UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

SAMUEL CABRAL

UM SISTEMA DE TELEMETRIA COM TECNOLOGIAS GSM/GPRS PARA A ÁREA AMBIENTAL

BAURU
2023

SAMUEL CABRAL

UM SISTEMA DE TELEMETRIA COM TECNOLOGIAS GSM/GPRS PARA A ÁREA AMBIENTAL

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. João Eduardo Machado Perea Martins

BAURU
2023

C117s

Cabral, Samuel

Um sistema de telemetria com tecnologias GSM/GPRS para a área ambiental / Samuel Cabral. -- Bauru, 2023

66 p. : il.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru

Orientador: João Eduardo Machado Perea Martins

1. Internet das Coisas. 2. Arduino. 3. Redes de Dados. 4. Sensores. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Samuel Cabral

Um sistema de telemetria com tecnologias GSM/GPRS para a área ambiental

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. João Eduardo Machado Perea Martins
Orientador
Departamento de Computação
Faculdade de Ciências
Unesp - Bauru

Profª.Drª. Simone das Graças Domingues Prado
Departamento de Computação
Faculdade de Ciências
Unesp - Bauru

Prof.Dr Edvaldo José Scoton
Departamento de Engenharia Agronômica
Unisagrado - Bauru

Bauru, _____ de _____ de _____.

Dedico esta monografia a Deus, minha família, professores, amigos, e todos que me apoiaram para que eu chegasse até aqui.

Agradecimentos

Agradeço a Deus pela minha vida, por me acompanhar ao longo do curso e abençoar durante a realização deste projeto.

Agradeço a minha família pelo apoio durante a graduação, pelo suporte e por acreditarem em mim.

Agradeço a todos os professores pelo conhecimento oferecido e a contribuição com a minha formação, especialmente a meu orientador, por me direcionar durante o desenvolvimento do trabalho e todo apoio e esforço.

Por fim, agradeço a todos os amigos do curso, por todo o apoio, risadas e momentos compartilhados ao longo da graduação.

"A erva seca, a flor murcha. Mas a palavra do nosso Deus permanecerá para sempre."
Isaías 40:8

Resumo

O monitoramento ambiental remoto é importante sob o ponto de vista social, ecológico e agrícola, pois oferece melhor visualização do estado de um ambiente e, consequentemente, providencia o melhor controle dele. A fim de criar uma maneira de monitorar um parâmetro físico ambiental à distância, mesmo sem rede *Wi-Fi*, este trabalho propõe e detalha o desenvolvimento de um sistema de monitoramento ambiental remoto utilizando rede GSM e GPRS. Os dados de temperatura e umidade relativa do ar são lidos através de sensores, verificados e analisados pela placa Arduino UNO, e transmitidos para um servidor na *web* através de rede GPRS, podendo ser exibidos e analisados através de um *front-end* composto de páginas na *web*. O sistema utiliza um serviço de hospedagem na *web* para armazenamento dos arquivos das páginas, *scripts* e sistema de gerenciamento de banco de dados. Além da transmissão de dados através da *Internet*, caso os valores das medições físicas ultrapassem limites definidos, o usuário é automaticamente notificado através de mensagens alerta de SMS em seu telefone, utilizando a tecnologia de rede GSM.

Palavras-chave: Internet das Coisas, Arduino, Redes de dados, Sensores.

Abstract

Remote environmental monitoring is important from a social, ecological and agricultural point of view, as it offers better visibility of the state of an environment and, subsequently, provides better control over it. In order to create a way to monitor a physical environmental parameter from a distance, even without Wi-Fi, this work proposes and details the development of a remote environmental monitoring system using GSM and GPRS networks. Temperature and relative humidity air data are read using the sensors, checked and analyzed by the Arduino UNO board, and transmitted to a web server through GPRS network, which can be displayed and analyzed through a front-end composed of web pages. The system uses a web hosting service for storing the files for the pages, scripts and database management system. In addition to the data transmission through the Internet, if the physical measurement values exceed defined limits, the user is automatically notified through SMS alert messages on their phone, using GSM network technology.

Keywords: Internet of Things, Arduino, Data networks, Sensors.

Lista de figuras

Lista de abreviaturas e siglas

GSM	*Global System for Mobile Communications*
GPRS	*General Packet Radio Service*
SMS	*Short Message Service*
SMSC	*SMS Center*
MS	*Mobile Station*
ME	*Mobile Equipment*
SIM	*Subscriber Identity Module*
TDMA	*Time Division Multiple Access*
IMSI	*International Mobile Subscriber Identity*
MSISDN	*Mobile Station ISDN Number*
PDP	*Packet Data Protocol*
APN	*Access Point Name*
IoT	*Internet of Things*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
UART	*Universal Asynchronoys Receiver Transmitter*
AT	*Attention*
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
PHP	*Hypertext Preprocessor*

Sumário

1 Introdução

O monitoramento ambiental é importante em ecossistemas, tanto em escalas locais quanto mundiais. Na visão geral, problemas ambientais podem ter consequências graves e podem causar sérios problemas à saúde e segurança humana, prejuízos à biodiversidade, ciclo hidrológico e economias regionais. Incêndios florestais, por exemplo, são principalmente detectados manualmente por observação humana e envolvem ferramentas primitivas (BARMPOUTIS et al., 2020).

Pensando em um agricultor regional, saber o estado atual de sua produção, mesmo que esteja à distância, pode ser útil para tomar ações de emergência; controlar a plantação de forma mais eficiente (LOHCHAB et al., 2018), ou analisar a situação atual de forma prática e confortável a fim de aumentar sua produção.

Logo, propõe-se um sistema de monitoramento ambiental remoto através da plataforma de desenvolvimento do Arduino, componente de rede GSM (*Global System for Mobile Communications*) e sensores. O Arduino é utilizado para medir os parâmetros ambientais lidos e enviados pelos sensores, realizar análises e enviar dados e mensagens para conhecimento ou alerta de um usuário final.

Para a comunicação e transmissão de dados, é proposto utilizar um dispositivo com tecnologia de rede GSM e GPRS (*General Packet Radio Service*), que apresentam boa capacidade de rede, facilidade de itinerância mundial, baixo preço para utilização e dispensabilidade de configuração adicional, e por isto são amplamente utilizadas para comunicação ao redor do mundo (AMIN; KHAN, 2014).

1.1 Problema

O problema a ser resolvido tem foco em desenvolver uma maneira de realizar a medição dos parâmetros climáticos de temperatura e umidade relativa do ar; e permitir que os resultados lidos possam ser vistos e analisados de forma imediata por um indivíduo, mesmo em lugares onde possa não existir uma conexão direta ou consistente à *Internet*, porém em que exista sinal de telefonia celular. Este tipo de situação ocorre com frequência em locais mais distantes dos centros urbanos, onde o sinal móvel de celular pode ser captado, porém não existe conexão à *Internet* por *Wi-Fi* ou cabeamento físico.

Deseja-se observar a utilidade e aplicabilidade do sistema proposto, ou seja, se tem a capacidade de monitorar e analisar com precisão os parâmetros ambientais, transmitir os dados e mensagens necessários através da rede GPRS e GSM, e cumprir a proposta inicial.

1.2 Justificativa

A escolha da tecnologia de rede de telefonia celular (GSM e GPRS) para transmissão dos dados se deu ao fato de que é um dos sistemas de celular líderes no mundo (GEORGIEV; GEORGIEVA; SMIRIKAROV, 2004) com foco em transmissões a longas distâncias, possuindo uma infraestrutura de comunicação bem estabelecida e desenvolvida, com grande cobertura, segurança e baixo custo para implementação (MULLA et al., 2015). No Brasil, a rede de telefonia móvel está presente em todos os 5570 municípios brasileiros através das tecnologias 2G, 3G e 4G (ANATEL, 2022), evidenciando sua capacidade de cobertura e disponibilidade em diversas regiões do país.

