


SISTEMA EMBARCADO PARA MONITORAMENTO DE SALA DE SERVIDORES

EMBEDDED SYSTEM FOR SERVER ROOM MONITORING

Lucas Batista dos Santos¹ 

Felipe Sella Lopes² 

Edimaldo Fialho Nunes de Oliveira³ 

Thiago Berticelli Ló⁴ 

Resumo: Este trabalho objetivou aplicar as tecnologias da Internet das Coisas (Internet of Things - IoT) na construção de um Sistema Embarcado (SE) para monitoramento do ambiente da sala de servidores do Instituto Federal do Paraná - Campus Cascavel remotamente, fazendo com que a detecção de problemas, como temperatura e umidade anormais e quedas de energia seja facilitada, proporcionando maior eficiência e velocidade na manutenção do ambiente. A utilização deste dispositivo visa auxiliar os profissionais de Tecnologia da Informação (TI) de qualquer organização possuidora de uma sala de servidores. Foi utilizado um dispositivo IoT (módulo ESP32) integrado ao aplicativo *Telegram*, o que permitiu a notificação instantânea de ocorrências, além da requisição de informações do ambiente a qualquer momento. Adicionalmente, as informações foram armazenadas a cada cinco minutos em um banco de dados na nuvem, a ferramenta *Cloud Firestore*, fornecida pela plataforma *Google Firebase*, o que possibilita consultas e análises posteriores para prevenção de problemas. O desenvolvimento do projeto se iniciou com o estudo dos elementos de hardware: o módulo ESP32, os sensores de temperatura e umidade e o módulo relé, bem como elementos de software, como a definição do protocolo de comunicação com o aplicativo *Telegram* (realizada por meio de um chatbot) e o armazenamento dos dados através da *Firestore*. Após os estudos, foi iniciada a construção do protótipo, realizando testes de funcionamento tanto do hardware quanto do software por meio de experimentos envolvendo temperatura e umidade executados em ambiente controlado. Por meio do protótipo foi possível um monitoramento do ambiente, identificando eventos como temperatura e umidade anormais e quedas de energia, sendo estes devidamente notificados aos usuários responsáveis. Além disso, as medições dos sensores e os eventos registrados foram armazenados corretamente na base de dados.

Palavras-chave: sistema embarcado. servidores. monitoramento. ESP32.

¹ Estudante do Curso Técnico em Informática, Instituto Federal do Paraná, batistalucas451@gmail.com.

² Estudante do Curso Técnico em Informática, Instituto Federal do Paraná, felipesellalopes@gmail.com.

³ Mestre em Matemática, Instituto Federal do Paraná, edimaldo.oliveira@ifpr.edu.br.

⁴ Doutor em Engenharia Agrícola e Mestre em Ciência da Computação, Instituto Federal do Paraná, thiago.lo@ifpr.edu.br.

Abstract: This work aimed to apply the technologies of the Internet of Things (IoT) in the construction of an Embedded System (SE) for monitoring the environment of the server room of the Federal Institute of Paraná - Campus Cascavel remotely, making the detection of problems such as abnormal temperature and humidity and power outages is facilitated, providing greater efficiency and speed in maintaining the environment. The use of this device is intended to help Information Technology (IT) professionals in any organization that has a server room. An IoT device (ESP32 module) integrated with the *Telegram* application was used, which allowed instant notification of occurrences, in addition to requesting information from the environment at any time. Additionally, the information was stored every five minutes in a cloud database, the *Cloud Firestore* tool, provided by the *Google Firebase* platform, which enables queries and subsequent analysis to prevent problems. The development of the project began with the study of the hardware elements: the ESP32 module, the temperature and humidity sensors and the relay module, as well as software elements, such as the definition of the communication protocol with the *Telegram* application (performed through of a chatbot) and data storage through *Firestore*. After the studies, the construction of the prototype was started, carrying out functional tests of both the hardware and the software through experiments involving temperature and humidity performed in a controlled environment. Through the prototype, it was possible to monitor the environment, identifying events such as abnormal temperature and humidity and power outages, which were duly notified to the responsible users. In addition, sensor measurements and recorded events were correctly stored in the database.

Keywords: embedded system. servers. monitoring. ESP32.

1 INTRODUÇÃO

Servidores são equipamentos eletrônicos capazes de armazenar dados, conectar computadores à uma rede, armazenar sistemas, entre outros. Como boa parte desses dispositivos necessita de condições específicas de temperatura e energia para fornecer o melhor desempenho, seu monitoramento é de extrema importância, seja em *data centers* de grande porte ou em salas com um único servidor. Por meio do monitoramento, é possível detectar ameaças físicas, humanas, ambientais e também anormalidades na climatização do ambiente. Assim, é possível não somente manter o desempenho e a disponibilidade da infraestrutura, mas também assegurar a integridade dos dados (RANGEL, 2020).

Em alguns casos é possível antecipar possíveis problemas que venham a prejudicar as máquinas, o que faz a utilização de um dispositivo para o monitoramento desse ambiente controlado ser essencial. O uso deste dispositivo pode proporcionar mais facilidade e rapidez na solução de eventuais problemas, auxiliando os profissionais de Tecnologia da Informação (TI) de organizações possuidoras de uma sala de servidores. Desse modo, foi realizada uma discussão com os técnicos de informática do IFPR - Câmpus Cascavel, e concebeu-se a ideia de criar um dispositivo capaz de monitorar tal ambiente de forma automática e remota.

Visto o que foi exposto anteriormente, este trabalho visou construir um protótipo de Sistema Embarcado (SE), utilizando o conceito de Internet das Coisas (*Internet of Things* - IoT), com sensores integrados, capaz de se comunicar por meio da internet, para monitorar o ambiente de uma sala de servidores remotamente. Desta forma, a detecção de problemas, como temperatura anormal e quedas de energia, seria facilitada, otimizando a manutenção do local.

