HENRIQUE DA SILVA CASARES
VINICIUS LOPES SILVA
LUCAS CAMPOS MARTINS FERREIRA BRAGA

PLATAFORMA IOT PARA TODOS

São Bernardo do Campo – SP
2020

HENRIQUE DA SILVA CASARES
VINICIUS LOPES SILVA
LUCAS CAMPOS MARTINS FERREIRA BRAGA

PLATAFORMA IOT PARA TODOS

Trabalho final submetido ao Centro Universitário FEI como parte dos requisitos à obtenção do título de Engenheiro Eletricista com ênfase em Computadores.

Orientador: Prof. Dr. Victor Sonnenberg

São Bernardo do Campo – SP
2020

Lopes Silva, Vinicius.
IoT Para Todos / Vinicius Lopes Silva, Lucas Campos Martins Ferreira Braga, Henrique Silva Casares. São Bernardo do Campo, 2020.
42 f. : il.

Trabalho de Conclusão de Curso - Centro Universitário FEI.
Orientador: Prof. Dr. Victor Sonnenberg.

1. Internet das Coisas. 2. Dispositivos conectados. 3. Código aberto. I. Campos Martins Ferreira Braga, Lucas. II. Silva Casares, Henrique. III. Sonnenberg, Victor, orient. IV. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

Henrique da Silva Casares
Vinicius Lopes Silva
Lucas Campos Martins Ferreira Braga

PLATAFORMA IOT PARA TODOS

Prof. Dr. Victor Sonnenberg
Orientador

Prof. Dr. Marco Antonio Assis de Melo
Examinador (1)

Prof. Dr. Plinio Thomaz Aquino Júnior
Examinador (2)

São Bernardo do Campo – SP
20 de novembro de 2020

Gostaríamos de agradecer primeiramente a todos os funcionários do Centro Universitário da FEI, em especial aos professores Pier Marco Ricchetti, em memória, pela inspiração e dedicação à ensinar; Leandro Alves da Silva, pela enorme ajuda e aconselhamento e ao Victor Sonnenberg, pela orientação e direcionamento. Agradecemos também à nossos familiares e amigos, que acreditaram nos nossos sonhos e nos apoiaram nos momentos mais difíceis de nossa formação.

“Quando tudo parecer dar errado em sua vida,
lembre-se que o avião decola contra o vento, e não a favor dele.“
(Henry Ford)

RESUMO

As tecnologias de comunicação transformaram a vida das pessoas permitindo que elas se conectem umas às outras através de dispositivos eletrônicos de fácil porte e utilização como celulares e computadores. Uma vez provada a eficiência e conveniência desses dispositivos, seu uso se expandiu de forma a integrar comunicação com sistemas de diferentes finalidades localizados em mercados, shoppings e hospitais. Por meio da internet esses dispositivos conectam-se entre si e trocam informações em uma fração de segundo. Sendo assim, pesquisas nesta área de dispositivos conectados são de grande valia para facilitar o dia a dia das pessoas e negócios. O uso destes dispositivos não se restringe ao simples acionamento de cargas e podem atingir até mesmo a análise de dados em massa (*Big Data*) e a Inteligência Artificial. Este modelo de dispositivos conectados é chamado de IoT (Internet das Coisas) e é a base do nosso projeto, que visa desenvolver um sistema que inclua facilidade para o usuário final ao utilizar dispositivos conectados entre si, com uma interface de controle amigável sem a dependência de serviços de terceiros e sem que o usuário tenha conhecimento profundo em programação ou eletrônica. Os métodos utilizados para criação desta solução foram escolhidos com base na praticidade, eficiência e aplicabilidade das ferramentas e principalmente nos conhecimentos adquiridos ao longo do curso.

Palavras-chave: Internet das Coisas. Dispositivos conectados. Código Aberto.

ABSTRACT

The communication technologies changed the way people live by allowing them to connect to each other through easy to carry and easy to use electronic devices such as computers and smartphones. Once proven the efficiency and convenience of those devices, their use got expanded to include communication with systems of different purposes located at supermarkets, malls and hospitals. Therefore, research in this field of connected device are of great value for people and businesses. The use of these devices are not limited to driving electric loads and can even reach the fields of mass data analysis (Big Data) and Artificial Intelligence. This model of connected devices is called IoT (Internet of Things) and is the base for our project, which aims to develop a system to ease the use of connected devices for the end user, with a friendly user interface for control without dependencies on third party solutions and without requiring a lot of knowledge in programming and electronics from the user. The methods used to create this solution were chosen based in pragmatism, efficiency and applicability of the tools and mainly in the knowledge acquired during the graduation.

Keywords: Internet of Things. Connected devices. Open Source.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

1 INTRODUÇÃO

Com o desenvolvimento da tecnologia, a quantidade de dispositivos conectados à internet hoje em dia é bem maior que nas décadas anteriores. Estipula-se que obteve um aumento de cinco vezes em dez anos (NEWS, 2020). O uso desses dispositivos são variados e muitos deles estão dentro de nosso alcance, seja em casa ou serviços que utilizamos. Houveram ações e investimentos para que o desenvolvimento dessa tecnologia fosse impulsionada dentro do Brasil, especificamente para a área de IoT (*internet of things*) voltada para telecomunicações (CRISTONI, 2020). Nos dias atuais com tantos serviços utilizados dentro de casa, seja para acessar um site, ouvir música, assistir vídeo ou navegar no GPS, estamos utilizando um dispositivo conectado e por trás dele tem uma empresa que controla esse acesso e provém facilidade e informações para o usuário final, seja uma pessoa com o intuito de automatizar a casa ou deixá-la mais inteligente, ou até mesmo para controlar aplicações em indústrias. Muitas plataformas e serviços só foram capazes devido à esse avanço, nos proporcionando conforto e praticidade diariamente, porém, devido à isso, muitas empresas cobram por esses serviços.

Cada vez mais pessoas estão conectadas e inseridas na tecnologia, onde os conceitos básicos da tecnologia têm se tornado cada vez mais comum na sociedade, tornando possível que pessoas com baixo nível de conhecimento técnico consigam compreender e desenvolver projetos simples. Nos últimos anos surgiram diversos projetos na área de *IoT* por conta das facilidades e praticidades oferecidas por essa tecnologia, no entanto grandes obstáculos para desenvolver projetos são encontrados, como plataformas de preço elevado e que demandam alto nível de conhecimento técnico.

Os projetos em geral são considerados complexos pois envolvem diversas áreas da tecnologia, por este motivo são considerados como sistemas heterogêneos, envolvendo conhecimentos em programação de sistemas embarcados, análise de esquemas eletrônicos, redes de computadores, protocolos e arquitetura WEB.

Esses fundamentos são essenciais para o desenvolvimento de uma simples aplicação *IoT*, portanto, para permitir que pessoas com baixo conhecimento técnico tenham acesso a um ambiente simples de desenvolvimento *IoT*, é necessário condensar o desenvolvimento desses recursos em uma única plataforma que transcreve de forma intuitiva as configurações essenciais para o usuário desenvolver (PATHIRANA et al., 2017).

A proposta desse projeto é desenvolver uma plataforma genérica que permite que pessoas de baixo ou alto nível de conhecimento técnico possam realizar uma aplicação *IoT* de forma fácil e intuitiva.

1.1 OBJETIVO

Este projeto têm seu valor implementado tanto em software quanto hardware para centralizar um sistema intuitivo e prático com os seguintes objetivos:

a) visualizar periféricos e dispositivos conectados ao sistema;
b) transmitir as informações dos dispositivos para o *Raspberry host*;
c) controlar periféricos e dispositivos através da interface web, mesmo fora de casa;
d) proporcionar ao usuário a flexibilidade de programar as funcionalidades desta plataforma de forma fácil, sem exigir conhecimentos de programação;
e) disponibilizar interface de baixo nível para maior liberdade de programação do usuário, incentivando o conhecimento e aprendizado.

