CURSO DE ENGENHARIA ELÉTRICA
TRABALHO DE FINAL DE CURSO

DESENVOLVIMENTO DE UM SISTEMA SUPERVISÓRIO PARA MONITORAMENTO DO CONSUMO DE ÁGUA

Artur Miranda Oliveira

SÃO JOÃO DEL REI – MINAS GERAIS

2023

ARTUR MIRANDA OLIVEIRA

DESENVOLVIMENTO DE UM SISTEMA SUPERVISÓRIO PARA MONITORAMENTO DO CONSUMO DE ÁGUA

Trabalho apresentado como requisito parcial para a Conclusão do Curso de Engenharia Elétrica da Universidade Federal de São João Del Rei.

COMISSÃO EXAMINADORA

_____

Prof. Me. Davidson Lafitte Firmo
Universidade Federal de São João Del Rei

_____

Prof. Dr. Samir Ângelo Milani Martins
Universidade Federal de São João Del Rei

_____

Prof. Dr. Marco Túlio Alves Evo
Universidade Federal de São João Del Rei

São João Del Rei, 09 de março de 2023

OLIVEIRA, Artur Miranda. **Desenvolvimento de um sistema supervisório para monitoramento do consumo de água.** Trabalho Final de Curso. Curso de Engenharia Elétrica. Universidade Federal de São João Del Rei, 2022.

RESUMO

Com os avanços na eletrônica e na tecnologia de Internet das Coisas, a disponibilidade de microcontroladores e sensores a baixo custo aumentou sensivelmente. Este trabalho, diante desse cenário, propõe o desenvolvimento de um sistema supervisório para monitoramento do consumo de água, focado em aplicações comerciais e domésticas. O desenvolvimento deste produto permite o acompanhamento dos gastos de um dos recursos mais importantes e em risco de escassez: a água. O sistema desenvolvido utiliza um microcontrolador ESP8266 de baixo custo e tem uma interface web amigável ao usuário, onde todos os dados podem ser remotamente acessados e analisados graficamente.

Palavras-chave: Sistema supervisório; Monitoramento de Água; ESP8266; Sensor; IoT.

OLIVEIRA, Artur Miranda. ***Supervisory System Development for Monitoring Water Consumption****. Monograph for the Bachelor Degree in Electrical Engineering. Federal University São João Del Rei, 2022.*

ABSTRACT

With the advancements of electronics and internet of things technology, the availability of low-cost microcontrollers and sensors has increased significantly. Given this scenario, this work proposes the development of a supervisory system for monitoring water consumption focused on commercial and domestic applications. The development of this product makes it possible to monitor the consumption of one of the most important resources at risk of scarcity: water. The developed system uses a low-cost ESP8266 microcontroller and has a user-friendly web interface, where all data can be remotely accessed and graphically analysed.

Key Words: *Supervisory system, Water monitoring, ESP8266, Sensor, IoT.*

Lista de Ilustrações

Lista de Tabelas

Lista de siglas

API - *Application Programming Interface*

ASCII - *American Standard Code for Information Interchange*

COMPESA - Companhia Pernambucana de Saneamento

CSS – *Cascading Style Sheets*

GMT – *Greenwich Mean Time Zone*

GPIO - G*eneral Purpose Input Output*

HTML – *HyperText Markup Language*

HTTP - *Hypertext Transfer Protocol*

I2C - *Inter-Integrated Circuit*

I2S - *Inter-IC Sound*

ICMP – *Internet Control Message Protocol*

IEEE – *Institute of Electrical and Electronics Engineers*

IoT – *Internet of Things*

IP – *Internet Protocol*

JSON – *JavaScript Object Notation*

M2M – *Machine to Machine*

MQTT - *Message Queuing Telemetry Transport*

OSI – *Open Systems Interconnection*

PHP – *Hypertext Preprocessor*

PLC - *Programmable Logic Controller*

PWM – *Pulse Width Modulation*

RAM – *Rapid Access Memory*

RFC – *Request For Comments*

RTU – *Remote Terminal Unit*

SCADA - *Supervisory Control and Data Acquisition*

SDK – *Software Development Kit*

SEEG/OC - Sistema de Estimativas de Emissões de Gases de Efeito Estufa do Observatório do Clima

SGBD – Sistema de Gerenciamento de Banco de Dados

SNIS **–** Sistema Nacional de Informações sobre o Saneamento

SPI – *Serial Peripheral Interface*

TCP - *Transmission Control Protocol*

TI – Tecnologia da Informação

UART - *Universal Asynchronous Receiver / Transmitter*

UDP – *User Datagram Protocol*

URL - *Uniform Resource Locator*

USART - *Universal Synchronous/Asynchronous Receiver/Transmitter*

USB - *Universal Serial Bus*

Wi-Fi – *Wireless Fidelity*

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação

Devido a problemas climáticos, como o aquecimento global, o crescente número de secas e o aumento populacional, muitas pessoas estão vivendo atualmente sob estresse hídrico. Segundo o Relatório Mundial das Nações Unidas sobre Desenvolvimento Dos Recursos Hídricos 2021: O Valor da Água: Fatos e Dados (UNESCO WORLD WATER ASSESSMENT PROGAMME, 2021), em 2018, mais de 2 bilhões de pessoas em todo mundo viviam em situação de estresse hídrico. Já Hoekstra et al. (HOEKSTRA e MEKONNEN, 2016) afirmam em seu estudo, que aproximadamente 4 bilhões de pessoas vivem em áreas que sofrem grave escassez física de água por pelo menos um mês do ano. Vários dos principais aquíferos do mundo estão sob estresse hídrico crescente, e, 30% dos maiores sistemas de água subterrânea estão se esgotando (RICHEY, THOMAS, *et al.*, 2015), sendo a captação para irrigação um dos principais fatores (GREVE, KAHIL, *et al.*, 2018).

No Brasil não é diferente, cada vez mais o país se vê diante de problemas hídricos. Segundo dados do Sistema de Estimativas de Emissões de Gases de Efeito Estufa do Observatório do Clima (SEEG/OC), o Brasil perdeu nos últimos 30 anos 15,7% de sua superfície de água, o que representa 3,1 milhões de hectares. Em 1991, a área hídrica nacional era de quase 20 milhões de hectares. Em 2020, essa área caiu para 16,6 milhões (GIFE, 2021). Um levantamento feito pelo Plano Nacional de Segurança Hídrica revelou que até 2035 cerca de 74 milhões de brasileiros estarão sob algum grau de insegurança hídrica (PNSH, 2023).

Para enfrentar estes problemas anteriormente citados, várias políticas, investimentos e estudos são realizados. Exemplos são: aperfeiçoamentos nos reservatórios, políticas contra o aquecimento global, fomento ao desenvolvimento sustentável, conscientização da população, racionamento de água, redução de desperdício, entre outras. Este trabalho foca na redução do desperdício no consumo de água, propondo um sistema de monitoramento de consumo de água para instalações domésticas e comerciais. Com o monitoramento, cidadãos em suas residências, produtores rurais, estabelecimentos comerciais, shoppings e clubes teriam muito mais controle do quanto se gasta de água, o que reduziria o desperdício, que pode se dar por uso não consciente de água, por perdas em ligações ou pela simples falta de controle. Numa escala mais ampla, um sistema distribuído de monitoramento remoto de água, comparando dados vazão de entrada e de saída, poderia identificar com muito mais facilidade as perdas em um sistema de distribuição como da COMPESA (JÚNIOR, 2017). Para se ter uma ideia,

segundo dados do SNIS, o município de Belo Horizonte tem uma perda de 466,83 litros/ligação/dia e um total de 43.07% de toda água captada nos reservatórios é desperdiçada (SNIS, 2023).

Para o desenvolvimento do sistema de monitoramento, seria ideal acompanhar os dados de consumo de água e estados gerais do sistema em tempo real além de acessá-los remotamente. Para um usuário doméstico ou comercial, uma aplicação Web satisfaz bastante bem essa necessidade, na qual o usuário através de algum cliente, seja ele Desktop ou Mobile, acompanhe o sistema ao longo do tempo. Há a possibilidade ainda de rotinas que informem estados específicos, como perda e consumo além de um certo limiar.

Devido aos avanços na área da eletrônica e Internet das Coisas, existem atualmente vários dispositivos que permitem implementar o conceito explicado no parágrafo anterior e a um custo acessível. Kits de desenvolvimento, microcontroladores e sensores são exemplos disso.

Sendo assim, o presente trabalho propõe o uso de microcomputadores e microcontroladores, com o objetivo de desenvolver um sistema de monitoramento gasto de água em uma residência ou instalação comercial.

1.2 Delimitação do tema

O escopo do trabalho se restringe ao desenvolvimento do sistema supervisório de água, abrangendo desde a montagem do circuito de medição ao desenvolvimento de uma página Web, onde os dados de interesse do sistema são visualizados graficamente.

O sistema consiste em três dispositivos chaves: o sensor de vazão, o microcontrolador e um hospedeiro final, seja um computador ou um smartphone. No trabalho foi utilizado um microcontrolador ESP8266.

Este microcontrolador pode ser encontrado no mercado em formato de placa para desenvolvimento, na qual diversas portas GPIOs (*General Purpose Input Output*), alimentação, conectores USB, já vem implementados. A figura 1 ilustra uma placa ESP8266 NodeMCU v1 (WIKIMEDIA, 2023).

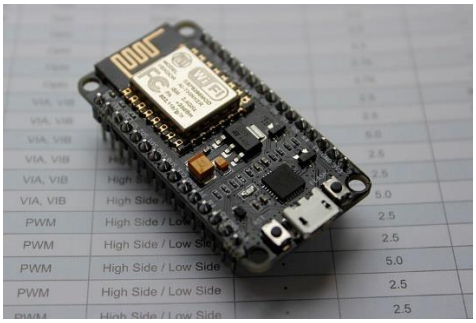


Figura : 1 - ESP8266

Este microcontrolador foi adotado neste projeto devido ao baixo custo, por volta de 25 reais na Internet, e atendimento às especificações de desempenho. O ESP82866 tem entradas suficientes, Wi-Fi nativo e um poder de processamento suficiente para a aplicação desenvolvida. O dispositivo conta com um processador L106 32-bit RISC, que atinge uma frequência máxima de Clock de 160Mhz (ESPRESSIF, 2023).

Como em todo sistema onde há múltiplos dispositivos, deve haver integração entre eles para permitir a troca de informações. O sensor e o microcontrolador são conectados fisicamente, por meio de algum circuito físico de medição. Quanto ao microcontrolador e ao hospedeiro final, a conexão, no contexto de Internet das Coisas, é feita comumente pelo protocolo IEEE 802.11, também conhecido como Wi-Fi, além de processos intermediários. O Wi-Fi abrange tanto a camada de enlace quanto a camada física, camadas essas definidas pelo modelo OSI (KUROSE e ROSS, 2013). Entretanto, como muitos dados do sensor são enviados pelo microcontrolador e há o interesse de se manter um histórico e a visualização destes, um servidor Web e um banco de dados são necessários.

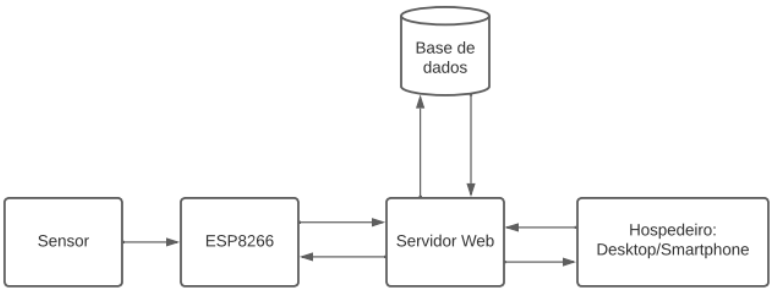


Figura : 2 - Arquitetura do sistema

Na figura 2 (AUTOR), é mostrada a arquitetura do sistema desenvolvido neste trabalho. Primeiramente, o sensor, conectado ao microcontrolador, transmite algum sinal elétrico que é interpretado pelo ESP8266, que por sua vez transmite os dados de vazão ao servidor, que os grava no banco de dados. O hospedeiro através de uma requisição ao servidor Web solicita estes dados. O procedimento completo em detalhes será descrito ao longo do trabalho.

1.3 Problemas e premissas

O sistema deve monitorar o consumo de água continuamente e proporcionar ao usuário um modo de ver esses dados de maneira clara e objetiva, onde conclusões acerca do consumo possam ser inferidas.

Para satisfazer tal premissa, há alguns problemas a serem considerados. Os problemas e desafios do projeto englobam a escolha adequada do sensor, a comunicação entre os dispositivos, como armazenar os dados de medição e como desenvolver uma interface intuitiva para visualização de informações.

O sensor escolhido deve ser de baixo valor e preciso o suficiente para a realização da medição.