De forma mais específica, o objetivo foi desenvolver um protótipo capaz de: monitorar temperatura e umidade; monitorar quedas de energia; disponibilizar acessos remotos as informações do ambiente; gerir cadastro para usuários que desejam receber alertas; enviar mensagens e alertas para os

usuários responsáveis informando os eventos ocorridos, como falta de energia e condições climáticas indesejáveis; armazenar histórico dos dados e eventos ocorridos.

2 DESENVOLVIMENTO

Para facilitar a compreensão da metodologia e desenvolvimento do projeto, foi feita uma separação entre fundamentação teórica e desenvolvimento do protótipo em si, como é colocado abaixo.

2.1 Fundamentação teórica

Posteriormente à discussão feita com o pessoal da TI, foram pensados e selecionados os elementos que iriam compor o sistema, tendo estes sido estudados para facilitar a construção do protótipo.

2.1.2 Sala de Servidores

Uma sala de servidores é o espaço físico que armazena os principais equipamentos responsáveis pelo processamento e distribuição dos dados, que trafegam pela rede de computadores de uma empresa ou organização. Nela, os profissionais de TI passam muito tempo resolvendo problemas de rede ou nos servidores, bem como realizando manutenções de rotina. É prudente que essas salas sejam monitoradas 24 horas por dia para identificar anormalidades, como aumento de temperatura e umidade. Estes fatores são de extrema importância, já que as salas precisam se manter frias e secas para evitar o superaquecimento dos equipamentos. Montar um ambiente de processamento e armazenamento de dados seguro é essencial para se ter uma base sólida para a infraestrutura das operações de TI. (Como Projetar uma Sala de Servidores, 2019).

2.1.3 Sistemas Embarcados

Segundo Cunha (2007), o conceito de Sistema Embarcado (SE) constitui-se de agregar capacidade computacional dentro de um circuito integrado, equipamento ou sistema. O sistema formado, normalmente, é mais do que um simples computador. É um sistema completo e independente, mas

preparado para realizar apenas determinadas tarefas. Normalmente, o usuário final não tem acesso a esse programa que foi embarcado no dispositivo, mas pode interagir com o equipamento através de interfaces como teclados, displays, etc, desde que o sistema embarcado seja projetado para comportar tal interação.

Nos últimos anos a utilização desses sistemas cresceu consideravelmente, e cada vez mais dispositivos possuem SEs integrados em seus circuitos, o que gera uma maior necessidade de desenvolvedores desse ramo da tecnologia. As corporações também têm aumentado o uso dessas ferramentas em sua infraestrutura, tendo como retorno uma estrutura operacional mais eficiente, agregando em si dispositivos inteligentes que executam tarefas de forma independente, diminuindo assim o desgaste gerado pela atenção dos funcionários para com as máquinas (BORTOLUCCI; CARVALHO; MORAES; PEREIRA, 2014).

2.1.4 Módulo ESP32

O módulo ESP32 é composto de um microcontrolador desenvolvido pela empresa chinesa Espressif. Ele opera em 3,3V e possui um processador *dual-core* Tensilica L108 de 32 bits, com frequência de 160MHz. O processador também possui 520KB de memória RAM e 34 pinos de propósito geral (GPIO), além de suporte para Bluetooth e Wi-Fi. Por apresentar todas estas características, este módulo vem ganhando destaque na utilização de aplicações de IoT (KOLBAN, 2018).

2.1.5 Módulos Relé

Segundo Netto (2008), os relés são capazes de facilitar a resolução de problemas voltados para manutenção, comunicação, proteção e medição de Sistemas Elétricos de Potência (SEP). Os SEP estão sempre expostos a alguns fenômenos como uma queda de energia, curto-circuito, uma carga maior que o normal, tais fenômenos são capazes de danificar os SEP. Neste contexto, com o auxílio de um relé, a fonte de energia pode ser monitorada e

cortada caso necessário. Ainda, é possível automatizar o controle da tensão integrando uma placa de microcontrolador ao módulo relé. Um exemplo seria o desligamento de uma máquina caso um sensor de temperatura ou umidade indique que as condições do ambiente possam prejudicar a operação de máquina ou sistema eletrônico.

2.1.6 Sensores de temperatura e umidade DHT11 e DS18B20

Segundo Oliveira (2017), o sensor DHT11 é utilizado para mensurar a temperatura e umidade do ar. Ele possui internamente um pequeno microcontrolador de 8 bits que transmite as informações por meio de um protocolo específico, utilizando apenas um pino para dados. Por causa dessa característica, esse sensor torna-se muito interessante, pois, além de ser preciso, apenas necessita de uma entrada digital do microprocessador. A plataforma Arduino possui bibliotecas específicas para a leitura desse sensor, por meio das funções *readTemperature* e *readHumidity*.

Segundo Locatelli (2021), o sensor DS18B20 é um sensor digital de temperatura e umidade capaz de realizar medições precisas em ambientes secos, úmidos e também em ambientes submersos, suas medidas podem ser apresentadas em graus Celsius e as margens de erro podem ser ajustadas por incremento. Com seu protocolo *One Wire*, o sensor possui apenas um fio para transmissão de dados, o que permite que até 127 sensores ocupem o mesmo barramento. Por último, possui previamente um alarme programável, podendo estabelecer que o alarme dispare somente na ocorrência da temperatura se encontrar fora de uma determinada faixa programada no sensor.

2.1.7 Telegram

O *Telegram* é um aplicativo de troca de mensagens que possibilita a criação de um bot que pode ser usado para diversas aplicações. Um dos possíveis meios para integrar essa funcionalidade é em uma plataforma embarcada como o módulo ESP32. Para isso, é necessário configurar um perfil-bot por meio de uma plataforma conhecida como *BotFather*. Este permite

a criação de perfis únicos e exclusivos para as mais diversificadas aplicações. Ao configurarmos o perfil, um *token* de autenticação é gerado. Desta forma, este *token* pode ser utilizado nas rotinas de programação do microcontrolador, possibilitando que o módulo ESP32 receba as mensagens enviadas a este perfil, podendo interpretá-las e executar comandos correspondentes (Bot *Telegram* com ESP8266 NodeMCU, 2019).

