UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

JOÃO LAURO BESERRA DA COSTA

SISTEMA DE COLETA DE DADOS POR MEIO DE REDE DE SENSORES SEM FIO APLICADO AO MONITORAMENTO DE ESTRUTURAS NA CONSTRUÇÃO CIVIL

RUSSAS

2022

JOÃO LAURO BESERRA DA COSTA

SISTEMA DE COLETA DE DADOS POR MEIO DE REDE DE SENSORES SEM FIO APLICADO AO MONITORAMENTO DE ESTRUTURAS NA CONSTRUÇÃO CIVIL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Ms. Filipe Maciel Roberto

RUSSAS

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C873s Costa, João Lauro Beserra da.
Sistema de coleta de dados por meio de rede de sensores sem fio aplicado ao monitoramento de estruturas na construção civil / João Lauro Beserra da Costa. – 2022.
56 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2022.
Orientação: Prof. Me. Filipe Maciel Roberto.

1. Monitoramento de Integridade estrutural. 2. Rede de Sensores sem Fio. 3. Sistema de coleta de dados. I. Título.

CDD 005.1

JOÃO LAURO BESERRA DA COSTA

SISTEMA DE COLETA DE DADOS POR MEIO DE REDE DE SENSORES SEM FIO APLICADO AO MONITORAMENTO DE ESTRUTURAS NA CONSTRUÇÃO CIVIL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em:

BANCA EXAMINADORA

Prof. Ms. Filipe Maciel Roberto (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Alexandre Matos Arruda
Universidade Federal do Ceará (UFC)

Prof. Dr. Dmontier Pinheiro Aragão Jr
Universidade Federal do Ceará (UFC)

RESUMO

A técnica de Monitoramento de Integridade Estrutural (MIE) caracteriza-se pela utilização de dados coletados via sensores para avaliar a saúde de estruturas de engenharia. Com o monitoramento contínuo de uma estrutura por um determinado período, é possível avaliar e detectar possíveis danos. Nessa técnica, os sensores são instalados em vários pontos, os dados coletados precisam ficar disponíveis para análise em ambiente seguro, que é característica de serviços de computação em nuvem. Entretanto, em alguns cenários não é possível, ou é extremamente custoso criar uma infraestrutura de comunicação. Analisando esta complexidade, o presente trabalho visa o desenvolvimento de um sistema de coleta de dados por meio de Rede de Sensores sem Fio (RSSF) integrado com um serviço de armazenamento em nuvem que possa ser aplicado no monitoramento de estruturas na construção civil. Para construção do sistema de coleta foi utilizado o protocolo *ESP-MESH* integrado com o serviço *IoT Core* da *Amazon* por meio do protocolo de comunicação Message Queuing Telemetry Transport (MQTT). O armazenamento de dados foi feito através do serviço *DynamoDB*. Também foi utilizado o protocolo General Packet Radio Service (GPRS) para comunicação com a internet em cenários em que não é disponibilizada uma conexão por rede WiFi local. Cada uma das etapas foram testadas durante o desenvolvimento. Por fim, para validar a proposta deste trabalho, foi utilizado o sistema em um cenário de teste em laboratório, no qual foram coletados e disponibilizados dados de vários pontos monitorados de uma viga de concreto armado.

Palavras-chave: monitoramento de integridade estrutural; rede de sensores sem fio; sistema de coleta de dados.

ABSTRACT

The Structural Health Monitoring (SHM) technique is characterized by the use of data collected via sensors to assess the health of engineering structures. With continuous monitoring of a structure for a certain period, it is possible to evaluate and detect possible damage. In this technique, sensors are installed at various points, the data collected need to be available for analysis in a safe environment, which is characteristic of cloud computing services. However, in some scenarios it is not possible, or is extremely expensive to create a communication infrastructure. Analyzing this complexity, the present This work aims to develop a data collection system through a Sensor Network Wireless (WSN) integrated with a cloud storage service that can be applied in monitoring structures in civil construction. For the construction of the collection system, using the ESP-MESH protocol integrated with Amazon's IoT Core service through the Message Queuing Telemetry Transport (MQTT) communication protocol. the storage of data was done through the DynamoDB service. The General protocol was also used. Packet Radio Service (GPRS) for internet communication in scenarios where it is not A local WiFi connection is available. Each of the steps was tested during the development. Finally, to validate the proposal of this work, the system in a laboratory test scenario, in which data were collected and made available from various monitored points of a reinforced concrete beam.

Keywords: structural health monitoring; sensor network wireless; data collection system.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AWS	Amazon Web Services
GPRS	General Packet Radio Service
JSON	JavaScript Object Notation
LAREB	Laboratório de Reabilitação e Durabilidade das Construções
LTI	Laboratório de Tecnologias Inovadoras
M2M	Machine to Machine
MIE	Monitoramento de Integridade Estrutural
MQTT	Message Queuing Telemetry Transport
RSSF	Rede de Sensores sem Fio
SHM	Structural Health Monitoring
TCP	Transmission Control Protocol
TLS	Transport Layer Security
WLAN	Rede Local Sem Fio

SUMÁRIO

1 INTRODUÇÃO

Na construção civil, estruturas comprometidas podem ocasionar prejuízos financeiros além de ser um potencial risco na vida de quem as utilizam. A avaliação do estado de saúde das estruturas comprometidas deve ser realizada através de estudos e experimentos, esses estudos têm por finalidade fornecer métricas e indicadores que auxiliem uma análise confiável acerca dos padrões de segurança (SILVA, 2017).

Esses dados sobre a integridade de estruturas, podem ser coletados através do uso de sensores que são aplicados diretamente na estrutura. Uma das técnicas, o Monitoramento de Integridade Estrutural (MIE), utiliza vários sensores que fornecem dados para processamento. Através da análise desses dados pode-se chegar a uma avaliação confiável sobre o estado da estrutura monitorada (PRADA *et al.*, 2012).

Como o monitoramento não se dá em apenas um lugar específico da estrutura, há vários lugares monitorados, portanto, é preciso que haja uma comunicação deles, para que sejam coletados dados de todos esses pontos monitorados, com a finalidade de avaliar a saúde da estrutura como um todo (MESQUITA *et al.*, 2018). Em alguns cenários, não é possível ou é extremamente custoso criar uma infraestrutura de comunicação. É necessário que, em tais situações, seja utilizada uma Rede de Sensores sem Fio (RSSF) em modo *ad hoc*, que por definição não precisa de uma infraestrutura para sua construção (GTA/UFRJ, 2010). As RSSF são formadas por diversos nós, cada um possui um ou mais sensores. Porém, a RSSF tem capacidade limitada de recursos de processamento, armazenamento e comunicação por causa da limitação energética (CORDEIRO; AGRAWAL, 2011). Diante disso, as RSSFs podem usar estratégias que visam a economia de energia, de forma que, recursos disponíveis na rede possam ser desativados e reativados em momentos estratégicos. Com isso, a coleta de dados dos sensores precisa ser sincronizada com tais estratégias.

Os dados coletados nesse processo precisam ficar disponíveis para análise, em ambientes seguros e com grande capacidade de processamento. Tais características são encontradas na computação em nuvem, ou, em inglês, *Cloud Computing* (DÍAZ *et al.*, 2016).

1.1 Objetivo geral

O objetivo deste trabalho é desenvolver um sistema de coleta de dados via rede de sensores sem fios aplicado ao monitoramento de estruturas na construção civil.

1.2 Objetivos Específicos

- Montar uma rede de sensores sem fio de monitoramento de um caso de teste;
- Enviar os dados coletados para AWS via protocolo de comunicação MQTT;
- Estabelecer comunicação segura com a internet utilizando módulo GSM;
- Disponibilizar os dados para análise em um ambiente em nuvem.

1.3 Justificativa

O presente trabalho foi desenvolvido junto ao Laboratório de Tecnologias Inovadoras (LTI) e colaboração do Laboratório de Reabilitação e Durabilidade das Construções (LAREB), ambos pertencentes à Universidade Federal do Ceará. O sistema de coleta de dados desenvolvido neste trabalho poderá ser utilizado em serviços desenvolvidos pela Startup Integrity Engenharia, que atualmente desenvolve soluções de monitoramento estrutural por meio de sensores.

1.4 Metodologia

O desenvolvimento deste trabalho, foi realizado em quatro etapas: pesquisa bibliográfica, análise/projeto, desenvolvimento e testes. A pesquisa bibliográfica baseia-se em trabalhos da área de sistemas distribuídos e trabalhos sobre monitoramento de estruturas, coletados em livros e artigos obtidos das bases: IEEE Xplore, ACM Digital Library e Google Acadêmico. A fase de análise e projeto foi feito mediante um levantamento das tecnologias e equipamentos que foram utilizados no mesmo. Na fase de desenvolvimento, é onde de fato foi implementada a solução proposta por este trabalho. Por fim, a fase de teste foi realizada em laboratório, a fim de validar a implementação do sistema proposto.

1.5 Organização do trabalho

Este trabalho está dividido em sete capítulos: o primeiro introdutório, o segundo apresenta os conceitos que fundamentam o desenvolvimento deste trabalho, já o terceiro mostra os trabalhos relacionados, o quarto retrata da metodologia adotada para o desenvolvimento do mesmo, o quinto e sexto apresentam, respectivamente, o desenvolvimento da solução proposta e os testes feitos em laboratório. Por fim, o sétimo apresenta as conclusões e trabalhos futuros a serem implementados.

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo, serão apresentados os principais conceitos necessários para o entendimento e desenvolvimento deste trabalho. A Seção 2.1, apresenta conceitos sobre a técnica de monitoramento de integridade estrutural. A Seção 2.2, aborda sobre rede de sensores sem fio. As Seções 2.3 e 2.4, descrevem respectivamente sobre microcontroladores e tecnologias de comunicação. Por fim, na Seção 2.5 são abordados conceitos sobre computação em nuvem.

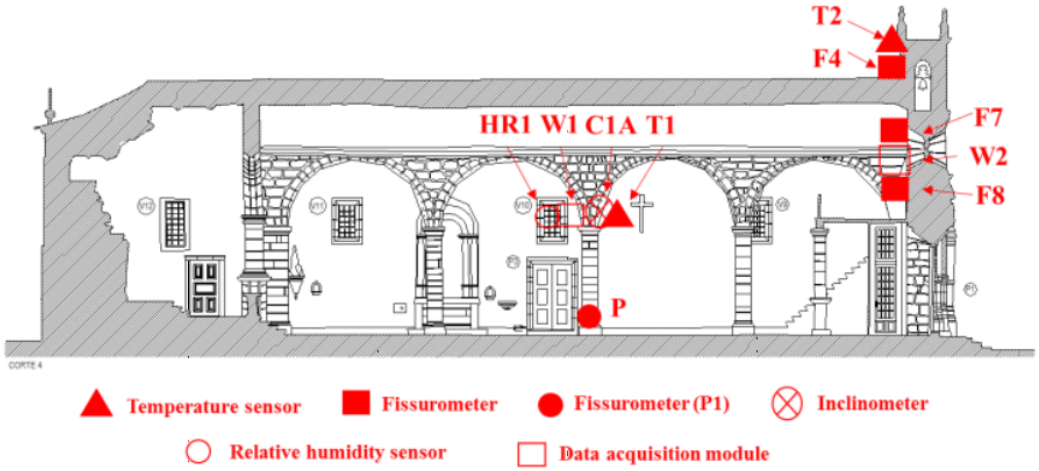
2.1 Monitoramento de Integridade Estrutural - MIE

O MIE, ou em inglês, *Structural Health Monitoring (SHM)* é uma técnica que tem por objetivo avaliar a saúde estrutural por meio de análise de dados coletados por sensores aplicados à estrutura. A partir desta avaliação é possível identificar danos na estrutura ainda em estados iniciais, a identificação precoce destes eventos pode diminuir os custos de manutenção, assim como evitar o comprometimento da estrutura (SILVA, 2017). Segundo Abdulkarem *et al.* (2020), o sistema típico de MIE contém três elementos principais:

- Sistema de sensores: utilizados no monitoramento de integridade estrutural. Têm o objetivo de coletar dados sobre a estrutura sensoreada, como por exemplo: aceleração, deslocamento e estresse, assim como coletar grandezas do ambiente da estrutura, como: temperatura e umidade;
- Sistema de processamento de dados: consiste no conjunto das seguintes etapas: aquisição, transmissão, agregação, processamento e armazenamento de dados.
- Sistema de avaliação de saúde: nesta etapa que os dados coletados são avaliados de acordo com critérios de monitoramento estabelecidos para cada grandeza. Deste modo, é avaliado a segurança geral e/ou estabilidade da estrutura sensoreada.

A Figura 1 mostra a distribuição de sensores em uma estrutura monitorada com a técnica de monitoramento de integridade estrutural.

Figura 1 – Distribuição de sensores na estrutura da igreja de Vila Nova de Foz Côa, Portugal.



Fonte: (MESQUITA *et al.*, 2018)

Mesquita *et al.* (2018) utilizam as técnicas de monitoramento de integridade estrutural para avaliar a integridade da estrutura da Igreja de Vila Nova de Foz Côa, em Portugal, uma construção danificada do século XVI. Esta construção possui danos estruturais em algumas colunas e na fachada principal, nas quais foram instalados sensores a fim de coletar dados de grandezas como temperatura, umidade relativa do ar e deslocamento. A partir dos dados coletados ao longo de um ano, os autores constataram que os deslocamentos observados durante o estudo não apresentam riscos à integridade da estrutura monitorada.

2.2 Redes de sensores sem fio - RSSFs

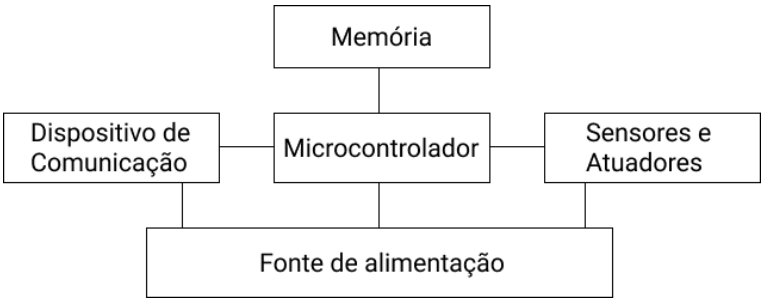
Redes de sensores sem fio consistem em estruturas formadas por vários nós sensores que se comunicam através de protocolos de comunicação no meio sem fio. Os componentes presentes nesse tipo de rede, são capazes de coletar dados de grandezas do meio físico, como por exemplo: temperatura, umidade, luminosidade, vibração e corrosão. A partir destes dados é possível monitorar ambientes e fornecer métricas para tomada de decisões (KARL; WILLIG, 2005).

