Universidade Federal de Campina Grande - UFCG
Centro de Engenharia Elétrica e Informática - CEEI
Unidade Academica de Engenharia Elétrica - UAEE

Matheus Vilarim Pereira dos Santos

Relatório de Estágio Supervisionado

Campina Grande, Brasil
28 de outubro de 2021

Matheus Vilarim Pereira dos Santos

Relatório de Estágio Supervisionado

Relatório de Estágio Supervisionado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Edmar Candeia Gurjão
Supervisor: Valber Aragão

Campina Grande, Brasil
28 de outubro de 2021

Matheus Vilarim Pereira dos Santos

Relatório de Estágio Supervisionado

Relatório de Estágio Supervisionado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovado em: 28/10/2021

Prof. Dr. Edmar Candeia Gurjão
Orientador

Profa. Dra. Luciana Ribeiro Veloso
Avaliador

Campina Grande, Brasil
28 de outubro de 2021

Dedico esse trabalho àqueles que me deram tudo, meus pais.

Agradecimentos

Agradeço a Deus por ter me dado forças pra chegar até aqui. Agradeço aos meus pais, Veridenes Vilarim e Moisés Vilarim, por terem sido exemplos de amor, força e determinação. Muito do que sou é por causa de vocês. Assim, os dedico essa conquista.

Sou grato especialmente a minha irmã, Mayara Vilarim, que diante de todas as adversidades, me deu todo suporte, oportunidade e paciência que eu precisava para trilhar todo o caminho que me trouxe até aqui. Devo muito a você.

A minha família, entre quais destaco minhas avós, Josefa dos Santos e Maria Margarida. Ao meu avô Carlos Antônio. A estes devo todo carinho e cuidado que recebi. Agradeço a minha namorada, Maria Luana, que ao longo da confecção desse trabalho me deu todo apoio e incentivo, estando comigo nos melhores e piores dias.

Aos amigos que o curso me deu: Natan dos Santos, Taís Lima, Fabrícia Paola, Camila Machado e Igor Paiva. Obrigado pelas ajudas nas disciplinas, conversas e risadas. e os ensinamentos que vão além dos muros da universidade. Deixo também meu obrigado a todas as outras pessoas que cruzaram meu caminho ao longo do curso e me ajudaram a chegar até aqui.

Por fim, mas não menos importante, agradeço ao professor orientador Edmar Candeia Gurjão, pela paciência, compreensão, palavras de apoio e sábios ensinamentos para comigo em todas as vezes que nos encontramos no percurso da graduação e pela orientação ao longo do trabalho.

"Mude, mas comece devagar, porque a direção é mais importante que a velocidade."

Clarisse Lispector

Resumo

Neste trabalho são apresentadas as atividades do aluno Matheus Vilarim Pereira dos Santos, realizadas durante o Estágio Supervisionado no Laboratório de Metrologia (LabMet), sob orientação do Professor Edmar Candeia Gurjão. O estágio é um trabalho integrado ao projeto BINGO (Baryon Acoustic Oscillations from Integrated Neutral Gas Observation) e dividido em etapas, as quais constituem: desenvolvimento de um sistema de acionamento remoto para o Uirapuru, projeto de um sistema de sensoriamento remoto e por fim realização dos testes e validação. Como ferramentas metodológicas utilizou-se os microcontroladores ESP01 e ESP32, esses foram programados com o Arduino IDE, afim de solucionar as problemáticas propostas.

Palavras-chaves: Estágio Supervisionado, LabMet, BINGO, Acionamento Remoto, ESP32.

Abstract

Pereira dos Santos, carried out during the Supervised Internship at the Metrology Laboratory (LabMet), under the supervision of Professor Edmar Candeia Gurjão. The internship is a work integrated into the BINGO project (Baryon Acoustic Oscillations from Integrated Neutral Gas Observation) and divided into stages, which constitute: the development of a remote actuation system for the Uirapuru, the project of a remote sensing system, and finally the realization of testing and validation. As methodological tools were used the microcontrollers ESP01 and ESP32, were programmed with Arduino IDE, in order to solve the proposed problems.

Key-words: Supervised Internship, LabMet, BINGO, Remote Actuation, ESP32.

Lista de ilustrações

Lista de abreviaturas e siglas

UFCG	Universidade Federal de Campina Grande
LabMet	Laboratório de Metrologia
BINGO	*Baryon Acoustic Oscillations from Integrated Neutral Gas Observation*
MCU	*Microcontroller Unit*
SoC	*System-on-a-Chip*
RF	*Radio Frequency*
TSMC	*Taiwan Semiconductor Manufacturing Company*
API	*Application Programming Interface*
LNA	*Low Noise Amplifier*

Sumário

1 Introdução

A execução do estágio supervisionado é um requisito obrigatório para alunos concluintes e previsto pelo Projeto Pedagógico do Curso de Engenharia Elétrica da Universidade Federal de Campina Grande (UFCG) para obtenção do grau de Bacharel em Engenharia Elétrica.

É esperado que, com a realização do estágio obrigatório, o aluno se familiarize com a área de atuação e possa, de forma prática, consolidar os conhecimentos adquiridos durante a graduação, sendo, portanto, ferramenta de extrema importância à formação profissional.

O estágio supervisionado, cujas atividades realizadas pelo estudante Matheus Vilarim Pereira dos Santos estão descritas no presente relatório, foi ambientado no Laboratório de Metrologia (LabMet), compreendendo o período de 06 de agosto à 25 de outubro de 2021, com uma carga horária de 25 horas semanais, totalizando 270 horas (9 créditos), sob a orientação do professor Dr. Edmar Candeia Gurjão e a supervisão de Valber Aragão.

1.1 Projeto BINGO

O BINGO (Baryon Acoustic Oscillations from Integrated Neutral Gas Observation) é um projeto de colaboração internacional entre Brasil, Inglaterra, China, França, Itália, Espanha, Alemanha, África do Sul e Suíça. As instituições de ensino e pesquisa brasileiras envolvidas no projeto são a Universidade de São Paulo (USP), Instituto Nacional de Pesquisas Espaciais (INPE) e a Universidade Federal de Campina Grande (UFCG).

O Projeto foi idealizado pela University of Manchester e por meio de uma parceria com a USP, iniciou-se a fase de estudo e desenvolvimento no ano de 2016, com o apoio financeiro da FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo). Primeiro e único radiotelescópio que será dedicado a detecção e estudo das Oscilações Acústicas de Bárions por meio de rádio frequência. A detecção será feita na banda de rádio na faixa de 980 a 1260MHz.

Embora o projeto tenha seu principal foco na detecção das oscilações de bárions, em virtude da sua estrutura, da faixa de frequência analisada e posição imóvel, é possível estabelecer metas secundárias de estudo como a detecção de FRB (Fast Radio Bursts), sendo pioneiro no hemisfério sul, e emissão de pulsares.

O radiotelescópio será construído na Serra do Urubu, localizado na cidade de Piancó, Paraíba. Ele é constituído de dois espelhos refletores, ou parabólicas, cada uma com aproximadamente 40 metros de diâmetro e um conjunto de 50 cornetas acopladas.