2.1.8 Cloud Firestore

O *Cloud Firestore* é um banco de dados hospedado na nuvem, flexível e escalonável fornecido pela plataforma *Google Firebase*, ideal para desenvolvimento focado em dispositivos móveis, *Web* e servidores. Ele mantém os dados em sincronia em aplicativos-cliente usando *listeners* em tempo real, e oferece suporte off-line para dispositivos móveis e *Web*. Tudo isso torna possível a criação de aplicativos responsivos que funcionem independentemente da latência da rede ou da conectividade com a Internet (*Cloud Firestore*, 2022).

2.2 Desenvolvimento do Projeto

A Figura 1 apresenta uma visão geral do ambiente do projeto, no qual o controlador monitora a temperatura, umidade e rede elétrica da sala de equipamentos e notificando (via *Telegram*) os técnicos responsáveis caso as medidas ultrapassem os intervalos predefinidos no sistema.

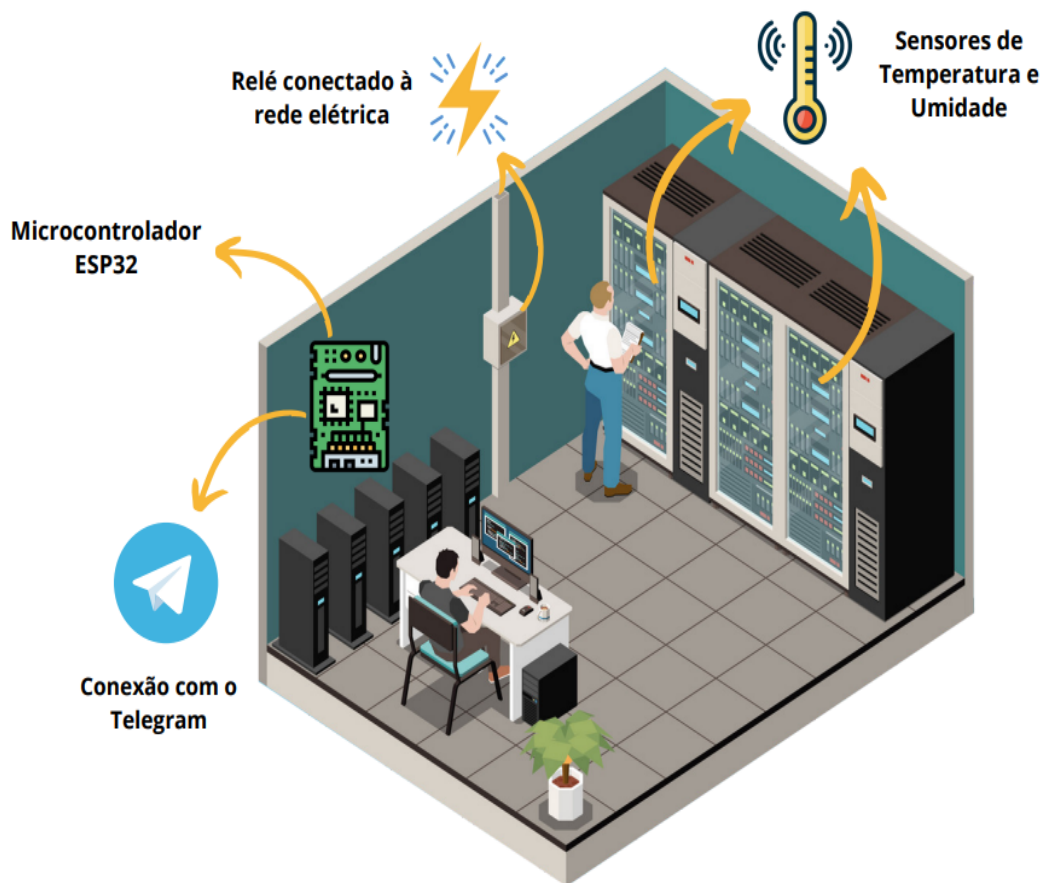
Foi realizado um estudo IoT, e suas aplicações práticas em SE. Então foram feitas análises sobre os sensores usados, o sensor de temperatura e umidade DHT11, o sensor de temperatura DS18B20 e o módulo relé, destinado a controle de corrente (chaveamento). Também foi estudado sobre a criação de Bots do *Telegram* e como podem ser integrados ao módulo ESP32.

Após este passo, realizou-se uma análise mais detalhada do módulo ESP32, explorando suas particularidades e funcionamento como: pinagem, níveis elétricos, protocolos de comunicação, tipos de dados, bibliotecas de software necessárias. Após isso, foi dedicado tempo ao estudo sobre a forma

que os dados seriam transmitidos e armazenados na plataforma *Cloud Firestore*.

Utilizando-se da *Cloud Firestore*, projetou-se o armazenamento dos IDs dos usuários, os quais são requisitados por meio de ações de cadastro no perfil-bot. Desta forma, somente são notificados aqueles usuários que são cadastrados, ou seja, seu ID encontra-se armazenado. Também visou-se fazer um *log* das medições feitas pelos sensores e dos eventos anormais ocorridos.

Figura 1 - Exemplificação da estrutura do sistema no contexto de uma sala de servidores.



