Instituto Federal de Educação Ciências e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Curso Superior de Bacharelado em
Engenharia de Computação

Arquitetura de *hardware* e *software* para sistema de controle e sensoriamento remoto residencial

Erick Spinelli Pimentel
Isaque Gabryel Brasileiro de Melo

Campina Grande
2022

Instituto Federal de Educação Ciências e Tecnologia da Paraíba
Campus Campina Grande
Coordenação do Curso Superior de Bacharelado em
Engenharia de Computação

Erick Spinelli Pimentel
Isaque Gabryel Brasileiro de Melo

Arquitetura de *hardware* e *software* para sistema de controle e sensoriamento remoto residencial

Trabalho de conclusão de curso apresentado à Coordenação do Curso Superior de Bacharelado em Engenharia de Computação do IFPB - Campus Campina Grande, como requisito parcial para a conclusão do curso de Bacharelado em Engenharia de Computação.

Orientador: MSc. Henrique do Nascimento Cunha.

Campina Grande
2022

Erick Spinelli Pimentel
Isaque Gabryel Brasileiro de Melo

Arquitetura de *hardware* e *software* para sistema de controle e sensoriamento remoto residencial

Trabalho de conclusão de curso apresentado à Coordenação do Curso Superior de Bacharelado em Engenharia de Computação do IFPB - Campus Campina Grande, como requisito parcial para a conclusão do curso de Bacharelado em Engenharia de Computação.

_____

Henrique do Nascimento Cunha
Orientador

_____

Avaliador 1
Membro da banca

_____

Avaliador 2
Membro da banca

Campina Grande
2022

P644a Pimentel, Erick Spinelli.
Arquitetura de hardware e software para sistema de controle e sensoriamento remoto residencial / Erick Spinelli Pimentel, Isaque Gabryel Brasileiro de Melo. - Campina Grande, 2022.
70 f. : il.

Trabalho de Conclusão de Curso (Curso de Graduação em Engenharia de Computação) - Instituto Federal da Paraíba, 2022.
Orientador: Prof. Me. Henrique do Nascimento Cunha.

1. Sensoriamento remoto 2. Automação 3.Monitoramento remoto I. Melo, Isaque Gabryel Brasileiro II. Título.

CDU 528.8

AGRADECIMENTOS

Erick

Agradeço primeiramente a Deus por estar sempre presente na minha vida, concedendo-me força, capacidade e motivação para enfrentar os obstáculos vivenciados durante este ciclo.

Aos meus pais, por sempre fornecerem apoio emocional, acreditarem em meu potencial e nunca desistirem de me incentivar.

Teço considerações ao amigo Isaque, pela coragem de juntos termos enfrentado e concluído as disciplinas do curso com antecipação, dedicação e compromisso. Sua parceria foi fundamental para que as nossas metas fossem alcançadas.

A todos os meus amigos pelo auxílio, companheirismo, discussões e debates construtivos.

Ao meu orientador Henrique Cunha, por dar o direcionamento necessário para seguir o melhor caminho.

Isaque

A Deus, que fez com que meus objetivos fossem alcançados segundo o seu propósito soberano, e sobretudo sempre manteve sua graça evidente aos meus olhos nos dias mais difíceis, me sustentando em minha debilidade.

A minha família, que sempre esteve ao meu lado, com grande paciência nos momentos árduos e com compreensão na minha ausência.

Externo também meus agradecimentos ao meu companheiro de jornada Erick, que desde o início desse percurso académico esteve comigo em todos os desafios e sempre se fez disponível quando precisei.

Aos meus familiares e amigos, por toda cordialidade, suporte e palavras de apoio.

Ao grande mestre Henrique Cunha que acompanhou minha jornada de perto desde o primeiro período, e somou de forma ímpar em meu conhecimento. Além de possibilitar momentos de alegria, diversão e insanidade junto com nossa equipe de robótica. É uma honra tê-lo como orientador.

RESUMO

Hodiernamente, fundamentando-se na busca por eficiência temporal e laboral, a automação tem se tornado cada vez mais presente no cotidiano popular. Adjunto a isso, inclui-se o monitoramento sensorial, que possibilita o acompanhamento dos dados de sensores, inclusive de maneira remota. Além disso, é possível controlar atuadores do mesmo sistema em função de uma análise dos dados coletados dos sensores. No entanto, para a implementação de um sistema de controle e monitoramento sensorial remoto é necessário o estudo de diversas áreas de conhecimento, como programação, protocolos de comunicação, *hardware*, comunicação de rede, entre outros. Contudo, percebe-se que parte dessas podem ser abstraídos e generalizados por um artefato, possibilitando ainda assim, que seja aplicável em diversas áreas. Deu-se como proposta, portanto, a criação de um sistema que possibilita o sensoriamento, monitoramento remoto e controle de atuadores fundamentando-se nos dados provenientes de sensores. Dados esses que também podem ser utilizados para criação de notificações customizadas. A ativação desses gatilhos fundamenta-se no uso de condicionais lógicas, pelas quais, estão inseridas dentro do conceito nomeado de “gatilhos”, podendo ser estes de atuação ou notificação.

Palavras-chave: Sensoriamento, automação, monitoramento remoto, IoT.

ABSTRACT

Nowadays, due to the search for temporal and labor efficiency, automation has become increasingly present in popular daily life. Furthermore, there is also sensory monitoring, which makes it possible to monitor sensor data, even remotely. Moreover, it is possible to control actuators of the same system based on an analysis of the data collected from the sensors. However, for the implementation of a remote control and sensor monitoring system, it is necessary to be aware of several areas of knowledge, such as programming, communication protocols, hardware, network communication, among others. Nevertheless, it is clear that part of these can be abstracted and generalized by an artifact, still allowing it to be applicable in several areas. Therefore, it was proposed the creation of a system that allows the sensing, remote monitoring, and control of actuators based on the data sourced from the sensors. These data can also be used to create custom notifications. The activation of these is based on the use of logical conditionals, by which they are inserted within the concept named triggers, which can be action or notification.

Keywords: Sensing, automation, remote monitoring, IoT.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

API - Application Programming Interface
BLE - Bluetooth Low Energy
CD - Continuous deployment
CI - Continuous integration
CVS - Concurrent Versions System
DB - Database
DOM - Document Object Model
ECDSA - Elliptic Curve Digital Signature Algorithm
EMI - Electromagnetic Interference
GPIO - General Purpose Input and Output
GSM - Global System for Mobile Communications
HMAC - Hash-based Message Authentication Code
HTTP - HyperText Transfer Protocol
I2C - Inter-Integrated Circuit
IoT - Internet of Things
IP - Internet Protocol
JS - JavaScript
JSON - JavaScript Object Notation
JWT - JSON Web Token
KVS - Key Value Storage
LDR - Light Dependent Resistor
ORM - Object–relational mapping
PC - Personal Computer
pH - Potencial hidrogeniônico
PoE - Power over Ethernet
RAM - Random Access Memory
REPL - Read–eval–print loop
REST - Representational State Transfer
ROM - Read Only Memory
SCK - Serial clock
SDA - Serial data line
SHMM - Smart House Monitor & Manager

SMS - Short Message Service

SQL - Structured Query Language

SSID - Service Set Identifier

SVN - Apache Subversion

UI - User interface

URL - Uniform Resource Locator

USD - United States Dollar

UX - User experience

VM - Virtual Machine

WSN - Wireless sensor network

WWW - World Wide Web.

XML - Extensible Markup Language

CAPÍTULO 1

1. INTRODUÇÃO

1.1 CONSIDERAÇÕES PRELIMINARES

Segundo Groover, Mikell (2014) “A automação descreve uma ampla gama de tecnologias que reduzem a intervenção humana nos processos. (...)”. Hodiernamente, fundamentando-se na busca por eficiência temporal e laboral, a automação tem tornando-se presente, trazendo praticidade, melhor qualidade de desenvolvimento e maior consistência nos processos. Com base nisso, nota-se a eliminação de tarefas manuais trabalhosas para a esfera humana, processo visto com assiduidade durante a segunda revolução industrial e propiciada pela industrialização (SOUSA, R. Segunda Revolução Industrial, 2022).

Com avanço desse processo, adentramos na quarta revolução industrial, pelo qual, destaca-se o aumento da interconectividade e a automação inteligente. Adjunto a isso, inclui-se o monitoramento sensorial, que possibilita a originação de diversas informações em função dos dados, além de derivar ações que precisam ser tomadas baseando-se em uma análise que também pode ser automatizada (Automação de Processos Industriais e a Indústria 4.0 - Total Automação Industrial, 2020).

No entanto, para a implementação de um sistema de controle e monitoramento sensorial remoto é necessário o estudo de diversas áreas de conhecimento, como programação, protocolos de comunicação, *hardware*, comunicação de rede, entre outros. Além disso, os custos para desenvolvimento são elevados e requer profissionais altamente requisitados no mercado.

Todavia, em parte dos cenários de monitoramento sensorial e controle de atuadores, nota-se a possibilidade de generalização das soluções. Isso se dá porque tais situações tipicamente baseiam-se na leitura dos dados, envio à nuvem para armazenamento e possível uso desses para ativação de atuador, que se dá por uma análise desses dados por condições lógicas. Sendo assim, o desenvolvimento de uma tecnologia adaptável fundamentada em um alto nível de abstração consegue absorver boa parte do processo.

Partindo dessa necessidade, estruturou-se o projeto ora descrito. É proposta, portanto, a criação de um sistema que possibilita o sensoriamento, monitoramento remoto e controle de atuadores fundamentando-se nos dados provenientes dos sensores.

O sistema proposto estrutura-se em dois módulos, interface de comunicação e aplicação web. O primeiro deles destina-se a fornecer um dispositivo que possibilitará a inserção de sensores genéricos (delimitados em uma lista de suporte), pelo qual, terão seus dados devidamente enviados para um serviço de armazenamento na nuvem. Além disso, esse dispositivo também terá portas de saída que poderão ser ativadas partindo de uma lógica definida pelo segundo módulo.

A aplicação por sua vez terá como um dos propósitos a definição dos sensores conectados. Ademais, nela será possível definir gatilhos de ativação ou desativação de atuadores registrados, que serão compostos de lógica simples baseando-se nos dados dos sensores, possibilitando ao usuário a automação e monitoramento remoto de forma simples e objetiva.

1.2 OBJETIVOS

1.2.1 Objetivo geral

O objetivo deste trabalho consiste no desenvolvimento de uma arquitetura de *hardware* e *software* para controle e sensoriamento com acesso aos dados por meio de dispositivos com acesso à internet.

1.2.2 Objetivos específicos

- Possibilitar o sensoriamento e realizar o envio dos dados para a nuvem
- Permitir a visualização dos valores mais recentes de cada um dos sensores de forma remota
- Permitir a visualização do histórico de valores dos sensores
- Permitir a criação de gatilhos de notificação e atuação. Define-se como gatilho uma condicional lógica estabelecida em função dos dados de um sensor, que resultará em uma ação final.
- Fornecer uma plataforma simples, genérica, de custo reduzido, eliminando a necessidade de conhecimentos de programação e protocolos de comunicação.
- Fornecer segurança e autenticação: somente o usuário autorizado pode acessar o sistema para gerenciar, controlar e monitorar suas informações.

CAPÍTULO 2

2. FUNDAMENTAÇÃO TEÓRICA

Nos tempos modernos, é evidente a necessidade da realização de monitoramento sensorial e controle de atuadores de forma autônoma, tanto para remotamente realizar ações, quanto para acionar atuadores em função dos dados. Pode-se destacar aplicações em diversas esferas da sociedade, como agricultura, indústria e o universo ascendente da *Internet of Things* (KUMAR; TIWARI; ZYMBLER, 2019). Porém, na maioria dos sistemas existentes com tal propósito, o usuário regular necessita ter demasiado grau de conhecimento com automação ou necessita pagar valores expressivos em soluções industriais (Controle de Automação Industrial, 2022).

2.1 A TECNOLOGIA

Os sistemas de controle e monitoramento sensorial existem há alguns anos e vem ganhando reconhecimento com o avanço da tecnologia sem fio. Dentre os objetivos desse ramo, o alvo mais destacado é fornecer a infraestrutura necessária para acessar sensores de forma transparente, processos e atuadores usando protocolos padronizados independentemente do hardware, sistemas operacionais ou localização (PRESSER, 2009).

As aplicações de controle e sensoriamento remoto começaram a se popularizar com a maior adesão de código aberto e *hardware* aberto. Partindo disso, surgiram novas tecnologias nos últimos anos, diferentes pesquisadores, designers e hobistas começaram a desenvolver plataformas extensíveis, registrando cada vez mais dados do ambiente, e por conseguinte, possibilitando maior nível de automação (MCGRATH; SCANAILL; NAFUS, 2013).

Alguns trabalhos destacam-se no meio acadêmico nesse viés, dentre essas iniciativas podemos destacar os seguintes:

Rao et.al., (2016) desenvolveram um dispositivo embarcado usando Arduino UNO com conexão Wi-Fi para monitoramento de temperatura, umidade, pressão, intensidade de luz, níveis de intensidade sonora, níveis de dióxido de carbono na atmosfera e com capacidade de armazenamento dos dados na nuvem. Este sistema tem um microcontrolador Atmega328 como principal unidade de processamento para todo o sistema.

Ram e Gupta (2016) desenvolveram um registrador de dados para monitoramento climático, muito semelhante ao proposto por Rao et al. (2016) nos parâmetros observados: temperatura, dióxido de carbono, umidade e intensidade da luz. Contudo, utilizando o ESP8266 como dispositivo de comunicação.

Prescott et. al., (2016) produziram uma plataforma de código aberto compatível com Arduino chamada *HydroSense* para monitoramento hidroclimático, pelo qual, incluem pH, redução do oxigênio, oxigênio dissolvido, condutividade, turbidez, temperatura e profundidade da água.

Wijnen, et.al., (2014) desenvolveram uma plataforma de código aberto de teste de qualidade de água. Bitella et.al., (2014) construíram uma plataforma para monitorar o teor da água do solo e de vários parâmetros do ar, solo e vegetação - baseada em Arduino. Masseroni et.al., (2016) desenvolveram um dispositivo semelhante utilizando a tecnologia Arduino para o monitoramento contínuo do potencial hídrico do solo para auxiliar o agendamento da irrigação do campo.

Tseng et.al., (2014) propuseram uma aplicação IoT que disponibiliza sensoriamento residencial e controle de atuadores de forma remota, no qual nomearam de *Smart House Monitor & Manager* (SHMM). Em essência, todos os sensores e atuadores são conectados por uma rede sem fio Zigbee. Foi projetado um *smart socket* simples controlado remotamente via Zigbee, fazendo com que todos os aparelhos conectados pela tomada inteligente possam ser controlados pelo SHMM. Um host de PC é usado como coletor de dados. Na nuvem, uma máquina virtual (VM) é usada para processamento e armazenamento de informações. O usuário pode usar o PC ou telefone Android para monitorar ou controlar sua *smart house* via internet.

Kumar (2014) apresentou um sistema para controle e monitoramento remoto do ambiente de casa inteligente. O projeto consiste em uma aplicação desenvolvida utilizando a plataforma Android e um micro servidor web baseado em Arduino *Ethernet*. A plataforma Arduino é utilizada como o principal controlador que hospeda o micro servidor web e executa as ações necessárias. Os sensores e atuadores / relés são conectados diretamente ao controlador principal. O ambiente *smart home* pode ser controlado e monitorado a partir de um local remoto usando o aplicativo da casa inteligente, que se comunica com o micro servidor web via internet.

O sistema de automação residencial proposto por Ahmed ElShafee e Karim Alaa Hamed (2012) é dividido em uma API de servidor e firmware para a plataforma

Arduino. A API do servidor é uma aplicação web construída usando asp.net. Tal aplicação pode ser acessada da rede interna ou da internet se o servidor tiver IP real na internet usando qualquer navegador de internet que suporte a tecnologia asp. O *software* de aplicação do servidor é responsável pela instalação, configuração e manutenção de todo o sistema de automação residencial. Para o banco de dados, foram utilizados arquivos XML para salvar o log dos componentes. Já o *software* do Arduino, é responsável por coletar eventos de sensores conectados, e então aplicar a ação dos atuadores pré-configurados no servidor.

O sistema desenvolvido por Temilselvi et al., (2020) foi projetado usando uma técnica IoT conhecida como ThingSpeak. Esse sistema usa dois sensores: um sensor que monitora o piscar de olhos, e outro sensor que monitora a saturação de oxigênio de um paciente em coma. Esses sensores são conectados ao microcontrolador para monitorar os parâmetros de saúde dos pacientes. Caso seja encontrada alguma anormalidade em um dos parâmetros de saúde, o microcontrolador aciona imediatamente uma mensagem de alerta através do módulo GSM e módulo Wi-Fi. Nesta aplicação, a estrutura proposta usou sensores de saúde Nemours, como sensor de temperatura, sensor de piscar de olhos, sensor de batimentos cardíacos, sensor de movimento corporal e sensor SPO2. Esses sensores são responsáveis por transmitir os dados médicos utilizando o módulo Wi-Fi ESP8266, fazendo com que os dados do paciente possam ser salvos, analisados, exibidos em formas de gráficos e podem ser visualizados por meio de aplicativo móvel.

Mainwaring et al., (2002) desenvolveram um sistema de redes de sensores sem fio com o objetivo de monitoramento de habitat. A aplicação foi desenvolvida em uma arquitetura de camadas. O nível mais baixo consiste nos nós sensores que realizam computação e *networking* de propósito geral, além de realizar o sensoriamento. Os nós sensores transmitem seus dados através da rede de sensores para o *gateway* da rede de sensores. O *gateway* é responsável por transmitir os dados do sensor, provenientes do *patch* do sensor, através de uma rede local para a estação base que fornece conectividade *WAN* e registro de dados. A estação base se conecta ao banco de dados pela Internet. Por fim, os dados são exibidos por meio de uma interface de usuário.

O sistema proposto por Kanagaraj et al., (2015) consiste em seis estações meteorológicas implantadas e conectadas à estação base, por meio de nós WSN.

Essas estações base locais coletam e armazenam os dados da estação meteorológica, nós WSN com sensores adicionais e integridade da rede para fins de diagnóstico antes de enviar todos os dados para um servidor de nuvem central. O cenário de implantação de rede de sensores sem fio para realizar o sensoriamento consiste em três camadas de software distintas. Além de uma camada de nuvem adicional colocada entre a camada do servidor e a camada do cliente, para conectar as estações base distribuídas.

