ISSN 2526-7574, Anais do V Encontro de Computação do Oeste Potiguar (ECOP), UFERSA, Pau dos Ferros/RN, v.6, 2022

Controle de Iluminação Através da Internet Utilizando as Tecnologias *ReactJS*, *Firebase* e ESP32

Allyson Nascimento, Carlos Sousa, Georgia Oliveira
Dept. de Ciência da Computação
UERN, Natal, Brasil
{allysonnascimento, carloslucena, georgiaoliveira}@alu.uern.br

Felipe Oliveira, Glaucia Campos
Dept. de Ciência da Computação
UERN, Natal, Brasil
{glauciamelissa, felipeoliveira}@uern.br

***Resumo*—O conceito de Internet das Coisas (IoT) está associado à conexão dos diferentes objetos da vida cotidiana à Internet, de modo inteligente e sensorial, permitindo ao indivíduo comunicação e interação com outras pessoas ou objetos. Embora bastante comentada, a IoT ainda não é uma realidade para muitos, devido ao custo dos dispositivos, assim como a complexidade das tecnologias envolvidas. Então, é preciso pensar em soluções que popularizem a IoT. Este trabalho propõe uma solução simples e de baixo custo para controlar remotamente uma ou mais lâmpadas, utilizando a placa de desenvolvimento Doit ESP32 Devkit V1 e as tecnologias *ReactJS* e *Firebase*. A solução foi desenvolvida no laboratório LUMEN, vinculado ao grupo de Pesquisa GSET/CNPQ da UERN - Campus de Natal. Ao final, conclui-se que, além de simples e de baixo custo, a solução também pode ser utilizada para outros dispositivos eletroeletrônicos de acionamento simples, como condicionadores de ar e ventiladores.**

Palavras-chave*—IoT, ESP32, *Firebase*, *ReactJS

I. Introdução

Em um mundo onde tudo e todos estão cada vez mais conectados, o conceito de Internet das Coisas (IoT - *Internet of Things*) está em constante crescimento e, com esta evolução, vem expandindo também as possibilidades de aplicações voltadas para automação residencial. A IoT é um conceito que integra diversas tecnologias para a conexão de uma "coisa" à Internet, dando a capacidade também de sensoriamento, atuação e controle desta "coisa" [1].

No mercado atual, já é possível encontrar objetos que podem ser remotamente controlados pela Internet, tais como lâmpadas, termostatos, geladeiras, entre outros, porém muitas vezes esses produtos não estão acessíveis para uma parcela da população devido ao seu alto custo [2]. Então, é preciso propor novas soluções, considerando tecnologias com preços mais acessíveis e facilidade de uso.

Este trabalho propõe uma solução simples e de baixo custo para controlar remotamente uma ou mais lâmpadas através do ESP32, via uma interface WEB, disponível em nuvem para o usuário, utilizando as tecnologias *ReactJS* e *Firebase*, sendo essa uma solução acessível para automação residencial. O protótipo foi desenvolvido no Laboratório de Sistemas Embarcados e de Tempo Real (LUMEN), pertencente à Universidade do Estado do Rio Grande do Norte (UERN) - Campus de Natal e vinculado ao grupo de pesquisa GSET (Grupo de Sistemas Embarcados e de Tempo Real)/CNPQ.

O sistema funciona da seguinte maneira: A placa ESP32 controla um relê, onde também está conectada uma lâmpada a ser acionada. Uma aplicação WEB baseada em *ReactJS* apresenta ao usuário dois botões: um para ligar e outro para desligar a lâmpada. Ao escolher um dos botões, a interface WEB envia o valor do botão para um banco de dados online com a tecnologia *Firebase*, que atualizará o seu estado e enviará a resposta, por WiFi para a placa ESP32 que, por sua vez, acionará o relê a depender do estado atual da variável de controle enviada pelo *Firebase*.

Este trabalho está organizado da seguinte maneira: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 descreve a metodologia; o desenvolvimento da solução está expresso na Seção 4 e; os resultados e a conclusão nas Seções 5 e 6.

II. Trabalhos relacionados

Santos e Lara Junior [3] desenvolveram um sistema de baixo custo para automação residencial, utilizando o microcontrolador ESP32 controlado por uma interface mobile através de um servidor WEB. Apesar do sistema acionar, além de lâmpadas, outros dispositivos eletroeletrônicos, como fechadura com trava elétrica e alarme residencial, o acesso a interface WEB se restringiu à rede local da residência, limitando os usuários de acessarem via, Internet, os dispositivos controlados pelo sistema.