O sistema proposto também é importante no aspecto social por auxiliar na preservação de florestas ou aperfeiçoar a produção agrícola através do monitoramento remoto.

No quesito de preservação florestal, o sistema pode, por exemplo, reduzir os problemas causados por incêncios florestais ao criar uma forma de alerta prévio, considerando que é possível identificar períodos críticos de perigo extremo de fogo antes de sua ocorrência (GROOT et al., 2006) para avisar as autoridades responsáveis, com velocidade, através de um alerta.

No âmbito do produtor agrícola, torna-se possível realizar comunicações remotas para visualização de parâmetros importantes da produção. O monitoramento preciso à distância pode providenciar o melhor controle do ambiente da plantação, de forma direta e facilmente acessível. Ao analisar os dados, também é providenciado ao produtor a capacidade de melhor tomada de decisão com relação à sua produção (LOHCHAB et al., 2018). Outra vantagem do monitoramento à distância é a facilitação da rotina do produtor, devido ao conforto, e aumento da produtividade através da confiança na tecnologia.

1.3 Objetivos

1.3.1 Objetivos Gerais

Desenvolver um sistema de monitoramento ambiental remoto utilizando Arduino, componente de rede GSM/GPRS e sensores de temperatura e umidade relativa do ar para ler e transmitir esses parâmetros em locais que tenham acesso a redes de telefonia móvel.

Os dados lidos devem ser enviados a um servidor na *web* utilizando a tecnologia GPRS do componente de rede GSM/GPRS e podem ser consultados através de um *website* hospedado na *Internet*.

1.3.2 Objetivos Específicos

- Estudar conceitos de telemetria, IoT (*Internet of Things*), Rede GSM e GPRS.

- Definir e analisar componentes eletrônicos utilizados para desenvolvimento do projeto.
- Análise de *hardware* para conexão e interface do Arduino com os módulos definidos.
- Realizar testes de interface de cada componente e sua comunicação com o Arduino.
- Mostrar ferramentas de desenvolvimento *web* para a construção de *websites*, busca e visualização de dados armazenados na *Internet*.
- Escolher serviço de *web hosting* para hospedar a base de dados que armazena os dados lidos, explicar seu funcionamento e o *website* para a visualização.
- Montar o circuito eletrônico físico do sistema, interligando seus componentes.
- Desenvolver código do software de integração geral de todos os componentes do sistema para o Arduino.
- Testar o sistema desenvolvido, verificar sua funcionalidade e cumprimento da proposta.

1.4 Organização da monografia

Os capítulos seguintes apresentam cada etapa do projeto:

- Capítulo 2: **Fundamentação teórica**, detalha os conceitos e tecnologias que servem de base para a metodologia aplicada.
- Capítulo 3: **Metodologia**, apresenta de forma específica os componentes eletrônicos utilizados no sistema e suas características. Também introduz as ferramentas de desenvolvimento e hospedagem utilizadas no projeto.
- Capítulo 4: **Desenvolvimento**, explica não apenas a arquitetura do projeto e como seus componentes são interligados, mas também o desenvolvimento do código e *scripts* responsáveis por ler, analisar, transmitir, armazenar e visualizar os dados lidos.
- Capítulo 5: **Conclusão**, discorre sobre as conclusões obtidas em todas as etapas do projeto, verifica se cumpre a proposta e aponta possíveis melhorias.

2 Fundamentação teórica

2.1 IoT

A IoT é a sigla para *Internet of Things* (Internet das Coisas), e surge da ideia de que as conexões na *Internet* não serão mais feitas apenas de humanos para humanos, mas que outras coisas possam ser conectados e trocar informações (LAN; WANG, 2010). Essas "coisas", ou entidades da IoT se tratam de aparelhos que são aumentados com microcontroladores, transmissores ópticos ou de rádio, sensores, atuadores e protocolos de rede adequados para comunicação em ambientes controlados, onde o *hardware* tem recursos limitados, mas que permitem que eles obtenham dados destes ambientes e ajam sobre eles, oferecendo a eles uma interface com o mundo físico (CIRANI et al., 2018).

Um dos maiores avanços da IoT é juntar o mundo físico e o mundo das informações. Os sensores têm um grande papel para criar uma ponte entre os dois mundos ao coletar dados de seus ambientes e gerar conhecimento sobre eles através do contexto em que estão instalados. Como consequência, o ambiente pode ser monitorado e ações podem ser tomadas em cima dos resultados oferecidos (LAN; WANG, 2010).

O armazenamento de dados, processamento e análises são elementos fundamentais e necessários para enriquecer o dado oferecido pelo objeto de IoT e transformá-lo em informação útil, desta forma, a adesão da computação nestes cenários é essencial e pode trazer diversos benefícios, como: baixa latência, conhecimento do contexto inserido e capacidade de ação em tempo-real (CIRANI et al., 2018).

As aplicações com conceitos de IoT são encontradas em diversas situações (CIRANI et al., 2018), como:

- Automação de casas e prédios;
- *Smart cities* e *Smart grids*;
- Monitoramento industrial;
- Agricultura inteligente.

2.1.1 HTTP

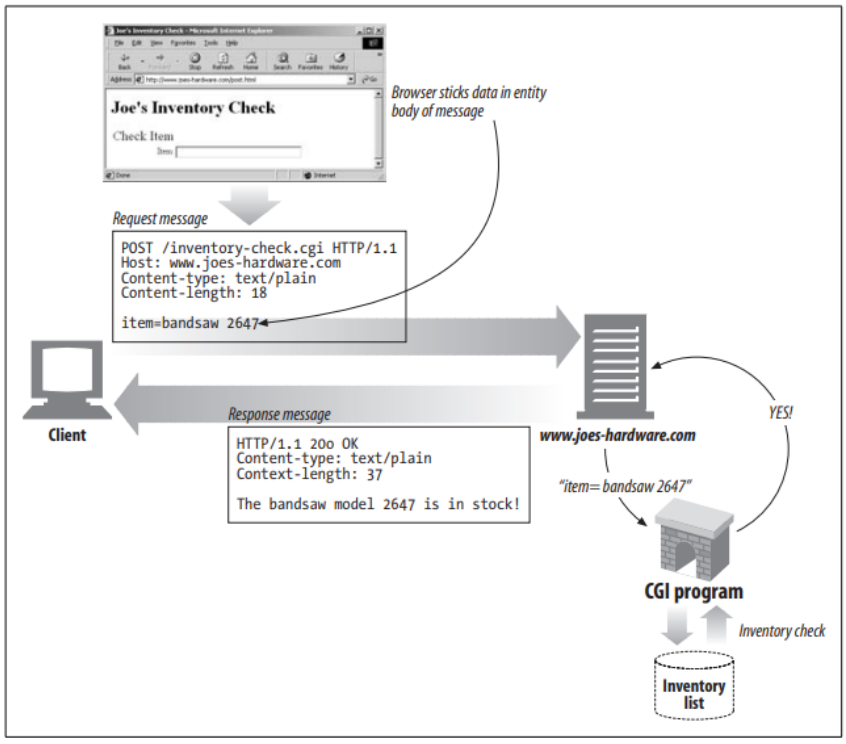
O maior motivo da popularidade da *Internet* é a configuração e desenvolvimento de seus protocolos da camada de aplicação, que são disponíveis aos desenvolvedores como um meio efetivo para transmitir a informação de diversos pontos de aplicações sem precisar especificar

algum tipo de protocolo personalizado. Para realizar esta transmissão de dados para a *Internet*, que impacta as aplicações de IoT, o mais popular protocolo utilizado é o *Hypertext Transfer Protocol* (HTTP) (CIRANI et al., 2018).