Demais questões devem ser solucionadas:

- O microcontrolador deve se comunicar com o servidor Web, como seria feita essa comunicação?
- Qual o protocolo deve ser utilizado para a comunicação?
- O servidor Web deve ser local ou rodado em serviços de hospedagem?
- O armazenamento dos dados deve ser feito de maneira que proporcione a recuperação efetiva dos dados e uma persistência confiável, como seria feito esse armazenamento?
- Qual tipo de aplicação seria melhor para tal, um aplicativo mobile, uma página Web ou um código rodado no hospedeiro final?

1.4 Objetivos

1.4.1 Objetivos Gerais

Desenvolver um sistema supervisório para monitoramento do consumo de água em uma instalação comercial ou residencial.

1.4.2 Objetivos Específicos

1 - Realizar a revisão bibliográfica sobre sistemas supervisórios conectados;
2 - Pesquisa sobre métodos de medição de fluxo de água;
3 - Pesquisa sobre os protocolos de comunicação mais utilizados no contexto de Internet das Coisas e na Internet;
4 - Pesquisa sobre bancos de dados;
5 - Pesquisa sobre aplicações cliente;
6 – Montagem do protótipo;
7 – Análise dos resultados.

1.5 Justificativa

O sistema supervisório almeja, de alguma forma, contribuir para o desenvolvimento de soluções que visem o consumo consciente de água, haja vista a situação hídrica que o mundo se encontra. Acredita-se que a escassez de água pode ser enfrentada através de soluções micro e o sistema supervisório foca nisso.

2 CONCEITOS PRELIMINARES

Nesta seção são abordados os conceitos mais importantes relacionados ao tema de trabalho, além de um breve levantamento bibliográfico.

2.1 Sistema supervisório

Segundo Boyer (BOYER, 2004), um sistema supervisório é uma tecnologia que possibilita ao usuário coletar dados de uma ou mais plantas e enviar instruções de controle. O supervisório tem a função de tornar desnecessária a inspeção local de medidores de grandezas físicas. Um sistema supervisório inclui uma interface com o operador e fornece meios para a manipulação de dados relacionados à aplicação. Um sistema supervisório também é comumente chamado de SCADA, um acrônimo para "*supervisory control and data aquisition".*

O termo SCADA, segundo McCrady (MCCRADY, 2013), foi primeiramente utilizado no final dos anos 1980, mas não se tornou popular até o início dos anos 1990. Os primeiros

sistemas supervisórios datam da década de 1960, mas eram pouco integrados e confinados a uma única planta, ou aplicação.

Geralmente, o SCADA inclui um processamento local, máquinas, equipamentos de monitoramento local, PLC, instrumentos, sensores, unidades terminais remotas (RTU), dispositivos eletrônicos inteligentes e um computador com interface humana. A figura 3, adaptada de (What is SCADA?, 2023) mostra uma visão geral da estrutura de um sistema supervisório.

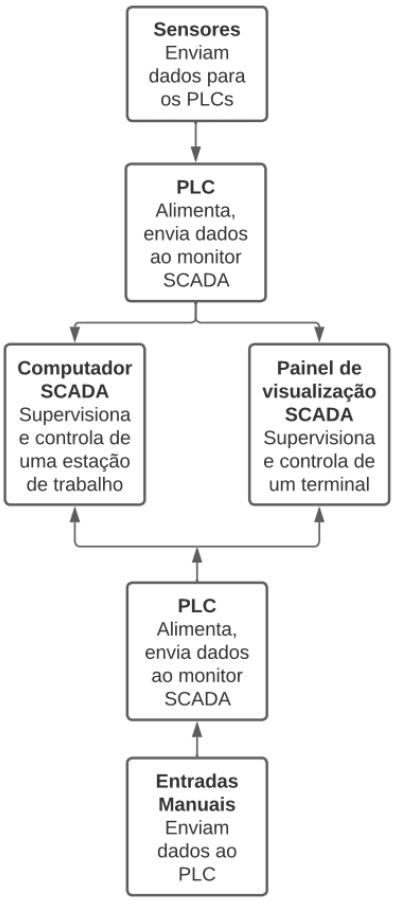


Figura : 3 - SCADA

Em consonância com o conceito de sistemas supervisórios, e mencionando sua ampla utilização ao longo das últimas décadas em plantas e aplicações industriais, houve, ao longo dos últimos anos, o desenvolvimento de várias aplicações Open-Source e ferramentas para sistemas supervisórios no contexto de Internet das Coisas. O Emoncms (Emoncms, 2023), um Web-app de código aberto, por exemplo, é uma solução análoga ao problema tratado neste trabalho. Essa aplicação Web permite o monitoramento de energia elétrica, disponibilizando

logs e dados relacionados à energia. A aplicação permite ao usuário configurar dados, linhas temporais, visualizar gráficos e dashboards. Na aplicação também há uso de diversos conceitos e ferramentas que serviram de base para o desenvolvimento do sistema supervisório deste trabalho. Dentre os principais conceitos e tecnologias base estão: CSS, HTML, API, HTTP. Além do Emoncms há diversas outras ferramentas, como o node red, que foca em uma rede de dispositivos utilizando o protocolo MQTT, o IBM Watson e o Mostquitto.

2.2 Internet das Coisas

Com os avanços na área de redes de computadores e de sensores, a Internet das Coisas foi identificada como uma das tecnologias emergentes na área de TI. Internet das Coisas é uma tecnologia que proporciona a criação de uma rede de dispositivos e máquinas, que através desta possam se comunicar. O maior diferencial da tecnologia é a possibilidade que os dispositivos têm para interagir uns com os outros e, além disso, interagir com aplicações mais complexas do lado da Internet, como sistemas de gerenciamento de dados e aplicações finais em geral. Exemplos de aplicações seriam as casas inteligentes, bairros inteligentes, carros inteligentes (LEE e LEE, 2015), segurança e identificação de pessoas, manutenção preditiva, controle de almoxarifado em indústrias, monitoramento de pacientes, shopping inteligentes, etc.

Apesar de promissora, a tecnologia ainda é emergente, tem-se uma série de desafios, como o volume crescente de dados - consequência de uma larga adoção da tecnologia -, privacidade, segurança e padronização (WORTMANN e FLÜCTHER, 2015).

Os dispositivos capazes de interagir com o ambiente de Internet das Coisas são chamados de dispositivos inteligentes. Dispositivos inteligentes são capazes de realizar comunicações e cálculos computacionais. Algumas características importantes desses dispositivos são a autonomia, conectividade, interação com o usuário, mobilidade e armazenamento de dados (SILVERIO-FERNÁNDEZ, RENUKAPPA e SURESH, 2018).

A estrutura básica para a IoT é composta por 5 camadas, como ilustrado na figura 4 adaptada de Khan et al. (KHAN, KHAN, *et al.*, 2012). A camada mais abaixo na pilha é a *Perception Layer*, ela consiste nos equipamentos físicos e nos sensores. São exemplos dessa camada o ESP8266 e o sensor de fluxo. As informações coletadas por esses dispositivos são então passadas para a rede, *Network Layer*, que no contexto de Internet das Coisas são os protocolos de rede, como o IP. Os dados dessa camada são então passados para a *Middleware Layer*, que consiste no gerenciamento de serviços, processamentos de informações, banco de dados, etc. A camada de aplicação por sua vez tem a tarefa de através de algum protocolo,

extrair dados da *Middleware Layer,* como por exemplo de um banco de dados. Aplicações podem ser desenvolvidas em várias linguagens de programação e atendem às demandas de cada usuário. Demandas essas que são definidas pela *Business Layer.* Nesta última camada tem-se a fotografia final do sistema de Internet das Coisas, suprindo as necessidades do usuário final, recuperando todos os dados de interesse dos sensores e dos dispositivos inteligentes.

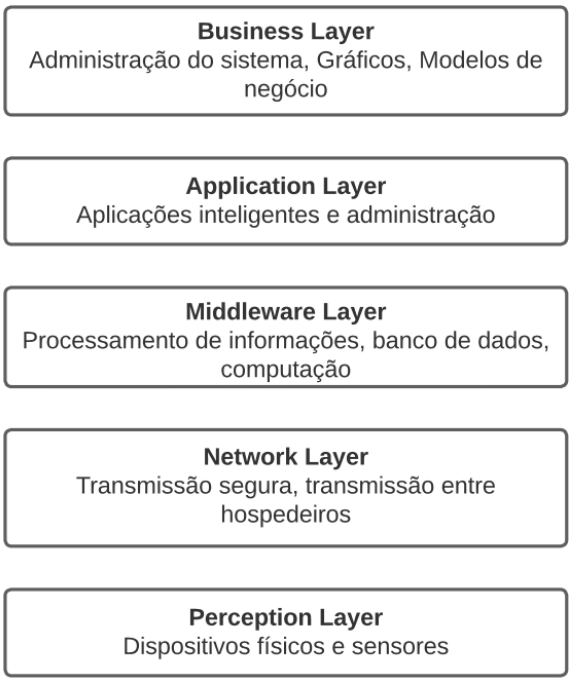


Figura : 4 - Estrutura IoT

Cada uma dessas camadas, excetuando-se a *Business Layer,* tem seu próprio protocolo de comunicação. Protocolos amplamente utilizados no ambiente de Internet das Coisas normalmente se baseiam na pilha do TCP/IP (VITORIANO, 2022). Na camada superior ao TCP/IP, a de aplicação, protocolos comuns são o HTTP (*Hypertext Transfer Protocol*) e o MQTT (*Message Queuing Telemetry Transport*). O MQTT é um protocolo bem leve aplicável a redes com largura de banda limitada, principalmente utilizado quando um ou mais dispositivos inteligentes devem se comunicar. Já o HTTP é um protocolo da camada de aplicação para modelos colaborativos, distribuídos e sistemas de informação de hipermídia (T. BERNERS-LEE, 1996). Hipermídia, seria a reunião de várias mídias num ambiente computacional, incluindo textos, sons, animações e vídeos (BUGAY e ULBRICHT, 2000). O HTTP é a base para comunicação de dados na Web. Ao aproximar-se das camadas de mais

baixos níveis, o protocolo usado irá depender do hardware e dos enlaces físicos empregados. Exemplos de protocolos mais baixo nível são o Wi-Fi, Ethernet, I2C, UART e vários outros.

2.3 Protocolos de comunicação

Protocolos de comunicação definem o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou recebimento de uma mensagem ou outro evento (KUROSE e ROSS, 2013).

A pilha de protocolos da Internet é formada por cinco camadas: a camada física, de enlace, de rede, de transporte e de aplicação, como mostrado na figura 5 (KUROSE e ROSS, 2013).

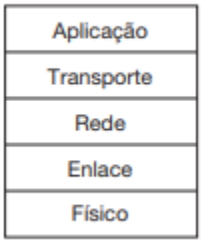


Figura : 5 - Pilha de camadas de protocolos na Internet

2.3.1 Camada de aplicação

A camada de aplicação é onde residem as aplicações de rede, como a Web, e seus protocolos. No contexto de Internet das Coisas, como mencionado anteriormente, os dois protocolos mais populares dessa camada são o HTTP e o MQTT. Um protocolo da camada de aplicação define como processos de aplicações que funcionam em sistemas diferentes, por exemplo, num microcontrolador ESP8266 e num servidor, irão passar as mensagens entre si. Um protocolo da camada de aplicação define os tipos de mensagens trocadas, como de requisição ou de resposta, os campos de mensagem, o significado dos campos e regras para determinar como e quando um processo envia as mensagens.

O HTTP especificado no RFC 1945 (NETWORK WORKING GROUP, 1996), define dois tipos de mensagem, a de requisição e a de resposta. Uma mensagem de requisição HTTP é escrita em ASCII, onde a primeira linha é a linha de requisição, as linhas subsequentes são as linhas de cabeçalho e, por fim, tem-se os dados (no caso de uso do método POST). A linha

de requisição contém o método, a URL e a versão do HTTP. Há alguns métodos disponíveis, como o método POST, utilizado como o nome sugere para enviar, escrever algum dado a ser persistido ou interpretado no lado servidor, e o método GET, utilizado quando a aplicação cliente quer apenas ler os dados vindos de algum hospedeiro. Linhas de cabeçalho definem instruções de fechar ou não uma conexão, a versão do navegador, idioma, tipo de dados enviados etc. Já a mensagem de resposta HTTP, contém três seções. São elas: linha de estado, linhas de cabeçalho e o corpo da entidade, que contém os dados solicitados pela mensagem de requisição. A linha de estado informa a versão do HTTP, um código de estado e uma mensagem do estado correspondente. Um exemplo de código de estado é o 200 (OK), que significa que o servidor encontrou e enviará o objeto solicitado.