A limitação de energia é uma característica dos nós sensores das RSSFs, pois, os mesmos, na maioria dos contextos de uso, não possuem acesso ao sistema de fornecimento de energia padrão. Com isso, a alimentação dos componentes é feita usando baterias, que são acopladas ao nó sensor (LOUREIRO *et al.*, 2003) Além da unidade de fornecimento de energia, os nós sensores são constituídos principalmente de quatro componentes (KARL; WILLIG, 2005):

- Microcontrolador: unidade que realiza todo o processamento dos componentes no sistema, responsável pela execução do algoritmo utilizado no monitoramento;
- Memória: responsável por armazenar os dados coletados pelo sistema, além de ser utilizada na execução da aplicação. Um nó pode possuir tipos de memórias diferentes, como por exemplo: memória RAM e memória flash;
- Sensores e atuadores: responsável por coletar dados do meio físico do ambiente monitorado;
- Protocolos de comunicação: responsável por transmitir os dados coletados, além de receber comandos de outros nós.

A Figura 2 faz uma representação dos componentes interligados. Formando a estrutura de um nó sensor.

Figura 2 – Visão geral dos principais componentes de hardware do nó sensor.

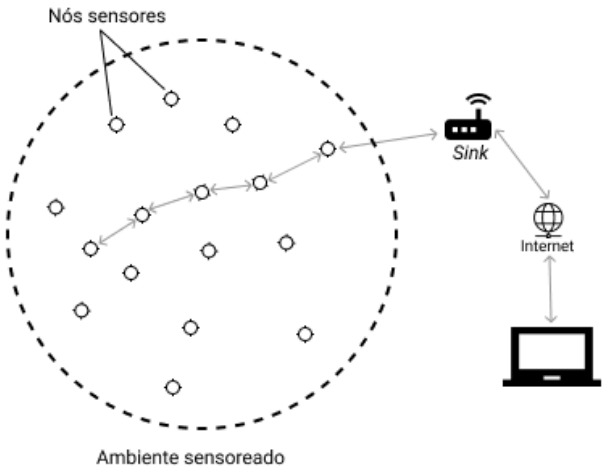


Fonte: (KARL; WILLIG, 2005)

As RSSFs podem ser utilizadas em diversos contextos de aplicação, como por exemplo: indústria, medicina, agricultura, aviação e construção civil. Estes contextos de aplicação compartilham de características semelhantes, que justificam o uso das RSSFs nesses ambientes. Entre as características, pode-se destacar: muitos pontos a serem monitorados, ausência de uma infraestrutura fixa e necessidade de escalonamento (CORDEIRO; AGRAWAL, 2011).

Um exemplo de arquitetura de redes de sensores sem fio é demonstrado na Figura 3. Os nós sensores são espalhados em um ambiente a ser monitorado. Os dados coletados por estes sensores são transmitidos através de uma arquitetura de múltiplos saltos (*multihop*). Estes dados são compartilhados com os nós sensores vizinhos até que cheguem ao ponto de acesso (*Sink*). Este último, é responsável por enviar os dados para a internet, que posteriormente serão utilizados conforme a necessidade do usuário (TAVARES, 2002).

Figura 3 – Arquitetura de uma rede de sensores sem fio.



Fonte: Autor.

A viabilidade de projetos de monitoramento em ambientes sensoreados, depende de diversos fatores fundamentais para o funcionamento do sistema. Um destes fatores é a capacidade do sistema saber lidar em caso de falha de algum dos componentes da rede. A falha de um dos nós sensores não deve afetar o funcionamento da rede como um todo. Outro fator importante é o baixo consumo de energia, pois na maioria dos casos de uso de redes de sensores sem fio os componentes são alimentados por baterias, tornando este recurso bastante limitado. Além destes fatores, o baixo custo de construção dos nós sensores é um fator determinante para a construção da rede, pois a quantidade de nós necessários para o monitoramento de um ambiente pode ser bastante significativa dependendo do projeto (TAVARES, 2002).

Nos últimos anos, os avanços em eletromecânica foram bastante significativos, isso contribuiu para a construção de sensores cada vez mais inteligentes, precisos e com tamanhos cada vez menores. Além disso, os custos destes componentes têm se tornado cada vez menores (NAKAMURA *et al.*, 2007), tornando a utilização de RSSFs soluções eficientes e viáveis para aplicações de sensoriamento.

2.3 Microcontrolador

Como citado na Seção anterior, os microcontroladores são componentes essenciais para a construção de nós sensores. De modo geral, os microcontroladores são pequenos computadores que possuem um único circuito integrado que reúne um núcleo de processador, memórias voláteis e não voláteis e diversos periféricos de entrada e de saída de dados (KARL; WILLIG,

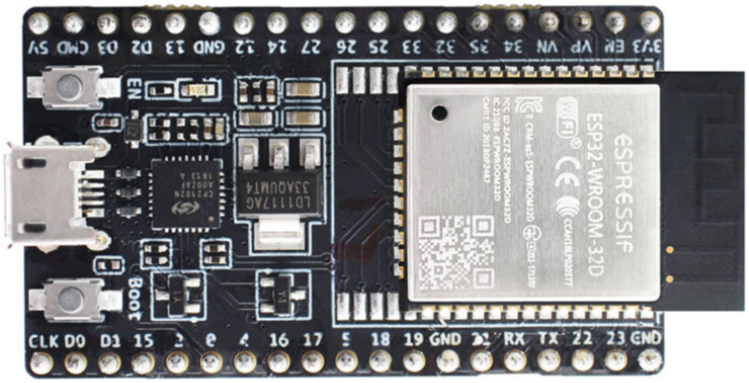
2005).

Uma opção de microcontrolador é o modelo ESP32, desenvolvido pela *Espressif System*. De acordo com o fabricante, o ESP32 oferece uma plataforma robusta e altamente integrada, que ajuda a atender às demandas contínuas de uso eficiente de energia, design compacto, segurança, alto desempenho e confiabilidade. Dentre as principais características do ESP32, temos (ESPRESSIF, 2021):

- Baixo consumo de energia: através da combinação de softwares proprietários e estratégias de escalonamento dinâmico;
- Alto nível de integração: diversos tipos de componentes podem ser integrados ao ESP32. Como, por exemplo: sensores, atuadores, baterias e dispositivos de comunicação;
- Robustez: segundo o fabricante, o ESP32 é capaz de funcionar de forma confiável em ambientes com uma temperatura de operação variando de –40 ° C a + 125 ° C.

As características apresentadas tornam o ESP32 uma solução viável para a construção de nós sensores para redes de sensores sem fio aplicadas ao monitoramento de estruturas na construção civil. A Figura 4 mostra o microcontrolador ESP32.

Figura 4 – Microcontrolador ESP32.



Fonte: (ESPRESSIF, 2021)

2.4 Tecnologias de comunicação

2.4.1 ZigBee

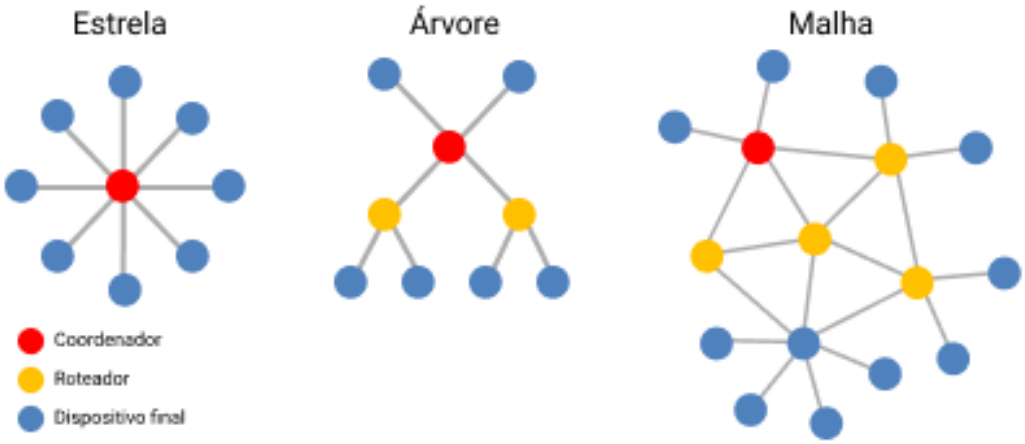
ZigBee é um protocolo de comunicação sem fio que usa o padrão IEEE 802.15.4. Este protocolo foi desenvolvido para redes de baixo custo e baixo consumo. Este protocolo pode ser aplicado em diversos cenários como, automação industrial, automação residencial e em cenários de monitoramento e coleta de dados. Tem como características o baixo consumo de

energia e baixa taxa de transferência de dados (OMOJOKUN, 2015).

Diferente de uma rede WiFi convencional onde todos os dispositivos se conectam a um roteador, os dispositivos ZigBee podem formar topologias de redes diferentes. A Figura 5 representa os tipos de topologias possíveis em uma rede ZigBee. Estas redes são formadas por diferentes tipos de dispositivos:

- Coordenador: dispositivo que inicia a rede e mantém as informações de todos os dispositivos conectados. Este tipo de dispositivo é responsável por se conectar a rede WiFi;
- Roteador: dispositivo intermediário que realiza a transmissão de dados para outros dispositivos próximos;
- Dispositivo final: dispositivos que coletam e transmitem informações para o roteador.

Figura 5 – Tipos de topologia ZigBee.



Fonte: Autor.

- Estrela: nessa topologia a rede é gerenciada por um único coordenador que é responsável pela comunicação com os dispositivos finais;
- Árvore: Formada por sub-redes em que um coordenador é responsável por gerenciar um parte da rede;
- Malha: todos os dispositivos podem se comunicar um com os outros.

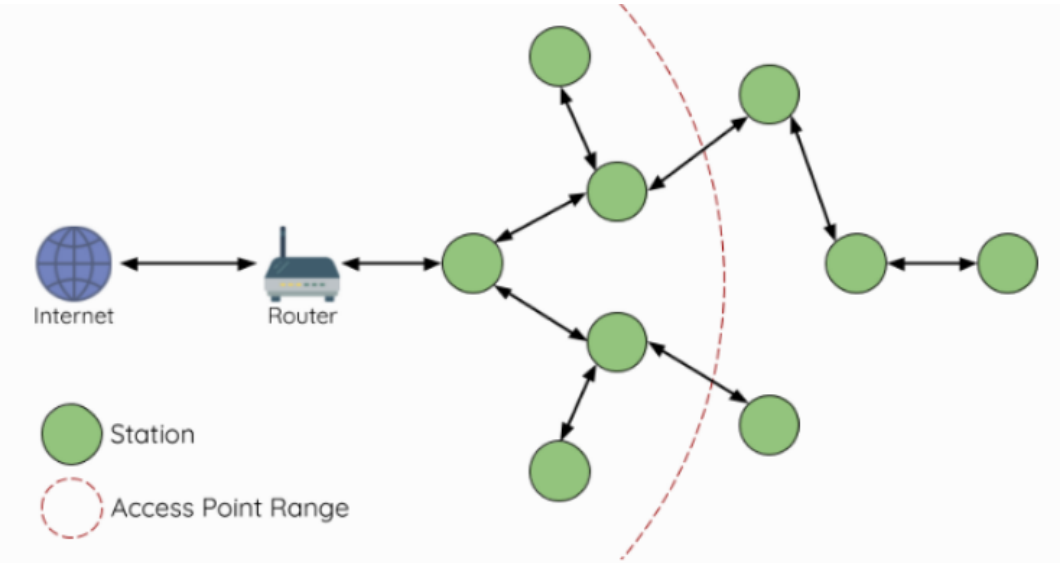
2.4.2 ESP-MESH

ESP-MESH é um protocolo de rede construído sobre o protocolo WiFi, esse protocolo permite que vários nós sensores espalhados em um ambiente sejam interconectados em uma única Rede Local Sem Fio (WLAN). Uma característica deste protocolo é a capacidade de auto-organização, caso um ou mais nós da rede falhem ou com o surgimento de um novo nó

(ESPRESSIF, 2021).

Diferente de uma rede WiFi tradicional, onde um único nó central é responsável por servir de ponto de acesso a todos os outros nós, no protocolo ESP-MESH, todos os nós podem se conectar um com os outros através de uma arquitetura de múltiplos saltos (multi-hop), dessa forma, os nós podem transmitir os seus pacotes, assim como retransmitir os pacotes dos outros nós, desde que haja um caminho entre eles. A Figura 6 mostra um exemplo de uma rede ESP-MESH (ESPRESSIF, 2021).

Figura 6 – Arquitetura de rede ESP-MESH.



Fonte: https://docs.espressif.com/

Uma rede ESP-MESH possui características que torna esse tipo de rede uma possível solução para o uso de rede de sensores sem fio aplicada a técnica de monitoramento de integridade estrutural, entre essas características podemos destacar:

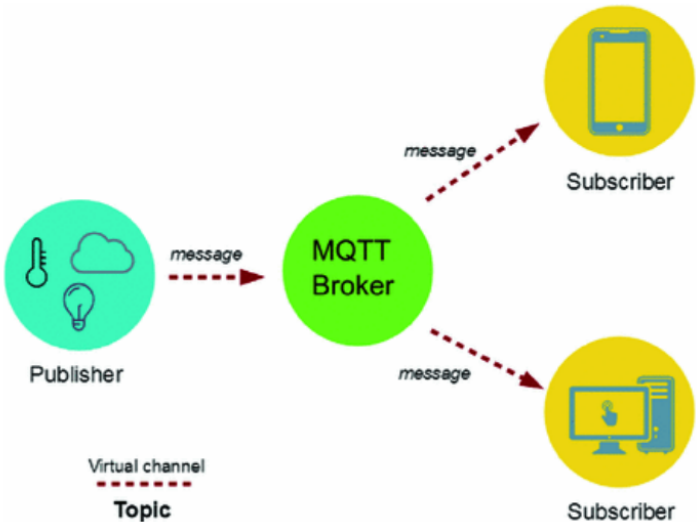
- Baixo custo de construção de rede;
- Capacidade de auto organização da rede;
- Diferentes modos de operações, com modos focados em baixo consumo de energia;
- Rede escalável, capacidade de formar redes com até 1000 nós;
- Ampla cobertura, pois tem capacidade de conectar dois nós com até 200 metros de distância entre eles;
- Taxa de transferência de dados de até 10 Mbps;

2.4.3 MQTT

MQTT é um protocolo de comunicação baseado em fila de mensagens, que funciona sob o protocolo Transmission Control Protocol (TCP), seu principal objetivo é fornecer um meio de comunicação de máquina para máquina, Machine to Machine (M2M). As características deste protocolo, fazem do MQTT uma opção para o desenvolvimento de projetos de internet das coisas, dentre suas características, temos: baixo consumo de banda, fácil implementação, baixo consumo de recursos dos dispositivos, assíncrono e escalável (VADLURI; SAGAR, 2018).