2 FUNDAMENTAÇÃO TEÓRICA

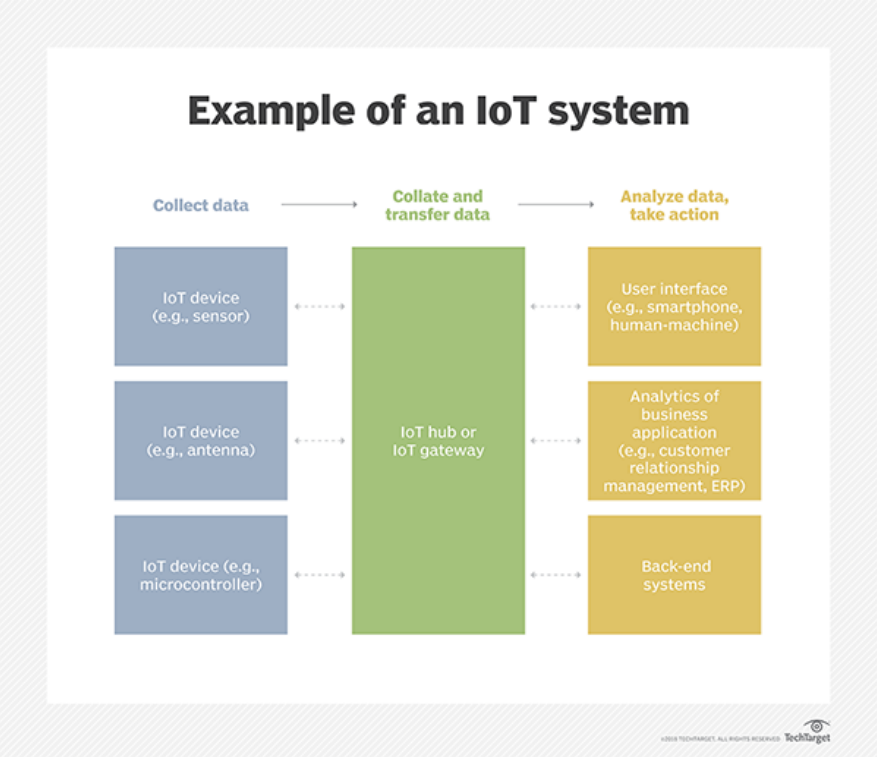
2.1 INTERNET DAS COISAS

Com o desenvolvimento e demanda massiva de um mundo mais tecnológico, muitos conceitos sobre tecnologia foram surgindo e moldando a sociedade ROUSE, 2016. Uma dessas tecnologias em que fazemos uso é a *IoT*, na qual podemos brevemente resumir em um sistema de equipamentos, dispositivos, máquinas, objetos ou pessoas que possam se comunicar entre si, com identificações únicas, capazes de transferir dados através de conexões ou redes sem necessitar interação humana ou de um humano com uma máquina (ROUSE, 2016a). Exemplos reais são carros capazes de enviar sinais ao celular do motorista quando a pressão dos pneus está baixa ou monitoramento de um implante de coração pela central de um hospital, em que esses dados possam ser endereçados utilizando um protocolo de internet (IP). O objetivo principal da *IoT* é fazer com que essas comunicações sejam mais eficientes, rápidas, seguras e até influenciar tomadas de decisões. Um sistema em que utiliza o conceito de internet das coisas pode ser organizado em:

a) coleta de dados gerados por sensores, aparelhos ou máquinas;
b) consolidação e envio dos dados através de uma conexão;
c) análise dos dados para realizar ações ou tomada de decisões.

Na figura 1 de um sistema *IoT* se comunicando, onde uma camada coleta os dados por meio de um dispositivo (sensor, antena ou microcontrolador), esses dados são transmitidos por um *hub* (dispositivo intermediário, ou responsável pela transmissão dos dados), por fim são analisados e as ações são executadas.

Figura 1 – Exemplo de um sistema *IoT*



Fonte: Retirado de (ROUSE, 2016b)

2.1.1 Evolução da Internet das Coisas

Na década de 90 a internet foi capaz de facilitar muitos negócios, porém era muito impactada ainda pela velocidade de conexão e dificuldade de acesso (CHASE, 2013). Nos anos 2000 a conectivade com a internet se tornou mais popular ao ponto de muitas aplicações se tornarem dependentes dela. Atualmente a dependência da internet aumentou porém, ainda é necessária a interação do ser humano e/ou de outras máquinas para que as ações sejam executadas. É nesse contexto que se insere a *IoT*,por exemplo, o acesso à uma página da internet para pesquisas ou realização vendas online podem ser executados de maneira dinâmica e automatizada.

Essa ideia é mais comum ainda nos dias atuais, pois temos várias “coisas” dentro de residências que se comunicam de maneira automatizada e muitas vezes podem ser controladas via um aplicativo de celular, um navegador de internet no computador ou até mesmo uma central eletrônica. Na Figura 2 temos um exemplo de uma casa inteligente conectada.

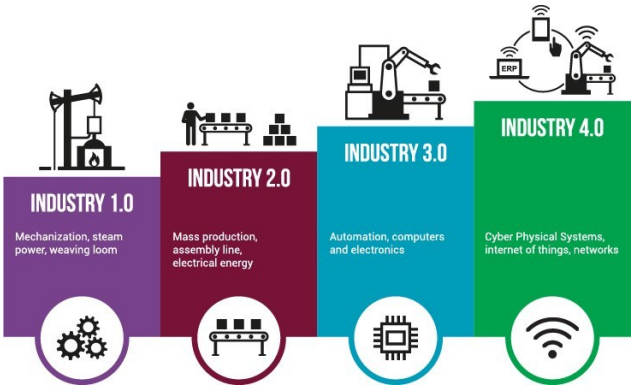
Figura 2 – Exemplo de uma casa conectada por um sistema *IoT*

Fonte: Retirado de (CHASE, 2013)

2.1.2 Indústria 4.0

Com a evolução da era digital, muitos setores foram impctados para ter seu melhor desempenho e crescimento. A transformação digitalização da manufatura que ocorreu durante vários anos, desde a primeira revolução industrial em que eram utilizadas máquinas à vapor, a segunda na qual utilizava eletricidade como revolução, a terceira em que se fez o uso de computadores e máquinas eletrônicas até a quarta na qual tende a emergir a terceira revolução com computadores mais avançados capazes de automatizar uma linha completa com sistemas inteligentes e interligados em que o ponto principal é o tratamento de dados sendo alimentados e gerados por esses computadores (MARR, 2018). A figura 3 demonstra essa evolução.

Figura 3 – Revoluções Industriais



Fonte: Retirado de (MARR, 2018)

Muitos modelos de tecnologia podem ser abordados para alavancar a Indústria 4.0. Exemplos desses modelos são:

a) combinações de sistemas ciberfísicos, computacionais colaborativos que controlam entidades ou equipamentos físicos (SAHRA SEDIGH, 2018);
b) internet das coisas;
c) aprendizado de máquina (*Machine Learning*).

Esses exemplos citados são meios utilizados para trazer valores dentro da indústria por exemplo, para que sejam feitas análises através de inúmeros dados gerados e coletados sobre performance, manutenção, rendimento e defeitos com o objetivo de identificar oportunidades, otimizar logística e a sua cadeia de gestão (*Supply Chain*). Robôs inteligentes para facilitar o processo ou logística de linhas, carros autônomos para automatizar o processo e segurança de locomoção, máquinas autônomas que possam ser totalmente independentes da interação humana para executar sua função e até impressão 3D são alguns dos muitos sucessos da aplicação da Indústria 4.0 no mercado atual. Apesar de parecer que estamos vivenciando essa revolução hoje em dia, estima-se que essa revolução irá permanecer em evolução durante 30 anos em que aos poucos as empresas vão adotando em seus processos.

2.2 MODELO UTILIZADO PARA APLICAÇÃO

Para o desenvolvimento da aplicação utilizamos o modelo cliente servidor em três camadas. Esse modelo permite a modularização da aplicação em: camada de interface com usuário, também conhecida como *front-end*, camada lógica onde estão inseridas as regras de negócios, também conhecido como *back-end* e a camada de dados onde se realiza a comunicação com o banco de dados. Essa arquitetura permite um desacoplamento de código que viabiliza a atualização das partes de forma independente da tecnologia (CHEN; FENG, 2015).

2.3 CONCEITO DO REST API

REST (Transferência representacional de estado) API (Interface de programação de aplicação) analisando os conceitos isolados.

API é uma interface de comunicação disponível por software com regras estabelecidas para que haja o entendimento de ambas as partes. Esse conceito é aplicado para criar uma camada de abstração, no qual para utilizar os recursos de um determinado software é apenas necessário o entendimento da interface disponibilizada por um software, eliminando a necessidade do entendimento total do software, além de viabilizar o conceito de modularização de arquitetura de software. (ADAM; RACHMAT ANOM BESARI; BACHTIAR, 2019)

REST é um modelo de arquitetura criado para suprir conceitos de qualidade e funcionalidade, em princípio para sistemas distribuídos. Acabou sendo utilizada largamente em servidores WEB, tanto para a maneira em que a arquitetura define um conjunto de restrições para uma arquitetura estilo REST, quanto para padrões uniformes para interfaces, sem conservação de estado

e separação total do cliente e servidor. Com isso, permitindo a interoperabilidade entre sistemas. Quando um servidor WEB segue esses conceitos, pode ser denominado de REST FULL.

Portanto uma REST API é uma interface que aplica os conceito REST para sua implementação(COSTA et al., 2014). Quando o protocolo HTTP é utilizado, o URI (Identificador de recursos uniformes) se torna uma representação dos dados e os métodos HTTP são utilizados para manipular esses dados (ADAM; RACHMAT ANOM BESARI; BACHTIAR, 2019), com os métodos:

a) GET para recuperar os dados;
b) POST para criar novos dados;
c) PUT para atualizar os dados;
d) DELETE para apagar os dados.

2.4 BACKEND

2.4.1 Conceito de Back-end

Uma aplicação WEB em geral é dividida em duas principais partes, a camada do *front-end* que é responsável pela interação com o usuário de forma gráfica e o *back-end* é responsável por entregar toda a lógica necessária para o *front-end*, geralmente o *back-end* é dividido em três elementos responsáveis por implementar a lógica do negócio (ADAM; RACHMAT ANOM BESARI; BACHTIAR, 2019):

a) aplicação;
b) banco de dados;
c) servidor.

2.4.2 Linguagem Back-end

A linguagem mais utilizada para programar dispositivos IoT é a linguagem C por ser mais otimizada para essa finalidade, mas por ser uma linguagem de baixo nível, exige uma extensa dedicação de tempo e torna-se moroso para soluções mais complexas. Para contornar essas deficiências, a linguagem Elixir pode ser uma ótima alternativa, por ser uma linguagem de alto nível que não compromete tanto a performance do sistema.

