal de Goiás. Possui experiencia com lógica e linguagens de programação. Atualmente atua como Analista de Desenvolvimento na Maxnível, trabalhando em projetos utilizando linguagens como PHP, MySql e Javascript, além de programação em linux e utilização de serviços como Docker e Kubernetes. Estagiou na Volga Engenharia onde obteve experiência com projetos elétricos de baixa e média tensão, e soluções de subestação abrigada.