O MQTT utiliza o padrão *publish/subscribe* para a troca de mensagens. Em uma estrutura tradicional cliente servidor, uma requisição é enviada pelo cliente diretamente para o servidor, assim como no recebimento da resposta do servidor. Diferente disso, no padrão *publish/subscribe*, existe um elemento central responsável por gerir a comunicação entre as partes, esse elemento é denominado *broker*. Dessa forma, quando um cliente precisa disponibilizar informações para outro cliente, o mesmo envia as informações para o *broker*. Da mesma forma, quando um cliente precisa receber um determinado dado, este cliente subscreve fazendo uma requisição ao *broker*. Vadluri e Sagar (2018) disponibilizou uma representação da comunicação do protocolo MQTT, como mostra a figura 7.

Figura 7 – Representação da comunicação do protocolo MQTT.



Fonte: (VADLURI; SAGAR, 2018)

2.5 Computação em nuvem

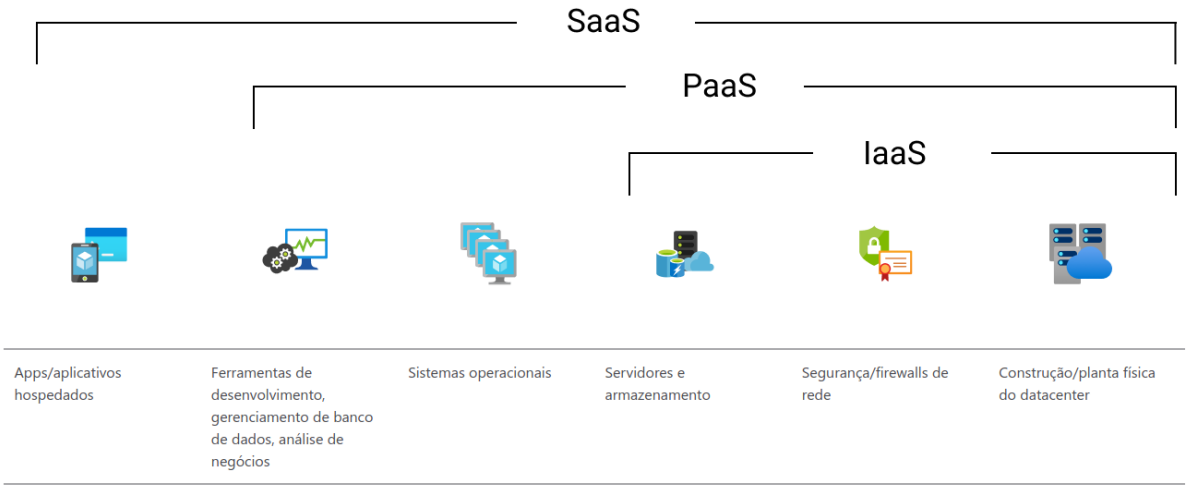
A Computação em nuvem ou em inglês, *Cloud Computing*, é o termo adotado para utilização de serviços e recursos computacionais através da internet. A necessidade de grandes organizações compartilharem os recursos computacionais de seus data centers com seus clientes, fez surgir o conceito de computação utilitária, onde os recursos são cobrados de acordo com o uso (STEEN; TANENBAUM, 2017).

Diversos tipos de recursos e serviços de computação em nuvem podem ser contratados, como por exemplo: poder de processamento, armazenamento de dados e aplicações que forneçam algum serviço sem a necessidade de instalação na máquina do usuário. Estes recursos podem ser utilizados de qualquer lugar, a qualquer hora, desde que o dispositivo utilizado para acessar estes recursos esteja conectado à internet (RAJARAMAN, 2014).

Os recursos são distribuídos geograficamente e replicados em vários data centers, dessa forma, esses serviços são entregues com alta disponibilidade. Os mesmos são alocados dinamicamente aos clientes contratantes de acordo com a demanda. Nos últimos anos, a demanda de computação em nuvem tem aumentado consideravelmente. Um dos fatores para essa crescente, é a quantidade de dados a serem armazenados de forma segura e com alta disponibilidade. Tais características são encontradas na computação em nuvem, além dos serviços de armazenamento, outros tipos de serviços podem ser empregados aos dados armazenados, como por exemplo, a virtualização de dados por meio de análise e otimização de processamento paralelo (BOJANOVA *et al.*, 2013).

A computação em nuvem é usada nos mais diversos tipos de serviços na internet, como por exemplo: armazenamento de dados, fornecimento de software sob demanda, transmissão de áudio e vídeo e ambientes de desenvolvimentos e testes. Estes serviços são divididos principalmente em três tipos como mostra a Figura 8 (AZURE, 2021):

Figura 8 – Tipos de serviços em nuvem.



Fonte: (AZURE, 2021) adaptado

- Infraestrutura como serviços (IaaS): tipo de serviço que oferece recursos computacionais, como armazenamento e rede. Com isso, uma empresa ou um indivíduo podem contratar esses tipos de serviços e pagar conforme o uso. O uso desses serviços pode ajudar na redução de manutenções em *data centers* locais, assim como diminuir os custos de aquisição de hardware.
- Plataforma como serviço (PaaS): tipo de serviços que fornecem um ambiente sob demanda para desenvolvimento, testes, fornecimento e gerenciamento de softwares. Dessa forma os desenvolvedores podem criar aplicações, sem se preocupar com configurações, infraestrutura de servidores, armazenamento necessários para desenvolvimento.
- Software como serviço (SaaS): método de entrega de aplicações de software sob demanda na internet, geralmente, esses serviços são vendidos por assinatura. Nisso, os provedores de nuvem hospedam e gerenciam as aplicações e infraestruturas, assim como também gerenciam qualquer tipo de manutenção e atualizações necessárias.

3 TRABALHOS RELACIONADOS

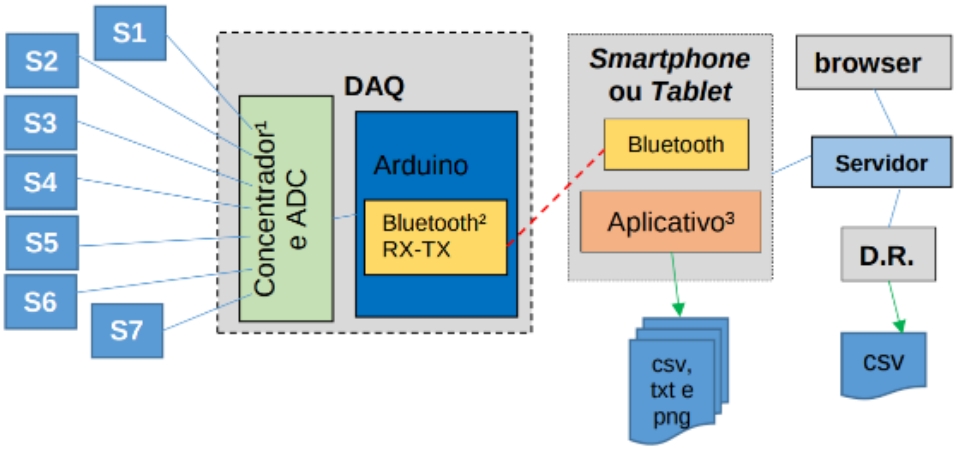
Este Capítulo, descreve alguns trabalhos relevantes para o desenvolvimento do presente trabalho. Na Seção 3.1, é apresentada uma abordagem de coleta de dados via sensores. Já a Seção 3.2, é apresentada uma abordagem de coleta de dados de monitoramento de uma estrutura histórica danificada, utilizando redes de sensores sem fio aplicada à técnica de monitoramento de integridade estrutural. A Seção 3.3 é mostrada uma visão geral das vantagens, problemas e desafios da integração de redes de sensores sem fio com a computação em nuvem. Por fim, na Seção 3.4, é apresentada a contribuição desses trabalhos, bem como, uma tabela comparativa dos mesmos com o presente trabalho.

3.1 Desenvolvimento de sistema para monitoramento de estruturas com utilização da plataforma arduino.

Silva (2018) apresenta um sistema de baixo custo, para monitoramento remoto de deformações e deslocamentos em estruturas de concreto armado. Nesse trabalho, foi realizado um estudo sobre o estado da arte de sistemas de medição de deformação e deslocamento. Após essa etapa identificou-se a necessidade do uso de amplificadores de sinais, pois as grandezas relacionadas às deformações, são relativamente pequenas quando transformadas em sinais elétricos. A partir disso, foram realizados experimentos com módulo amplificador de sinais HX711 que é compatível com a plataforma *arduino*. Após os resultados dos experimentos foi desenvolvido um sistema de coleta de dados denominado *e-structure* com o objetivo de obter os dados provenientes de sensores de deformação, deslocamento e força. Esse sistema é controlado por uma aplicação *mobile*, através do protocolo de comunicação *bluetooth*.

Para validação do sistema, foram realizados testes de laboratório de flexão de lâminas metálicas e vigas de concreto armado, além de testes de campo, onde foram coletados dados de sensores instalados na estrutura de um pilar de uma ponte. Os experimentos de curta duração, mostraram que, o sistema proposto se mostrou eficiente à proposta de monitoramento de integridade estrutural. A Figura 9, mostra a estrutura do sistema de coleta de dados desenvolvido.

Figura 9 – Estrutura de funcionamento do sistema de monitoramento *e-structure*.



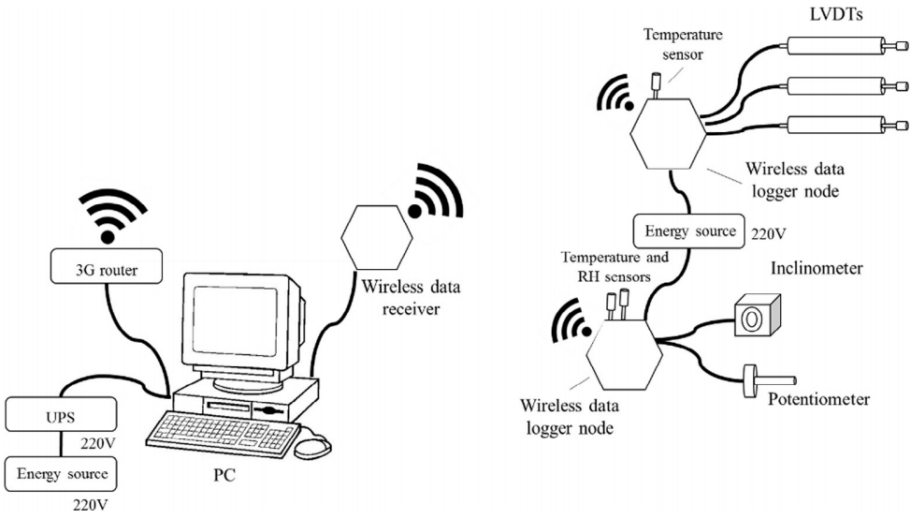
Fonte: (SILVA, 2018)

3.2 Monitoramento a longo prazo de uma estrutura histórica danificada usando rede de sensores sem fio

Mesquita *et al.* (2018) utilizaram redes de sensores sem fio para aplicar a técnica de monitoramento de integridade estrutural na avaliação da saúde de uma estrutura histórica danificada. A estrutura avaliada é a Igreja de Vila Nova de Foz Côa, em Portugal, uma construção do século XVI, onde foram identificados danos estruturais na fachada e em algumas colunas.

Foram instalados quatro sensores ao longo de uma das colunas que apresentou dano estrutural e três sensores em pontos específicos da fachada da igreja. Esses sensores têm o objetivo de coletar dados referentes à temperatura, umidade relativa do ar, inclinação e deslocamento. Os dados foram coletados em um período de longa duração, iniciando em 2015 e finalizando em 2016. Com a avaliação dos dados coletados, foram identificados movimentos estruturais na coluna, esses movimentos estão relacionados à influência da temperatura do ambiente e concluiu-se que a estrutura não corre risco de colapso.

Figura 10 – Instalação do sistema SHM implementado na Igreja de Foz Côa.



Fonte: (MESQUITA *et al.*, 2018)

3.3 Nuvem de Sensores: Integrando redes de sensores sem fio com computação em nuvem

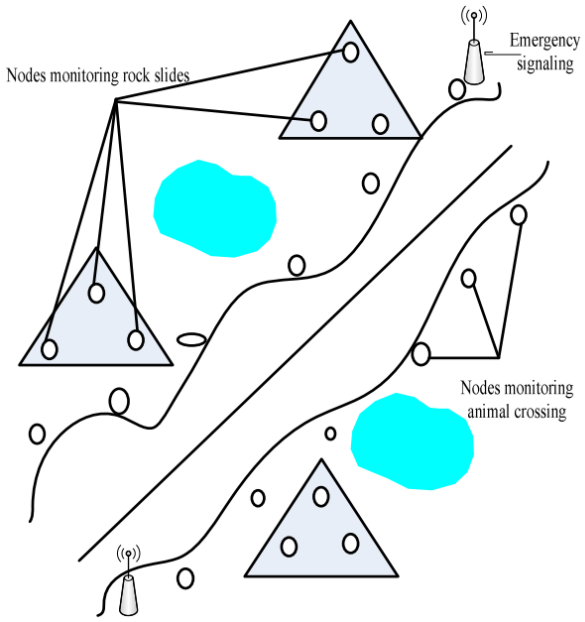
O trabalho de Dwivedi e Kumar (2018) mostra uma visão geral das vantagens, problemas e desafios da integração de redes de sensores sem fio com a computação em nuvem. As RSSFs possuem uma série de limitações como: processamento, armazenamento, memória, disponibilidade e segurança. Partes desses problemas, podem ser solucionados com a integração da RSSF com a computação em nuvem. Essa integração pode ser denominada como nuvem de sensores. É um paradigma que coleta os dados de sensores físicos e os envia para uma infraestrutura de computação em nuvem. Dessa forma, os dados podem ser solicitados por usuários a qualquer momento e em qualquer lugar. Entre as vantagens, problemas e desafios mostrados neste trabalho, temos:

- Vantagens: maior poder de processamento e armazenamento de processamento de dados, flexibilidade, escalabilidade, otimização e compartilhamentos de recursos, alta disponibilidade e múltiplos usuários;
- Problemas e desafios: suporte de segurança e privacidade, política de uso de recursos, preços dos serviços, compatibilidade de recursos e hardware, manutenção e qualidade dos serviços e limitação de largura de banda.