O HTTP é um protocolo baseado em requisições ou respostas em texto e define a comunicação entre o cliente e o servidor em uma conexão TCP (*Transmission Control Protocol*). Ele é caracterizado, principalmente, por utilizar endereços de redes para recursos que são alvo de requisições; utilizar campos no cabeçalho para adicionar mais informações às mensagens; e utilizar métodos específicos para o tipo de atividade a ser executada pelo servidor (CIRANI et al., 2018). Estes métodos são configurados pelo servidor, logo, podem variar de *site* a *site*, e se tratam de solicitações que pedem algo ao servidor (GOURLEY et al., 2002). Os mais comumente utilizados são:

- GET: Solicita ao servidor que envie um recurso.
- POST: Utilizado para enviar dados de entrada ao servidor para serem processados, comumente usado para suportar formulários (Exemplo na Figura 1).
- PUT: Escreve documentos para o servidor, é o inverso do GET.
- DELETE: Solicita ao servidor que exclua os recursos especificados no endereço da rede.

Figura 1 – Esquema do método de POST



Fonte: Gourley et al. (2002)

Os métodos dizem ao servidor o que fazer, e este responde ao cliente com uma mensagem de *status* sobre o que ocorreu com a mensagem enviada (GOURLEY et al., 2002).

2.2 GSM

O *Global System for Mobile Communication* (GSM) se trata de um padrão de rede de comunicação digital para aparelhos móveis (BODIC, 2005). Seu desenvolvimento se iniciou na década de 1980, durante o comitê *Groupe Spéciale Mobile* da organização europeia *Européenne des Postes et Télécommunications* (CEPT) (HEINE, 1999). O objetivo era substituir os modelos de redes móveis implementados até então, que eram incompatíveis e permitir o *roaming* (capacidade de utilizar o aparelho móvel fora da localidade onde foi registrado e se conectar a uma rede de celular) dos usuários de aparelhos móveis além das fronteiras internacionais. O foco do comitê era padronizar uma rede de comunicação celular pública na banda de rádio na banda de rádio de 900 MHz utilizando tecnologia digital (BODIC, 2005).

Com o passar dos anos, foram surgindo derivados do GSM, como o *Digital Cellular System 1800* (DCS 1800), que representava o sistema GSM em uma banda de frequência de 1800 MHz; o *Personal Communication System 1900* (PCS 1900), que foi adaptado para uma banda de 1900 MHz nos Estados Unidos. Nos anos 1990, com novos recursos adicionados, o GSM começou a se tornar uma atração mundial para fabricantes de sistema e operadoras de rede (HEINE, 1999).

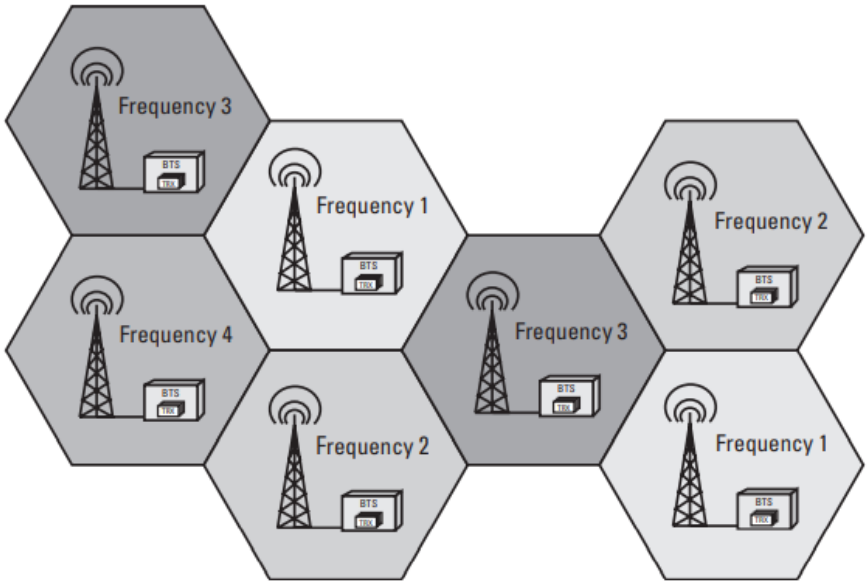
A tecnologia do GSM se populariza na segunda geração de telefonia móvel (2G), que foi responsável por popularizar serviços de voz de maior qualidade para as massas e se beneficia do conceito celular, em que recursos de rádio escassos são usados simultaneamente por diversos usuários de aparelhos móveis sem nenhum tipo de interferência. Após o 2G, surge o 2.5G, que se caracteriza pela introdução da tecnologia GPRS e serve de transição para a popularização da terceira geração (3G) em 2004. O 3G possibilitou a convergência das tecnologias móveis com a *Internet* e adicionou diversas opções de multimídia, custo-benefício e suporte para mobilidade mundial (BODIC, 2005).

2.2.1 Funcionamento e arquitetura

O modo de acesso da rede GSM é baseado em *Time Division Multiple Access* (TDMA), que permite o compartilhamento da banda de rádio entre diversos usuários ao alocar um ou mais intervalos de tempo para cada assinante. No modo de configuração de intervalo único, chamado de *Circuit Switched Data* (CSD), é possível atingir fluxos de transferência de bits de até 14.4 Kbps; já na configuração de múltiplos intervalos de tempo, chamada de *High Speed CSD* (HSCSD), é possível atingir uma taxa de transferência de 57.6 Kbps, considerando que é alocado mais de um intervalo de tempo para a conexão (BODIC, 2005).

O GSM/GPRS utiliza uma estrutura celular de rede. O conceito deste tipo de rede é particionar o espectro de rede disponível, de forma que apenas parte desta faixa de rede possa ser atribuída a uma estação de rádio, consequentemente, reduzindo o alcance desta estação para reutilizar as frequências escassas o máximo possível (HEINE, 1999). Desta maneira, cada célula tem uma frequência diferente das outras e os recursos de rádio podem ser utilizados simultaneamente por diversos assinantes sem interferência (Figura 2). Em uma rede GSM, quanto menor o tamanho das células, maior o fator de frequência de reuso (BODIC, 2005).

Figura 2 – Cobertura de rádio de uma área dividida em células

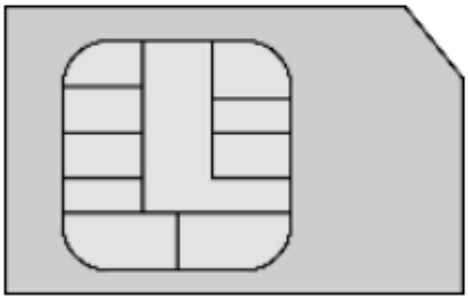


Fonte: Heine (1999)

Nesta arquitetura de rede, o *Mobile Station* (MS) é o aparelho que envia ou recebe sinais de rádio dentro de uma célula. O MS é composto de um *Mobile Equipment* (ME) e um *Subscriber Identity Module* (SIM). O ME se trata do aparelho físico em que o SIM é alocado. Além disso, possui um código chamado *International Mobile Equipment Identity* (IMEI), que, unicamente, identifica o aparelho quando conectado a uma rede móvel (BODIC, 2005).