Uma de suas parabólicas receberá incidência de ondas eletromagnéticas, que por sua vez refletirá para a segunda parabólica e por fim enviará os sinais para o conjunto de cornetas, estas por sua vez farão o envio desses sinais até os receptores.

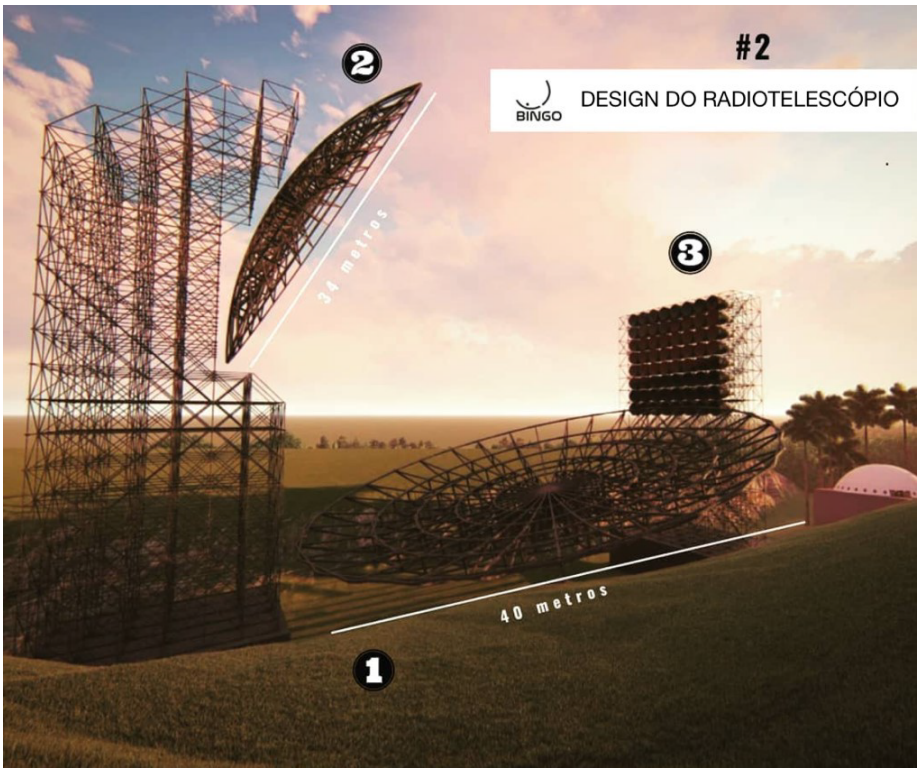

Figura 1 – Radiotelescópio BINGO. Fonte: *Bingo Project*

1.2 Objetivo

1.2.1 Objetivo Geral

A atividade realizada teve como objetivo implementar um sistema remoto, o qual tem como objetivo o controle do acionamento da alimentação dos componentes presentes na cadeia de recepção da corneta, além também de possibilitar o sensoriamento de temperatura e tensão, que será utilizado no controle e monitoramento do sistema da corneta Uirapuru que faz parte do projeto BINGO (Baryon Acoustic Oscillations from Integrated Neutral Gas Observation).

1.2.2 Objetivos Específicos

- Escolha dos componentes;
- Montagem do sistema;
- Desenvolvimento do código dos microcontroladores;

- Melhoria e testes.

1.3 Estrutura do Trabalho

O capítulo 1 é reservado para apresentação suncinta das informções mais relevantes acerca do estágio realizado.

No capítulo 2 é fornecido, de forma breve, um panorama geral do laboratório onde o estágio foi realizado.

O capítulo 3 é destinado à fundamentação teórica acerca dos principais temas empregados na realização das atividades.

No capítulo 4 estão descritas de forma detalhada as atividades desenvolvidas pelo estagiário.

No capítulo 5 são apresentadas as conclusões acerca do trabalho.

2 Local do Estágio

O estágio foi realizado nas dependências do LabMet, no Departamento de Engenharia Elétrica (DEE), localizado na Universidade Federal de Campina Grande (UFCG), no endereço Rua Aprigio Veloso 822, 58429-900, Campina Grande. O laboratório tem como funções a criação, desenvolvimento e aperfeiçoamento de ensaio e calibração de sistemas.

Para o cumprimento das atividades foi disponibilizada a sala 18 do laboratório, onde estavam disponíveis todos os equipamentos necessários.

O LabMet hoje abriga a estrutura de testes da corneta do Uirapuru, que faz parte do projeto BINGO.

3 Fundamentação Teórica

3.1 Microcontrolador

Um microcontrolador é um computador pequeno e de baixo custo, que é projetado para realizar as tarefas específicas de sistemas embarcados, como exibir informações, receber sinais remotos, controlar e automatizar processos (1).

O microcontrolador geral consiste no processador, na memória (RAM, ROM, EPROM), portas seriais, periféricos (temporizadores, contadores, sensores e módulos de rede).

3.1.1 ESP01

O módulo microcontrolador ESP8266 ESP-01 é uma ferramenta de baixo custo desenvolvido pelo Sistema Espressif que tem a capacidade de executar conexões TCP IP e possui conectividade WiFi (2). Este MCU contém muitas funcionalidades para seu tamanho e tem portas que pode-se usar para controlar dos projetos mais simples aos mais complexos. Na Figura 1 pode-se ver um exemplo desse microcontrolador.

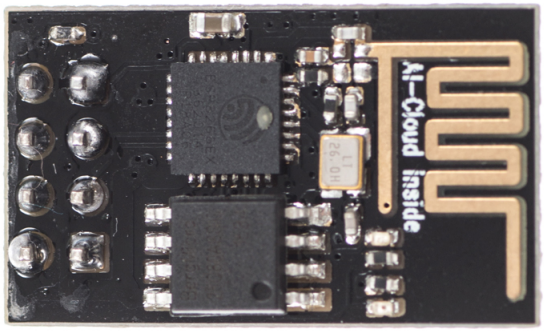

Figura 2 – Microcontrolador ESP01. Fonte: Google Imagens.

O ESP8266 ESP-01 vem com um firmware AT pré-instalado. É possível programar o chip com outro firmware como NodeMCU, por exemplo. No entanto, o firmware AT é compatível com o IDE do Arduino, o que faz da sua programação e carregamento de código atividades práticas. Na Figura 2 pode-se observar todos os pinos presentes no ESP, os quais são usados para alternar entre os modos de funcionamento normal e carregamento de dados.

