Arnaldo Rafael Morais Andrade

ESPWater - Uma solução IoT para gerenciamento automático de água

Universidade Federal de Pernambuco
secgrad@cin.ufpe.br
https://portal.cin.ufpe.br/graduacao/

Recife
2021

Arnaldo Rafael Morais Andrade

ESPWater - Uma solução IoT para gerenciamento automático de água

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Área de Concentração: Sistemas distribuídos e IoT
Orientador: Nelson Souto Rosa

Recife
2021

AGRADECIMENTOS

Aos meus pais, que sempre me apoiaram nos momentos difíceis e fizeram de tudo para me garantir a melhor educação possível. Meus agradecimentos à todos meus amigos, em especial Filipe, Tiago, Lucas, Paulo, Leonardo, Cyrus, que tornaram o processo de construção deste trabalho algo muito mais prazeroso e animador. À instituição pelo ambiente criativo e pela estrutura de qualidade. Por fim, agradeço ao meu orientador Nelson Souto Rosa, que esteve disponível sempre que precisei e por tornar este projeto possível.

ABSTRACT

In June 2021, Brazil began to worry about a health crisis and a water crisis. Some solutions to minimize the effects of scarcity are present at the governmental, community and individual levels. These facts lead us to, among other things, reason about how to do automatic water management, especially in residences, aiming at a conscious consumption of the natural resource. For this purpose, several mechanisms exist, from the more conventional ones, such as the electric float, to the most sophisticated ones, such as microcontrollers and sensors to perform the control. This work proposes ESPWater, a self-adaptive IoT solution to automatically manage reservoirs, aiming at reduced energy consumption by the devices. To evaluate the proposed solution, an assessment was carried out to measure the impact of the system's adaptation, indicating the gain in battery life. ESPWater showed to be low-cost, flexible, and capable of managing water resources in an automated way.

Keywords: Adaptive Software. IoT. Water Management.

RESUMO

Em junho de 2021 o Brasil passou a se preocupar não só com uma crise sanitária, mas também com uma crise hídrica. Algumas soluções para minimizar efeitos da escassez estão presentes no âmbito governamental, comunitário e individual. O que nos leva a, entre outras coisas, questionar como fazer um gerenciamento automático de água, especialmente em residências, almejando um consumo consciente do recurso natural. Para o feito, diversos mecanismos existem, desde os mais convencionais, como a boia elétrica, até os mais sofisticados, como o uso de microcontroladores e sensores para realizar o controle. Este trabalho propõe a *ESPWater*, uma solução IoT auto-adaptativa para gerenciar automaticamente reservatórios, visando consumo reduzido de energia por partes dos dispositivos. Para avaliar o trabalho foi feito um estudo sobre o impacto positivo da adaptação do sistema, indicando o ganho no tempo de vida da bateria. A *ESPWater* demonstrou ser uma alternativa de baixo custo, flexível e capaz de administrar o recurso hídrico de forma automatizada.

Palavras-chave: Software adaptativos. IoT. Gerenciamento de recursos hídricos.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ADC	*Analog to Digital Converter*
AMQP	*Advanced Message Queuing Protocol*
ANEEL	Agência Nacional de Energia Elétrica
CoAP	*Constrained Application Protocol*
GPIO	*General Purpose Input/Output*
HTTP	*Hypertext Transport Protocol*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
M2M	*Machine to Machine*
MQTT	*Message Queuing Telemetry Transport*
NTP	*Network Time Protocol*
OASIS	*Organization for the Advancement of Structured Information Standards*
QoS	*Quality of Service*
SMS	*Short Message Service*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo está organizado em três seções: motivação, objetivos e estrutura. Nele são apresentados o tema de uma forma detalhada e o problema de pesquisa.

1.1 MOTIVAÇÃO

Desde 2012, diferentes municípios brasileiros têm enfrentado problemas de reduções da pluviosidade, resultando em um cenário de escassez hídrica (JACOBI; CIBIM; SOUZA LEãO, 2015). Apesar do Brasil apresentar situação de disponibilidade hídrica privilegiada, o panorama da crise hídrica se instalou em diversas partes da região Sudeste, em destaque na cidade de São Paulo, cujos mananciais encontram-se com o nível de água reduzido (CIRILO, 2015). Esse fenômeno está atrelado não só a causas climáticas, mas também a uma falta de planejamento de instituições governamentais (JACOBI; CIBIM; SOUZA LEãO, 2015). Por fim, o desperdício de água também se mostra um grande problema.

Em 2021, essa situação se agravou, levando a Agência Nacional de Energia Elétrica (ANEEL) a anunciar um aumento de 52% no valor da bandeira vermelha patamar 2, justificado pelo aumento nos custos de compra de energia das distribuidoras (AID, 2021). Uma campanha nacional também foi lançada incentivando o consumo consciente de energia e água, medida simples em direção contrária à crise (gov.br, 2021).

No Brasil, o desperdício é alto e frequente, seja no transporte, na distribuição ou no uso do recurso hídrico (CIRILO, 2015). Banhos prolongados, utilização de água bruta para usos que não requeiram água tratada, bacias sanitárias ineficientes, são alguns dos exemplos que evidenciam o problema. O uso não gerenciado da água acarreta até problemas de transbordamento de reservatórios. Como no caso de uma bomba d'água operada manualmente para encher tanques.

Esse controle humano atrelado a mecanismos envolvendo água pode gerar transtornos. Nesse contexto, é possível definir como problema a elaboração do gerenciamento automático do recurso natural, especialmente em residências. Para esse fim, a boia elétrica ou chave boia é comumente utilizada. Trata-se de um sensor de nível capaz de realizar o chaveamento automático. Quando a boia detecta que o nível do líquido está baixo, ela emite sinais para que a bomba seja acionada. Na situação inversa, a bomba é desligada (FAME, c2021).

Embora seja amplamente difundida, a boia elétrica possui desvantagens. Uma delas é a degradação do material com o passar do tempo, afinal, tem-se água e eletricidade no mesmo espaço. Outro problema é a necessidade de uma mão de obra minimamente especializada para instalação e manutenção do equipamento. Do contrário, pode-se provocar falhas até na bomba d'água. Além disso, ainda que não seja seu objetivo, o usuário é incapaz de monitorar o nível de água no reservatório.

Esses fatores negativos contribuíram para a criação de soluções alternativas. REZA; TARIQ; REZA (2010) introduziu a noção de monitoramento e controle de água utilizando micro-controladores. No sistema proposto e construído em SARASWATI; KUANTAMA; MARDJOKO (2012) utiliza-se além de um microcontrolador, um sensor e o monitoramento de água é feito por meio de *Short Message Service* (SMS). Já em WADEKAR et al. (2016), ainda que o sensor tenha sido implementado, a automatização dos processos de leitura do nível de água e acionamento do motor foi atingida. Desta vez, informações do reservatório foram apresentadas em um aplicativo móvel. Também é possível observar trabalhos relacionados em escalas maiores. Em SIDDULA; BABU; JAIN (2018) é proposto um sistema para automatizar o controle de represas sem interferência humana. De mesmo modo, são utilizados sensores e pequenos hardware a fim de atingir o objetivo. Vale destacar a troca de informações dos dispositivos feita via *Bluetooth*.

Percebe-se, então, a transição das soluções para um cenário de Internet das Coisas (do inglês *Internet of Things* (IoT)). A nova preocupação passa a ser a comunicação entre os dispositivos e a disponibilidade de energia dos mesmos. É nesse cenário que a proposta *ESPWater* reside, uma solução auto-adaptativa que permite o gerenciamento automático de água, de baixo custo e que independe da distância física entre os componentes: reservatório e bomba. O foco da solução está em reservatórios de pequeno porte, como a caixa d'água, usada em residências.

1.2 OBJETIVOS

O objetivo deste trabalho é implementar uma solução no cenário IoT para gerenciamento automático de água, visando baixo consumo de energia dos dispositivos através da adaptação do sistema. Para avaliar o projeto, será realizada uma abordagem analítica do impacto positivo da adaptação em relação à descarga de bateria.

1.3 ESTRUTURA DO DOCUMENTO

Este trabalho está organizado da seguinte forma: O Capítulo 2 introduz os conceitos básicos necessários para melhor entendimento do projeto; O Capítulo 3 descreve a *ESPWater*, a solução proposta para gerenciar automaticamente reservatórios de água; O Capítulo 4 fornece avaliações acerca do projeto e no Capítulo 5 são apresentadas as conclusões, limitações e trabalhos futuros.

2

CONCEITOS BÁSICOS

Este capítulo introduz os conceitos básicos que permitem a compreensão da solução proposta. A começar pela definição e contexto do termo IoT. Em seguida é feita uma introdução sobre microcontroladores e sobre o protocolo de comunicação *Message Queuing Telemetry Transport* (MQTT). O capítulo finaliza apresentando os conceitos básicos de software adaptativos.

2.1 IOT

Em 1982 um grupo de estudantes de graduação em ciência da computação na Universidade Carnegie Mellon em Pittsburgh evidenciaram a máxima da necessidade ser a mãe da invenção. Sedentos por Coca-Cola, eles muitas vezes se encontravam frustrados por conta da máquina de refrigerantes da universidade estar vazia ou com produtos quentes. Esse problema só era percebido após uma longa caminhada até a maquina. Então os cientistas conectaram a máquina à rede da universidade e isso permitiu a visualização do catálogo de produtos através da internet, incluindo a verificação da temperatura de seus itens. Foi um grande passo em direção à conectividade de objetos não computacionais (ORNES, 2016).

O paradigma da computação difusa reside em um mundo onde a computação não se limita a *tablets*, *smartphones* e *laptops*. A ideia da Internet das Coisas sugere que, ao invés de possuir uma pequena quantidade de dispositivos poderosos de propósito geral, tenha-se dispositivos menos poderosos, porém com funções específicas e em maior número (MCEWEN; CASSIMALLY, 2013). Qualquer objeto físico, juntamente com controladores e sensores, pode observar o ambiente e fazer interações através da internet. Etiquetas inteligentes em vacas alertam o fazendeiro sobre o estado de saúde do animal, ou a geladeira que informa detalhes de seu estoque ao computador são alguns dos casos de uso (ORNES, 2016).

São muitos os benefícios trazidos com a Internet das coisas. Saúde, agricultura e cidades inteligentes são potenciais segmentos contemplados com IoT. Porém, há desafios atrelados aos benefícios, sendo os principais: grande volume de dados gerados e o consumo de energia dos dispositivos (ORNES, 2016). A necessidade de troca constante de bateria torna um projeto IoT inviável, mesmo que resolva bem determinado problema. Logo, é necessário se ater nesses desafios a fim de construir um projeto eficiente.