Martins [4] desenvolveu uma solução de automação residencial usando o protocolo MQTT, através da proposta que utilizou a ferramenta *Node-RED* e o software *Mosquitto Broker*, além dos microcontroladores ESP32 e ESP8266. O sistema permitiu o acionamento remoto, via Internet, de um conjunto de LEDs conectados às placas microcontroladas, não refletindo a realidade da utilização de lâmpadas elétricas de alta tensão.

Araujo [5] desenvolveu um protótipo em maquete de uma residência automatizada, simulando o acionamento de lâmpadas e outros dispositivos eletroeletrônicos por meio do ESP8266. A programação foi realizada no Arduino IDE e a comunicação entre os dispositivos se deu por meio do protocolo MQTT. O *broker* utilizado foi o *CloudMQTT*, o qual pode ser acessado via WEB. Tal como descrito no trabalho de Martins [4], nota-se que esta implementação não promoveu o acionamento real de lâmpadas elétricas, não sendo possível refletir uma aplicação de automação residencial na prática.

III. METODOLOGIA

Esta seção apresenta as tecnologias e componentes eletrônicos que foram utilizados para a construção do projeto.

A. Materiais utilizados

A Tabela I apresenta a lista de componentes utilizados (não considerando o *notebook* destinado ao desenvolvimento do protótipo nem o dispositivo móvel empregado nos testes de acionamento da iluminação) e seus preços no mercado (média encontrada em mercados nacionais), convertidos para o dólar do dia. Nota-se que o custo do protótipo é bem acessível. Convertendo-se o valor em dólar para valores atuais (do período em que esse trabalho foi feito) o valor é de, aproximadamente, 112 reais.

Tabela I
LISTA DE COMPONENTES UTILIZADOS

MATERIAIS	QUANTIDADE	VALOR UNITÁRIO ($)
Placa microcontrolada DOIT ESP32 DevKit V1	1	6.70
Módulo relê 5v canal único	1	2.90
Fonte ajustável 5VCC e 3,3VCC para protoboard	1	2.00
Fonte de alimentação de 5VCC	1	2.90
Lâmpada LED 9W bivolt com soquete	1	1.90
Protoboard	2	2.50
Cabo USB	1	1.50
Jumpers	7	0.90
Arduino IDE	Software	0.00
Firebase	Software	0.00
ReactJS	Software	0.00
	Total	21.30

B. Ferramentas utilizadas

O DOIT ESP32 Devkit V1 é uma placa de desenvolvimento compacta criada pela empresa DOIT (*Doctors of Intelligence & Technology*) que embarca o microcontrolador ESP-WROOM-32, integrando suporte a Wi-Fi, *Bluetooth* e *Ethernet* em um único chip, com baixo consumo de energia [6]. A Fig. 1 mostra a placa ESP32 Devkit V1 e a Fig. 2 a sua pinagem.

O *Firebase* é um conjunto de serviços de hospedagem para qualquer tipo de aplicação (Android, iOS, Javascript, ReactJS, Java, Unity, PHP, C++...). Ele oferece hospedagem NoSQL e em tempo real de bancos de dados, conteúdo, autenticação social (Google, Facebook, Twitter e Github) e notificações ou serviços, como um servidor de comunicação em tempo real [7]. Na versão gratuita, é disponibilizado até 10GB por mês para atender a requisições do tipo *request/reply*. O Firebase foi utilizado como interface de conexão entre a aplicação WEB desenvolvida em *ReactJS* e o envio de dados, por WiFi para o ESP32, comandando o estado (ligado ou desligado) da lâmpada utilizada no protótipo de testes.

Fig. 1. Placa de desenvolvimento ESP32 (fonte: [6])

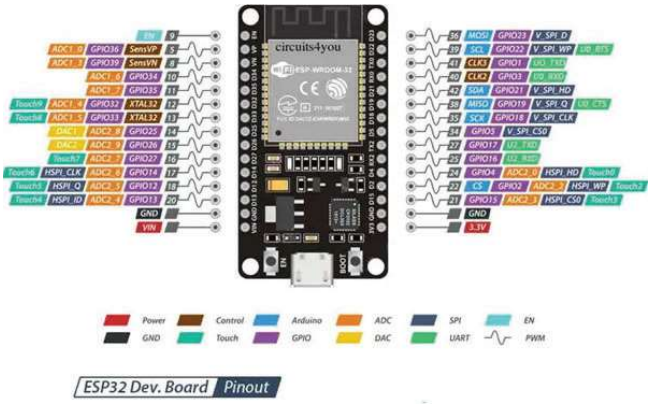

Fig. 2. Pinagem da ESP32 Dev Kit V1 (fonte: [6])