A aplicação proposta neste trabalho é fundamentada em um sistema de monitoramento de sensores, com a adição da funcionalidade de acionamento de atuadores em função dos dados provenientes dos sensores. Ademais, os sensores do sistema proposto suportados na versão inicial possuem grande disponibilidade no mercado e são amplamente conhecidos no mundo da prototipagem. No entanto, além disso, deve-se destacar que a construção da API possui um funcionamento abrangente, tornando simples e objetiva a agregação de sensores, dispensando mudanças na API e sendo limitado apenas pelo controlador utilizado.

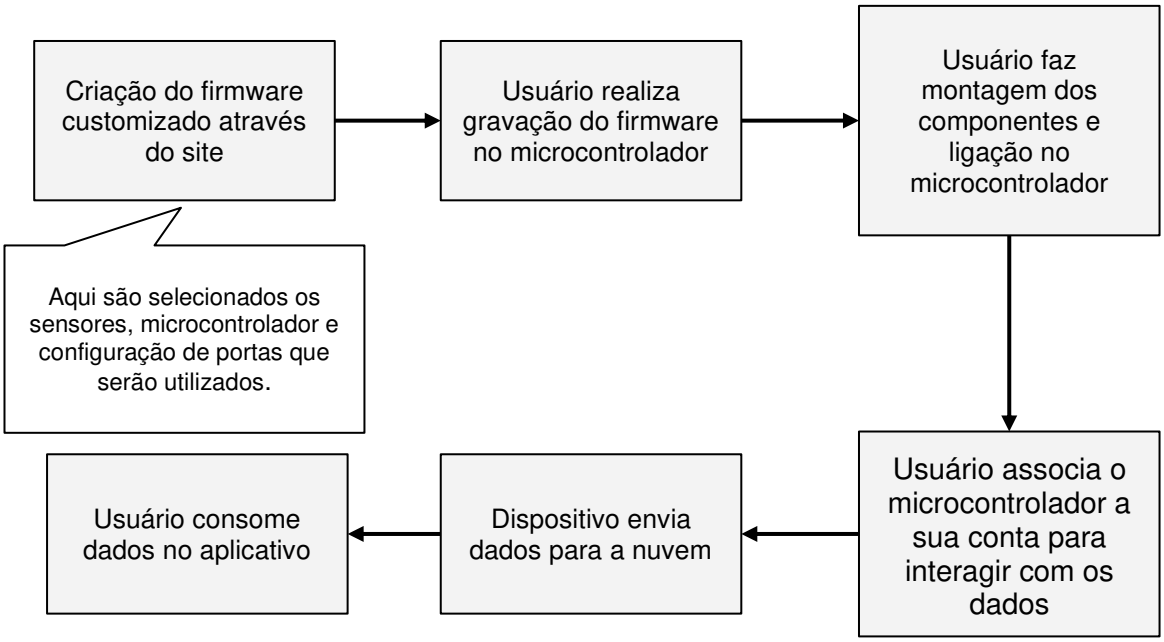
2.2 PROPOSTAS DO MERCADO

Destacam-se no mercado diversas soluções sólidas, que atendem diferentes nichos de usuários. Nestes, podemos listar Sensatio, Swift Sensors, Ruuvi, Monnit, dentre outros. No entanto, o diferencial do projeto proposto aos que já existem é toda a arquitetura focada em um sistema extensível e direcionado ao usuário, de comportamento simples, baixo custo e personalizável para distintas aplicações. Lista-se abaixo as características dos principais produtos analisados.

2.2.1 Sensatio®

O Sensatio® (2022) é um produto focado em monitoramento sensorial remoto desenvolvido pela empresa austríaca Sensate Digital Solutions GmbH. Sua construção é focada em uma aplicação para dispositivos móveis multiplataforma que se comunica com um servidor. Para realização do sensoriamento é utilizado um microcontrolador da família ESP, pelo qual os sensores são conectados via cabos e seus dados transmitidos através da rede para a nuvem, que armazena-os e possibilita o consumo por parte do usuário. Além disso, a aplicação possibilita a criação de *push notifications* em função dos dados dos sensores cadastrados. Na figura 2.1 temos uma visão geral de como funciona essa plataforma.

Figura 2.1 - Visão geral do funcionamento do Sensatio®



Fonte: Elaboração própria a partir de informações do site do Sention, 2022.

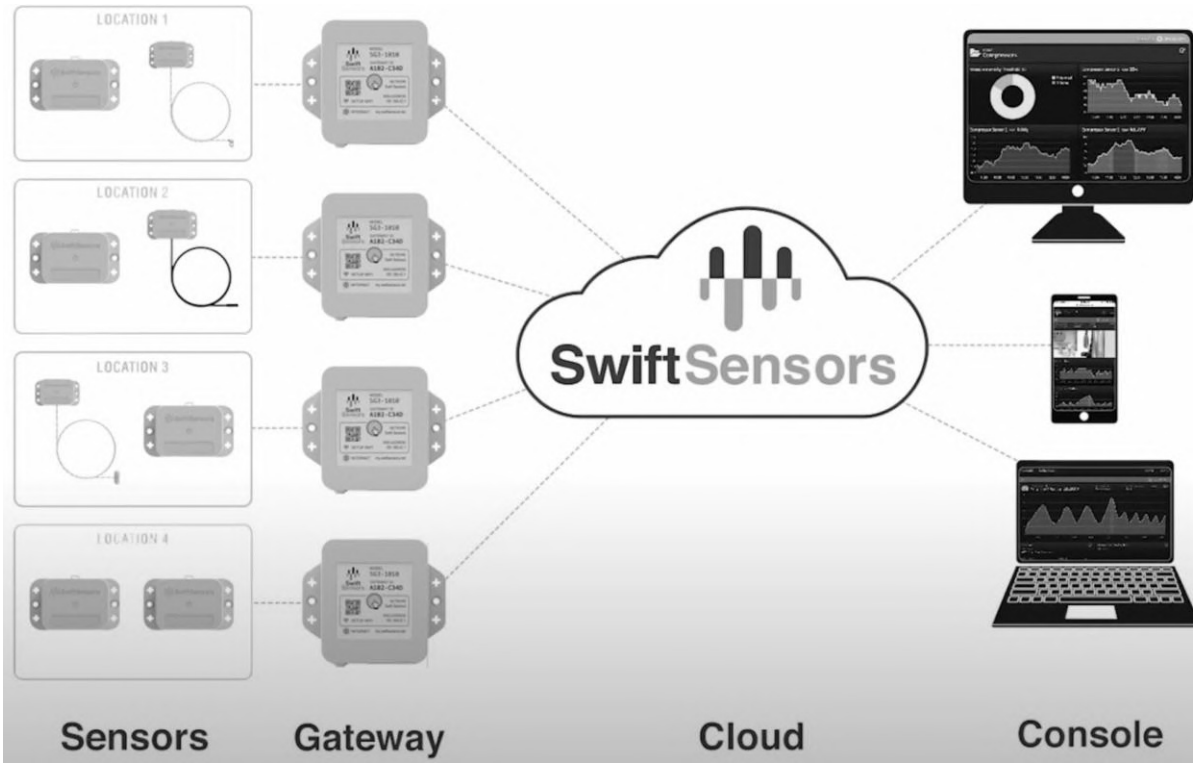
O Sensatio® apresenta uma plataforma robusta para monitoramento sensorial e que consegue possuir um baixo custo, e atingir uma grande variedade de sensores. No entanto, demanda do usuário conhecimentos técnicos, como por exemplo, realizar gravação de firmwares, fazer escolha de componentes, realizar montagem etc. A aplicação apresenta uma interface amigável e direta, mas baseando-se na premissa de que a aplicação só pode ser utilizada em conjunto com um dispositivo que demanda certo conhecimento, percebe-se a problemática.

2.2.2 Swift Sensors

Swift Sensors® (2022) é uma empresa focada no desenvolvimento de sensores sem fio para equipamentos industriais de uso crítico. A empresa possui em seu catálogo diversos tipos de sensores de construção proprietária, além de também fornecer um *gateway* que é necessário para realizar o envio dos dados dos sensores para a nuvem - vale destacar que o uso desse dispositivo é indispensável para o funcionamento.

Acrescido a isso, para o monitoramento remoto é fornecido um serviço web que exibe os valores e também que possibilita a criação de alertas em função dos dados, podendo realizar notificações através de chamadas, SMS e e-mail. Na figura 2.2 podemos verificar a visão geral do funcionamento.

Figura 2.2 - Visão geral do funcionamento do Swift Sensors®



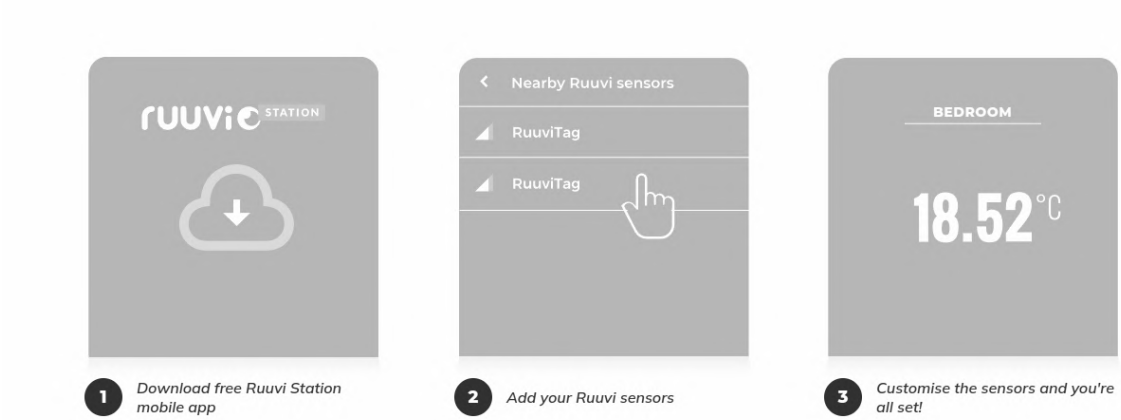
Fonte: Swift Sensors®, 2022.

Apesar de seu simplório processo de uso, por se tratar de uma solução com ênfase industrial, todos os dispositivos apresentam um alto custo para o usuário final e por tratar-se de uma empresa que produz seus sensores, não existe a possibilidade de integração de outros à sua API. Além disso, cada sensor pode custar na data de consulta (29 de agosto de 2022) de USD$109,99 à USD$399,99, também é demandado para seu funcionamento um *gateway* custando de USD$149,00 à USD$399,99. Ademais, se faz necessário a assinatura de um plano de dados para envio das informações à conta associada que as consumirá.

2.2.3 Ruuvi

O Ruuvi® (2022) é uma empresa finlandesa de monitoramento focado em medições de temperatura, umidade e pressão atmosférica. Como produto principal destaca-se o RuuviTag que é um sensor *wireless bluetooth* para realização de medições. Para complemento, também é possível utilizar o RuuviGateway Router*,* um dispositivo para tornar dados dos sensores acessíveis à internet, haja vista que o sensor possui apenas conexão *bluetooth*. Para o caso de uso mais simples podemos acompanhar o funcionamento pela figura abaixo:

Figura 2.3 - Visão geral do funcionamento do Ruuvi® sem utilizar o RuuviGateway Router



Fonte: Ruuvi®, 2022.

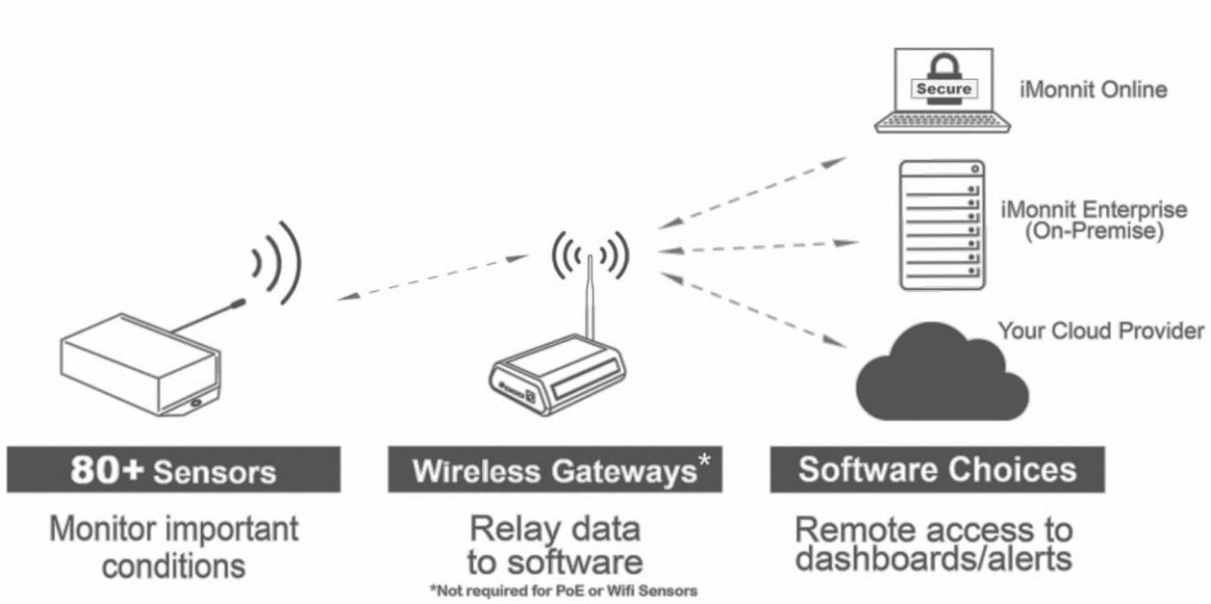
O funcionamento do produto principal em questão é simplório de custo mediano. No entanto, deve-se considerar a limitação do módulo do sensor apenas contemplar a conexão *bluetooth*. Tal fato cria limitações de acessibilidade remota, que é sanada com o uso de um outro produto fornecido (*RuuviGateway Router),* que realiza o fornecimento dos dados à internet. Mas, nota-se que para maioria dos casos de usos domésticos tal funcionalidade não interfere em seu uso.

Ademais, destaca-se o fato da disponibilidade de apenas 4 tipos de dados (temperatura, umidade e pressão atmosférica) provenientes dos sensores, e que não são extensíveis.

2.2.4 Monnit

Monnit (2022) é uma empresa que constrói soluções industriais com a proposta de fornecer os dados necessários para auxiliar os usuários na tomada de decisões em suas empresas, portanto, caracteriza-se de um sistema de monitoramento remoto. A empresa possui uma ampla variedade de 80 sensores próprios, que se comunicam com um *Wireless Gateway*, transmitindo os dados relacionados para o sistema escolhido pelo usuário. Tal *Gateway* não se faz necessário caso o sensor seja Wi-Fi ou PoE. O cliente pode ainda escolher consumir dados através de três diferentes plataformas: iMonnit Online, através de um provedor de serviço de nuvem como Microsoft Azure, Amazon Web Services, IBM Watson, uma aplicação customizada, ou através da iMonnit Enterprise, que na data de consulta, está em desenvolvimento.

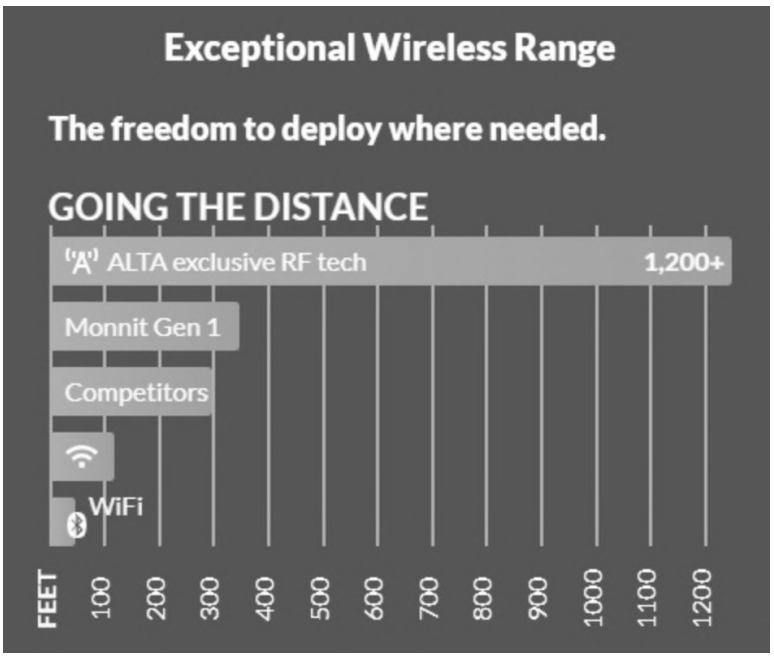
Figura 2.4 - Visão geral do funcionamento do Monnit®



Fonte: Monnit®, 2022.

Outrossim, um dos pontos positivos a se destacar do Monnit é a proposta da distância do alcance sem fio de seus dispositivos, como pode-se visualizar na figura 2.5.

Figura 2.5 - Comparação alcance sem fio Monnit®



Fonte: Monnit®, 2022.

Apesar do seu sistema robusto e amplo alcance sem fio dos seus dispositivos, o Monnit tem altos preços, por se tratar de uma solução que visa atender o comércio e a indústria.

Na tabela abaixo são comparados os sistemas analisados:

Tabela 2.1: Comparativa entre os analisados.

Sistema	Faz uso de *gateway* para comunicação?	Interface	Aplicações
Sensatio	Sim	Mobile	Todos os sensores genéricos com suporte oferecidos ao ESP
SwiftSensors	Sim	Mobile	Temperatura, umidade, presença de água, vibração, sensor de porta, inclinação, atividade, interruptor, corrente, tensão
Ruuvi	Não	Mobile e web	Temperatura, umidade, altitude
Monnit	Não	Web	Temperatura, umidade, aberto-fechado, movimento infravermelho, tensão, luz, corrente trifásica, medidores de vibração, detecção de água, couters de pulso, pressionamento de botão, detecção de gás, resistência, ultrassônico, qualidade do ar, velocidade do ar, pressão do ar, umidade do solo
Sention	Sim	Web + PWA	Todos os sensores genéricos com suporte ofertado ao ESP

Fonte: De autoria própria, 2022.