2.2 MICROCONTROLADORES

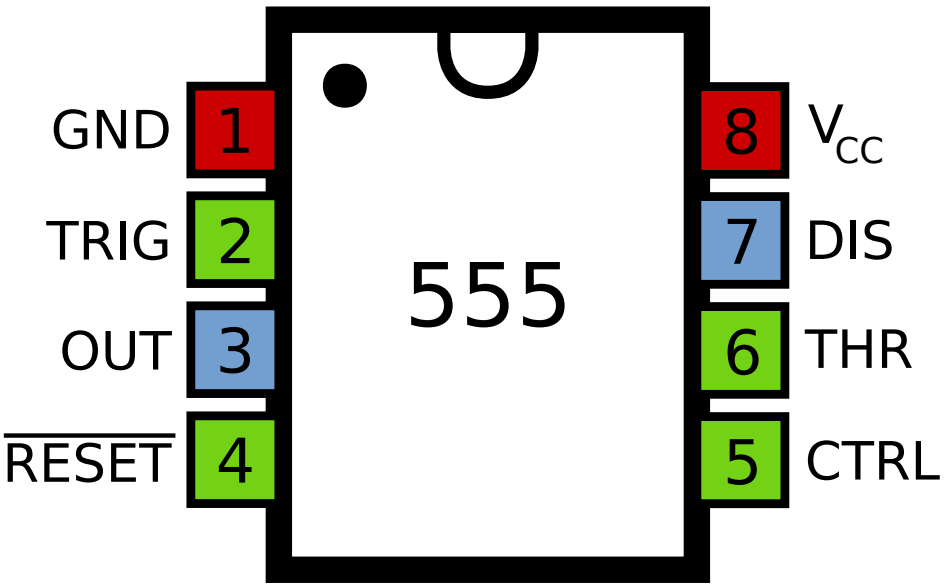
Microcontrolador é definido como um sistema em um chip (AXELSON, 1997). O termo *micro* sugere que o dispositivo é pequeno e *controlador* informa que o dispositivo pode ser usado para controlar objetos, processos ou eventos. Outro termo para descrevê-lo é Sistema Embarcado porque o microcontrolador e seus circuitos de suporte são frequentemente integrados. Ele contém, além da *central processing unit* (CPU), memória e interfaces de entrada e saída (I/O *interfaces*). Entretanto, por possuírem quantidade limitada de memória, microcontroladores tendem a ser usados em sistemas pequenos (AXELSON, 1997).

Em 1997 microcontroladores eram usados em diversos segmentos, como em periféricos de computadores e até automóveis. Com o passar do tempo, produtos mais potentes e baratos foram criados, e, com isso, seus casos de uso aumentaram. Existem recipientes de remédio, juntamente com dispositivos de lembrete, que alertam ao médico do paciente caso o comprimido não seja consumido naquele dia, por exemplo (MCEWEN; CASSIMALLY, 2013).

Tecnicamente, ainda que a CPU do dispositivo seja um microprocessador de capacidade limitada, ela contém as duas principais partes de um processador comum: a Unidade de Controle, usada para buscar dados da memória e a Unidade de Execução, responsável por executar os dados. Sua memória armazena o código do *firmware*, o qual é executado desde que seja fornecido energia ao microcontrolador. A escrita do *firmware* na memória do dispositivo é facilitada com uso de *Integrated Development Environment*s (IDEs) e a linguagem de programação a ser utilizada varia desde C e Assembly, até Java, Python e C/C++. O *setup* e o *loop* formam estrutura comum aos códigos do *firmware*. O primeiro é a parte do código que é executada uma vez, já o segundo é o trecho que fica em execução contínua após o término do *setup*.

Comunicar com sensores também é tarefa do microcontrolador. Essa comunicação pode ser feita através de seus pinos, dos quais podem ser: de entrada/saída de propósito geral (do inglês *General Purpose Input/Output* (GPIO)), conversor de analógico para digital (do inglês *Analog to Digital Converter* (ADC)), entre outros. Portanto, os pinos fornecem uma interface de conversa com outros componentes e cada microcontrolador tem seu próprio conjunto, normalmente descritos no *datasheet* do produto. Para exemplificar, a Figura 1 ilustra o esquema de pinos, *pinout*, da correspondente placa.

Figura 1: Diagrama de pinos da placa 555 *timer integrated circuit* mostrando seus 8 pinos e suas funções correspondentes. Os pinos 1 e 8 são pinos de alimentação. 2,4,5 e 6 são pinos de entrada e 3 e 7 são pinos de saída.



Fonte: Wikimedia Commons (2009).

Vale observar que combinações específicas do uso de pinos pode resultar em mau funcionamento da placa. Por isso, é imprescindível a consulta ao *datasheet* do microcontrolador. Esse documento sumariza o desempenho e características importantes do produto. Desenvolvidos pelo fabricante, ele é normalmente usado comercialmente e tecnicamente para que o usuário entenda o papel de cada componente.

Esses fatores demonstram alguns dos desafios enfrentados em projetos envolvendo microcontroladores, afinal lida-se com a aspectos eletrônicos e de software. Não basta a resolução do problema de *design*, deve ser feita uma solução eficiente. Como citado em MARTíNEZ-SANTOS; ACEVEDO-PATINO; CONTRERAS-ORTIZ (2017), isto significa desenvolver o sistema sob restrições de baixo consumo de energia, tempo de execução reduzido e hardware mínimo.

2.3 MQTT

No contexto IoT, a transmissão de mensagens entre diferentes dispositivos é essencial no gerenciamento de sistemas. Por conta de suas aplicações estarem construídas em ambientes de pouca largura de banda, o esquema de comunicação *push* é mais adequado, se comparado ao tradicional *polling* (SONI; MAKWANA, 2017). A necessidade de comunicação com perda mínima de bateria e o uso otimizado de largura de banda levou a IBM a introduzir um protocolo para mensagens chamado MQTT (*Message Queuing Telemetry Transport*) (SONI; MAKWANA, 2017). Já em 2013, o MQTT tornou-se o protocolo padrão da *Organization for the Advancement*

of Structured Information Standards (OASIS).

De acordo com a especificação oficial, Versão 3.1.1 (OASIS, c2015), MQTT é um protocolo de transporte de mensagens baseado no modelo *publish/subscribe*. O protocolo é leve, aberto, simples e de fácil implementação. Essas características o tornam ideal para ambientes limitados, conforme comunicação em contextos *Machine to Machine* (M2M) e IoT (SONI; MAKWANA, 2017). O protocolo é executado sobre a pilha *Transmission Control Protocol/Internet Protocol* (TCP/IP) ou outros protocolos de rede capazes de fornecer conexões bidirecionais ordenadas e sem perdas. Os seções a seguir exploram alguns de seus conceitos básicos.

2.3.1 O padrão *publish/subscribe*

Também conhecido como *pub/sub*, esse padrão fornece uma alternativa ao tradicional paradigma *request/reply*. Enquanto no *request/reply* há comunicação direta entre o remetente e o destinatário, no padrão *publish/subscribe* essa troca de informações é desacoplada pelo uso de um componente intermediário (*broker*). Há clientes que enviam mensagens, *publishers*, e há aqueles que recebem mensagens, *subscribers*. O detalhe é que as partes não se conhecem e não existe contato direto entre elas.

O *broker* forma a outra parte do padrão. Ele é responsável por tratar a conexão entre os outros componentes, filtrando todas as mensagens recebidas e as distribuindo corretamente para os *subscribers*. Há diversos filtros que podem ser aplicados às mensagens e um deles é o baseado em assuntos ou tópicos. O cliente receptor se inscreve em um tópico de interesse através do *broker*, que por sua vez garante que o cliente interessado receberá todas as mensagens destinadas àquele tópico.

2.3.2 Arquitetura

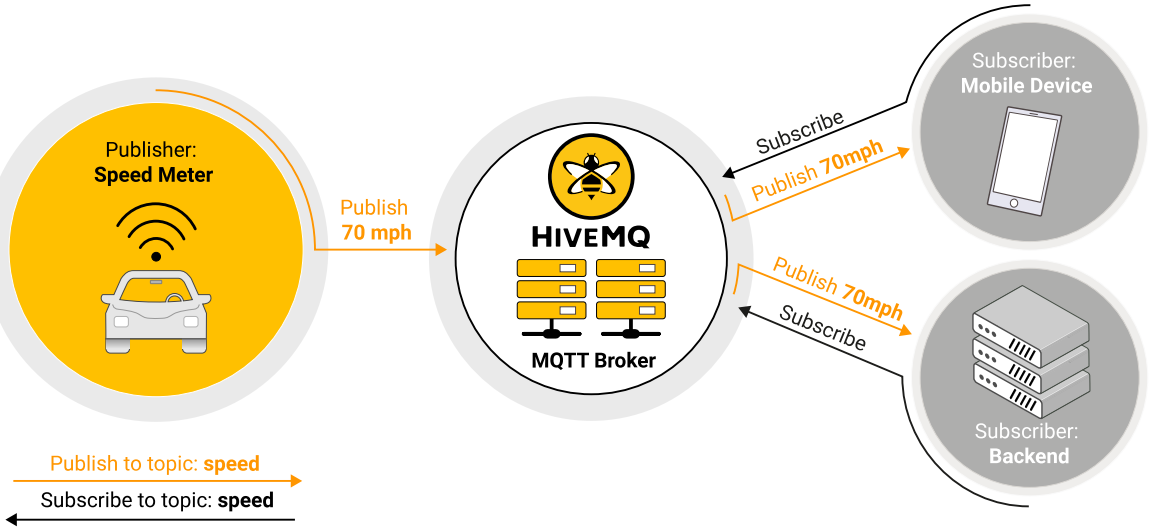
A arquitetura típica do protocolo pode ser dividida em dois principais componentes: o Cliente e o *Broker*. Ambos *publishers* e *subscribers* são clientes MQTT. Essa terminologia é usada para diferenciar os clientes que publicam mensagens dos que estão inscritos para recebê-las, porém o mesmo cliente pode ser as duas coisas. Qualquer dispositivo contendo uma biblioteca MQTT e capaz de conectar-se a um *broker* MQTT é um potencial cliente.

O *broker* tem papel fundamental na comunicação entre os clientes (SONI; MAKWANA, 2017). Ele controla a distribuição de informações e é responsável principalmente por receber todas as mensagens, filtrando-as e as direcionando de acordo com o interesse dos clientes *subscribers*. Autenticação, autorização e capacidade de manter dados de sessões persistentes são outras das funções desse componente.

HiveMQ MQTT Broker (HiveMQ, c2021a) é uma implementação de *broker* existente atualmente e destaca-se por ser de fácil acesso, confiável e escalável. Pelo seu *dashboard* é possível visualizar o tráfego gerado. Em um cenário com o *broker* MQTT sendo público, o

cliente precisará de uma identificação única e evitar tópicos de nome comum para impedir o recebimento de mensagens indesejadas. Existem ainda ferramentas que simulam o cliente MQTT. Além da usual troca de informações, elas fornecem capacidade de testes de carga. Uma delas é a MQTTBox (workswithweb, 2017), disponível como um *Chrome App*. Em outras palavras, é possível montar um ambiente para utilizar o protocolo MQTT sem escrever linhas de código.

Figura 2: Arquitetura *Publish/Subscribe* do MQTT



Fonte: HiveMQ (c2021b)

A Figura 2 apresenta os conceitos chaves abordados e ilustra o fluxo de troca de mensagens de um cenário hipotético. O medidor de velocidade publica no tópico *speed* a velocidade atual do veículo, enquanto o *broker* direciona o conteúdo para os assinantes: o celular e o servidor *backend*.