O trabalho mostra um exemplo de aplicação de RSSF com computação em nuvem. Como mostra na Figura 11, trata-se de um sistema de monitoramento de deslizamento de rochas em regiões montanhosas, assim como, o monitoramento de presença de animais. Os dois sistemas

de monitoramento, compartilham do mesmo sistema de sinalização.

Figura 11 – Exemplo de sistemas de monitoramento utilizando RSSF com computação em nuvem.



Fonte: (DWIVEDI; KUMAR, 2018)

3.4 Comparativo dos trabalhos relacionados.

O trabalho de Silva (2018) foi fundamental para entender o funcionamento dos sensores. Já Mesquita *et al.* (2018), foi relevante para o entendimento da rede de sensores sem fio aplicada à técnica de monitoramento de integridade estrutural. Por último, os conceitos apresentados por Dwivedi e Kumar (2018) foram fundamentais para a integração da rede de sensores sem fio com ambientes na nuvem. São apresentados na tabela a seguir os aspectos que diferenciam os trabalhos relacionados a este.

Tabela 1 – Trabalhos relacionados.

Lista de atividades	Silva (2018)	*Mesquita et al. (2018)*	Dwivedi e Kumar (2018)	TCC - Costa (2022)
Coleta de dados via sensores.	X	X	X	X
Uso de Rede de Sensores Fem Fio (RSSF).	-	X	X	X
Aplicação da técnica de Monitoramento de Integridade Estrutural (MIE).	-	X	-	X
Integração com serviços de computação em nuvem.	-	-	X	X
Comunicação com internet via GPRS.	-	-	-	X

Fonte: Autor.

4 METODOLOGIA

O desenvolvimento deste trabalho seguiu quatro etapas, sendo elas: pesquisa bibliográfica, análise/projeto, desenvolvimento, validação e testes. As quatro etapas serão descritas neste capítulo.

Figura 12 – Etapas metodologia.

Fonte: Autor.

4.1 Pesquisa bibliográfica

Nesta etapa, foi feito um levantamento de alguns artigos e livros relacionados com temas pontuais para o desenvolvimento deste trabalho. Os principais temas pesquisados foram: monitoramento de integridade estrutural, uso de rede de sensores sem fio, protocolos de comunicação sem fio e construção de nós sensores. Para isso, foram utilizadas as seguintes bases: IEEE Xplore, ACM Digital Library e Google Acadêmico. Os estudos estão descritos nos capítulos de fundamentação teórica e trabalhos relacionados.

4.2 Análise e projeto

Nesta etapa, foram identificados os principais elementos necessários para a construção do sistema, bem como, as tecnologias que foram utilizadas para implementação e comunicação das partes. Algumas das decisões tomadas após está etapa de análise foram:

- Escolha do microcontrolador: pelas características apresentadas e pela disponibilidade dos microcontroladores, o ESP32 se mostrou uma alternativa atrativa para a construção dos nós do sistema;
- Plataforma de desenvolvimento: o *Visual Studio Code* foi o ambiente escolhido pra desenvolvimento do projeto;
- Linguagem de programação: para os *scripts* dos nós sensores e nó sincronizador foi usada a linguagem de programação C++;
- Protocolos de comunicação: para a comunicação entre os nós da rede foi utilizado o padrão

ESPMESH, pois este padrão foi desenvolvido para facilitar a comunicação e construção da rede utilizando microcontroladores ESPs;

- Sensores: foram utilizados sensores do tipo extensômetros (*strain gauges*) para coletar dados relacionados à deformação, e sensores do tipo *termopar* que são usados para coletar dados referentes à temperatura;
- Amplificador de sinais: o modelo hx711 foi utilizado para ampliar os sinais coletados pelos *strain gauges*.

4.3 Desenvolvimento

Para o desenvolvimento do sistema, foram aplicados conhecimentos adquiridos durante a graduação, especialmente os conhecimentos das áreas de sistemas distribuídos e de desenvolvimento de software. Outros conhecimentos necessários para a construção do sistema, foram adquiridos durante o desenvolvimento do presente trabalho, pode-se destacar o entendimento de noções básicas de eletrônica.

A primeira etapa do desenvolvimento, foi dedicada para a construção dos nós sensores e do nó sincronizador, assim como a implementação dos algoritmos necessários para a coleta e transmissão dos dados entre sensores da rede. A segunda etapa, foi dedicada à integração da rede de sensores com os serviços da Amazon Web Services (AWS)[1].

Para desenvolvimento dos *scripts* foi usado o computador pessoal do autor, por meio de uma plataforma de edição de código *Visual Studio Code* além da biblioteca *PlatformIO*, que tem a finalidade de integrar o *visual studio* com a plataforma *arduino*. A montagem dos nós sensores foi realizada no campus da UFC em Russas, com a ajuda do Laboratório de Tecnologias Inovadoras, LTI.

4.4 Validação e testes

A fim de validar a proposto deste trabalho, foram realizados testes em cada uma das principais etapas de desenvolvimento, na respectiva ordem: construção da rede *ESP-MESH*, transmissão de mensagens, coleta de dados por sensores, envio de dados para nuvem, integração com Amazon IoT Core via MQTT, conexão GPRS e armazenamento de dados com *DynamoDB*. Parte desses testes foram realizados na residência do autor, e, os demais, foram feitos em um dos

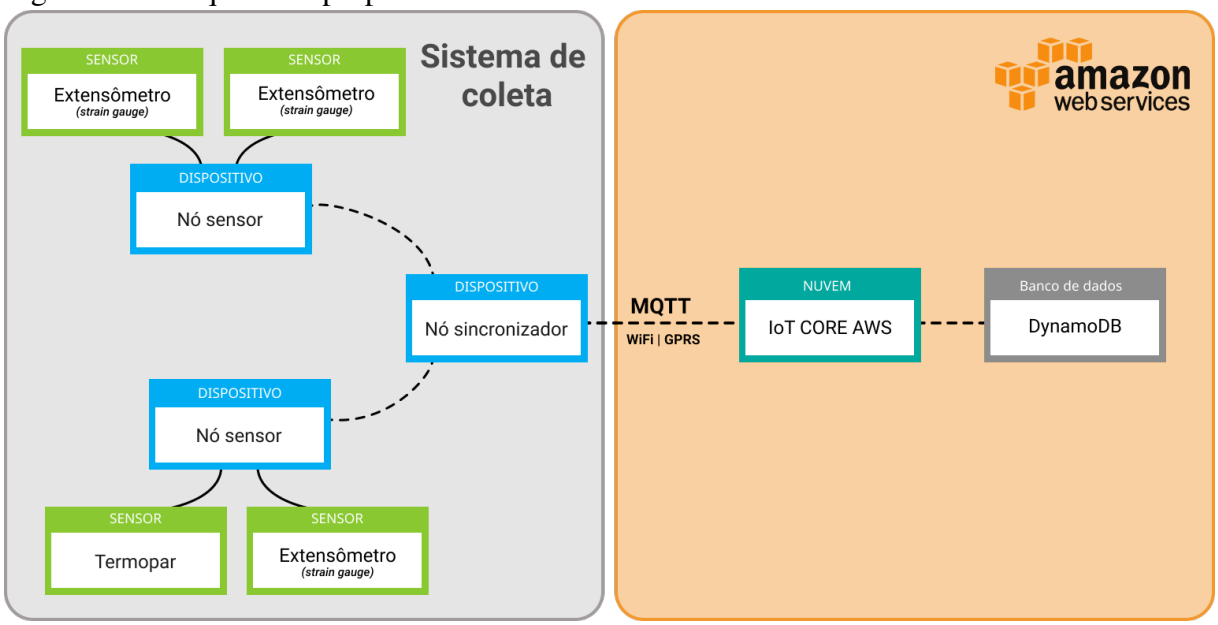
[1] Plataforma de serviços de computação em nuvem, ofertados pela Amazon.com

laboratórios da UFC campus de Russas.

5 ARQUITETURA PROPOSTA

A proposta deste trabalho consiste em um sistema distribuído de coleta de dados por meio do uso de RSSF, de modo que possa ser aplicado ao monitoramento de estruturas na construção civil. Os dados obtidos por este sistema de coleta serão enviados para um ambiente em nuvem, onde ficarão disponíveis para análise. A Figura 13 apresenta uma visão geral do sistema proposto.

Figura 13 – Arquitetura proposta.



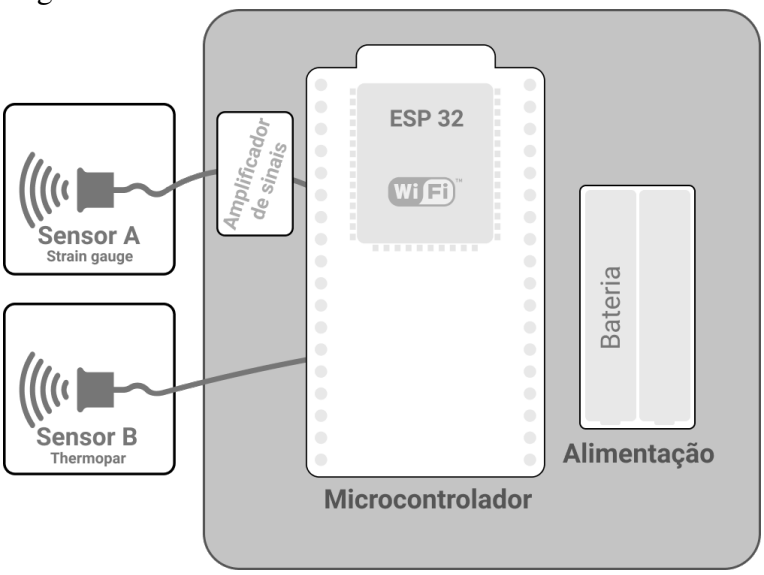
Fonte: Autor.

Para um melhor entendimento, serão apresentados os componentes principais do sistema. As Seções 5.1 e 5.2, dissertam sobre o desenvolvimento e construção dos nós da rede. A Seção 5.3 mostra como é feita a comunicação entre os nós. Já as Seções 5.4 e 5.5 descrevem, respectivamente, como os dados coletados são enviados e armazenados na nuvem.

5.1 Nó sensor

O nó sensor é responsável por coletar os dados dos sensores aplicados à estrutura, processar e transmiti-los para o nó sincronizador. Como visto em (KARL; WILLIG, 2005), os nós sensores são constituídos, principalmente, de cinco componentes: microcontrolador, memória, dispositivo de comunicação, sensores/atuadores, além de uma unidade de fornecimento de energia. A Figura 14, representa a construção de um nó sensor.

Figura 14 – Estrutura do nó sensor.



Fonte: Autor.

- Microcontrolador: responsável em executar os algoritmos de coleta, processamento e transmissão de dados;
- Sensor: componente que é aplicado diretamente no ponto a ser monitorado, sendo responsável por coletar uma grandeza específica, como por exemplo, a deformação;
- Alimentação: componente responsável por fornecer energia para funcionamento do nó sensor.

Alguns tipos de sensores, como os extensômetros*(strain gauges)*, que são utilizados para medir deformações, necessitam do uso de amplificadores de sinais. Pois, como visto no trabalho de Silva (2018), os dados coletados por esse tipo de sensor são relativamente pequenos, quando transformados em variações de sinais elétricos.

Para construção do nó sensor foi utilizado o microcontrolador ESP32, algumas características desse modelo foram fundamentais para sua escolha, como: possuir módulo de comunicação WiFi acoplado nativamente, maior poder de processamento, maior quantidade de pinos digitais e analógicos. A comparação foi feita em relação a outros dois modelos de microcontroladores com propostas semelhantes, Arduino Uno e ESP8266. A Tabela 2 faz uma análise comparativa entre os três microcontroladores. Outro fator determinante para a escolha do ESP32, é a compatibilidade do mesmo com o protocolo de comunicação *ESP-MESH*, no qual foi o protocolo adotado para construção da rede de sensores.

Tabela 2 – Comparação de microcontroladores.

	ESP32	ESP8266	Arduino Uno R3
Processadores	2	1	1
Arquitetura	32 bits	32 bits	8 bits
WiFi	Sim	Sim	Não
GPIO (pinos)	36	17	14
RAM	512 kb	160 kb	2 kb
Flash	16 mb	16 mb	32 kb
ESP-MESH	Sim	Sim	Não

Fonte: Autor.

Para coleta de dados, foram utilizados dois tipos de sensores na construção dos nós sensores: termopar e extensômetro (*strain gauge*), que respectivamente, são responsáveis por coletar sinais de deformação e temperatura. A decisão do uso destes sensores específicos, partiu do consenso entre os laboratórios LAREB e LTI.

O nó sensor, realizará a coleta de dados em intervalo de tempo pré definido e configurado no nó sincronizador. A coleta será realizada quando o nó sensor receber uma mensagem de solicitação, e, ao receber, o mesmo deverá coletar os dados dos sensores, agrupar em uma única *string* de texto e atribuir ao seu identificador único, seguindo o formato JavaScript Object Notation (JSON). A Figura 15 mostra uma representação de uma resposta à solicitação de um nó sincronizador.

Figura 15 – Padronização dos dados coletados pelo nó sensor.

