UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

RENAN CAPARRA SPADIM

DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO PARA GÁS DE COZINHA GLP

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2019

RENAN CAPARRA SPADIM

DESENVOLVIMENTO DE UM SISTEMA DE MONITORAMENTO PARA GÁS DE COZINHA GLP

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Trabalho de Conclusão de Curso 1, do curso de Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof. Cristiano Agulhari

CORNÉLIO PROCÓPIO
2019

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia de Controle e Automação

UTFPR
CAMPUS CORNÉLIO PROCÓPIO

FOLHA DE APROVAÇÃO

Renan Caparra Spadim

Desenvolvimento de um sistema de monitoramento para gás de cozinha GLP

Trabalho de conclusão de curso apresentado às 10:20hs do dia 01/07/2019 como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação no programa de Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Avaliadora composta pelos professores abaixo assinados. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

Prof(a). Dr(a). Cristiano Marcos Agulhari - Presidente (Orientador)

Prof(a). Dr(a). Luiz Francisco Sanches Buzachero - (Membro)

Prof(a). Dr(a). André Sanches Fonseca Sobrinho - (Membro)

A folha de aprovação assinada encontra-se na coordenação do curso.

RESUMO

Este trabalho foi motivado pelas possibilidades criadas pelo avanço tecnológico, permitindo que seja utilizado conhecimentos adquiridos no curso de engenharia de controle e automação para solucionar problemas no dia a dia. Neste documento detalha o desenvolvimento de um protótipo, que serve como base para projetar a solução de um problema real, no mercado de distribuição de gás de cozinha GLP. O protótipo utiliza um sensor de fluxo do modelo YF-S201 e um microcontrolador NodeMCU esp8266 para armazenar e receber dados do banco de dados em tempo real Firebase. O acesso remoto ocorre através de um smartphone com sistema operacional Android, no qual foi desenvolvido um aplicativo através do software Android Studio.

Palavras-chave: NodeMCU, esp8266, GLP, gás de cozinha, Firebase, Android.

ABSTRACT

This work was motivated by the possibilities created by technological advancement, allowing the use of knowledge acquired in the course of control and automation engineering to solve problems on a daily basis. This document details the development of a prototype, which serves as a basis for designing solutions for real problems, in the gas distribution market. The prototype uses a YF-S201 flow sensor and a NodeMCU esp8266 microcontroller to store and receive data from the Firebase real-time database. Remote access occurs through a smartphone with an Android operating system on which an application has been developed through Android Studio software.

Keywords: NodeMCU, esp8266, Firebase, Android.

SUMÁRIO

1. INTRODUÇÃO

O monitoramento de sistemas tem como principal objetivo coletar dados para analisar o comportamento, permitindo obter um resultado desejado, e que decisões sejam tomadas afim de evitar situações inoportunas.

O uso de microcontroladores para o monitoramento de sistemas tem se tornado cada vez mais viável, por diversos fatores como: portabilidade; gama de periféricos que aumentam o número de aplicações; linguagem de programação simples e de fácil implementação; atualmente os microcontroladores são equipamentos de baixo custo [ORDONEZ 2006].

Atualmente smartphones tem ganhado um grande espaço no mercado, devido a sua praticidade. São muitos usuários, por isso o uso de aplicativos tem se apresentado cada vez mais viável, pois é uma interface de comunicação entre o usuário e o sistema simples, rápida e que está na palma da mão [COUTINHO 2015].

Uma das áreas que atualmente não contam com monitoramento é o comércio de botijões de gás. Segundo dados da Agência Nacional Petróleo, Gás Natural, e Biocombustíveis (ANP), somente em setembro de 2018 foram vendidos mais de 430 milhões botijões de gás comum em todo o país. O estado de São Paulo representa cerca de 90 milhões dessas vendas. Isso significa que, em média, são vendidos mais que 125 mil botijões por hora no estado de São Paulo. Os dados apresentados são disponibilizados pela própria ANP [ANP 2018].

Como não é comum o monitoramento do gás, a necessidade de comprar um botijão vem enquanto os clientes estão utilizando o fogão. Dessa forma, o gás do botijão acaba no mesmo momento do dia, normalmente próximo ao horário do almoço. Esse problema afeta a todos os clientes, sejam redes de restaurante, até mesmo residencial. A grande demanda nos horários de pico gera um tumulto e, caso o fornecedor não possa oferecer um botijão novo instantaneamente, resulta em perdas significativas nas vendas.

Esse é um problema notável há anos e as soluções exploradas não demonstram ser eficientes. É possível monitorar o gás através de manômetros (sensores de pressão) analógico, porém além de imprecisa, a informação analógica exige que o cliente realize todo o trabalho de leitura e armazenamento de dados.

Esse trabalho propões o monitoramento da quantidade de gás ainda presente em um botijão por meio de estimativas realizadas em um microcontrolador. Dentro dessa

área, é possível aplicar diversos conhecimentos da área de engenharia de controle e automação, porém o primeiro passo é a digitalização da informação. Converter um dado analógico é essencial para que o microcontrolador seja responsável por todo o tratamento de dados.

Os dispositivos que são utilizados em sistemas de gás GLP devem cumprir as normas ABNT NBR 15526 e no mercado não há opções viáveis de sensores metálicos que possam ser utilizados nessa aplicação, pois normalmente são robustos e de alto custo.

2. OBJETIVOS GERAIS

O desenvolvimento deste trabalho tem como finalidade apresentar a solução do problema apresentado, através do monitoramento do botijão, permitindo que o usuário possa acessar a informação utilizando uma interface. O protótipo desenvolvido tem como finalidade comprovar que a solução é factível, para isso utilizasse água. Também tem-se como objetivo comentar as normas exigidas pela ABNT NBR 15526, que determinam as características físicas do sistema.

3. OBJETIVOS ESPECÍFICOS

- Montar o protótipo físico, contemplando uma garrafa plástica com volume de 2 litros, um registro manual em série com o sensor de fluxo YF-S201.
- Inserir o microcontrolador NodeMCU esp8266 ao protótipo físico, de forma a receber pulsos do sensor e processar a informação.
- Criar um banco de dados em tempo real na plataforma firebase, que será responsável pelo intermédio da comunicação entre o microcontrolador e o usuário.
- Realizar a comunicação entre o microcontrolador e o banco de dados em tempo real, permitindo enviar e receber dados.
- Desenvolver um aplicativo para Android através do software Android Studio, capaz de enviar e receber dados do banco de dados em tempo real.

4. MATERIAIS E MÉTODOS

4.1. Banco de Dados

O Firebase Realtime Database é uma plataforma utilizada para gerenciar dados em tempo real. A plataforma armazena dados no formato de documento JavaScript Object Notation (JSON), que é um modelo para armazenamento e transmissão de informações no formato texto [VIEIRA 2012]. A plataforma utiliza uma biblioteca cliente própria, permitindo enviar e subscrever em qualquer endereço do banco; receber e enviar notificações das mudanças nos dados. Apesar de muito simples, tem sido bastante utilizada por aplicações Web devido a sua capacidade de estruturar informações de uma forma bem mais compacta.

O Firebase tem se tornado uma plataforma muito completa. Oferece sincronismo entre todos os clientes conectados; conta com autenticação de forma facilitada; armazenamento de arquivos; analytics e notificações push para Android e iOS; hosting de aplicações Web; e o Firebase Realtime Database.

Normalmente, a comunicação entre o cliente e o banco de dados exige a programação de um servidor, como ilustra a Figura 1. O servidor é responsável por estabelecer a conexão entre os periféricos. Os periféricos dependem diretamente do servidor para se comunicar [KUROSE 2006].