Outra característica importante do protocolo é que sua conexão é sempre feita entre um cliente e o *broker* e clientes nunca se conectam diretamente. Uma vez a conexão estabelecida, o *broker* a mantém aberta até o cliente ordenar desconexão ou caso haja perda de conectividade. O MQTT ainda conta com recursos adicionais, como *Quality of Service* (QoS), sessões persistentes, *keep alive*, entre outros.

2.4 SOFTWARE ADAPTATIVOS

Em 2001, a IBM expôs sua perspectiva sobre o estado da tecnologia da informação, observando que uma futura crise de complexidade de software seria a maior barreira no setor tecnológico (HORN, 2001). A empresa mostrou exemplos de programas que contém milhões de linhas de código e exigem profissionais qualificados para instalar, configurar e manter. Essa dificuldade serviu de motivação para a concepção da computação autônoma, em que sistemas

podem se gerenciar de acordo com os objetivos de seus administradores (KEPHART; CHESS, 2003).

Seguindo a mesma motivação, houve a necessidade de sistemas de software tornarem-se cada vez mais versáteis, flexíveis e resilientes a fim de acompanhar a evolução constante de sistemas de software de grande escala (CHENG et al., 2009). Essa necessidade pode ser suprida com a auto-adaptação de sistemas, que é a capacidade de ajustar seus comportamentos em resposta às suas percepções do ambiente e do próprio sistema.

De uma maneira geral, software adaptativos podem ser vistos como um sistema de *loop* fechado com *feedback* de si mesmo e do contexto (SALEHIE; TAHVILDARI, 2009). Em termos de componentes, há dois principais: o Sistema Gerenciado e o Sistema Gerenciador. O primeiro refere-se à parte existente do sistema que funciona independentemente de adaptação. Já o segundo, é o componente capaz de adaptar o Sistema Gerenciado a medida que for necessário.

Metas formam outro conceito importante no contexto de software adaptativos. Trata-se de objetivos dos quais o sistema deve atingir. Podem ser associados com o tempo de vida do sistema ou com cenários relacionados. CHENG et al. (2009) apresenta um bom exemplo desse conceito. A meta para um veículo não tripulado, por exemplo, é de evitar colisões.

Uma das formas de construir o módulo adaptativo é através do Gerente Autônomo, cuja função é gerenciar outros componentes de software ou hardware utilizando o *loop* de controle. Esse *loop* automatiza o fluxo de coleta e análise de informações, de criação de um plano de ações e da execução do plano. Essa arquitetura, de monitorar, analisar, planejar e executar (MAPE-K), definida em IBM (2005), possibilita um gerenciamento autônomo simples e estruturado.

Detalhando mais a arquitetura MAPE-K, tem-se o elemento *Monitor* cuja função é fornecer ao *Analyser* mecanismos de coleta, agregação, filtragem e métricas coletadas a partir de um recurso gerenciado. O *Analyser*, por sua vez, analisa as informações obtidas e decide se o sistema precisa de adaptação. É nesse componente que reside a tomada de decisão sobre a adaptação. Caso as mudanças sejam necessárias, o *Planner* cuidará da criação de um plano ou sequência de ações necessárias para atingir as metas. Finalmente, o *Executor* se encarrega de executar as ações previamente estabelecidas. Em um aspecto menos abstrato, o *Monitor* é o elemento que está mais próximo aos sensores, enquanto o *Executor* se situa próximo aos atuadores.

3

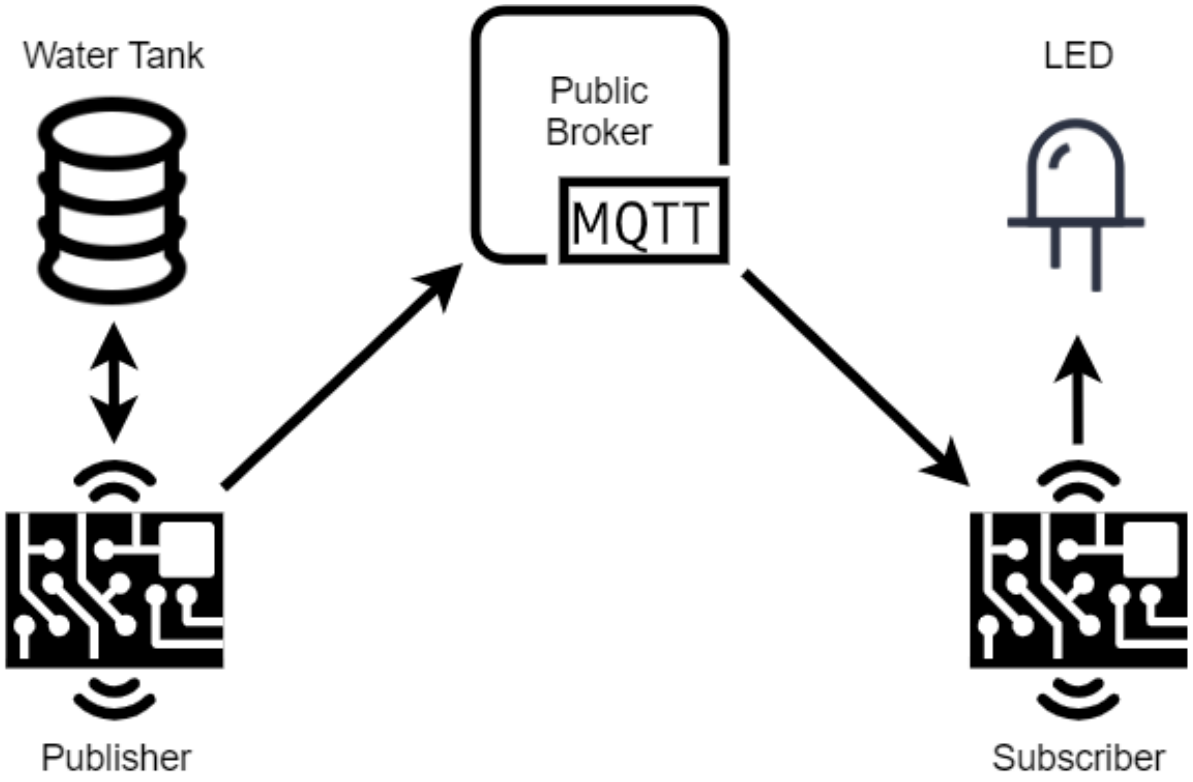
ESPWATER

Este capítulo apresenta a solução proposta, *ESPWater*, usada para realizar o gerenciamento automático de água, especialmente em residências. Também serão apresentados os desafios encontrados e as decisões tomadas ao longo do desenvolvimento do projeto.

3.1 VISÃO GERAL

Antes de apresentar os componentes físicos da solução, vale mostrar uma visão geral da *ESPWater*. A abordagem será *top-down*, explicando primeiramente os aspectos genéricos até chegar aos específicos. Para isso tem-se a Figura 3, a qual indica os fluxos entre os elementos da solução.

Figura 3: Visão geral da *ESPWater*



Fonte: Autoria própria

O *Publisher* e o *Subscriber* são microcontroladores e clientes MQTT. A principal função do *Publisher* é fazer a leitura do nível de água a partir do reservatório (*Water Tank*) e publicar a ordem de acionamento ou desligamento da bomba, representado pelo acendimento do LED, a medida que for necessário. Já o *Subscriber* apenas aguarda as instruções a serem executadas, no caso, acender ou não o LED. *Public Broker* é uma implementação pública do *broker* do protocolo. Ele pode residir na nuvem ou em servidores externos, sendo apenas acessado através da Internet.

A partir da Figura 3 é possível identificar uma das decisões chaves do projeto. Ainda que na Seção 2.3 tenha-se explorado os conceitos sobre o protocolo, a justificativa de sua escolha merece ser observada. Há diversos protocolos capazes de realizar comunicação entre máquinas. MQTT, *Constrained Application Protocol* (CoAP), *Advanced Message Queuing Protocol* (AMQP) e *Hypertext Transport Protocol* (HTTP) são os mais estabelecidos em sistemas IoT. Em NAIK (2017) é feita uma avaliação entre eles, analisando as vantagens e desvantagens de cada um e seus melhores cenários de uso. A escolha do MQTT foi feita a partir desta avaliação. MQTT é o protocolo mais difundido e é utilizado por grandes empresas (NAIK, 2017). O ferramental tecnológico para trabalhar com ele, como descrito na Seção 2.3.2, evidencia sua popularidade. Outro fator decisivo é seu baixo consumo de energia. Juntamente com o CoAP, são os protocolos que melhor se encaixam em dispositivos de recursos limitados e de baixa largura de banda.

3.2 ARQUITETURA

Segundo KRUPITZER et al. (2015), em sistemas adaptativos, há perguntas que precisam ser respondidas para guiar sua implementação: Por que? Quando? Onde? O que? Como?. Todas relacionadas à adaptação.

A começar da pergunta que remete à razão da adaptação, Por que adaptar?, há dois fatores no sistema que precisam de atenção: o nível de água do reservatório e o consumo de energia dos dispositivos. O primeiro traz a necessidade de adaptação para que o reservatório nunca fique vazio ou transborde. Já o segundo revela a adaptação com o propósito de maximizar o tempo de vida útil da bateria usada na alimentação dos microcontroladores.

O momento da adaptação também é fator importante: Quando adaptar?. A resposta da pergunta anterior indica o aspecto proativo da adaptação, ou seja, antes do comportamento indesejado. Neste caso, o nível de água não deve estar próximo aos extremos, definidos posteriormente.

Em terceiro, o questionamento Onde adaptar? aponta o componente do sistema em que a adaptação deverá ser implantada. Na *ESPWater*, ela ocorre tanto no nível de aplicação, no *firmware* das placas, quanto no contexto, acendendo o LED em circunstâncias determinadas.