A linguagem Erlang/Elixir é uma linguagem funcional desenvolvida para atingir alta performance e confiabilidade. Foi desenvolvida para aplicações voltadas para telecomunicações que exigem, baixa tolerância de falhas, distribuida e *real-time* (tempo de execução). Por isso conta com um conjunto bibliotecas para desenvolvimento de sistemas de alta confiabilidade conhecida como OTP.

Elixir foi desenvolvida para ser executada na VM (máquina virtual) do Erlang nomeada como BEAM. A linguagem Elixir é relativamente nova, mas já existem diversas aplicações que

a utilizam e inclusive existem recursos para dispositivo embarcado como o *framework* Nerves (FEDRECHESKI; COSTA; ZUFFO, 2016) e para implementação de aplicações WEB como o *framework* Phoenix.

2.4.3 Nerves

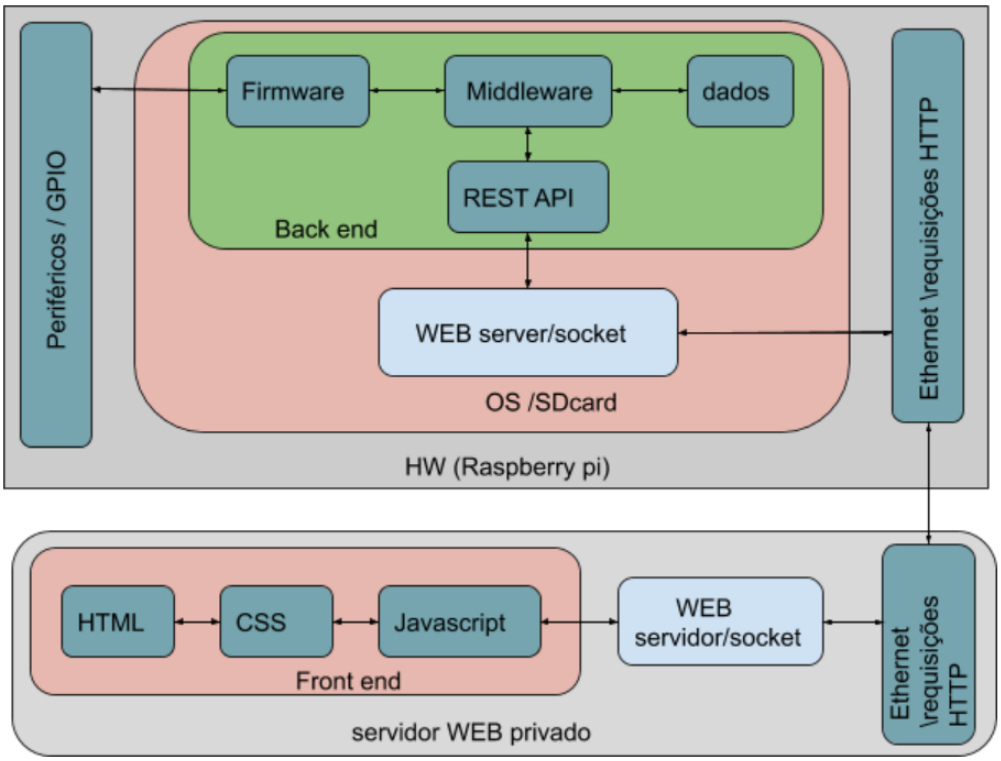
Nerves é um *framework* para desenvolvimento de projetos embarcados, baseado em Linux que apenas executa a VM BEAM, portanto, proporcionando a utilização da linguagem Elixir para o desenvolvimento das aplicações, desfrutando do potencial da linguagem para implementação de projetos embarcados. Além disso, este *framework* proporciona diversas vantagens em relação ao processo tradicional de programação de embarcados (SCHNECK, 2020), como:

a) permitir a fácil portabilidade para diferentes HW (hardware) embarcados e com uma vasta abrangência para diferentes HW embarcados;
b) fácil manutenção e atualização de firmware por viabilizar a atualizações via OTA (atualização sobre o ar);
c) dispensa o processo tradicional de dispor do acesso físico, à flash do dispositivo para o armazenamento do firmware;
d) recursos para facilitar e agilizar o desenvolvimento de firmware;
e) vasta biblioteca para manipulação dos sistemas embarcados.

2.4.4 Phoenix

Phoenix é um *framework* desenvolvido em Elixir para implementação de aplicações WEB, que usa todo o potencial da linguagem Elixir, com isso desfrutando da concorrência fornecida pela VM BEAM, com baixa tolerância a falha e baixa latência. Além de aumentar o desempenho do desenvolvimento de uma aplicação WEB também conta com diversos recurso que são disponibilizados de forma modular (MCCORD, 2020). Na figura 4 temos um diagrama detalhado da arquitetura utilizada no projeto.

Figura 4 – Diagrama simplificado da arquitetura do software



2.5 FRONTEND

2.5.1 Conceito de Front-end

Em uma aplicação WEB a camada de *front-end* é responsável por realizar a interação com usuário, e também é conhecida por *client-side* (lado do cliente), pois todos os recursos desenvolvidos são executados em um *web browser* (navegador web) alocado em um computador pessoal. Para que essa interface com o usuário seja criada sob demanda, é necessário um fluxo de comunicação entre o *browser* e o servidor. Em geral, o *browser* solicita a partir do protocolo HTTP uma requisição para o servidor onde está hospedado todas as informações visuais necessárias, o backend do servidor recebe essa requisição, então realiza as alterações necessárias no servidor e retorna com o conteúdo solicitado. O conteúdo é composto por códigos HTML, CSS e JavaScript, o *browser* recebe esse pacote e renderiza de forma visual a informação solicitada. Essas são as três tecnologias essenciais para a construção de uma interface visual WEB (MDN, 2020).

2.5.2 Mobile Front-end

Por muitos anos o *front-end* teve como objetivo o *browser* para PC, portanto, os *layouts* eram desenvolvidos para telas dentro das dimensões padrões dos computadores. Entretanto, com

o advento dos dispositivos móveis, foram necessárias adaptações à tecnologia que apesar de muito prematura, já demonstrava o seu potencial. Criaram-se então os conceitos de *design* responsivo e *mobile-first* (movél primeiro) que tem como objetivo criar interfaces que podem ser adaptadas para quaisquer que sejam as dimensões das telas, desde um computador até dispositivos como *smartwatch*.

Com a tendência de acesso à páginas WEB a partir de quaisquer dispositivos móveis, houve um realce no potencial da tecnologia IoT, viabilizando o controle de qualquer dispositivo a partir de até mesmo um pequeno *smartwatch* em qualquer lugar do mundo (SABURIDO, 2018).

2.5.3 Hyper Text Markup Language (HTML)

É uma linguagem de marcação padrão para o desenvolvimento WEB, que descreve a estrutura de uma *webpage* de forma semântica, por isso é considerada uma linguagem de marcação ao invés de uma linguagem de programação. Essa descrição é renderizada pelo navegador WEB em uma forma gráfica (MDN, 2020).

2.5.4 Cascading Style Sheets (CSS)

Também é considerada uma linguagem de marcação, porém com o objetivo de criar uma folha de estilo para páginas WEB. É muito utilizada para modificar a estrutura de uma página WEB escrita em HTML. O CSS demonstra um aspecto mais moderno com muito mais recursos visuais e de design para as aplicações WEB. Isso se dá pois o CSS permite alterações no *layout* geral do texto e da página, além de permitir mais agilidade nas aplicações, por diminuir o tamanho dos arquivos em relação ao HTML (MDN, 2020).

2.5.5 Javascript

Javascript é uma linguagem de programação interpretada, multi-paradigma, dinâmica e sem tipos, voltada para desenvolvimento WEB. Em comparação às linguagens HTML e CSS, que são voltadas para a criação de elementos estáticos na aplicação WEB, o Javascript permite o desenvolvimento de aplicações mais complexas, trazendo mais dinâmica e versatilidade para as aplicações WEB (MDN, 2020).