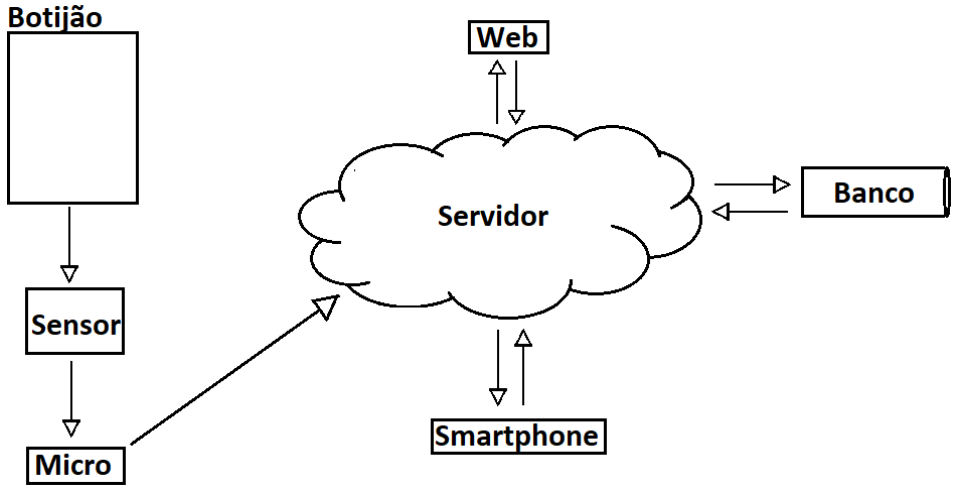


Figura 1 Diagrama de comunicação servidor cliente

fonte: autoria própria

A plataforma Firebase, além do banco de dados em tempo real, também faz o papel de servidor, sendo responsável pelas etapas de verificação de usuário, armazenamento de

dados, e acesso. Também permite a criação de aplicações web sem a necessidade de programar um servidor, pois atua como um, assim como ilustra a Figura 2.

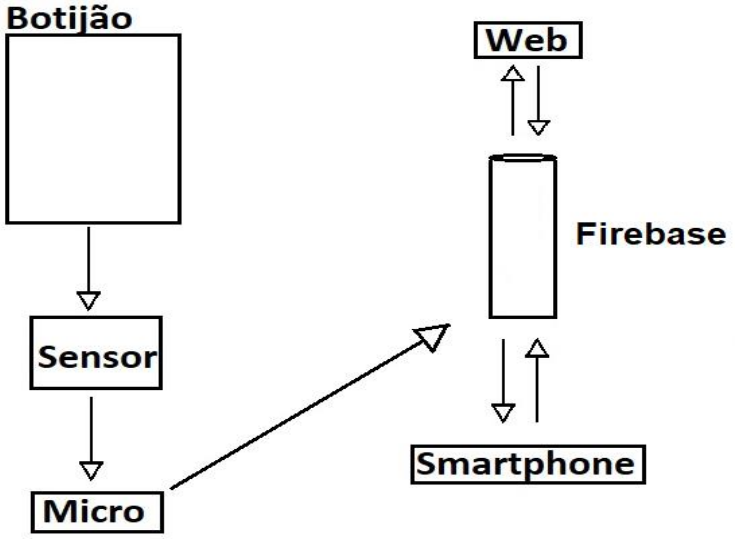


Figura 2 Comunicação entre periféricos utilizando a plataforma Firebase
fonte: autoria própria

Outra vantagem para a aplicação é de que os dados são armazenados de maneira não relacional. Um banco de dados relacional pode ser representado por uma tabela, onde cada linha está relacionada a uma ou mais colunas. Muitas vezes ações como adicionar uma nova linha, ou coluna, podem não ser tão simples, principalmente pelo fato de que todas as informações que estão relacionadas a linha 1 estarão obrigatoriamente relacionadas as outras linhas, isso é, todas as linhas estão relacionadas a todas as colunas. Já num banco de dados como o da plataforma, os dados armazenados são independentes, ou seja, é possível excluir ou adicionar qualquer informação, relacionada a qualquer classe, sem comprometer a integridade dos outros dados [VIEIRA 2012].

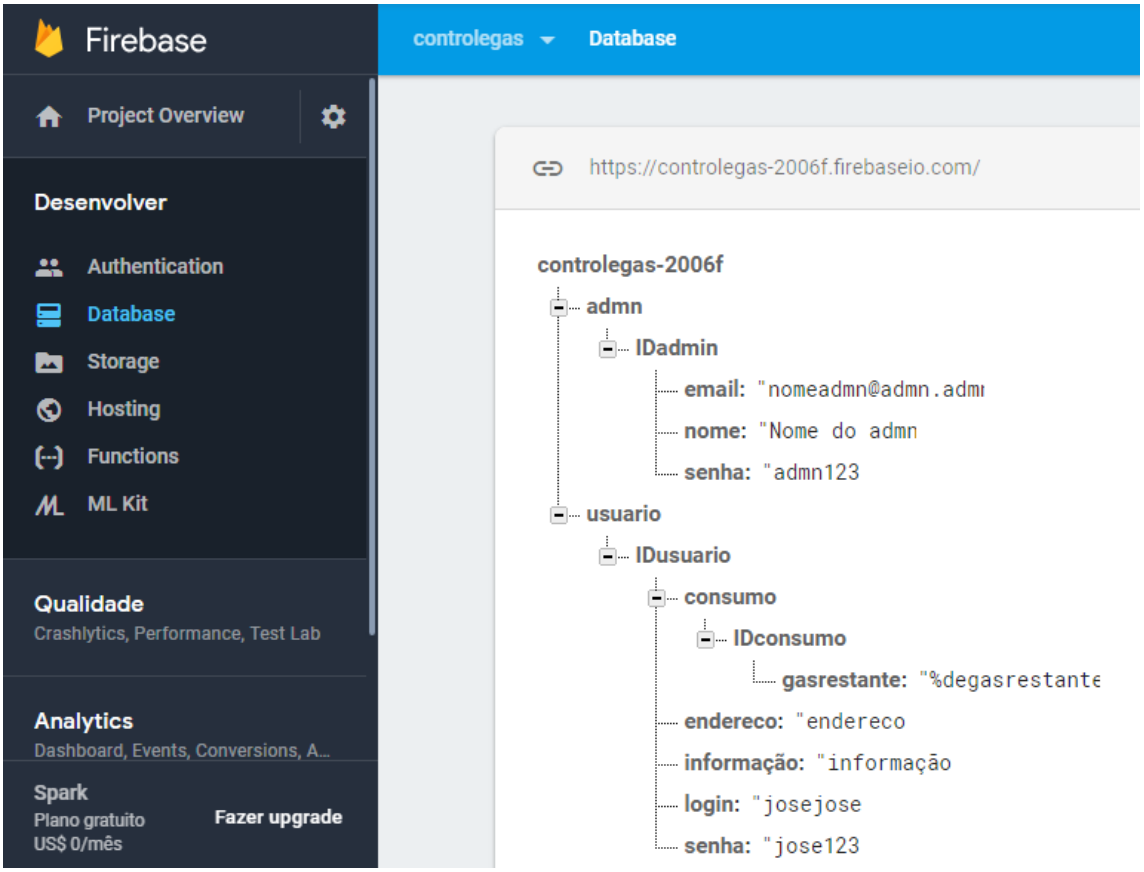


Figura 3 Exemplo de banco de dados em tempo real na plataforma Firebase
fonte: autoria própria

Como demonstra o banco apresentado na Figura 3, o Firebase proporciona uma ferramenta simples para acesso e controle de dados, sendo apropriado para aplicações que utilizam comunicação em tempo real, como este projeto. O armazenamento é feito por chave-valor. Na imagem anterior, a chave “usuário” contém as chaves “consumo”, “endereço”, “login”, “senha”, e qualquer outra informação pertinente ao usuário. A chave “login” está relacionada ao valor “josejose”. Neste exemplo a massa de gás restante seria atualizada na chave “IDconsumo”. Toda vez que o microcontrolador inserir uma nova informação ao dado, uma chave “gasrestante” será adicionada, relacionada ao valor. É possível que as informações sejam acumulativas ou que a chave seja deletada quando uma nova for inserida.

A plataforma também proporciona restrições para acesso aos dados. Quando identificado um administrador, é proporcionado acesso à todos os valores, e gerenciamento de todas as chaves. O desenvolvimento das restrições são livres, podendo ser diferente em cada aplicação.