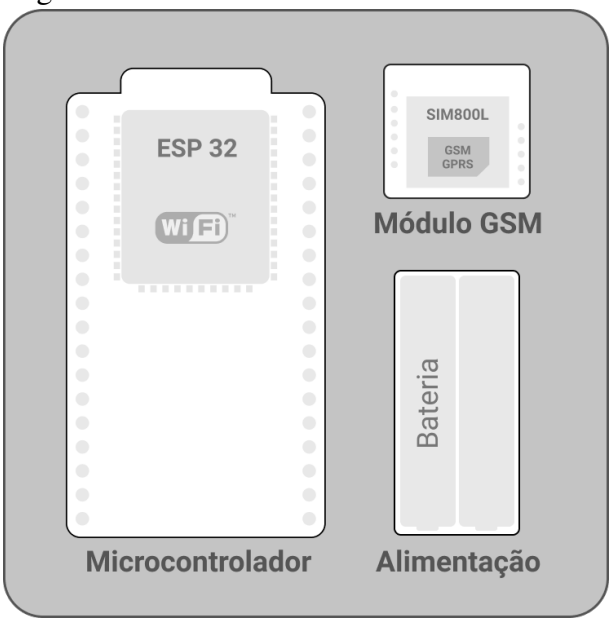
```
{
  2550155645:"{\"STRING_GAUGE_05\":8388608,\"STRING_GAUGE_06\":16772621,}",
}
```

Fonte: Autor.

5.2 Nó sincronizador

O nó sincronizador tem como objetivo principal solicitar periodicamente os dados dos nós sensores, agrupar e repassá-los para um ambiente em nuvem. A Figura 16 mostra a estrutura do nó sincronizador, que assim como o nó sensor, também é composto de um microcontrolador ESP32, além de um módulo de comunicação GPRS, que na ausência de uma rede WiFi poderá ser utilizado para transmitir os dados para a internet.

Figura 16 – Estrutura do nó sincronizador.



Fonte: Autor.

O microcontrolador ESP32 é integrado de um módulo de comunicação WiFi, que é utilizado para comunicação entre os nós da rede, além de estabelecer uma conexão com a internet para envio dos dados. Em alguns cenários de monitoramento de estruturas, a rede WiFi não está disponível, dessa forma, foi utilizado o módulo SIM800L para comunicação com a internet através do uso do protocolo GPRS. Com isso, a internet é disponibilizada por uma operadora de telefonia móvel.

5.3 Comunicação da rede de sensores.

A comunicação entre os nós da rede é feita seguindo o protocolo de comunicação *ESP-MESH*. Como visto anteriormente, este protocolo utiliza comunicação WiFi, mas que, diferente do modelo tradicional, onde todos os dispositivos se conectam a um único ponto de acesso, em uma rede *ESP-MESH*, onde cada dispositivo pode se conectar com os outros, formando, dessa forma, uma rede de dispositivos. Em uma rede deste formato, cada dispositivo funciona de forma independente, no qual, a falha de um deles não irá interromper o funcionamento dos outros.

Em uma rede WiFi tradicional os dispositivos são identificados por um endereço IP, já em uma rede *ESP-MESH* cada nó é identificado pelo *chipId* do microcontrolador. Dessa forma, cada nó possuirá um identificador único de 32bits. As mensagens podem ser transmitidas para todos os nós da rede ou enviadas especificamente para um nó individual por meio de seu

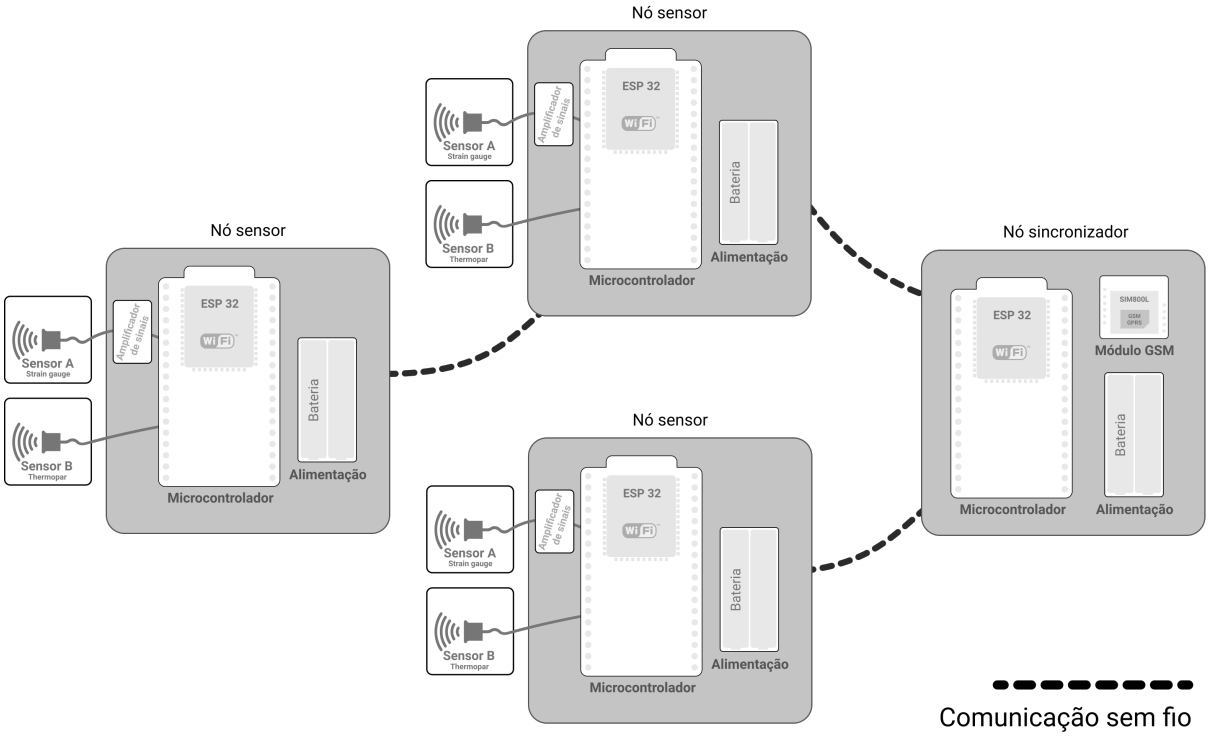
identificador.

Para implementação da rede *ESP-MESH* foi utilizado a biblioteca *painlessMesh*. Essa biblioteca, é usada para criação de rede do tipo mesh específica para microcontroladores ESP32 e ESP8266. O uso da *painlessMesh* permite trabalhar com uma rede *mesh* sem ter que se preocupar como a rede é estruturada ou gerenciada. Algumas rotinas do nó sincronizador precisam ser executadas em segundo plano para que não interrompa o funcionamento da rede. Para isso, utilizaremos a biblioteca *TaskScheduler*, é uma biblioteca de agendamento de tarefas, e que permite controlar em que momento uma determinada tarefa deve ser executada.

A comunicação entre o nó sincronizador e os nós sensores, funcionam de forma reativa, ou seja, os nós sensores só realizam as coletas, quando forem solicitados. Desse modo, em um intervalo de tempo configurado no nó sincronizador, é enviado uma mensagem *broadcast* de solicitação de dados para todos os nós da rede. Os dados retornados dos nós sensores dentro de um intervalo de tempo predefinido, serão agrupados, formando assim, a mensagem que será enviada para a nuvem.

Para a construção da rede, os nós sensores são distribuídos em um ambiente a ser monitorado. Uma representação de comunicação entre nós sensores e nó sincronizador é demonstrada na Figura 17, temos três nós sensores transmitindo os dados coletados para um nó sincronizador. Na mesma figura, também é demonstrado a coordenação dos dados por meio de nós sensores vizinhos, os dados de um nó sensor são transmitidos ao nó sensor mais próximo. Este é responsável por transmitir os seus dados, assim como retransmitir os dados recebidos do nó vizinho para o nó sincronizador ou para um outro nó vizinho.

Figura 17 – Comunicação entre nó sensor e nó sincronizador.



Fonte: Autor.

5.4 Comunicação entre nó sincronizador e nuvem

Para disponibilizar os dados em nuvem, precisamos utilizar uma plataforma IoT, essa plataforma é que será responsável por receber as mensagens de forma segura e enviá-las para algum serviço de armazenamento de dados. Entre as plataformas avaliadas, estão: Amazon IoT Core, Google Cloud IoT Core e Azure IoT Hub. De modo geral, todas elas dispõem de um plano gratuito, já os planos pagos são delimitados pela quantidade de uso, número de mensagens e tempo de conexão. A Tabela 3 mostra um comparativo dos planos gratuitos e dos planos pagos, as informações contidas na tabela são referentes ao quarto trimestre de 2021.

Tabela 3 – Comparação de custos de plataformas IoT.

Plataforma	Plano	Plano gratuito
Amazon IoT Core	$ 0,7 - 1,00 por milhão de mensagens $ 0,08 por milhão de minutos de conexão	-Primeiros 12 meses -500,000 mensagens -2.250.000 minutos de conexão
Google Cloud IoT Core	$ 0,0045 250 mb até 250 gb	-Primeiros 250 mb
Azure IoT Hub	$ 25,00 por 400.000 mensagens/dia	-Primeiros 12 meses -8.000 mensagens por dia

Fonte: Autor.

Neste trabalho, optamos por utilizar o serviço IoT Core da Amazon, além do plano gratuito e dos custos mostrados na Tabela 3, outros fatores foram determinantes para essa escolha:

- Integração facilitada com o serviço de banco de dados *dynamoDB*;
- Ambiente de configuração, gerenciamento e testes dos dispositivos;

Para estabelecer uma comunicação segura com o serviço IoT Core da AWS, o nó sincronizador precisa de um certificado digital do tipo X.509[1]. Este certificado pode ser emitido pela autoridade de certificação (CA) raiz da AWS, o mesmo é responsável por gerar um par de chaves pública/privada que são utilizadas para criptografar, assinar e descriptografar as mensagens. Todas as mensagens devem ser enviadas de forma segura, para isso utilizamos o protocolo de segurança Transport Layer Security (TLS), que tem a finalidade de facilitar a segurança e privacidade dos dados na internet.

O nó sincronizador se comunica com a nuvem de duas formas: utilizando o módulo WiFi nativo do microcontrolador para se conectar a uma rede disponível no ambiente ou através do módulo SIM800L que estabelece uma conexão com a rede por meio do protocolo GPRS disponibilizada por uma operadora de telefonia móvel. Desse modo, precisamos adicionar os certificados e chave privada do nó sincronizador aos dois tipos de conexão. Para isso, foram utilizadas duas bibliotecas:

- *WifiClientSecure*: essa biblioteca já possui nativamente suporte para adição dos certificados a conexão WiFi.
- *SSLClient*: solução encontrada para adicionar manualmente os certificados a conexão GPRS, uma vez que, não foi encontrada uma biblioteca com suporte de adição de certificados nativamente.

Para transmissão dos dados do nó sincronizador para a nuvem, optamos pela utilização do protocolo de comunicação MQTT. Trata-se de um protocolo leve, no sentido de consumo de banda, além de sua fácil implementação e integração com o serviço de IoT Core da AWS, essas características foram relevantes para a escolha deste protocolo.

Os dados recebidos dos nós sensores, são agrupados em formato JSON, logo em seguida, são enviados para o *broker MQTT*. A Figura 18 mostra uma representação do agrupamento dos dados recebidos pelo nó sincronizador, e, abaixo, a descrição de cada um dos itens da mensagem.

[1] X.509 é um formato padrão para certificados de chave pública, documentos digitais que associam com segurança pares de chaves criptográficas a identidades como sites, indivíduos ou organizações.

Figura 18 – Agrupamento de dados recebidos dos nós sensores.

Fonte: Autor.

- *node_id:* identificador do nó mestre;
- *mac_esp32:* endereço físico do nó sincronizador;
- *uuid_collect:* usado pra identificar uma coleta, cada coleta possuí um identificador único;
- *client:* descrição do cliente que está usando o sistema de coleta;
- *location_monitoring:* descreve o local monitorado;
- *active_sensors:* quantidade de sensores ativos em uma coleta;
- *battery_level:* informa sobre o nível da bateria do nó sincronizador;
- *collection_time:* Instante de tempo em que a coleta foi feita;
- *collected_data*: dados recebidos dos nós sensores;
- *status:* informa sobre o estado atual do nó sincronizador.

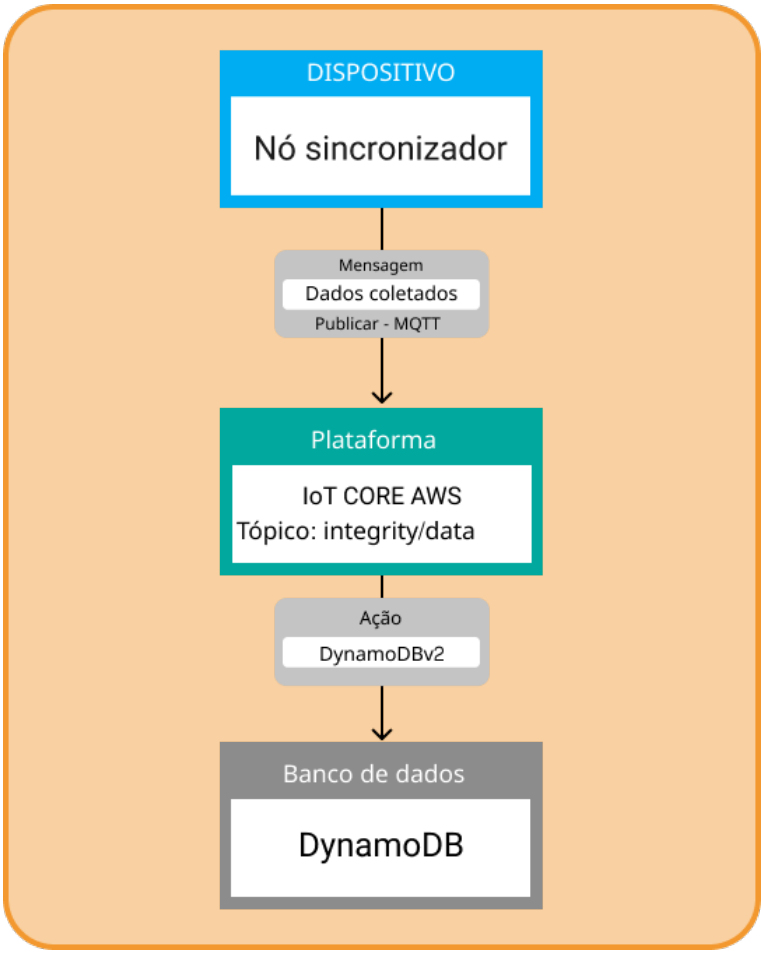
5.5 Armazenamento de dados

Os dados coletados pelos nós sensores precisam ficar disponíveis em ambiente em nuvem. Para armazenamento de dados, foi utilizado o *DynamoDB*, a integração facilitada com o serviço de IoT Core foi um fator determinante para essa escolha. *Amazon DynamoDB* é um banco de dados de chave-valor NoSQL, totalmente gerenciado que fornece um desempenho rápido e previsível com escalabilidade integrada.