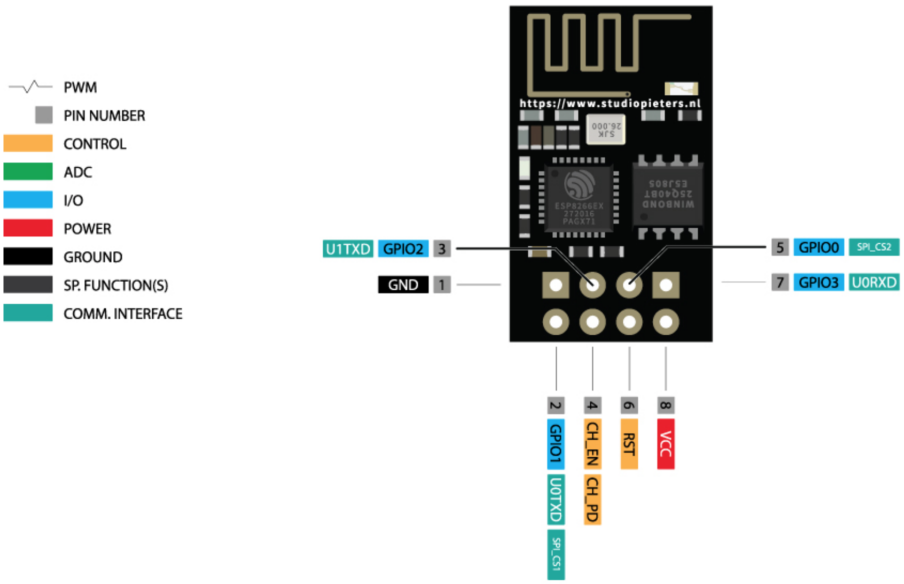

Figura 3 – Pinos ESP01. Fonte: Google Imagens.

3.1.2 ESP32

ESP32 é um microcontrolador *System on Chip* (SoC) de baixo custo da *Espressif Systems*, os desenvolvedores do famoso ESP8266 SoC. É um sucessor do ESP8266 SoC e vem em variações *single-core* e *dual-core* do microprocessador Xtensa LX6 de 32 bits da Tensilica com Wi-Fi e Bluetooth integrados (3).

O ESP32, como o ESP8266, tem como diferencial seus componentes de RF integrados, como amplificador de potência, amplificador de recepção de baixo ruído, interruptor de antena, filtros e Balun RF. Isso torna o projeto de hardware em torno do ESP32 muito prático, pois são necessários poucos componentes externos (4).

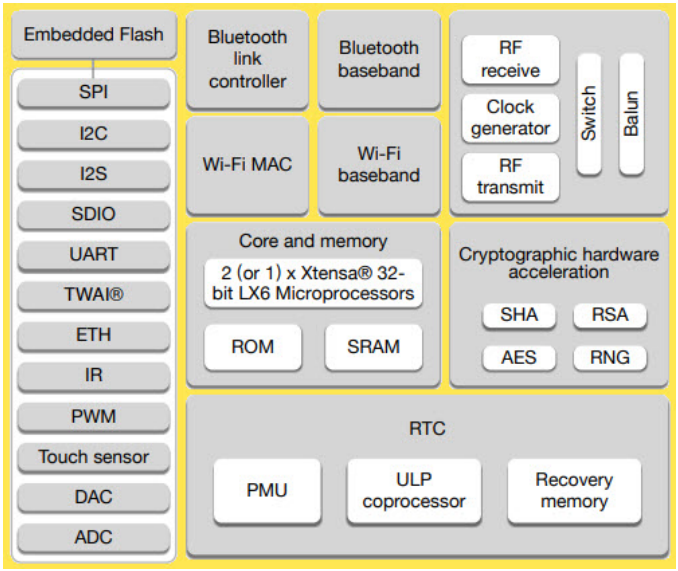

Figura 4 – Componentes ESP32. Fonte: Electronics Hub.

Outro importante tópico a destacar sobre o ESP32 é que ele é fabricado com a tecnologia de ultra-baixa potência de 40 nm da TSMC. Portanto, projetar aplicativos operados por bateria, como *wearables*, equipamentos de áudio, monitoradores, relógios inteligentes, etc., usando o ESP32 faz-se uma tarefa deveras interessante.

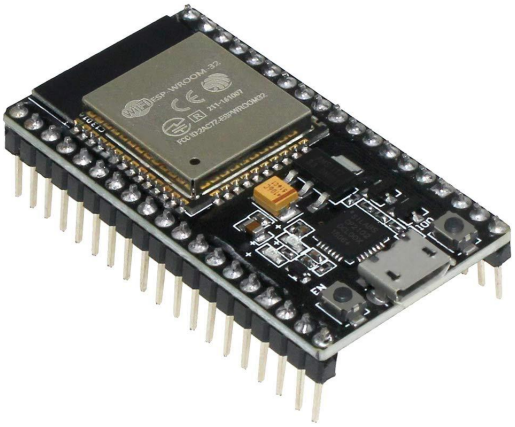

Figura 5 – Microcontrolador ESP32. Fonte: Electronics Hub.

Algumas especificações importantes do ESP32 são:

- Microprocessador LX6 de 32 bits com um ou dois núcleos com freqüência de clock de até 240 MHz;
- 520 KB de SRAM, 448 KB de ROM e 16 KB de RTC SRAM;
- Suporta conectividade Wi-Fi 802.11 b/g/n com velocidades de até 150 Mbps;
- Suporte para as especificações Classic Bluetooth v4.2 e BLE;
- 34 GPIOs programáveis;
- Até 18 canais de ADC SAR de 12 bits e 2 canais de DAC de 8 bits;
- A conectividade serial inclui 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART;
- Ethernet MAC para comunicação de LAN física (requer PHY externo);
- 1 controlador Host para SD/SDIO/MMC e 1 controlador Slave para SDIO/SPI;
- Módulo PWM e até 16 canais de LED PWM;
- Criptografia de inicialização segura e Flash;
- Aceleração criptográfica de hardware para AES, Hash (SHA-2), RSA, ECC e RNG.

3.2 Sensores

Um sensor adquire uma quantidade física e a converte em um sinal adequado para processamento (por exemplo, óptico, elétrico, mecânico). Os sensores comuns convertem a medição de fenômenos físicos em um sinal elétrico. O elemento ativo de um sensor é denominado transdutor.

Os sensores podem ser classificados de forma geral como: ativos ou passivos, analógicos ou digitais.

3.2.1 DHT22

O DHT-22 (também denominado AM2302) é um sensor de saída digital, umidade relativa e temperatura. Ele usa um sensor de umidade capacitivo e um termistor para medir o ar circundante e envia um sinal digital no pino de dados (5).

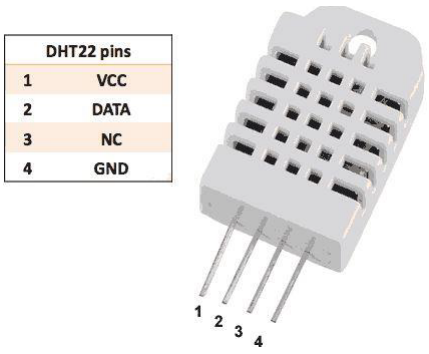

Figura 6 – Sensor DHT22. Fonte: Tutorials Point.

Detalhes técnicos:

- Alimentação: 3/5 V;
- Corrente máxima: 2.5 mA;
- Humidade: escala de 0 à 100%, 2-5% de precisão;
- Temperatura: -40 à 80 ℃, ±0.5 ℃de precisão.

3.3 Interface Humano-Máquina e Banco de Dados

Uma Interface Humano-Máquina (IHM) é uma interface de usuário ou painel que conecta uma pessoa a uma máquina, sistema ou dispositivo. Embora o termo possa ser tecnicamente aplicado a qualquer tela que permita a um usuário interagir com um dispositivo, a IHM é mais comumente usada no contexto de um processo industrial.