CAPÍTULO 3

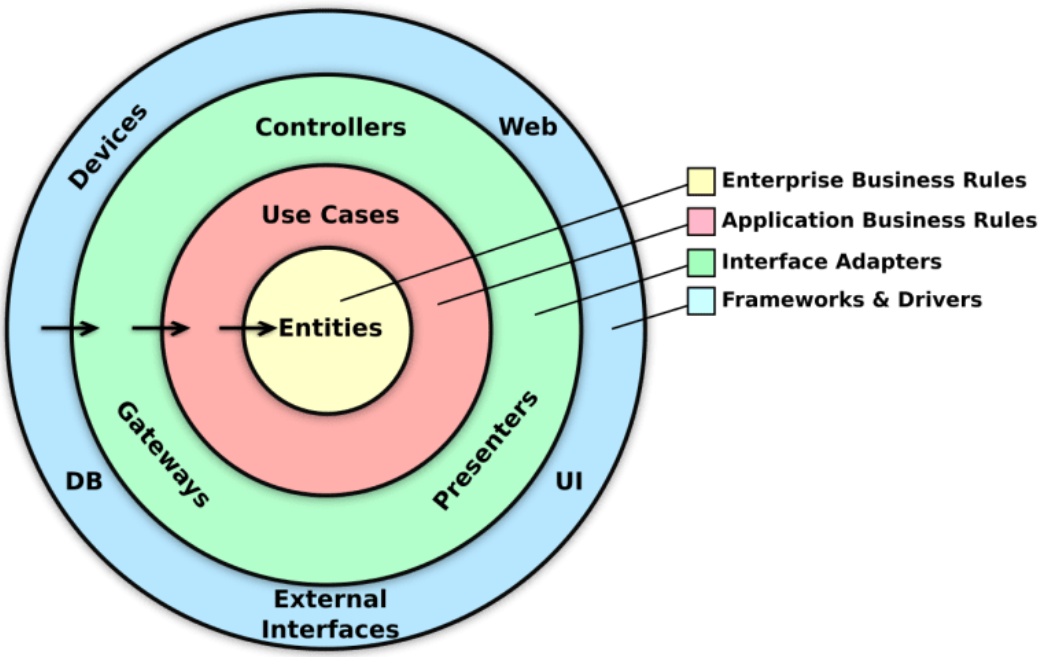
3. METODOLOGIA

3.1 BACKEND

3.1.1 Arquitetura de código

A arquitetura de código da API, de forma simplória, segue o modelo proposto por Robert Cecil Martin, explanado em sua obra "*Clean Architecture: A Craftsman's Guide to Software Structure and Design*" (2017). Além disso, em seu livro são detalhadas as vantagens desta arquitetura em comparação a outras. Pode-se demonstrar de forma usual a arquitetura limpa através da seguinte figura:

Figura 3.1- Arquitetura limpa proposta por Robert Martin



Fonte: *The Clean Code Blog*, 2022.

As vantagens de utilizar uma arquitetura em camadas são diversas, dentre essas podemos pontuar as que mais se destacam:

- **Independente da interface do usuário**:

Partindo da segregação das responsabilidades, a interface do usuário pode mudar facilmente, sem alterar o restante do sistema. Uma interface web pode ser substituída por uma aplicação de linha de comandos, por exemplo, sem alterar as regras de negócios, já que fazem parte de camadas distintas.

- **Independentemente de qualquer agente externo:**

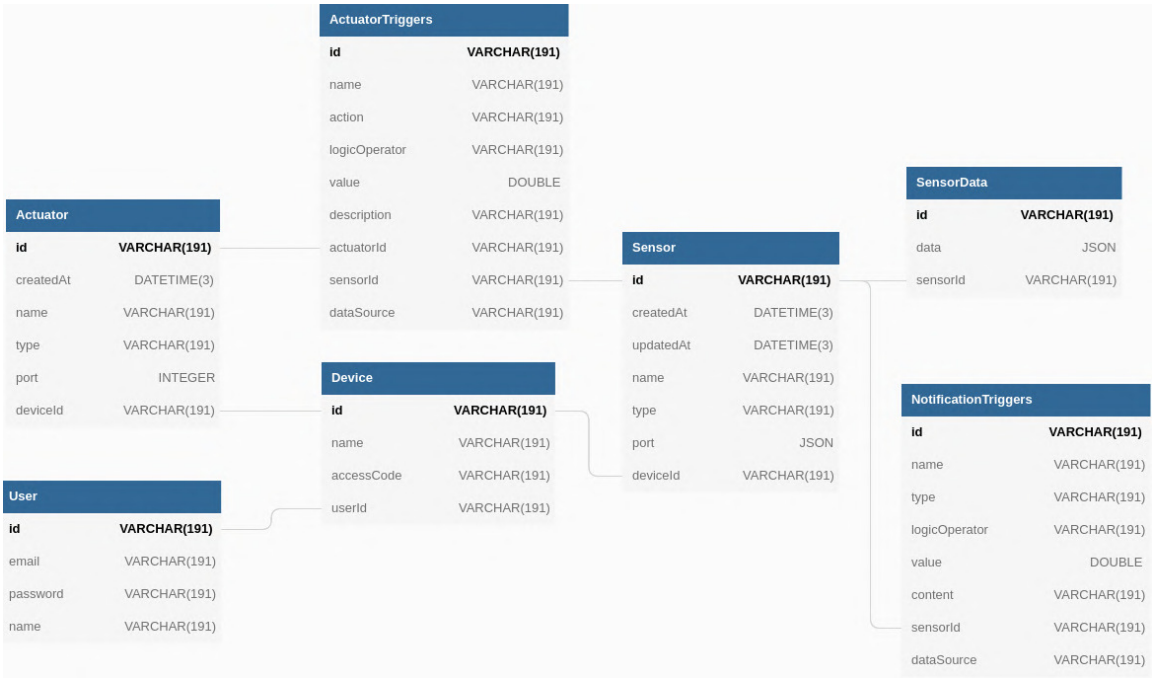
Propiciado pela separação das regras de negócios, qualquer agente externo pode ser substituído sem influenciar no funcionamento da aplicação. Na realidade, as regras de negócios simplesmente não sabem nada sobre o que está no seu exterior, não estão ligadas a nenhum *framework* ou qualquer outro tipo de dependência.

3.1.2 Arquitetura do sistema

Entidades

Na figura 3.2 é exposto o diagrama de entidades e relacionamentos explicitados nas entidades em seguida.

Figura 3.2 - Diagrama de entidades e relacionamentos



Fonte: De elaboração própria, 2022.

3.1.2.1 User

Um *User* representa o cliente da aplicação. Fazem parte de sua estrutura os campos *email* e *password* que serão utilizados para autenticação do usuário. Para garantir a segurança do armazenamento do dado, o campo *password* é encriptado pelo *bcrypt* (descrito posteriormente) na camada de negócios, garantindo que o texto plano seja criptografado e devidamente protegido.

3.1.2.2 Device

O *Device* dá-se como uma abstração de um microcontrolador, como por exemplo, um ESP32. Ele é composto por *sensors* e *actuators*, que representam os sensores e atuadores conectados no dispositivo em questão. Ademais, o *device* é possuído por um usuário e toda manipulação dessa entidade deve ser autenticada com as credenciais do possuidor.

3.1.2.3 Sensor

O *Sensor* é a representação genérica de uma fonte de informação proveniente de uma das portas do *device*.

Cada sensor possui seu comportamento e formato de dados, em sensores analógicos temos o retorno de dados expressos em um único valor de tensão lido pelo microcontrolador (0 até 3.3v mapeados digitalmente no intervalo de 0 a 4095). Já em sensores com comunicação I2C, como o caso do BMP280, vários valores são fornecidos (temperatura, pressão e derivados através desses a altitude) e sua leitura não é feita de forma direta, necessitando de manipulação. Dessa forma, cada tipo de sensor tem uma interface distinta para consumo de dados, e para realizar a diferenciação entre os tipos de interface de sensores é aplicado o uso do campo *type*, que é utilizado pelo microcontrolador para selecionar qual interface deve ser usada para leitura.

Outrossim, essa entidade possui também um campo *port* que especifica de qual porta serão lidos os dados. Esse campo pode representar apenas um valor ou um conjunto de valores, como por exemplo para o caso da comunicação I2C que faz uso das portas SCK e SDA.

Por fim, também é disposto o atributo *name,* que se trata de um nome para que o usuário identifique o sensor com algo de sua preferência.

3.1.2.4 SensorData

SensorData é a entidade que representa os dados que são provenientes do sensor, podendo ser do tipo JSON ou inteiro dependendo da interface estabelecida pelo fornecedor da informação. Tal variação dá-se com o intuito de possibilitar o suporte a sensores que transmitem apenas um dado, ou vários dados, como supracitado o BMP280.

3.1.2.5 NotificationTrigger

O *NotificationTrigger* representa um gatilho que há de ser disparado através de um valor provido pelo sensor, seu disparo é controlado através de uma condicional lógica que é definida pelo campo *logicOperator*, que pode ser do tipo *GREATER_THAN* ou *LESS_THAN*, representando o operador lógico “maior que” e “menor que”, respectivamente. A condicional lógica será aplicada sob o valor lido do sensor e o campo *value.* Com o disparo do gatilho, o campo *type* é utilizado para estabelecer o tipo da notificação que será enviada, inicialmente apenas será aceitável o tipo de e-mail.

Para o caso dos sensores que transmitem mais de um valor, o campo *dataSource* é responsável por identificar qual o valor deve ser utilizado para execução da comparação lógica e definir o tratamento do gatilho.

Além destes campos, também são acrescidos os campos *name* e *content*, que definem respectivamente o nome e conteúdo da notificação.

3.1.2.6 Actuator

O *Actuator* é a representação de uma porta de saída do *device* e que pode resultar em uma ação física realizada em um determinado ambiente. A porta em questão é definida pelo campo *port*, estabelecendo a porta em que o atuador está conectado ao dispositivo. Para a primeira implementação o firmware do microcontrolador só suportará a saída binária (ligado ou desligado), no entanto, o campo *type* já prevê o suporte futuro com a saída de um valor de tensão qualquer definido pelo usuário.

Assim como o *sensor,* também é fornecido um campo *name* com o mesmo propósito de identificação para o uso do usuário.

3.1.2.7 ActuatorTrigger

Assim como o *NotificationTrigger*, o *ActuatorTrigger* define um gatilho fundamentando-se em uma condicional lógica com dados provenientes de um sensor. Todavia, em caso de disparo do gatilho, o resultado será uma ação no atuador associado, definido pelo campo *action* podendo resultar na ativação ou desativação do atuador. Como supracitado, já é planejado a saída de um valor analógico, sendo assim o campo *action* também poderia ser definido com um valor de 0 a 4095, representando uma saída na escala de 0 a 3.3v, no entanto, não será suportado na primeira versão do firmware.

Deve-se destacar que não é possível realizar aninhamento de operadores dentro de um mesmo gatilho de atuação, propondo-se o desenvolvimento em trabalhos futuros. Todavia, é possível que um atuador possua múltiplos gatilhos de distintos sensores (fontes de informação).

3.1.3 *Frameworks* e tecnologias

Javascript

Segundo as definições por Flanagan (2011), JavaScript ou JS é uma linguagem interpretada, orientada a objetos com funções de primeira-classe e sendo mais conhecida como a principal linguagem para criação de scripts de páginas web, porém, também é muito usado em ambientes que independem de navegadores.

Ordinariamente, JavaScript é utilizado no lado do cliente (*client-side*) na web, usufruído para projetar e programar como páginas web se comportam na ocorrência de um evento (*event-driven programming model*).

Com o desenvolvimento da linguagem também se tornou factível utilizar JavaScript como linguagem no lado do servidor (*server-side*). Direcionada à tal propósito, um servidor web JavaScript expõe objetos de *host* que representam uma solicitação HTTP (*Hyper Text Transfer Protocol*) e objetos de resposta, que são então manipulados por um programa em JavaScript que gera a resposta que será devolvida dinamicamente ao usuário. (*Node.js*, 2022).

Em conjunto com o Javascript, também se introduz o TypeScript, que é um superconjunto sintático estrito de JavaScript que adiciona tipagem estática opcional à linguagem. Seu uso dá-se para melhorar a estruturação do projeto e organização em camadas para alinhamento à arquitetura limpa. (*Typescript*, 2022).

Node.js

Aduzindo de seus criadores: O Node.js é um ambiente de execução *run-time* de JavaScript back-end, multi-plataforma, *open-source* executado no *Chrome V8 Javascript engine* possibilitando a execução do código JavaScript fora de um navegador da web*. (Node.js foundation, 2014)*. A ferramenta estabeleceu-se segundo a pesquisa do Stack Overflow feita em 2021 como a sexta tecnologia mais utilizada entre desenvolvedores profissionais.

O Node.js destaca-se por seu ecossistema gigantesco com uma comunidade ativa e colaborativa. Além disso, estabeleceu-se com solidez dado a facilidade de

intercâmbio de contexto entre o *front-end* e *back-end*, trazendo mais produtividade e redução de custos para as companhias. (*KingHost*, 2022).

Express

É descrito por Brown (2019) como um *framework* relativamente pequeno que roda em um servidor Node.js buscando simplificar suas APIs e adicionar funcionalidades novas e úteis ao mesmo. O uso do Express se dá por sua flexibilidade, minimalismo e performance.

Prisma

Prisma é definido por seus criadores como um ORM (*Object–relational mapping*) de Node.js e TypeScript que pode ser usado para criar servidores GraphQL, APIs REST, microsserviços e mais. (“*Prisma Client*”, 2022). O Prisma facilita a interação com o banco de dados e fornece uma camada de abstração que torna o desenvolvimento mais ágil e objetivo. Além disso, possui dentro de sua caixa de ferramentas o gerenciamento de *migrations* do banco de dados e o suporte a tipagem com Typescript.

3.1.4 Banco de dados

MySQL

O MySQL (2022) destaca-se por ser o mais popular sistema *open-source* para gerenciamento de bancos de dados. Dentre seus principais pontos de ênfase podemos citar:

- **Flexibilidade e facilidade de uso:**

Projetado para atingir a mais vasta gama de cenários, ainda é possível caso seja necessário adaptar o código-fonte para atender as necessidades. Outrossim, por sua vasta dominância na indústria, sua instalação pode ser feita de forma simplória na maioria dos provedores de armazenamento na nuvem.

- **Alto desempenho:**

É possível utilizar um *array* de servidores “*cluster*” para escalar em função da demanda. Ademais, o MySQL consegue garantir um alto desempenho e velocidade mesmo em cenários de alta demanda.

- **Um Padrão da Indústria:**

O MySQL tem sido usado nas indústrias há décadas, tempo esse que construiu estabilidade e solidez para a tecnologia, além de fornecer uma vasta gama de documentação construída por desenvolvedores ao longo do tempo.

- **Segurança:**

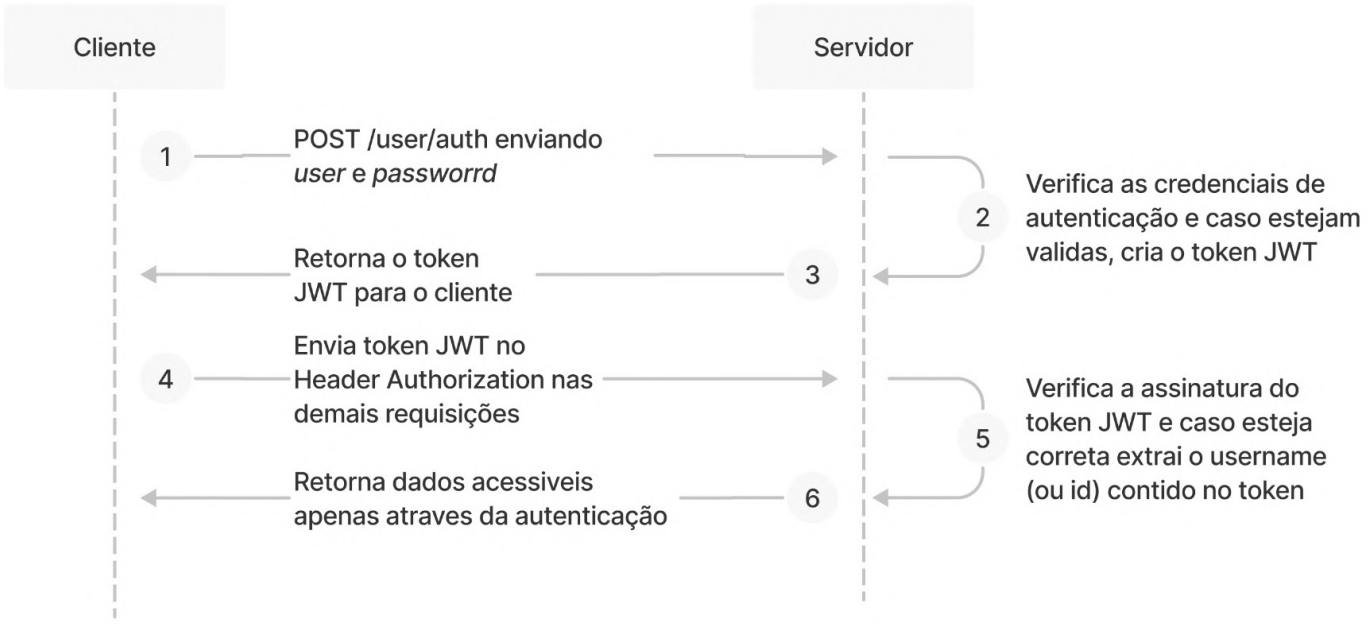
Dado por seu sistema de privilégios de acesso e gerenciamento de contas de usuários, o MySQL possui um alto nível de segurança já validado no mercado. Ademais, fornece um sistema de verificação auto hospedada e criptografia de senhas por padrão.

3.1.5 JWT

O *JSON Web Token* (JWT) é um padrão aberto (RFC 7519) que define uma maneira compacta e independente de transmitir informações com segurança entre as partes como um objeto JSON (*Javascript Object Notation*). (“JSON web tokens - jwt.Io”, 2022). Essas informações podem ser verificadas e confiáveis porque são assinadas digitalmente. Os JWTs podem ser assinados usando um segredo (com o algoritmo HMAC) ou um par de chaves pública / privada usando RSA ou ECDSA.

No contexto do proposto projeto, o JWT é utilizado para criação do *token* de autenticação do usuário. Esse *token* é utilizado em todas as pontas para validação da autenticidade e autoria da informação. Na figura 3.3 é compreendido o fluxo de autenticação que será desenvolvido em conjunto com o JWT.

Figura 3.3 - Fluxograma de autenticação utilizando o JWT em um serviço web



Fonte: De autoria própria, 2022.

3.1.6 *Bcrypt*

Desenvolvido em 1999 por Niels Provos e David Mazières, o Bcrypt (2022) tem como objetivo esconder conteúdos criados em forma de texto “puro” em dados criptografados partindo do uso de um algoritmo *hash*.

O Bcrypt (2022) baseia-se no *Blowfish* que é uma cifra de bloco de chave simétrica, projetada em 1993 por Bruce Schneier. Além disso, é apresentada uma segurança maior em relação a maioria dos outros métodos criptográficos dado ao uso da implementação da variável “custo”, que se estabelece proporcional à capacidade de processamento necessária para criptografar a senha.