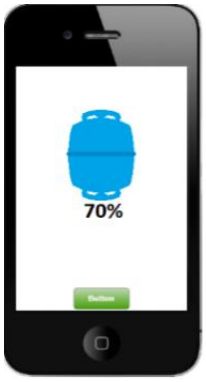
Essas características proporcionam uma larga escalabilidade do sistema, ou seja, o desempenho para o controle, inserção, deleção e alteração de dados não decaí com o aumento do número de usuários.

O acesso de cada usuário é definido por regras pelo próprio Firebase, e o gerenciamento é determinado por funções dentro da plataforma.

4.2. Interface de Comunicação

Nos dias de hoje, tem se tornado cada vez mais comum o uso de smartphones nas atividades cotidianas, devido a praticidade que essa tecnologia tem oferecido [FEIJÓ 2013]. Quando bem utilizado, é uma ferramenta poderosa para as aplicações de engenharia de controle e automação.

Existem diversas maneiras de desenvolver um software para smartphones e tabletes[GOMES 2013]. Para o desenvolvimento desse projeto será utilizado o software Android Studio, que é um ambiente de programação em JAVA [MENDES 2009]. O motivo da escolha é porque é fácil estabelecer a comunicação entre o software e a plataforma Firebase, pois o Android Studio conta com uma biblioteca do Firebase já inclusa, fazendo a conexão entre o banco de dados em tempo real e o aplicativo de uma maneira muito intuitiva. A Figura 4 apresenta uma possível tela de interface, apresentando somente as informações de interesse do usuário.



.

Figura 4 Ilustração da tela do celular

Fonte: autoria própria

Com comandos pré-definidos da biblioteca firebase[FIREBASE 2018], é possível realizar ações como definir ou ler o valor de uma variável, tornando possível a comunicação

entre enviar e receber dados do banco. Somado à programação das bibliotecas convencionais, pode-se utilizar caixas de textos e botões para exibir ou inserir dados do banco.

Assim como será detelhado na seção seguinte, o objetivo é que um microcontrolador insira o valor do volume restante no recipiente, ao mesmo tempo em que o aplicativo busca e exibe esse valor na tela.

4.3. Microcontrolador

A plataforma utilizada para o desenvolvimento do projeto é chamada NodeMCU, composta por um modulo ESP8266 e um microprocessador de 32 bits com suporte a conexão Wi-Fi, além de portas de alimentação e programação, 10 entradas digitais e uma analógica [KOLBAN 2016].

O uso do ESP8266(NodeMCU) tem se tornado comum por dois fatores: o baixo custo e os recursos suficientes para diversas aplicações que envolvam a Internet das Coisas (Internet of Things - IoT) [DE OLIVEIRA 2017]. Como o sistema proposto deve possuir um baixo custo e realizar uma comunicação sem fio, esse dispositivo é o mais adequado. Seus pinos são especificados na Figua 5.

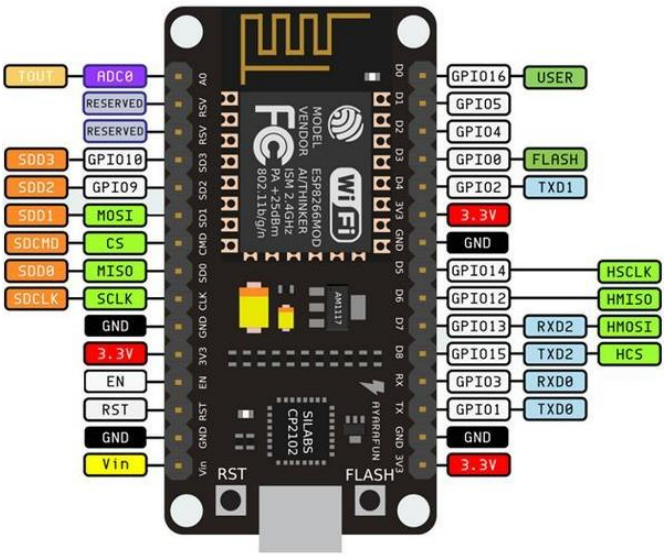


Figura 5 Pinos do microcontrolador nodeMCU esp8266
Fonte: https://iotbytes.wordpress.com/nodemcu-pinout

O microcontrolador foi programado através do software IDE Arduino®, em linguagem C. Suas especificações técnicas permitem aplicações ainda mais completas, pois tem memória SRAM de aproximadamente 36kB, 16 MB de SPI flash, Wireless padrão 802.11 b/g/n e antena embutida. Opera nos modos STA/AP/STA+AP e suporta 5 conexões TCP/IP. Opera na faixa de tensão de 3~3.6V e dispões de 11 portas GPIO, com funções de PMW, I2C, SPI, etc. Contém conversor analógico digital (ADC). [Espressif Systems IOT Team 2015].

Outra grande vantagem do nodeMCU nessa aplicação é que é possível adicionar uma biblioteca do firebase, disponível na internet. Dessa forma fica muito fácil estabelecer a comunicação entre o microcontrolador e a plataforma.

Para este protótipo, o micro será programado para realizar as funções mínimas exigidas no projeto, que são elas: estabelecer conexão wi-fi, para ter acesso a internet; estabelecer a conexão com a plataforma Firebase; obter a vazão de fluído, através do sensor de fluxo; subtrair do volume de fluído total, que ainda contém no recipiente, o fluído deslocado; calcular a porcentagem, estabelecendo uma relação entre o volume atual com o volume inicial; inserir o valor da variável no endereço do banco de dados da plataforma.

Para que o microcontrolador tenha conhecimento da vazão, é necessário inserir o sensor de fluxo na saída do reservatório, assim como é explicado na próxima sessão.

4.4. Sensor de Fluxo

Sensores de fluxo que possam ser utilizados para medir a vazão de saída do gás de cozinha não são comuns, devido aos materiais exigidos nas normas da ABNT NBR 15526. Os sensores para gás costumam ser robustos, e de alto custo. Para o desenvolvimento do produto final e posterior inserção no mercado, é necessário produzir o sensor. Este trabalho não tem como finalidade o desenvolvimento final do dispositivo, mas sim especificar os requisitos de projeto, e propor um protótipo funcional, que realiza a medição da quantidade de água em um recipiente. Para isso será utilizado um sensor de fluxo, detalhado a seguir.

4.4.1. Princípio de Funcionamento

O Efeito Hall foi descoberto em 1879 por Edwin H. Hall (1855-1938) e consiste no aparecimento de uma diferença de potencial (também chamada de diferença de potencial de Hall), transversal ao fluxo de corrente que atravessa um condutor, quando este se aproxima de um campo magnético transversal ao condutor (HALLIDAY, 2007).

Um sensor Hall se baseia em um transdutor (BORGES 2015) que, quando sob a aplicação de um campo magnético, responde com uma variação em sua tensão de saída. Em outras palavras, quando um imã se aproximar do transdutor será produzida uma tensão de saída.

Acoplando um imã a um impulsor, é possível converter o movimento circular em pulsos elétricos. Por isso sensores baseados nesse fenômeno são muito utilizados para medir fluxo. Na sequência é apresentada a arquitetura do sensor YF-S201. O sensor YF-S201 funciona através do efeito Hall, tem um baixo custo, e proporciona uma leitura através da quantidade de pulsos, ao invés da frequência, facilitando a programação do micro. A tabela 1 e a Figura 6 apresentam a composição da arquitetura do sensor.