2.6 ARQUIVO JSON E XML

Para padronizar a transmissão de informações atavés de aplicações e dispositivos diferentes, foram criados vários formatos de texto para sua formatação ideal à aplicação, desenvolvedores e ao próprio usuário. Dessa forma a escolha deve ser analisada para melhor praticidade, desempenho e segurança. O formato JSON (*JavaScript Object Notation*) é utilizado em aplicações compactas,

de padrão aberto e de fácil compreensão , sendo muito prática para serem visualizadas de forma geral e escritas ao mesmo tempo, ao contrário de formatações como o XML (*Extensible Markup Language*), o JSON é uma formatação totalmente independente que surgiu do JavaScript, porém usa convensões familiares de outras linguagens da família C (C, C++, C#), Python, Perl, o que a torna muito prática para formatação em várias plataformas. A seguir, exemplos da formatação em XML e JSON:

a) XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<cardapio>
<comida>
  <nome>Hot Dog</nome>
  <preco>R\$ 8,99</preco>
  <descricao>
  Pão com salsicha completo (acompanha batata palha, ketchup e
  ↪ maionese)
  </descricao>
  <calorias>650</calorias>
</comida>
<comida>
  <nome>Hamburguer</nome>
  <preco>R\$ 10,99</preco>
  <descricao>
  Pão com 1 fatia de hambúrguer, 3 fatias de tomate, salada, ketchup e
  ↪ 1 fatia de queijo
  </descricao>
  <calorias>950</calorias>
</comida>
</cardapio>
```

b) JSON:

```
[
  {
    "nome": "Hot Dog",
    "preco": "R\$ 8,99",
    "descricao": "Pão com salsicha completo (acompanha batata palha,
    ↪ ketchup e maionese)",
    "calorias": "650"
```