A ferramenta é conhecida por ser uma forma segura para armazenar senhas no banco de dados e por possuir bibliotecas para boa parte das linguagens de programação do mercado, facilitando assim sua introdução em qualquer aplicação. (*Codahale*, 2010).

Seu uso no projeto proposto, se dá na encriptação das senhas dos usuários salvos no banco de dados.

3.2 FRONT-END

3.2.1 React

O React (2022) é uma biblioteca *open-source* JavaScript, criada pelo Fa-

cebook®, para criação de interfaces de usuário. Segundo sua documentação, a biblioteca estrutura-se nos seguintes aspectos:

- **Declarativo:**
  As *views* declarativas tornam o código mais previsível, mais simples de entender e mais fácil de depurar.

- **Baseado em componentes:**
  A criação de componentes encapsulados que gerenciam seu próprio estado e que podem ser usados em cascata possibilita a produção de interfaces de usuário complexas. Como a lógica do componente é escrita em JavaScript em vez de modelos, é possível facilmente passar dados avançados e manter o estado fora do DOM (*Document Object Model*).

- **Escreva em qualquer lugar:**
  Não é feita suposições sobre o resto da pilha de tecnologia utilizada, possibilitando o desenvolvimento de novos recursos no React sem reescrever o código existente. O React também pode renderizar no servidor usando Node.js e em aplicativos móveis usando React Native.

3.3 HARDWARE

3.3.1 ESP32

Projetado pela renomada empresa de desenvolvimento de tecnologia Espressif Systems, o microcontrolador ESP32 foi lançado em 2016 e é considerado um dos controladores mais poderosos e notórios do mercado, com ótimos recursos como velocidade de processamento, acessibilidade e conectividade. Este último é demonstrado principalmente pela inteligibilidade de sua conexão ao Wi-Fi.

O ESP32 (2022) é constituído por um poderoso processador, que pode ser *single-core* ou *dual-core* de 32 *bits* (com dois núcleos de processamento físico) e pode operar em frequências de *clock* de até 240 MHz. Outrossim, apresenta um regulador de tensão projetado para nivelamento da tensão de entrada para 3,3V, utilizado para sua alimentação interna. Para armazenamento volátil são disponibilizados 520 Kb de memória RAM. São acrescidos a isso para armazenamento não volátil 448 KB de memória ROM e 4MB de memória flash integrada. Para comunicação são dispostos uma interface Wi-Fi de 2,4 GHz e Bluetooth BLE 4.2 que utilizam de uma antena

embutida e blindagem EMI (*eletromagnetic inference*). Além de sua possibilidade de conectividade embutida, seu desempenho é superior em relação aos microcontroladores Arduino já estabelecidos. Na Tabela 3.1 são listadas as especificações do microcontrolador:

Tabela 3.1: Lista de especificações do ESP-32

Especificações técnicas ESP32	**Detalhes**
Tensão	3.3 V
Memória RAM	520 Kb
Processador	single/dual-core 32 bit
Frequência de clock	160 a 240 MHz
GPIOs	34
Conversores analógico/digital	7

Fonte: Elaboração própria, 2022.

A programação simples do ESP32 (2022) é factível através de um ambiente de desenvolvimento integrado com linguagens como MicroPython, C / C++, Lua, dentre outras. Além da conexão sem fio integrada, a programação remota também é possível, utilizando por exemplo, Wi-Fi.

Tomando os fatos acima como referência, fica evidente a viabilidade do uso do ESP32 como microcontrolador para projetos de baixo custo e alta conectividade aos componentes de sistemas de automação no geral.

No entanto, por tratar-se de uma proposta de arquitetura deve-se destacar que o ESP-32 será apenas uma ferramenta comprobatória do funcionamento da ideia proposta.

3.3.2 MicroPython

O MicroPython (2022) é uma implementação limpa e otimizada da linguagem de programação Python. O seu desenvolvimento fundamentou-se em ser otimizado para execução em microcontroladores e em ambientes restritos e de recursos limitados. Em seu conjunto de ferramentas, possui um pequeno subconjunto da biblioteca padrão do Python, além de também incluir por padrão módulos que dão ao

programador acesso a hardware de baixo nível e as interfaces de comunicação do microcontrolador.

MicroPython consiste em um conjunto composto dos seguintes ferramentais:

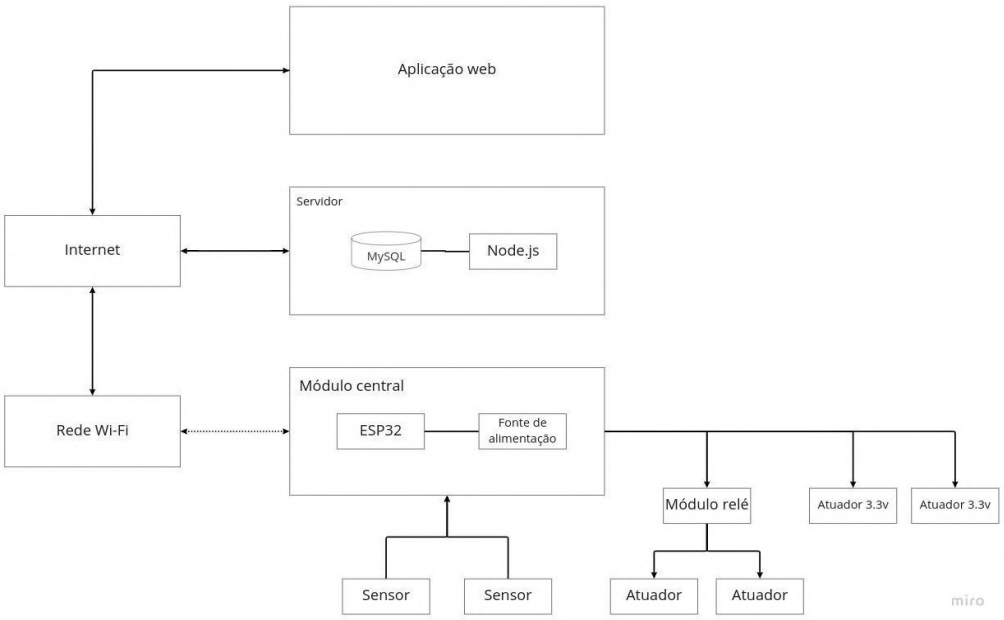
- Compilador Python para bytecode
- Interpretador de tempo de execução desse bytecode
- Prompt interativo (o REPL) para executar comandos suportados.

O uso do MicroPython se dá pela agilidade de desenvolvimento e facilidade de construção de protótipos. No entanto, para casos de uso que requerem muita performance uma migração para C ou C++ poderia ser aplicada. Além disso, a construção da API desassocia completamente a tecnologia utilizada para realizar a sua chamada, assim como o microcontrolador ou dispositivo que a usará.

3.3.3 Conexão das partes

A organização arquitetural de relação entre as partes apresentadas é disposta na figura abaixo:

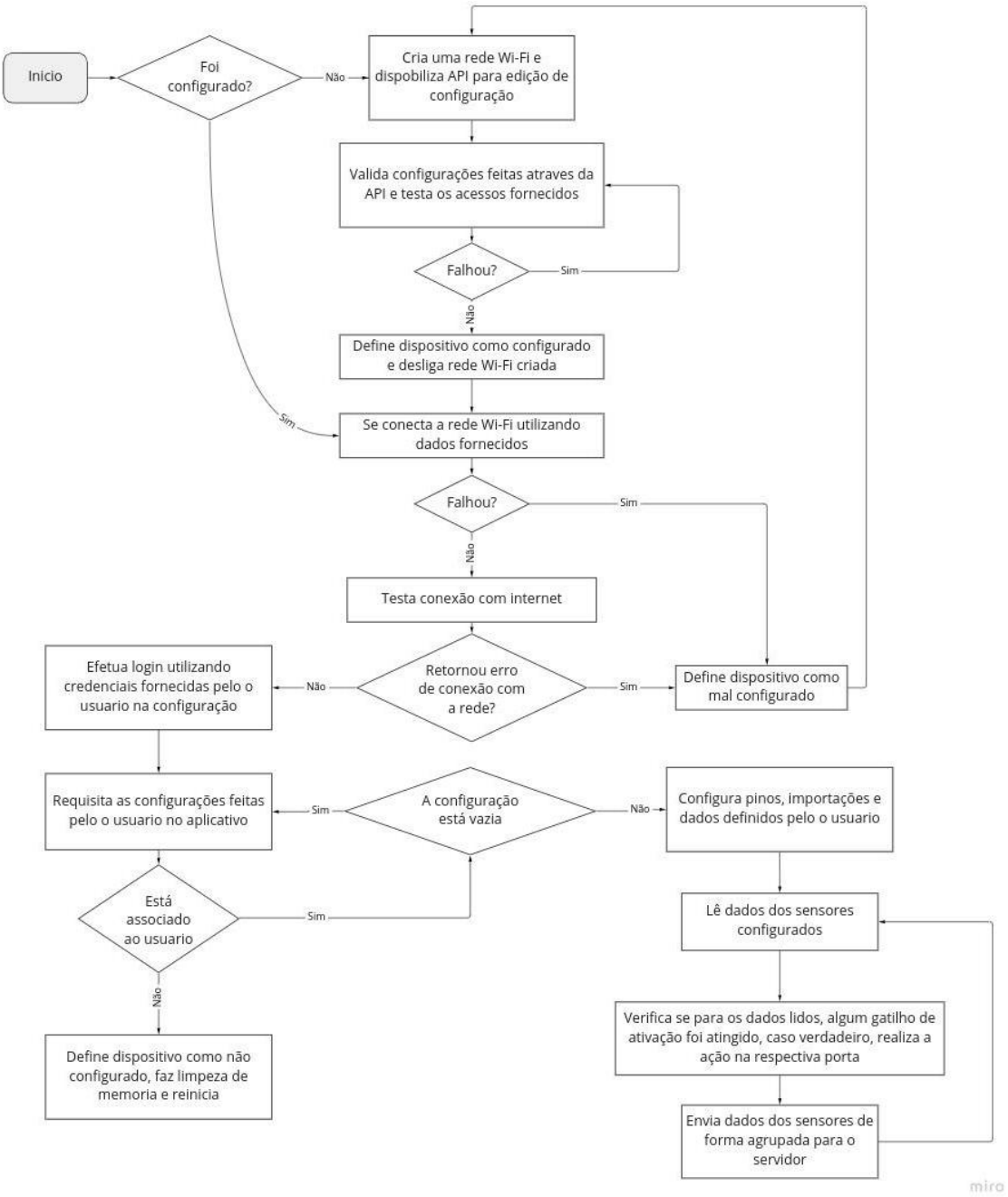
Figura 3.4 - Diagrama de relação das partes



Fonte: Elaboração própria, 2022.

Pode-se descrever o comportamento da aplicação utilizada no módulo central através do seguinte fluxograma:

Figura 3.5 - Fluxograma do firmware do dispositivo



Fonte: Elaboração própria, 2022.

3.3.4 Módulos estruturais do firmware

Para estruturação do projeto arquitetônico do firmware foi proposto a separação em módulos para organizar e distribuir responsabilidades de maneira replicável e escalável. Estruturam-se os módulos supracitados em:

- **Módulo de armazenamento fixo:**
    - Possibilita o armazenamento de informações na memória persistente do dispositivo.
    - Estrutura-se em um armazenamento chave e valor, também descrito como armazenamento KVS (*key value store*).
- **Módulo de autenticação:**

Como descrito previamente, toda a aplicação requer autenticação para validação da autorização. Esse módulo encarrega-se de chamar a API de autenticação e obter um token que será disponibilizado para uso futuro.

- **Módulo de rede:**

Dado o requerimento da conexão com a internet para comunicação das partes, tal módulo realiza o gerenciamento de rede, tendo como principais funcionalidades criar uma rede Wi-Fi para configuração do dispositivo, conectar-se a uma outra rede *wireless* para obter acesso à internet e testar a conexão com a internet utilizando a rede provida.

- **Módulo servidor:**

Na etapa de configuração inicial, o usuário se conectará a rede Wi-Fi provida pelo microcontrolador, para fornecer os dados de rede que serão utilizados para estabelecer uma conexão com a rede de internet do usuário. A aplicação web realizará chamadas a esse servidor provido por tal modulo, que por sua vez irá salvar as preferências do usuário utilizando o armazenamento fixo.

- **Módulo gerente:**

Esse módulo gerencia o fluxo da aplicação, chamando outros módulos e definindo o comportamento estabelecido no fluxograma da figura anterior.

- **Módulo de carregamento:**

Tal módulo estabelece a requisição das preferências do usuário realizadas na aplicação web, como por exemplo, sensores conectados, atuadores e gatilhos de atuadores. Nessa etapa também são carregadas as dependências e interfaces necessárias para o funcionamento dos sensores demandados pelo cliente. Em seu retorno são fornecidas interfaces para cada uma das instâncias dos sensores e atuadores. Tais instâncias possibilitam a leitura de dados do sensor e também a execução de gatilhos em um atuador. Após seu processamento, mesmo na ausência

de internet, todos os gatilhos de atuação são mantidos em memória e continuam a funcionar.

- **Módulo de contato:**

O módulo de contato realiza o envio dos dados lidos dos sensores agrupados e de forma assíncrona ao loop de processamento principal, haja vista que tal envio de informações (uma requisição de rede) não poderia ser bloqueante ao fluxo. Para resolução de tal problema deu-se o uso de *threads*.

3.4 FERRAMENTAS AUXILIARES

3.4.1 Git

Atlassian (2022) descreve o Git como um sistema de controle de versão gratuito e de código aberto originalmente criado por Linus Torvalds em 2005. Ao contrário de sistemas de controle centralizados mais antigos, como SVN e CVS, o Git é distribuído, ou seja, cada desenvolvedor tem um histórico completo do repositório de código local. Isso torna os clones de repositório iniciais mais lentos, mas as operações subsequentes, como *commits*, *blame*, *diffs*, *merges* e *journals*, são muito mais rápidas.

O Git também oferece excelente suporte para *branching*, *merging* e reescrita do histórico de um repositório, o que permite a criação de muitas ferramentas e fluxos de trabalho. As *pull requests* são um exemplo de uma ferramenta tão popular, que permite que as equipes colaborem nas *branches* do Git e revisem efetivamente o código de toda a equipe. Atualmente, é o sistema de controle de versão mais utilizado no mundo e é considerado o padrão moderno para desenvolvimento de software.

O seu funcionamento segue a seguinte ordem de etapas:

1. É criado um repositório (projeto) com uma ferramenta de hospedagem de *Git* (como Github)
2. Copie (ou *clone*) o repositório para a sua máquina local
3. Adicione um arquivo ao seu repositório local e salve as alterações (faça um *commit*)
4. Coloque suas alterações (*push*) na ramificação principal do repositório
5. Faça uma alteração no seu arquivo com uma ferramenta de hospedagem *Git* e faça *commit*.
6. Puxe (*pull*) as alterações para sua máquina local
7. Crie uma ramificação (*branch*), faça uma alteração, faça *commit* da alteração

8. Abra uma solicitação *pull* (*pull request*), propondo mudanças na branch principal
9. Faça a mesclagem (*merge*) da sua ramificação com a ramificação principal

3.4.2 GitHub

O Git possui algumas ferramentas de hospedagem para armazenar e gerenciar repositórios e *branches*. O mais popular deles é o Github, atualmente mantido pela Microsoft.

O Github permite utilizar o Git para gerenciar repositórios hospedados nele, além de fornecer ferramentas para colaboração no desenvolvimento e manutenção de software, com opções de CI / CD (*continuous integration* / *continuous deployment*), *deploy* automatizado. Além disso, é facilitado a colaboração em projetos de código aberto através da opção de criar *forks* baseados em projetos existentes.

Os repositórios utilizados durante o desenvolvimento do projeto podem ser consultados nas URLs abaixo:

- sention-app: Contempla a aplicação React do *front-end*
- sention-api: API fornecedora dos serviços
- sention-device: Contém o código do *firmware* utilizado no dispositivo

3.4.3 Docker

O Docker é descrito pela IBM (2022) como uma plataforma de conteinerização de código aberto. A sua principal característica dá-se pela possibilidade de empacotar seus aplicativos em componentes executáveis padronizados e em contêiner, que combinam o código do aplicativo com as bibliotecas do sistema operacional e as dependências necessárias para executar esse código em qualquer ambiente. Os contêineres simplificam a entrega de aplicativos distribuídos e estão se tornando mais populares à medida que as empresas migram para o desenvolvimento centrado na nuvem.

3.4.4 Figma

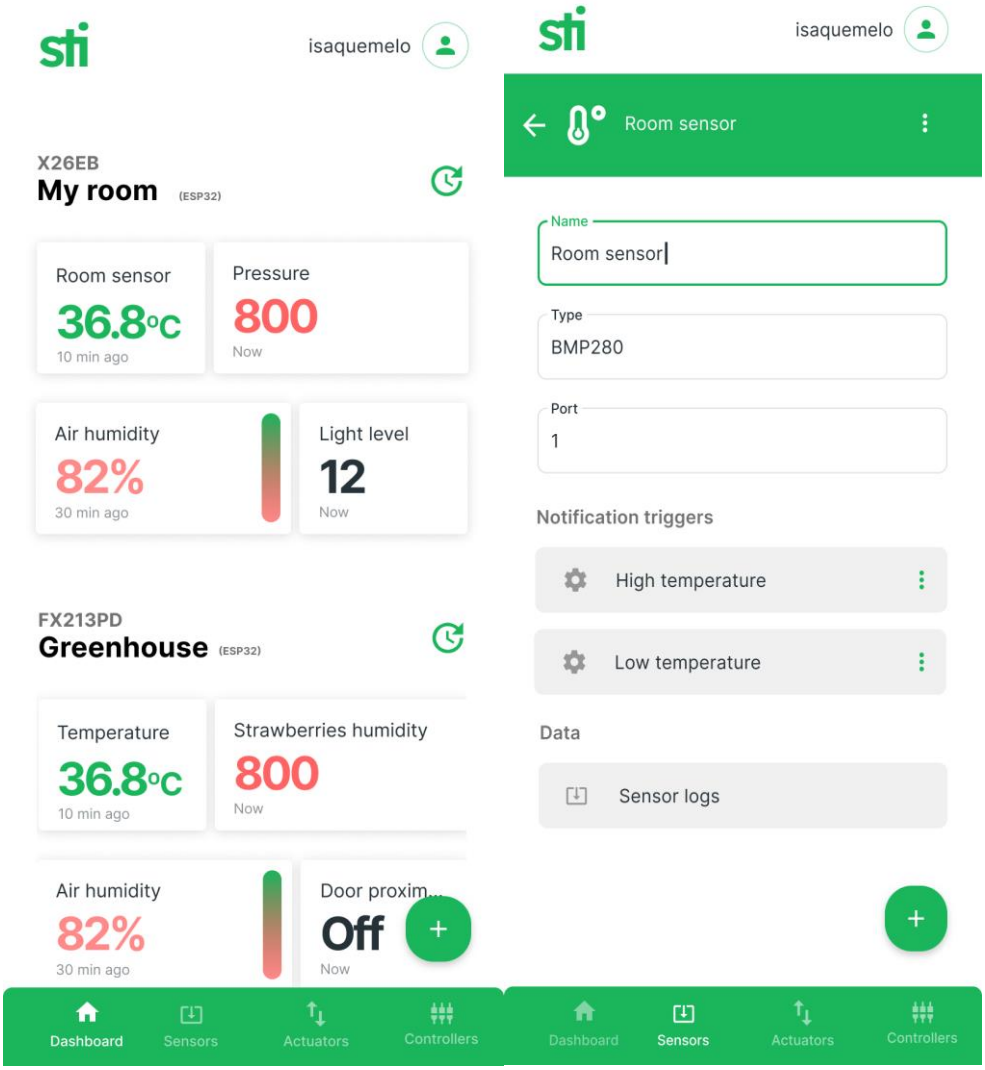
O Figma (2022) é um editor de gráficos vetoriais, uma ferramenta online colaborativa de prototipagem e uma plataforma de design para equipes, baseada principalmente na web. A função do Figma é dar apoio aos times em criar, compartilhar, testar e enviar designs melhores do início ao fim, consolidando

ferramentas, simplificando fluxos de trabalho ou colaborando entre equipes, tornando o processo de design mais rápido e mais eficiente. Mantendo sempre os integrantes cientes de como deverá ser cada parte e comportamento da aplicação.