Nº	Item	Qtd.
1	Cabo de conexão	1
2	Bonnet	1
3	Parafuso	4
4	Corpo da válvula	1
5	Válvula de alívio de pressão	1
6	Imã	1
7	Sensor Hall	1
8	Impulsor	1
9	Eixo de aço sem ferrugem	1

Tabela 1: Itens que compõe o sensor

Fonte: https://www.hobbytronics.co.uk/datasheets/sensors/YF-S201.pdf

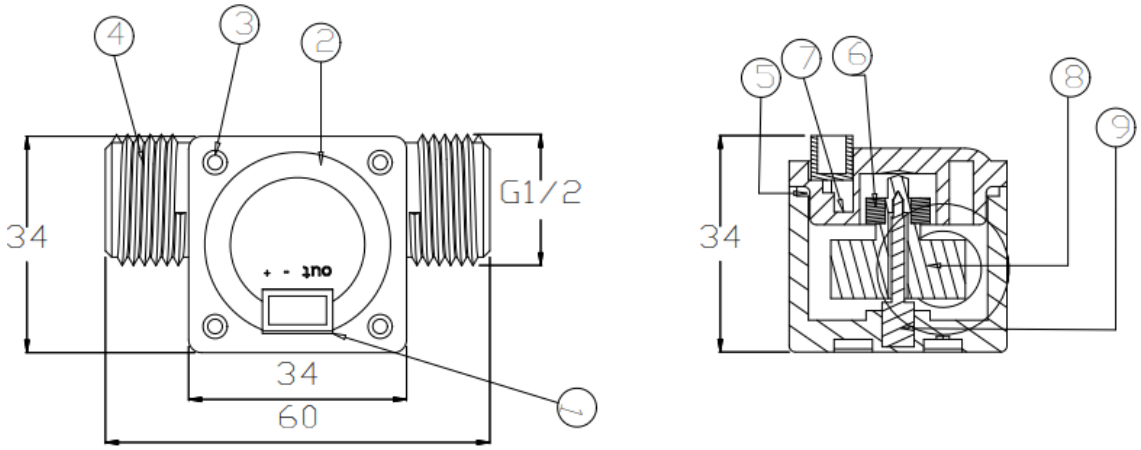


Figura 6 Arquitetura do sensor de fluxo YF-S201

Fonte: https://www.hobbytronics.co.uk/datasheets/sensors/YF-S201.pdf

Quando um fluído atravessa o corpo da válvula (4), ele empurra as pás do impulsor (8). Acoplado ao eixo está o imã (6), que descreve um movimento circular em torno do impulsor(8). No momento em que o imã (6) se aproxima do transdutor (7), a saída do sensor (1) vai para nível lógico alto, retornando para nível lógico baixo somente depois que o ímã se afasta. Observando a saída com um osciloscópio é possível ver que o movimento circular do impulsor é visto como uma onda quadrada pelo microcontrolador. Medindo a quantidade de pulsos em um intervalo de tempo é possível determinar o fluxo. A tabela a seguir relaciona a frequência de pulsos com o fluxo computado.

Fluxo (L/H)	Freq. (Hz)	Erro
120	16	±10%
240	32.5	
360	49.3	
480	65.5	
600	82	
720	90.2	

Tabela 2 Relação de frequência de pulsos com fluxo computado

Fonte: https://www.hobbytronics.co.uk/datasheets/sensors/YF-S201.pdf

Para determinar a massa de fluído que ainda resta no recipiente, é possível relacionar a quantidade de fluído deslocada por pulso. Dessa forma se torna desnecessário conhecer o consumo, e exige um nível de programação mais baixo. Pode-se relacionar o volume deslocado para cada pulso, pela relação:

$$D = \frac{\varphi}{3600\, f},$$

onde D é o volume deslocado (em litros por pulso), φ é o fluxo (em litros por hora), f é a frequência (em Hz) e 3600 é a conversão do fluxo para litros por segundo. A tabela a seguir apresenta a relação entre a frequência e a massa de fluído deslocada.

Fluxo (L/H)	Fluxo (mL/s)	Freq. (Hz)	Massa deslocada (mL/pulso)	Erro
120	33,34	16	2,08	±10%
240	66,67	32.5	2,05	
360	100,00	49.3	2,03	
480	133,34	65.5	2,03	
600	166,67	82	2,03	
720	200,00	90.2	2,22	

Tabela 3 Relação frequência com massa de fluido descolada por pulso

Fonte: autoria própria

Conforme apresentado na Tabela 3, é possível notar que a relação não é linear, porém como a aplicação não exige uma precisão elevada, é possível aproximar que 2,05mL terão vazado para cada pulso gerado pelo sensor. Uma vez que o volume inicial é conhecido, basta subtrair 2,1mL para cada borda de subida gerada no pulso.

Os valores apresentados anteriormente são especificados para a água, porém com alguns ajustes de parâmetros é possível aplicar, de maneira análoga, a um sistema de gás. Os parâmetros podem ser obtidos de maneira empírica, porém a grande dificuldade em se utilizar o gás é o requisito de que o dispositivo cumpra as normas exigidas pela ABNT, detalhadas na sessão seguinte.

4.4.2. Norma ABNT NBR 15526

A norma ABNT NBR 15526 estabelece os requisitos mínimos exigíveis para o projeto e execução de redes de distribuição interna para gases combustíveis em instalações residenciais.

Os materiais, equipamentos e dispositivos utilizados na rede de distribuição interna

devem possuir resistência físico-química adequada à sua aplicação e compatível com o gás utilizado.

Para a execução das conexões são admitidas:

- Conexões de aço forjado, com as especificações da ASME/ANSI B.16.9;
- Conexões de ferro fundido maleável, conforme ABNT NBR 6943, ou ABNT NBR 6925;
- Conexões de cobre e ligas de cobre para acoplamento soldado ou roscado dos tubos de cobre, conforme ABNT NBR 11720;
- Conexões com terminais de compressão para uso com tubos de cobre, conforme ABNT NBR 14463;
- Conexões para transição entre tubos PE e tubos metálicos, para redes enterradas, conforme ASTM D 2513, ASTM F 1973 e ASMT F 2509;
- Conexões de ferro fundido maleável com terminais de compressão para uso com tubos PE, ou transição entre tubos PE e tubos metálicos, para redes enterradas, conforme ISSO 10838-1 ou DIN 3387.

A norma não trata de materiais, equipamentos e dispositivos não explicitamente citados, porém não tem a intenção de restringir o desenvolvimento de outros itens e tecnologias.

Os materiais citados possuem características e comportamentos conhecidos para esses propósitos.

A consideração de outros materiais, equipamentos e dispositivos leva, normalmente, em conta os seguintes itens:

- Existência de especificação dos materiais, equipamentos e dispositivos em norma ou regulamentação técnica em âmbitos nacional ou internacional, incluindo sua utilização;
- A garantia de que os materiais, equipamentos e dispositivos atendem às referências normativas citadas;
- Existência de histórico de mercado;
- Avaliação do uso de materiais, equipamentos e dispositivos no ambiente, incluindo análise de ensaios quando pertinente;
- Existência de recomendação técnica referente à aplicação e utilização dos materiais, equipamentos e dispositivos nas redes internas de distribuição de gases combustíveis, no âmbito da normalização internacional;

- Avaliação de validade da aprovação dos materiais, equipamentos e dispositivos no cenário internacional nas redes internas de distribuição de gases combustíveis, com evidência de uso e aplicação em diversos lugares.

Uma vez que o fluído utilizado no desenvolvimento do protótipo é a água, ele não precisa atender esses requisitos, porém para o desenvolvimento do produto final é de extrema importância não só cumprir as normas, como também ser apresentados relatórios de testes, para obter um selo de segurança.

5. METODOLOGIA

5.1. Princípios de Funcionamento do Protótipo

O sistema físico que será apresentado como protótipo consiste em uma garrafa plástica de 2 litros, com um registro de plástico acoplado na saída da garrafa. Em série com o registro está um sensor de fluxo hall YF-S201, conectado ao microcontrador nodeMCU esp8266. É fornecida a alimentação ao node através de um cabo USB. Após o sensor, uma mangueira de silicone dispensa o líquido em um reservatório com indicação de volume, para que possa ser feita a leitura visual do volume de líquido. A porcentagem de líquido restante na garrafa é obtido através da subtração entre o volume dispensado no reservatório do volume inicial conhecido, em seguida de um cálculo de porcentagem simples.