O *ReactJS* é uma biblioteca Javascript *front-end*, gratuita e de código aberto para construir interfaces de usuário baseadas em componentes. O código *React* é feito de entidades chamadas componentes. Esses componentes são reutilizáveis e podem ser renderizados para um elemento específico no DOM usando a biblioteca React DOM [8]. O *ReactJS* foi utilizado neste trabalho para servir de conexão entre a aplicação WEB do usuário e a requisição do estado do botão que comandará a iluminação feita no Firebase. Para a hospedagem e acesso a aplicação WEB pelo usuário foi utilizada a plataforma *front-end* gratuita Vercel [9].

C. Fluxograma da Aplicação

A Fig. 3 mostra o fluxograma base do projeto. O usuário deve, primeiramente, ter acesso a um dispositivo que possua conexão à Internet (passo 2); então, o usuário pode acessar o *browser* e seguir para o *link* onde a página que controla o estado da lâmpada de testes está hospedada (passo 3); uma vez aberta a página, o usuário pode escolher entre ligar ou desligar a iluminação, clicando no botão correspondente (passos 4, 5 e 6); após a escolha do usuário, uma requisição, montada através do *ReactJS*, é enviada ao servidor do *Firebase*, contendo um dos dois estados que o usuário escolheu: *"p=true"* indica que a lâmpada deve ser ligada ou; *"p=false"*, indica que ela deve ser desligada (passos 7 e 8); o *Firebase*, por sua vez, recebe

a requisição e se comunica, via rede Wi-Fi onde o ESP32 esteja conectado, fazendo com que o ESP32 se comunique com o módulo relê, ligando ou desligando a lâmpada de testes (passos 9 e 10). Vale salientar que o ESP32 consegue encontrar o servidor do *Firebase* através da biblioteca "IOXhop_FirebaseESP32.h"desenvolvida pela IOXHop [10].

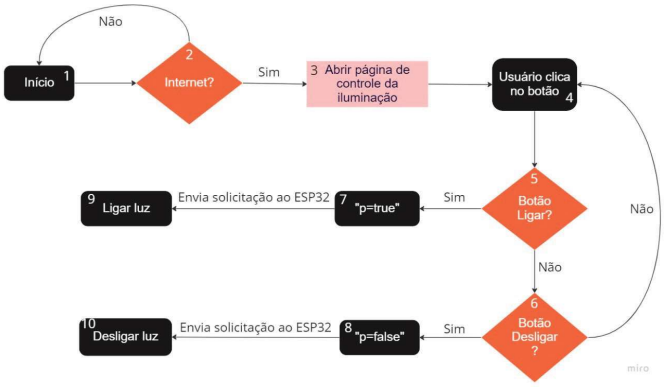

Fig. 3. Fluxograma base (Fonte: próprio autor)

D. Aplicação Web

A Fig. 4 mostra a aplicação WEB que o usuário tem acesso para ligar ou desligar a lâmpada. Ela foi concebida utilizando HTML5 e CSS.

Fig. 4. Aplicação WEB (Fonte: próprio autor)

E. Conexão do ReactJS com o Firebase

A Fig. 5 mostra como foi feita a conexão da aplicação em *ReactJS* com o *Firebase*. Inicialmente, foi criada, pelo *Firebase*, uma constante com todas as configurações necessárias para conexão com a apiKey (linha 5) e outras informações que são necessárias para que a conexão seja feita, tais como a autorização de uso do domínio que interconecta o ESP32 ao *Firebase* (linha 6) e a URL de acesso à requisição do *Firebase* (linha 7).

A Fig. 6 ilustra como é tratada a requisição dos botões de ligar e desligar entre o aplicativo WEB e o Firebase: inicialmente, na linha 9, é criada uma constante com uma função que basicamente faz com que, ao ser clicado o botão de desligar, o *ReactJS* envie a requisição para o *Firebase* com o valor de "OFF".