O Figma foi utilizado na criação dos *layouts* da aplicação e na criação dos *wireframes* que possibilitaram a visualização do funcionamento das partes e das funcionalidades. O projeto desenvolvido pode ser visualizado aqui.

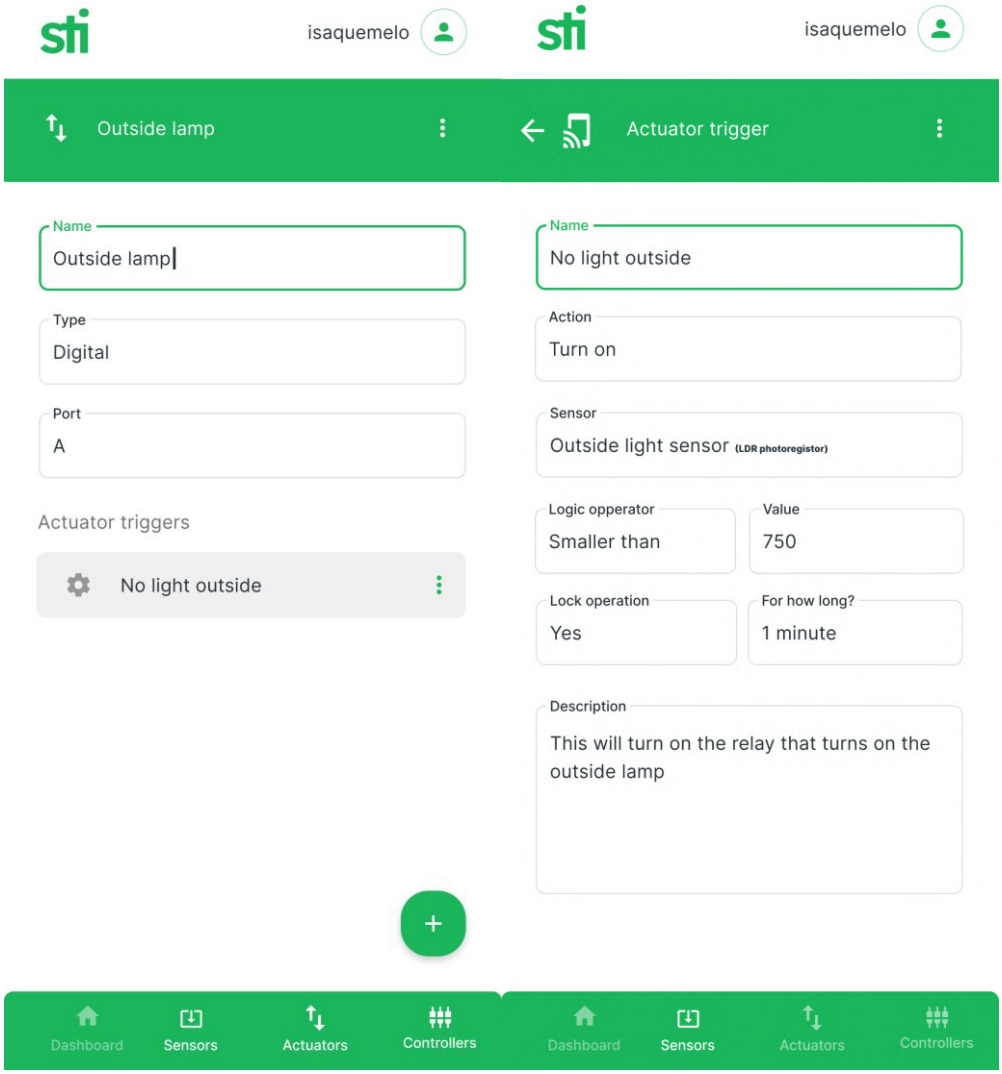
Listam-se abaixo algumas telas protótipos de layout desenvolvidas utilizando o Figma.

Figura 3.6 - Tela inicial da aplicação proposta no layout, nomeada como “Dashboard” e a direita a tela de edição de um sensor



Fonte: Wireframe do Sention desenvolvido no Figma, 2022.

Figura 3.7 - Tela de edição de atuador e tela de edição de um gatilho de atuação propostas no layout



Fonte: Wireframe do Sention desenvolvido no Figma, 2022.

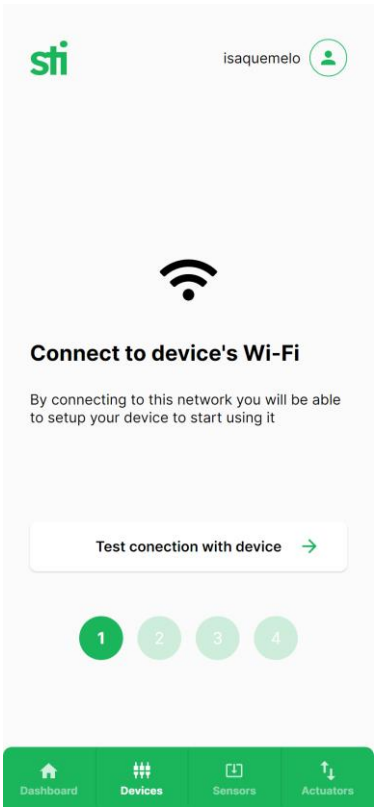
CAPÍTULO 4

4. SENTION

Neste capítulo, será elucidado o sistema proposto, o qual é constituído por três segmentos: aplicação web, API fornecedora de serviços e firmware do dispositivo. Tal sistema foi desenvolvido utilizando as ferramentas explicitadas no capítulo anterior. Nos tópicos seguintes, serão detalhadas as secções do Sention.

4.1 CONEXÃO AO WI-FI DO DISPOSITIVO

Figura 4.1 - Tela de conexão com o Wi-Fi do dispositivo

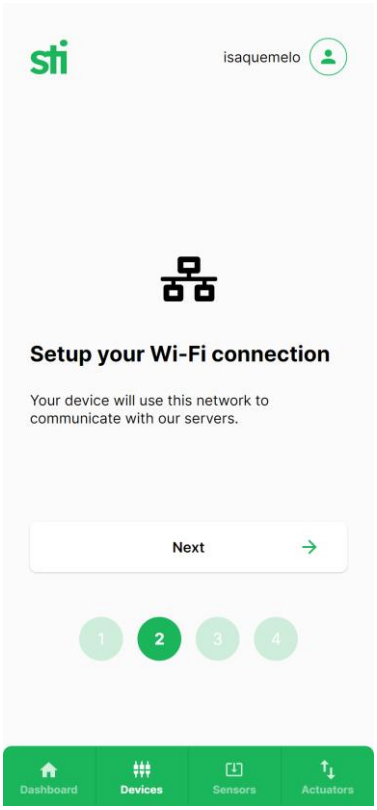


Fonte: Sention, 2022.

Nesta secção ocorre a conexão do aparelho do usuário com o seu dispositivo. O cliente utiliza o seu computador ou *smartphone*, ao qual utilizou para acessar ao site do Sention, para conectar-se ao Wi-Fi do seu respectivo dispositivo, no caso o ESP32. Após efetuar a conexão ao Wi-Fi que foi inicializado pelo ESP32, o usuário pode clicar no botão "*Test connection with device*", caso tenha sido efetuada corretamente, o usuário segue para a secção de pré-configuração do seu Wi-Fi pessoal.

4.2 PRÉ-CONFIGURAÇÃO DO WI-FI PESSOAL

Figura 4.2 - Tela de pré-configuração do Wi-Fi do usuário

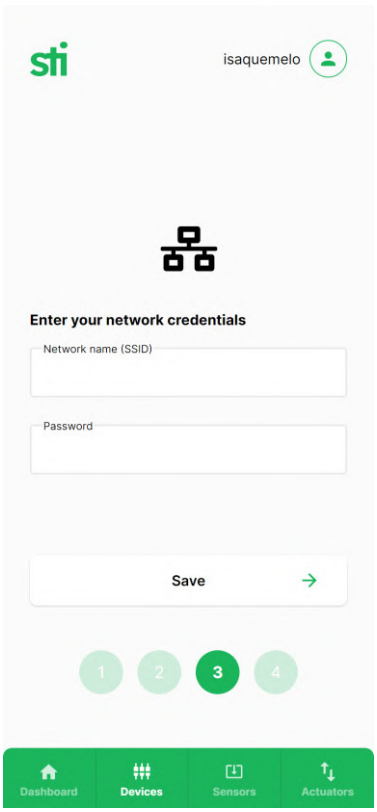


Fonte: Sention, 2022.

A secção de pré-configuração do Wi-Fi pessoal é uma etapa elucidativa, que tem o intuito de informar ao usuário que o Wi-Fi que será inserido na etapa seguinte será utilizado para realizar a comunicação com o servidor do Sention.

4.3 CONFIGURAÇÃO DO WI-FI PESSOAL DO USUÁRIO

Figura 4.3 - Tela de configuração do Wi-Fi do usuário



Fonte: Sention, 2022.

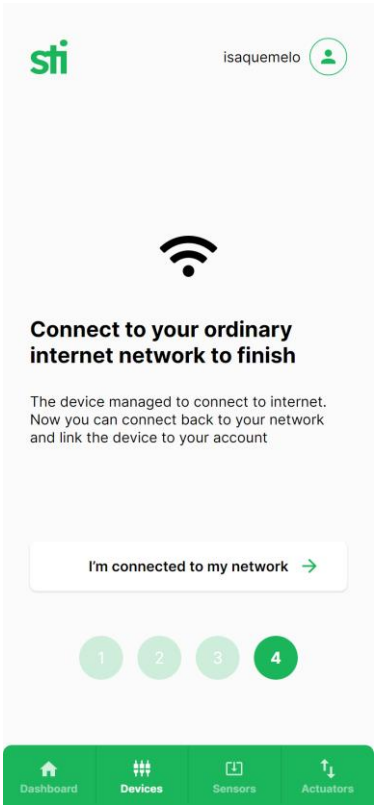
Nesta etapa o usuário insere o SSID e a senha do seu Wi-Fi pessoal. Ao clicar no botão "*Save*", a aplicação se encarrega de enviar o *username* do usuário, o SSID do Wi-Fi e suas respectivas senhas em uma requisição do tipo *POST* para API que foi levantada pelo dispositivo ESP32. Caso a conexão seja efetivada a partir das credenciais fornecidas pelo usuário, a API interna salva os dados recebidos na memória utilizando o módulo KVS do dispositivo.

Em decorrência do passo anterior, o dispositivo envia uma requisição do tipo *POST* para a API fornecedora de serviços com o intuito de criar o *device* no banco de dados, caso seja efetivada, é retornado o *DEVICE_ID* e o *ACCESS_CODE* do *device*, que são salvos na memória KVS do ESP32 em questão.

Posteriormente, uma nova requisição do tipo *POST* é enviada para realizar a associação do *device* ao usuário, consecutivamente o *USER_ID* é salvo na memória KVS, juntamente com a variável *CONFIGURED*, que é definida como "*True*", finalizando a etapa de configuração do dispositivo descrita no fluxograma do *firmware*.

4.4 FINALIZAÇÃO DA CONFIGURAÇÃO

Figura 4.4 - Tela de finalização do processo de configuração

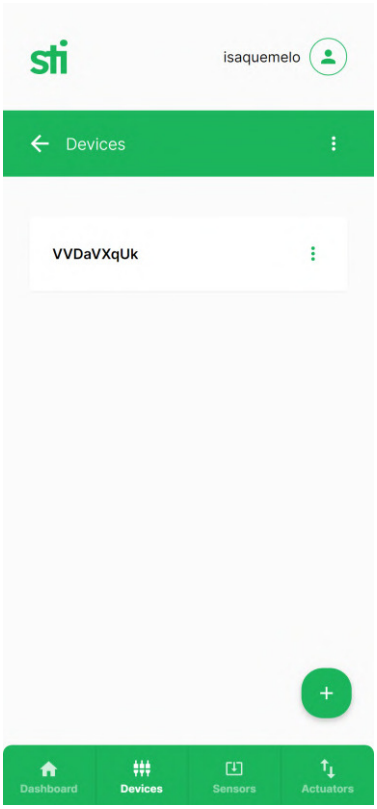


Fonte: Sention, 2022.

Neste momento o usuário pode voltar a se conectar com a sua rede Wi-Fi padrão e ao clicar no botão "*I'm connected to my network*" ocorre o redirecionamento para a secção de visualização dos *devices* associados à conta do usuário.

4.5 *DEVICES* ASSOCIADOS

Figura 4.5 - Tela de *devices* associados à conta do usuário

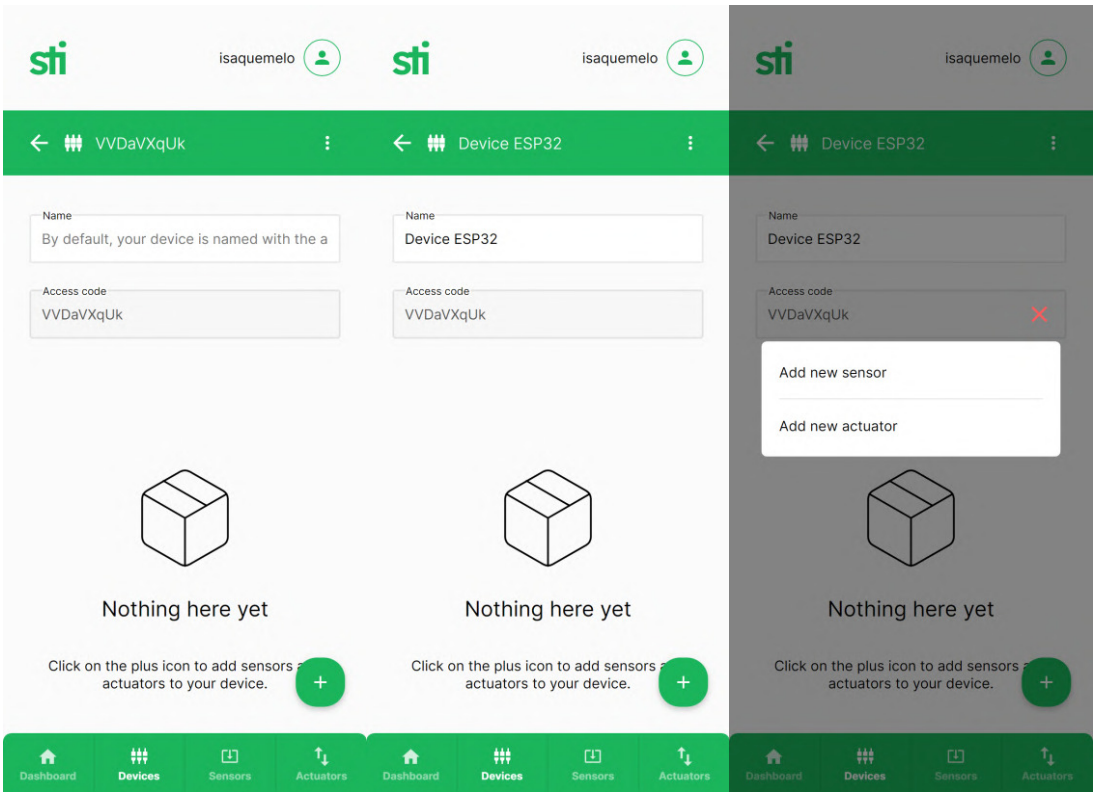


Fonte: Sention, 2022.

Na secção “*Devices”*, o usuário visualiza os dispositivos que estão associados à sua conta. Possuindo também a capacidade de associar um novo *device* ao clicar no botão da extremidade direita inferior da tela apresentada na figura 4.5. Além disso, pode-se também deletar um *device*, ou selecioná-lo para edição. Este último redireciona o usuário para a tela de visualização / edição do *device* escolhido, representada na figura 4.7.

4.6 *DEVICE*

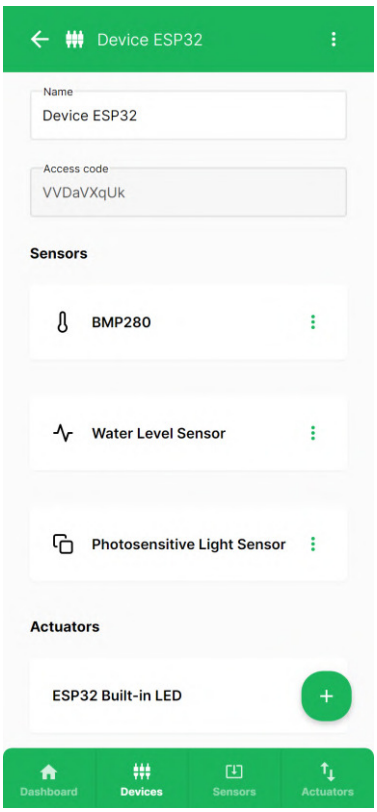
Figura 4.6 - Tela de visualização / edição de um *device*



Fonte: Sention, 2022.