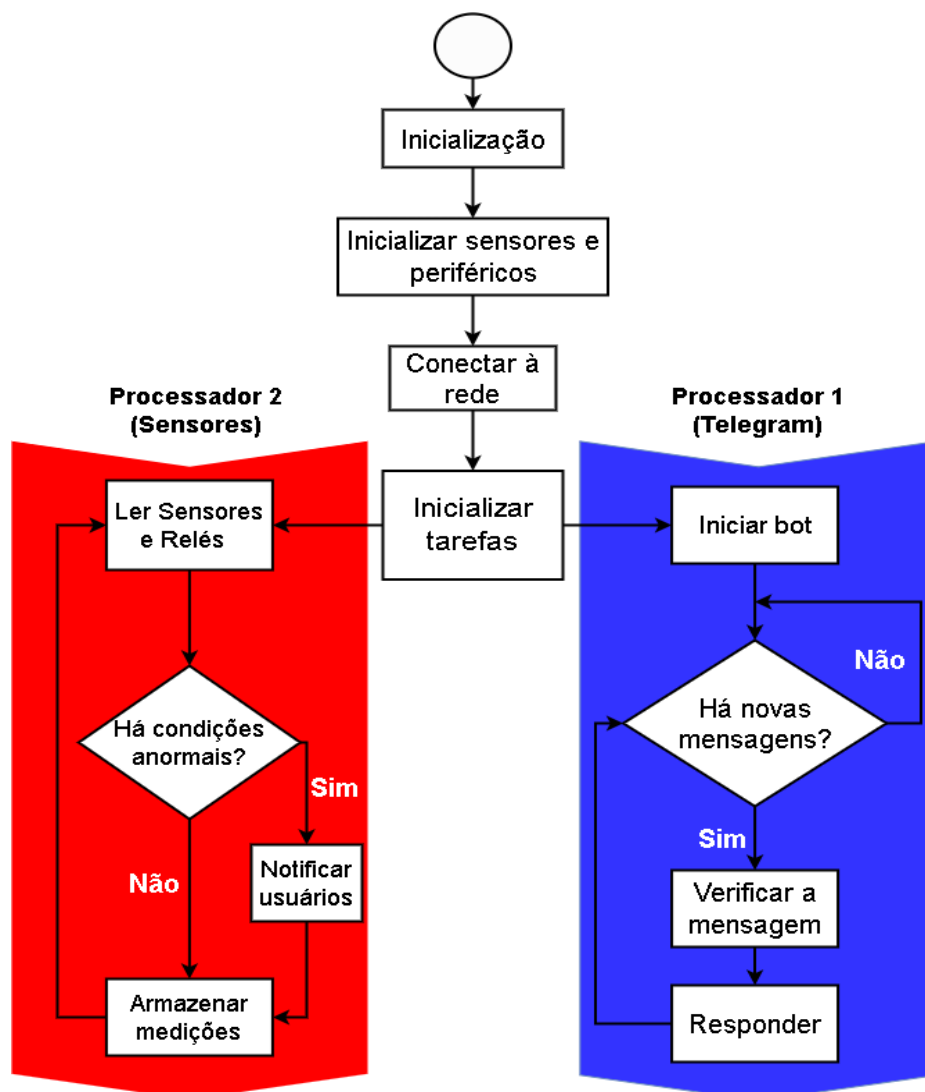
Fonte: Dos autores.

Assim, foi estabelecido uma estrutura para o funcionamento do sistema: ao ligar o dispositivo, os sensores e periféricos conectados são inicializados, então o ESP32 é conectado à rede wifi. Após isso, a conexão do módulo ao

Telegram é estabelecida e o *ChatBot* assume um modo *standby*, enviando informações apenas se o usuário requisitar. O outro núcleo fica responsável pela rotina das leituras dos sensores, dos relés e armazenamento dos dados na *Firestore*. Caso haja condições anormais identificadas pela verificação das leituras, é feita a notificação dos usuários cadastrados por meio do *Telegram*, com alertas diferentes de acordo com a situação ocorrida. Além disso, também serão enviados alertas caso a situação tenha sido normalizada, como por exemplo caso haja o restabelecimento do suprimento de energia.

Um diagrama de fluxo foi construído para demonstrar o comportamento de forma simplificada (Figura 2).

Figura 2 - Diagrama de fluxo exemplificando o funcionamento do sistema.



Fonte: Dos autores.

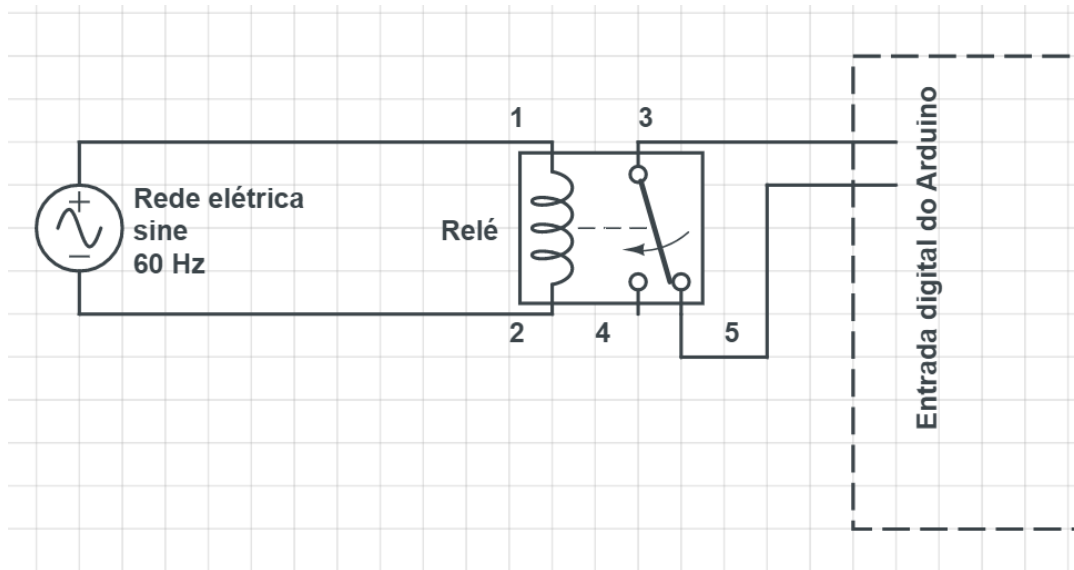
Na etapa seguinte foi iniciada a montagem do hardware, no qual foi usada uma protoboard de 830 pontos, um módulo ESP32 WROOM, um sensor DS18B20, um sensor DHT11, jumpers macho-macho e macho-fêmea e um botão, simulando o funcionamento de um relé para detectar tensão elétrica.

Posteriormente, foi adicionado o relé, o qual serve como sensor de tensão. Caso falte tensão, o relé comuta, indicando a falha de energia (Figura 3).