O SIM é um cartão (Figura 3) providenciado pela operadora de rede para que o usuário possa ser identificado (BODIC, 2005). Ele transforma o ME em um MS com um conjunto de serviços de rede que podem ser utilizados através de uma assinatura. É possível se registrar a uma rede local com o cartão SIM utilizando MEs diferentes, o que permite o *roaming* internacional independente de equipamentos físicos ou tecnologia de rede, com a ressalve de que o modo de acesso na rede visitada tenha suporte ao GSM. Outros recursos importantes do SIM são: a capacidade de armazenamento interno de dados de rede e funções de segurança, como algoritmos de criptografia (KUKUSHKIN, 2018).

Figura 3 – Cartão SIM



Fonte: Bodic (2005)

Ao assinar um serviço com uma operadora de redes móveis, cada usuário recebe um identificador único chamado *International Mobile Subscriber Identity* (IMSI). Ele é armazenado no cartão SIM, raramente transmitido pela interface no ar e conhecido apenas dentro de uma rede GSM (BODIC, 2005), servindo para proteger sua confidencialidade e do usuário (KUKUSHKIN, 2018). Ao contrário do IMSI, que é confidencial, o *Mobile Station ISDN Number* (MSISDN) serve para identificar um usuário fora de uma rede GSM (BODIC, 2005). O MSISDN é utilizado como identificador para ligações telefônicas e segue a seguinte estrutura (KUKUSHKIN, 2018):

- *Country Code* (CC), código de país de até 3 dígitos;
- *National Destination Code* (NDC), código regional dentro de um país de 2 a 3 dígitos;
- *Subscriber Number* (SN), código de identificação do assinante na rede, com máximo de 10 dígitos.

O responsável por cuidar das tarefas de rádio e providenciar a conectividade entre a rede e os MSs através da interface do ar é o *Base Transceiver Station* (BTS) (HEINE, 1999). O BTS é instalado com infraestrutura adicional que inclui antenas, fonte de alimentação e equipamentos de transmissão (KUKUSHKIN, 2018). As atividades incluem modularização de sinal, igualação de sinal e codificação de erros. Cada BTS realiza as comunicações com os MSs localizados dentro da sua área de cobertura (célula) (BODIC, 2005).

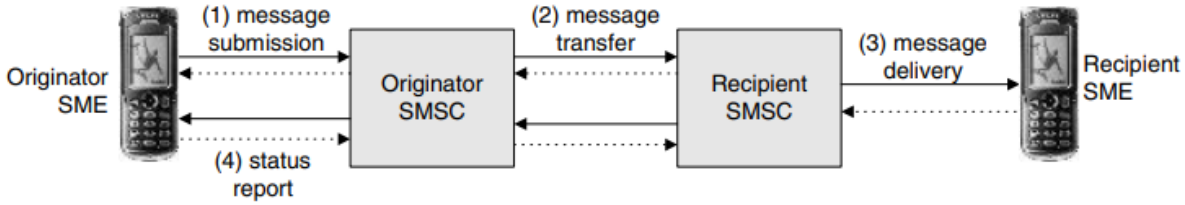
2.3 SMS

O SMS (*Short Message Service*) é um dos serviços utilizados neste projeto, e permite que seus assinantes (estações móveis ou outros tipos de aparelhos conectados à rede) troquem mensagens de texto curtas através da tecnologia GSM/GPRS. Inicialmente foi introduzido na segunda geração (2G), e foi posteriormente portado para o GPRS (BODIC, 2005).

O *SMS Center* (SMSC) é indispensável e o responsável por gerenciar as operações de SMS em uma rede sem fio. Ele tem a função de retransmitir e armazenar as mensagens entre os MSs. A vantagem é que mesmo se ocorrer uma falha no aparelho receptor, a mensagem fica armazenada até que possa ser enviada (BODIC, 2005).

Normalmente, as operadoras de redes móveis têm acordos comerciais mútuos para permitir a troca de mensagens entre redes diferentes. Isto significa que uma mensagem curta enviada de um aparelho conectado à uma rede possa chegar a outro aparelho conectado em outra rede. A capacidade de enviar mensagens para redes em que o usuário não é cadastrado é um dos recursos principais do SMS e por que ele é tão utilizado. Neste recurso, apresentado na Figura 4, o SMSC originador é o responsável por transmitir a mensagem para o SMSC receptor, através de um protocolo, funcionando como um meio-termo (BODIC, 2005).

Figura 4 – Transferência de mensagens entre redes diferentes



Fonte: Bodic (2005)

2.4 GPRS

O *General Packet Radio Service* (GPRS) é uma extensão do GSM que permite que os usuários transmitam e recebam dados através de conexões de comutação de pacotes (BODIC, 2005). Neste modo de conexão, ao contrário do GSM, tem-se o conceito de compartilhamento dos canais disponíveis na célula para diversos usuários. Desta forma, um intervalo de tempo não é reservado exclusivamente o tempo todo para um usuário, mas pode ser compartilhado entre diversos usuários de acordo com a prioridade (KUKUSHKIN, 2018).

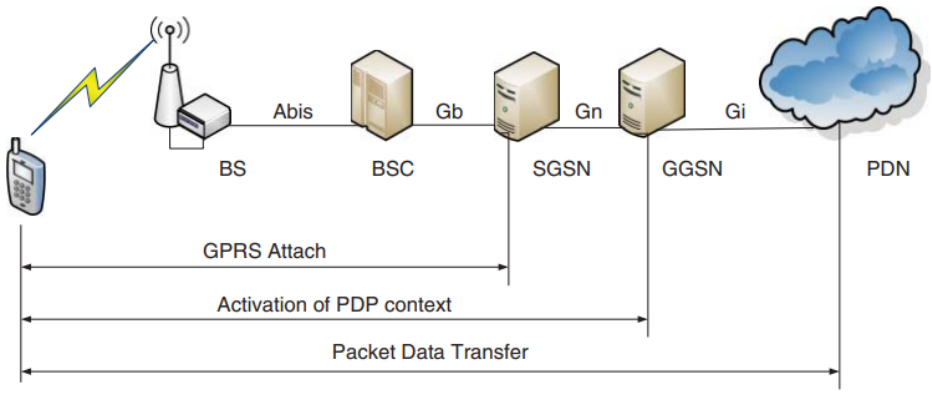
O conceito básico da transmissão do GPRS está na habilidade de permitir que recursos de rádio sejam alocados para transmissão apenas quando as aplicações têm dados para serem transmitidos. Após os dados serem transmitidos, os recursos de rádio são liberados para outras aplicações. Como consequência, os recursos de rádio escassos são usados de forma mais eficiente neste tipo de transmissão, podendo atingir taxas de transferência de bits maiores (BODIC, 2005).

A interface no ar do GPRS é igual à do GSM, ambos têm a mesma modulação de rádio, bandas de frequência e estrutura, porém, o GPRS necessita de um subsistema com dois

elementos de rede adicionais: nós servidor e *gateway* de GPRS. O nó servidor é chamado de *Serving GPRS Support Node* (SGSN), e se conecta a uma ou mais subsistemas de um BTS, operando como um roteador para os pacotes de dados de todos os MS presentes em uma área, além de realizar funções de segurança e acesso. Já o nó de *gateway* é apelidado de *Gateway GPRS Support Node* (GGSN), e providencia o ponto de ligação entre o domínio do GPRS e outras redes de dados como a *Internet*. Para estabelecer a conexão com a rede de destino, é necessário um *Access Point Name* (APN) (BODIC, 2005).