Na secção "*device*", o usuário tem a capacidade de deletar o dispositivo selecionado, editar seu nome, ou adicionar sensores e atuadores ao clicar no botão da extremidade direita inferior da tela representada na figura 4.6. Este último, redireciona o usuário para a secção de criação de um sensor ou para a de criação de um atuador, telas representadas nas figuras 4.9 e 4.10 respectivamente.

Figura 4.7 - Tela de visualização / edição de um *device* com sensores e atuadores adicionados

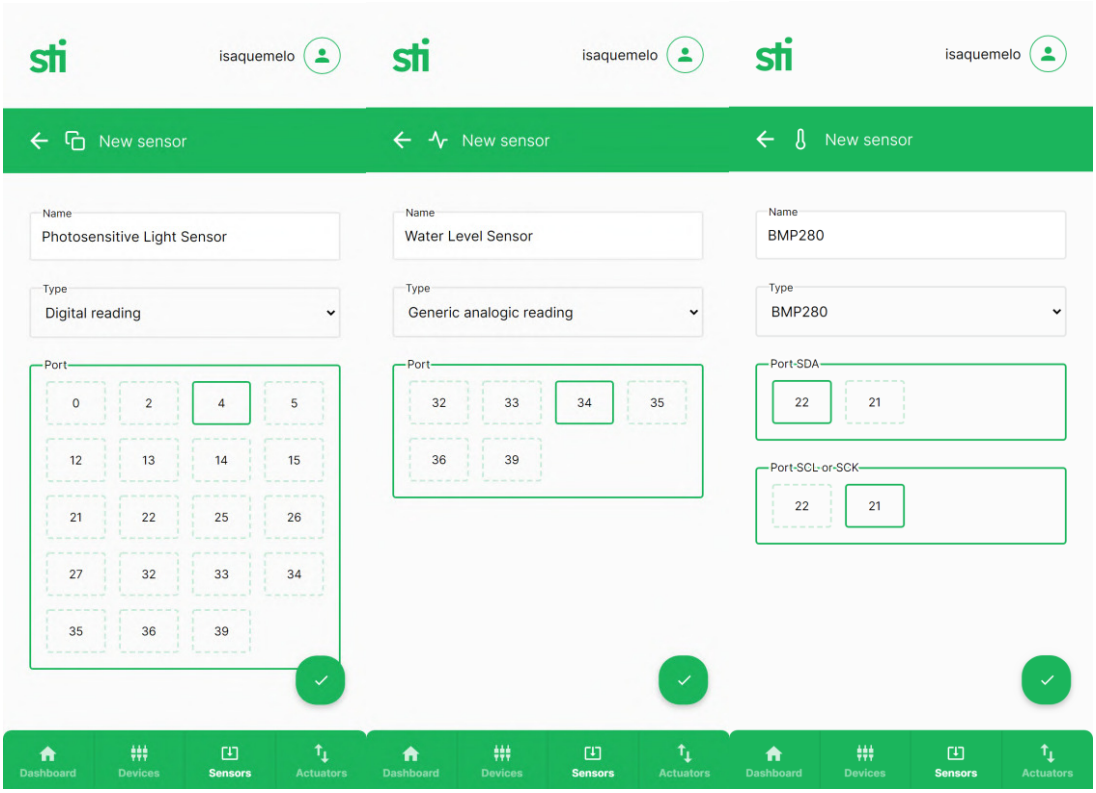


Fonte: Sention, 2022.

Mantendo-se ainda na secção “*device*”, após a adição de sensores e atuadores, o usuário consegue realizar a visualização, remoção e edição de sensores e atuadores. Outrossim, esta última opção redireciona o usuário para a secção de visualização / edição de um *sensor* ou *actuator* específico, exibido nas figuras 4.9 e 4.11 respectivamente.

4.7 *SENSOR*

Figura 4.8 - Tela de criação de um *sensor*

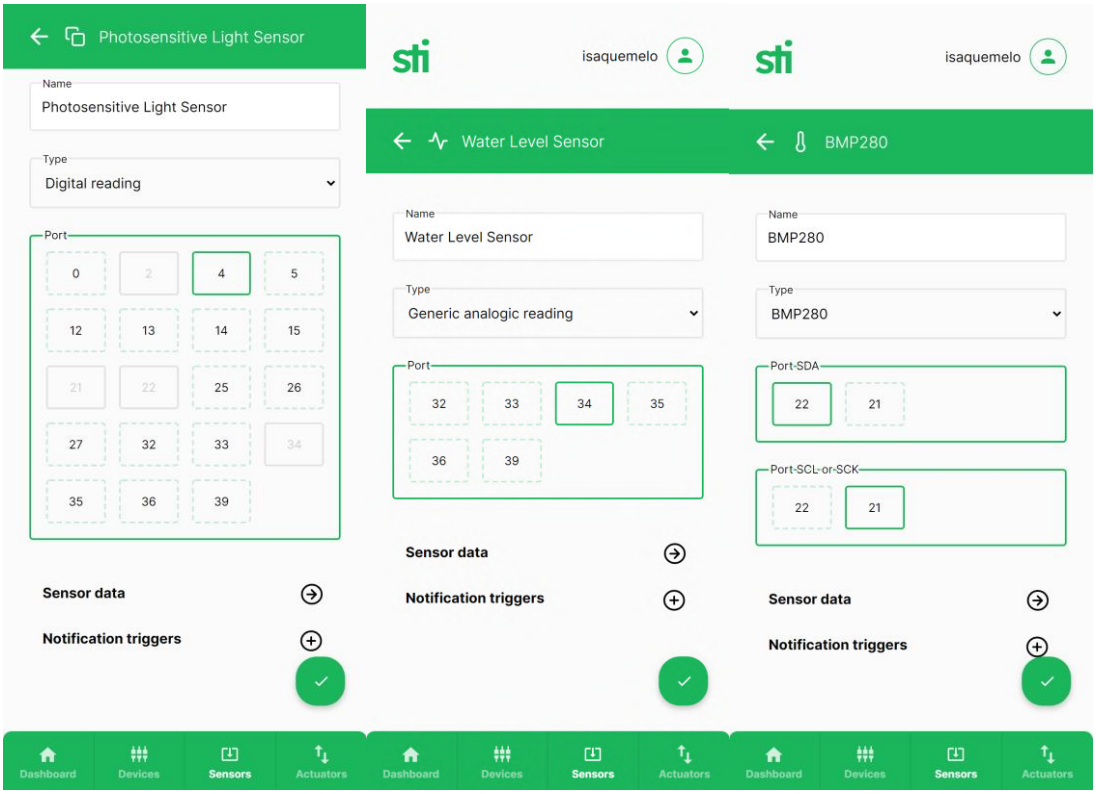


Fonte: Sention, 2022.

Na secção “*Sensor*”, o usuário tem a capacidade de inserir a identificação do sensor no campo “*Name*” e selecionar o tipo do sensor no campo “*Type*”. Este último possui 3 opções: “*Digital reading*” para sensores que realizam a leitura de dados digitais, “*Generic analogic reading*” para sensores que efetuam leituras analogicas, e BMP280 para o sensor BMP280 que faz uso da comunicação I2C, utilizando duas portas, SCK e SDA. De acordo com o tipo do sensor escolhido, a aplicação apresenta as possíveis portas, as quais o usuário pode escolher para realizar a conexão do sensor com o dispositivo ESP32 no campo “*Port*”.

Ademais, as portas que já foram selecionadas para conexões anteriores são indisponibilizadas.

Figura 4.9 - Tela de visualização / edição de um *sensor*

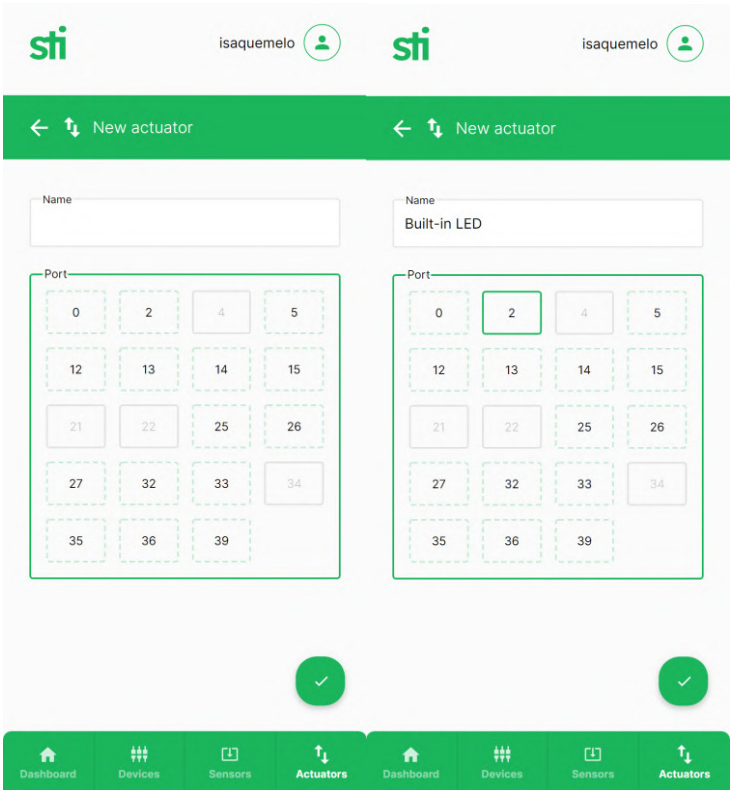


Fonte: Sention, 2022.

Após a adição de um sensor, o usuário detém a possibilidade de visualizá-lo e / ou editá-lo. Ainda nesta secção, na divisão "*Sensor data"*, ao clicar no botão ao qual possui o ícone que simboliza uma seta para a direita, na extremidade inferior da tela demonstrada na figura 4.9, o usuário é redirecionado para a visualização dos dados do sensor, representada na figura 4.14. Já na divisão "*Notification triggers*", ao clicar no botão ao qual simboliza o ícone de sinal de adição, o usuário é direcionado para a secção de criação de um gatilho de notificação, representada na figura 4.12.

4.8 *ACTUATOR*

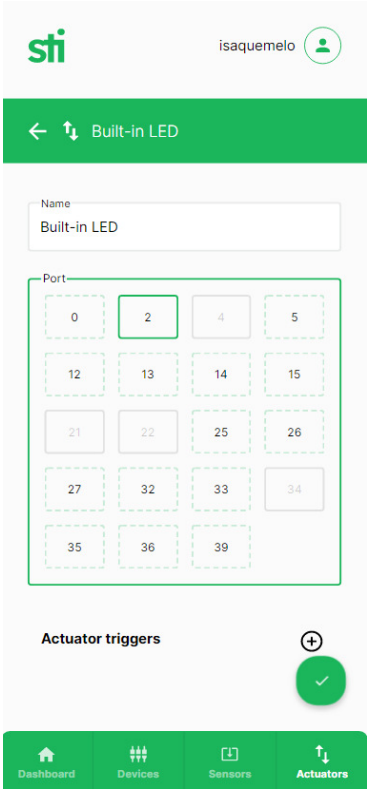
Figura 4.10 - Tela de criação de um *actuator*



Fonte: Sention, 2022.

Na secção “*Actuator”*, o usuário tem a capacidade de inserir a identificação do atuador no campo “*Name”* e selecionar a porta escolhida para realizar a conexão do atuador com o dispositivo ESP32 no campo “*Port”*. As portas que já foram selecionadas para conexões anteriores são indisponibilizadas.

Figura 4.11 - Tela de visualização / edição de um *actuator*

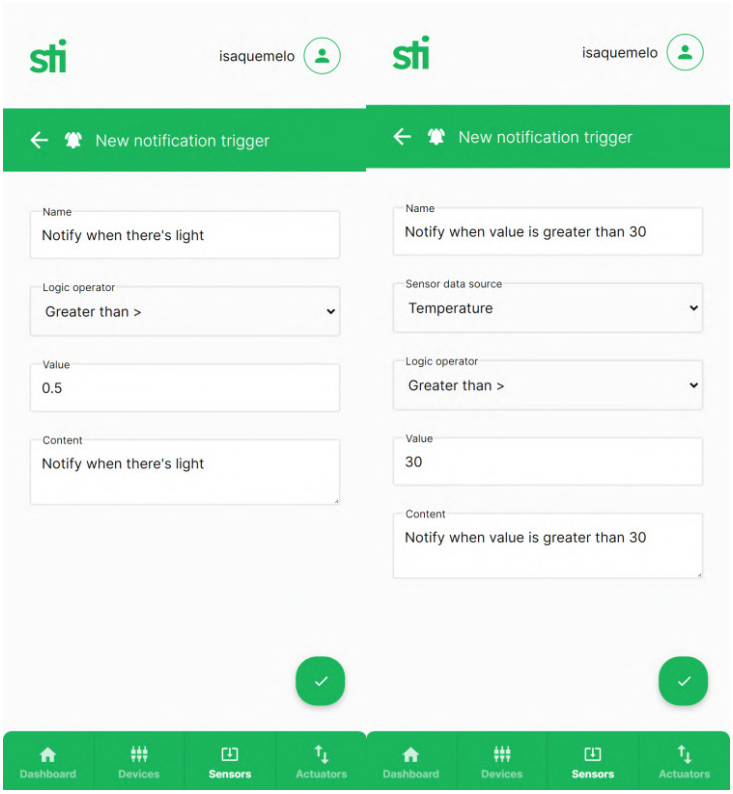


Fonte: Sention, 2022.

Mantendo-se na secção “*Actuator*”, após adicioná-lo, o usuário tem a capacidade de adicionar um *actuator trigger* ao atuador em questão, clicando no botão que retrata o sinal de adição. O último passo supracitado redireciona o usuário à tela de criação de um gatilho de atuação, a qual pode ser evidenciado na figura 4.13.

4.9 *NOTIFICATION TRIGGER*

Figura 4.12 - Tela de criação / visualização / edição de um *notification trigger*



Fonte: Sention, 2022.

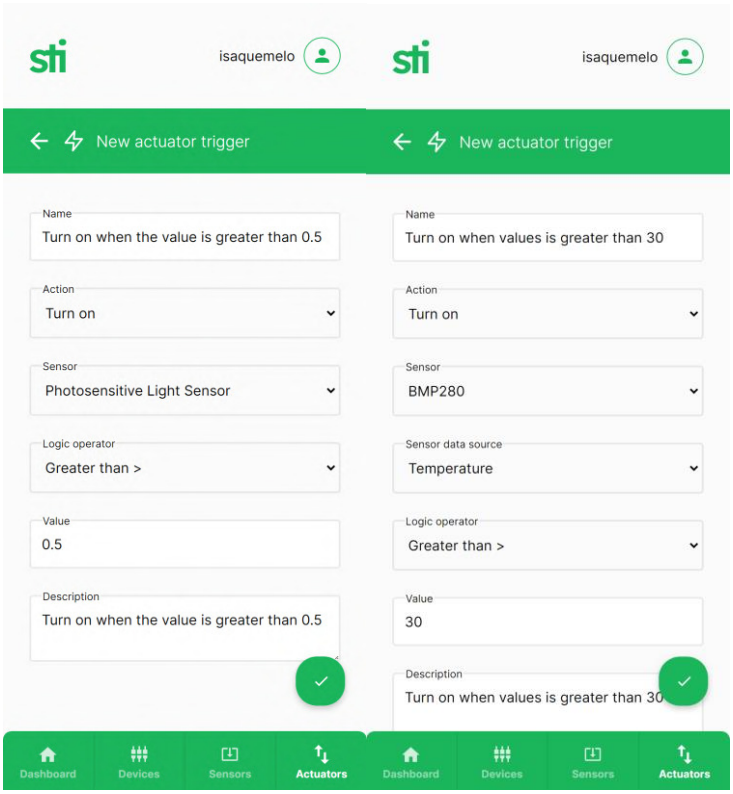
Na secção “*Notification trigger”*, os campos “*Name*” e “*Content”*, são responsáveis pela identificação do gatilho de notificação e armazenamento de uma descrição adicional. O campo “*Logic operator”* é utilizado para definir o operador lógico que será aplicado sob o valor lido do sensor, nele o usuário escolhe entre “*Greater than* >” e “*Less than* <” representando “maior que” e “menor que”, nesta ordem. Para sensores de dois estados (ligado e desligado), pode-se criar um gatilho com o limiar em 0,5. No campo “*Value*" o usuário define o valor que juntamente com o operador lógico será utilizado para determinar se o gatilho deve ou não ser disparado.

Para o caso dos sensores que transmitem mais de um valor, como o BMP280 por exemplo, o campo “*Sensor data source”*, possui as opções “*Temperature”*, “*Air pressure”* e “*Altitude”*, sendo este responsável por identificar qual o valor deve ser utilizado para execução da comparação lógica e definir o tratamento do gatilho.

Caso o gatilho de notificação seja disparado é enviado um e-mail para o endereço do usuário, sendo o valor do campo “*Name”* referente ao título e o conteúdo do campo “*Content”* relativo ao corpo da mensagem.

4.10 *ACTUATOR TRIGGER*

Figura 4.13 - Tela de criação / visualização / edição de um *actuator trigger*



Fonte: Sention, 2022.

Assim como no *notification trigger*, na secção *actuator trigger* o campo "*Name*" é reservado para identificação, enquanto o campo "*Description"* tem a função de armazenar uma descrição adicional. No campo "*Action"*, o usuário pode escolher entre "*Turn on*" ou "*Turn off*", ou seja, configurar o gatilho de atuação para ligar ou desligar o atuador em questão.