```
  },
  {
    "nome": "Hamburguer",
    "preco": "R\$ 10,99",
    "descricao": "Pão com 1 fatia de hambúrguer, 3 fatias de tomate,
    ↪  salada, ketchup e 1 fatia de queijo",
    "calorias": "950"
  }
]
```

O formato JSON, como mencionado, tem um visual mais limpo, com uma leitura mais simples do arquivo e maior velocidade na manipulação e transmissão dos dados.

2.7 SEGURANÇA EM SOFTWARE

Com o crescimento da tecnologia *IoT*, nasceram novos problemas de segurança, uma vez que existe uma grande diversidade e quantidade de objetos conectados na rede. O principal problema se dá pela falta de cuidado no design de software o que abre caminho para que *malwares* e *backdoors* se instalem. Além disso, a dificuldade de identificar os objetos na rede e gerar métodos de autenticação seguros (métodos tradicionais não são aplicáveis devido a heterogeneidade dos objetos) são importantes problemas a serem solucionados. Outro obstáculo está em manter a privacidade dos dispositivos *IoT*. Durante o uso da tecnologia, dados do comportamento e da rotina diária são coletados a fim de melhorar os serviços e a experiência pessoal de cada usuário. Entretanto, as informações coletadas descrevem o usuário detalhadamente, assim é necessário cautela quanto à exposição desses dados, para evitar o uso indevido de informações pessoais. É importante ainda ter em mente, que alguns dispositivos possuem capacidade de computacional limitada e restrições de consumo elétricos por terem baterias limitadas, portanto, não suportam sistemas complexos de criptografia por exigirem poder computacional elevado e consequentemente o aumentando do consumo elétrico. Algoritmos mais simplificados de criptografia são mais adequados para dispositivos com restrição de recursos. Uma segunda solução é ocultar alguns dados, porém isso ainda é um obstáculo uma vez que os objetos conectados precisam ter acesso a tais informações (ZHANG et al., 2014).

2.8 PROTOCOLOS DE COMUNICAÇÃO

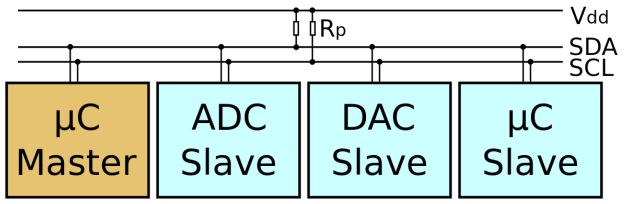
Com a demanda do uso de tecnologias relacionadas à área de Internet das Coisas, várias camadas de comunicação e estruturas se adaptam para maior praticidade do uso de dispositivos diversos, que por sua grande maioria se tratam de microcontroladores e sensores, na qual estão

conectados entre si, seja por LAN (*Local Area Network*) ou WiFi (*Wireless Fidelity*). Para que essa comunicação seja eficaz e segura, foram padronizados protocolos de comunicação que são vastamente utilizados no período atual por várias empresas.

2.8.1 i2c

O I2C é um protocolo de comunicação em que o dado é transferido *bit* a *bit* através de uma única conexão de uma maneira síncrona, fazendo com que a saída desse sinal seja baseada no *clock* (período de oscilação) enviado do *Master* (Mestre) ao *Slave* (Escravo), onde esse *clock* é apenas dependente do Master. Esse método de comunicação necessita apenas de 2 conexões, o SDA (*Serial Data Access*) e o SCL (*Serial Clock Access*).

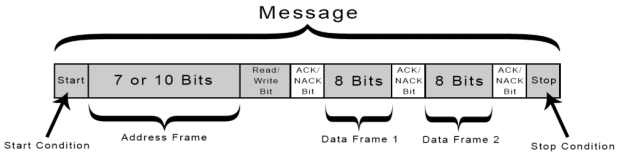
Figura 5 – Protocolo I2C



Fonte: Retirado de WIKIPEDIA, 2019

Com a comunicação via I2C, os dados são quebrados em pequenos pacotes em que formam esse padrão. Esses pacotes contém informações do endereço do *Slave*, *bits* de condição de início e fim da menssagem transmitida, o dado a ser transmitido e um sinal de *feedback* chamado de positivo ACK (*Acknowledge bit*) quando a mensagem é entregue ao escravo, ou negativo NACK (*No-acknowledge bit*) quando a mensagem não é entregue ao escravo, de acordo com a figura 6.

Figura 6 – pacote i2c



Fonte: Retirado de (BASICS, 2016)

2.8.2 UART

A comunicação UART (*Universal Asynchronous Receiver-Transmitter*) é um dispositivo presente dentro de um microcontrolador em que seu objetivo é controlar a entrada e saída de dados através de sequências de 8 *bits*. Atualmente é o método mais comum para comunicação serial *full-duplex* (CODREY, 2020) A comunicação UART não se refere apenas à um protocolo, mas um

conjuto completo de circuitos integrados. Os dispositivos UART possuem duas conexões, uma para transmitir dados (TX) e outra para receber os dados (RX), de uma maneira totalmente assíncrona ao *clock*. Da mesma forma que citamos do I2C, o UART também utiliza *bits* para identificação do início e fim do pacote enviado ou recebido. Importante citar que todos os dispositivos no circuito devem ter o mesmo *baud rate* (taxa de transmissão) como padronização.

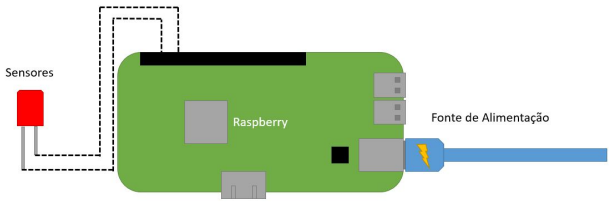
3 MODELAGEM DO HARDWARE

A montagem e relação entre os dispositivos, microcontroladores e conexões pode ser resumida de acordo com a seguinte exemplificação: o microcomputador comunica ou com os sensores e dispositivos, sejam eles sensores de temperatura, humidade, luminosidade que captam valores do ambiente em que se localiza ou com lâmpadas ou leds e enviam para esse microcomputador, na qual é o próprio servidor em que armazena localmente suas informações recebidas e processa de forma organizada para otimizar seu uso.

Esse microcomputador processa então essas informações para representar como interface humana para que o usuário possa integarir, através de um navegador da internet em outro aparelho qualquer, seja celular ou computador. Na Figura 7 é representado o diagrama de blocos do hardware deste projeto, onde está separada por 3 partes principais:

a) Fonte de alimentação;
b) Raspberry Pi 3 B+, o microcomputador do projeto;
c) Sensores, responsáveis pela comunicação externa do microcomputador ao ambiente instalado.

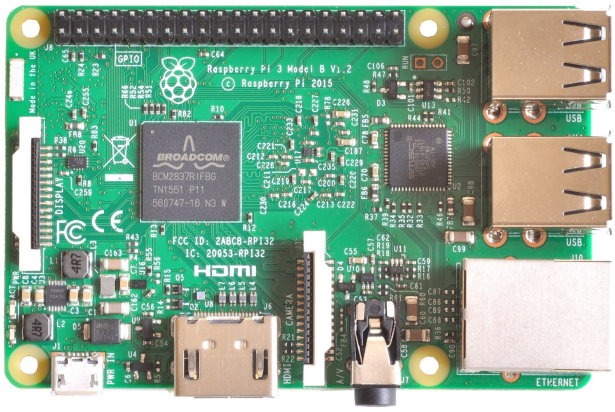
Figura 7 – Diagrama de *Hardware*