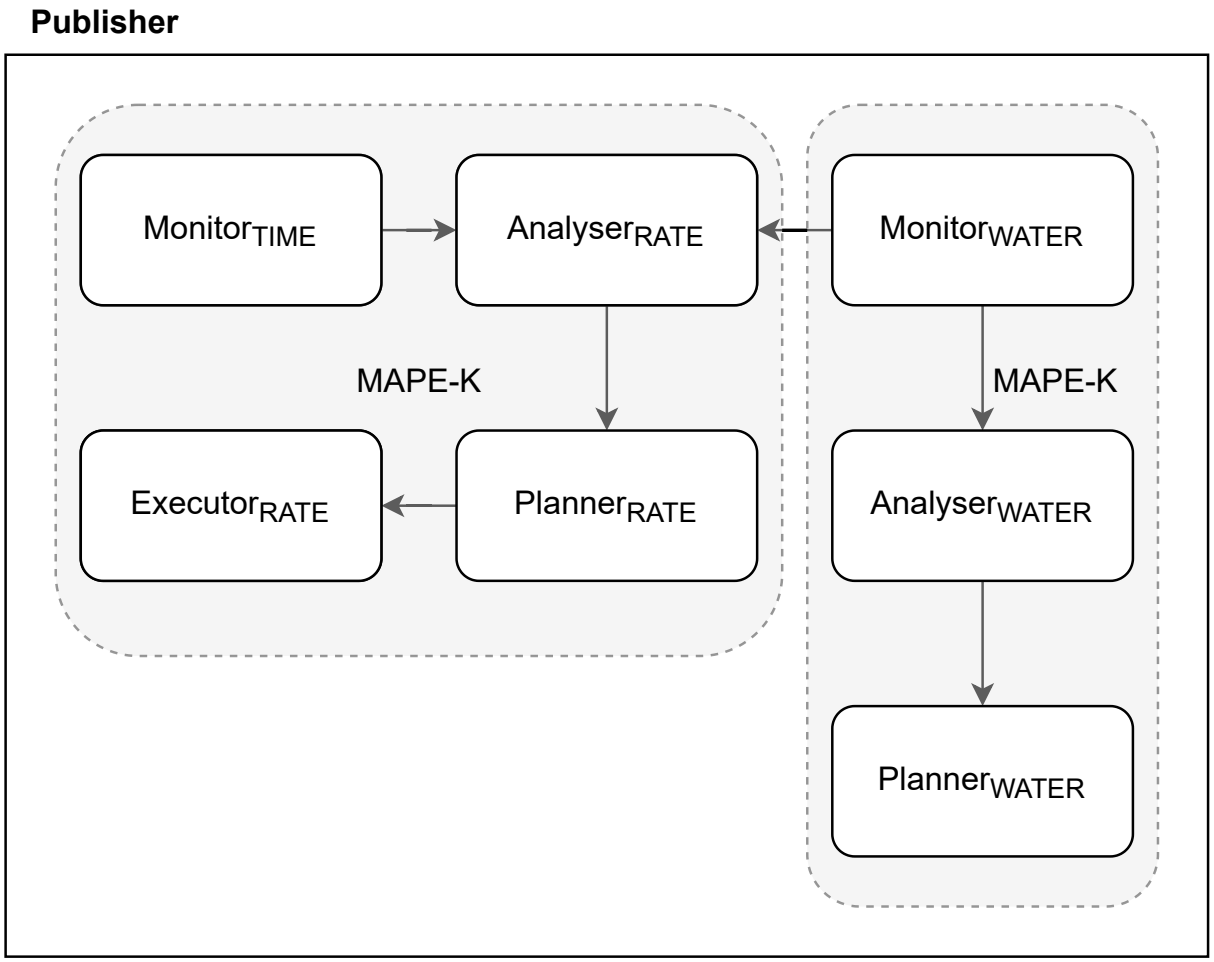
Os recursos do sistema pelos quais a adaptação age revelam a quarta pergunta: O que?. Nesta solução, o estado do LED, se aceso ou desligado, é um deles. Como representa a bomba

d'água, ficará aceso uma vez que o sistema detecte a necessidade de atuar no reservatório e desligado caso contrário. A economia de bateria será feita no código, ainda que de forma sutil, através da taxa de monitoramento dos sensores, melhor explicada na Seção 3.3.

Por fim, a pergunta Como adaptar? reflete-se ao controle de adaptação realizado, responsabilidade do *Publisher*, o componente lógico. Como apresentado na Seção 2.4, esse componente lida com decisões e métricas com o objetivo de modificar o componente gerenciado quando necessário. *ESPWater* implementa a unidade lógica através do *loop* de controle de acordo com o MAPE-K (IBM, 2005). O que permite um *design* descentralizado e de responsabilidade bem definida entre suas partes.

Respondidas as questões, é possível mostrar a arquitetura dos elementos que compõem a *ESPWater*. Começando pelo *Publisher*, que atua como monitor, ficando próximo ao reservatório de água. A Figura 4 mostra a estrutura interna do *Publisher*.

Figura 4: Arquitetura do *Publisher*.



Fonte: Autoria própria

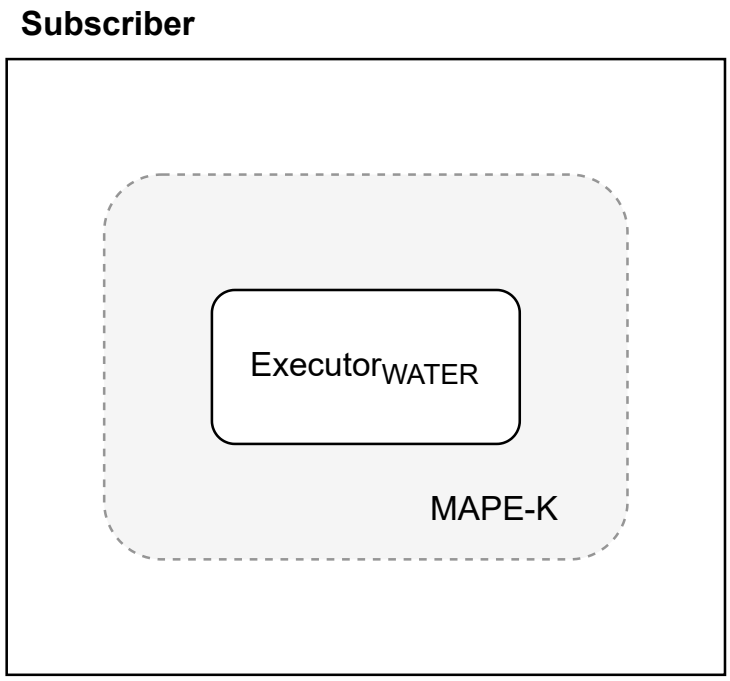
É possível visualizar dois fluxos MAPE-K. O completo, caracterizado pelo *subscript RATE*, tem como função atuar na taxa de monitoramento do sistema. Diminuindo-a em horários específicos e aumentando conforme a altura da água no tanque. A diminuição visa a redução

de corrente por parte da placa. Caso o tanque esteja próximo de estar cheio ou vazio, a taxa de monitoramento é elevada para impedir o transbordamento ou a falta d'água. Embora esteja incompleto, o fluxo *WATER* é responsável por agir no mecanismo de encher, representado pelo LED. O elemento faltante está presente na arquitetura do outro agente, o *Subscriber*.

Enquanto o $Monitor_{TIME}$ coleta o horário local via *Network Time Protocol* (NTP), o $Monitor_{WATER}$ lê o nível de água do reservatório de tempos em tempos. É a taxa de monitoramento que define esses tempos. Quando o *Publisher* não está monitorando ele fica em estado ocioso, economizando bateria. É um passo fundamental para reduzir o consumo, já que neste sistema não é necessário monitorar continuamente (na ordem de microssegundos) os recursos. O $Analyser_{RATE}$ recebe dados dos dois monitores e decide se a taxa precisa de alteração. Caso precise, o $Planner_{RATE}$ criará um plano de acordo com a necessidade de aumento ou diminuição e o $Executor_{RATE}$ o realizará.

Também faz parte do $Monitor_{WATER}$ o envio de informações para o $Analyser_{WATER}$, componente fundamental na decisão de alterar o nível de água. O $Planner_{WATER}$ além de criar a sequência de ações, ele a publica como mensagem ao *broker* MQTT, posteriormente interpretada pelo $Executor_{WATER}$, presente na Figura 5.

Figura 5: Arquitetura do *Subscriber*.



Fonte: Autoria própria

Embora os dois agentes possuam diferentes responsabilidades e tamanhos, eles atuam para adaptar do sistema. Como ambos são clientes, eles precisam da biblioteca MQTT, omitida da arquitetura por questões de simplificação. A mensagem publicada pelo $Planner_{WATER}$ é recebida pelo $Executor_{WATER}$ e ela indica o modo de funcionamento do LED: aceso ou desligado. Essa interferência no LED sugere que o componente seja visto de maneira semelhante à um

atuador (*effector*). No estado atual da solução, a atuação é feita no LED. Entretanto, caso o circuito de acionamento da bomba fosse implementado, não haveria mudanças drásticas no *firmware* da placa pois o trecho de código responsável por acender a lâmpada pode ser o mesmo que aciona a bomba.

3.3 ESPECIFICAÇÕES

Esta seção foca no processo de construção da solução, incluindo questões sobre a taxa de monitoramento. Taxa a qual a *ESPWAter* agrega informações do horário local e do nível de água do reservatório, como mencionado na Seção 3.2. Com a coleta feita, o *Publisher* entra em modo *sleep* para economizar bateria.

Começando pelo aumento da frequência de monitoramento, evento que ocorre quando o volume do reservatório está acima ou abaixo de determinados níveis. Níveis esses encontrados empiricamente. Eles correspondem à 80% e 20%. Ou seja, com o nível superior à 80% ou inferior à 20% a taxa será aumentada. Resta saber o valor desse aumento.

Para isso é preciso definir o valor base ou ideal. O evento de diminuição da frequência de coleta acontece durante intervalo diário específico. De acordo com um estudo realizado em GATO-TRINIDAD; JAYASURIYA; ROBERTS (2011), identificando os principais padrões de consumo de água em residências por hora, há o uso reduzido do recurso entre 00:00 e 05:00. Baseado nisso, a taxa de monitoramento será reduzida entre os horários citados, já que não ocorre consumo intenso de água.

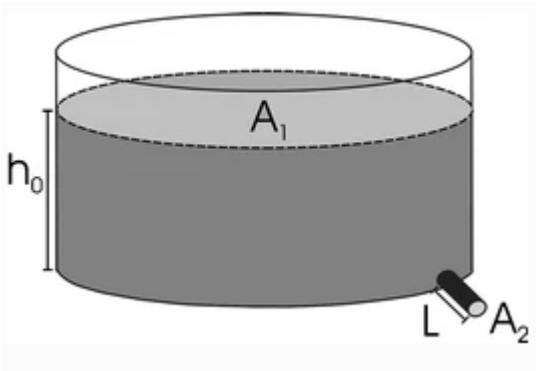
Visando facilitar o desenvolvimento do projeto, uma relação foi estabelecida entre as taxas, podendo ser vista através da Equação 3.1.

$$\begin{aligned} TAXA\_IDEAL &= 2 * TAXA\_MÍNIMA \\ TAXA\_MÁXIMA &= 2 * TAXA\_IDEAL \end{aligned} \tag{3.1}$$

A TAXA_IDEAL é a frequência de monitoramento padrão da *ESPWater*, tendo como unidade leituras por segundo ou requisições por hora. Por se tratar de frequência, outra forma de interpretar o dado é através de seu inverso, resultando no período de inativação entre coletas. Essa inativação é atingida ao colocar o microcontrolador em estado ocioso ou *sleep*.

Enquanto não houver necessidade de adaptação, o sistema irá monitorar baseado na frequência ideal. Caso haja necessidade, a taxa irá variar de acordo com a Equação 3.1. Portanto, bastou encontrar apenas uma variável e derivar as demais. Neste caso, a taxa seminal escolhida foi a TAXA_MÍNIMA. O tempo de drenagem do tanque influenciou no valor da taxa, que pôde ser obtido através do Teorema de Torricelli (FIORILLO, 2011). Segundo esse teorema, em um reservatório de Torricelli a viscosidade é desconsiderada dos cálculos e essa classificação ocorre por causa da área A_1 ser muito maior que a área A_2, ilustrado pela Figura 6.