O mesmo procedimento descrito anteriormente ocorre quando o usuário clica no botão de ligar: o *ReactJS* envia a requisição ao *Firebase* do estatus "ON"(Fig. 7).

```
import { initializeApp } from 'firebase/app';
import { getDatabase } from 'firebase/database';

const config = {
    apiKey: "AIzaSyApQvtyR1sHeeDm5mTmuClOhTDuu8XsQjI",
    authDomain: "esp32-firebase-project-32b1b.firebaseapp.com",
    databaseURL: "https://esp32-firebase-project-32b1b-default-rtdb.firebaseio.com",
    projectId: "esp32-firebase-project-32b1b",
    storageBucket: "esp32-firebase-project-32b1b.appspot.com",
    messagingSenderId: "678460532956",
    appId: "1:678460532956:web:f3a2100755bc0005212c19"
};

export const app = initializeApp(config);
export const database = getDatabase(app);
```

Fig. 5. Código de conexão do *ReactJS* com o *Firebase* (fonte: próprio autor)

```
import React from 'react'
import  Button  from 'react-bootstrap/Button'
import './BotaoDanger.css'
import { database } from '../../firebaseConfig'
import { ref, set } from 'firebase/database'

function BotaoDanger() {

  const changeStatusToOff = () => {
    set(ref(database, '/'), {
      LED_STATUS: 'OFF'
    })
  }

  return (
    <div>
        <Button variant="danger" size="lg" id="btn-danger" onClick={() => changeStatusToOff()}>Desligar</Button>{' '}
    </div>
  )
}

export default BotaoDanger
```

Fig. 6. Código de conexão com o *firebase* do botão desligar (fonte: próprio autor)

A Fig. 8 mostra a configuração feita no *Firebase* do *status* do botão de ligar a lâmpada escolhido pelo usuário. No caso, "LED_STATUS='ON'", faz com que o *Firebase* envie a requisição ao ESP32, para que ele acione o módulo relê, acendendo a lâmpada. Segue-se o mesmo procedimento em relação ao botão de desligar ("LED_STATUS='OFF'").

F. Diagrama do circuito

A Fig. 9 mostra o diagrama do circuito, onde pode-se ver o ESP32 Devkit V1, o módulo relê de 1 canal, a fonte

```
import React from 'react'
import  Button  from 'react-bootstrap/Button'
import "./BotaoSucesso.css"
import { database } from '../../firebaseConfig'
import { ref, set } from 'firebase/database'

function BotaoSucesso() {

  const changeStatusToOn = () => {
    set(ref(database, '/'), {
      LED_STATUS: 'ON'
    })
  }

  return (
    <div>
        <Button variant="success" size="lg" id="btn-sucess" onClick={() => changeStatusToOn()}>Ligar</Button>{' '}
    </div>
  )
}

export default BotaoSucesso
```

Fig. 7. Código de conexão com o *firebase* do botão ligar (fonte: próprio autor)

Fig. 8. Requisição de ligar a lâmpada feita pelo *Firebase* ao ESP32 (fonte: próprio autor).

ajustável de 5VCC (necessária para alimentar o módulo relê, já que o ESP32 só fornece 3,3VCC) e a lâmpada, que trabalha em 220VCA. Ao receber o comando de acionamento da lâmpada, proveniente do *Firebase*, o ESP32, via GPIO 23, aciona o módulo relê, que comuta o contato "comum" e "normalmente aberto" para ligar a lâmpada. Caso o *Firebase* envie o comando de desligar a lâmpada, o ESP32 negativa a porta 23, desenergizando o relê fazendo, assim, com que a lâmpada seja desligada.

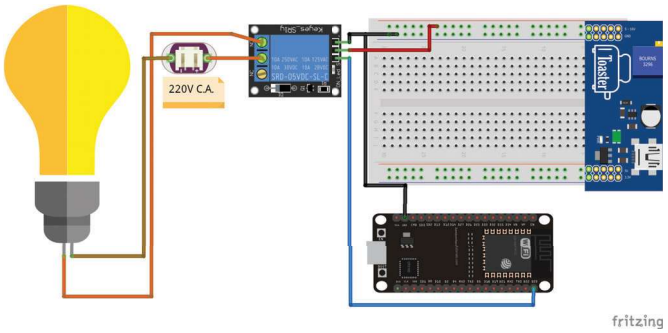

Fig. 9. Diagrama do circuito (fonte: próprio autor)

IV. Resultados

A Fig. 10 mostra a disposição da placa ESP32 Devkit V1 e do módulo da fonte ajustável de 5VCC nas protoboards utilizadas. Uma fonte externa de 5VCC alimenta o módulo da fonte ajustável e um cabo USB fornece a tensão de alimentação para a placa ESP32.