Neste projeto foi escolhido o microcomputador Raspberry Pi 3 B+, exibido na Figura 8, desenvolvido no final de 2014 pela *Raspberry Pi Foundation* com as seguintes especificações:

a) SoC (*System On a Chip*): Broadcom BCM2837B0 quad-core Cortex-A53 (ARMv8) 64-bit 1.4GHz;
b) 1 entrada *Gigabit Ethernet* e WiFi (2,4 e 5 GHz);
c) memória: 1GB LPDDR2 SDRAM;
d) GPIO (*General Purpose Input/Output*)de 40 pinos;
e) Entrada de 5V/ 2,5A;
f) memória: 1GB LPDDR2 SDRAM;
g) armazenamento: cartão *Micro SD*.

Figura 8 – Raspberry Pi



Fonte: Retirado de (EAMES, 2016)

O motivo deste microprocessador ser escolhido foi devido à praticidade de seu uso, seja informações na internet, quanto a portabilidade nas conexões de GPIO, conexão à internet via cabo ou WiFi, quantidade de memória suficiente para um sistema operacional ideal, baixo consumo de energia e preço agressivo comparado à outros modelos de processadores ou microprocessadores.

3.1 SENSORES

3.1.1 Conexão dos sensores à Raspberry Pi 3B+

Todas as conexões que realizamos dos sensores e leds foram feitas atráves da GPIO do microprocessador, o Raspberry, através de uma *protoboard* e cabos para conectá-los. Sensores, atuadores, microcontroladores são componentes vastamente utilizados para projetos de Internet das Coisas para sua conectividade e interação sejam realizadas (EMBARCADOS, 2020). Sensores por sua vez, são práticos e de baixo custo para trazer um valor final ao projeto, como monitorar temperatura de uma sala de servidor, controlar humidade de um campo de plantações, abertura de cortinas, entre outros objetivos.

3.1.2 Sensor de Luminosidade - BH1750

O sensor de luminosidade BH1750 9 é um dispositivo sensível à luz que possui uma interface de comunicação 16 *bits* e interface de comunicação de acordo com o protocolo I2C e uma característica de sensibilidade ao espectro de luz muito parecido ao olho humano. O sensor BH1750 possui as seguintes especificações:

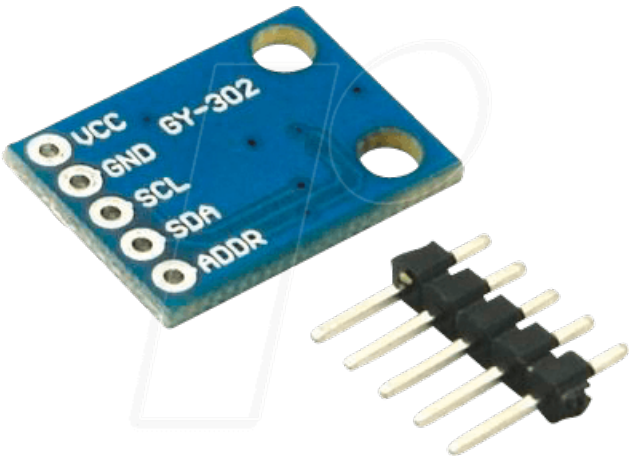
a) sensor de luminosidade com faixa de operação de 1 lux a 65.535 lux, precisão de +/- 1 lux e resolução de 1 lux.

Figura 9 – Sensor de Luminosidade - BH1750

Fonte: Retirado de (IMASTERS, 2017)

Esse dispositivo tem configuração de Entrada e Saída para 5 pinos de acordo com a Figura 10, sendo eles: alimentação (Vcc e Terra), pinos para o protocolo I2C (SCL e SDA) e um pino analógico para o valor do sensor, nomeado como ADDR.

Figura 10 – Pinos BH1750



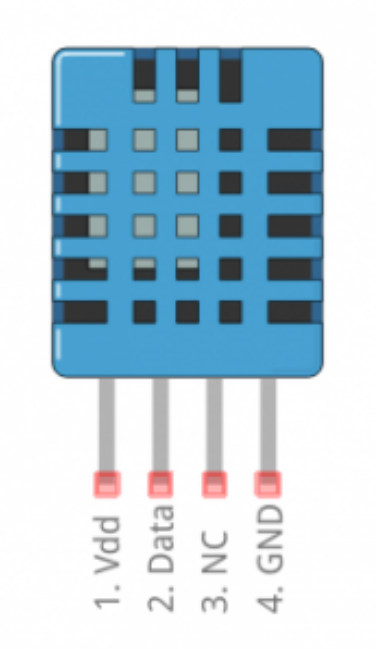
Fonte: Retirado de (REICHELT, 2020)

3.1.3 Sensor de Temperatura e Humidade - dht11

O sensor dht11, mostrado na Figura 11 é um dispositivo sensível à temperatura e humidade que pode ser utilizado em conjunto com vários microcontroladores, através do sensor capacitivo de umidade e um termistor para medir o ar e enviar um sinal no pino de saída do mesmo. O sensor dht11 possui as seguintes especificações:

a) sensor de temperatura com faixa de operação de 0ºC a 50ºC, precisão de +/- 2ºC e resolução de 1ºC;
b) sensor de humidade com faixa de operação de 20%HR a 90%HR, precisão de +/- 5%HR e resolução de 1%HR.

Figura 11 – Pinos dht11



Fonte: Retirado de (MAXPHI, 2017)

A comunicação que é mais utilizada para esse dispositivo é a *adhoc* dh11.

4 MÉTODO

Para demonstrar como foi desenvolvido o material técnico e a aplicação de valor deste projeto, seguimos por diversas pesquisas e análises de como chegar à um desenvolvimento onde fosse viável sua execução, aplicação e escalabilidade, para que tenha necessidade futura de seu uso contínuo.

4.1 LEVANTAMENTO BIBLIOGRÁFICO

A aplicação do projeto de forma adequada, levou à muitas buscas teóricas e estudos variados da maneira que pode ser avaliada e aplicada às diversas tecnologias e ferramentas.

Dentre elas, podemos citar as mais importantes considerações levadas:

a) levantamento de requisitos;
b) linguagem de programação;
c) ferramentas tenológicas para auxiliar o desenvolvimento e organização do projeto;
d) casos de uso;
e) aplicações reais com objetivos similares;
f) aplicações com as mesmas *frameworks* utilizadas no projeto;
g) viabilidade.

4.1.1 Levantamento de Requisitos

Foco em analisar quais objetivos queremos chegar e por qual método iremos realizar isso. Utilizamos o microprocessador Raspberry pela sua facilidade de acesso e quantidade de informações de diversos usuários e desenvolvedores, para controlar dispositivos pela internet através de uma maneira simples e otimizada via celular ou computador.

4.1.2 Linguagem de Programação

Como existem diversas linguagens de programação, foram realizadas análises variadas das mesmas para entender qual o melhor uso e como se encaixam à essa aplicação que escolhemos. Dentre as muitas, optamos por selecionar linguagens que possuem boa interação com o *front-end* e o *back-end*, mantendo um bom desempenho e com um visual prático para o desenvolvimento.

4.1.3 Ferramentas Tenológicas

Ferramentas são necessárias para auxiliar no desenvolvimento, sejam anotações em um tablet, dispositivos auxiliares para realizar análises ou medições, que não fazem parte exatamente do projeto final, mas tem um grande envolvimento com o mesmo.

4.1.4 Casos de Uso

Empresas focam em atender a necessidade de seus clientes de uma forma em que eles saiam satisfeitos e recomendem o serviço, produto ou suporte. Isso faz com que muitas soluções sejam críticamente pensadas e desenvolvidas. Para validação do uso do Elixir, pesquisamos grandes empresas que utilizam essa linguagem em seu *background* (RUBIO, 2019) e encontramos soluções como as seguintes:

a) Pinterest, uma rede social com 200 milhões de usuários ativos, utiliza Elixir para gerenciar rotas de eventos no sistema;