Antes do MS conseguir acessar os serviços do GPRS, é preciso executar um procedimento chamado de *GPRS attachment*, em que ele é ligado à rede para indicar a sua presença (BODIC, 2005). Ao ser apenso à rede, o MS se move para o estado pronto, é autenticado, uma chave criptográfica é gerada e é alocado ao MS uma identidade lógica temporária (KUKUSHKIN, 2018). Após o *attachment*, para poder transferir e receber dados, deve ser ativado o contexto do *Packet Data Protocol* (PDP) (BODIC, 2005). Neste processo de ativação, a rede providencia um endereço de IP para o MS, que pode ser dinâmico ou estático, dependendo da configuração da rede (KUKUSHKIN, 2018). Um esquema geral da arquitetura do GPRS e seus procedimentos pode ser visto na Figura 5.

Figura 5 – Procedimentos GPRS



Fonte: Kukushkin (2018)

3 Metodologia

Neste capítulo, são apresentados, de forma mais detalhada, os componentes físicos que compõem o sistema e as ferramentas de *software* utilizadas para o desenvolvimento do projeto de monitoramento remoto utilizando rede GSM e GPRS.

3.1 Componentes eletrônicos

Para desenvolver o sistema de IoT, existem 3 partes principais. O módulo de processamento responsável por programar a interface entre os sensores e o módulo de rede, e também processar e analisar as informações; o componente sensor, que obtém as informações físicas do ambiente em que está inserido e as comunica; e o componente de rede, responsável por transmitir as informações adquiridas à *network* para que foi configurado.

3.1.1 Arduino

O Arduino é um módulo de processamento que pode ser programado para processar entradas e saídas entre o aparelho e os componentes externos que são conectados a ele. Mais conhecido como uma plataforma de computação, através dele, é possível interagir com o ambiente em que está inserido através do uso de *hardware* e *software* (MCROBERTS, 2010).

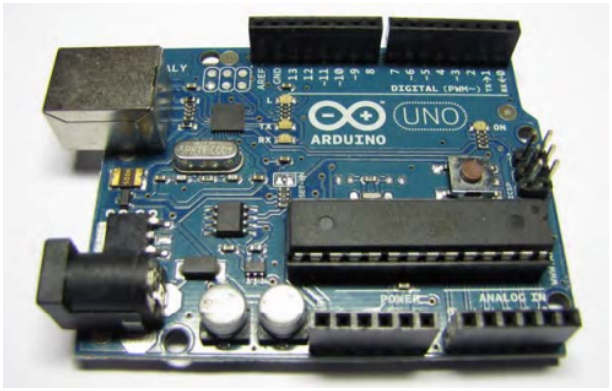
Ele pode ser utilizado para desenvolver objetos interativos individuais, ou podem ser conectados a outro computador ou rede para enviar e receber dados, e então agir sobre eles (MCROBERTS, 2010). Através dessa interação, é possível enviar dados de um ambiente físico específico para a *Internet*, e visualizá-los à distância, em tempo real.

O Arduino UNO (Figura 6) foi o modelo de placa escolhido para o projeto por atender plenamente às necessidades do projeto proposto e seu custo benefício. Ele tem as seguintes especificações:

- Microcontrolador ATmega16U2, utilizado para converter sinais USB para serial (EVANS; NOBLE; HOCHENBAUM, 2013);
- Microcontrolador ATmega328P, responsável por executar os comandos do Arduino;
- 14 pinos digitais de entrada e saída;
- 6 pinos analógicos de entrada;
- Voltagem de entrada e saída de 5V, com 20mA de corrente por pino;
- 16MHz de velocidade de *clock*;

- Conector USB (utilizado para transmitir e enviar dados USB);
- Conector para fonte de alimentação externa.

Figura 6 – Arduino UNO



Fonte: McRoberts (2010)

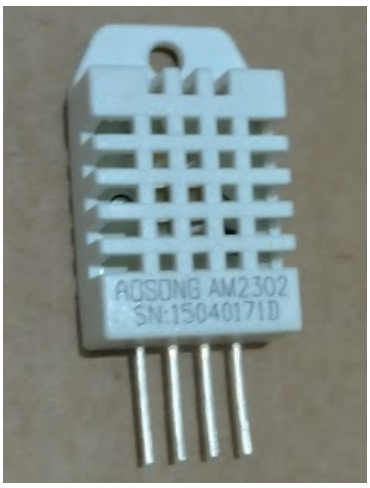
Tanto o *hardware* como o *software* do Arduino são *open source* (código aberto), logo, o desenvolvimento para a plataforma pode ser feito de forma livre por qualquer pessoa (MCROBERTS, 2010). Este fato está de acordo com o intuito inicial do Arduino, que é ser um sistema de baixo custo, fácil de usar, e que possa servir de introdução para a programação para pessoas que querem desenvolver seus próprios projetos (EVANS; NOBLE; HOCHENBAUM, 2013).

3.1.2 DHT22

O componente eletrônico escolhido para fazer a leitura dos parâmetros de temperatura e umidade relativa do ar foi o DHT22, também chamado de AM2302 (Figura 7), que consegue ler ambos parâmetros. A escolha dos parâmetros se deu pelo fato de que eles cumprem, de forma geral, a proposta do projeto. Possuindo diversas aplicações, desde monitoramento de agricultura, controle de irrigação (YAMAZOE; SHIMIZU, 1986), estação meteorológica, controle em ambientes controlados (AOSONG, 2018a), entre outros. O módulo físico foi escolhido devido ao custo-benefício (considerando preço e precisão), resposta rápida, interface simples de apenas um pino para troca de informações, tamanho pequeno e o fato de que os sensores já vêm calibrados (AOSONG, 2018a).

A leitura da umidade relativa do ar se dá através de um componente sensor de umidade. Como pode ser visto na Figura 8, ele possui dois eletrodos (um na base, outro no topo), e entre eles um substrato que tem a capacidade de segurar a umidade dentro dele. Quando a umidade muda, a condutividade do substrato se altera e a resistência entre os eletrodos muda. A mudança é medida e processada pelo sensor, e depois enviada ao Arduino (DEJAN,

Figura 7 – DHT22

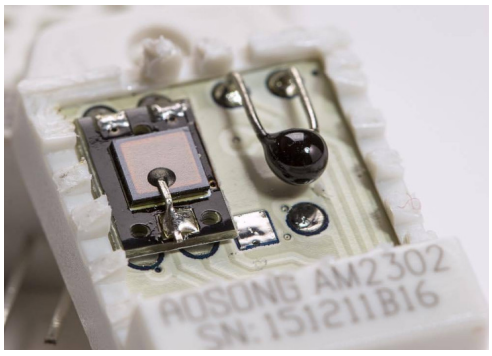


Fonte: Elaborada pelo autor.

2016). Para medir a temperatura do ar, utiliza-se um termistor. Este componente é um resistor variável que muda a sua resistência com a variação de temperatura. Eles são feitos de materiais semicondutores como cerâmicas ou polímeros para providenciar grandes mudanças na resistência com apenas pequenas mudanças na temperatura. Neste caso, trata-se de um NTC (*Negative Temperature Coefficient*), em que a resistência diminui com o aumento da temperatura (DEJAN, 2016).

Uma visão interna do DHT22, mostrando ambos componentes de sensor, pode ser observada na Figura 8.

Figura 8 – Visão interna do DHT22



Fonte: Avelar (2018)

O sensor possui uma interface de apenas um pino para transferência de dados. De acordo com a atribuição dos pinos, temos: dois pinos para alimentação, e um pino para transferência de dados. A transferência de dados ocorre de forma serial e bi-direcional através de um protocolo de comunicação. Ele funciona na estrutura de *master-slave*, em que o microcontrolador chama o sensor, para depois o sensor enviar sua resposta. Primeiramente, o

microcontrolador envia um sinal de iniciação, para depois receber o signal de resposta, seguido dos bits seriais dos parâmetros lidos (AOSONG, 2018b).