O HTTP, por ser parte quase que intrínseca da Internet, é ideal para aplicações Web, tanto que o primeiro endereço de página Web construído, The World Wide Web project (cern.ch) (Word Wide Web, 1991) , utilizava este protocolo. Entretanto, o HTTP foi construído com o intuito de solicitar páginas de texto, numa arquitetura Request-Response. Em algumas aplicações de Internet das Coisas, com mais dispositivos interessados nos dados que o outro produz, é interessante ter uma abordagem Publisher-Subscribe. Esse é um dos motivos para a criação do MQTT, um protocolo M2M (Machine to Machine) com foco em IoT construído em cima do TCP/IP.

Um sistema MQTT se baseia na comunicação Publisher-Subscriber, na qual os dispositivos inteligentes podem postar e receber informações, enquanto o servidor, chamado de broker, administra os dados. Na arquitetura Publisher-Subscriber tem-se três elementos chaves: o publisher, o subscriber e o broker. No MQTT cada subscriber ou publisher se comunica diretamente com o broker, através de mensagens simples e não restritas a mensagens de texto. Um broker faz o intermédio entre os dispositivos, administrando os dados e repassando aos dispositivos interessados. Um exemplo de arquitetura MQTT pode ser vista na Figura 6 (Observer vs Pub-Sub pattern, 2017). Nesta figura um publisher publica nos tópicos A e B, administrados pelo broker. O subscriber 2, como está inscrito nos dois tópicos, receberá informação de ambos. O broker provê serviços como a entrega de dados aos inscritos no tópico quando há algum dado novo, controle de conexão, message queueing, etc (MQTT, 2019).

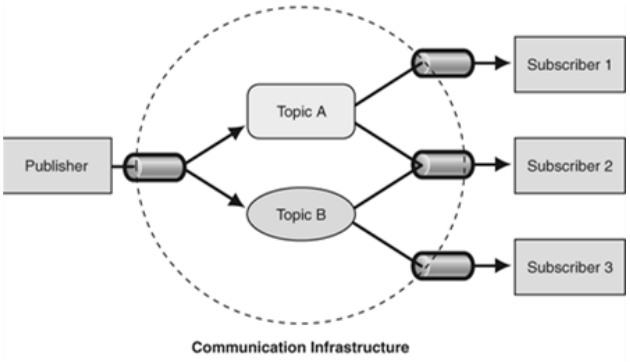


Figura : 6 - Estrutura do MQTT

Resumidos os dois principais protocolos da camada de aplicação utilizados no ambiente IoT, cabe ao projetista determinar qual atende melhor às suas demandas. O protocolo MQTT tem ganhado espaço, entretanto, deve-se avaliar as ferramentas disponíveis e serviços compatíveis com a tecnologia.

2.3.2 Camada de transporte

Segundo Ross et al. (KUROSE e ROSS, 2013), a camada de transporte fornece a comunicação lógica entre processos de aplicação que estão sendo executados em hospedeiros diferentes. É importante ressaltar que a camada de transporte está limitada à comunicação entre processos, um protocolo de rede, por outro lado, fornece a comunicação lógica entre os hospedeiros. Ou seja, a rede é encarregada de fornecer o roteamento até o endereço final do hospedeiro, enquanto que a camada de transporte, quando em posse do segmento, indica qual aplicação deve receber os dados ou encapsulá-los corretamente quando no papel de remetente. A comunicação lógica entre processos permite que os processos possam enviar mensagens sem a preocupação dos detalhes da infraestrutura física utilizada para transportá-las. Os protocolos da camada de transporte são implementados nos sistemas finais, como um Desktop, um servidor, um microcontrolador.

No hospedeiro remetente, a camada de transporte encapsula a mensagem da camada de aplicação, como requisições HTTP, em um segmento, e adiciona informações como porta de origem, porta de destino e no caso do TCP, número de sequência, checksum e outros mais. Este segmento é entregue à camada de rede que se encarregará de levar o segmento ao outro hospedeiro. No lado destinatário, a camada de rede extrai do datagrama o segmento da camada de transporte, que por sua vez irá se encarregar de enviar o campo de dados à aplicação correta.

Um protocolo de transporte pode oferecer alguns serviços, como a entrega confiável de dados, verificação de erros e controle de congestionamento. Na Internet há dois protocolos distintos para a camada de transporte: o UDP e o TCP. O UDP é mais simples, não orientado à conexão, não provê controle de congestionamento e não possui entrega confiável de dados. O TCP, por sua vez, garante a entrega confiável de dados, controle de congestionamento e é orientado à conexão. Segue na tabela adaptada de Ross et al. (KUROSE e ROSS, 2013) abaixo alguns protocolos de camada de aplicação e seus protocolos de transporte subjacentes.

Tabela 1- Protocolos da camada de transporte e seus protocolos de transporte subjacentes

Aplicação	**Protocolo de camada de aplicação**	**Protocolo de transporte subjacente**
E-mail	SMTP	TCP
Acesso terminal remoto	Telnet	TCP
Web	HTTP	TCP
Transferência de arquivos	FTP	TCP
Multimídia (ex. YouTube)	HTTP	TCP
Telefonia por Internet	Skype, SIP, RTP	UDP ou TCP

2.3.3 Camada de rede

A camada de rede vem logo abaixo da camada de transporte e, como brevemente mencionado, tem a função de estabelecer a comunicação entre os hospedeiros. Na Internet, hospedeiros finais estão conectados entre si por enlaces de comunicação e comutadores de pacotes. A sequência de enlaces e comutadores de pacotes que um pacote percorre é denominada rota ou caminho. O conjunto desses caminhos é a rede.

Para fazer a comunicação entre dois hospedeiros, duas funções são de extrema importância: o repasse e o roteamento. O repasse ocorre quando um pacote ao chegar no enlace de entrada de um roteador, este deve o encaminhar ao enlace de saída adequado. O roteador sabe para qual enlace enviar cada pacote pelas tabelas de roteamento presentes em cada roteador. Estas tabelas, por sua vez, são determinadas pelos algoritmos de roteamento, que determinam o caminho ótimo a ser seguido para ligar dois sistemas finais.

A estrutura da camada de rede da Internet se baseia em três protocolos chaves: os

protocolos de roteamento, o protocolo IP e o ICMP (Internet Control Message Protocol). Os protocolos de roteamento definem o melhor caminho a ser tomado para entregar pacotes entre dois hospedeiros; o IP define convenções de endereçamento, formato dos pacotes (datagramas) e convenções de manuseio de pacotes (os campos indicam como tratar um pacote); o ICMP é utilizado por hospedeiros e roteadores para trocar dados da camada de rede em si, um exemplo é a comunicação de erros.

Todo hospedeiro conectado à rede deve possuir um endereço IP, seja este configurado de maneira estática ou dinamicamente e de maneira automática pelo DHCP. Endereço IP é um dos muitos campos presentes em um datagrama IP, que contém campos de tempo de vida, soma de verificação, comprimento do datagrama, identificador, versão, comprimento do cabeçalho, protocolo da camada de transporte, etc. Atualmente há duas versões em uso do IP, o IPv4 e o IPv6, que suporta um número maior de endereços - de 32 bits do IPv4 para 128 bits no IPv6 - e alterou alguns campos e funcionalidades do IPv4, como a remoção do campo de opções presente no IPv4.

2.3.4 Camada de enlace e física

As camadas de enlace e física são as camadas mais abaixo na pilha de protocolos. A camada de enlace tem a função de repassar os quadros vindos da camada de rede para a camada física, que é incumbida de transportar os bits individuais que estão dentro de um quadro, os convertendo em sinais elétricos, eletromagnéticos que são passados por um meio físico como fios trançados de cobre, o ar e fibra ótica.

A camada de enlace é onde o software encontra o hardware. Esta camada é normalmente implementada em um adaptador de rede. No núcleo do adaptador está o controlador da camada de enlace, que executa diversos serviços, como o enquadramento, acesso ao enlace, detecção de erros e entrega confiável. Apesar da maior parte da camada de enlace ser implementada em hardware, funções como o endereçamento da camada de enlace e a ativação do hardware são executadas por software.

O Wi-Fi é um exemplo de protocolo da camada de enlace amplamente utilizado no ambiente de Internet das Coisas. O Wi-Fi, por ser um protocolo sem fio, é necessariamente mais complexo que protocolos em redes com fio, como o Ethernet. Enlaces sem fios estão sujeitos a:

- Reduções da força de sinal, pois radiações eletromagnéticas são enfraquecidas quando

passam por diversos materiais como o ar, paredes e até o corpo humano;

- Interferência de outras fontes como telefones sem fio, motores e outros equipamentos Wi-Fi;
- Propagação multivias, quando ondas percorrem caminhos de comprimentos diferentes e, portanto, chegam ao receptor em tempos diferentes.

Estes fatores tornam o Wi-Fi muito mais propenso a erros que enlaces cabeados, logo, algoritmos anti colisões são essenciais. Arquiteturas que fazem uso do Wi-Fi normalmente são redes com salto único entre o hospedeiro e o access point - equipamento que recebe sinais vindos dos hospedeiros e que comunica diretamente com um roteador, normalmente através de conexões cabeadas – e do access poit à infraestrutura de rede. A infraestrutura de rede seria a rede maior com a qual o hospedeiro quer estar inserido.

As faixas de frequência que a camada física, adjacente ao Wi-Fi, opera, em suas versões 802.11b, 802.11a, 802.11g são entre 2.4-2.485GHz, 5.1-5.8GHz e 2.4-2.485GHz, respectivamente. As versões 802.11g e 802.11a têm taxas de transmissão de até 54mbps/s e a versão 802.11b até 11mbps/s.

2.4 Desenvolvimento Web

A Web é a parte da Internet que consiste de servidores que armazenam documentos eletrônicos chamados de *webpages* (CAMPBELL e SHELLY, 2017). Uma *webpage* é um documento formatado de maneira especial que contém imagens, texto, elementos interativos e hyperlinks. Páginas web atualmente são desenvolvidas utilizando linguagem de marcação de hipertexto, o HTML, linguagens client side quanto de server side, fazem uso de banco de dados e muitos outros elementos. Elementos de desenvolvimento Web utilizados neste trabalho são:

- HTML, define a estrutura textual da página, como títulos, parágrafos, links e imagens;
- CSS, define o estilo da página, como cores dos elementos, elementos gráficos, fontes, etc;
- JavaScript, linguagem rodada no lado cliente que permite programar o comportamento de uma página Web, promovendo interatividade, ferramentas funcionais à aplicação, o uso de APIs, etc. (obs: Pode ser implementada no lado servidor, mas neste trabalho foi utilizada apenas no lado cliente.);
- PHP, linguagem server side utilizada em muitos dos servidores comerciais da Internet.

Aproximadamente 77.7% dos servidores atuais rodam em PHP (Usage statistics of PHP for websites, 2023). A Hostinger, serviço de hospedagem utilizado neste trabalho, roda em PHP. A linguagem foi utilizada neste trabalho para fazer as requisições HTTP ao servidor comercial, acessar o banco de dados e na própria página Web;

- Sistemas de gerenciamento de banco de dados, são sistemas onde são feitas a persistência de dados. As vantagens de se utilizar um SGBD são diversas, como o controle de redundância, facilidade de acesso e recuperação de dados, restrição de acesso não autorizado e backup.

2.5 ESP8266

O ESP8266 é um microcontrolador produzido pela Espressif com Wi-Fi incorporado. A popularização do microcontrolador se deu em 2014, após a comercialização de um módulo customizado pela empresa Ai -Thinker. Logo depois, a popularidade aumentou ainda mais com a junção desses módulos em placas com reguladores de tensão, interface usb e leds. Placas modernas desse tipo, como a NodeMCU, são mais fáceis de se trabalhar. Outro ponto que ajudou a popularizar o uso do dispositivo é o custo, bem abaixo de outros microcontroladores e dispositivos IoT como o Raspberry. Por último, o controlador ainda conta com muitas SDKs open-source.

O microcontrolador conta com 17 interfaces GPIO que podem ser configuradas como entradas ou saídas digitais. Possui quatro interfaces de saída PWM e uma entrada analógica com 10 bits de precisão. Há interfaces de comunicação seriais SPI, I2C e I2S e assíncronas como a USART.

Quanto ao desempenho do controlador, o ESP8266 conta com um microcontrolador de 32 bits operando a uma frequência máxima de 160 MHz. O processamento da pilha de protocolos Wi-Fi consome cerca de 20% de seu processamento. A memória disponível para o programa principal é de 4MB (OLIVEIRA, 2017).

2.6 Medição de consumo de água

Segundo a Portaria n.º 155 do INMETRO (INMETRO, 2022), o medidor de água potável é definido como um instrumento destinado a medir continuamente, memorizar e exibir o volume que escoa através do transdutor de medição, sob condições de medição.

Quanto ao princípio de funcionamento, os hidrômetros normalmente utilizados são

classificados em volumétricos ou de velocidade. Os hidrômetros de velocidade funcionam com o fluxo de água movendo uma hélice ou turbina. O número de rotações é proporcional ao volume de água escoado. Nos hidrômetros volumétricos, há um êmbolo ou anel e uma câmara de medida, que se enche e esvazia com a passagem do líquido. O transporte ocorre pela diferença de pressão entre a entrada e a saída do aparelho, que também faz o êmbolo girar em torno do seu eixo. Assim, a cada movimento do êmbolo é medido o volume que atravessa o hidrômetro (RECH, 1999).