Montado o hardware, passou-se para a fase de desenvolvimento do software, onde foram realizadas as seguintes ações: definição de variáveis e

constantes; inclusão de bibliotecas; conexão com a *Firestore* e WIFI; definição de núcleos e de suas funções; definição das rotinas de tempo para as leituras dos sensores; criação das funções de leitura, validação e envio de mensagens; simulação de um relé usando o botão, estando este apertado significa que há tensão e estando solto significa que não há tensão.

Figura 3 - Esquema do funcionamento de um relé

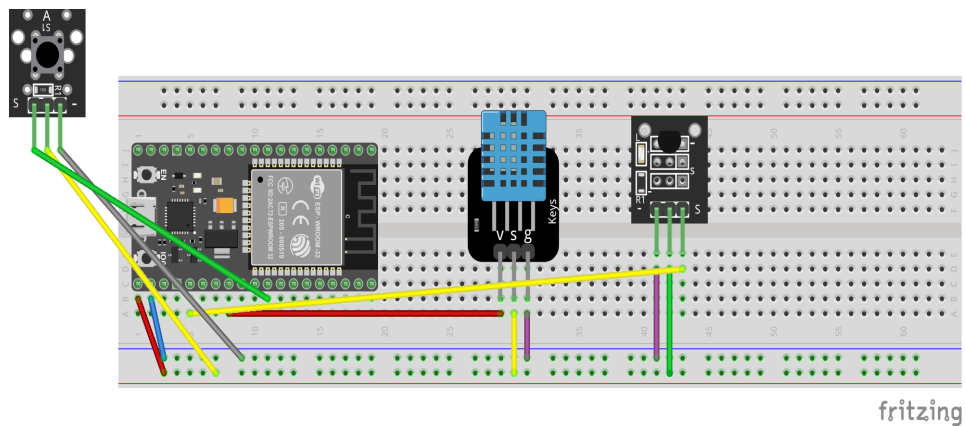


Fonte: Dos autores.

2.3 Resultados e Discussões

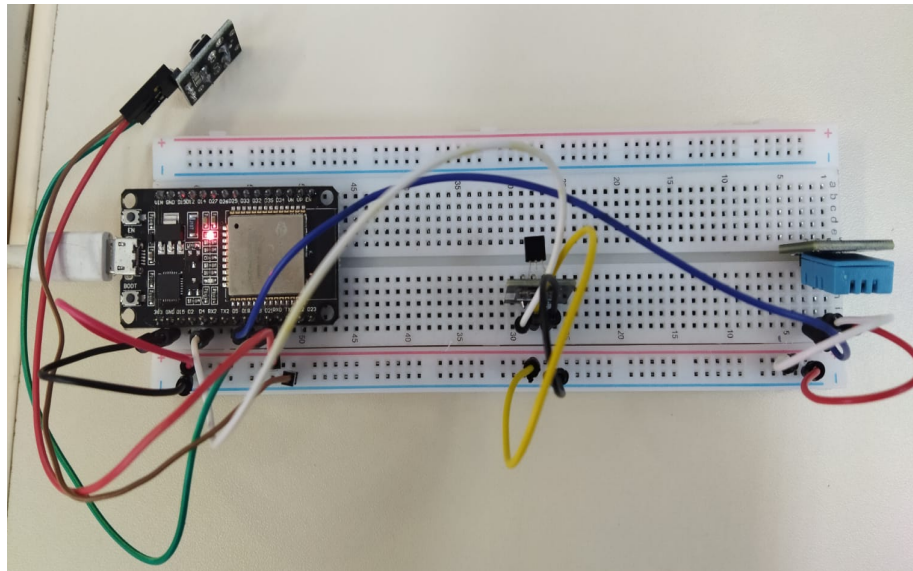
Na Figura 4 está sendo apresentado o projeto do circuito, feito em meio digital. Da esquerda para a direita, estão os seguintes componentes: Botão (simulando o relé), Placa ESP-WROOM-32, sensor DHT11 e sensor DS18B20. Na Figura 5 está sendo apresentado o protótipo montado, Da esquerda para a direita, estão os seguintes componentes: Botão (simulando o relé), Placa ESP-WROOM-32, sensor DS18B20 e sensor DHT11.

Figura 4 - Circuito de Hardware desenvolvido



Fonte: Dos autores.

Figura 5 - Protótipo montado.



Fonte: Dos autores.

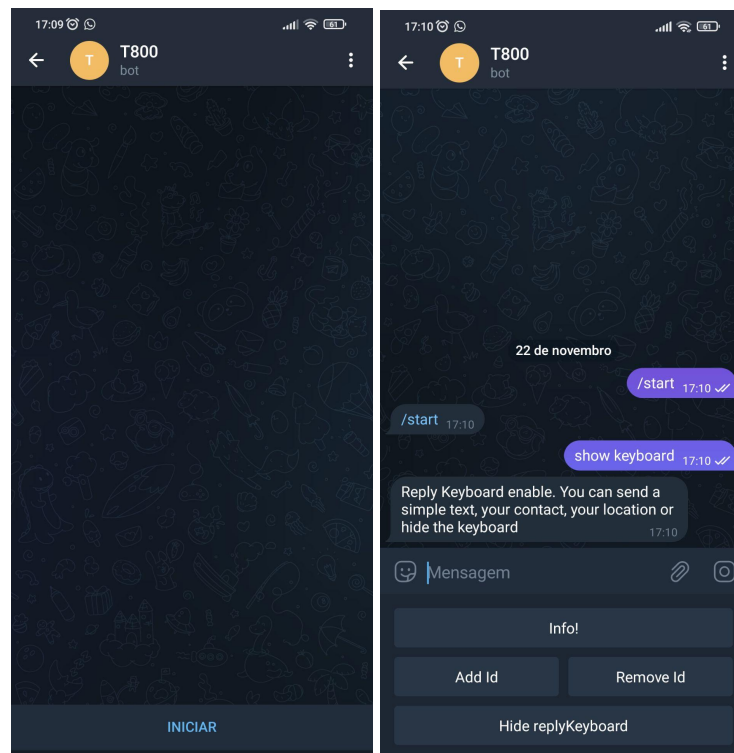
Conectados os componentes ao ESP32 e verificado seu funcionamento, procedeu-se para a realização de testes com o objetivo de verificar as leituras dos sensores e do botão. Então foram gerados logs utilizando o monitor serial da IDE do Arduino para esta validação. Com esta abordagem foi possível observar que os sensores obtiveram as leituras precisas e que o sistema consegue detectar corretamente se há ou não tensão. Foi estabelecido como intervalo entre cada coleta de informações dos sensores o período de 10 segundos, o que vem a proporcionar cerca de 8640 leituras por dia.