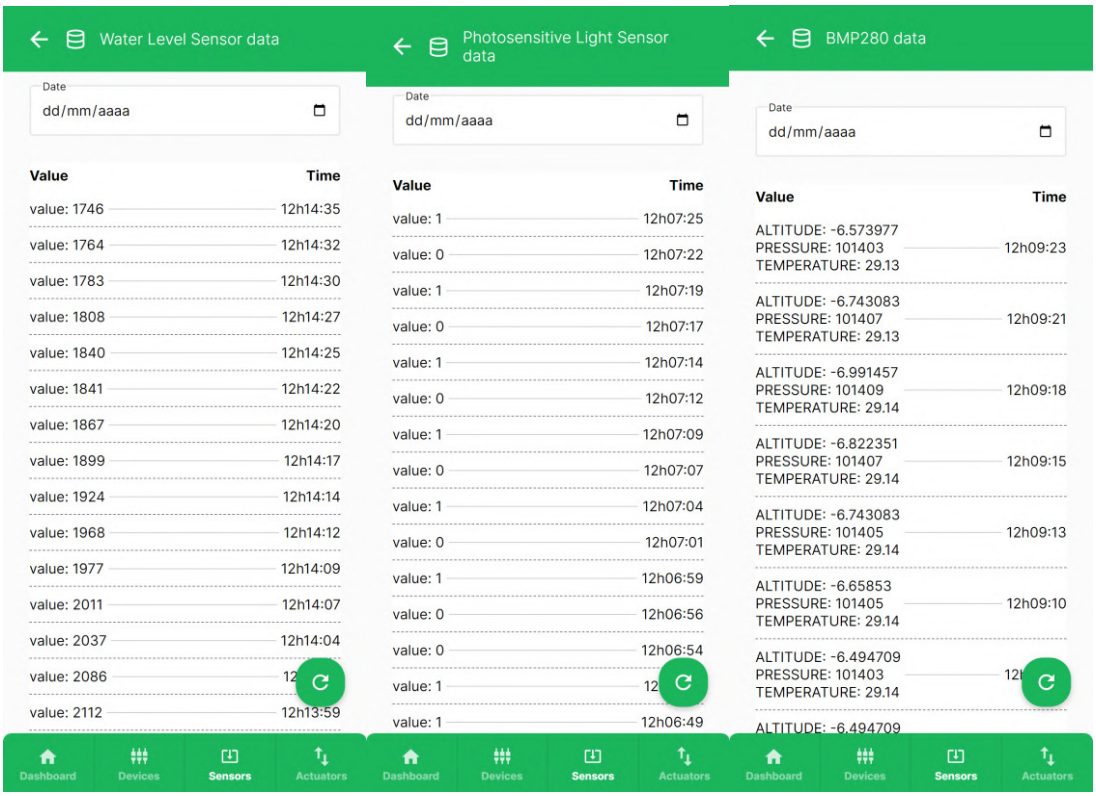
No campo "*Sensor"*, o cliente tem a capacidade de selecionar um, dentre os sensores cadastrados, ao qual a partir dos dados deste será definida a ativação do gatilho. Caso seja escolhido um sensor que realiza o envio de mais de um valor, como é o caso do BMP280, o campo "*Sensor data source"* será disponibilizado, contendo as opções "*Temperature"*, "*Air Pressure"* e "*Altitude"*, para que o usuário selecione qual o valor deve ser utilizado para execução da comparação lógica e para a definição do tratamento do gatilho.

Já quanto ao campo "*Logic operator",* ele é utilizado para definir o operador lógico que será aplicado sob o valor lido do sensor, nele o usuário escolhe entre "*Greater than >*" e "*Less than <*" representando "maior que" e "menor que", nesta ordem. E no

campo “*Value”* o usuário define o valor que juntamente com o operador lógico será utilizado para determinar se o gatilho deve ou não ser disparado.

4.11 *SENSOR DATA*

Figura 4.14 - Tela de visualização dos dados de um *sensor*

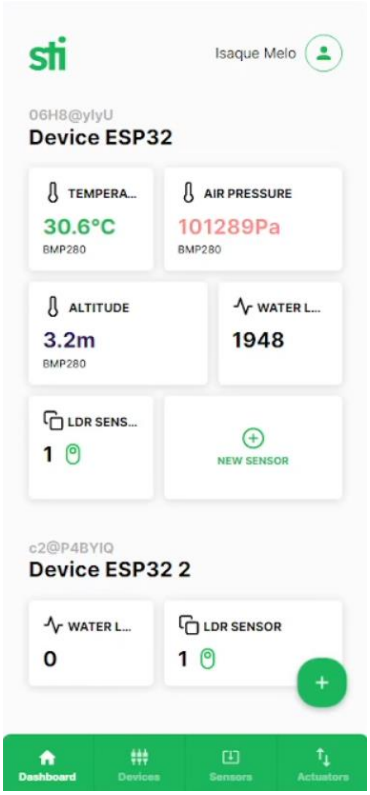


Fonte: Sention, 2022.

Na secção “*Sensor data*”, o usuário tem a capacidade de visualizar a lista dos dados mais recentes do sensor selecionado. Como supracitado anteriormente, para o caso dos sensores que realizam o envio de dados agrupados, o seu conjunto na exibição, também será agrupado. Ao chegar ao fim da página, é possível contemplar o botão “*Load more data*”, permitindo o carregamento de mais dados anteriores aos existentes no início da tela. Outrossim, é possível filtrar tais dados a partir de uma determinada data inserida no campo “*Date*”.

4.12 *DASHBOARD*

Figura 4.15 - Tela de visualização dos *devices* e dos valores dos *sensors*



Fonte: Sention, 2022.

A secção “*Dashboard”* é a tela principal da aplicação web, onde podem ser visualizados todos os *devices* associados e os dados provenientes dos sensores conectados. Ademais, todas as informações são atualizadas automaticamente sem necessidade de intervenção do usuário.

Assim como a secção “*Devices”*, o botão da extremidade direita inferior representada na figura 4.15 pode ser utilizado para iniciar um novo processo de associação de um *device*, fazendo com que o usuário seja redirecionado para a secção “Conexão ao Wi-Fi do dispositivo”, descrito previamente no tópico 4.1.

CAPÍTULO 5

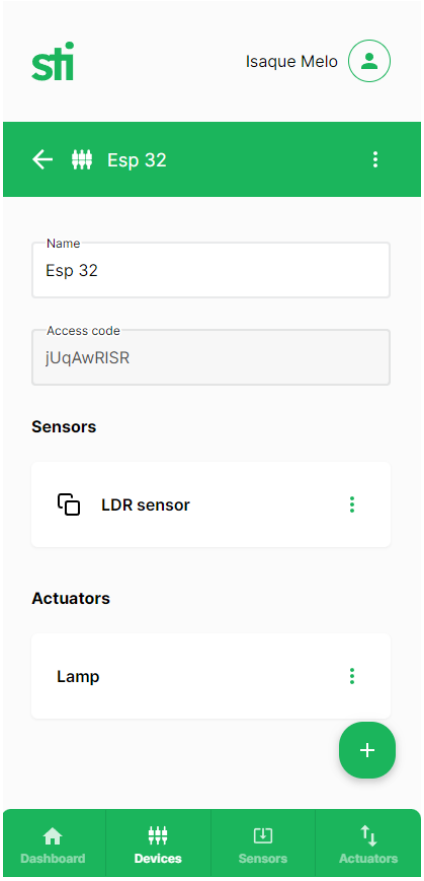
5. EXEMPLOS DE APLICAÇÕES

5.1 CONTROLE DE UMA LÂMPADA EM FUNÇÃO DE SENSOR DE LUMINOSIDADE

Fazendo o uso de um sensor de luminosidade LDR conectado na porta 4 e de um relé controlado pela saída da porta 2 do dispositivo. Realiza-se a seguinte configuração na aplicação:

Acessa-se a tela do *device* e utilizando o botão do canto inferior com o símbolo de “+” é realizado o direcionamento para a tela de criação de um sensor.

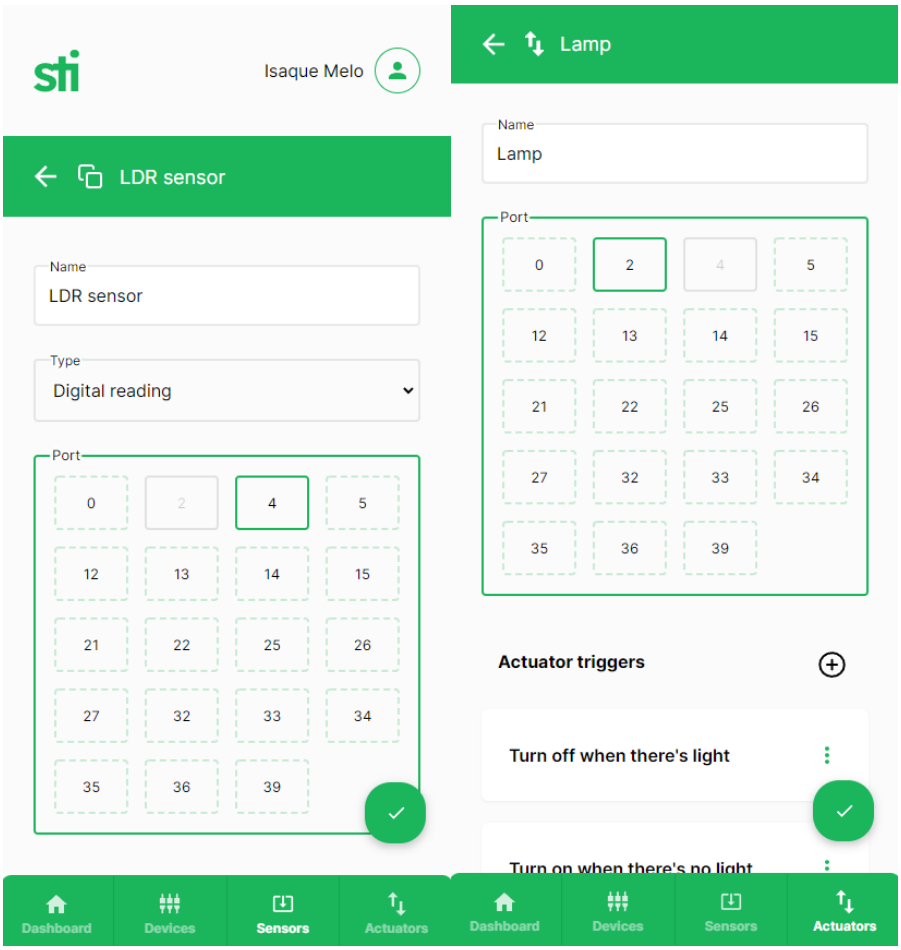
Figura 5.1 - Tela do dispositivo mostrando o sensor LDR cadastrado e a lâmpada como atuador



Fonte: Sention, 2022.

Ao adentrar na tela do sensor (exibido na figura 5.2 no lado direito), o usuário atribui um nome e o tipo de leitura que será realizado. Após isso, ele pode então selecionar a porta na qual o sensor está conectado.

Figura 5.2 - Tela do sensor configurado para a porta 4 e atuador na saída 2

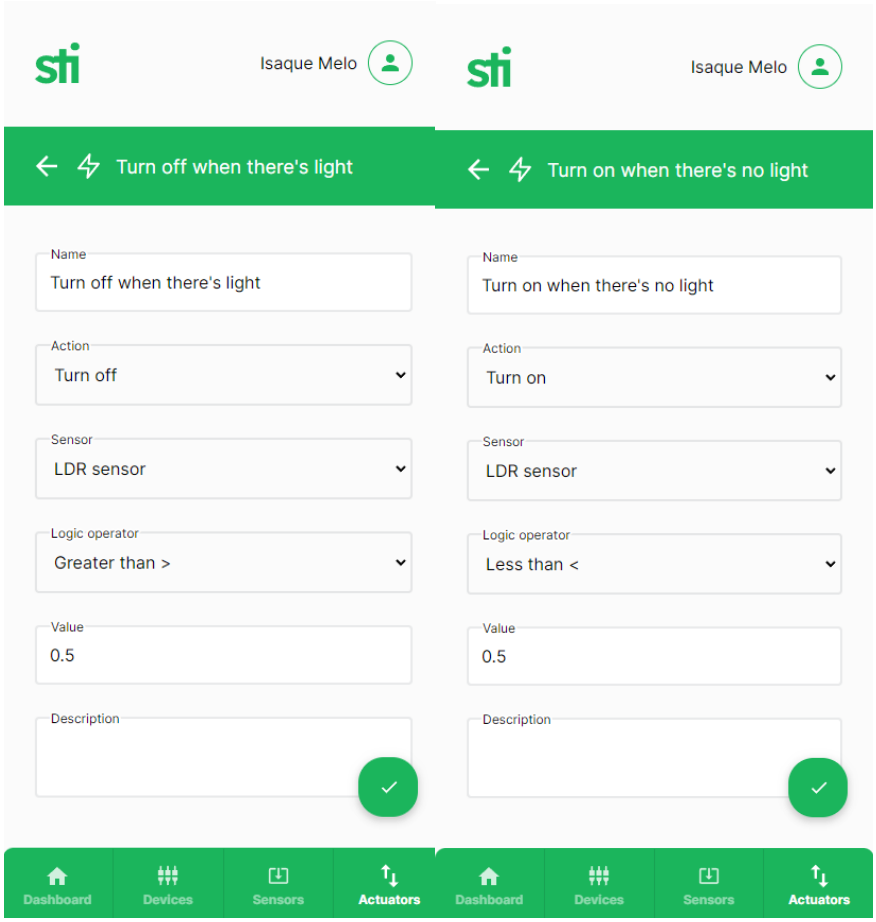


Fonte: Sention, 2022.

Voltando a tela do *device*, o usuário pode realizar a criação do atuador utilizando o mesmo botão com o ícone “+”. Ao clicar, ele é redirecionado para a tela do atuador, exibida na figura 5.2 no lado esquerdo. Nesta tela, o usuário atribui um nome ao sensor e a porta de saída utilizada.

Tendo o atuador cadastrado, pode-se então realizar a criação do gatilho de atuação na tela do atuador. Clicando no ícone “+” ao lado de “*Actuator triggers*”, o usuário é direcionado para as telas exibidas na figura 5.3, nas quais pode realizar a criação do seu gatilho.

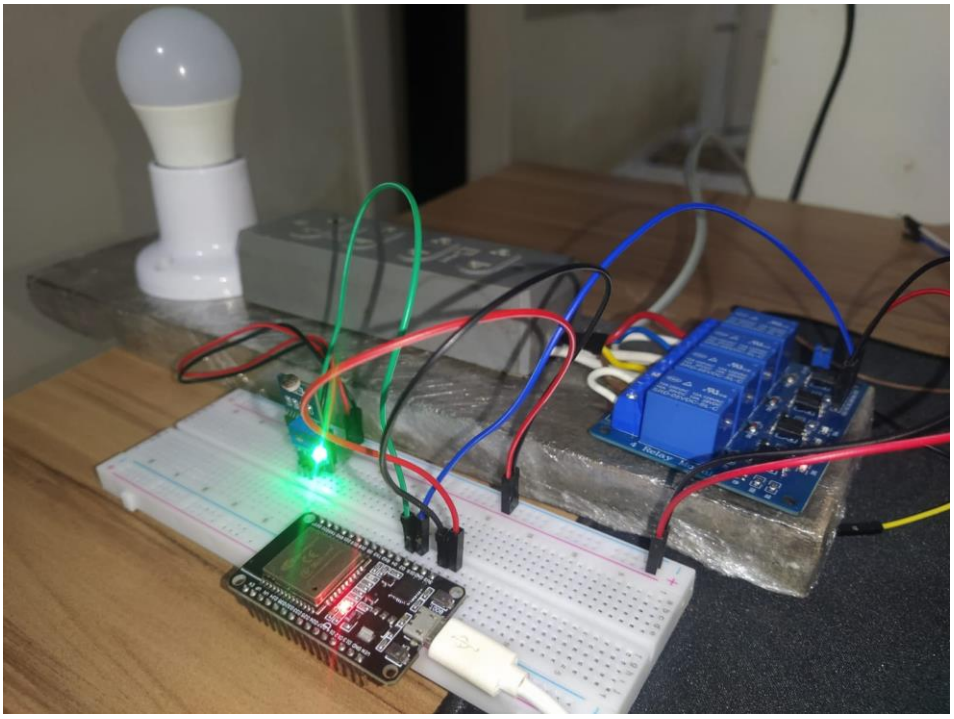
Figura 5.3 - Telas dos gatilhos de atuação que controlarão o atuador em função do sensor LDR conectado no dispositivo



Fonte: Sention, 2022.

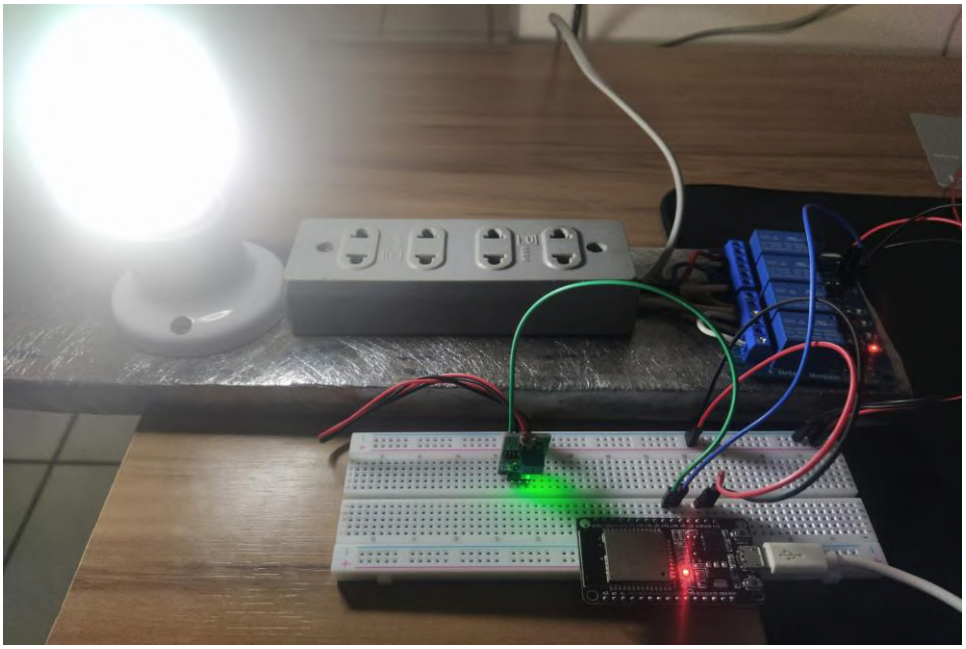
Nas figuras abaixo é exibido o resultado montado para funcionamento da solução.

Figura 5.4 - Exibição da solução montada para controlar a lâmpada em função do sensor de luminosidade (LDR)



Fonte: Acervo pessoal, 2022.

Figura 5.5 - Simulação da lâmpada ligada na leitura de um valor abaixo de 0.5 pelo sensor de luminosidade



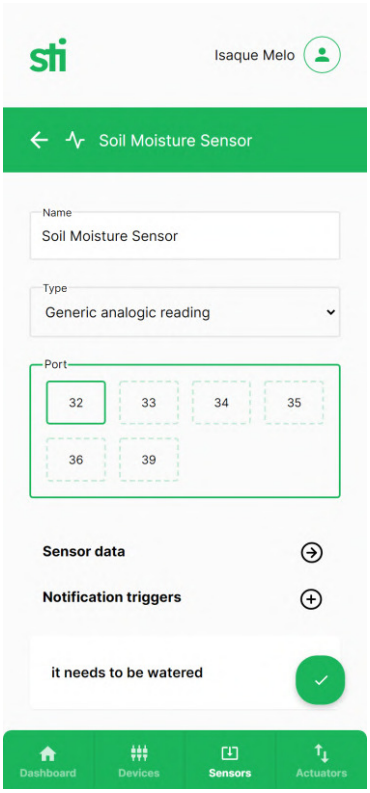
Fonte: Acervo pessoal, 2022.