3.3.1 HTML

Hypertext Markup Language (HTML) é uma linguagem de computador que compõe a maioria das páginas da web e aplicativos online. Um hipertexto é um texto usado para fazer referência a outras partes do texto, enquanto uma linguagem de marcação é uma série de marcações que informa aos servidores da web o estilo e a estrutura de um documento (6).

HTML não é considerado uma linguagem de programação, pois não pode criar funcionalidade dinâmica. Em vez disso, com HTML, os usuários da web podem criar e estruturar seções, parágrafos e links usando elementos, tags e atributos.

Aqui estão alguns dos usos mais comuns para HTML:

- Desenvolvimento web. Os desenvolvedores usam o código HTML para projetar como um navegador exibe os elementos da página da web, como texto, hiperlinks e arquivos de mídia;
- Navegação na Internet. Os usuários podem navegar facilmente e inserir links entre páginas e sites relacionados, pois o HTML é muito usado para inserir hiperlinks;
- Documentação da web. O HTML permite organizar e formatar documentos, de forma semelhante ao Microsoft Word.

Documentos HTML são arquivos que terminam com uma extensão .html ou .htm. Um navegador da web lê o arquivo HTML e renderiza seu conteúdo para que os usuários da Internet possam visualizá-lo.

Todas as páginas HTML possuem uma série de elementos HTML, consistindo em um conjunto de tags e atributos. Os elementos HTML são os blocos de construção de uma página da web. Uma tag informa ao navegador da web onde um elemento começa e termina, enquanto um atributo descreve as características de um elemento.

3.3.2 Firebase

O *Firebase Realtime Database* é um banco de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados. Quando você cria apps multiplataforma com nossos SDKs para iOS, Android e JavaScript, todos os clientes compartilham uma instância do *Realtime Database* e recebem automaticamente atualizações com os dados mais recentes (7).

No *Firebase Realtime Database* é possível criar aplicativos avançados e colaborativos, ao conceder acesso seguro ao banco de dados diretamente do código do cliente. Os dados são mantidos localmente e, mesmo *off-line*, os eventos em tempo real continuam

sendo acionados, proporcionando uma experiência responsiva ao usuário final. Quando o dispositivo recupera a conexão, o *Realtime Database* sincroniza as alterações feitas nos dados locais com as atualizações remotas que ocorreram enquanto o cliente estava *off-line*, mesclando qualquer conflito automaticamente.

O *Realtime Database* fornece uma linguagem de regras flexíveis baseadas em expressão, denominadas regras de segurança, para definir como os dados são estruturados e quando podem ser lidos e gravados. Por meio da integração com o *Firebase Authentication*, os desenvolvedores podem definir quem tem acesso, a quais dados e como esses dados podem ser acessados.

O *Realtime Database* é um banco de dados NoSQL e, por isso, tem otimizações e funcionalidades diferentes de um banco de dados relacional. A API do *Realtime Database* foi desenvolvida para autorizar apenas operações que possam ser executadas com rapidez. Isso possibilita uma ótima experiência em tempo real que atende a milhões de usuários sem comprometer a capacidade de resposta. Por isso, é importante analisar como os usuários precisam acessar os dados e estruturá-los adequadamente.

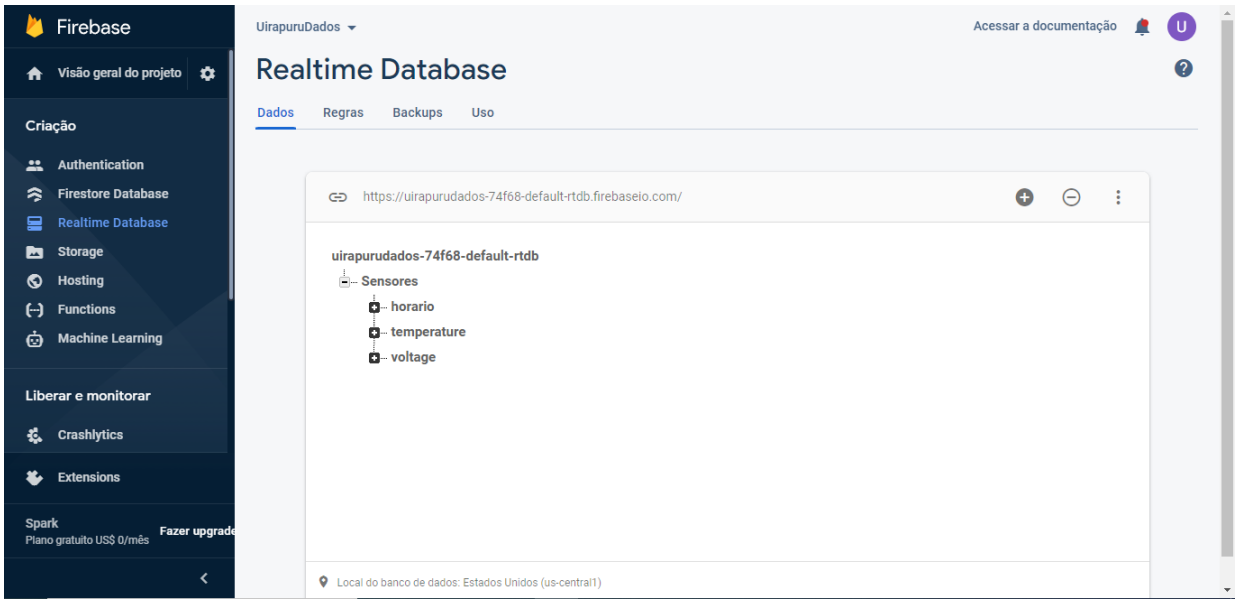

Figura 7 – Painel do Firebase. Fonte: Autor.

4 Atividades Desenvolvidas

Neste capítulo serão descritas as principais atividades desenvolvidas durante o estágio. Inicialmente trabalhou-se nas possibilidades de acionamento remoto que poderiam ser utilizadas pelo sistema. Em seguida projetou-se o circuito de sensoriamento, o qual é responsável pela aquisição de dados de temperatura, tensão e uma base de tempo de referência.

4.1 Sistema de acionamento remoto

A corneta do Uirapuru recebe ondas de rádio de potências muito baixas. Dessa forma, faz-se necessário uso de amplificadores para aquisição dos sinais e posterior tratamento desses. Para evitar a inserção de muitas componentes de sinal ruidosas, foi utilizado uma cadeia de LNAs.

Um amplificador de baixo ruído (LNA) amplifica um sinal de potência muito baixa sem degradar significativamente sua relação sinal-ruído. Quando amplificadores regulares amplificam sinais, ruído adicional é frequentemente introduzido no sistema. No entanto, usando um LNA, esse ruído pode ser reduzido significativamente.

O sistema de acionamento remoto se propõe então a controlar a ativação da cadeia de LNAs e do arranjo que faz toda aquisição e transmição dos dados de sensores do Uirapuru.

Procurou-se utilizar modulos de componentes prontos, pois esses possibilitam a rápida e fácil reposição em caso de falha ou queima. Este aspecto do projeto se faz deveras importante, visto que as cornetas do BINGO ficarão em localização remota e de difícil acesso, além disso, o sistema não pode ficar desativado por longos períodos, assim, precisou-se visar formas de reparo ágeis.