Segundo a Portaria n.º 246 do INMETRO (INMETRO, 2000) e alterada pela Portaria n.º12 do INMETRO (INMETRO, 2011), atualmente em vigor, os hidrômetros comercializados no país são classificados em três categorias: A, B e C. Na Figura 7 (INMETRO, 2000) estão listadas as classes, vazões nominais, mínimas e máximas de cada categoria.

Classes Metrológicas		Vazão Nominal (m^3/h)									
		0,6	0,75	1,0	1,5	2,5	3,5	5,0	6,0	10,0	15,0
A	Q_{min} (m^3/h)	0,024	0,030	0,040	0,040	0,100	0,140	0,200	0,240	0,400	0,600
	Q_t (m^3/h)	0,060	0,075	0,100	0,150	0,250	0,350	0,500	0,600	1,000	1,500
B	Q_{min} (m^3/h)	0,012	0,015	0,020	0,030	0,050	0,070	0,100	0,120	0,200	0,300
	Q_t (m^3/h)	0,048	0,060	0,080	0,120	0,200	0,280	0,400	0,480	0,800	1,200
C	Q_{min} (m^3/h)	0,006	0,0075	0,010	0,015	0,025	0,035	0,050	0,060	0,100	0,150
	Q_t (m^3/h)	0,009	0,0110	0,015	0,0225	0,0375	0,0525	0,075	0,090	0,150	0,225

Figura : 7 - Classe dos Hidrômetros

Conforme a portaria, é exigida a determinação dos erros de indicação admitindo os erros máximos na indicação da vazão escoada (Q) pelos hidrômetros: (i) ± 5% entre Qmin (vazão mínima), inclusive, e Qt (vazão de transição), exclusive, e, (ii) ± 2% entre Qt, inclusive, e Qmax (vazão máxima), inclusive. No entanto, para verificações periódicas e eventuais de hidrômetros em uso, a portaria considera, na determinação da curva de erros, valores menos rigorosos para os erros máximos admissíveis: (i) ± 10% entre Qmin, inclusive, e Qt, exclusive, e, (ii) ± 5% entre Qt, inclusive, e Qmax.

Para a medição do consumo de água foi empregada neste trabalho uma solução análoga aos hidrômetros de velocidade, utilizando um sensor de fluxo que move uma hélice. A cada rotação, detectada por um sensor de efeito hall, a saída do sensor retorna um pulso em estado alto. A quantidade de rotações em um intervalo de tempo retornará o volume de água escoado. O consumo total de água escoado ao longo dos dias é calculado somando-se as leituras escritas no banco de dados.

O efeito hall consiste na diferença de potencial em um condutor, causada pela presença

de um campo magnético. Se um condutor é percorrido por uma corrente no sentido indicado na figura 8 (HALLIDAY, RESNICK e WALKER, 2016), os elétrons estarão se movendo no sentido oposto. Devido ao campo magnético, os elétrons irão se concentrar no lado direto da fita, o que produzirá um campo elétrico entre as extremidades longitudinais do condutor. Essa diferença de potencial pode ser medida e usada em sistemas digitais. No caso do sensor de fluxo de efeito hall, um ímã é colocado na extremidade de cada hélice, e, ao passar pelo sensor de efeito hall, uma tensão é medida, e, através do chaveamento algum transistor, convertida em pulsos elétricos com amplitude relacionada à tensão de alimentação do circuito do sensor (HALLIDAY, RESNICK e WALKER, 2016).

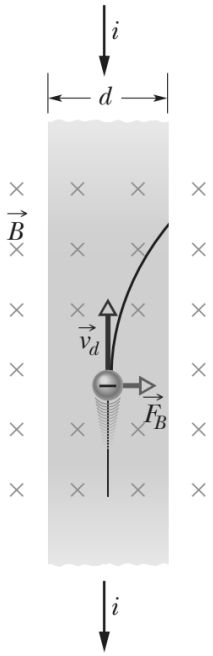


Figura : 8 - Efeito Hall

3 DESENVOLVIMENTO DO SISTEMA SUPERVISÓRIO

Nesta seção será abordada a metodologia de pesquisa e o detalhamento dos passos para o desenvolvimento do sistema supervisório. Este trabalho se trata de um experimento com resultados colhidos de 02/02/2023 a 06/02/2023 e a construção de um protótipo de sistema supervisório.

3.1 Escolha dos componentes e serviços

3.1.1 Microcontrolador

Para definir qual o microcontrolador a ser utilizado foi levado em consideração os seguintes fatores: preço, ferramentas disponíveis ou suporte, e, desempenho. As opções mais comuns de microcontroladores no mercado são o ESP32, o ESP8266, as versões do Arduino e o Raspberry. O ESP32 apresenta melhor desempenho de processamento que o ESP 8266, tem alguns diferenciais importantes como dois núcleos - interessantes quando se quer fazer paralelismo em aplicações - além de suporte embutido da tecnologia Bluetooth. O preço dos dispositivos varia pouco, em sites de marketplace o preço do ESP8266 se encontra em torno dos 22 reais e o do ESP32 em torno dos 33 reais. O Raspberry Pi, por sua vez, tem um preço acima de 500 reais nos marketplaces nacionais. O Raspberry seria algo sobre dimensionado para a aplicação em questão, conta com muito mais memória RAM, podendo chegar a 8GB no modelo pi 4, quatro núcleos de processamento, sistema Linux, etc. O microcontrolador escolhido, portanto, foi o ESP8266

3.1.2 Sistema de hospedagem e Banco de dados

Para o sistema de hospedagem, procurou-se alguns serviços disponíveis na Internet. Entre eles estão a Hostinger, HostGator, GoDaddy, Bluehost, etc. A Hostinger foi o sistema escolhido. O sistema conta com hospedagem de um domínio online e um banco de dados, útil no armazenamento dos dados do sistema supervisório. O banco de dados disponível no sistema é o MySql. O sistema trabalha com um servidor rodando em PHP.

A escolha de um servidor online se deve a praticidade e viabilidade. Seria mais custoso implementar um servidor dedicado local, sem diversos fatores de confiabilidade e suporte que um sistema de hospedagem pago oferece. Além disso, não se faz necessário o uso de um aparelho dedicado à função de servidor, facilitando a instalação do protótipo.

3.1.3 API gráfica

Para mostrar os dados do sistema ao usuário, pesquisou-se diversas bibliotecas que poderiam desempenhar este papel. Atualmente muitas aplicações fazem uso do JavaScript no front-end (lado cliente, como os navegadores) para atribuir funções às páginas web. Diversas APIs em JavaScript estão disponíveis hoje. Dentre as APIs, foi escolhida a plotly, capaz de fazer gráficos interativos e muito funcionais. A API é utilizada por diversas empresas como a

Blizzard, Tesla, T-Mobile entre outras.

O uso de uma API se faz interessante pela confiabilidade e elimina a necessidade de se “reinventar a roda”, gastando tempo em soluções que já estão amplamente consolidadas.

3.1.4 Sensor de Fluxo

O sensor de fluxo foi escolhido pelo desempenho e custo. O sensor escolhido, YF-S201, tem um custo em torno dos 30 reais e tem uma vazão reduzida se comparada com os valores de hidrômetros da tabela 2. Portanto, o uso do sensor de fluxo neste trabalho seria aplicável a um ponto hidráulico apenas, como torneira e chuveiros. Embora limitado, o procedimento seria análogo em caso de maior vazão. O datasheet do sensor (YF-S201 Datasheet) se encontra no ANEXO B.

3.1.5 Componentes de circuito

Quanto aos componentes de circuito foram utilizados uma placa de fenolite, resistores, jumpers, barra de pinos fêmea, estanho, uma power bank e o próprio sensor de fluxo. Todo o circuito foi colocado em uma caixa de MDF. Segue a imagem do protótipo montado na figura 9 (AUTOR). O sensor de fluxo tem o esquema de conexão mostrado na figura 10. Como pode-se observar, há uma resistência entre o terminal de sinal e de alimentação. Esse valor não se encontra no datasheet, entretanto, foi medido e o valor correspondente foi de aproximadamente 10 KOhms. Sendo a tensão máxima de entrada em um pino do ESP8266 de 3.3 V, foi necessário o uso de um divisor de tensão. Para reduzir a tensão de saída do ESP8266 (pino Vin) de cerca de 5 V, que alimenta o sensor e que será a tensão de saída do sinal, o pino amarelo foi conectado a dois resistores de 10 KOhms em série. O diagrama do circuito está na figura 11 (AUTOR).

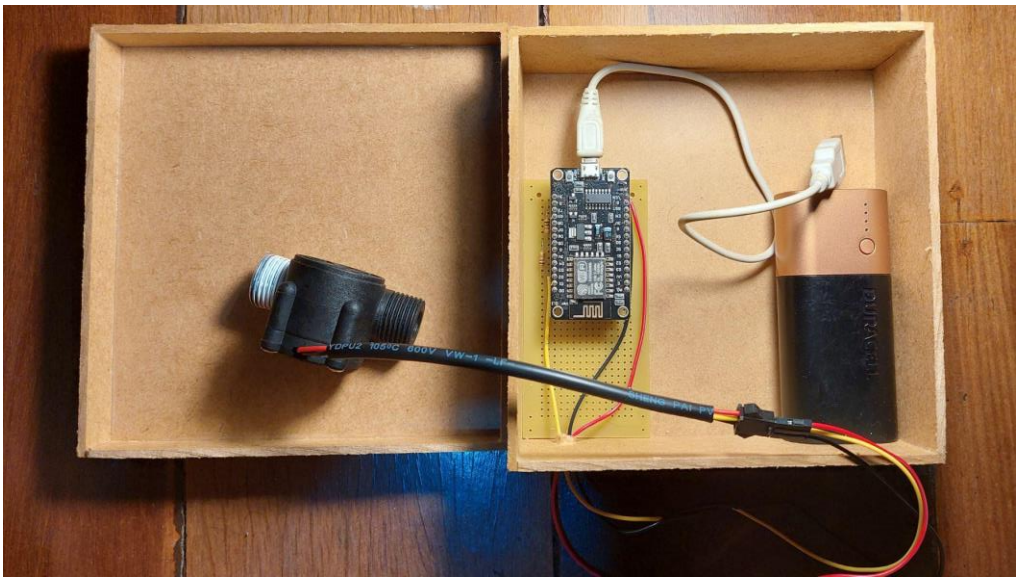


Figura : 9 - Protótipo do circuito de medição

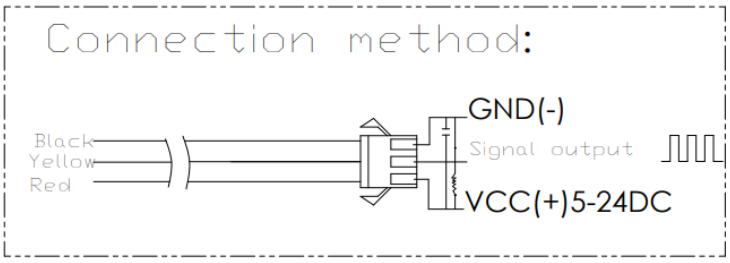


Figura : 10 - Esquema de conexão do sensor de fluxo

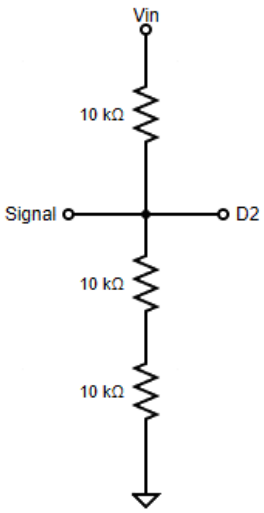


Figura : 11 - Diagrama do circuito de medição

3.1.6 Custo do protótipo

Segue na tabela abaixo os componentes utilizados e o custo em 05/01/2023:

Tabela 2- Custos do protótipo

Componente	Quantidade	Custo Unitário	
ESP8266	1	R$	27.45
Placa de fenolite	1	R$	6.90
Sensor de Fluxo YF-S201	1	R$	34.46
Resistores 10K	2	R$	0.40
Jumpers	3	R$	0.80
Caixa MDF	1	R$	4.85
Estanho	1	R$	22.90
Barra de pinos	1	R$	2.90
Serviço de hospedagem (anual)	1	R$	143.88
Cadastro de domínio	1	R$	5.94
Power Bank	1	R$	146.90
Total		R$	399.37

3.2 Desenvolvimento dos algoritmos de servidor, banco de dados, página Web e ESP8266

Neste trabalho, serão necessários quatro códigos: o código do servidor hospedado, o da página Web para a exibição dos resultados e os códigos do microcontrolador para postagem dos dados e calibração.