O microcontrolador está conectado o tempo todo com o banco, de maneira inviável para aplicações com muitos usuários. Da maneira que foi desenvolvido, o micro está continuamente enviando o valor da porcentagem ao banco, como também está lendo uma segunda variável, que será responsável por resetar a contagem, simulando a troca do recipiente vazio por um cheio. Quando o sensor gera um pulso na porta de entrada, ele gera uma interrupção, chamando a função responsável pela contagem de pulsos e subtração do volume de água deslocado. A alimentação do micro é feita através de um cabo USB e uma fonte de carregadores de celular convencional.

O banco de dados em tempo real desenvolvido na plataforma firebase contempla informações simbólicas, para representar mais de um usuário e simular como o microcontrolador se comunicaria com seu endereço específico. Somenta duas variáveis do banco são de fato utilizadas, sendo uma responsável por armazenar a porcentagem de

líquido na garrafa e outra para reiniciar a contagem para 100% através do smartphone.

O aplicativo para android foi desenvolvido para que seja exibida uma tela com uma caixa de texto lendo a variável de porcentagem em tempo real e exibindo esse valor, uma imagem centralizada e um botão, abaixo da caixa de texto, para que a contagem possa reiniciada quando a garrafa estiver cheia novamente.

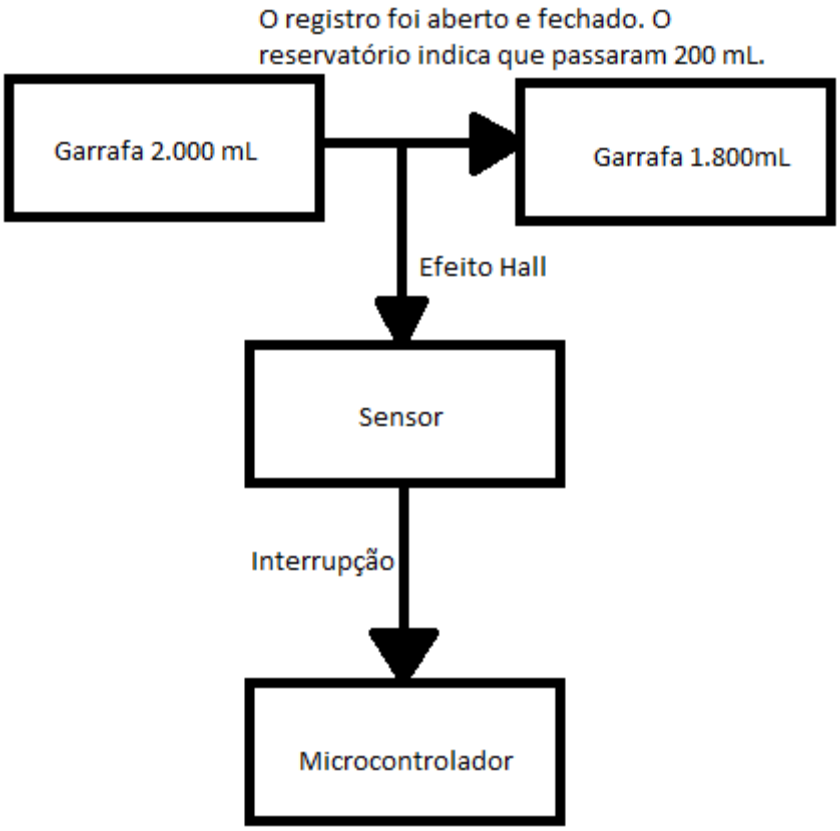


Figura 7 Diagrama de Blocos do Sistema Físico

Fonte: Autoria Própria

Conforme ilustra a figura 7, quando o registro for aberto, uma quantidade de água irá vazar da garrafa e ir para o reservatório. No exemplo da Figura 7, foram observados 200mL visualmente, através das marcações do reservatório. Logo é possível saber que o volume restante na garrafa é 1800 mL, basta subtrair o valor vazado do valor inicial. O sensor irá gerar aproximadamente 95 pulsos, cada um deles gerando uma interrupção, fazendo com que o micro pare sua rotina para processar esse dado, como apresentado a seguir.

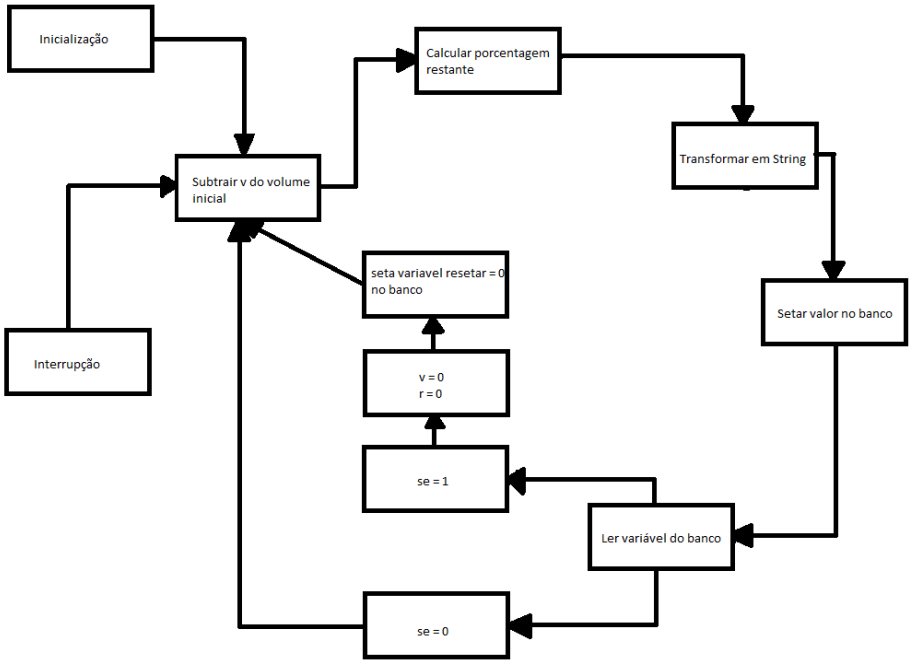


Figura 8 Diagrama de Blocos do Sistema Programado no Micro

Fonte: Autoria Própria

A rotina apresentada na figura 8 tem 4 variáveis como foco, sendo elas denominadas de “v”, “r”, “porcentagem” e “vz”. A variável “v” representa o volume que vazou da garrafa, sendo incrementado toda vez que uma interrupção é gerada. A variável “porcentagem” representa a porcentagem que o volume restante representa, comparado à 2000. A variável vz é necessária pois o aplicativo para smartphone estará exibindo a informação em uma caixa de texto, logo é necessário converter a porcentagem de float para string. A variável r está associada à variável “resetar” do banco, e é utilizada como condição de reiniciar a contagem.

Inicialmente, o programa se inicia definindo as configurações do micro, isso é, realizando a conexão Wi-Fi e com o banco de dados. Depois, o micro entra em uma rotina fechada, onde ele subtrai a variável “v” do valor 2000. Em seguida, calcula a porcentagem restante e armazena essa informação no banco. Logo após, lê o endereço do banco “resetar” e analisa a condição para caso o valor seja 0 ou 1. Se for 1, significa que o experimento deve ser resetado e o volume inicial voltar para 2000, sendo assim ele zera o valor de “v”, zera o valor de “r” e seta que o endereço do banco resetar volte para zero.

Cada interrupção gerada pelo sensor irá acrescentar 2,1 ao valor de “v”. Seguindo o exemplo da Figura 7, aproximadamente 95 pulsos foram gerados, então o valor de “v” é igual a 199,5. Subtraindo de 2000, o valor estimado é que restam 1800,5 mL dentro da garrafa. Calculando a porcentagem simples, restam 90,02% e o valor de 90,02 deve ser armazenado ao banco. O valor correto deveria ser 90, porém o erro é aceitável para a precisão exigida pela aplicação.

Uma vez que o valor esteja armazenado no banco, o aplicativo desenvolvido para android deve estar buscando a informação no banco, exibindo no display do celular e, quando pressionado o botão, setar o endereço do banco “resetar” para 1, indicando que a contagem deve ser reiniciada.