Dado o cenário de projeto descrito acima, optou-se pelo uso de um módulo relé comumente encontrado em lojas locais, esse pode ser visto na Figura 8.

Figura 8 – Módulo de Acionamento Relé. Fonte: Google Imagens.

Ao módulo de acionamento relé, acoplou-se o ESP01. Foram feitas pequenas modificações na placa, como a remoção do resistor R2, o que possibilitou o correto funcionamento do microcontrolador, visto que o esse resistor fazia o ESP01 iniciar em modo *boot* e não no modo normal. Na Figura 9 pode-se observar a montagem final.

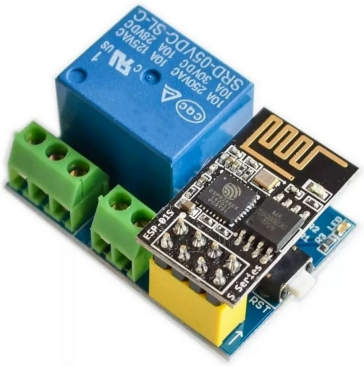

Figura 9 – Módulo de Acionamento Relé ESP01. Fonte: Google Imagens.

O ESP01 foi programado utilizando o Arduino IDE. A esse foi adicionado um complemento de firmware para ESP8266, o qual possibilita o uso das bibliotecas do controlador e o carregamento de código.

O código desenvolvido pode ser visto no anexo A, nesse foi implementado um servidor para o ESP, o qual responde a requisições feitas pela porta 80. Foi construído um código HTML para exibir o status do sistema e possibilitar interações com o usuário, quando esse seleciona um dos botões presentes na página, o ESP01 aciona uma função que altera o nível lógio do GPIO-0, dessa forma enviando o comando de acionamento ou desligamento do relé.

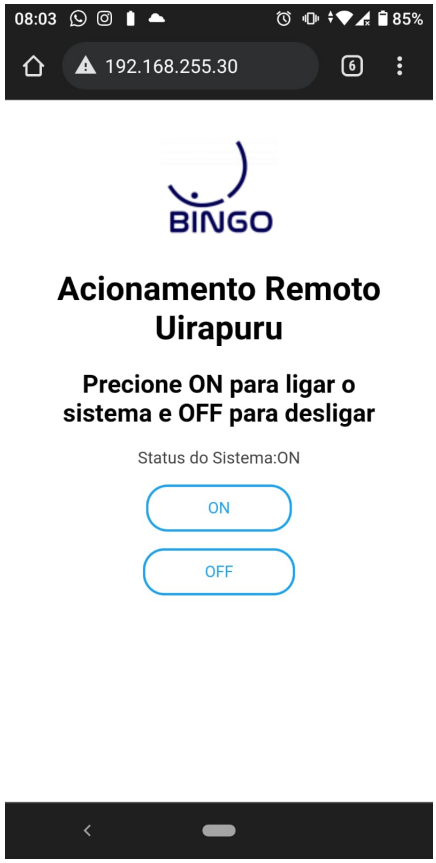

Figura 10 – Página de acionamento remoto. Fonte: Autor.

O microcontrolador se comunica com a rede local por meio de WiFi. Optou-se por essa forma de comunicação devido o sinal estar em 2.4 GHz, evitando interferências eletromagnéticas na faixa do Uirapuru, ou se esse for captado pelo sistema, possui características de fácil identificação do sinal.Na Figura 11, pode-se observar o sistema instalado na caixa de componentes do Uirapuru.

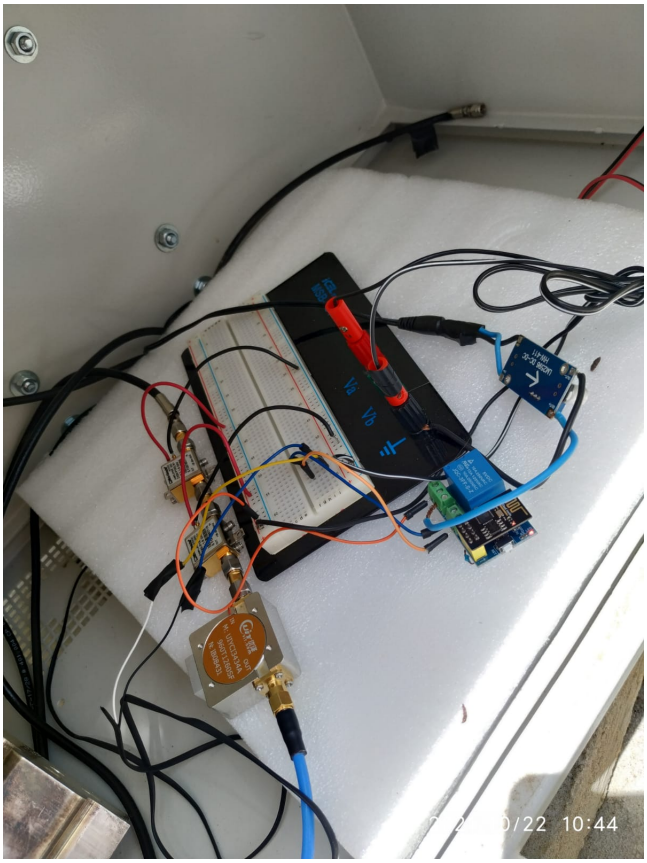

Figura 11 – Sitema de acionamento remoto instalado. Fonte: Autor.

4.2 Sistema de aquisição de dados dos sensores

Os LNAs utilizados na recepção de sinal do Uirapuru, possuem considerável sensibilidade nos seus ganhos com relação a temperatura. Ou seja, variações de temperatura podem alterar o ganho aplicado pelos amplificadores ao sinal.

Dessa forma, faz-se necessária a aferição da temperatura na cabine onde estão localizados os LNAs. Podendo-se fazer uso desses dados na composição da análise final feita sobre os sinais recebidos pela corneta.

Optou-se pela utilização do microcontrolador ESP32. A esse foram conectados o sensor de temperatura DHT22 e um módulo para aferição de tensão. Esse módulo consiste apenas de um divisor de tensão, o qual faz-se necessário para reduzir as tensões a níveis que não danifiquem a entrada analógica do ESP.

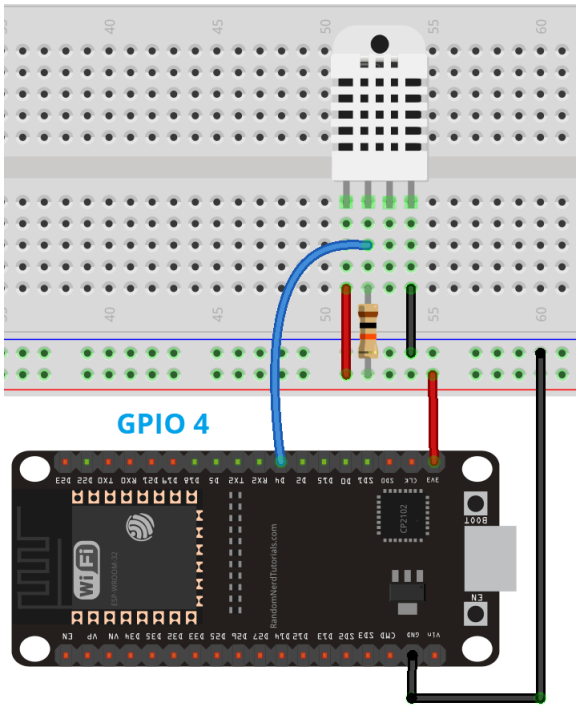

Figura 12 – Esquema de ligação do DHT22 ao ESP32. Fonte: Autor.