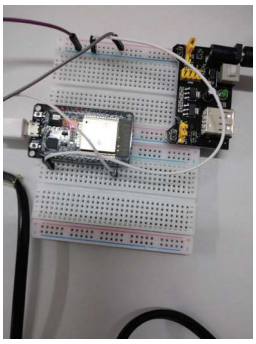

Fig. 10. ESP32 Dev Kit V1 e módulo da fonte ajustável conectados nas protoboards

A Fig. 11 mostra o circuito completo, onde se tem a visão geral do hardware descrito na seção IV(B), bem como a página WEB do sistema.

Foram realizados um total de 10 testes com a página carregada na mesma rede onde o ESP32 estava conectado (rede local) e 10 testes utilizando a Internet, com o equipamento de testes (*smartphone*) em rede diferente da rede local. Em cada teste, o botão de ligar e desligar a lâmpada foi acionado e verificado o tempo de resposta do circuito à ação imprimida ao botão. Ao final, foi realizada a média aritmética dos testes. O tempo de resposta do circuito em rede local foi de 3,12s

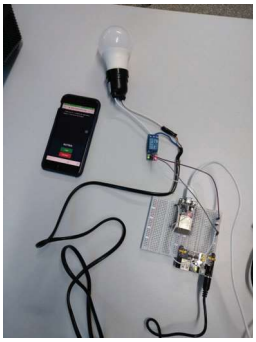

Fig. 11. Imagem do projeto físico

contra 3,67s quando conectado à Internet. A diferença de 0,55s é considerada aceitável para aplicações baseadas em IP, onde parâmetros de QoS são sempre variáveis [11]. Mais cargas podem ser controladas, bastando-se inserir um módulo Relê com mais canais, bem como o acréscimo dos botões de controle correspondentes na interface WEB.

V. Conclusão

Neste trabalho, foi desenvolvido um sistema para controle de iluminação residencial através da Internet, de custo acessível, onde foi utilizada a placa ESP32 Devkit V1, o *framework ReactJS* para desenvolver a aplicação WEB e o *Firebase* para a hospedagem em tempo real. O uso da aplicação de controle da iluminação pode ser feito tanto em computadores quanto em dispositivos móveis, permitindo que o usuário controle as lâmpadas em qualquer local onde exista conexão à Internet. Os testes de utilização demonstraram a viabilidade do protótipo e a possibilidade de aplicação em outros dispositivos eletroeletrônicos de acionamento simples.

Referências

[1] P. Carrion and M. Quaresma, "Internet da coisas (iot): Definições e aplicabilidade aos usuários finais," *Human Factors in Design*, vol. 8, pp. 049–066, Mar. 2019.

[2] G. A. R. M. Esmeraldo, "Um relato breve da evolução do computador, as mudanças e as principais tendências tecnológicas," in *Anais do III WCEA - Workshop de Ciências Exatas Aplicadas*, Crato (CE), Jun. 2020, p. 19.

[3] J. W. Santos and R. C. de Lara Júnior, "Sistema de automatização residencial de baixo custo controlado pelo microcontrolador esp32 e monitorado via smartphone," Universidade Tecnológica Federal do Paraná - Ponta Grossa, p. 46, 2019.

[4] V. F. Martins, "Automação residencial usando protocolo mqtt, node-red e mosquitto broker com esp32 e esp8266," Universidade Federal de Uberlândia, p. 53, 2019.

[5] J. F. de Araujo, "Automação residencial com objetos inteligentes: qualidade de vida e controle de energia," UniCEUB – Centro Universitário de Brasília, p. 19, 2020.

[6] *DOIT.AM Doctors of Intelligence & Technology Co. ESP-F WiFi module*, DOIT.AM, 2017.

[7] Google, "Firebase - an app development platform," https://firebase.google.com, [Acesso em Outubro de 2022].

[8] React, https://reactjs.org/, [Acesso em Outubro de 2022].

[9] Vercel, https://vercel.com/, [Acesso em Outubro de 2022].

[10] IOXhop, "Arduino esp8266," https://twitter.com/ioxhop, [Acesso em Outubro de 2022].

[11] M. Singh and G. Baranwal, "Quality of service (qos) in internet of things," in *3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Bhimtal, India, Feb. 2018, pp. 1–6.