5.2. Desenvolvimento do Protótipo

Inicialmente foi realizada a montagem do sistema de armazenamento de água. A tampa da garrafa plástica foi furada, para que o registro possa ser encaixado. Para fazer a vedação, foi utilizada pasta de adesivo de silicone branca.

Dois pedaços de mangueira de silicone foram encaixados nas saídas do sensor e vedadas com fita isolante. A fita isolante também foi utilizada para a fixação do sensor na saída do registro.

Após soldar os cabos do sensor no nodeMCU, a placa foi fixada no sensor com o uso de duas abraçadeiras. Conforme ilustra a Figura 9.

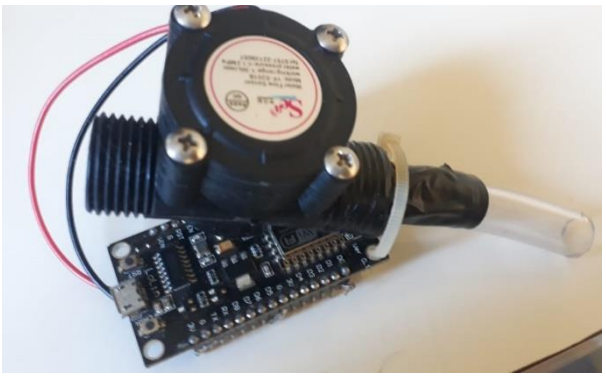


Figura 9 Microcontralador conectado ao sensor.
Fonte: Autoria Própria

Figura 10 Sistema de armazenamento de água montado

Fonte: Autoria Própria

A figura 10 apresenta o conjunto do sensor e micro fixado na garrafa plástica com o registro.

Através do software IDE Arduino foi implementado o código no anexo A.

O código fonte para o desenvolvimento do aplicativo no software Android Studio é apresentado no anexo B.

6. RESULTADOS

Após alimentar o nodeMCU na fonte, foram realizados diversos testes. O registro foi aberto, permitindo que saísse água da garrafa e fosse para o reservatório. Foi verificado que a porcentagem era atualizada em tempo real, no banco de dados e na tela do smartphone.

O registro poderia ser aberto novamente ou não, acumulando ao reservatório um maior volume de água. Quando a garrafa se esvaziava ou era de interesse reiniciar o teste,

a garrafa era reabastecida de água e o botão na tela do celular era pressionado. O endereço do banco nomeado “resetar” era setado para 1, conforme o aplicativo foi designado. O microcontrolador, que está lendo esse endereço testa a condição e, uma vez satisfeita “r = 1”, ele volta o volume da garrafa para 100% e redefine o endereço “resetar” para 0.

Alguns dos experimentos serão relatados a seguir.

O registro foi aberto e foi verificado que aproximadamente 300mL’s haviam saído da garrafa, conforme mostra o reservatório apresentado na Figura 11.

Figura 11 Recipiente de água do primeiro teste.

Fonte: Autoria Própria.

O volume inicial da garrafa é 2 000 mL. Subtraindo 300 mL, sabia-se que haviam então 1700 mL restantes, o que representava, aproximadamente, 85%.

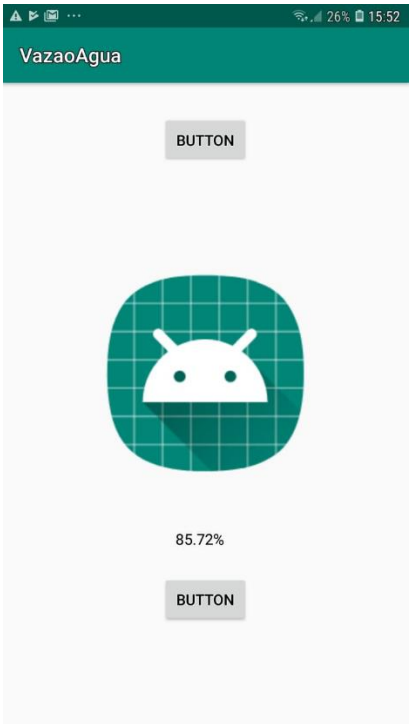


Figura 12 Foto da tela do celular.

Fonte: Autoria Própria.

O esperado era 85%, então para o primeiro experimento os resultados obtidos na Figura 12 foram aceitáveis. Também foi verificado o banco de dados, onde o valor também tinha sido alterado, conforme a imagem a seguir.

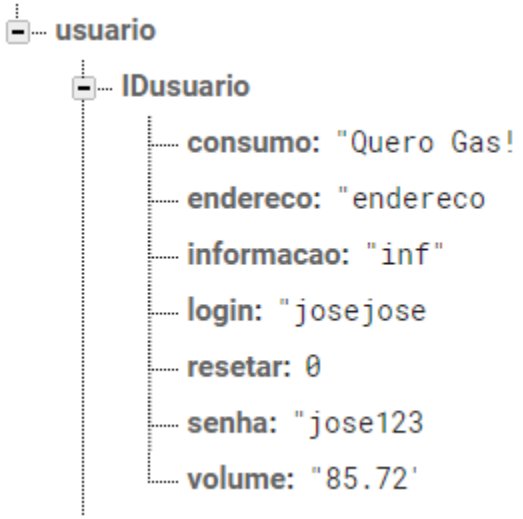


Figura 13: Imagem do banco de dados em tempo real da plataforma Firebase.

Fonte: https://firebase.google.com/?hl=pt-BR.

Conforme ilustra a Figura 13, o banco é somente um exemplo onde o endereço “consumo”, com o valor “Quero Gás” pode ser um endereço destinado a realizar um pedido.

Quando o usuário pressionar outro botão o valor “Quero Gás!” é armazenado no endereço. Variáveis como “endereço”, “nome” ou “informação” podem ser utilizadas para armazenar quaisquer informações relevantes do usuário. A identificação do usuário é realizada através dos endereços “login” e “senha”. O endereço “resetar” é o endereço utilizado para informar o micro que ele deve reiniciar a contagem e “volume” é o campo para armazenamento da porcentagem.

Após o primeiro teste, foi pressionado o botão para reiniciar a contagem para 100%. Foi verificado que a variável “resetar” alterou para 1, voltou para 0 e a variável “volume” retornou para 100, conforme ilustra as Figuras 13, 14 e 15.

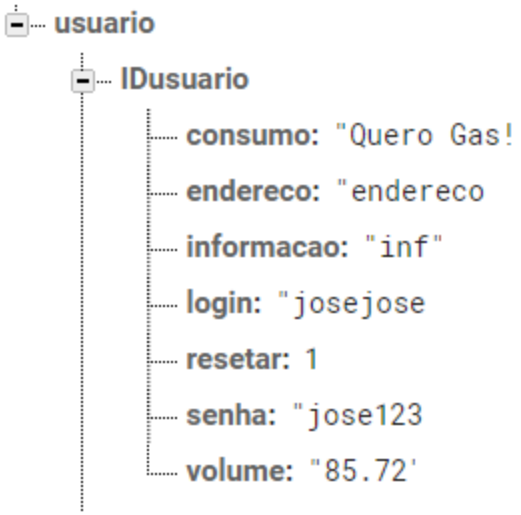


Figura 13: Imagem do banco de dados com a variável “resetar” definida para 1.

Fonte: https://firebase.google.com/?hl=pt-BR.

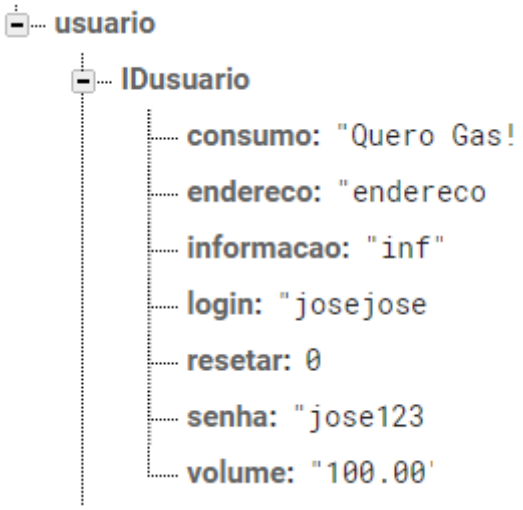


Figura 14: Imagem do banco de dados com a variável “resetar” definida para 0 e “volume” 100 %.