Figura 6: Reservatório cilíndrico, com área de base A_1. A descarga ocorre através do orifício de seção A_2, a partir de uma altura de água inicial h_0.



Fonte: FIORILLO (2011)

Uma caixa d'água comum, de 1000 litros (NBR 14799), pode ser transformada em cilindro conforme a Figura 6. Basta manter a capacidade, a área da base e modificar sua altura. Logo, a caixa com diâmetro da base 1.16m precisa da altura inicial do líquido, h_0, sendo 0.95 m a fim de conservar a mesma quantidade de conteúdo. A fórmula que descreve a altura da água em qualquer momento durante a drenagem e a fórmula que determina o tempo de escoamento total podem ser vistas nas Equações 3.2 e 3.3:

$$h(t) = (\sqrt{h_0} - \frac{kt}{2})^2, \; onde \; k = \frac{A_2}{A_1}\sqrt{2g} \tag{3.2}$$

$$t_{final} = \frac{2\sqrt{h_0}}{k} \tag{3.3}$$

As variáveis das equações são referentes à Figura 6. A constante de gravidade na Terra é caracterizada por g e t_{final} representa o tempo de drenagem total. Ainda em relação à Equação 3.2, nota-se um comportamento não linear no escoamento do fluido. Outra propriedade fundamental é a proporção da altura $h(t)$ em relação ao volume atual do líquido. Reduzir 20% da altura inicial é equivalente a reduzir a mesma porcentagem no volume. Então, fazendo a substituição de variáveis pelos parâmetros declarados anteriormente e admitindo que o orifício pelo qual a água sai tem 50mm de diâmetro, chega-se aos seguintes valores:

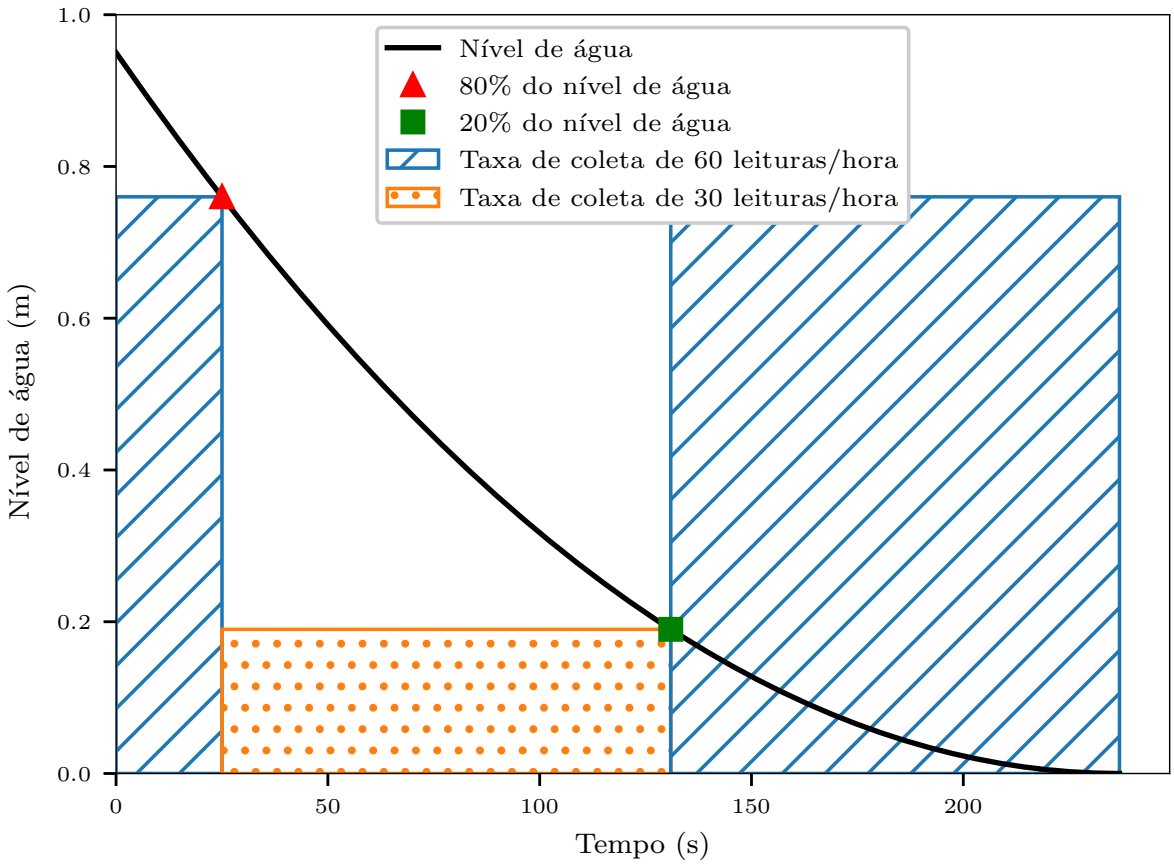
$$\begin{aligned} A_1 &\approxeq 1.056m^2 \\ A_2 &\approxeq 0.002m^2 \\ h_0 &\approxeq 0.95m \\ t_{final} &\approxeq 237s \\ t_{80\%} &\approxeq 25s \\ t_{20\%} &\approxeq 131s \end{aligned} \tag{3.4}$$

As presentes porcentagens denotam os níveis de altura da água em que a *ESPWater* realiza a adaptação da taxa de monitoramento. Estando completamente cheio e iniciando o escoamento, leva cerca de 25 segundos para o reservatório perder 20% do conteúdo. Da mesma forma, leva-se aproximadamente 131 segundos para redução de 80% e 237 segundos para o esvaziamento total.

Com os tempos calculados, o valor da TAXA_MÍNIMA foi escolhido. Nesse cenário de escoamento, ainda que irreal, o sistema precisaria de pelo menos uma leitura a cada 237 segundos para não deixar o reservatório vazio. Esse é o valor da menor frequência de coleta da *ESPWater*, aproximadamente 15 leituras por hora, aplicado durante a madrugada através da adaptação. A TAXA_IDEAL representa cerca de 30 leituras por hora e a frequência máxima 60 leituras por hora.

Estabelecidos os principais parâmetros do sistema, cabe apresentar como eles são combinados na aplicação. As Figuras 7 e 8 mostram as combinações desses parâmetros.

Figura 7: Adaptação da taxa de monitoramento em relação ao nível de água - Drenagem do reservatório. Os valores são referentes à uma caixa de 1000 L, com diâmetro da base = 1.16 m, diâmetro da tubulação de saída = 50 mm e altura de 0.95 m.

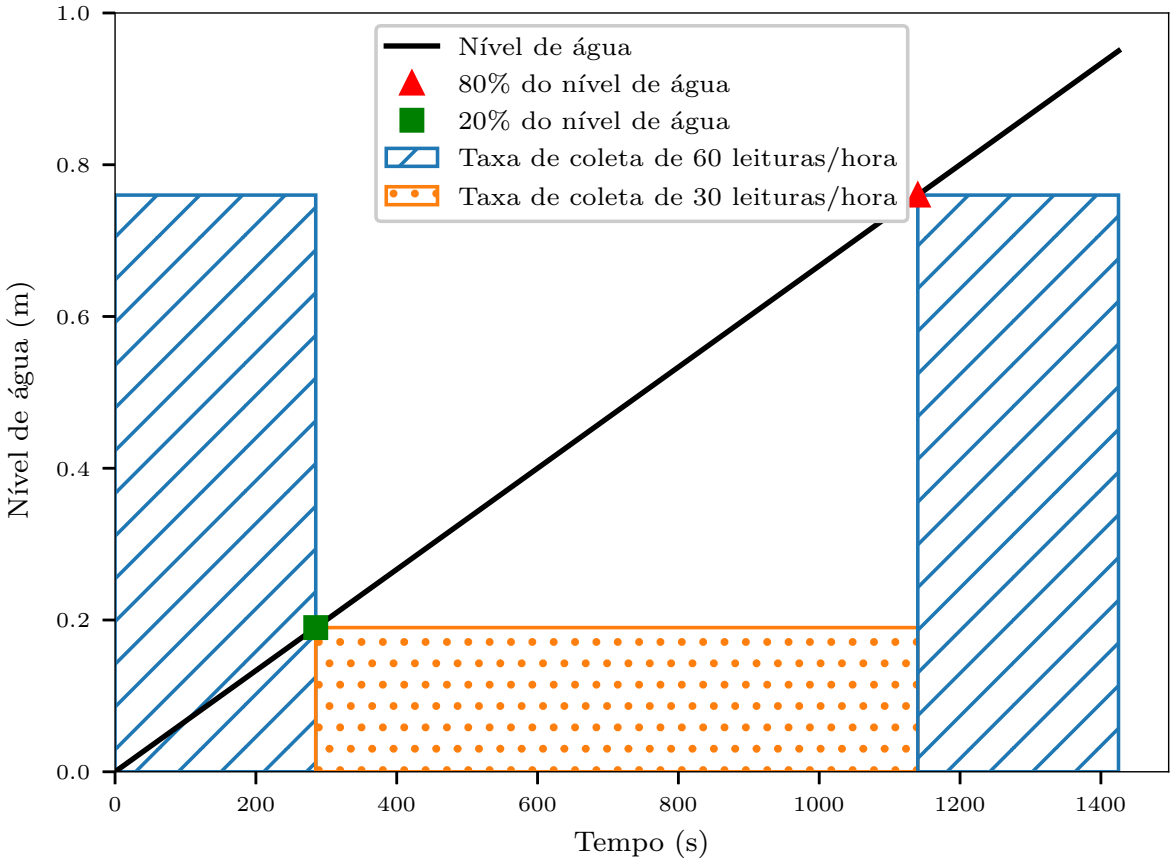


Fonte: Autoria própria

As áreas hachuradas destas duas figuras representam as diferentes taxas de monitoramento obtidas através da adaptação. Elas obedecem à relação fornecida na Equação 3.1. A

hachura com linhas diagonais indica a TAXA_MÁXIMA, enquanto a hachura com pontos denota a TAXA_IDEAL. A multiplicação da taxa pelo tempo em que ela foi ativada define o número de leituras realizado pelos sensores.