O código do ESP32 foi implementado e gravado utilizando o Arduino IDE. Projetou-se um servidor e uma página HTML para o microcontrolador, essa página possibilitou a visualização dos dados aferidos com os sensores. Na Figura 13, pode-se observar a página elaborada.

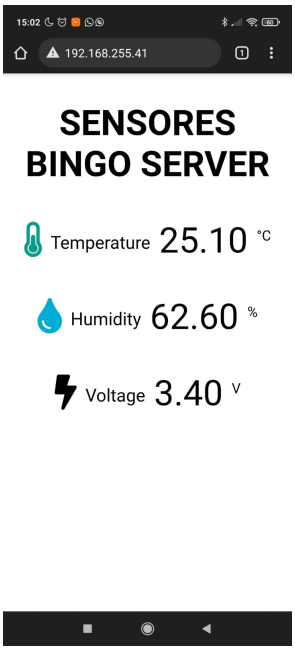

Figura 13 – Página de dados dos sensores. Fonte: Autor.

O código implementado para o projeto pode ser visto no anexo B deste trabalho. Como base de tempo para o sistema, utilizou-se o servidor NTP.br, o qual fornece o

horário atual baseado no UTC. A seguir pode-se ver o sistema montado.

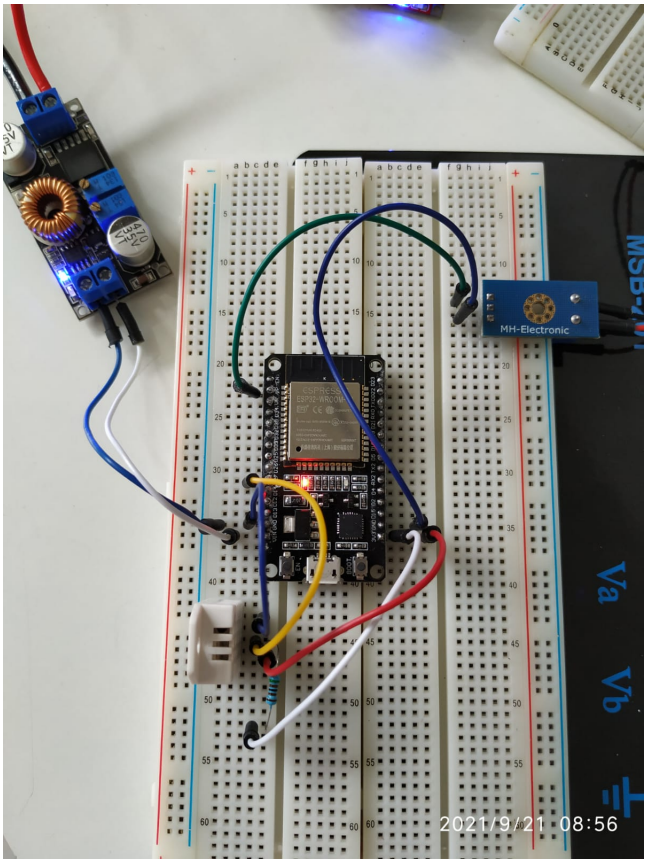

Figura 14 – Sistema montado com o ESP32. Fonte: Autor.

Para gerenciar o histórico de dados, utilizou-se o serviço Firebase do Google. Os dados foram salvos em um banco NoSQL, o qual está na nuvem. O envio dos dados é feito em tempo real para o *Realtime Database*, que é o serviço de BD presente no Firebase. Na Figura 15, pode-se observar os dados recebidos em um dos testes performados.

Foram implementadas regras de acessos aos dados do Firebase, optou-se por usar autenticação por conta do Google. A segurança dos dados é fundamental para dar confiabilidade ao sistema. Para a comunicação do ESP32 com o Firebase, foram usadas funções presentes em um complemento de firmware disponível para o Arduino IDE.

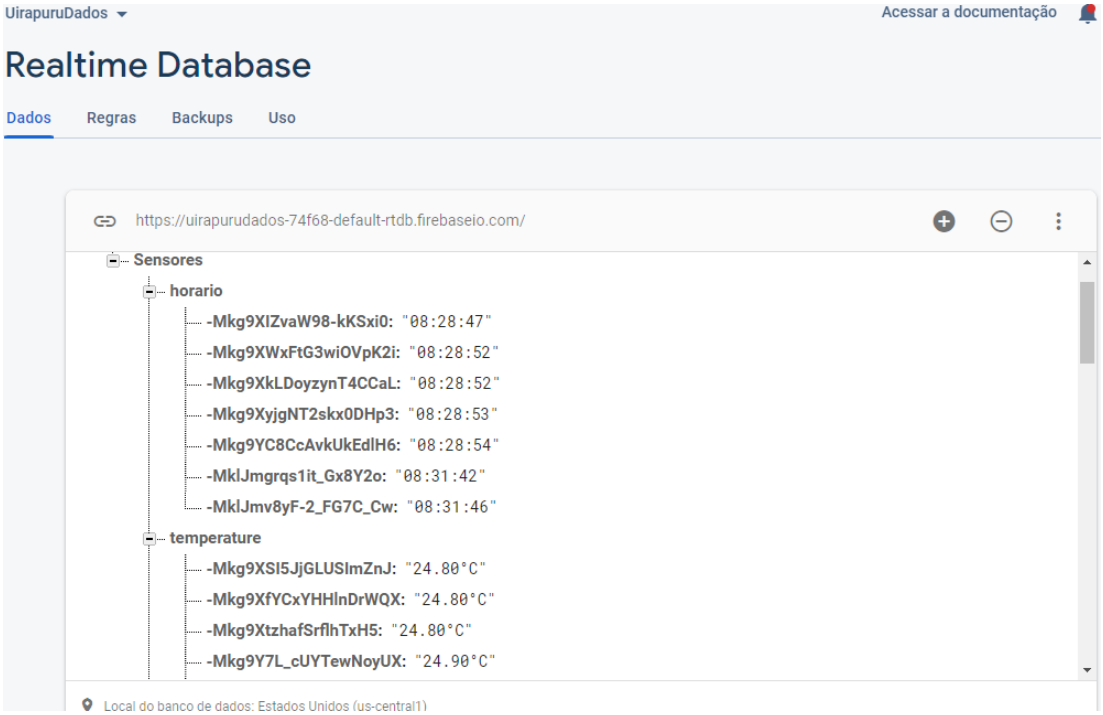

Figura 15 – Informações recebidas no Banco de dados. Fonte: Autor.

5 Considerações Finais

No trabalho desenvolvido foram analisadas as problemáticas existentes para o acompanhamento de grandezas fundamentais para o funcionamento do radiotelescópio BINGO Uirapuru. Por meio dessas análises, tomou-se decisões de projeto que melhor se adequacem ao cenário apresentado.