b) Financial Times, um jornal eletrônico inglês, utiliza Elixir como ferramenta de meta-programação para criar os DSLs (domínio de linguagens específicas);
c) Toyota Connected, um serviço de conexão veícular, utiliza Elixir para enviar eventos em tempo real para a central, sobre o trânsito e comportamento do motorista;
d) SquareEnix, uma grande desenvolvedora de jogos, utiliza Elixir para autenticar jogadores online e comunicação durante o jogo;
e) PepsiCo, empresa americana de alimentos, utiliza o Elixir para manter sua ferramenta de comércio virtual ativa.

Com base em todas as empresas analisadas e seus exemplos citados, o uso no Elixir ficou mais claro e objetivo.

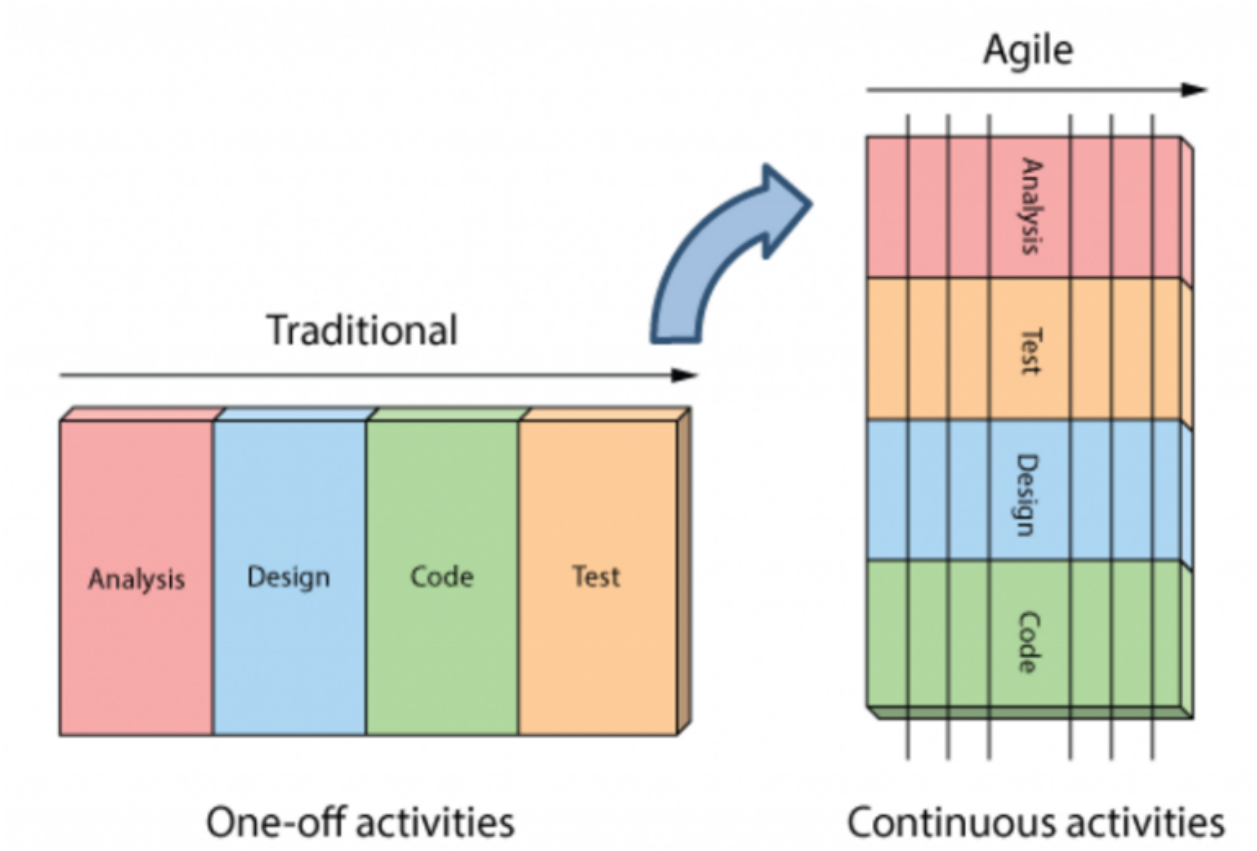
4.1.5 Aplicações Reais

Pesquisas com aplicações IoT foram realizadas, em que muitas delas utilizam Elixir como base da linguagem e um microprocessador principal, o Rasberry Pi e um microprocessador auxiliar, no caso, um Arduino UNO (GILMAR, 2018), para fazer uma comunicação via WEB. Provando que aplicações com Elixir e Raspberry são utilizadas e aplicadas.

4.2 MODELO DE DESENVOLVIMENTO

Existem vários modelos de desenvolvimento que seguem um padrão no momento de criação de negócios, dentre eles podemos citar o Agile, Scrum e Kanban. Cada um desses com seus objetivos e vantagens. Neste projeto focamos em utilizar o modelo Agile, na qual foca em entregar valor através de um *software* que funciona em todos seus quesitos, do que documentações vastas e diversas sobre todas as ações tomadas, execuções detalhadas e implementações sucedidas, porém, não deixa de documentar fatos mais importantes, fazendo com que a análise, teste, desenvolvimento e *design* sejam realizados em paralelo. Na Figura 12 temos uma demonstração do método citado.

Figura 12 – Método Agile



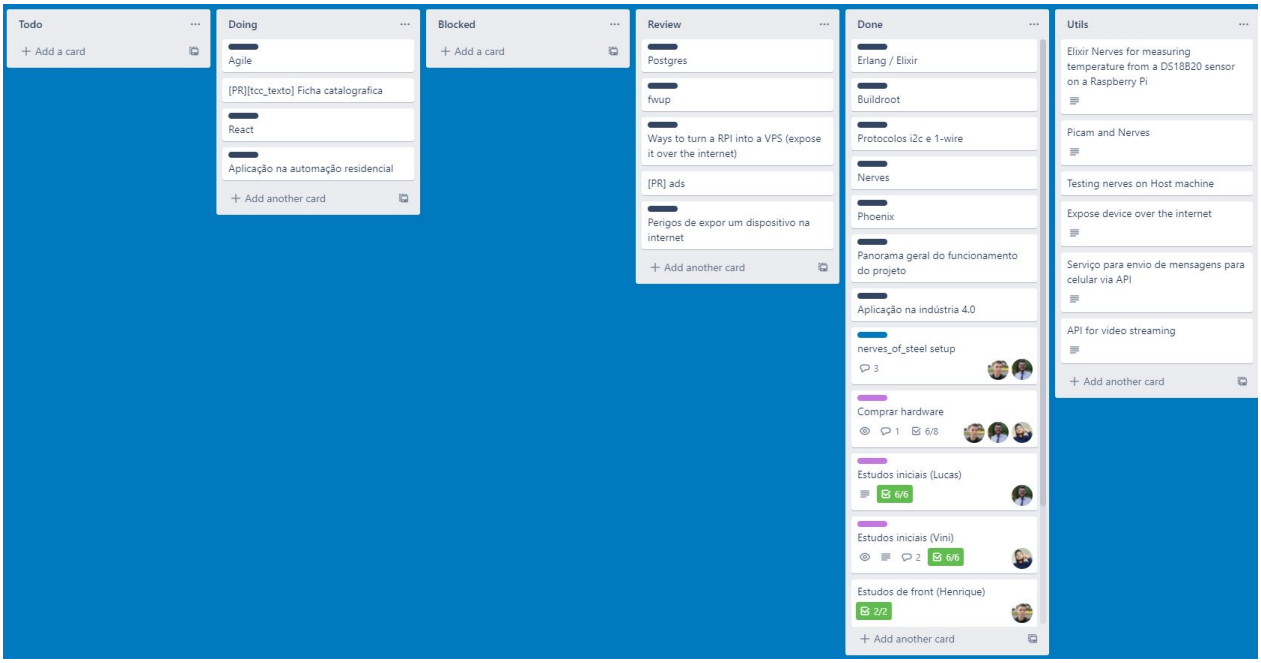
Fonte: Retirado de (GHUNTER, 2018)

O Kanban, o outro método baseado, trata de um quadro onde colunas de tarefas a ser realizadas, sendo realizadas e já realizadas, são preenchidas com atividades e ao longo de seu desenvolvimento vamos movendo essas tarefas para demonstrar que está ocorrendo evolução. O intuito é que a equipe de desenvolvimento mova essas tarefas assim que cumpridas, de acordo com o perfil e prazo de cada atividade, facilitando a visualização do andamento geral do desenvolvimento. Existem diversas ferramentas com esse intuito, sendo a que escolhemos para nosso projeto o Trello.

4.3 TRELLO

Trello é um ferramenta onde é possível gerenciar projetos com anotações e cartões, com visual intuitivo e simples manuseio. O objetivo de seu uso é facilitar o desenvolvimento em partes deste projeto e que sejam compartilhadas as conclusões com o time envolvido. Para o Trello utilizamos uma adaptação do método Kanban para realizar as atividades. Na Figura 13 temos um exemplo do visual do Trello durante o desenvolvimento:

Figura 13 – Ferramenta Trello



4.4 FEATURE MAP

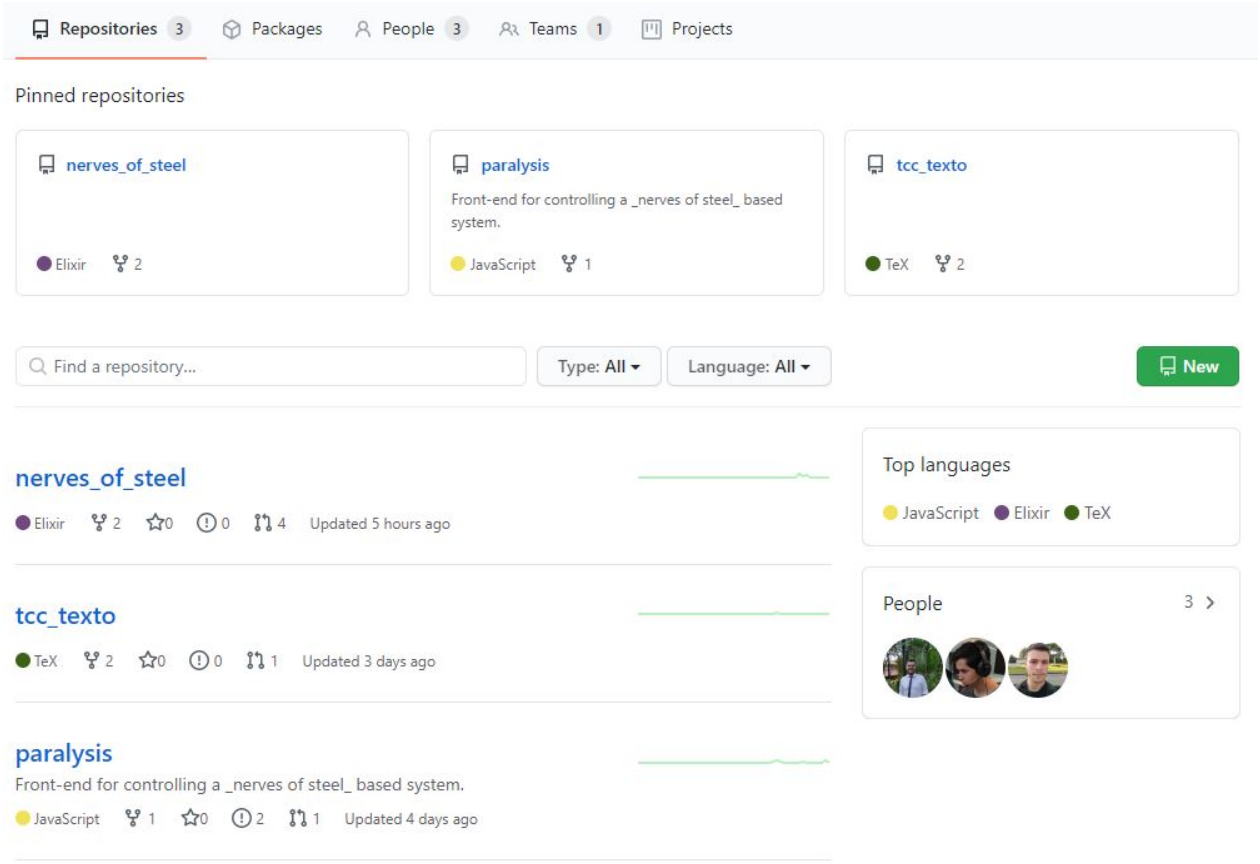
Para aprofundar em nossos objetivos de projeto, analisamos quais seriam os objetivos finais dos nossos usuários, com isso utilizamos ferramentas em que podemos traçar as necessidades de clientes e suas estórias (*User Stories*) de forma adequada para cada característica (*Feature*). O Feature Map auxilia na visualização dessas características de como caminhar no desenvolvimento com foco em alcançar os objetivos dos usuários finais.

4.5 GITHUB

Para ferramenta auxiliar de gerenciamento de códigos optamos pelo GitHub, na qual tem o intuito de realizar a hospedagem de código fonte, arquivos com controle de versão e integração com várias ferramentas e ambientes. Nele podemos realizar alterações no código em tempo real e acomapnhar todas essas modificações de uma maneira integrada. No projeto criamos 3 repositórios:

a) nerves of steel, onde hospedamos nosso código *backend*;
b) paralysis, onde hospedamos nosso código *frontend*;
c) tcc texto, onde hospedamos nossa dissertação e documentação do tcc.

Figura 14 – Ferramenta GitHub



4.6 PHOENIX

O *framework* Phoenix foi de extrema importância para o desenvolvimento do *back-end* desse projeto. Os primeiros passos foram estudar a documentação do *framework* e realizar as sugestões de projetos iniciais da documentação. Todo o desenvolvimento e simulação do webserver foi realizado em *localhost* (hospedeiro local). Então realizamos o desenvolvimento de um web server que consiste em configurar todos os parâmetros de acesso como porta e IP, além das definições de rotas, definição das funções do controlador e os templates do *view*(dados visuais). As rotas foram definidas de forma que representasse uma ação no dispositivo empregando a arquitetura REST. O controlador foi configurado para receber uma requisição GET com uma simples rota e renderizar o template JSON.

Após a consolidação e sucesso na etapa anterior, começamos a de fato desenvolver o *back-end* para o nosso projeto. Começamos com os recursos mais simples, como a definição para controlar os GPIOs e até a leitura de sensores mais complexos. Portanto, definimos todas as rotas necessárias para interagir com o dispositivo. Para cada rota definimos uma função específica no

controlador que por sua vez recebe as requisições e realiza a lógica necessária para executar a ação. Dependo da requisição é necessário retornar informações, por tanto usamos templates em formato JSON. Todos os testes de requisições foram realizados com a ferramenta Postman que permite realizar diferentes tipos de requisições. Após a validação do funcionamento realizamos a integração com *firmware*.

Figura 15 – Rotas definidas para acesso a os recurcos do hardware