Fonte: https://firebase.google.com/?hl=pt-BR.

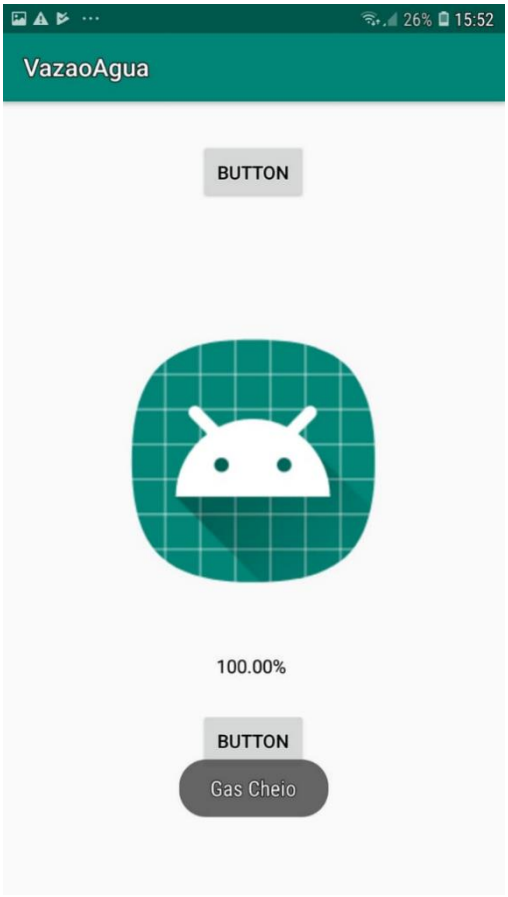


Figura 15: Tela do celular após o botão ser tocado.

Fonte: Autoria Própria.

O teste foi refeito e o seguinte resultado foi obtido.

Figura 16: Reservatório do segundo teste.

Fonte: Autoria Própria.

Analisando a Figura 16 pode-se notar que aproximadamente 200 mL haviam saído da garrafa, logo os 1800 mL restantes representavam 90% do volume inicial. O valor apresentado no banco e na tela do smartphone estavam aceitáveis, conforme apresentado nas Figuras 17 e 18.

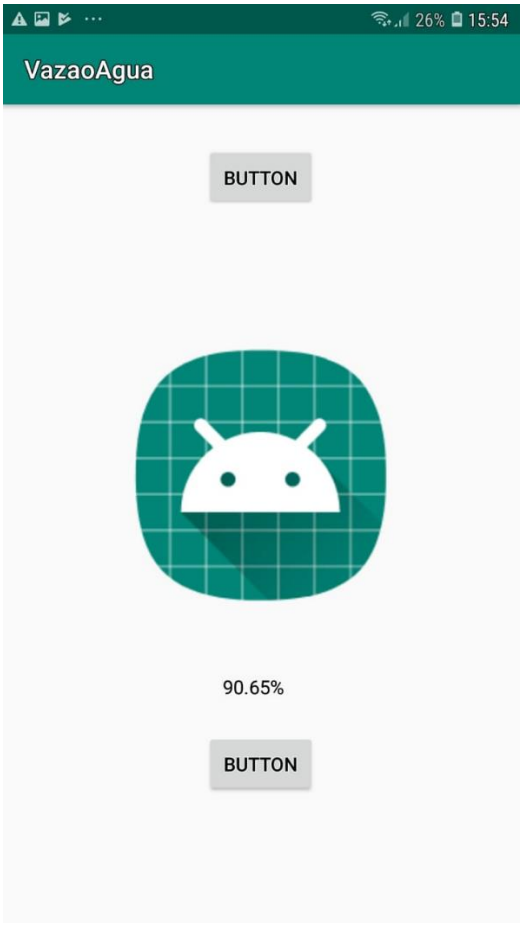


Figura 17: Tela do celular.

Fonte: Autoria Própria.

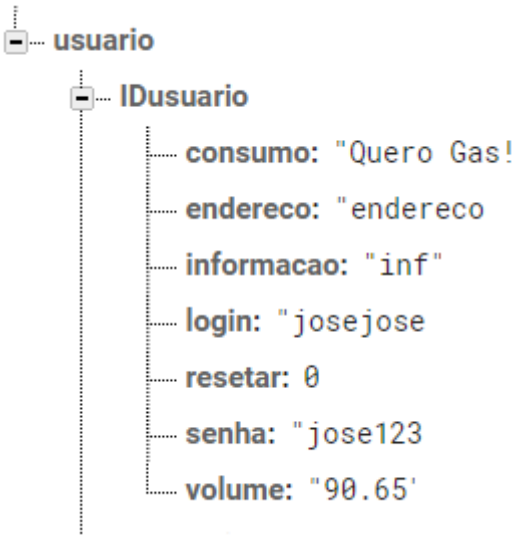


Figura 18: Banco de dados Firebase.

Fonte: https://firebase.google.com/?hl=pt-BR.

Para finalizar o experimento, a garrafa foi cheia e aberta completamente, até que

se esvaziasse por completo. Os resultados foram uma pequena porcentagem, conforme a Figura 19 apresenta. Segundo a medida do sensor vazaram aproximadamente 20 mL's a mais, o que é justificado por que a garrafa quando completamente cheia tem um volume um pouco maior que 2000 mL e não foi possível ter precisão na hora de enche-la. Apesar disso é válido considerar que o protótipo obteve o desempenho esperado.

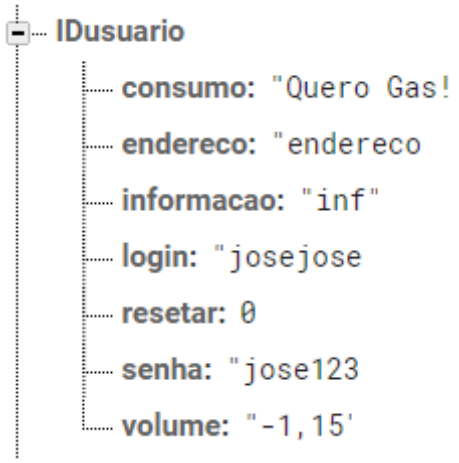


Figura 19: Banco de dados Firebase.

Fonte: https://firebase.google.com/?hl=pt-BR.

7. CONSIDERAÇÕES FINAIS

A plataforma Firebase se apresentou uma excelente ferramenta para aplicações de engenharia, já que oferece bibliotecas prontas para facilitar a programação em diversos softwares de desenvolvimento. Dessa maneira, é possível desenvolver sistemas complexos de maneira fácil.

O microcontrolador nodeMCU é um excelente componente, seu módulo wi-fi embutido diminui drasticamente o nível de programação exigido para comunicar o micro com a internet. Ideal para aplicações que envolvam comunicação entre redes sem fio e que possam ser utilizados sinal wi-fi.

O sensor de fluxo YF-S201 é recomendado para outras aplicações na área, uma vez que tem um funcionamento simples, é compacto, leve e principalmente de baixo custo. Apresentou uma precisão aceitável.

As normas da ABNT NBR 15526 são importantes, pois gases apresentam riscos à

vida. Por isso é importante segui-las para garantir que o material irá resistir ao fluído, afim de evitar possíveis vazamentos. Lembrando que o gás GLP é altamente inflamável, logo qualquer problema pode representar bastante perigo. Dessa forma, também é necessário emitir um relatório de testes, caso seja dada continuidade ao projeto.

O Android Studio proporciona um ambiente de programação didático. Com pouco conhecimento da área é possível programar softwares para uma gama enorme de aplicações. Em contra partida, para o desenvolvimento de um software seguro, que mantenha a integridade das informações do cliente, é necessário um conhecimento mais aprofundado.