Foi verificado o envio de mensagens aos usuários, tendo como resultado as leituras sendo enviadas para o chatbot de acordo com as requisições. Também foi feita uma função com objetivo de enviar automaticamente mensagens notificando os usuários cadastrados sobre variações indesejadas nas medições, função essa que também foi testada e validada.

Foi verificada a função de envio de dados para a *Cloud Firestore*, os quais foram enviados com sucesso. Entretanto, foram encontradas dificuldades envolvendo o armazenamento dos dados na *Firestore*, especificamente uma falha na requisição de envio. O problema foi causado pois a função responsável por enviar os dados para a nuvem estava precedendo a função encarregada de conectar o protótipo à internet, causando a incapacidade de armazenar os dados na base. O problema foi resolvido reposicionando a função de armazenamento após a função de conexão WIFI.

A Figura 6 apresenta a comunicação com o aplicativo *Telegram* para realizar o cadastro dos usuários que receberão os alertas.

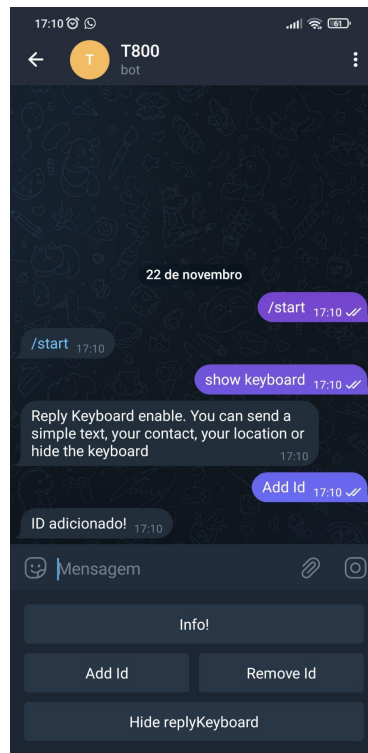
Figura 6 - Inicialização do bot pelo usuário.



Fonte: Dos autores.

Na Figura 7 é observada a solicitação para adicionar o ID do usuário, com sua respectiva resposta. Desta forma, quando ocorrer qualquer evento, este usuário será notificado imediatamente.

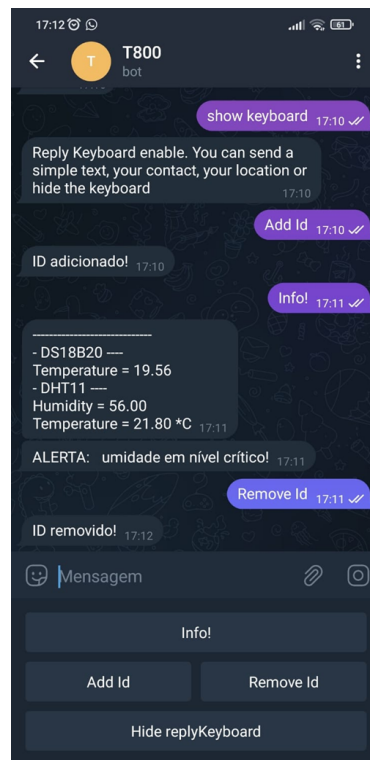
Figura 7 - Cadastramento do usuário pelo bot.



Fonte: Dos autores.

Na Figura 8 é possível observar solicitação e resposta, por meio do *Telegram*, de uma requisição de monitoramento do ambiente (valores dos sensores de temperatura e umidade), juntamente com um alerta que indica que o ambiente apresenta condições desfavoráveis. Nesta mesma figura, ao final, observa-se a solicitação da retirada do ID por parte do usuário e a confirmação da remoção.

Figura 8 - Interação usuário-bot, requisição de informações, alerta automatizado e retirada do ID.



Fonte: Dos autores.

A base de dados da *Firestore* foi dividida três coleções: uma reservada para o armazenamento de leituras, chamada de “LOG_Medicoes”, uma para o log de eventos, chamada de “LOG_Alertas”, e outra para o armazenamento de IDs de usuários, chamada “Dados”.

Dentro da coleção “LOG_Medicoes” são criados documentos periodicamente, tendo como identificação um carimbo de tempo (Epoch), e tem como dados as leituras dos sensores e uma timestamp legível.