5.2 NOTIFICAR NECESSIDADE DE REGAR UTILIZANDO SENSOR DE UMIDADE DE SOLO

Partindo da utilização de um sensor de umidade de solo do tipo analógico, é possível processar seus dados e configurar um gatilho de notificação. Esse gatilho alertará ao usuário quando um valor lido do sensor for maior que 2500, informando que a planta em questão necessita ser regada. A seguinte configuração é realizada na aplicação para obter tal resultado:

Inicialmente o usuário realiza a criação do seu sensor, seguindo os passos já descritos previamente. Após a criação do sensor, pode-se então ser utilizada a opção do ícone “+” ao lado do titulo “*Notification triggers*” para criação de um gatilho de notificação.

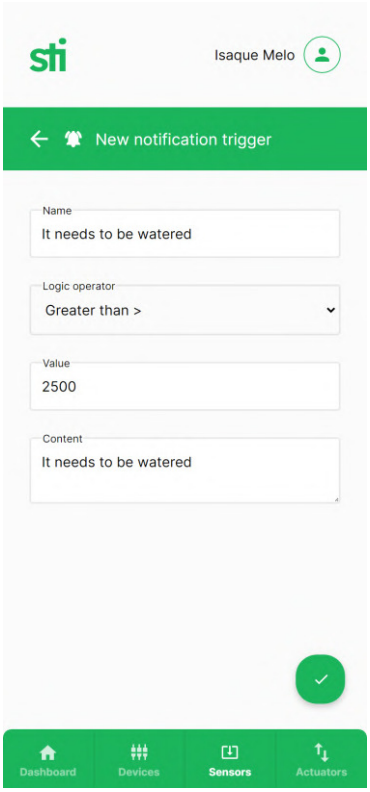
Figura 5.6 - Tela de visualização do sensor configurado para a porta 32



Fonte: Sention, 2022.

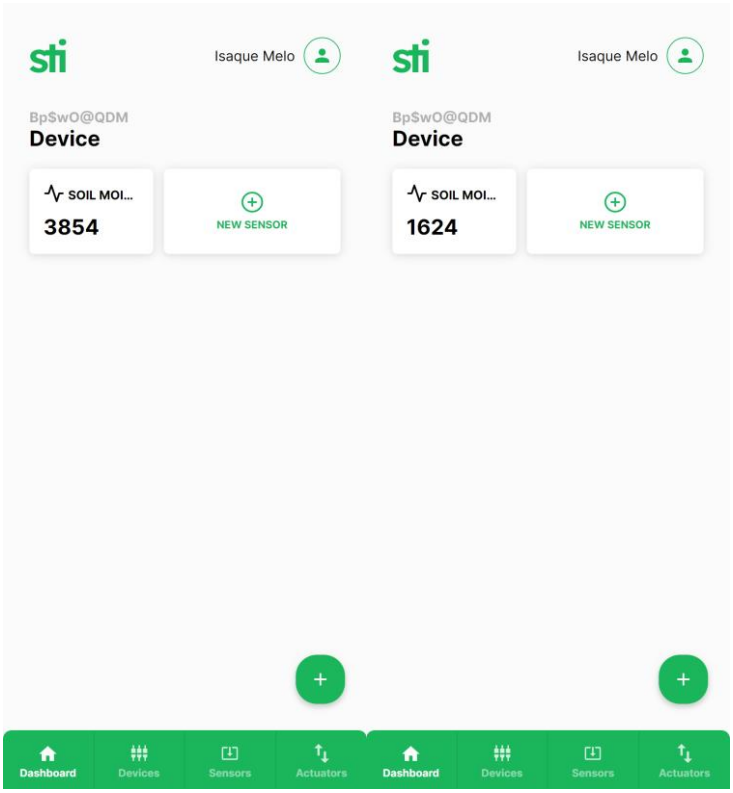
A figura 5.7 exibe a tela de criação de um gatilho de notificação, pelo o qual, foi adicionado um nome, operador lógico, valor e conteúdo. Sendo esse valor relativo ao considerado como limiar para a ativação do gatilho.

Figura 5.7 - Tela do gatilho de notificação definido para o sensor de umidade de solo



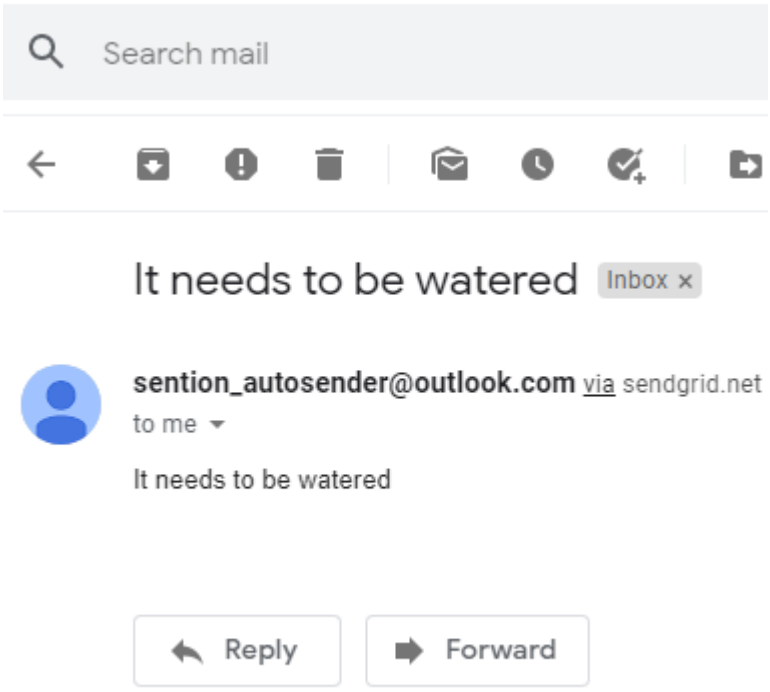
Fonte: Sention, 2022.

Figura 5.8 - Dashboard exibindo os dados do sensor antes e após a planta ser regada



Fonte: Sention, 2022.

Figura 5.9 - E-mail recebido quando o gatilho de notificação é acionado



Fonte: *Print screen* da aplicação Gmail, 2022.

Na figura seguinte é possível visualizar a estruturação da aplicação de exemplo.

Figura 5.10 - Exibição da solução montada, fazendo o uso do sensor de umidade de solo

Fonte: Acervo pessoal, 2022.

CAPÍTULO 6

6. CONSIDERAÇÕES FINAIS

Após o desenvolvimento do proposto projeto, deu-se como resultado uma ferramenta funcional e extensível com potencial significativo. Sua aplicabilidade pode ser dada em diversas áreas da sociedade, além disso, as informações aqui providas tornam-na facilmente replicável e traduzível em outros microcontroladores e tecnologias, abrindo possibilidades de abranger ainda mais cenários de maior complexidade.

6.1 TRABALHOS FUTUROS

A plataforma ainda tem grandes possibilidades de melhorias e introdução de novas ferramentas, como por exemplo, desenvolvimento de firmware para outros microcontroladores. Além disso, por restrições de tempo algumas funcionalidades foram mantidas apenas nos planejamentos, destacam-se as seguintes:

- **Estabelecimento de tempo de ativação / desativação de atuador:**

Quando um gatilho de atuador é disparado em função de uma condicional lógica partindo de dados de um sensor, esse gatilho permanecerá enviando o sinal para o atuador enquanto essa condicional for atingida. No entanto, para alguns cenários pode ser necessário o uso de temporizadores, caso o evento que realize a ativação do gatilho seja breve, mas que necessite manter o atuador realizando a ação por um certo tempo.

Para a implementação, será necessário o uso de um novo campo no gatilho de atuador especificando o tempo que a ação deve ser realizada. Outrossim, o firmware deve também fazer o consumo desse campo, e implementar temporizadores internos para controle individual de cada um dos atuadores cadastrados.

- **Suporte a controle de atuador de forma manual:**

Pode-se pensar no dispositivo como uma máquina que é configurada na inicialização e tem todo seu comportamento posterior definido nessa configuração, por exemplo, quais sensores estão conectados, quais atuadores e quais suas funções de ativação. Dessa forma, o usuário não pode interferir no dispositivo depois que ele está ligado, já que seu comportamento é autônomo, previamente definido na inicialização. Necessitando de uma reinicialização para que uma nova configuração de portas seja carregada.

Partindo disso, hoje o usuário não é passível de controlar um atuador de forma manual pela aplicação, já que se priorizou o conceito de dispositivo autônomo, o que para alguns cenários pode não ser adequado. Para desenvolvimento, uma integração com um serviço de mensageria **MQTT** (2022) pode ser utilizado, sendo de simples implementação e adição ao conjunto da aplicação.

- **Reinicialização automática para aplicar mudanças feitas na aplicação:**

Como mencionado no tópico anterior, para efetivação das mudanças feitas, é necessário a reinicialização do dispositivo. Para contornar tal necessidade de reinicialização manual, o uso de mensageria pode ser também aplicado nesse cenário.

- **Atuador do tipo analógico:**

Como mencionado no tópico **3.1.2** na seção "*Actuator"*, apenas saídas digitais são suportadas pelo firmware. No entanto, para algumas aplicações pode ser necessário o uso de saídas do tipo analógico, fornecendo uma tensão especificada na porta de saída. Para tal, diversos microcontroladores possibilitam esse feito partindo do uso de PWM (*pulse width modulation*), que pode ser utilizado para implementação de uma saída desse tipo.

- **Controle de atuador em função de dados provenientes de outro dispositivo:**

Um dispositivo pode criar gatilhos de atuação apenas com dados de sensores conectados a ele mesmo. Contudo, se o usuário possui vários dispositivos associados à mesma conta, pode ser que um atuador necessite ficar em outro dispositivo por questões de distância e o sensor no ambiente de suas medições. Por estarem em *devices* distintos não será possível criar um gatilho de atuação que consome dados de um sensor conectado em outro dispositivo.

Para resolução desse problema, pode-se usar mensageria na rede local (garantindo que a presença de internet não interfira no controle autônomo dos atuadores), além disso, deve-se também destacar a importância da criptografia e comunicação segura entre os dispositivos.

Problematiza-se também nesse caso uma situação pelo qual os dispositivos podem apenas se comunicar por internet (conectados em redes distintas), de forma que a latência para controle do atuador deve se tornar um ponto notório.

- **Notificação de dispositivo *offline*:**

Caso o dispositivo tenha algum problema de comunicação com a internet, os dados exibidos na aplicação ficarão fixos até que um novo dado seja recebido. Isso requer que o usuário vá na tela do sensor e verifique quando ocorreu o último recebimento de dados, não sendo a forma mais prática para o cliente. Dessa forma, ao identificar que o dispositivo do usuário não está se comunicando com os servidores uma mensagem deve ser exibida no dashboard e uma notificação enviada.

REFERÊNCIAS

ABOUT NODE.js. Disponível em: <https://nodejs.org/en/about/>. Acesso em: 20 jun. 2022.

ATLASSIAN. **O que é o Git: seja um profissional do Git com este guia**. Disponível em: <https://www.atlassian.com/br/git/tutorials/what-is-git>. Acesso em: 20 jun. 2022.

Automação de Processos Industriais e a Indústria 4.0 - Total Automação Industrial. Disponível em: <https://www.totalautomacao.com.br/automacao-de-processos-industriais-e-a-industria-4-0/>. Acesso em: 29 aug. 2022.

Bcrypt. Disponível em: <https://codahale.com/how-to-safely-store-a-password>. Acesso em: 20 jun. 2022.

BITELLA, G. et al. *A novel low-cost open-hardware platform for monitoring soil water content and multiple soil-air-vegetation parameters*. **Sensors** (Basel, Switzerland), v. 14, n. 10, p. 19639–19659, 2014.

BROWN, E. ***Web development with node and express: Leveraging the JavaScript stack***. 2. ed. Sebastopol, CA, USA: O'Reilly Media, 2019.

Controle de Automação Industrial. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/4203366/mod_resource/content/0/Apostila_ControleEAutomacaoIndustrial.pdf>. Acesso em: 29 aug. 2022.

Codahale, 31 jan. 2010. Disponível em: <https://codahale.com/how-to-safely-store-a-password/>. Acesso em: 30 ago. 2022

DOCKER**. O que é o Docker?** Disponível em: <https://www.ibm.com/br-pt/cloud/learn/docker>. Acesso em: 20 jun. 2022.

ELSHAFEE, AHMED & ALAA HAMED, KARIM. ***Design and Implementation of a WiFi Based Home Automation System***. WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY. 2012.

ESP32. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. Acesso em: 23 jun. 2022.

Espressif Announces the Launch of ESP32 Cloud on Chip and Funding by Fosun Group. Disponível em: <https://www.espressif.com/en/media_overview/news/20160907-esp32briefing>. Acesso em: 20 jun. 2022.

Figma. ***About Figma, the collaborative interface design tool***. Disponível em: <https://www.figma.com/about/>. Acesso em: 20 jun. 2022.

FLANAGAN, D. ***JavaScript: the definitive guide***. *Cambridge*: O'reilly, 2011.

GROOVER, M. P. ***Fundamentals of modern manufacturing: Materials, processes, and systems***. 4. ed. *Chichester, England: John Wiley & Sons*, 2010.

JSON web tokens - jwt.lo. Disponível em: <https://jwt.io/>. Acesso em: 20 jun. 2022.

KANAGARAJ, E. et al. ***Cloud-based remote environmental monitoring system with distributed WSN weather stations***. 2015 IEEE SENSORS. Anais...IEEE, 2015.

KUMAR, S. ***Ubiquitous smart home system using Android application***. 2014.

KUMAR, S.; TIWARI, P.; ZYMBLER, M. ***Internet of Things is a revolutionary approach for future technology enhancement:*** *a review***.** Journal of Big Data, v. 6, n. 1, dez. 2019.

MAINWARING, A. et al. ***Wireless sensor networks for habitat monitoring***. Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications - WSNA '02. Anais.New York, New York, USA: ACM Press, 2002.

MARTIN, R. C. ***Clean architecture:*** *A craftsman's guide to software structure and de****sign***. Filadélfia, PA, USA: Prentice Hall, 2017.

MASSERONI, D. et.al ***Field irrigation management through soil water potential measurements: a review.*** 2016.

MCGRATH, M. J.; SCANAILL, C. N.; NAFUS, D. **Sensor technologies: Healthcare, wellness and environmental applications**. 1. ed. Berlin, Germany: APress, 2013.

MicroPython. Disponível em: <https://micropython.org/>. Acesso em: 20 jun. 2022.

Monnit. ***Remote monitoring solutions with wireless sensors for IoT***. Disponível em: <https://www.monnit.com/>. Acesso em: 21 jun. 2022.

MQTT. Disponível em: <https://mqtt.org/>. Acesso em: 27 jun. 2022.

MySQL. Disponível em: <https://www.mysql.com/>. Acesso em: 20 jun. 2022.

Node.js. Disponível em: <https://www.opus-software.com.br/node-js/>. Acesso em: 30 ago. 2022.

KingHost. Disponível em: <https://king.host/wiki/artigo/node-js-caracteristicas/>. Acesso em: 30 ago. 2022.

PRESCOTT, E. et al. ***Hydrosense:*** *An Open Platform for Hydroclimatic Monitoring*. 2016.

PRESSER, M. et al. **The SENSEI project: integrating the physical world with the digital world of the network of the future.** IEEE communications magazine, v. 47, n. 4, p. 1–4, 2009.

***Prisma Client*.** Disponível em: <https://www.prisma.io/client>. Acesso em: 20 jun. 2022.

RAM, K. S.; GUPTA, A. N. P. S. ***IoT based Data Logger System for weather monitoring using Wireless sensor networks.*** International journal of engineering trends and technology, v. 32, n. 2, p. 71–75, 2016.

RAO, B. et al. ***Internet of Things (IOT) Based Weather Monitoring system***. 2016.
React. Disponível em: <https://reactjs.org/>. Acesso em: 20 jun. 2022.

Ruuvi. Disponível em: <https://ruuvi.com/>. Acesso em: 21 jun. 2022.

Sensatio. ***Bring your IoT project ideas to life with the Sensatio Monitor App***. Disponível em: <https://www.sensatio.io/>. Acesso em: 21 jun. 2022.

SOUSA, R. Segunda Revolução Industrial. Disponível em: <https://brasilescola.uol.com.br/historiag/segunda-revolucao-industrial.htm>. Acesso em: 29 aug. 2022.

Swift Sensors. Disponível em: <https://www.swiftsensors.com/>. Acesso em: 21 jun. 2022.

TAMILSELVI, V. et al. ***IoT based health monitoring system.*** 2020 *6th International Conference on Advanced Computing and Communication Systems* (ICACCS). Anais...IEEE, 2020.

TSENG, S.-P. et al. ***An application of Internet of things with motion sensing on smart house***. 2014 International Conference on Orange Technologies. Anais...IEEE, 2014.

Typescript. Disponível em: <https://www.typescriptlang.org/>. Acesso em: 30 ago. 2022.

WIJNEN, B. et al. **Open-source mobile water quality testing platform. Journal of water, sanitation, and hygiene for development: a journal of the International Water Association**, v. 4, n. 3, p. 532–537, 2014.

Cópia de documento digital impresso por GUSTAVO NOGUEIRA (1653598) em 06/02/2023 09:41.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Restrito

Versão Final do TCC - Trabalho de Conclusão de Curso - Arquitetura de hardware e software para sistema de controle e sensoriamento remoto residencial - Erick Pimentel e Isaque Melo

Assunto:	Versão Final do TCC - Trabalho de Conclusão de Curso - Arquitetura de hardware e software para sistema de controle e sensoriamento remoto residencial - Erick Pimentel e Isaque Melo
Assinado por:	Erick Pimentel
Tipo do Documento:	Projeto
Situação:	Finalizado
Nível de Acesso:	Restrito
Hipótese Legal:	Informação Pessoal (Art. 31 da Lei no 12.527/2011)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Erick Spinelli Pimentel**, **ALUNO (201811250001) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 07/12/2022 15:35:39.

Este documento foi armazenado no SUAP em 15/12/2022. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 694435
Código de Autenticação: 450693a647