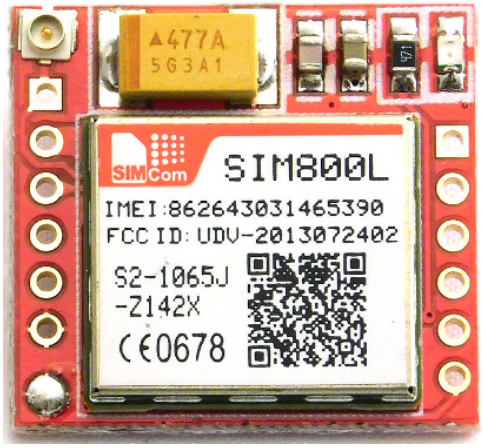
A resposta lida pelo sensor vem em 5 bytes. O primeiro byte corresponde à parte inteira do valor de umidade lido; o segundo à sua parte decimal; o terceiro corresponde à parte inteira do valor de temperatura do ar lido; e o quarto à sua parte decimal. O último byte é um *checksum* dos dados e verifica se a leitura foi correta (AOSONG, 2018b).

3.1.3 SIM800L

A fim de fazer a transmissão de dados para o servidor da *web* e notificar os usuários através de SMS, neste projeto, utiliza-se o módulo SIM800L (Figura 9). O módulo é desenvolvido pela SIMCom e opera em quatro bandas de telefonia (850, 900, 1800 e 1900MHz) (SIMCOM, 2013). Listando algumas das principais características:

- Opera nas quatro bandas de telefonia (850, 900, 1800 e 1900 MHz);
- Possui uma entrada para um cartão SIM micro;
- Consegue se conectar a qualquer rede GSM com qualquer SIM;
- Capaz de fazer e receber chamadas de voz (se utilizado *speaker* ou microfone externos);
- Manda e recebe mensagens de SMS;
- Transmite e recebe dados GPRS (como TCP/IP ou HTTP);
- Escanea e recebe transmissões de rádio FM.

Figura 9 – SIM800L



Fonte: Nettigo (2016)

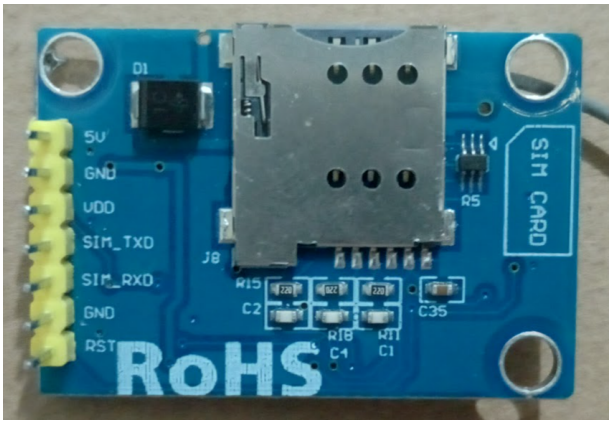
Para o projeto, foi escolhido utilizar a *evaluation board* SIM800L V2 (Figuras 10 e 11), ou SIM800L EVB, no lugar do SIM800L comum. A *evaluation board* é um sistema construído em volta de uma parte específica, nesse caso, o V2. A escolha foi feita porque o SIM800L V2 já vem com pinos soldados, antena e possui um pino de entrada de 5 volts. O SIM800L padrão funciona na faixa de 3.3 a 4.4V, já a *evaluation board* tem um tradutor de nível de tensão para derrubar o nível desta entrada. Logo, a *board* é mais adequada e prática para ser utilizada no projeto com Arduino.

Esse chip possui sete pinos: dois para alimentação (5v e GND), quatro para a comunicação serial (VDD, SIM_TXD, SIM_RXD e GND) e um pino de hard reset (RST). Neste caso, os pinos de serial usados são o SIM_RXD (*receiver*, envia comandos para o módulo) e o SIM_TXD (*transmitter*, transmite comandos do módulo para o Arduino). A placa possui um *slot* para entrada do cartão SIM, que é usado para a rede GSM/GPRS.

Figura 10 – SIM800L V2, vista superior

Fonte: Elaborada pelo autor.

Figura 11 – SIM800L V2, vista inferior

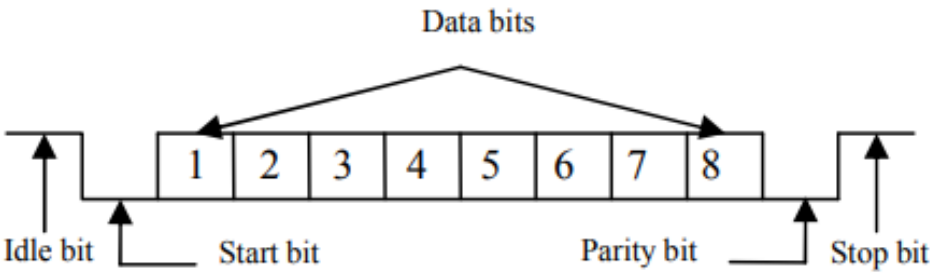


Fonte: Elaborada pelo autor.

Para realizar a comunicação com o Arduino, o módulo utiliza o protocolo de comuni-

cação UART (Universal Asynchronoys Receiver Transmitter). Neste protocolo, a comunicação é assíncrona, as informações são enviadas serialmente (bit a bit) e ocorre a sincronização do *clock* do transmissor com o do receptor a cada byte (8 bits) (FANG; CHEN, 2011). Pode-se observar o formato de quadro na Figura 12.

Figura 12 – Formato de quadro do UART



Fonte: Fang e Chen (2011)

A programação do SIM800L acontece através de comandos AT (abreviação de *Attention*), que são cadeias de caracteres que servem como identificador de comando. Estes comandos AT são enviados através do pino serial para o módulo de rede GSM, que vai interpretar e realizá-los. Após a possível realização do comando, o módulo SIM800L irá seguir com uma resposta, que aponta se o comando foi bem-sucedido, se houve erro, ou então retornar uma outra saída requisitada através do comando (SIMCOM, 2015). Um exemplo de sintaxe e possíveis respostas de comando AT para o SIM800L pode ser observado na Figura 13.

Figura 13 – Exemplo de comando AT

AT+HTTPINIT	**Initialize HTTP Service**
Test Command **AT+HTTPINIT=?**	Response **OK**
Execution Command **AT+HTTPINIT**	Response **OK** If error is related to ME functionality: **+CME ERROR: <err>**
Parameter Saving Mode	NO_SAVE
Max Response Time	-
Reference	Note HTTPINIT should first be executed to initialize the HTTP service.