3.2.1 Algoritmo criação da tabela no MySql

O primeiro passo foi a criação do banco de dados e uma tabela na qual onde os dados foram armazenados. Foi criada uma tabela com o nome MonitoramentoAgua, na qual foram inseridos os campos id, sensor, localizacao, consumo e reading_time. O campo id fornece a chave primária para os dados na tabela, podendo ser recuperados inequivocamente. Os campos id e reading_time são calculados pelo próprio sistema de gerenciamento de banco de dados. Segue abaixo o código utilizado.

```
CREATE TABLE MonitoramentoAgua (
       id int(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
```

```
        sensor VARCHAR(30) NOT NULL,
        localizacao VARCHAR(30) NOT NULL,
        consumo VARCHAR(30),
      reading_time TIMESTAMP DEFAULT DATE_SUB(CURRENT_TIMESTAMP, INTERVAL 3 HOUR) ON
UPDATE DATE_SUB(CURRENT_TIMESTAMP, INTERVAL 3 HOUR)
)
```

3.2.2 Algoritmo do servidor

O servidor PHP, única linguagem server side disponível no serviço de hospedagem, tem a função de receber os dados enviados pelo ESP8266 e adicioná-los ao banco de dados. Há de se ressaltar que o PHP é uma linguagem fracamente tipada, portanto, não é necessário definir o tipo de dado que variável irá conter. O código começa definindo algumas variáveis para o funcionamento do servidor:

- servername: variável a qual é atribuída o nome do servidor. Segundo a Hostinger (HOSTINGER, 2021), “A Hostinger usa o “localhost” como o hostname para os seus servidores MySQL”;
- dbname e username: variáveis as quais são atribuídas o nome da base de dados e o nome de usuário. Estes dados podem ser encontrados no hpanel da plataforma. Segue na figura 12 estes dados;

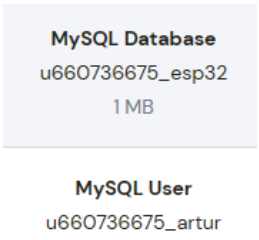


Figura : 12 - Banco de dados

- password: senha para acessar a base de dados;
- api_key: variável cujo valor coincide com a api_key definida no ESP8266. Os dados são gravados na base de dados apenas se os valores das duas chaves coincidirem;
- sensor: indica o sensor que fez a leitura do dado;
- localizacao: indica o local onde foi realizada a medição;
- consumo: consumo de água em um determinado tempo de amostragem, definido no

ESP8266.

A sintaxe da definição das variáveis é a seguinte:

```
$servername = "localhost";
$dbname = "u660736675_esp32";
$username = "u660736675_artur";
$password = "*********";
$api_key_value = "AXYLeagmiaYFo";
$api_key= $sensor = $localizacao = $consumo = "";
```

Após a definição das variáveis, foi feita a inserção no banco dos dados enviados através da requisição HTTP do ESP8266 ao servidor. A variável $SERVER é um array indexado contendo as informações do servidor e, no campo REQUEST_METHOD, tem-se o tipo de requisição HTTP. Como o ESP8266 publica essas informações, o método de requisição HTTP contido na sua mensagem enviada é do tipo POST. O conteúdo útil da requisição HTTP do ESP8266, que é um JSON, será armazenado no servidor no diretório php://input. O JSON é então convertido em um array PHP. Se a api_key do ESP8266 for igual a api_key definida no servidor, então os dados são gravados no banco de dados. A conexão com o banco de dados é feita criando-se um objeto da classe mysqli, o $conn. Caso o atributo connect_error for verdadeiro apresenta-se um erro de conexão. Os dados de sensor, localizacao e consumo são inseridos no banco através do comando armazenado na variável $sql. A fins de debugar, o código imprime quando há uma inserção no banco bem sucedida e quando houve um erro. Segue abaixo o trecho de código das funções expostas neste parágrafo:

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
$json = file_get_contents('php://input');
$data = json_decode($json,true);
if ($data["api_key"]==$api_key_value)
     {
        $sensor = $data["sensor"];
        $localizacao = $data["localizacao"];
$consumo = $data["consumo"];
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
  }
$sql = "INSERT INTO MonitoramentoAgua (sensor, localizacao, consumo)
        VALUES ('" . $sensor . "', '" . $localizacao . "', '" . $consumo/1000 . "')";
```

```
        if ($conn->query($sql) === TRUE) {
            echo "New record created successfully";
        }
        else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }
        $conn->close();
    } else {
        echo "Wrong API Key provided."; }
}
else {
    echo "No data posted with HTTP POST.";
}
```

3.2.3 Algoritmo da página Web

Para o desenvolvimento da página Web, buscou-se desenvolver uma página na qual fosse possível visualizar os dados de consumo ao longo do tempo. Uma API gráfica em JavaScript foi utilizada, a plotly.

O código completo desenvolvido para página Web está no ANEXO F deste documento. Nesta seção será detalhado o princípio de funcionamento do código e principais funções.

O código começa da mesma maneira que o código do servidor, definindo as variáveis servername, dbname, username, password. Logo após, é criada a conexão com o banco de dados. No algoritmo desenvolvido para a página Web, são realizadas três consultas ao banco de dados para disponibilizar ao usuário os dados de consumo mensal, consumo diário e consumo atual. Seguem abaixo as consultas ao banco:

```
$sql1 = "SELECT consumo, reading_time FROM MonitoramentoAgua WHERE DAY(reading_time)
= DAY(DATE_SUB(CURRENT_TIMESTAMP, INTERVAL 3 HOUR)) ORDER BY id ASC";

$sql2= " SELECT SUM(consumo), day(reading_time) FROM `MonitoramentoAgua` where
month(reading_time)=month(DATE_SUB(CURRENT_TIMESTAMP, INTERVAL 3 HOUR)) group by
day(reading_time)";

$sql3 = " SELECT SUM(consumo), month(reading_time) FROM `MonitoramentoAgua` where
year(reading_time)=year (DATE_SUB(CURRENT_TIMESTAMP, INTERVAL 3 HOUR)) group by
month(reading_time)";
```

A consulta armazenada na variável sql1 retorna os valores de consumo de água na janela de tempo entre as requisições POST vindas do ESP8266. A consulta sql2 soma os dados de

consumo ao longo dos dias da tabela MonitoramentoAgua nas quais o mês seja igual ao mês da data atual, desse modo, tem-se o valor consumido a cada dia do mês. Além disso, a consulta retorna os valores dos dias. Por fim a consulta sql3, soma os dados de consumo ao longo dos meses do ano atual e retorna um vetor com os meses consultados, possibilitando extrair a informação de consumo ao longo dos meses do ano. Em todas as consultas atualizou-se o fuso horário para GMT -3.

Após definidas as consultas, foi armazenado o valor de cada variável de interesse, nos vetores a serem plotados pela API. Para recuperar os dados inseridos no banco de dados utilizou-se o método fetch_assoc() dentro de um laço while, que retorna linha por linha da consulta em forma de uma matriz associativa, com as chaves iguais a consumo, e reading_time. Segue abaixo a consulta com o código sql1:

```
if ($result = $conn->query($sql1)) {
$vetorx=[];
$vetory=[];
$row_consumo = 0;
    while ($row = $result->fetch_assoc()) {
        $row_reading_time = $row["reading_time"];
        $vetorx[]=$row_reading_time;
        $row_consumo=$row_consumo+$row["consumo"];
        $vetory[]=$row_consumo;
    }    $result->free();
}
```

Quando não há mais linhas a serem tratadas, o método retorna NULL. A variável, $row_reading_time armazena o timestamp do dado, que é fornecido acessando a matriz associativa $row com a chave reading_time. O mesmo é feito com o consumo, armazenando em $row_consumo o valor da matriz associativa $row com a chave consumo, entretanto, aqui, $row_consumo é somada com ela mesma, para que se tenha o consumo cumulativo ao longo do ano. Esses valores de timestamp e consumo são respectivamente armazenados em vetorx e em vetory. Logo após, o objeto result é liberado. Um procedimento análogo é realizado para as consultas sql2 e sql3. Seguem abaixo as consultas sql2 e sql3:

```
if ($result = $conn->query($sql2)) {

    while ($row = $result->fetch_assoc()) {
        $dia[]= $row["day(reading_time)"];
        $somadia[] = $row["SUM(consumo)"];
```

```
    }
     $result->free();
}
if ($result = $conn->query($sql3)) {
    while ($row = $result->fetch_assoc()) {
        $mes[]= $row["month(reading_time)"];
        $somames[] = $row["SUM(consumo)"];
    }
     $result->free();
}
```

Definidas as variáveis a serem plotadas, o próximo passo foi o desenvolvimento da página Web. A página Web foi desenvolvida em HTML, uma linguagem de marcação que é interpretada por navegadores. Elementos visuais e de estilização foram implementados utilizando códigos CSS. Os códigos CSS podem ser referenciados dentro de um elemento HTML específico, no cabeçalho ou fazendo-se referência a um arquivo externo. O código da página Web começa no cabeçalho, referenciando o arquivo em JavaScript da API gráfica, tornando possível a chamada de métodos da API ao longo do código, e definindo o estilo CSS das três divisões da página HTML: center, graph1, graph2 e graph3. Segue abaixo o trecho de código de cabeçalho:

```
<head>
<script src="https://cdn.plot.ly/plotly-2.16.1.min.js"></script>
<style> #graph1{
     margin-left: 20%;
     margin-right: 20%;
  }
  #graph2{
     margin-left: 20%;
     margin-right: 20%;
  }
  #graph3{
     margin-left: 20%;
     margin-right: 20%;
  }
  #center {
      text-align: center;
      font-family: verdana;
        }
</style>
</head>
```

Na seção de corpo da página, foram criadas as divisões anteriormente citadas, a inserção de códigos da API, a definição dos títulos dos gráficos e título da página. As divisões e o título da página foram criados utilizando a seguinte sintaxe:

```
<div id="center" >
         <h1>SCADA</h1>
    </div>
    <div id="graph1" >
    </div>
    <div id="graph2" >
    </div>
    <div id="graph3" >
    </div>
```

Para a criação dos gráficos, os códigos da API gráfica foram inseridos entre o delimitador <script > - que indica o começo de instruções em JavaScript - e delimitador </script> que indica o final de um código JavaScript. Para plotar um gráfico utilizando-se da API, primeiro define-se as variáveis do eixo x e y a serem plotadas, e, logo em seguida, o tipo de gráfico. A sintaxe dessas definições para o gráfico de consumo diário é mostrada abaixo. Há códigos PHP embutidos, isso se faz necessário para passar dados de uma variável em PHP obtida das consultas ao banco, para variáveis JavaScript.

```
var trace1 = {
    x:<?php echo json_encode($dia); ?>,
    y: <?php echo json_encode($somadia); ?>,
    type: 'bar'
};
```

A variável data então recebe esses valores. Na variável layout, são definidos o título do gráfico, e a opção de inserir legenda. Após isso, um novo gráfico é instanciado pelo método estático newPlot() da classe Plotly. Segue o trecho de código. O procedimento é análogo para os demais gráficos criados.

```
var data = [trace1];
var layout = {
    title: "Consumo diário",
     xaxis: {title: 'Dia'},
     yaxis: {title: 'Consumo (L)'}
};
```

```
Plotly.newPlot("graph2", data, layout, {scrollZoom: true});
```

3.2.4 Algoritmo do ESP8266

Para a programação do ESP8266 foram construídos dois códigos: um para a calibração do sensor, que imprime através do monitor serial os dados de consumo a cada 2 segundos, e outro para realizar o envio dos dados de consumo. O código para calibração calcula o período entre dois pulsos e, através da curva do sensor, determina-se a vazão. Segue na tabela 3 a curva do sensor retirada do ANEXO B.

Tabela 3 - Curva do sensor de fluxo YF-S201

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

Por se tratar de uma curva linear, a função pode ser aproximada por uma reta. Calculando a equação da vazão em função da frequência para os pontos (65.5,480) e (82,600), tem-se que a equação é a seguinte, onde Q é a vazão em L/h e f é a frequência em Hz:

$$Q(f) = 7.27273\, f + 3.63636$$

No código, criou-se a função detectPulse, que é armazenada na memória RAM do microcontrolador, para tornar mais rápida sua chamada diante de uma interrupção. A interrupção foi programada para assim que a porta D2 do ESP8266 detectar uma borda de subida, executar uma função previamente definida, a detectPulse. O código completo se encontra no ANEXO C. Segue a sintaxe do código para esse trecho:

```
ICACHE_RAM_ATTR void detectPulse() {

   tempo_atual = micros();
   T = ((float)tempo_atual - (float)tempo_anterior)/1000000;
   f =   1/T;
   vazao = 3.63636 + 7.27273*f;
```

```
   consumo = consumo +  (vazao/3600)/f ;
   tempo_anterior = tempo_atual;

   }

void setup() {
  Serial.begin(115200);
  attachInterrupt(digitalPinToInterrupt(input), detectPulse, RISING);

}
```

O código para a postagem dos dados de consumo tem a tarefa tanto de contar os pulsos do sensor como de tratar esse dado e enviar os dados para o servidor. O primeiro passo é a definição das variáveis:

- ssid: string com o nome da rede Wi-Fi;
- password: string com a senha da rede;
- serverName: “destinatário” das mensagens HTTP, ou seja, o arquivo PHP do servidor;
- int ResponseCode: código de resposta retornado pelo servidor. Um código 200 significa que a mensagem foi entregue;
- input: pino correspondente ao GPIO D2;
- variáveis tempo_atual, tempo_anterior, vazao, f, T, consumo, t, t1 são autoexplicativas e são valores numéricos necessários para os procedimentos do algoritmo;
- WiFiClient client e HTTPClient http criam os objetos client e http de suas respectivas classes.

O código conta com três funções criadas: setup, loop, postar, conectar e detectPulse. Esta última é idêntica à do código de calibração.

A função conectar tem o objetivo de se conectar a uma rede WiFi. Ela começa iniciando o método begin, onde são inseridos o nome da rede e a senha da rede. Enquanto a rede não conectar é exibida a mensagem *Connecting…* caso se conecte, é mostrado pela interface serial o IP adquirido. Segue o trecho de código.