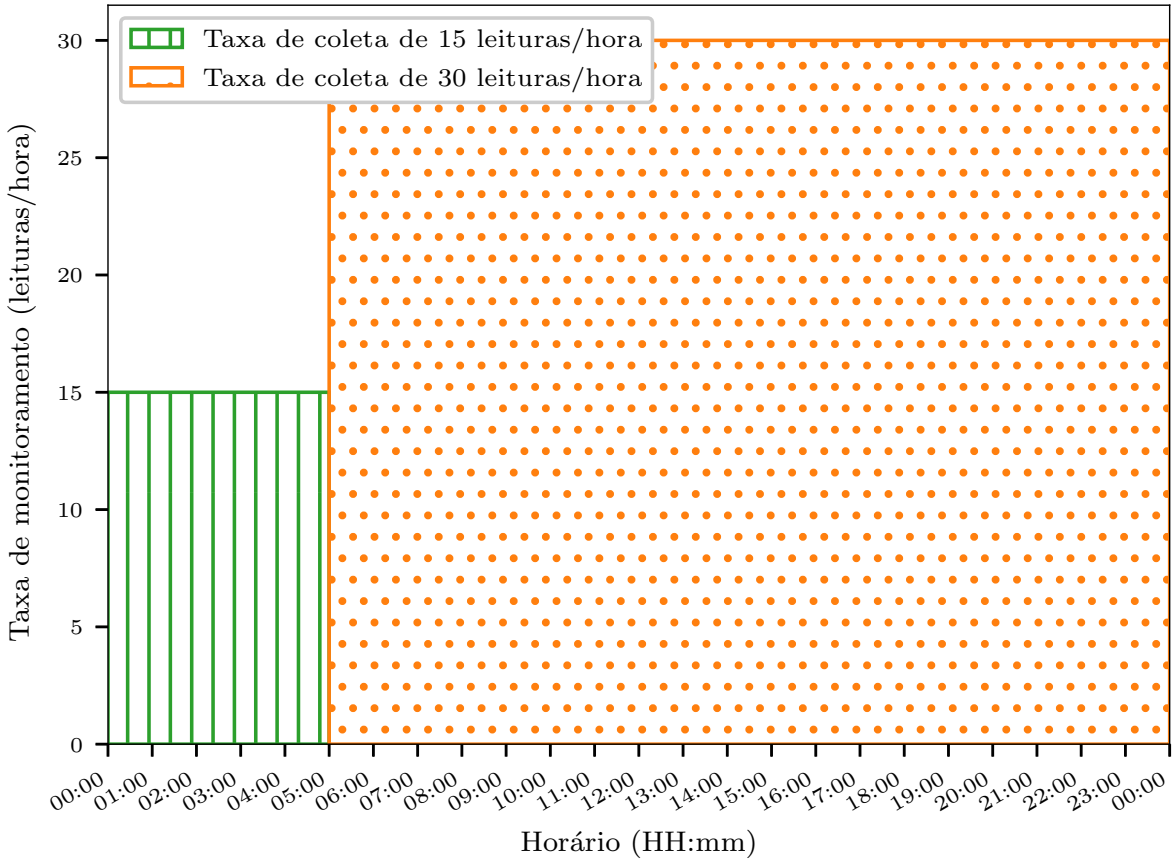
Figura 8: Adaptação da taxa de monitoramento em relação ao nível de água - Acionamento da bomba d'água. Os valores são referentes à uma caixa de 1000 L, com diâmetro da base = 1.16 m, diâmetro da tubulação de saída = 50 mm e altura de 0.95 m. A vazão da bomba é de 40 L/min.



Fonte: Autoria própria

Embora a bomba d'água não esteja presente na versão atual da *ESPWater*, a Figura 8 ilustra como a adaptação iria ocorrer caso o mecanismo de encher estivesse ativado.

Figura 9: Adaptação da taxa de monitoramento em relação ao horário.



Fonte: Autoria própria

Da mesma maneira que as Figuras 7 e 8, a Figura 9 também mostra a alteração do sistema. Desta vez influenciada por horários do dia. A área hachurada com linhas verticais compõe a TAXA_MÍNIMA.

Finalmente, é importante destacar a precedência do $Monitor_{WATER}$ sobre o $Monitor_{TIME}$ com relação ao $Analyser_{RATE}$. Em um cenário durante a madrugada, com o nível de água inferior à 20%, o sistema irá atuar com taxa de monitoramento máxima e não mínima, visto que a precisão é fator mais determinante que a economia de bateria. Erros de precisão podem resultar em transbordamento.

3.4 COMPONENTES ELETRÔNICOS

A Figura 3 apresenta como os componentes eletrônicos (atuais e futuros) irão se encaixar. São os dispositivos que atuam como os agentes *Publisher* e *Subscriber*, além do sensor responsável pela leitura do nível de água.

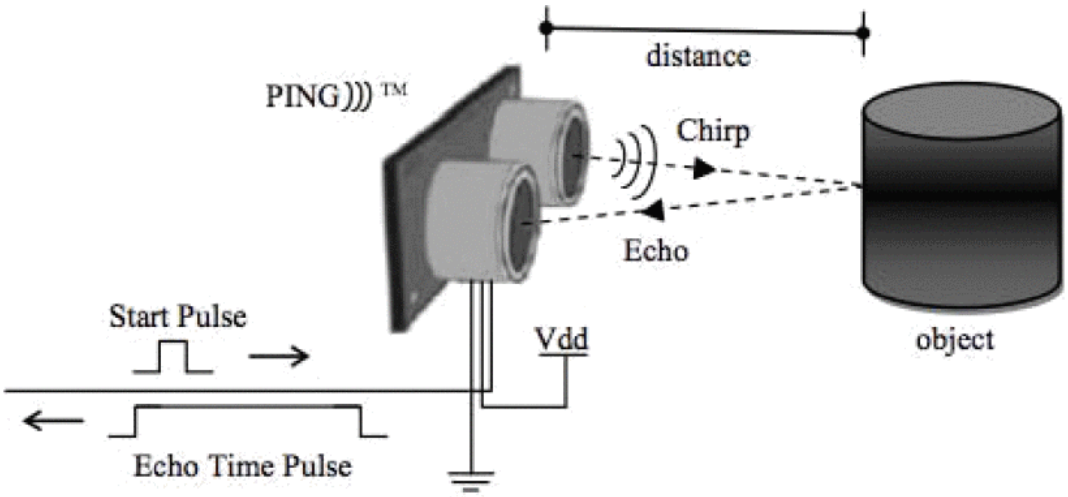
A placa de desenvolvimento escolhida foi a NodeMCU ESP8266-12 V2 (ESPRESSIF, c2021): uma é utilizada como *Publisher* e outra como *Subscriber*. Por ter recursos de rede Wi-Fi integrados em seu chip, custo reduzido, baixo consumo de energia e compatibilidade com

plataformas populares, como *Arduino Software* (IDE), a escolha desse microcontrolador acabou sendo ideal para o projeto. Outro fator relevante é sua difusão em sistemas IoT e suporte nas comunidades de programadores, o que acelera a resolução de *bugs* e problemas.

Diferente de soluções que fazem a leitura do nível de água através de sensores mecânicos como resistivo, capacitivo ou sensor magnético, mencionadas em SARASWATI; KUANTAMA; MARDJOKO (2012), a *ESPWater* utiliza sensor ultrassônico. Em comparação aos outros sensores, ele não precisa de contato direto com a água, estendendo assim, sua vida útil. Para essa função o dispositivo utilizado foi o HC-SR04, capaz de medir precisamente distâncias de até 4m.

O sensor ultrassônico detecta a distância de objetos ou fluidos através da transmissão de ondas ultrassônicas. Ele possui um transmissor e um receptor. É possível medir a duração do momento em que a onda é transmitida até o momento em que ela é refletida e recebida pelo receptor, como mostra a Figura 10. O tempo é parâmetro chave para calcular a distância.

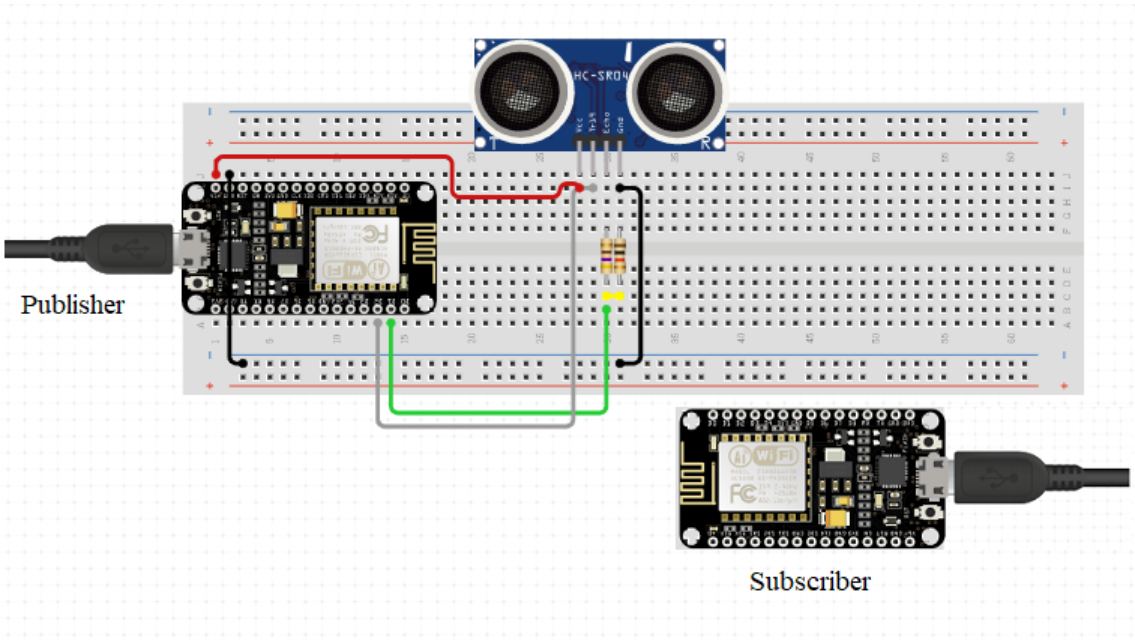
Figura 10: Teoria de funcionamento do sensor ultrassônico.



Fonte: SARASWATI; KUANTAMA; MARDJOKO (2012)

O papel do microcontrolador é ativar o sensor para iniciar a transmissão da onda e converter os dados recebidos em valores de distância. A ativação é feita ao fornecer 10 μs de pulso *HIGH* ao pino *trigger* do sensor. A Figura 11 mostra os elementos da versão atual da *ESPWater* e como eles estão interligados.

Figura 11: Versão atual da *ESPWater*.



Fonte: Autoria própria

Ambas placas são alimentadas pela porta *Universal Serial Bus* (USB) 2.0 de um computador. Apesar do circuito da imagem utilizar resistores para ligação da ESP8266 com o sensor HC-SR04, na prática esses componentes resistivos não foram necessários.

4

AVALIAÇÕES

Neste capítulo é feita a avaliação da *ESPWater* por meio de seu módulo adaptativo. A Seção 4.1 descreve os objetivos da avaliação e a Seção 4.2 descreve os parâmetros usados na geração dos resultados. Por fim, a Seção 4.3 apresenta os resultados e suas análises.

4.1 OBJETIVO

O principal objetivo da avaliação é descobrir o impacto da adaptação do sistema em relação ao consumo de energia dos dispositivos. A fim de alcançá-lo, será feito uma abordagem analítica, embora o ideal para medição de corrente seja através de um osciloscópio. Essa abordagem requer aferição de tempo em trechos de código dos dispositivos. Com os tempos aferidos, basta encontrar nominalmente o uso de corrente em cada parte do código, calcular a carga e comparar os resultados com e sem adaptação.