O serviço AWS IoT Core contém um mecanismo de regras que permite o processa-

mento contínuo de dados enviados pelos dispositivos conectados. As ações da regra especificam o que fazer quando uma regra é acionada. Desse modo, pode-se definir uma ação que armazenará os dados recebidos, toda vez que uma mensagem MQTT chegar a um determinado tópico. A Figura 19 representa o as etapas da transmissão de dados, partindo da publicação da mensagem pelo nó sincronizador.

Figura 19 – Fluxo de armazenamento de dados.



Fonte: Autor.

A ação utilizada neste trabalho é a *DynamoDBv2*, que grava uma mensagem MQTT inteira ou parte dela em uma tabela do banco de dados *DynamoDB*. Com essa ação, cada atributo de uma mensagem no formato JSON é gravado em uma coluna separada no banco de dados. Para isso, a mensagem deve conter uma chave de nível raiz que corresponda à chave de partição primária da tabela.

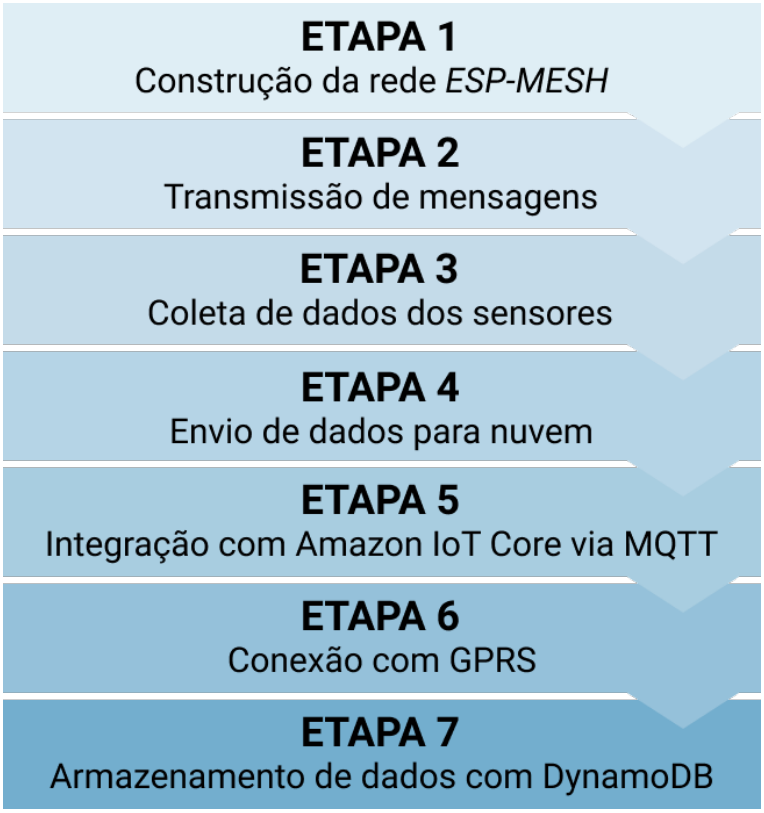
6 VALIDAÇÃO E TESTES

Este capítulo apresenta os testes realizados em cada etapa de desenvolvimento, assim como a validação do sistema proposto por meio de um experimento realizado em laboratório.

6.1 Testes

Esta Seção, descreve os testes feitos durante cada etapa de desenvolvimento. Cada uma destas etapas é descrita na Figura 20. De modo geral, os testes relacionados aos microcontroladores foram validados através da exibição de mensagens por meio do serial monitor que é disponibilizado pelo *platformIO*. Também foi utilizado o cliente de testes MQTT fornecido na plataforma IoT Core da AWS.

Figura 20 – Etapas de testes.



Fonte: Autor.

6.1.1 Etapa 1 - Construção da rede ESP-MESH

Como visto no capítulo anterior, utilizamos a biblioteca *painlessMesh* para construção da rede. Para isso, cada nó da rede é configurado com as mesmas credenciais, que são: identificador da rede, senha de autenticação e porta de acesso. Essas credenciais são passadas

para a função *init*, disponível em um objeto do tipo *painlessMesh*. Além das credenciais, um ponteiro para o agendador de tarefas também é passado para a função.

O teste consiste em identificar se os dispositivos estão de fato formando uma rede. Para isso, o *painlessMesh* dispõe de funções baseadas em eventos, ou seja, ficam esperando por um evento na rede e executa uma tarefa toda vez que este evento de fato ocorre. Para esta etapa, foi utilizada duas funções:

- *onChangedConnections*: recebe o endereço da função que será executada toda vez que a rede for modificada, seja pela entrada ou saída de um dispositivo da rede;
- *onNewConnection*: tem o funcionamento semelhante, porém o identificador do dispositivo é passado para a função.

Utilizou-se as duas funções para exibir mensagens na tela do computador toda vez que a rede for modificada, assim como exibir o identificador do dispositivo que se conectou à rede. Para isso, um dos dispositivos precisa estar conectado ao computador pelo cabo USB.

Para essa etapa de testes, foram utilizados quatro microcontroladores, um deles conectado ao computador e exibindo mensagens na tela do serial monitor. Os outros foram separados em cômodos diferentes da residência do autor, com uma distâncias de dois a cinco metros de um para o outro. A Figura 21 mostra os eventos de alteração e entrada de novos nós na rede.

Figura 21 – Alteração da rede visualizada pelo serial monitor.

```
setLogLevel: ERROR | STARTUP |
STARTUP: init(): 0
STARTUP: AP tcp server established on port 5555
Alteração detectada na rede...
Novo nó na rede, nodeId: = 4071310441
Alteração detectada na rede...
Novo nó na rede, nodeId: = 2550155645
Alteração detectada na rede...
Alteração detectada na rede...
Novo nó na rede, nodeId: = 2826617597
Alteração detectada na rede...
Alteração detectada na rede...
Alteração detectada na rede...
Novo nó na rede, nodeId: = 2550155645
Alteração detectada na rede...
Novo nó na rede, nodeId: = 4071310441
Alteração detectada na rede...
Alteração detectada na rede...
Novo nó na rede, nodeId: = 2826617597
Alteração detectada na rede...
Alteração detectada na rede...
```

Fonte: Autor.

6.1.2 Etapa 2 - Transmissão de mensagens

Verificado que os dispositivos estavam se conectando e formando uma rede *mesh*, iniciou-se uma outra etapa de testes. Nesta etapa, o objetivo era verificar se os dispositivos conseguiam transmitir mensagens uns para os outros. Para isso, utilizamos a abordagem reativa, ou seja, um nó solicitará dados aos outros nós, e estes precisam responder com um número randômico, simulando dados coletados por sensores. Como visto no capítulo anterior, o dispositivo que irá solicitar os dados é denominado nó sincronizador, já quem responde as solicitações é chamado de nó sensor.

O nó sincronizador solicitará os dados periodicamente em um período de tempo pré configurado. Para isso, é necessário o uso de um agendador de tarefas. Como visto na seção anterior, o endereço de uma instância do agendador de tarefas, precisa ser passado junto com as credenciais da rede. Para essa etapa, foi criada a tarefa que envia uma mensagem *broadcast* para a rede, através da função *sendBroadcast* disponível nas funções do objeto *painlessMesh*.

Para a criação da tarefa, precisa-se passar como parâmetro três informações: tempo em que uma função precisa ser executada em milissegundos, um parâmetro que define quantas vezes essa função será executada e o endereço da função.

Após a criação da tarefa, adicionamos a mesma ao agendador de tarefas, por meio da função *addTask*, logo em seguida ativamos a tarefa por meio da função *enable*. Com isso, periodicamente a função *sendMessage* é chamada, enviando o seu identificador para todos os nós da rede. Os nós sensores ao receberem essa mensagem, retornarão um número aleatório para o identificador recebido. A função *receivedCallback* receberá as respostas dos nós sensores, e cada mensagem recebida, irá exibir o identificador do nó e o conteúdo da mensagem. Essas informações serão exibidas através do serial monitor do *platformIO*. A Figura 22 mostra as respostas recebidas pelo nó sincronizador em um cenário de testes, no qual, três nós sensores respondem as solicitações do nó sincronizador.

Figura 22 – Transmissão de mensagens entre dispositivos.

```
--- More details at https://bit.ly/pio-monitor-filters
--- Miniterm on /dev/ttyUSB0  115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
setLogLevel: ERROR | STARTUP |
STARTUP: init(): 0
STARTUP: AP tcp server established on port 5555
Solicitando dados...
dhcps: send_nak>>udp_sendto result 0
dhcps: send_nak>>udp_sendto result 0
Solicitando dados...
Nó 4071310441 Número recebido: 982
Nó 2550155645 Número recebido: 262
Solicitando dados...
Nó 2550155645 Número recebido: 706
Nó 4071310441 Número recebido: 645
Solicitando dados...
Nó 2550155645 Número recebido: 231
Nó 2550158053 Número recebido: 301
Nó 4071310441 Número recebido: 366
Solicitando dados...
Nó 4071310441 Número recebido: 711
Nó 2550155645 Número recebido: 844
Nó 2550158053 Número recebido: 779
Solicitando dados...
Nó 2550155645 Número recebido: 206
Nó 2550158053 Número recebido: 832
Nó 4071310441 Número recebido: 822
Solicitando dados...
Nó 2550155645 Número recebido: 938
Nó 4071310441 Número recebido: 559
Nó 2550158053 Número recebido: 464
```

Fonte: Autor.

6.1.3 Etapa 3 - Coleta de dados dos sensores

Nesta etapa, foi-se testado a coleta de dados por sensores. Os circuitos eletrônicos que ligam os sensores aos microcontroladores foram feitos no laboratório de engenharia civil da UFC campus de Russas, com a colaboração do LTI. Foram utilizados quatro sensores do tipo extensômetro e um sensor do tipo termopar. Os sensores foram conectados a três microcontroladores, formando dessa forma três nós sensores. O nó sincronizador foi preparado para exibir no serial monitor apenas as mensagens de solicitações e as respostas recebidas com os dados coletados, como mostra a Figura 23.

Figura 23 – Dados coletados dos sensores.

```
--- More details at https://bit.ly/pio-monitor-filters
--- Miniterm on /dev/ttyUSB0  115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
setLogLevel: ERROR | STARTUP |
STARTUP: init(): 0
STARTUP: AP tcp server established on port 5555
Solicitando dados...
Solicitando dados...
Nó 2989952141 Dados: {"THERMOPAR":"30.25"}
Solicitando dados...
Nó 2989952141 Dados: {"THERMOPAR":"30.25"}
Solicitando dados...
Nó 2989952141 Dados: {"THERMOPAR":"30.50"}
Nó 2990032209 Dados: {"STRING_GAUGE_03":5273202,"STRING_GAUGE_04":1748082}
Nó 2989922961 Dados: {"STRING_GAUGE_05":8388608,"STRING_GAUGE_06":16773759}
Solicitando dados...
Nó 2989952141 Dados: {"THERMOPAR":"29.75"}
Nó 2990032209 Dados: {"STRING_GAUGE_03":5267870,"STRING_GAUGE_04":1746310}
Nó 2989922961 Dados: {"STRING_GAUGE_05":8388608,"STRING_GAUGE_06":16775366}
Solicitando dados...
Nó 2989952141 Dados: {"THERMOPAR":"30.25"}
Nó 2990032209 Dados: {"STRING_GAUGE_03":5259473,"STRING_GAUGE_04":1861163}
Nó 2989922961 Dados: {"STRING_GAUGE_05":8388608,"STRING_GAUGE_06":16772800}
Solicitando dados...
```

Fonte: Autor.

6.1.4 Etapa 4 - Envio de dados para nuvem

Com os dados sendo coletados, a próxima etapa foi de enviá-los para um ambiente em nuvem. Inicialmente, foi utilizado o serviço de planilhas do Google para armazenamento. Desse modo, é necessário que o nó sincronizador estabeleça uma conexão com a internet. Neste primeiro momento, foi testado apenas o envio através da conexão por WiFi, onde a internet é disponibilizada por um roteador local. A biblioteca *painlessMesh* dispõe de uma função que permite que dispositivos se conectem a uma rede externa por meio de um conexão TCP. Para isso, é preciso passar como parâmetro, o identificador e senha da rede para a função *stationManual*.

Na seção anterior, foi visto que as respostas dos nós sensores são recebidas pelo nó sincronizador. Logo, estes dados serão salvos em uma variável temporária do tipo JSON, que armazenará as respostas até que as mesmas sejam enviadas, após o envio, a variável volta ao estado inicial. Este agrupamento de dados é realizado na função *onReceive*, que atribui os dados recebidos ao identificador do nó sensor que os enviou, seguindo o padrão chave valor.

Após estabelecer uma conexão com a internet e preparar os dados recebidos para envio, é necessário enviá-los para um ambiente em nuvem. Neste primeiro momento, os envios foram realizados utilizando o protocolo HTTP, por meio de uma requisição do tipo *post*. Foi

necessário a criação de uma nova tarefa com o intuito de serializar a variável temporária em uma *string*, e em seguida passá-la para a função s*endData* que é responsável por fazer o envio através da requisição HTTP. Para efetuar a requisição é necessário passar o endereço do servidor, e uma chave de autenticação.

Ao criar uma planilha do Google, podemos enviar informações e armazená-las de forma personalizada, através da criação de *scripts* no próprio serviço. Foi criado um *script* para separação da mensagem em colunas na planilha, de modo que cada dado coletado fique em uma coluna separada. A Figura 24 mostra os dados recebidos e armazenados no serviço de planilhas do Google. Seguindo da esquerda para direita temos a primeira coluna o número da coleta, em seguida a data e hora em que os dados foram recebidos. A terceira coluna são os dados do sensor de temperatura. Por fim, as seis colunas seguintes os dados coletados dos extensômetros.

Figura 24 – Dados armazenados na planilha do Google.