```
void conectar() {
```

```
  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
}
```

A função postar tem o objetivo de passar via requisição HTTP o JSON com o dado de consumo a ser gravado no banco de dados. A função começa invocando o método begin ao objeto http, onde é fornecido o destinatário e o cliente. Depois no método header é inserido o cabeçalho da requisição, onde é preenchido apenas o campo de Content-Type, informando ao servidor que estamos enviando um conteúdo JSON. Em httpRequestData é inserido o JSON a enviar, com os campos api_key, sensor, localizacao e consumo. Todos estes campos serão recebidos no servidor pela variável $json. Por último, é imprimido na interface serial o código de resposta da requisição. Segue abaixo o trecho de código.

```
void postar(String apiKeyValue, float consumo) {
  http.begin(client, serverName);
  http.addHeader("Content-Type", "application/json");
  String     httpRequestData     =     "{\"api_key\":\""     +     apiKeyValue     +
"\",\"sensor\":\"Fluxo\",\"localizacao\":\"Casa\",\"consumo\":\""+ consumo +"\"}";
  Serial.print("httpRequestData: ");
  Serial.println(httpRequestData);
  httpResponseCode = http.POST(httpRequestData);
  if (httpResponseCode > 0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
  } else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  http.end();
}
```

A função setup apenas conecta-se à uma rede e define a interrupção. Na função loop, realiza-se a postagem dos dados de consumo a cada 3 minutos, desse modo, o valor de consumo enviado corresponde à soma de consumo de água em litros ao longo de três minutos. Após esse

tempo, a variável de consumo é reiniciada caso a postagem dos dados ocorra com sucesso. Segue abaixo trecho de código. O código completo se encontra no ANEXO D.

```
void loop() {
  if (millis() - t1 > 1000 * 60 * 3) {
    t1 = millis();
    if (WiFi.status() == WL_CONNECTED) {
      postar(apiKeyValue, consumo * 1000);
      while (httpResponseCode != 200) {
        postar(apiKeyValue, consumo * 1000);
      }
    } else {
      Serial.println("WiFi Disconnected");
    }
    consumo = 0;
  }
}
```

3.3 Calibração

A calibração do sensor foi feita despejando-se uma determinada quantidade de água através do sensor, previamente medida em gramas num recipiente com auxílio de uma balança. Devido a densidade da água ser de 1 g/cm3 - assumindo-se que a água da concessionária tenha essa densidade - a massa de água em gramas é igual ao volume de água em mililitros. O valor da massa de água medido no recipiente foi então comparado com os valores de volume retornados pelo sensor e pelo código de calibração como um todo, para que se possa comprovar a validade da curva fornecida pelo fabricante. A balança segundo o manual apresenta uma margem de erro de 1%. Segue no ANEXO A o manual.

4 RESULTADOS

Nesta seção serão apresentados os resultados das medições ao longo do tempo e os dados de calibração do sensor.

4.1 Resultados de calibração

O procedimento de calibração foi realizado conforme descrito na seção 3.3. Foram realizadas 5 medições com diferentes volumes de água e comparado o valor pesado com o valor medido pelo sensor de fluxo. Segue na figura 14 (AUTOR) o recipiente onde foi realizada a medição do volume de água e a balança utilizada. Como pode-se observar pela indicação de volume do recipiente e pelo valor medido em gramas, é razoável a suposição de densidade da água igual a 1 g/cm3.

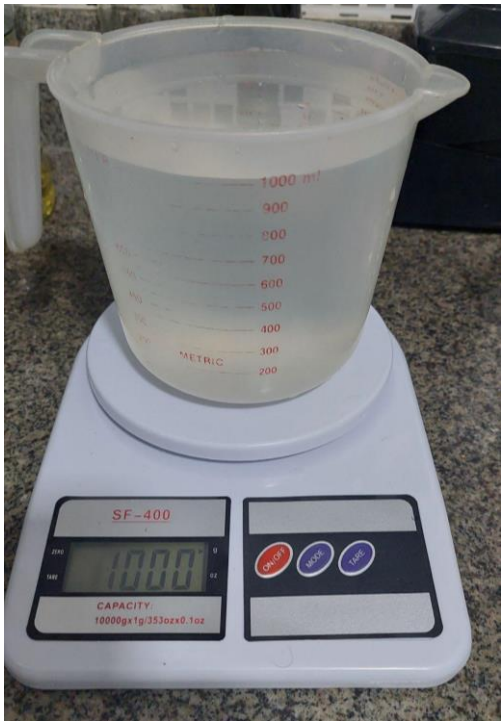


Figura : 13- Recipiente e balança

Para a primeira medição foram despejados 813 gramas de água e os resultados do algoritmo são mostrados na figura 15 (AUTOR).