4.2 EXPERIMENTOS

O tempo é calculado na aplicação, a qual estará em execução contínua por meio de seu *loop* de controle, explicado na Seção 2.4. Residindo no *firmware* do *Publisher*, esse *loop* pode ser decomposto em dois estados: ativação e inativação. O dispositivo fica ativo enquanto utiliza os sensores e ocioso caso contrário. O período de inativação (*sleep mode*) é definido pela taxa de monitoramento. Se a taxa equivale à uma leitura a cada 10 segundos, por exemplo, o *Publisher* dormirá por 10 segundos antes da próxima leitura. Esse tempo será chamado de T_{sleep}, já o outro será o T_{active}. Portanto, o tempo de um ciclo do *loop* é dado pela Equação 4.1:

$$T_{cycle} = T_{active} + T_{sleep} \tag{4.1}$$

Para encontrar o T_{active} foram feitas 30 medições e calculada a média dos valores. A medição foi realizada pelo próprio microcontrolador e disponibilizada através da comunicação serial. O cronômetro iniciava antes da primeira instrução do *loop* de controle e terminava antes de iniciar o modo *sleep*. Durante o experimento, a placa estava sendo alimentada por um

computador via porta USB 2.0, assim como mostra a Figura 11. A Tabela 1 mostra os tempos calculados e seus cenários de uso.

Tabela 1: Decomposição de tempos do *loop* de controle do *Publisher* em diferentes cenários.

Cenário	$T_{active}(ms)$	$T_{sleep}(s)$	Ciclos por hora
Entre 00:00 e 05:00	4985	237	15
Nível da água superior à 80% ou inferior à 20%	4985	118	60
Sem necessidade de adaptação	4985	60	30

Observando os valores obtidos na Tabela 1 é possível dizer que o tempo do estado de ativação, T_{active}, independe do cenário, o que não é surpresa. Assim também é o tempo do estado de inativação, T_{sleep}, sendo proporcional à taxa de monitoramento. Da mesma forma, percebe-se que o tempo de inativação é muito maior que o de ativação. A quantidade de ciclos por hora refere-se às execuções do *loop* de controle no intervalo de uma hora. A conclusão desse passo permite o cálculo aproximado da carga utilizada pelo dispositivo.

Segundo AKINTADE; YESUFU; KEHINDE (2019), a ESP8266 consome em média 70.89 mA quando está ativa. Seu *datasheet* informa o consumo de 20 μA, enquanto em estado de sono profundo (*Deep-sleep*), porém ao sair desse modo acaba sendo necessário reconectar-se à rede Wi-Fi e inicializar o ambiente, influenciando assim o T_{active}. A partir desses dados é possível encontrar analiticamente a carga em função do tempo, como calculado pela Equação 4.2.

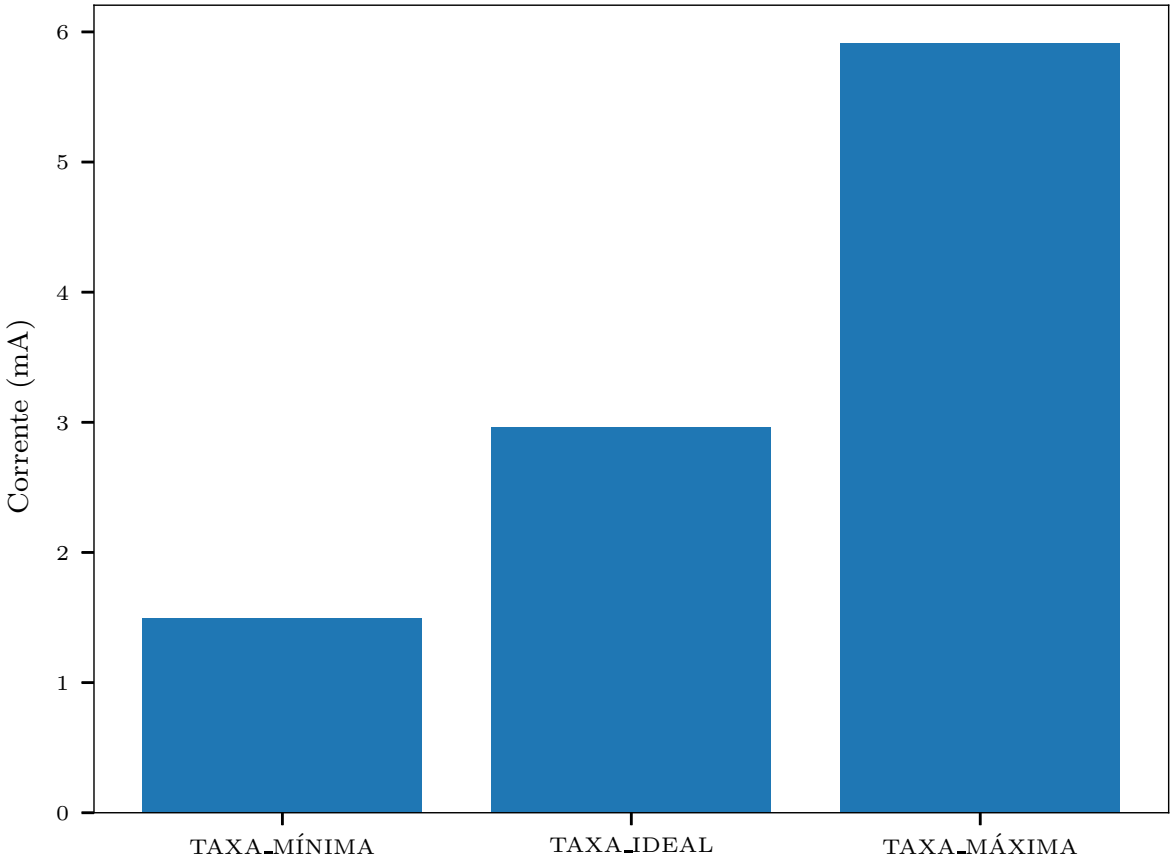
$$Q(t) = 70.89T_{active} + 0.02T_{sleep} \tag{4.2}$$

O resultado da equação pode ser dado em mAh a fim de comparar o consumo em relação à uma bateria padrão. A contribuição da parcela envolvendo o T_{active} é muito superior a outra parcela, ainda que o tempo de modo *sleep* seja bem maior que o de ativação.

4.3 RESULTADOS E DISCUSSÕES

Com os cenários e valores apontados na Tabela 1 uma comparação pode ser feita acerca da *ESPWater*. Sem a adaptação, o sistema atuaria em sua frequência ideal o tempo todo. Com a adaptação, entre meia noite e cinco da manhã, a frequência de coleta se reduziria à TAXA_MÍNIMA. Logo, esses dois cenários podem ser analisados através de seus consumos, de acordo com a Equação 4.2. A começar pela Figura 12, a qual apresenta o uso de corrente durante uma hora com as diferentes taxas empregadas.

Figura 12: Quantidade de corrente usada em um intervalo de 1 hora em diferentes cenários.

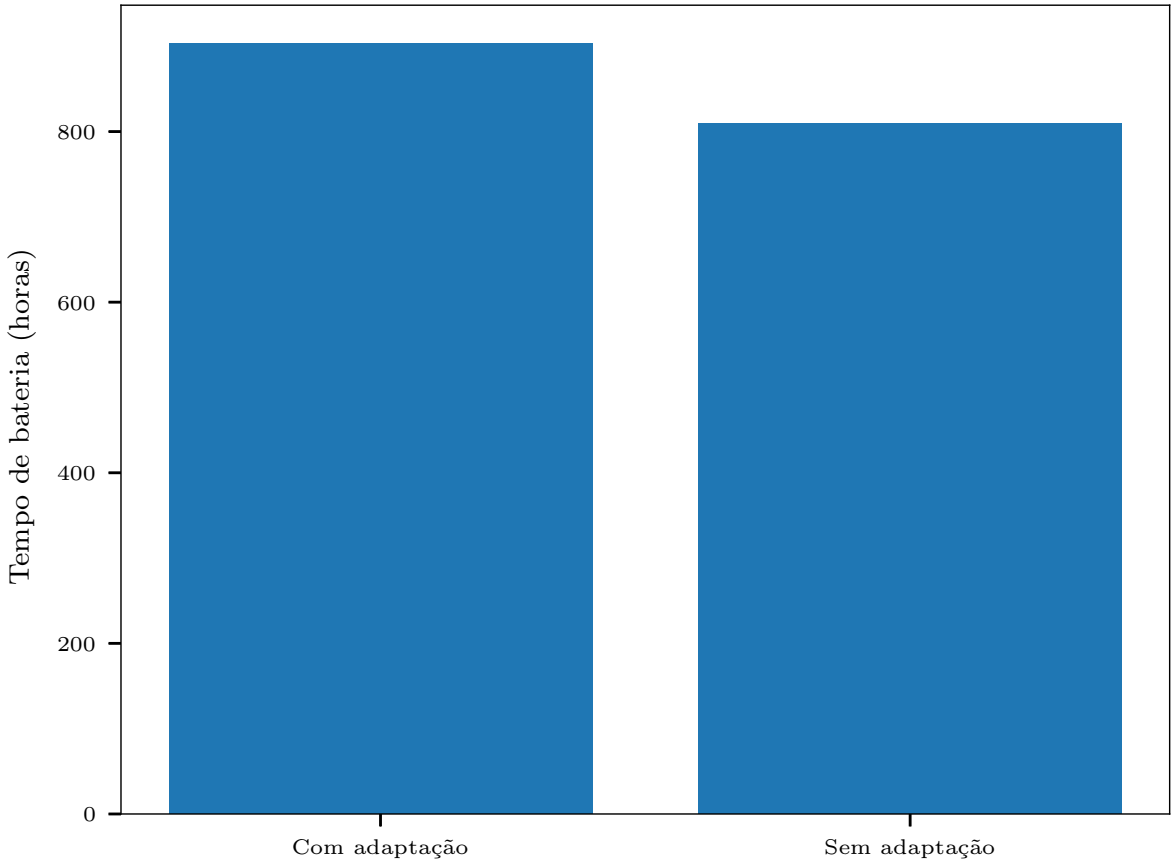


Fonte: Autoria própria

Observando a Figura 12, percebe-se uma semelhança dos valores com a relação estabelecida na Equação 3.1. O uso de corrente ao empregar a TAXA_IDEAL é aproximadamente o dobro de quando empregado a TAXA_MÍNIMA. E metade do gasto pela TAXA_MÁXIMA. Resultado esperado, pois o tempo de ativação, T_{active}, além de se manter o mesmo independente da taxa, é o componente principal na fórmula da carga, conforme citado na Seção 4.2.

A fim de realizar a comparação entre os cenários com e sem adaptação da *ESPWater*, utilizou-se uma bateria de capacidade 2400 mAh para execução dos experimentos. A faixa crítica da adaptação ocorre entre os horários 00:00 e 05:00, ou seja, durante as 5 horas iniciais do dia. Nessa faixa é utilizada a TAXA_MÍNIMA. Nas horas restantes, admitindo que o reservatório não perca volume e mantenha-se em um nível normal (volume de água inferior à 80% e superior à 20%), foi empregado a TAXA_IDEAL, correspondente a 30 leituras por hora. A Figura 13 apresenta os resultados usando as duas abordagens.