Pode-se inferir o correto funcionamento dos sistemas projetados, visto que os mesmos responderam de forma esperada aos testes. Todo o sistema implementado ainda está sendo colocado a prova por maiores períodos de tempo e é passível de contínuas melhorias, conforme os erros que se apresentarem.

Por fim, conclui-se que o trabalho desenvolvido foi fundamental para a prática de fundamentos teóricos absorvidos ao longo da graduação. Tais ferramentas teóricas foram colocadas a prova e puderam ser enriquecidas com experiência prática, ratificando o real papel do período de estágio.

Referências

1 GARCIA-RUIZ, M.; SANTANA, P. Introduction to microcontrollers and microcontroller boards. In: _____. [S.l.: s.n.], 2021. p. 19. ISBN 978-1800564138. 5

2 DAHOUD, A. A.; FEZARI, M. Presentation, programming and applications of the nodemcu-esp-01 module. In: _____. [S.l.: s.n.], 2019. 5

3 CAMERON, N. Esp32 microcontroller features. In: _____. [S.l.: s.n.], 2021. p. 641–682. ISBN 978-1-4842-6335-8. 6

4 ZIM, M. Z. H. Tinyml: Analysis of xtensa lx6 microprocessor for neural network applications by esp32 soc. In: _____. [S.l.: s.n.], 2021. 6

5 AHMAD, Y. et al. On the evaluation of dht22 temperature sensor for iot application. In: . [S.l.: s.n.], 2021. p. 131–134. 8

6 MOZILLA. *HTML: Linguagem de Marcação de Hipertexto*. 2021. <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. 9

7 SOFTWELL. *Firebase: Funções e Importância no Desenvolvimento Mobile*. 2021. <https://mundomaker.com.br/firebase/>. 9

Anexos

ANEXO A – Código

Abaixo está o código usado no ESP01 para o projeto de acionamento remoto.

```
//Inclui as funções de Wifi do ESP
#include <ESP8266WiFi.h>

//Cria um server na porta 80 (porta padrão para onde os navegadores
↪  enviam as requisições http)
WiFiServer server(80);

String SISTEM_STATE = "off";

void setup()
{
  //Inicializa a Serial apenas para efeito de log
  Serial.begin(115200);

  //Configura o GPIO0 e GPIO2 como output, ou seja, como saída para
  ↪  podermos alterar o seu valor
  pinMode(0, OUTPUT);
  //pinMode(2, OUTPUT);
  //Deixa o GPIO0 e o GPIO2 com saída LOW
  digitalWrite(0, LOW);
  //digitalWrite(2, LOW);

  Serial.print("Conectando");
  //Faz o ESP se conectar à rede WiFi. No nosso exemplo o ssid da rede é
  ↪  TesteESP e a senha é 87654321
  WiFi.begin("LABMET", "**********");

  //Enquanto o ESP não se conectar à rede
  while (WiFi.status() != WL_CONNECTED)
  {
    //Esperamos 100 milisegundos
    delay(100);
    Serial.print(".");
  }
```

```
  //Se chegou aqui é porque conectou à rede, então mostramos no monitor
  ↪  serial para termos um feedback
  Serial.println("");
  Serial.println("Conectou");

  //Configurações do IP fixo. Você pode alterar conforme a sua rede
  IPAddress ip(192, 168, 255, 30);
  IPAddress gateway(192, 168, 255, 3);
  IPAddress subnet(255, 255, 255, 0);
  Serial.print("Configurando IP fixo para : ");
  Serial.println(ip);

  //Envia a configuração
  WiFi.config(ip, gateway, subnet);

  //Inicializa o server que criamos na porta 80
  server.begin();

  //Mostramos no monitor serial o IP que o ESP possui para verificarmos
  ↪  se é o mesmo que configuramos
  Serial.print("Server em: ");
  Serial.println(WiFi.localIP());
}

void loop()
{
  //Verifica se algum cliente está tentando se conectar
  WiFiClient client = server.available();
  if (!client)
  {
    //Se não houver nenhum cliente podemos retornar pois não há nada a
    ↪  fazer
    return;
  }

  Serial.println("Novo cliente conectou");

  //Fazemos a leitura da requisição
```

```
String req = client.readStringUntil('\r');
Serial.print("Requisição: ");
Serial.println(req);

//Este é o html que iremos retornar para o cliente
//É composto basicamente de dois botões (ON e OFF) para o GPIO0 e dois
↪  botões (ON e OFF) para o GPIO2
//A parte que nos interessa é o <a href=' com a ação vinculada a cada
↪  botão
//Quando clicamos em um destes botões essa informação chegará até o
↪  ESP para que ele verifique qual ação deve executar
//A parte dentro de '<style>' é apenas para modificarmos o visual da
↪  página que será exibida, você pode alterá-la como queira
String html =
"<!DOCTYPE html><html>"
  "<head>"
    "<meta name='viewport' content='width=device-width,
    ↪  initial-scale=1, user-scalable=no'/>"
    "<title>""Acionamento Remoto Uirapuru""</title>"
    "<style>"
      "body{"
        "text-align: center;"
        "font-family: sans-serif;"
        "font-size:14px;"
        "padding: 25px;"
      "}"

      "p{"
        "color:#444;"
      "}"

      "button{"
        "outline: none;"
        "border: 2px solid #1fa3ec;"
        "border-radius:18px;"
        "background-color:#FFF;"
        "color: #1fa3ec;"
        "padding: 10px 50px;"
      "}"
```

```
        "button:active{"
          "color: #fff;"
          "background-color:#1fa3ec;"
        "}"
      "</style>"
    "</head>"

    "<body><img
    ↪  src='https://bingotelescope.org/wp-content/uploads/2021/04/logo-bingo-e1625
    "<body><center><h1>Acionamento Remoto Uirapuru</h1></center>"
    "<center><h2>Precione ON para ligar o sistema e OFF para
    ↪  desligar</h2></center>"
    "<body>"
    "<p> Status do Sistema:" + SISTEM_STATE + "</p>"
    "<p><a href='?acao=gpio0On'><button>ON</button></a></p>"
    "<p><a href='?acao=gpio0Off'><button>OFF</button></a></p>"
//    "<p>GPIO2</p>"
//    "<p><a href='?acao=gpio2On'><button>ON</button></a></p>"
//    "<p><a href='?acao=gpio2Off'><button>OFF</button></a></p>"
    "</body>"
  "</html>";

  //Escreve o html no buffer que será enviado para o cliente
  client.print(html);
  //Envia os dados do buffer para o cliente
  client.flush();

  //Verifica se a requisição possui a ação gpio0On
  if (req.indexOf("acao=gpio0On") != -1)
  {
      //Se possui a ação gpio0On colocamos a saída do GPIO0 como alta
      digitalWrite(0, HIGH);
      SISTEM_STATE = "ON";
  }
  //Senão, verifica se a requisição possui a ação gpio0Off
  else if (req.indexOf("acao=gpio0Off") != -1)
  {
    //Se possui a ação gpio0Off colocamos a saída do GPIO0 como baixa
```

```
    digitalWrite(0, LOW);
    SISTEM_STATE = "OFF";
  }
  //Senão, verifica se a requisição possui a ação gpio2On
//  else if (req.indexOf("acao=gpio2On") != -1)
//  {
//    //Se possui a ação gpio2On colocamos a saída do GPIO2 como alta
//    digitalWrite(2, HIGH);
//  }
//  //Senão, verifica se a requisição possui a ação gpio2Off
//  else if (req.indexOf("acao=gpio2Off") != -1)
//  {
//    //Se possui a ação gpio0Off colocamos a saída do GPIO2 como baixa
//    digitalWrite(2, LOW);
//  }

  //Fecha a conexão com o cliente
  client.stop();
  Serial.println("Cliente desconectado");
}
```