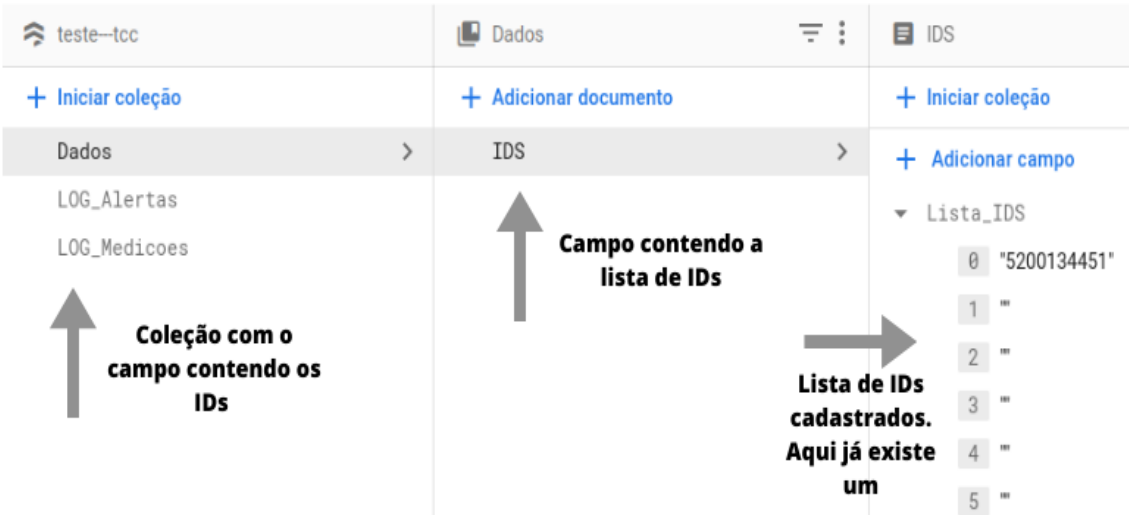
Dentro da coleção “Dados” existe um documento chamado IDS, cujo interior consiste em uma lista de 15 campos, cada um numerado de 0 a 14, contendo os IDs dos usuários cadastrados.

Na coleção “LOG_Alertas” ocorre o mesmo que em “LOG_Medicoes”, porém somente é armazenado registros de eventos, juntamente com o tipo de evento ocorrido, data e hora em que ocorreu.

Na Figura 9 é possível observar que a *Firestore* armazenou os IDs dos usuários após a solicitação de cadastro pelo *Telegram*, enquanto na Figura 10,

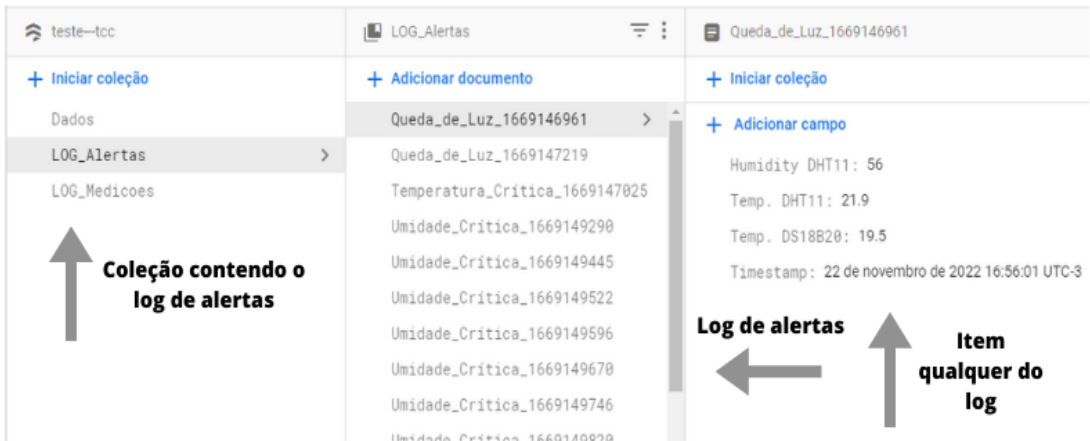
é mostrado que a *Firestore* armazena um histórico de eventos ocorridos, como temperatura e umidade fora do intervalo estabelecido e queda de tensão. Por fim, na Figura 11 observa-se que a *Firestore* armazena um log de medições.

Figura 9 - Armazenamento dos IDs na *Firestore*.



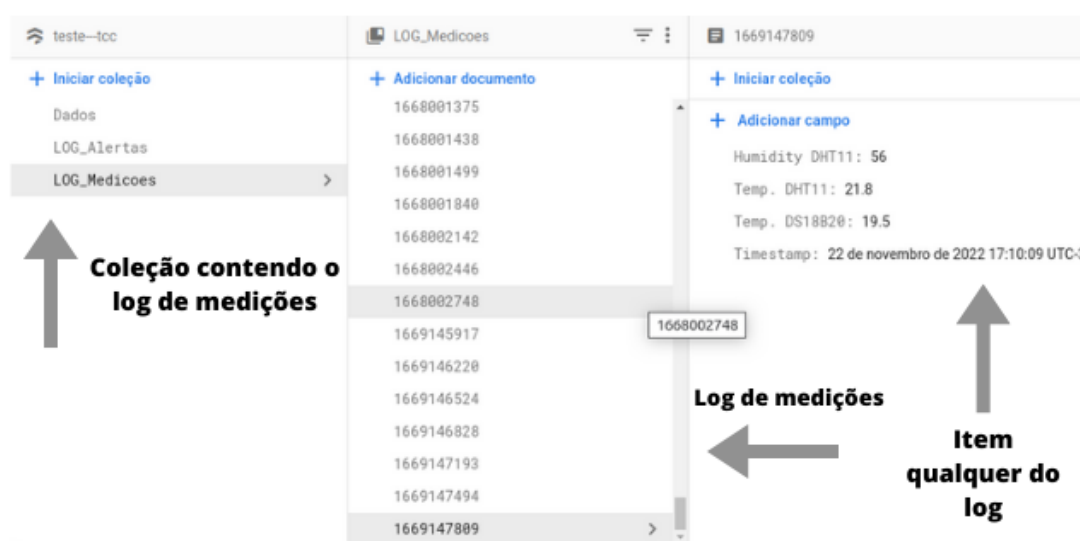
Fonte: Dos autores.

Figura 10 - Armazenamento dos eventos na *Firestore*.



Fonte: Dos autores.

Figura 11 - Armazenamento das medições na *Firestore*.



Fonte: Dos autores.

A cada adição de componente ao protótipo (sensores, *Telegram*, *Firestore*) foram feitos os testes correspondentes. Ao finalizar a construção, deixou-se o dispositivo ligado por cerca de 2 horas, em três dias diferentes, dentro do laboratório de informática do IFPR, o que permitiu ter a certeza do correto funcionamento do sistema como um todo.

Estudando-se o preço dos componentes e realizando orçamentos, chegou-se a conclusão que o custo do protótipo contendo os itens supracitados estaria na faixa de 120 reais, tomando como base o orçamento com valor mais baixo.