Figura 13: Tempo de uma bateria de 2400 mAh aplicada aos diferentes cenários no *Publisher*.



Fonte: Autoria própria

Apesar da Figura 13 não indicar grandes diferenças, o uso da adaptação foi capaz de estender o tempo de vida da bateria em aproximadamente 11%, uma vez utilizada para alimentar o *Publisher*. Valor relevante no contexto de sistemas IoT, em que toda redução de consumo de energia é importante. Vale destacar que para os cálculos realizados para obtenção dos resultados apresentados na Figura 13, foi necessário desconsiderar diversos fatores que impactam na descarga da bateria, como temperatura, material, entre outras coisas.

Não é tarefa simples avaliar soluções IoT. Por exemplo, SARASWATI; KUANTAMA; MARDJOKO (2012) e WADEKAR et al. (2016) não fazem avaliação da solução. O esforço maior está em descrever o funcionamento e componentes do sistema. Entretanto, para este projeto, foi possível fazer uma avaliação a respeito da adaptação. Embora tenha sido uma modelagem analítica, a contribuição positiva pôde ser vista através dos gráficos e alcançada com a *ESPWater*.

5

CONCLUSÃO E TRABALHOS FUTUROS

Este capítulo apresenta a conclusão do trabalho, algumas limitações encontradas no decorrer do desenvolvimento da solução e sugestões de trabalhos futuros.

5.1 CONCLUSÃO

A crise hídrica no Brasil acendeu alertas relacionados ao uso consciente de água e de energia. Para o consumo inteligente da água, especialmente em residências, é fundamental evitar o desperdício. Principalmente, quando ele ocorre por uso não controlado e gerenciado do recurso. Este problema pode ser minimizado com o emprego da boia elétrica ou de soluções IoT capazes de gerenciar o fluido de forma automática.

Este trabalho se propôs a implementar a *ESPWater* para possibilitar um gerenciamento automático de água através de microcontroladores e sensores, contando com adaptações para reduzir o consumo de energia dos dispositivos. Diferente de outras abordagens no contexto IoT, a distância entre o reservatório e o mecanismo de atuação não compromete a *ESPWater*. Como a aplicação está distribuída, o único fator relevante é a conexão com internet.

Através das avaliações analíticas realizadas, foi possível destacar o impacto positivo da adaptação em relação à descarga de bateria. Constatou-se que o módulo adaptativo foi capaz de estender o tempo de uma bateria de 2400 mAh em aproximadamente 11%, se usada para alimentar o *Publisher*. Esse valor está diretamente relacionado com as decisões tomadas envolvendo as taxas de monitoramento. Diferentes taxas resultariam em diferentes ganhos.

Embora o circuito de acionamento da bomba d'água esteja ausente, essa ausência não compromete a solução proposta. No estado atual, o acionamento foi representado por um LED, indicando o estado da bomba. A *ESPWater* mostrou a construção de um ambiente para gerenciamento automático de água de baixo custo, flexível e capaz de atender a um dos principais requisitos envolvendo componentes eletrônicos: consumo reduzido de bateria.

5.2 LIMITAÇÕES

O desenvolvimento da *ESPWater* envolveu o uso de placas de ensaio (*breadboard* ou *protoboard*) na integração dos componentes eletrônicos. Por ser um ambiente sensível, erros poderiam resultar em comprometimento das placas, prejudicando a solução. Além disso, o estado atual da solução está distante de um cenário de produção. Cenário que exigiria a construção de placas, estudo sobre baterias e até revestimentos para os componentes. O envolvimento de água e eletricidade dificulta a realização de testes reais, como por exemplo o monitoramento do uso de água na residência durante um dia. A medição de carga por meio de fórmulas é aceitável, porém o ideal seria o uso de instrumentos para medir o consumo da *ESPWater*.

5.3 TRABALHOS FUTUROS

A primeira ideia de trabalho futuro seria implementar o circuito de acionamento de bomba d'água. Com isso, a *ESPWater* alcançaria um nível mais completo em direção ao gerenciamento automático de água. O monitoramento do nível de água poderia ser apresentado ao usuário por meio de uma aplicação Web ou *mobile*. No aspecto arquitetural, as relações e valores acerca das taxas de monitoramento poderiam ser diferentes. Indicando até um trabalho de otimização de taxas para minimizar o uso de corrente. Há de se pensar em uma adaptação mais elaborada envolvendo outros recursos, tanto para o *Publisher* quanto o *Subscriber*. De exemplo, a adaptação poderia ocorrer de acordo com o consumo médio de água da residência. Ainda que seja de difícil instrumentação por envolver a rede, também pode ser feito um estudo qualitativo da adaptação através de sua latência.

REFERÊNCIAS

AID. **ANEEL define que bandeira tarifária de julho custará R$ 9,492 a cada 100 kWh**. [Online; acessado em 01/08/2021], URL: https://www.aneel.gov.br/web/guest/sala-de-imprensa-exibicao-2/-/asset_publisher/zXQREz8EVlZ6/content/id/22368525.

AKINTADE, O.; YESUFU, T.; KEHINDE, L. Development of Power Consumption Models for ESP8266-Enabled Low-Cost IoT Monitoring Nodes. **Advances in Internet of Things**, [S.l.], v.09, p.1–14, 01 2019.

AXELSON, J. **Microcontroller Idea Book**: circuits, programs and applications featuring the 8052-basic single-chip computer. [S.l.]: Lakeview Research, 1997.

CHENG, B. et al. Software Engineering for Self-Adaptive Systems: a research roadmap. In: SOFTWARE ENGINEERING FOR SELF-ADAPTIVE SYSTEMS. **Anais...** [S.l.: s.n.], 2009.

CIRILO, J. Crise hídrica: desafios e superação. **Revista USP**, [S.l.], p.45, 09 2015.

ESPRESSIF. **ESP8266**. [Online; acessado em 26/07/2021], URL: https://www.espressif.com/en/products/socs/esp8266.

FAME. **Chave Boia - Manual de instruções para instalação e uso**. [Online; acessado em 01/08/2021], URL: https://www.fame.com.br/uploads/produtos/produtos/1378/manual_produto_path_pt.pdf.

FIORILLO, F. Tank-reservoir drainage as a simulation of the recession limb of karst spring hydrographs. **Hydrogeology Journal**, [S.l.], v.19, p.1009–1019, 08 2011.

GATO-TRINIDAD, S.; JAYASURIYA, N.; ROBERTS, P. Understanding urban residential end uses of water. **Water science and technology : a journal of the International Association on Water Pollution Research**, [S.l.], v.64, p.36–42, 07 2011.

gov.br. **Governo Federal lança campanha sobre consumo consciente de energia e água**. [Online; acessado em 01/08/2021], URL: https://www.gov.br/casacivil/pt-br/assuntos/noticias/2021/junho/governo-federal-lanca-campanha-sobre-consumo-consciente-de-energia-e-agua.

HiveMQ. **HiveMQ MQTT Broker**. [Online; acessado em 20/07/2021], URL: http://www.mqtt-dashboard.com/.

HiveMQ. **MQTT Publish / Subscribe Architecture**. [Online; acessado em 20/07/2021], URL: https://www.hivemq.com/img/svg/mqtt-publish-subscribe.svg.

HORN, P. **Autonomic Computing**: IBM's Perspective on the State of Information Technology. [S.l.: s.n.], 2001.

An Architectural Blueprint for Autonomic Computing. [S.l.]: IBM, 2005.

JACOBI, P. R.; CIBIM, J.; SOUZA LEãO, R. de. Crise hídrica na Macrometrópole Paulista e respostas da sociedade civil. **Estudos Avançados**, [S.l.], v.29, p.27–42, 08 2015.

KEPHART, J.; CHESS, D. The vision of autonomic computing. **Computer**, [S.l.], v.36, n.1, p.41–50, 2003.

KRUPITZER, C. et al. A survey on engineering approaches for self-adaptive systems. **Pervasive Mob. Comput.**, [S.l.], v.17, p.184–206, 2015.

MARTíNEZ-SANTOS, J. C.; ACEVEDO-PATINO, O.; CONTRERAS-ORTIZ, S. H. Influence of Arduino on the Development of Advanced Microcontrollers Courses. **IEEE Revista Iberoamericana de Tecnologias del Aprendizaje**, [S.l.], v.12, n.4, p.208–217, 2017.

MCEWEN, A.; CASSIMALLY, H. **Designing the Internet of Things**. 1st.ed. [S.l.]: Wiley Publishing, 2013.

NAIK, N. Choice of effective messaging protocols for IoT systems: mqtt, coap, amqp and http. In: IEEE INTERNATIONAL SYSTEMS ENGINEERING SYMPOSIUM (ISSE), 2017. **Anais...** [S.l.: s.n.], 2017. p.1–7.

OASIS. **MQTT Version 3.1.1 Plus Errata 01**. [Online; acessado em 20/07/2021], URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

ORNES, S. Core Concept: the internet of things and the explosion of interconnectivity. **Proceedings of the National Academy of Sciences**, [S.l.], v.113, n.40, p.11059–11060, 2016.

REZA, S.; TARIQ, S.; REZA, S. M. Microcontroller Based Automated Water Level Sensing and Controlling: design and implementation issue. **Lecture Notes in Engineering and Computer Science**, [S.l.], v.2186, 10 2010.

SALEHIE, M.; TAHVILDARI, L. Self-Adaptive Software: landscape and research challenges. **ACM Trans. Auton. Adapt. Syst.**, New York, NY, USA, v.4, n.2, May 2009.

SARASWATI, M.; KUANTAMA, E.; MARDJOKO, P. Design and Construction of Water Level Measurement System Accessible through SMS. In: SIXTH UKSIM/AMSS EUROPEAN SYMPOSIUM ON COMPUTER MODELING AND SIMULATION, 2012. **Anais...** [S.l.: s.n.], 2012. p.48–53.

SIDDULA, S. S.; BABU, P.; JAIN, P. C. Water Level Monitoring and Management of Dams using IoT. In: INTERNATIONAL CONFERENCE ON INTERNET OF THINGS: SMART INNOVATION AND USAGES (IOT-SIU), 2018. **Anais...** [S.l.: s.n.], 2018. p.1–5.

SONI, D.; MAKWANA, A. A survey on mqtt: a protocol of internet of things (iot). In: INTERNATIONAL CONFERENCE ON TELECOMMUNICATION, POWER ANALYSIS AND COMPUTING TECHNIQUES (ICTPACT-2017). **Anais...** [S.l.: s.n.], 2017. v.20.

WADEKAR, S. et al. Smart water management using IOT. In: INTERNATIONAL CONFERENCE ON WIRELESS NETWORKS AND EMBEDDED SYSTEMS (WECON), 2016. **Anais...** [S.l.: s.n.], 2016. p.1–4.

Wikimedia Commons. **Pinout of 555 single timer**. [Online; acessado em 16/07/2021], URL: https://upload.wikimedia.org/wikipedia/commons/c/c7/555_Pinout.svg.

workswithweb. **MQTTBox**. [Online; acessado em 20/07/2021], URL: https://chrome.google.com/webstore/detail/mqttbox/kaajoficamnjijhkeomgfljpicifbkaf.