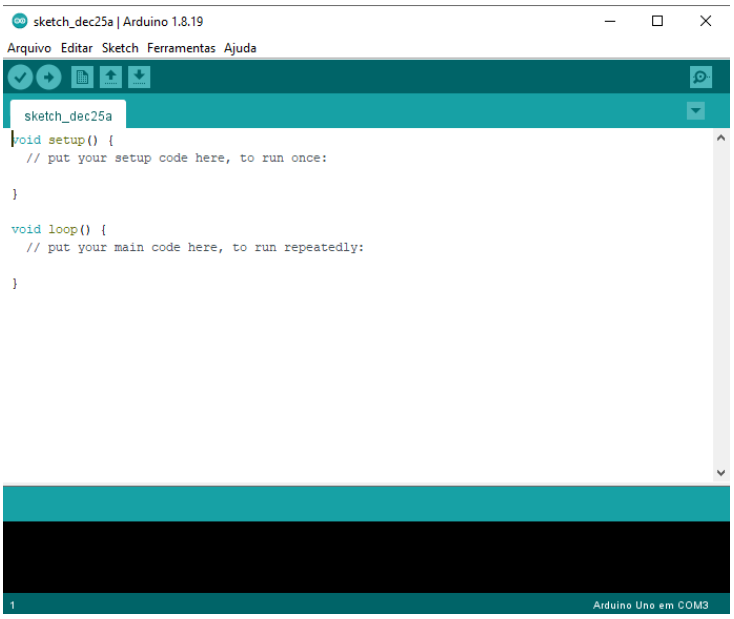
Fonte: SIMCom (2015)

3.2 Ferramentas utilizadas

3.2.1 Arduino IDE

Para programar o Arduino, é necessário utilizar o ambiente de desenvolvimento do Arduino, o Arduino IDE (*Integrated Development Environment*). Se trata de um software grátis, em que é possível escrever código na linguagem C++, entendida pelo Arduino. Após escrito, o código é verificado pela IDE e carregado para o Arduino através da interface USB. No Arduino IDE, os programas são chamados de *sketches* (esboços) (MCROBERTS, 2010). A visão básica da IDE se encontra na Figura 14.

Figura 14 – Arduino IDE



Fonte: Elaborada pelo autor.

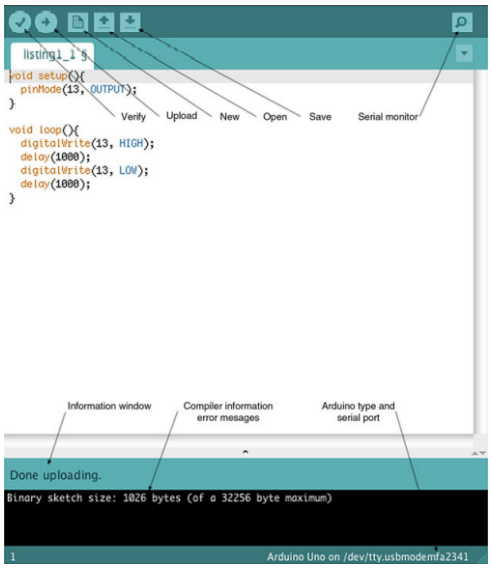
Ao carregar a IDE pela primeira vez, ele inicia com um *sketch* vazio, com apenas as funções de *setup* e *loop*.

A função de *setup* se trata de uma rotina que é executada apenas uma vez, na inicialização do programa. Nesta rotina, geralmente, são colocados comandos de inicialização de objetos, configuração de pinos digitais (entrada ou saída) e escolher uma *baud rate* (frequência com que o sinal muda) para a comunicação serial (EVANS; NOBLE; HOCHENBAUM, 2013). Em contrapartida, a rotina de *loop* é executada após o *setup*, e continua executando continuamente, a não ser que seja interrompida por alguma condição ou o Arduino seja desligado (EVANS; NOBLE; HOCHENBAUM, 2013).

Na barra de ferramentas da IDE, encontra-se funções para verificação de erros no *sketch* e carregamento para a memória física da placa do Arduino, além da opção do *Serial Monitor*. Um exemplo de código utilizando as funções de *setup* e *loop*, além dos componentes

da IDE, encontra-se na Figura 15.

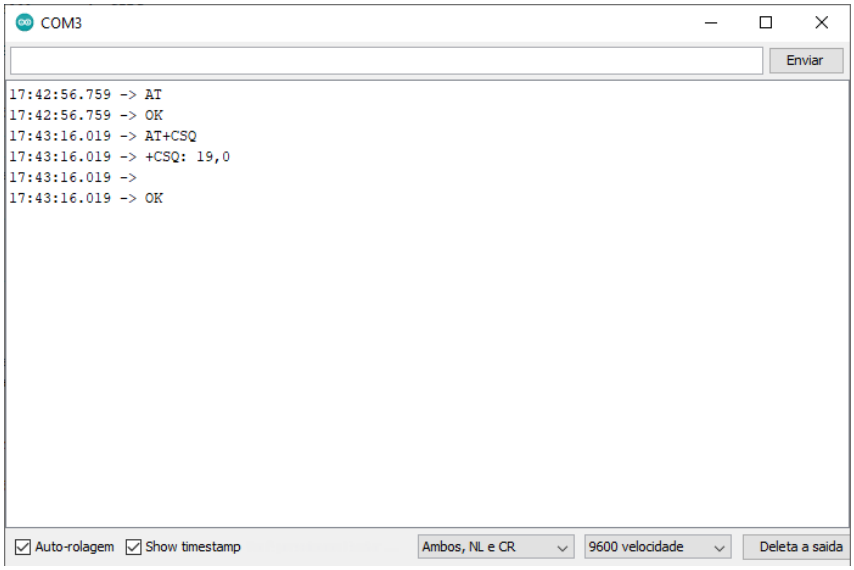
Figura 15 – Exemplo de código e posicionamento das informações



Fonte: Evans, Noble e Hochenbaum (2013)

O *Serial Monitor* é um monitor de comunicação serial, que monitora os dados que são transferidos entre o Arduino e o computador conectados pelo cabo USB. Ao utilizá-lo, pode-se visualizar, enviar e receber informações utilizando código (EVANS; NOBLE; HOCHENBAUM, 2013). Ele é a ferramenta base utilizada para o *debugging* (depuração) do código. No canto inferior direito, é possível selecionar a *baud rate* com que os dados seriais são enviados ou recebidos pelo Arduino. A aparência do *Serial Monitor* pode ser vista na Figura 16.

Figura 16 – Monitor serial do Arduino



Fonte: Elaborada pelo autor.

3.2.2 Linguagens web

Nesta subseção, são introduzidas as linguagens, ferramentas e serviços de desenvolvimento *web* utilizadas para a construção do *website* que exibe as informações de monitoramento do sistema. No desenvolvimento, são utilizadas ferramentas de *back-end* para o armazenamento de dados e respostas às solicitações do cliente; e também de *front-end* para desenvolvimento da interface e aparência das páginas *web*.

No quesito hardware, o servidor *web* armazena todos os arquivos relacionados ao *website* e a base de dados; já no aspecto de software, o servidor contém diversos componentes que controlam como os arquivos e dados são acessados pelos usuários (MOZILLA.ORG, 2022b).

3.2.2.1 HTML

O *HyperText Markup language* (HTML) é uma linguagem utilizada para descrever a estrutura das páginas *web* (DUCKETT, 2011). O "*Hypertext*"se refere aos *links* que interconectam as páginas; já o "*markup*"se refere às cadeias de texto (também chamadas de *tags*) que podem ser inseridas para representar como aquele trecho de texto deve ser exibido no navegador da *web*.

Um exemplo de estruturação de páginas e utilização de *tags* pode ser observado na Figura 17.