ANEXO B – Código

A seguir está o código implementado para o ESP32 no projeto de aquisição e transmissão de dados dos sensores.

```
// Import required libraries
#include "WiFi.h"
#include <NTPClient.h>
#include "ESPAsyncWebServer.h"
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <FirebaseESP32.h>

// Define Firebase Credentials
#define FIREBASE_HOST
↪  "https://uirapurudados-74f68-default-rtdb.firebaseio.com/"
#define FIREBASE_AUTH "nGI8KhyzgXXW44p8woENAnhhBlvSUziSkuPbA6EZ"
#define API_KEY "AIzaSyBcHbBjhsO-hoYO5L9qWGzBvxxQ6R-g_oo"
FirebaseData firebaseData;
FirebaseAuth auth;
FirebaseConfig config;

/* 4. Define the user Email and password that alreadey registerd or
↪  added in your project */
#define USER_EMAIL "uirapurudados@gmail.com"
#define USER_PASSWORD "Upd44123"

//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// Replace with your network credentials
const char* ssid = "LABMET";
const char* password = "labmet@dee";

#define DHTPIN 27     // Digital pin connected to the DHT sensor
```

```
// Uncomment the type of sensor in use:
//#define DHTTYPE    DHT11     // DHT 11
#define DHTTYPE    DHT22     // DHT 22 (AM2302)
//#define DHTTYPE    DHT21     // DHT 21 (AM2301)

DHT dht(DHTPIN, DHTTYPE);

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

// Client NTP
WiFiUDP udp;
NTPClient ntp(udp, "a.st1.ntp.br", -3 * 3600, 60000);
String horario;

/*String readHorario() {
  // Getting the local time
  String horario = ntp.getFormattedTime();
  return horario;
  }
*/

String readVoltage() {
  // Reading the sourge voltage
  int volt = analogRead(36);// read the input
  double vout = map(volt,0,1023, 0, 2500);// map 0-1023 to 0-2500 and
  ↪  add correction offset
  vout /=100;// divide by 100 to get the decimal values
  double voltage = vout*0.2;

  Serial.print("Voltage: ");
  Serial.print(voltage);//print the voltge
  Serial.println("V");

  return String(voltage);
 }

String readDHTTemperature() {
```

```
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow
  ↪  sensor)
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  //float t = dht.readTemperature(true);
  // Check if any reads failed and exit early (to try again).
  if (isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return "--";
  }
  else {
    Serial.println(t);
    return String(t);
  }
}

String readDHTHumidity() {
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow
  ↪  sensor)
  float h = dht.readHumidity();
  if (isnan(h)) {
    Serial.println("Failed to read from DHT sensor!");
    return "--";
  }
  else {
    Serial.println(h);
    return String(h);
  }
}

const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
  ↪  href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
  ↪  integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLb
  ↪  crossorigin="anonymous">
```

```
  <style>
    html {
     font-family: Arial;
     display: inline-block;
     margin: 0px auto;
     text-align: center;
    }
    h2 { font-size: 3.0rem; }
    p { font-size: 3.0rem; }
    .units { font-size: 1.2rem; }
    .dht-labels{
      font-size: 1.5rem;
      vertical-align:middle;
      padding-bottom: 15px;
    }
  </style>
</head>
<body>
  <h2>SENSORES BINGO SERVER</h2>
  <p>
    <i class="fas fa-thermometer-half" style="color:#059e8a;"></i>
    <span class="dht-labels">Temperature</span>
    <span id="temperature">%TEMPERATURE%</span>
    <sup class="units">&deg;C</sup>
  </p>
  <p>
    <i class="fas fa-tint" style="color:#00add6;"></i>
    <span class="dht-labels">Humidity</span>
    <span id="humidity">%HUMIDITY%</span>
    <sup class="units">&percnt;</sup>
  </p>
  <p>
    <i class="fas fa-bolt"></i>
    <span class="dht-labels">Voltage</span>
    <span id="voltage">%VOLTAGE%</span>
    <sup class="units">V</sup>
  </p>
</body>
<script>
```

```
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("temperature").innerHTML =
      ↪  this.responseText;
    }
  };
  xhttp.open("GET", "/temperature", true);
  xhttp.send();
}, 10000 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("voltage").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/voltage", true);
  xhttp.send();
}, 10000 ) ;

setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("humidity").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "/humidity", true);
  xhttp.send();
}, 10000 ) ;
</script>
</html>)rawliteral";

// Replaces placeholder with DHT values
String processor(const String& var){
  //Serial.println(var);
```

```
  if(var == "TEMPERATURE"){
    return readDHTTemperature();
  }
  else if(var == "HUMIDITY"){
    return readDHTHumidity();
  }
  else if(var == "VOLTAGE"){
    return readVoltage();
  }
  return String();
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);

  dht.begin();

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

//Configurações do IP fixo. Você pode alterar conforme a sua rede
 /* IPAddress ip(192, 168, 255, 41);
  IPAddress gateway(192, 168, 255, 3);
  IPAddress subnet(255, 255, 255, 0);
  Serial.print("Configurando IP fixo para : ");
  Serial.println(ip);

  //Envia a configuração
  WiFi.config(ip, gateway, subnet);
*/

  // Print ESP32 Local IP Address
  Serial.println(WiFi.localIP());
```

```
  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html, processor);
  });
  server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readDHTTemperature().c_str());
  });
  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readDHTHumidity().c_str());
  });
  server.on("/voltage", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readVoltage().c_str());
  });

   /* Assign the api key (required) */
  config.api_key = API_KEY;

  /* Assign the user sign in credentials */
  auth.user.email = USER_EMAIL;
  auth.user.password = USER_PASSWORD;

  /* Assign the RTDB URL (required) */
  config.database_url = FIREBASE_HOST;

  /* Assign the callback function for the long running token generation
  ↪  task */
  config.token_status_callback = tokenStatusCallback; //see
  ↪  addons/TokenHelper.h

  // Start server
  server.begin();
  Firebase.begin(&config, &auth);
  ntp.begin();
  ntp.forceUpdate();
}

void loop(){

  horario = ntp.getFormattedTime();
```

```
  String Voltage = readVoltage() + "V";
  String Temperature = readDHTTemperature() + "°C";

  Firebase.pushString(firebaseData, "/Sensores/voltage", Voltage);
  Firebase.pushString(firebaseData, "/Sensores/temperature",
  ↪  Temperature);
  Firebase.pushString(firebaseData, "/Sensores/horario", horario);
}
```