```
consumo: 0.000000
vazao: 0.000000

consumo: 0.000000
vazao: 0.000000

consumo: 185.864349
vazao: 268.987000

consumo: 337.403198
vazao: 263.265442

consumo: 490.936615
vazao: 261.315277

consumo: 644.472534
vazao: 262.932220

consumo: 793.954224
vazao: 251.615417

consumo: 851.360168
vazao: 122.223785

consumo: 851.360168
vazao: 122.223785
```

Figura : 14- Resultados do algoritmo de calibração

A partir da oitava saída, não há mais água fluindo e a variável de consumo não é mais incrementada e, o valor de vazão, passa a representar o valor da última vazão registrada pelo algoritmo. Nota-se que o valor de vazão se encontra dentro dos limites de leitura do sensor em todo o momento da medição. Os valores das demais medições se encontram na tabela 4. Como pode-se observar, os erros estão dentro da margem de erro estipulada pelo fabricante que é de 10%, portanto, foi adotada a curva de vazão por frequência fornecida no datasheet mesmo.

Tabela 4 - Resultados da calibração

Massa de água no recipiente(g)	**Algoritmo (ml)**	**Erro (%)**
813	851.36	4.72
844	837.93	-0.72
947	990.14	4.56
885	877.72	-0.82
927	932.32	0.57

O erro foi calculado segundo a fórmula abaixo, onde VA é o volume retornado pelo algoritmo e MR é a massa de água do recipiente:

$$Erro = \frac{VA - MR}{MR} \times 100$$

4.2 Resultados do supervisório

Para analisar o desempenho do sistema supervisório, foi deixado durante cinco banhos o protótipo do sistema rodando o algoritmo do ANEXO D na saída de água de um chuveiro elétrico. Segue na figura 15 (AUTOR) a instalação do protótipo no chuveiro.

Figura : 15 - Instalação do medidor junto ao chuveiro

Ao longo dos 5 dias de operação, do dia 02/02/2023 ao dia 06/02/2023, obteve-se as seguintes entradas não nulas ao banco de dados:

Tabela 5 - Dados do sistema supervisório entre os dias de operação

id	sensor	localizacao	consumo	reading_time
154	Fluxo	Casa	10.39395	02/02/2023 00:12
155	Fluxo	Casa	8.83153	02/02/2023 00:15
158	Fluxo	Casa	8.54747	02/02/2023 23:59
159	Fluxo	Casa	11.71634	03/02/2023 00:02
160	Fluxo	Casa	3.47678	03/02/2023 00:05
162	Fluxo	Casa	2.40201	04/02/2023 00:00
163	Fluxo	Casa	11.43542	04/02/2023 00:03
164	Fluxo	Casa	11.70469	04/02/2023 00:06
165	Fluxo	Casa	0.04632	04/02/2023 00:09
170	Fluxo	Casa	3.29898	05/02/2023 04:50
171	Fluxo	Casa	10.78924	05/02/2023 04:53
172	Fluxo	Casa	10.71652	05/02/2023 04:56
173	Fluxo	Casa	0.02255	05/02/2023 04:59
175	Fluxo	Casa	1.68685	06/02/2023 01:14
176	Fluxo	Casa	10.39709	06/02/2023 01:17
177	Fluxo	Casa	10.51672	06/02/2023 01:20
178	Fluxo	Casa	9.83847	06/02/2023 01:23
Consumo total (L)			125.82093	

O consumo total de água durante os dias de uso foi de 125.82 litros de água. Já a figura 16 (AUTOR), mostra o gráfico referente ao consumo de água diário retornado pela API da página Web desenvolvida.

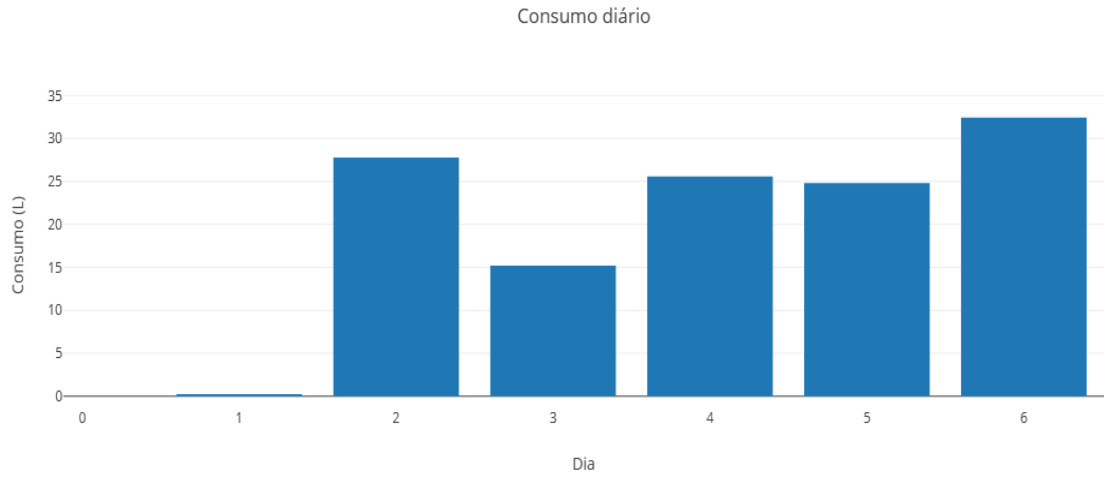


Figura : 16 - Gráfico de consumo diário de água

O gráfico de consumo mensal é mostrado na figura 17 (AUTOR).

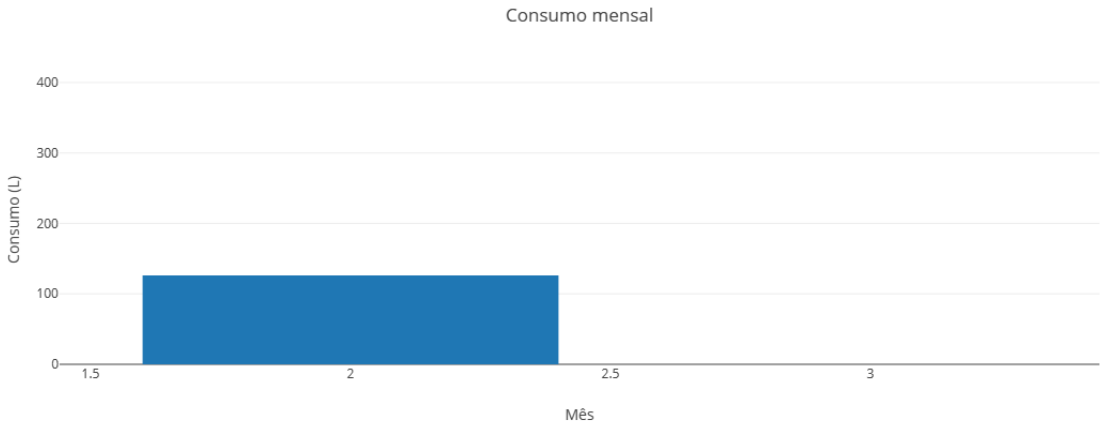


Figura : 17- Gráfico de consumo mensal

E por fim na figura 18 (AUTOR), o gráfico de consumo atual, do último dia de medição, com atualizações de consumo de três em três minutos.

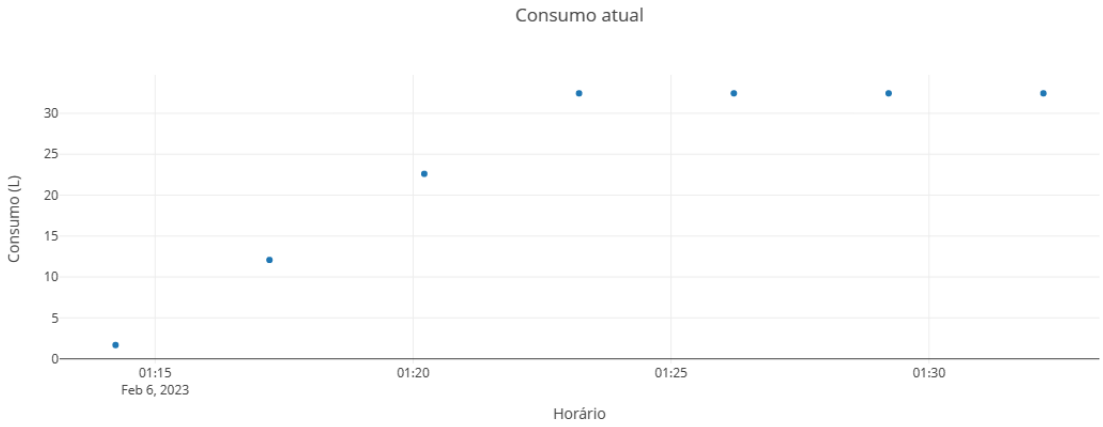


Figura : 18 - Gráfico de consumo atual

5 CONCLUSÕES

O trabalho desenvolvido cumpriu com o seu objetivo principal: desenvolvimento de um sistema supervisório para o monitoramento de consumo de água em uma instalação residencial. Conforme foi visto nos resultados, o protótipo se mostrou bem capaz de medir - dentro da margem de erro do sensor de fluxo e da balança – o volume de água consumido ao longo dos meses, dos dias e até entre alguns minutos de um ponto hidráulico.

As leituras foram feitas de forma continua ao longo do período de testes e não houveram interrupções de medição nem perdas de pacotes, muito devido ao HTTP utilizar o TCP, que garante que todas as mensagens sejam enviadas, a estabilidade do servidor Web, que se mostrou estável ao longo de todo o projeto, e, por fim, a qualidade da rede no local de medição. O ESP8266 também se provou como um microcontrolador capaz de realizar as leituras do sensor de fluxo e as postagens HTTP de maneira simultânea, sem gargalos de desempenho ou interrupções. No que diz respeito a elementos na rede, o banco de dados contribuiu em muito para a gestão dos dados enviados pelo ESP8266, permitindo recuperá-los de maneira rápida e com os filtros de tempo de interesse no projeto. O banco de dados eliminou a necessidade de manipulação de arquivos de logs, que poderiam exigir elementos de memória adicionais ao ESP8266 e maior complexidade na recuperação de informações, além de menor confiabilidade. Por último, a disponibilização dos dados de consumo de água ao usuário através da página Web permitiu que uma análise de consumo ao longo de certos períodos de tempo pudesse ser feita, desse modo, um maior controle pode ser feito.

O trabalho, apesar do escopo restrito, tem boa parte dos fundamentos tecnológicos que estão por trás de medidores inteligentes que possibilitam tanto a companhia energética (UK GOV, 2013) quanto o consumidor controlarem seus gastos e reduzirem despesas. Companhias podem economizar com leituras presenciais, monitorar os recursos que de fatos são entregues ao consumidor e controlar os desperdícios. Consumidores podem ver seus gastos e até mesmo realizar pagamentos. Na indústria, sensores acoplados a dispositivos inteligentes podem monitorar ativos, obtendo um histórico de operação para auxiliar na manutenção preditiva e preventiva. Diversas startups e empresas de tecnologia tem empregado esse conceito.

Logo, os conceitos deste trabalho podem ser aplicados em diversas áreas e mostra o poder que a Internet das Coisas tem. Desafios de robustez de conexão, segurança no envio de dados sensíveis, acesso à rede em locais remotos e quantidade de tráfego de dados adicional na rede são vários dos desafios que teremos que conviver e buscar soluções.

6 BIBLIOGRAFIA

BOYER, S. A. **Scada:** Supervisory Control And Data Acquisition. 3rd. ed. [S.l.]: ISA, 2004.

BUGAY, E. L.; ULBRICHT, V. R. **Hipermídia**. 1ed. ed. Florianópolis: Bookstore Livraria Ltda, 2000.

CAMPBELL, J. T.; SHELLY, G. B. **Web design introductory.** 6th. ed. [S.l.]: Campbell, 2017.

EMONCMS , 2023. Disponivel em: <https://github.com/emoncms/emoncms>. Acesso em: Fevereiro 2023.

ESPRESSIF. ESP8266, 2023. Disponivel em: <https://www.espressif.com/en/products/socs/esp8266>. Acesso em: Fevereiro 2023.

GIFE. Crise hídrica: Brasil já perdeu um Nordeste e meio de água, 2021. Disponivel em: <https://gife.org.br/crise-hidrica-brasil-ja-perdeu-um-nordeste-e-meio-de-agua/#:~:text=O%20Brasil%20perdeu%2C%20nos%20%C3%BAltimos,meia%20toda%20regi%C3%A3o%20do%20Nordeste.>. Acesso em: Fevereiro 2023.

GREVE, P. et al. Global assessment of water challenges under uncertainty in water scarcity projections. **Nature Sustainability**, 2018. 486-494. Disponivel em: <http://doi.org/10.1038/s41893-018-0134-9>. Acesso em: Fevereiro 2023.

HALLIDAY, D.; RESNICK, R.; WALKER, J. **Fundamentos de Física - Eletromagnetismo**. 10. ed. [S.l.]: LTC, v. 3, 2016.

HOEKSTRA, A. Y.; MEKONNEN, M. Four billion people affected by severe water scarcity. **Science Advances**, 2, 2016. 7. Disponivel em: <https://www.science.org/doi/abs/10.1126/sciadv.1500323>.

HOSTINGER. Como Conectar PHP a um Banco de Dados MySQL, 2021. Disponivel em: <https://www.hostinger.com.br/tutoriais/como-conectar-php-com-mysql/#Como_Usar_o_PDO_para_Conectar_um_Script_PHP_no_MySQL>. Acesso em: Fevereiro 2023.

INMETRO. **Portaria nº 246 de 17 de outubro de 2000.** INMETRO. [S.l.]. 2000.

INMETRO. **Portaria Inmetro n.º 12, de 04 de janeiro de 2011**. INMETRO. [S.l.]. 2011.

INMETRO. Portaria INMETRO / ME - número 155, 30 mar. 2022. Disponivel em: <http://sistema-sil.inmetro.gov.br/rtac/RTAC002971.pdf>. Acesso em: Fevereiro 2023.

JÚNIOR, J. F. D. S. **Detecção De Perdas Em Sistemas De Distribuição De Água Através De Rede De Sensores Sem Fio**. Recife. 2017.

KHAN, R. et al. Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. **International Conference on Frontiers of Information Technology**, Dezembro 2012. 257-260.

KUROSE, J.; ROSS, K. **Redes de Computadores**. 6. ed. [S.l.]: Pearson Education do Brasil Ltda., 2013.

LEE, I.; LEE, K. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. **Business Horizons**, 2015. 431-440.

MCCRADY, S. G. **Designing SCADA application software:** a practical approach. 1st. ed. [S.l.]: Elsevier, 2013.

MQTT , 2019. Disponivel em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/>. Acesso em: Fevereiro 2023.

NETWORK WORKING GROUP. Hypertext Transfer Protocol -- HTTP/1.0, 1996. Disponivel em: <https://www.rfc-editor.org/rfc/rfc1945>. Acesso em: Fevereiro 2023.

OBSERVER vs Pub-Sub pattern, 2017. Disponivel em: <https://hackernoon.com/observer-vs-pub-sub-pattern-50d3b27f838c>. Acesso em: Fevereiro 2023.

OLIVEIRA, S. D. **Internet das Coisas com ESP8266, Arduino e Raspberry Pi**. 1. ed. [S.l.]: Novatec, 2017.

PNSH, 2023. Disponivel em: <https://pnsh.ana.gov.br/seguranca>. Acesso em: Fevereiro 2023.

RECH, A. L. **Água, micromedição e perdas**. 2. ed. [S.l.]: Scortecci, 1999.

RICHEY, A. S. et al. Quantifying renewable groundwater stress with GRACE. **Water Resources Research**, 2015. 5217-5238. Disponivel em:

<https://agupubs.onlinelibrary.wiley.com/doi/full/10.1002/2015WR017349>. Acesso em: Fevereiro 2023.

SILVERIO-FERNÁNDEZ, M.; RENUKAPPA, S.; SURESH, S. What is a smart device? - a conceptualisation within the paradigm of the internet of things. **Visualization in Eng**, 2018. 1-10.

SNIS. Mapa de Indicadores de Água, 2023. Disponivel em: <http://appsnis.mdr.gov.br/indicadores/web/agua_esgoto/mapa-agua>. Acesso em: Fevereiro 2023.

T. BERNERS-LEE, R. F. H. F. **Hypertext Transfer Protocol -- HTTP/1.0**. RFC. [S.l.]. 1996.

UK GOV. Smart meters: a guide for households, 2013. Disponivel em: <https://www.gov.uk/guidance/smart-meters-how-they-work>. Acesso em: Fevereiro 2023.

UNESCO WORLD WATER ASSESSMENT PROGAMME. **The United Nations world water development report 2021: valuing water; facts and figures**. UNESCO. [S.l.], p. 11. 2021. (SC-2021/WS/3).

USAGE statistics of PHP for websites, 2023. Disponivel em: <https://w3techs.com/technologies/details/pl-php>. Acesso em: Fevereiro 2023.

VITORIANO, F. A. Assessment of an adaptable data diet for a self-powered wireless IoT sensor system, Belo Horizonte, Abril 2022. Disponivel em: <http://hdl.handle.net/1843/45609>. Acesso em: Fevereiro 2023.

WHAT is SCADA? **Inductive automation**, 2023. Disponivel em: <https://www.inductiveautomation.com/resources/article/what-is-scada>. Acesso em: Fevereiro 2023.

WIKIMEDIA. File:NodeMCU DEVKIT 1.0.jpg. **https:** //commons.wikimedia.org/wiki/File:NodeMCU_DEVKIT_1.0.jpg, 2023. Acesso em: Fevereiro 2023.

WORD Wide Web, 1991. Disponivel em: <http://info.cern.ch/hypertext/WWW/TheProject.html>. Acesso em: Fevereiro 2023.

WORTMANN, F.; FLÜCTHER, K. Internet of Things. **Bus Inf Syst Eng**, 2015. 221-224.

YF-S201 Datasheet. Disponivel em: <https://www.hobbytronics.co.uk/datasheets/sensors/YF-S201.pdf>. Acesso em: Fevereiro 2023.

7 ANEXO A – Manual da Balança

■ **Indicação de bateria fraca/sobrecarga**

h-Lo	Energia da bateria fraca para o visor de leitura. Substitua a bateria.
b-Lo	Energia da bateria fraca para a unidade principal. Substitua a bateria.
Err	Sobrecarga da balança.

⚠ **Conselhos de uso e manutenção**

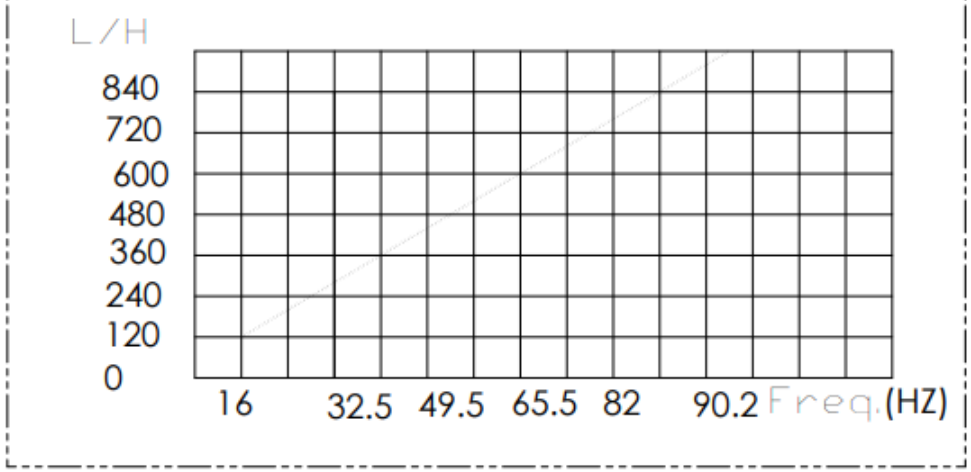
1. A plataforma fica escorregadia quando molhada. Mantenha-a seca!
2. Fique imóvel no durante a pesagem.
3. NÃO dê pancadas, bata ou deixe cair a balança. Trate a balança com cuidado pois ela é um instrumento preciso.
4. Coloque a balança sobre uma superfície plana e sólida. Superfícies macias ou desniveladas podem causar alterações no peso.
5. Evite a interferência de campo eletromagnético forte.
6. Limpe a balança com pano úmido e evite a entrada de água na balança. NÃO use produtos de limpeza químicos/abrasivos.
7. Mantenha a balança em local fresco e seco.
8. Sempre mantenha sua balança na posição horizontal.
9. Se não conseguir ligar a balança, verifique se a bateria está instalada, ou se a energia da bateria está fraca. Insira ou substitua por uma nova bateria.
10. Se houver um erro no visor ou não for possivel desligar a balança por um longo tempo, retire a bateria por cerca de 3 segundos e então instale-a novamente para remover falha do software. Se não puder resolver o problema, entre em contato com o representante.
11. Esta balança é para uso doméstico. NÃO use para finalidades comerciais.
12. É recomendável pesar-se no mesmo horário todos os dias de preferência sem roupas e mantendo-se ereto, alternância de horário nas pesagens podem apresentar resultados inconstantes.
13. Na utilização da balança para acompanhamento, não é aconselhável fazer comparações nos resultados entre diferentes marcas e modelos.
14. Segundo as normas, esta balança poderá apresentar uma variação de 1% para mais ou para menos na pesagem.

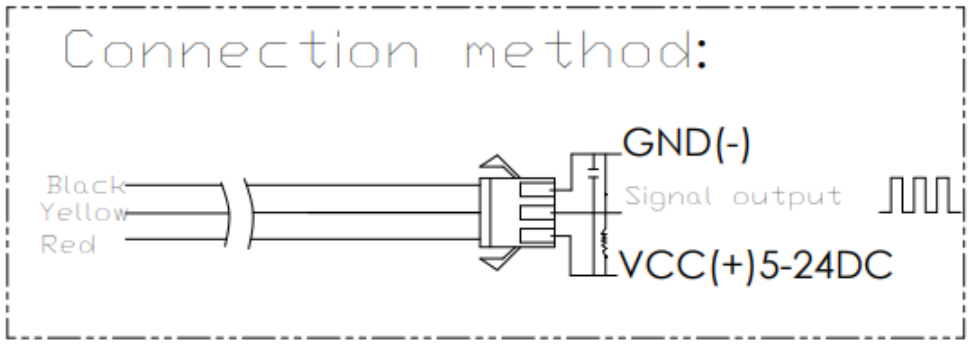
8 ANEXO B – Datasheet YF-S201

Flow-Pulse	Flow-Curve	Connection method

一.Flow Range:100L/H-/1800H-L/H

Flow (L/H)	Frezq.(HZ)	Erro range
120L/H	16	±10
240L/H	32.5	
360L/H	49.3	
480L/H	65.5	
600L/H	82	
720L/H	90.2	





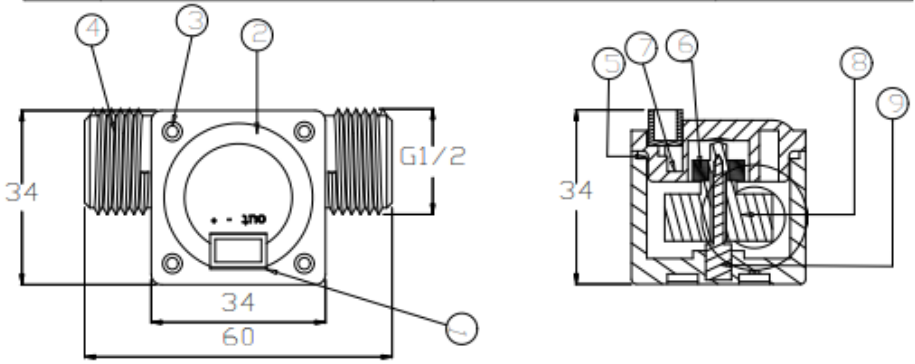
YIFA the plastics Ltd
Prodcut Introduction

1.Modle:YF-21
2.Product Name:Hall sensor
3.Flow Range: 1-30L/MIN
4.(1)Connection Method

(2)Voltage Range 3.5-24VDC, Pluse Characteristic:F=7Q(L/MIN).
(3)Extent of error:±5%.
(4)Flow-Pulse
2L/MIN=16HZ 4L/MIN=32.5HZ 6L/MIN=49.3HZ
8L/MIN=65.5HZ 10L/MIN=82HZ

5.Bom

No.	Item	Material	Qty.
1	Connection wire		1
2	Bonnet	PA	1
3	Screw		4
4	Valve body	PA	1
5	Leak press valve		1
6	Magnet		1
7	Hall		1
8	Impeller	POM	1
9	Rustless steel axis	SUS304	1
10			
11			



9 ANEXO C – Código de calibração no ESP8266