Para obter o projeto como um produto final e inserir no mercado, existiram conceitos de outras áreas além de engenharia de controle e automação, porém foi possível construir um protótipo funcional, com um bom desempenho e que pode ser melhorado ou adaptado a diferentes aplicações à qualquer momento.

É possível em trabalhos futuros adicionar funções, tais como detectar vazamento de gás, ou até mesmo aplicar controle em atuadores com PWM para temperatura do forno, por exemplo. Também é possível estimar essa porcentagem utilizando balanças, assim ficaria mais fácil inseri-lo no mercado.

REFERÊNCIAS

ANP. **Consumo Aparente de GLP**. 2018. Disponível em: <http://www.anp.gov.br/images/DISTRIBUICAO_E_REVENDA/Distribuidor/GLP/Dados_de_Mercado/Consumo_Aparente.xlsx> Acesso em: 23 de outubro de 2018>.

ORDONEZ, E. D. M.; Penteado, C. G.; Silva, A. C. R. Microcontroladores e FPGA Aplicações em Automação. São Paulo: **Novatec Editora**, 2006.

VIEIRA, Marcos Rodrigues et al. Bancos de Dados NoSQL: conceitos, ferramentas, linguagens e estudos de casos no contexto de Big Data. **Simpósio Brasileiro de Bancos de Dados**, 2012.

KUROSE, James F.; ROSS, Keith W. Redes de Computadores e a Internet. **Uma nova**, 2006.

FEIJÓ, Valéria Casaroto; GONÇALVES, Berenice Santos; GOMEZ, Luiz Salomão Ribas. Heurística para avaliação de usabilidade em interfaces de aplicativos smartphones: Utilidade, produtividade e imersão. **Design & Tecnologia**, v. 3, n. 06, p. 33-42, 2013.

GOMES, Tancicleide CS; DE MELO, Jeane CB. App inventor for android: Uma nova possibilidade para o ensino de lógica de programaçao. In: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. 2013.

DE OLIVEIRA, Sérgio. Internet das Coisas com ESP8266, Arduino e Raspberry Pi. **Novatec Editora**, 2017.

SOUSA, Flávio & Moreira, Leonardo & Machado, Javam. **Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios**. 2018.

KOLBAN, N. **Kolban's Book on ESP8266**. 2016. Disponível em: <https://leanpub.com/ESP8266_ESP32>.

Espressif Systems IOT Team. **ESP8266EX Datasheet Version 4.3**. 2015. Disponível em: <https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf>.

COUTINHO, Gustavo Leuzinger**. A Era dos Smartphones: Um estudo Exploratório sobre o uso dos Smartphones no Brasil**. 2015.

BORGES, Jacques Cousteau da Silva et al. **Estudo e desenvolvimento de um transdutor de torque para eixos rotativos por meio de sensores de efeito hall**. 2015.

ANEXO A – Código Fonte Para o NodeMCU esp8266

```
//incluindo bibliotecas
#include <Firebase.h>
#include <FirebaseArduino.h>
#include <FirebaseCloudMessaging.h>
#include <FirebaseError.h>
#include <FirebaseHttpClient.h>
#include <FirebaseObject.h>
#include "ESP8266WiFi.h"

//definindo endereço e autenticação do banco Firebase
#define FIREBASE_HOST "controlegas-2006f.firebaseio.com"
#define FIREBASE_AUTH "OSfCSY3brE9Xq0HHEm3FF2yYC4K3av8S56nqIPC9"

//declarando variáveis
float v = 0; //volume de líquido que passou pelo sensor
float porcentagem = 0; //porcentagem de líquido restante na garrafa
String vz; //recebe o valor da porcentagem em formato de texto do tipo string

//criando funcao de conexão Wifi
void setupWifi(){
  //iniciando conexão com o ID e a senha da rede wifi
  WiFi.begin("VIVOFIBRA-31AA" , "33d70b31aa");//nome da refe, senha da rede
  Serial.print("connecting");//exibindo no monitor serial o texto "coneccting"
  while (WiFi.status() != WL_CONNECTED) {//aguardando enquanto o status do Wifi for diferente de conectado
    Serial.print(".");//exibindo pontos no monitor serial informando que a conexão
```

```
ainda está sendo estabelecida
            delay(500);//gerando um atraso de 500 ms
           }
           Serial.println();//pulando uma linha no monitor serial
           Serial.print("connected: ");//exibindo conectado no monitor serial
           Serial.println(WiFi.localIP());//exibindo IP da rede
          }

          //criando função que registra o volume que passou pelo sensor
          void vazao(){
           v += 1.05;//para cada pulso será subtraído 1.05 mL
          }

          //criando função de configuração
          void setup() {
           //executando funções de conexão wifi e com o Firebase
           setupWifi();//
           Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
           attachInterrupt (digitalPinToInterrupt(5), vazao, CHANGE);//definindo a função de
interrupção

           }
           //criando função de repetição
           void loop() {
            porcentagem = 100 - (100 * v / 2000); //cálculando a porcentagem
            vz = String(porcentagem);//transformando a porcentagem em String
           Firebase.setString("usuario/IDusuario/volume",vz);//setando a variavel string no
endereço do banco de dados
           int r = Firebase.getInt("usuario/IDusuario/resetar");//lendo o endereco "resetar" no
banco de dados
           if (r == 1){//testando condição para valor igual 1
            v = 0; //se sim, o volume registrado pelo sensor volta a ser 0
            r = 0; // retornando variável auxíliar "r" para 0
```

```
            Firebase.setInt("usuario/IDusuario/resetar", r);//alterando o valor do endereço "resetar" para 0
          }
        }
```

ANEXO B – Código Fonte Para Desenvolvimento do Aplicativo Para Android

```
package com.renanspadim.vazaoagua; //endereço de armazenamento do projeto

//importando bibliotecas
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

//criando função principal
public class MainActivity extends AppCompatActivity {

    //declarando variáveis
    private ImageView imageView3;
```

```
    private TextView textView2;
    private Button button3, button2;
    private int vlr;

//referênciando um layout ao código fonte
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);//layout "activity_main" associado ao
código
//associando cada variável criada com um objeto do layout
        imageView3 = (ImageView) findViewById(R.id.imageView3);//imagem central
        textView2 = (TextView) findViewById(R.id.textView2);//caixa de texto
        button3 = (Button) findViewById(R.id.button3);//botao
        button2 = (Button) findViewById(R.id.button2);//botao
//conectando ao Firebase
        FirebaseDatabase database = FirebaseDatabase.getInstance();
//criando variavéis de referência à um endereço do banco
        final DatabaseReference myRef =
database.getReference("usuario/IDusuario/volume");
        final DatabaseReference myRef2 =
database.getReference("usuario/IDusuario/resetar");
//adicionado evento à refêrncia
        myRef.addValueEventListener(new ValueEventListener() {
            @Override
            //lendo o valor do endereço referênciado
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                String value = (String) dataSnapshot.getValue(String.class);//criando
uma variaveis String para receber o valor do endereço
                value = value + "%";//adicionado o símbolo de porcentagem à variável

                textView2.setText(value);//exibindo o valor da variável na caixa de texto
```

```
            }

            @Override
            //exibindo uma mensagem caso ocorra um erro de comunicação
            public void onCancelled(@NonNull DatabaseError databaseError) {
                Toast.makeText(MainActivity.this, "erro",
Toast.LENGTH_SHORT).show();
                }

        });
        //criando evento caso seja executado um toque no botão
        button2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                myRef2.setValue(1);//setando o endereço referenciado "resetar" para 1
                Toast.makeText(MainActivity.this, "Gas Cheio",
Toast.LENGTH_SHORT).show();//exibindo um texto informando que a contagem foi
reiniciada

            }
        });
    //criando evento para a referência 2
        myRef2.addValueEventListener(new ValueEventListener() {
            //definindo que a variável associada ao valor é um inteiro
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                Integer value = (Integer) dataSnapshot.getValue(Integer.class);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
```

```
        });

    }
}
```