Realizou-se uma pesquisa sobre sistemas semelhantes ao construído, e chegou-se à conclusão de que existem trabalhos acadêmicos com os mesmos objetivos (monitoramento de sala de servidores) que se utilizam de componentes análogos, tais como placas Arduino. No mercado, existe o dispositivo “NetBotz 200”, da fabricante americana APC⁵, que também se propõe a monitorar ambientes de TI (o que inclui salas de servidores) de forma remota e automática. Entretanto, este produto possui um preço de cerca de cinco vezes o custo do protótipo construído neste projeto (Tabela 1).

⁵ <https://www.apc.com/br/pt/>

Tabela 1 - Comparação do protótipo com outro dispositivo semelhante

Protótipo Construído	APC Netbotz 200
Custo: R\$: 120,00 Vantagens: Baixo custo, econômico, fácil adaptação e atualização, interface simples Desvantagens: Menor qualidade, depende de App de terceiros	Custo: R\$: 800,00 Vantagens: Qualidade maior, maior notoriedade, mais atualizado, aceita sensor sem-fio, usa API Web Desvantagens: Alto Custo, maior espaço ocupado.

Fonte: Dos autores

3 CONSIDERAÇÕES FINAIS

Com o protótipo desenvolvido foi possível monitorar temperatura, umidade e quedas de energia corretamente. Foram validadas as rotinas de gerenciamento dos usuários, contemplando o cadastro e a remoção de seus identificadores no banco de dados *Firestore*. Além disso, foi possível estabelecer conexões entre o protótipo e múltiplos dispositivos, o que possibilitou também a execução adequada dos alertas simultâneos para vários usuários cadastrados.

A funcionalidade de solicitação das informações do ambiente em tempo real, por meio do perfil-bot do *Telegram*, também foi validada. Além disso, os alertas de condições anormais foram enviados apropriadamente para os usuários cadastrados previamente no sistema, sendo estas notificações diferenciadas de acordo com o tipo de evento ocorrido.

O registro de medições e de alertas foi armazenado corretamente na *Firestore*, apresentando os carimbos de tempo e as medições correspondentes. De forma conclusiva, foi possível construir o SE alcançando todos os objetivos do projeto, reiterando que os componentes estabelecidos em sua totalidade e a programação do protótipo funcionaram de maneira adequada.

Este projeto foi desenvolvido inicialmente para uso no *campus* do IFPR de Cascavel, porém apresenta possibilidade de uso em todos os 27 campi do IFPR, podendo ainda ser implementado em toda e qualquer instituição possuidora de uma sala de servidores.

O protótipo apresenta possibilidade de ampliação de funcionalidades, como atuador, controlando outros dispositivos como ar-condicionado, exaustores de ar, climatizadores de ambientes, entre outros, possibilitando assim um serviço automático de manutenção da temperatura e umidade da sala de servidores e expandindo as possíveis aplicações do sistema. Pode-se também construir um *case* para o protótipo, visando torná-lo um produto comercial e mais robusto.

Como sugestões para trabalhos futuros: Poderiam ser desenvolvidos outros aparelhos, como um *nobreak* específico para o dispositivo; Poderiam também ser adicionados sensores diferentes ao protótipo, por exemplo, sensores de fumaça e fogo; Em adição, pode ser implementada uma atualização de código, onde seria desenvolvido um procedimento que armazene os dados temporariamente, caso haja indisponibilidade de conexão, para depois enviá-los para a nuvem (mecanismo de contingência).

REFERÊNCIAS

BORTOLUCCI, Thatiane; CARVALHO, Francine Roberta; MORAES, Marcelo Henrique; PEREIRA, Luiz Arthur Malta. **SOFTWARE EMBARCADO, O CRESCIMENTO E AS NOVAS TENDÊNCIAS DESTE MERCADO**. REVISTA DE CIÊNCIAS EXATAS E TECNOLOGIA, Leme, v. 6, n.6, p. 85-94, 28 abr. 2014.

Bot Telegram com ESP8266 NodeMCU. abr. 2019. Disponível em: <https://www.arduinoecia.com.br/bot-telegram-com-esp8266-nodemcu/>. Acesso em: 18 mai. 2022.

Cloud Firestore. ago. 2022. Disponível em: <https://firebase.google.com/docs/firestore>. Acesso em: 19 ago. 2022.

Como Projetar uma Sala de Servidores. jun. 2019. Disponível em: <https://blog.s4t.com.br/2019/06/como-projetar-uma-sala-de-servidores.html>. Acesso em: 13 abr. 2022.

CUNHA, A. F. **O que são sistemas embarcados**. [S.l.]: Saber Eletrônica, 2007. v. 43.

KOLBAN, Neil. **Kolban's book on ESP32**. Editora Leanpub, 2018.

LOCATELLI, Caroline. **Como usar o Sensor de Temperatura – DS18B20**. jun. 2021. Disponível em: <https://www.curtocircuito.com.br/blog/Categoria%20Arduino/como-utilizar-o-ds18b20>.

Acesso em: 27 jun. 2022.

NETTO, Ulisses Chemin. **Aplicações de Controle e Supervisão Distribuídas em Subestações de Energia Elétrica Através do Uso de Relés Digitais de Proteção**. Orientador: Prof. Tit. Denis Vinicius Coury. 2008. 172 f. Dissertação (Mestrado em Engenharia Elétrica) – Escola de Engenharia, Universidade de São Paulo, São Carlos, 2008.

OLIVEIRA, Sérgio de. **Internet das Coisas com ESP8266, Arduino e Raspberry Pi**. São Paulo: Editora Novatec, 2017.

RANGEL, Rafael. **Monitoramento de Datacenter e Salas de TI: o que você precisa saber sobre o assunto**. abr. 2020. Disponível em: <https://xtech.com.br/Blog/Monitoramento-De-Datacenter-E-Salas-De-Ti-O-Que-Voce-Precisa-Saber-Sobre-O-Assunto/b/68/>. Acesso em 13 dez. 2022.