```
const int input = 4;
volatile unsigned long tempo_atual, tempo_anterior = 0;
volatile float vazao, f, T, consumo = 0;
volatile unsigned long t=0;


ICACHE_RAM_ATTR void detectPulse() {

   tempo_atual = micros();
   T = ((float)tempo_atual - (float)tempo_anterior)/1000000;
   f =   1/T;
   vazao = 3.63636 + 7.27273*f;
   consumo = consumo +  (vazao/3600)/f ;
   tempo_anterior = tempo_atual;

   }

void setup() {
  Serial.begin(115200);
  attachInterrupt(digitalPinToInterrupt(input), detectPulse,
RISING);

}

void loop() {

if (millis()-t > 2000){
  t=millis();
  Serial.printf( "consumo: %f\n", consumo*1000);
  Serial.printf( "vazao: %f\n", vazao);
  Serial.println();
}
}
```

10 ANEXO D – Código de monitoramento de consumo de água e envio de dados do ESP8266

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>


const char* ssid = "ubnt";
const char* password = "**********";
const char* serverName =
"http://arturmiranda.online/definitivo/monitoramentoespPHPv2.php";
int httpResponseCode;
String apiKeyValue = "AXYLeagmiaYFo";
const int input = 4;
volatile unsigned long tempo_atual, tempo_anterior = 0;
volatile float vazao, f, T, consumo = 0;
unsigned long t = 0, t1 = 0;
WiFiClient client;
HTTPClient http;

ICACHE_RAM_ATTR void detectPulse() {

  tempo_atual = micros();
  T = ((float)tempo_atual - (float)tempo_anterior) / 1000000;
  f = 1 / T;
  vazao = 3.63636 + 7.27273 * f;
  consumo = consumo + (vazao / 3600) / f;
  tempo_anterior = tempo_atual;
}

void conectar() {

  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
}

void postar(String apiKeyValue, float consumo) {

  http.begin(client, serverName);
  http.addHeader("Content-Type", "application/json");
  String httpRequestData = "{\"api_key\":\"" + apiKeyValue +
"\",\"sensor\":\"Fluxo\",\"localizacao\":\"Casa\",\"consumo\":\"" +
consumo + "\"}";
  Serial.print("httpRequestData: ");
  Serial.println(httpRequestData);
  httpResponseCode = http.POST(httpRequestData);
```

```
  if (httpResponseCode > 0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
  } else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  http.end();
}


void setup() {
  Serial.begin(115200);
  conectar();

  attachInterrupt(digitalPinToInterrupt(input), detectPulse,
RISING);
}


void loop() {

  if (millis() - t1 > 1000 * 60 * 3) {

    t1 = millis();
    if (WiFi.status() == WL_CONNECTED) {
      postar(apiKeyValue, consumo * 1000);
      while (httpResponseCode != 200) {
      postar(apiKeyValue, consumo * 1000);
      }

    } else {
      Serial.println("WiFi Disconnected");
    }

    consumo = 0;
  }
}
```

11 ANEXO E – Código do servidor

```
<?php
$servername = "localhost";
$dbname = "u660736675_esp32";
$username = "u660736675_artur";
$password = "*******";
$api_key_value = "AXYLeagmiaYFo";
$api_key= $sensor = $localizacao = $value1 = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $json = file_get_contents('php://input');
    $data = json_decode($json,true);
    if ($data["api_key"]==$api_key_value)
     {
        $sensor = $data["sensor"];
        $localizacao = $data["localizacao"];
        $consumo = $data["consumo"];
        $conn = new mysqli($servername, $username, $password,
$dbname);
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }
        $sql = "INSERT INTO MonitoramentoAgua (sensor, localizacao,
consumo)
        VALUES ('" . $sensor . "', '" . $localizacao . "', '" .
$consumo/1000 . "')";
        if ($conn->query($sql) === TRUE) {
            echo "New record created successfully";
        }
        else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }

        $conn->close();
    }
    else {
        echo "Wrong API Key provided.";
    }

}
else {
    echo "No data posted with HTTP POST.";
}
```

12 ANEXO F – Código da página Web

```
<!DOCTYPE html>

<?php
$servername = "localhost";
$dbname = "u660736675_esp32";
$username = "u660736675_artur";
$password = "********";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql1 = "SELECT consumo, reading_time FROM MonitoramentoAgua WHERE
DAY(reading_time) = DAY(DATE_SUB(CURRENT_TIMESTAMP, INTERVAL 3
HOUR)) ORDER BY id ASC";
$sql2= " SELECT SUM(consumo), day(reading_time) FROM
`MonitoramentoAgua` where
month(reading_time)=month(DATE_SUB(CURRENT_TIMESTAMP, INTERVAL 3
HOUR)) group by day(reading_time)";
$sql3 = " SELECT SUM(consumo), month(reading_time) FROM
`MonitoramentoAgua` where year(reading_time)=year
(DATE_SUB(CURRENT_TIMESTAMP, INTERVAL 3 HOUR)) group by
month(reading_time)";


if ($result = $conn->query($sql1)) {

$vetorx=[];
$vetory=[];
$row_consumo = 0;
    while ($row = $result->fetch_assoc()) {
        $row_reading_time = $row["reading_time"];
        $vetorx[]=$row_reading_time;
        $row_consumo=$row_consumo+$row["consumo"];
        $vetory[]=$row_consumo;

    }
    $result->free();
}


if ($result = $conn->query($sql2)) {


    while ($row = $result->fetch_assoc()) {

        $dia[]= $row["day(reading_time)"];
        $somadia[] = $row["SUM(consumo)"];

    }
     $result->free();
}
```

```
if ($result = $conn->query($sql3)) {


    while ($row = $result->fetch_assoc()) {

        $mes[]= $row["month(reading_time)"];
        $somames[] = $row["SUM(consumo)"];

    }
     $result->free();
}


$conn->close();

?>

<html>

<head>
<script src="https://cdn.plot.ly/plotly-2.16.1.min.js"></script>
<style> #graph1{


     margin-left: 20%;
     margin-right: 20%;

  }
  #graph2{


     margin-left: 20%;
     margin-right: 20%;

  }

  #graph3{

     margin-left: 20%;
     margin-right: 20%;

  }

  #center {

      text-align: center;
      font-family: verdana;

  }

</style>
</head>

<body>
```

```
     <div id="center" >
         <h1>SCADA</h1>
    </div>

    <div id="graph1" >
    </div>

    <div id="graph2" >
    </div>

    <div id="graph3" >
    </div>

    <script >

var trace1 = {
    x:<?php echo json_encode($vetorx); ?>,

    y: <?php echo json_encode($vetory); ?>,
    mode: 'markers'
};

var data = [trace1];

var layout = {
    title: "Consumo atual",
      xaxis: {title: 'Horário'},
  yaxis: {title: 'Consumo (L)'}


};

Plotly.newPlot("graph1", data, layout, {scrollZoom: true});

</script>



 <script >


var trace1 = {
    x:<?php echo json_encode($dia); ?>,

    y: <?php echo json_encode($somadia); ?>,
    type: 'bar'
};

var data = [trace1];

var layout = {
    title: "Consumo diário",
      xaxis: {title: 'Dia'},
  yaxis: {title: 'Consumo (L)'}

};
```

```
Plotly.newPlot("graph2", data, layout, {scrollZoom: true});

</script>


<script >

var trace1 = {
    x:<?php echo json_encode($mes); ?>,

    y: <?php echo json_encode($somames); ?>,
    type: 'bar'
};

var data = [trace1];

var layout = {
    title: "Consumo mensal",
      xaxis: {title: 'Mês'},
  yaxis: {title: 'Consumo (L)'}

};

Plotly.newPlot("graph3", data, layout, {scrollZoom: true});

</script>


</body>

</html>
```