0	27/09/2021	32.25	0	0	0	0	0	0
1	27/09/2021	32.00	8000	16010051	5358191	1784617	8388608	265
2	27/09/2021	32.50	8000	16012292	5358237	1797025	8388608	16775776
3	27/09/2021	32.50	8056	16012292	5358237	1809544	8388608	16776621
4	27/09/2021	32.75	7830	16012292	5356972	1809544	8388608	16775776
5	27/09/2021	32.75	7830	16074630	5350574	1930884	8388608	941
6	27/09/2021	31.75	7830	16140066	5306847	2048034	8388608	941
7	27/09/2021	32.75	8160	16157246	5271951	2168904	8388608	2299
8	27/09/2021	32.25	8160	16147550	5234635	2168904	8388608	4888
9	27/09/2021	32.50	7419	16170984	5232355	2257837	8388608	5658
10	27/09/2021	32.25	3016	16147550	5178618	2319855	8388608	5658
11	27/09/2021	32.25	3016	15358432	5164174	2319855	8388608	7811
12	27/09/2021	32.50	16769444	14922903	5121415	2084392	8388608	7811
13	27/09/2021	32.00	16772018	14922903	5090411	2084392	8388608	9571
14	27/09/2021	32.25	16774136	14966605	5099240	1924785	8388608	16632494
15	27/09/2021	32.00	16774561	14974161	5099240	1978788	8388608	16632494
16	27/09/2021	32.00	16774561	14991151	5106201	1978788	8388608	18583
17	27/09/2021	32.00	16774561	14991748	5106201	1990310	8388608	13341
18	27/09/2021	32.00	16763476	14991151	5106201	1990310	8388608	12275
19	27/09/2021	31.75	9024	14923480	5151050	1985059	8388608	9983
20	27/09/2021	32.00	6197	14845157	5207134	1767579	8388608	9983
21	27/09/2021	31.50	4690	14845157	5229008	1767579	8388608	9983
22	27/09/2021	31.25	4690	14829071	5238789	1804969	8388608	10782
23	27/09/2021	31.25	4690	14840679	5238628	1871884	8388608	9918

Fonte: Autor.

6.1.5 Etapa 5 - Integração com Amazon IoT Core via MQTT

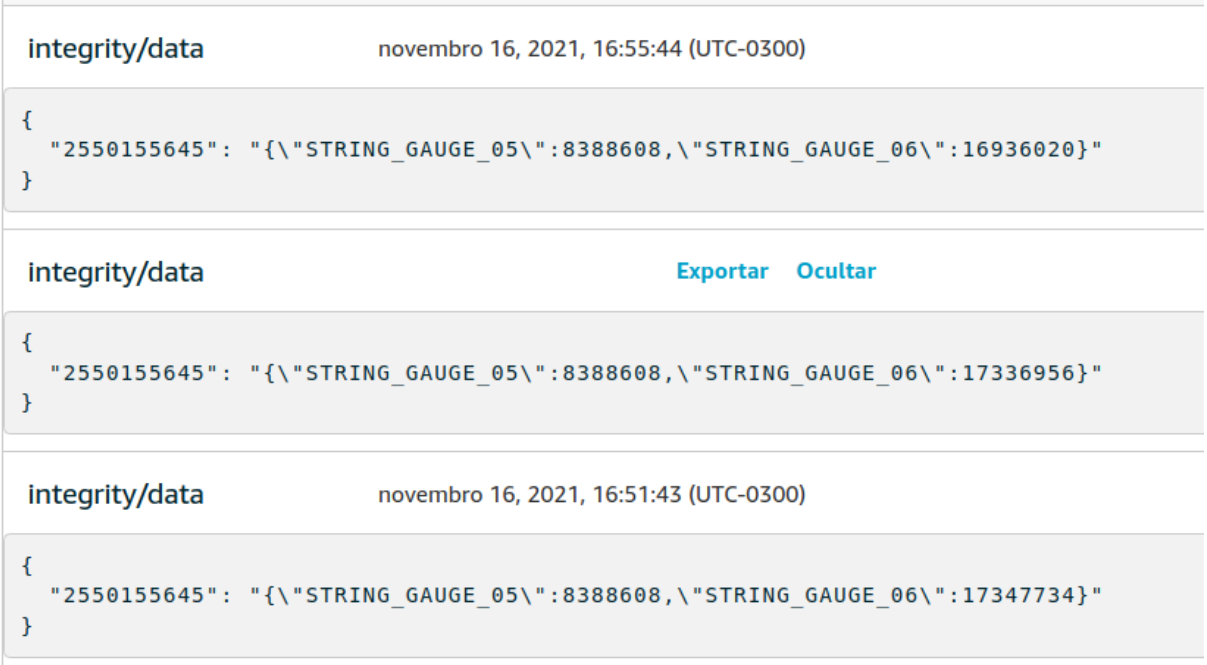
Como visto no capítulo anterior, para estabelecer uma conexão com a plataforma IoT Core da AWS é preciso adicionar os certificados e as chaves de acesso do dispositivo. Para isso, o dispositivo que fará a comunicação deve ser registrado na plataforma, este registro é uma representação do dispositivo físico na nuvem. O registro de um dispositivo segue três etapas:

- Primeira etapa: atribuir um identificador, informar o tipo e o grupo no qual o dispositivo pertence;
- Segunda etapa: gerar certificado e chaves do dispositivo, os arquivos devem ser salvos em local seguro, pois após a criação, as chaves não poderão ser consultadas;
- Terceira etapa: anexar política ao dispositivo. Uma política define as permissões deste dispositivo dentro da plataforma.

A configuração da conexão MQTT foi feita através da biblioteca *MQTTClient*. Com isso, necessita-se informar o tamanho do *buffer* que será utilizado na comunicação, isso é o que definirá o tamanho máximo das mensagens enviadas. Com a criação do objeto do tipo *MQTTClient* é disponibilizado a função *begin*, que recebe três informações como parâmetros: *endpoint* que é o ponto de comunicação com a plataforma da AWS, a porta em que será feita a comunicação e uma conexão segura com a internet. A conexão segura com a internet foi feita com o uso da biblioteca *WiFiClientSecure* que disponibiliza nativamente funções para adição de certificados e chaves do dispositivos.

Para testar a integração e comunicação entre nó sincronizador e a plataforma IoT Core, foi utilizado o cliente de teste MQTT disponibilizado na própria plataforma. Por meio do tópico cadastrado, o console exibe as mensagens recebidas, como mostra a Figura 25.

Figura 25 – Mensagens exibidas no cliente de teste MQTT.



Fonte: Autor.

6.1.6 Etapa 6 - Conexão GPRS

Como visto durante o trabalho, em alguns cenários não é possível se conectar com a internet por uma conexão local via WiFi. Desse modo, foi integrado ao nó sincronizador um módulo de comunicação GPRS modelo SIM800L. Esse módulo, se conecta à internet através de um chip de telefonia móvel. Entretanto, esse módulo não possui suporte nativo com pacotes de segurança, como os utilizados na seção anterior. Visto isso, foi encontrado uma adaptação feita com a biblioteca *SSLClient*. Essa adaptação foi feita por Herbert Parreira, aluno do curso de engenharia mecatrônica da Universidade Estadual de Campinas UNICAMP e autorizada pelo mesmo para ser usada neste trabalho. Mesmo com a implementação da adaptação da biblioteca *SSLClient*, o módulo SIM800L se mostrou instável, apresentando erros de conexão nos testes realizados. Um dos testes onde apresenta um caso de sucesso e uma falha de conexão é mostrado na Figura 26 por meio do monitor serial.

Figura 26 – Mensagens de conexão GPRS exibidas no monitor serial.

```
--- More details at https://bit.ly/pio-monitor-filters
--- Miniterm on /dev/ttyUSB0  115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
setLogLevel: ERROR | STARTUP |
STARTUP: init(): 0
STARTUP: AP tcp server established on port 5555
Solicitando dados dos nós sensores...
Solicitando dados dos nós sensores...
Solicitando dados dos nós sensores...
Nó 2989952141 Dados: {"THERMOPAR":"30.50"}
Nó 2990032209 Dados: {"STRING_GAUGE_03":5279304,"STRING_GAUGE_04":1769054}
Nó 2989922961 Dados: {"STRING_GAUGE_05":8388608,"STRING_GAUGE_06":17269856}
SEM CONEXAO WIFI
Enviando dados via GPRS...
Setup GSM...
SIM800 R14.18
100.82.210.42
Conectando com AWS via GPRS...
Adicionando certificados de segurança...
Sucesso - GPRS Conectado
Sucesso - Dados enviados via GPRS
(SSLClient)(SSL_WARN)(m_run_until): Terminating because the ssl engine closed
(SSLClient)(SSL_ERROR)(flush): Could not flush write buffer!

Conectando ao GPRS, tentativa 1
Restarting GSM Modem failed
Failed to connect to network
```

Fonte: Autor.

6.1.7 Etapa 7 - Armazenamento de dados com DynamoDB

A integração da plataforma IoT Core com o serviço de banco de dados *DynamoDB* é facilitado pelo uso de regras e ações. Na regra é definido a origem das mensagens que serão processadas, com isso, definiu-se que seriam todas as mensagens que chegarem a um determinado tópico. Após essa definição, é necessário indicar que ações devem ser executadas toda vez que a regra for acionada. Como visto no capítulo anterior, a ação utilizada neste trabalho é a *DynamoDBv2*, que separa em colunas cada elemento da mensagem recebida em formato JSON. A 27 mostra uma mensagem recebida pela plataforma, já na 28 demonstra como mensagens desse formato são armazenadas com base na ação *DynamoDBv2*.

Figura 27 – Formato de mensagem recebida.


```
{
  "collect": "1ab03024-65fd-4bce-ad4a-81e99ccedc69",
  "Mac_ESP32": "84:CC:A8:7A:C2:FC",
  "cliente": "LTI",
  "bateria": 89,
  "sensor": "XYZ",
  "coletas": {
    "2550155645": "{\"STRING_GAUGE_05\":8388608,\"STRING_GAUGE_06\":17296856}"
  },
  "status": "Ativo"
}
```


Fonte: Autor.

Figura 28 – Dados armazenados no dynamoDB.

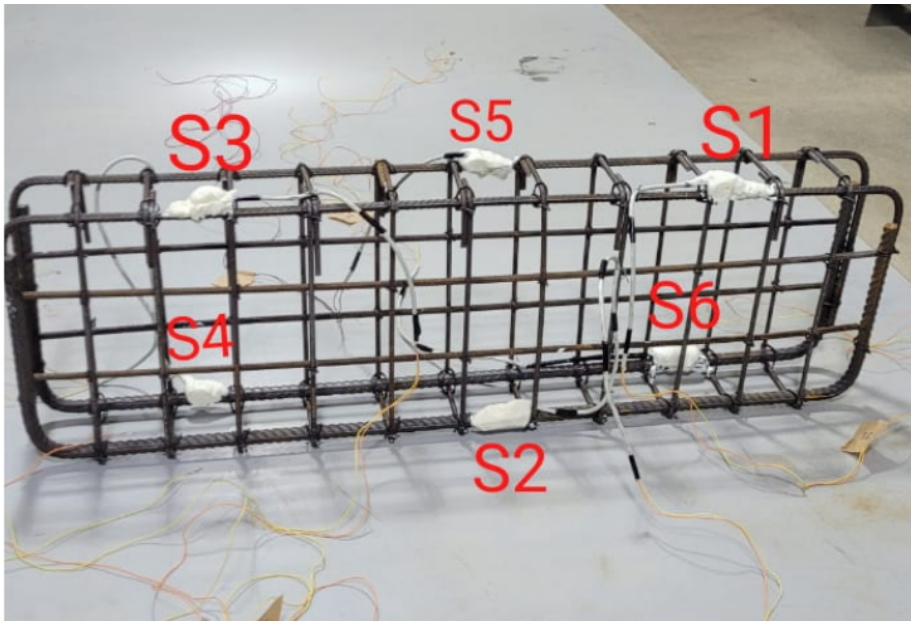
Mac_ESP32	collect	bateria	cliente	coletas	sensor	status
30:83:98:00:52:E4	fb7e660d-ddb3-43f4-9f4d-b1bb5b74e…	89	LTI	{ "2990032209" : { "S" : "{\"STRING_…	XYZ	Ativo
30:83:98:00:52:E4	2d9de99b-66d8-4751-8215-bcff4596a…	89	LTI	{ "2990032209" : { "S" : "{\"STRING_…	XYZ	Ativo
30:83:98:00:52:E4	e685f429-1424-42f5-a5e2-bc8b96612…	89	LTI	{ "2989952141" : { "S" : "{\"THERMO…	XYZ	Ativo
30:83:98:00:52:E4	38384cd8-dc31-4a37-a2eb-8ad9970e…	89	LTI	{ "2989922961" : { "S" : "{\"STRING_…	XYZ	Ativo
30:83:98:00:52:E4	63e58664-2932-4d61-ae9b-07a8bb50…	89	LTI	{ "2989922961" : { "S" : "{\"STRING_…	XYZ	Ativo
30:83:98:00:52:E4	e63b0966-3c14-4bb3-b901-aa4d334e…	89	LTI	{ "2989922961" : { "S" : "{\"STRING_…	XYZ	Ativo
30:83:98:00:52:E4	e8341e13-ddea-4150-b74f-7d0ddd57…	89	LTI	{ "2989922961" : { "S" : "{\"STRING_…	XYZ	Ativo
30:83:98:00:52:E4	16b11d1e-f535-407b-8a12-68b665c9b…	89	LTI	{ "2989922961" : { "S" : "{\"STRING_…	XYZ	Ativo
30:83:98:00:52:E4	7d6c223d-69c3-4af4-8a53-4cb57115f…	89	LTI	{ "2989922961" : { "S" : "{\"STRING_…	XYZ	Ativo

Fonte: Autor.

6.2 Cenário de validação

O cenário de validação foi construído pelos laboratórios LTI e LAREB. Trata-se de uma viga de concreto armado, no qual foram instalados sensores diretamente na armação, que foi posteriormente concretada. Após o processo de cura do concreto, uma solução aquosa que acelera o processo de corrosão foi aplicada regularmente a viga. O objetivo do experimento é coletar dados dos durante os processos de molhagem, e avaliar por meio da técnica de Monitoramento de Integridade Estrutural, a deformação causada pela corrosão da armação da viga. A instalação dos sensores é representada na Figura 31, as nomenclaturas na figura representam os extensômetros. Também foi instalado um sensor do tipo termopar, mas que não é mostrado na figura.