```
 led_path  GET  /led/enable              JsonApiWeb.LedController :enable
 led_path  GET  /led/disable             JsonApiWeb.LedController :disable
gpio_path  GET  /gpio/read/:port         JsonApiWeb.GpioController :read
gpio_path  GET  /gpio/write/:port/:value JsonApiWeb.GpioController :write
gpio_path  GET  /dht/:port               JsonApiWeb.GpioController :dht
gpio_path  GET  /lum                     JsonApiWeb.GpioController :lum
```

4.7 APLICAÇÃO COM NERVES

O primeiro passo para a utilização do *framework* Nerves em nosso projeto foi realizar o estudo da documentação. Após uma extensa revisão do documento, implementamos um projeto simples que foi sugerido pelo mesmo. Este, consistiu em controlar um led *onboard* do Raspberry pi a partir de uma requisição HTTP.

Após consolidação do processo de compilação e entendimento da lógica aplicada nessa aplicação, iniciamos o desenvolvimento do projeto de forma modular e gradativa. Primeiro, começamos com o controle dos GPIOs, em seguida com o sensor de temperatura/umidade e por fim o sensor de luminosidade. Para cada implementação foi necessário localizar e estudar os melhores recurso disponível.

Com o desenvolvimento do *firmware* finalizado, partimos para a integração da comunicação entre o *firmware* e o *back-end*, com a finalidade de testar todos os recursos implementados no *firmware*. Após a validação do funcionamento, iniciamos o desenvolvimento do *front-end* para criar uma interface intuitiva e fácil para o usuário.

Figura 16 – Resquisção para a escrita no GPIO 16

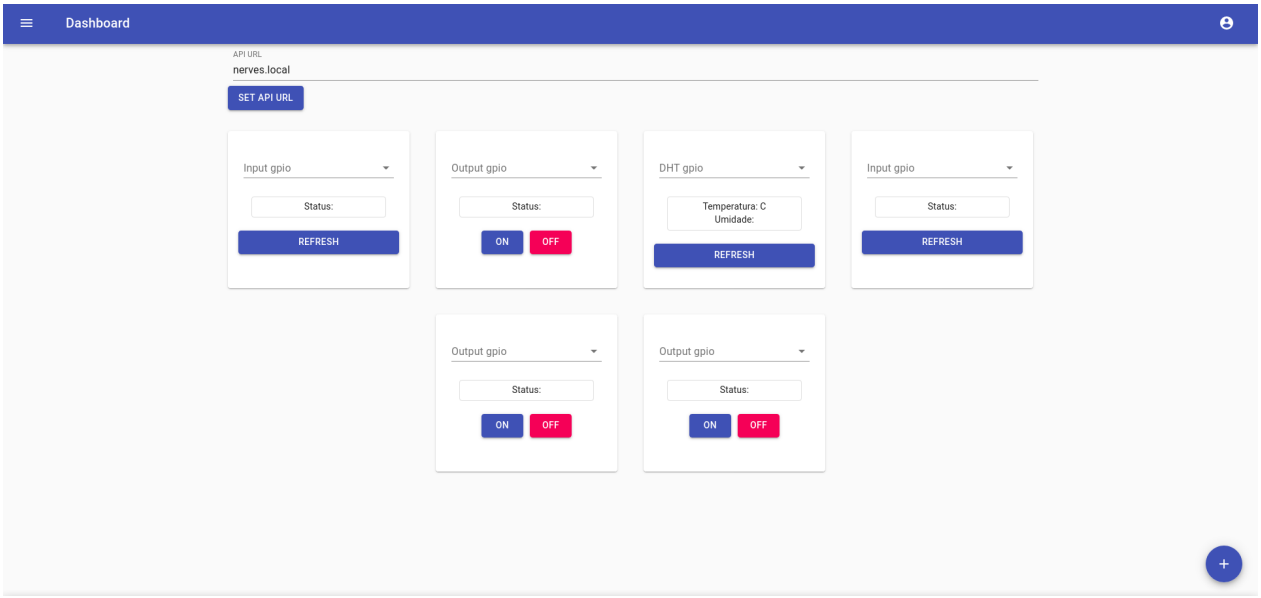
4.8 INTERFACE DE USUÁRIO

A interface do projeto tem como premissa apresentar as informações para o usuário de forma clara e objetiva. Sendo assim, criamos uma interface de tela única baseada em *cards*. O resultado final está apresentado na Figura 17.

Um *card* representa uma entidade renderizada na tela que tem por objetivo apresentar para o usuário uma informação. Temos três tipos de *cards*:

a) DHT: exibe as informações fornecidas pelo sensor de temperatura. O botão “REFRESH” força a atualização do valor exibido.
b) Entrada: exibe o estado lógico da entrada escolhida. O botão “REFRESH” força a atualização do valor exibido.
c) Saída: exibe o estado lógico e apresenta opções para ligar e desligar a saída escolhida nos botões “ON” e “OFF”.

Figura 17 – Interface de usuário para controle do hardware



O campo “API URL” deve ser preenchido com o endereço IP ou DNS do dispositivo na rede, para que possa acontecer a comunicação.

A interface de usuário para controle do hardware foi desenvolvida utilizando um *framework* de *front-end* chamado *React*, que usa uma abordagem modular para a composição de telas. Dentro do *framework* os módulos que representam as entidades a serem rederizadas na tela são chamados de *componentes*. Os *cards* foram construídos dentro dessa abstração de componentes, assim conseguimos compor eles na tela de forma dinâmica.

5 CONSIDERAÇÕES FINAIS

A plataforma desenvolvida teve como objetivo trazer os recursos de uma aplicação *IoT* que independe do nível técnico do usuário. Objetivamos alcançar pessoas com diferentes níveis de conhecimento técnico para assim difundir a tecnologia para a maior quantidade de indivíduos possível. Além disso, a plataforma possui baixo custo o que torna a tecnologia ainda mais palpável. Para atingir esse objetivo foi necessário desenvolver uma interface amigável e intuitiva.

A interface é composta por um painel com cartões que podem ser adicionados ou removidos o que traz versatilidade para a plataforma. Dessa forma, o usuário é capaz de configurar a plataforma de maneira dinâmica, fácil e intuitiva. Cada um desses cartões podem representar um GPIO do *hardware*, a criação de um cartão e atribuição do GPIO é de responsabilidade total do usuário, reforçando a versatilidade da plataforma. Além da escolha do GPIO o usuário pode configurar a direção dos pinos (saída ou entrada do hardware), nome do cartão e ainda o ícone de representação do cartão, como um botão, uma apresentação gráfica ou apenas o valor decimal lido pelo GPIO.

Durante o desenvolvimento da plataforma o maior obstáculo foi realizar os primeiros programas com a linguagem Elixir por ser uma linguagem de paradigma funcional diferente do método tradicional de programar. Entretanto, durante o desenvolvimento percebemos que foi uma ótima opção por ser uma linguagem de alto nível que não afeta a performance da aplicação e além de fornecer inúmeros recursos para acelerar o desenvolvimento de aplicações para sistemas embarcados.

Para a validação da plataforma realizamos testes de integração e funcional, visando mitigar possíveis falhas para o produto final. Para avaliar a interface foi necessário realizar o teste de usabilidade que foca na experiência do usuário portanto foi necessário a interação de usuários (com baixo nível técnico) com a plataforma a fim de comprovar a facilidade e a interatividade da interface. Todos os usuários conseguiram criar uma simples aplicação, sem grandes dificuldades, portanto a interface se mostrou intuitiva e amigável.

A plataforma desenvolvida atingiu o seu propósito inicial, porém para que se torne um produto final ainda são necessárias melhorias e desenvolvimento de novos recursos. Dentre esses podemos listar: a criação da autenticação do usuário com senha e *login* para restringir o acesso da plataforma apenas ao proprietário dos dispositivos; criar uma carcaça que identifica as conexões dos sensores de forma clara para evitar conexões incorretas e diminuir a exposição do *hardware*.

REFERÊNCIAS

ADAM, B. M.; RACHMAT ANOM BESARI, A.; BACHTIAR, M. M. Backend Server System Design Based on REST API for Cashless Payment System on Retail Community. In: 2019 International Electronics Symposium (IES). [S.l.: s.n.], 2019. P. 208–213. DOI: 10.1109/ELECSYM.2019.8901668.

BASICS, C. **BASICS OF THE I2C COMMUNICATION PROTOCOL**. Circuit Basics. Fev. 2016. Disponível em: <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/#:~:text=I2C%20is%20a%20serial%20communication,always%20controlled%20by%20the%20master.>. Acesso em: 29 set. 2020.

CHASE, J. **Evolution of Internet of Things**. Texas Instruments. Set. 2013. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0168169918316417>. Acesso em: 29 set. 2020.

CHEN, Q. L.; FENG, Y. H. The design and application of the online bookstore system .Net based on three tier architecture. In: 2015 10th International Conference on Information, Communications and Signal Processing (ICICS). [S.l.: s.n.], 2015. P. 1–3. DOI: 10.1109/ICICS.2015.7459824.

CODREY. **UART Communication Protocol – How it works?** Codrey. Jun. 2020. Disponível em: <https://www.codrey.com/embedded-systems/uart-serial-communication-rs232/>. Acesso em: 29 set. 2020.

COSTA, B. et al. Evaluating a Representational State Transfer (REST) Architecture: What is the Impact of REST in My Architecture? In: 2014 IEEE/IFIP Conference on Software Architecture. [S.l.: s.n.], 2014. P. 105–114. DOI: 10.1109/WICSA.2014.29.

CRISTONI, I. **Fundo espera levantar R$ 160 milhões para investir em IoT no Brasil**. Portal ef. Jun. 2020. Disponível em: <https://portalef.com.br/fundo-espera-levantar-r-160-milhoes-para-investir-em-iot-no-brasil/#:~:text=Com%20foco%20startups%20que%20desenvolvem,em%20est%C3%A1gio%20inicial%20de%20desenvolvimento.&text=No%20Brasil%2C%20%C3%A9%20estimado%20que,US%24%20200%20bilh%C3%B5es%20at%C3%A9%202025.>. Acesso em: 29 set. 2020.

EAMES, A. **Raspberry Pi 3 model B launches today**. RasPi.TV. Fev. 2016. Disponível em: <https://raspi.tv/2016/raspberry-pi-3-model-b-launches-today-64-bit-quad-a53-1-2-ghz-bcm2837>. Acesso em: 8 mai. 2020.

EMBARCADOS. **Sensores e Atuadores IoT**. Embarcados. Jan. 2020. Disponível em: <https://www.embarcados.com.br/sensores-e-atuadores-iot/>. Acesso em: 29 set. 2020.

FEDRECHESKI, G.; COSTA, L. C. P.; ZUFFO, M. K. Elixir programming language evaluation for IoT, p. 105–106, 2016. DOI: 10.1109/ISCE.2016.7797392.

GHUNTER. **Metodologia Agile, Scrum e Kanban**. Geek Hunter. Abr. 2018. Disponível em: <https://blog.geekhunter.com.br/tudo-que-voce-precisa-saber-sobre-agile-scrum-e-kanban/#Metodologia_Agile_foco_no_cliente_e_otimizacao_de_recursos>. Acesso em: 29 set. 2020.

GILMAR, C. **Learning IoT: First Steps with Elixir**. Medium. Set. 2018. Disponível em: <https://medium.com/@carlogilmar/learning-iot-first-steps-with-elixir-310c4ad4ab15>. Acesso em: 29 set. 2020.

IMASTERS. **Como funciona o sensor de luz BH1750**. iMasters. Jan. 2017. Disponível em: <https://imasters.com.br/desenvolvimento/como-funciona-o-sensor-de-luz-bh1750>. Acesso em: 29 set. 2020.

LLC, S. **Simio's 8 Reasons to Adopt Industry 4.0**. Cision PR Newswire. Abr. 2018. Disponível em: <https://mma.prnewswire.com/media/676382/Simio_LLC.jpg?p=publish&w=650>. Acesso em: 29 set. 2020.

MARR, B. **What is Industry 4.0? Here's A Super Easy Explanation For Anyone**. Forbes. Set. 2018. Disponível em: <https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/#70315c99788a>. Acesso em: 29 set. 2020.

MAXPHI. **DHT11 Humidity & Temperature Sensor Arduino Tutorial**. MaxPhi. Ago. 2017. Disponível em: <https://www.maxphi.com/temperature-humidity-sensor-dht11-arduino-tutorial>. Acesso em: 29 set. 2020.

MCCORD, C. **Phoenix documentation**. Hex. Out. 2020. Disponível em: <https://hexdocs.pm/phoenix/overview.html>. Acesso em: 30 out. 2020.

MDN. **What is JavaScript?** MDN Web Docs. Ago. 2020. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript>. Acesso em: 2 nov. 2020.

NEWS, R. **Total de dispositivos conectados à Internet das coisas deve atingir 75,44 bilhões até 2025**. Relevante News. Fev. 2020. Disponível em: <https://relevante.news/inova/total-de-dispositivos-conectados-a-internet-das-coisas-deve-atingir-7544-bilhoes-ate-2025/>. Acesso em: 29 set. 2020.

PATHIRANA, D. et al. WireMe - IoT development platform for everyone. In: 2017 Moratuwa Engineering Research Conference (MERCon). [S.l.: s.n.], 2017. P. 93–98. DOI: 10.1109/MERCon.2017.7980463.

REICHELT. **DEBO BH 1750 Developer Boards - Digital Light Sensor, BH1750**. reichelt. Disponível em: <https://cdn-reichelt.de/bilder/web/artikel_ws/A300/GY-302_1.jpg>. Acesso em: 29 set. 2020.

ROUSE, M. **internet of things (IoT)**. IoT Agenda. Mar. 2016. Disponível em: <https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>. Acesso em: 29 set. 2020.

ROUSE, M. **internet of things (IoT)**. IoT Agenda. Mar. 2016. Disponível em: <https://cdn.ttgtmedia.com/rms/onlineimages/iota-iot_system_desktop.png>. Acesso em: 29 set. 2020.

RUBIO, M. **Which companies are using Elixir, and why?** Erlang Solutions. Set. 2019. Disponível em: <https://www.erlang-solutions.com/blog/which-companies-are-using-elixir-and-why-mytopdogstatus.html>. Acesso em: 29 set. 2020.

SABURIDO, A. **Front End 101 for IoT Apps**. Alvaro Saburidos. Abr. 2018. Disponível em: <https://medium.com/@alvaro.saburido/front-end-101-for-iot-apps-24a725334af2>. Acesso em: 2 nov. 2020.

SAHRA SEDIGH, A. H. **Cyber-Physical Systems**. ScienceDirect. 2018. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780123965288000018>. Acesso em: 29 set. 2020.

SCHNECK, J. **Nerves documentation**. Hex. Out. 2020. Disponível em: <https://hexdocs.pm/nerves/getting-started.html>. Acesso em: 30 out. 2020.

WIKIPEDIA. **I2C**. Wikipedia. Out. 2019. Disponível em: <https://pt.wikipedia.org/wiki/I%C2%B2C>. Acesso em: 29 set. 2020.

ZHANG, Z. et al. IoT Security: Ongoing Challenges and Research Opportunities. In: 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications. [S.l.: s.n.], 2014. P. 230–234. DOI: 10.1109/SOCA.2014.58.