Figura 29 – Distribuição dos extensômetros na armação da viga.



Fonte: Autor.

Para construção do protótipo, foram utilizados quatro nós sensores e um nó sincronizador. Desse modo, um nó sensor ficou encarregado de coletar dados referente a temperatura interna da viga, por meio de um sensor do tipo termopar. Cada um dos três nós sensores restantes, foi constituído de um par de extensômetros, com a finalidade de coletar sinais de deformação. Os componentes necessários para a construção do protótipo são listados na Tabela 4.

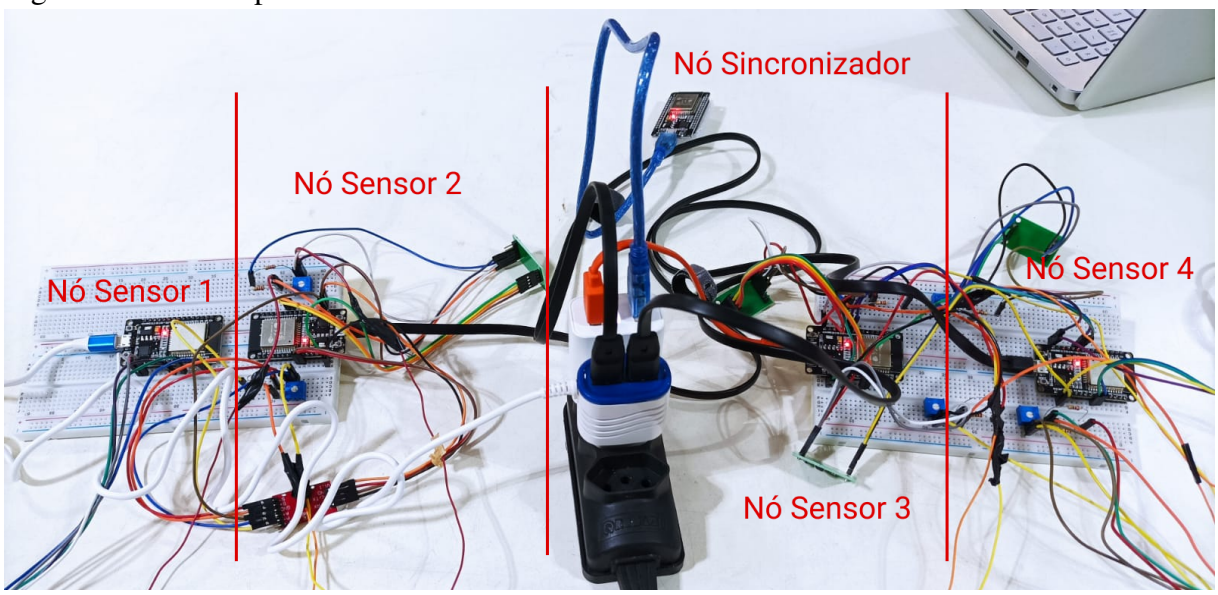
Tabela 4 – Componentes utilizados na construção do protótipo.

Descrição	Modelo/Tipo	Quantidade
Microcontrolador	ESP32	5
Módulo GPRS	SIM800L	1
Protoboard	830 furos	5
Trimpot	Linear	6
Módulo amplificador	HX711	5
Extensômetro	BX120-3AA	6
Termopar	Tipo J	1
Resistor	120r	13
Jumper	Macho/Macho	entre 70 e 100

Fonte: Autor.

Para construção do nó sensor, foi necessário conectar diversos *jumpers* ao microcontrolador, com isso utilizou-se duas *protoboards* para tornar acessíveis todos os pinos do microcontrolador ESP32. Com a limitação da quantidade de *protoboards* disponíveis, dois nós sensores compartilharam das mesmas unidades, como mostra a Figura 30.

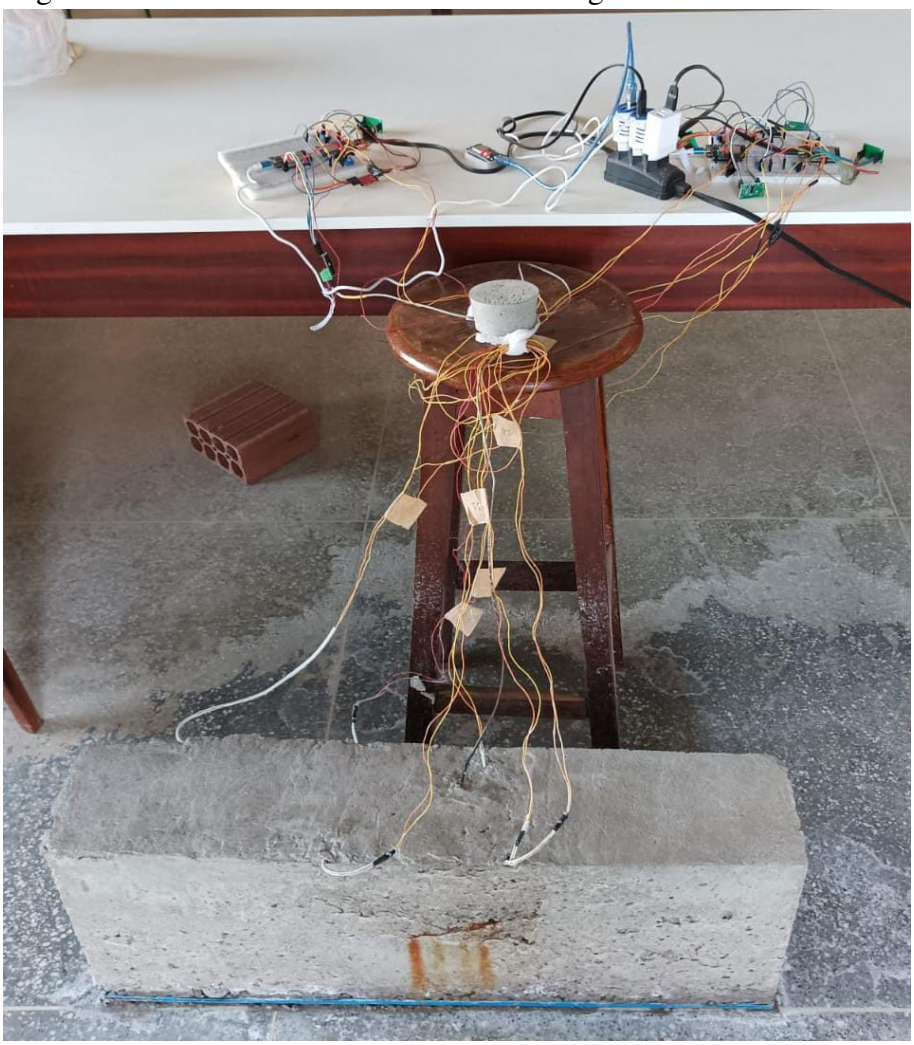
Figura 30 – Protótipo do sistema de coleta.



Fonte: Autor.

A Figura 31 mostra o protótipo construído conectado a viga concretada. Na mesma figura, é possível ver sinais de corrosão provocados pelo processo de molhagem da viga.

Figura 31 – Sistema de coleta conectado à viga.



Fonte: Autor.

A coleta de dados iniciou-se na data de 27 de setembro de 2021, logo após o desenvolvimento da etapa de enviar dados para a nuvem por meio do serviço de planilhas do Google. Nesta etapa, ainda não havia a integração do nó sincronizador com a plataforma IoT Core. Desse modo, seguiu-se o armazenamento de dados através do serviço de planilhas. A etapa de integração com a plataforma IoT Core, assim como a etapa de conexão GPRS e armazenamento com *DynamoDB*, foram testadas em paralelo com o experimento de validação, ou seja, os mesmo dados enviados para a plataforma por meio do protocolo MQTT, também eram enviados para o serviço de planilhas do Google, Dessa forma, os dados do experimento continuaram sendo coletados durante as outras etapas de desenvolvimento e testes. Durante o experimento, foram realizadas 21.926 coletas no período de 77 dias, sendo finalizado em 13 de dezembro de 2021, quando todos os extensômetros pararam de coletar variações. O mesmo serviu para validar todas as etapas vistas na seção anterior, além de disponibilizar um conjunto de

dados para análise. Com esses dados, será possível avaliar por meio da técnica de Monitoramento de Integridade Estrutural, se os sensores conseguiram detectar os danos causados pelo processo de molhagem realizado durante o período de monitoramento.

7 CONCLUSÃO E TRABALHOS FUTUROS

A proposta deste trabalho foi desenvolver um sistema de coleta de dados por meio de rede de sensores sem fio que fosse integrado com serviços de armazenamento em nuvem. Com o intuito de coletar e disponibilizar dados de estruturas na construção civil, de modo que possam ser utilizados na técnica de Monitoramento de Integridade Estrutural MIE a fim de avaliar e detectar possíveis falhas e danos na estrutura monitorada.

Durante o desenvolvimento foram realizados testes em cada uma das etapas, a fim de validar pontos importantes para o prosseguimento do trabalho. Conclui-se que o sistema é viável uma vez que o sistema como todo, funcionou como planejado.

O trabalho torna-se relevante no âmbito da avaliação da integridade de estruturas da construção civil, mas que também tem relevância para trabalhos com propostas semelhantes mas, em contextos diferentes. Pois trata-se de um sistema adaptável a outros domínios, uma vez que toda a comunicação, estratégias de coleta de dados e integração com serviços em nuvem podem ser reaproveitados.

Para trabalhos futuros, planeja-se aplicar o sistema proposto em cenário real visando identificar pontos de melhoria e eventos não observados no cenário de teste de laboratório. Além disso, pretende-se implementar melhorias no software e hardware da aplicação, como: integrar os nós sensores e nó sincronizador com baterias e carregamento fotovoltaico, criar estratégia de economia de energia, testar módulo GPRS com suporte nativo à criptografia e implementar estratégia de armazenamento local e sincronização de dados com a nuvem, e, também, implementar modelo de rede com mais de um nó sincronizador.

REFERÊNCIAS

ABDULKAREM, M.; SAMSUDIN, K.; ROKHANI, F. Z.; RASID, M. F. A. Wireless sensor network for structural health monitoring: A contemporary review of technologies, challenges, and future direction. **Structural Health Monitoring**, v. 19, n. 3, p. 693–735, 2020. Disponível em: <https://doi.org/10.1177/1475921719854528>.

AZURE. **O que é IaaS? infraestrutura como serviço**. 2021. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-iaas/>. Acesso em: 07 julho 2021.

BOJANOVA, I.; ZHANG, J.; VOAS, J. Cloud computing. **IT Professional**, v. 15, n. 2, p. 12 – 14, 2013. ISSN 15209202. Disponível em: <http://search-ebscohost-com.ez11.periodicos.capes.gov.br/login.aspx?direct=true&db=iih&AN=86919932&lang=pt-br&site=ehost-live>.

CORDEIRO, C. de M.; AGRAWAL, D. **Ad Hoc and Sensor Networks: Theory and Applications**. World Scientific, 2011. ISBN 9789814338882. Disponível em: <https://books.google.com.br/books?id=jKbjn_QwHPYC>.

DÍAZ, M.; MARTÍN, C.; RUBIO, B. State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing. **J. Netw. Comput. Appl.**, Academic Press Ltd., GBR, v. 67, n. C, p. 99–117, maio 2016. ISSN 1084-8045. Disponível em: <https://doi.org/10.1016/j.jnca.2016.01.010>.

DWIVEDI, R. K.; KUMAR, R. Sensor cloud: Integrating wireless sensor networks with cloud computing. p. 1–6, 2018.

ESPRESSIF. **ESP-MESH**. 2021. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/mesh.html>.Acesso em: 15 julho 2021.

GTA/UFRJ, G. D. T. E. A. **Redes de Sensores Sem Fio**. 2010. Disponível em: <https://www.gta.ufrj.br/grad/10_1/rssf/desafios.html>.Acesso em: 08 julho 2021.

KARL, H.; WILLIG, A. Protocols and architectures for wireless sensor networks. **Protocols and Architectures for Wireless Sensor Networks**, 2005.

LOUREIRO, A. A.; NOGUEIRA, J. M. S.; RUIZ, L. B.; MINI R. A. D. F.; NAKAMURA, E. F. Redes de sensores sem fio. **Redes de sensores sem fio. In: SN. Simpósio Brasileiro de Redes de Computadores (SBRC)**, p. 179–226, 2003.

MESQUITA, E.; ARÊDE, A.; PINTO, N.; ANTUNES, P.; VARUM, H. H. long-term monitoring of a damaged historic structure using a wireless sensor network. engineering structures. **Elsevier**, p. 108–117, 2018.

NAKAMURA, E. F.; LOUREIRO, A. A. F.; FRERY, A. C. Information fusion for wireless sensor networks: Methods, models, and classifications. **ACM Comput. Surv.**, v. 39, n. 3, p. 9–es, set. 2007. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/1267070.1267073>.

OMOJOKUN, G. A survey of zigbee wireless sensor network technology: Topology, applications and challenges. **International Journal of Computer Applications**, v. 130, p. 47–55, 11 2015.

PRADA, M. A.; TOIVOLA, J.; KULLAA, J.; HOLLMéN, J. Three-way analysis of structural health monitoring data. **Neurocomputing**, v. 80, p. 119–128, 2012.

RAJARAMAN, V. Cloud computing. **Resonance**, v. 19, p. 242–258, 2014.

SILVA, J. B. L. P. Desenvolvimento de sistema para monitoramento de estruturas com utilização da plataforma arduino. **Pontifícia Universidade Católica de Campinas**, 2018.

SILVA, R. N. F. Monitoramento de integridade estrutural utilizando a técnica da impedância eletromecânica aplicada em estruturas de concreto. 2017.

STEEN, M. van; TANENBAUM, A. **Distributed Systems**. 3a edição. ed. [S.l.: s.n.], 2017. ISBN 978-1543057386.

TAVARES, P. L. **Redes de Sensores Sem-fio**. 2002. Disponível em: <https://www.gta.ufrj.br/grad/02_2/Redes%20de%20sensores/Redes%20de%20Sensores%20Sem-fio.htm>.Acesso em: 08 julho 2021.

VADLURI, T.; SAGAR, K. Implementation of home automation system using mqtt protocol and esp32. **International Journal of Engineering and Advanced Technology**, p. 111–113, 12 2018.