Figura 17 – Exemplo de código HTML

```
<html>
  <body>
    <h1>This is the Main Heading</h1>
    <p>This text might be an introduction to the rest of
       the page. And if the page is a long one it might
       be split up into several sub-headings.</p>
    <h2>This is a Sub-Heading</h2>
    <p>Many long articles have sub-headings to help you
       follow the structure of what is being written.
       There may even be sub-sub-headings (or lower-level
       headings).</p>
    <h2>Another Sub-Heading</h2>
    <p>Here you can see another sub-heading.</p>
  </body>
</html>
```

Fonte: Duckett (2011)

3.2.2.2 CSS

A linguagem de Folhas de Estilo em Cascata (*Cascading Style Sheets*) conhecida como CSS é uma linguagem de estilo usada para descrever a apresentação de um documento escrito em HTML. A linguagem descreve como os elementos devem ser exibidos na tela. Ele pode ser utilizado, por exemplo, para mudar as cores e tamanhos dos elementos da tela; criar um

layout; ou até mesmo criar efeitos e animações visuais (MOZILLA.ORG, 2022c). Um exemplo de uso de CSS externo pode ser observado na Figura 18.

Figura 18 – Exemplo de código CSS

```
body {
    font-family: arial;
    background-color: rgb(185,179,175);}
h1 {
    color: rgb(255,255,255);}
```

Fonte: Duckett (2011)

3.2.2.3 PHP

PHP é um acrônimo para *Hypertext Preprocessor*, e se trata de uma linguagem de *scripting* de código aberto, utilizada para desenvolvimento *web*, podendo ser embutida ao HTML. No PHP, os *scripts* são executados no lado do servidor, gerando HTML que depois é enviado ao cliente (PHP.NET, 2015b).

Utilizando PHP, é possível gerar conteúdo de página dinâmico, gerenciar dados no lado do servidor, coletar dados de formulário, controlar acesso de usuário, entre outros. Um exemplo de trecho de código PHP pode ser visto na Figura 19.

No projeto de monitoramento, a linguagem é utilizada para receber os dados que são transmitidos pelo SIM800L, criar a conexão com o banco de dados e realizar *queries* de dados especificados pelo usuário.

Figura 19 – Exemplo de código PHP

```
<?php

$query  = "SELECT id, name, inserted, size FROM products
           WHERE size = '$size'";
$result = odbc_exec($conn, $query);

?>
```

Fonte: Php.net (2006)

3.2.3 000webhost

A fim de deixar o conteúdo do site do projeto disponível não apenas localmente, mas na *Internet*, foi utilizado um serviço de hospedagem na rede (*web hosting*). Ao utilizar este

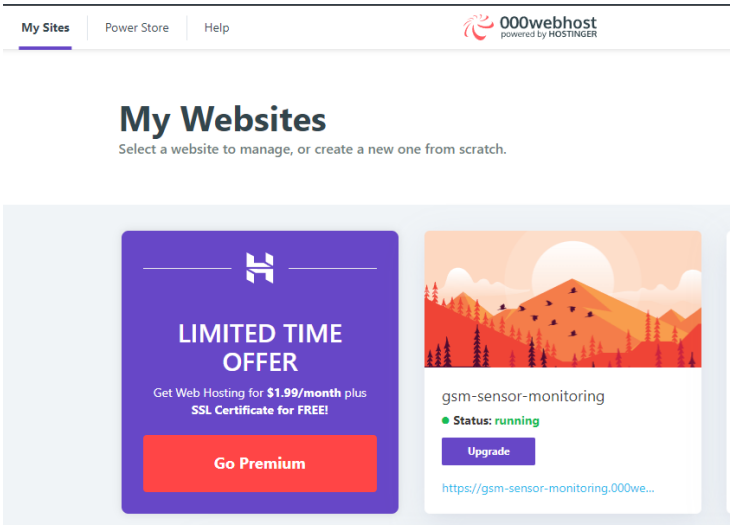
tipo de serviço, o desenvolvedor está alugando espaço em um servidor físico para armazenar os dados e arquivos do *website*.

O servidor que hospeda o *website* é um computador físico que executa continuamente para que o *site* esteja disponível para todos os visitantes o tempo todo. Assim que um usuário acessa o endereço do *site* em seu navegador, o servidor do *web host* irá transferir os arquivos necessários para carregar o *website* (DOMANTAS, 2022).

É possível hospedar um site por contra própria, porém é necessário criar um servidor *web* do zero, incluindo o *hardware*, *software*, equipamento e infraestrutura necessários, além de lidar com manutenção (DOMANTAS, 2022). Logo, para o projeto de monitoramento remoto, foi decidido utilizar um sistema de *web hosting* para simplificar os aspectos complexos de hospedagem, focando apenas na parte de configuração do back-end, base de dados e aspectos visuais das páginas hospedadas.

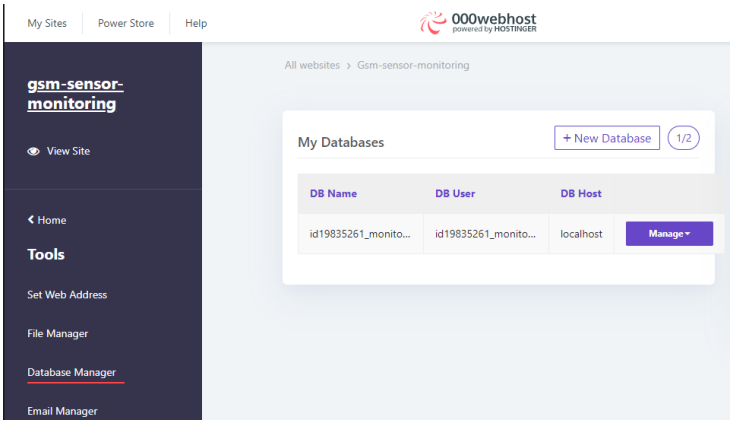
O serviço de *web hosting* escolhido para o trabalho foi o 000webhost (Figura 20), da empresa Hostinger (HOSTINGER, 2009). O serviço foi preferido por ter um plano grátis e incluir ferramentas de gerenciamento de base de dados e hospedagem de arquivos do servidor (Figura 21).

Figura 20 – Aparência geral do 000webhost



Fonte: Elaborada pelo autor.

Figura 21 – Ferramentas do 000webhost

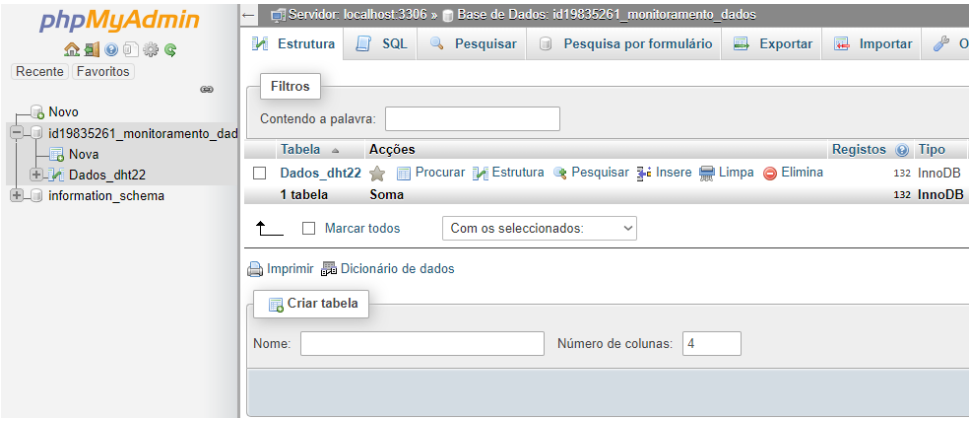


Fonte: Elaborada pelo autor.

O sistema de gerenciamento de banco de dados utilizado no serviço é o MySQL, desenvolvido pela Oracle. É um sistema de código aberto utilizado para implementar e gerenciar bancos de dados, como também realizar *queries* (MYSQL.COM, 2012). Ele é baseado no SQL (Structured Query Language), apresentando bases de dados relacionais, logo, é possível configurar os relacionamentos entre os diferentes campos das tabelas criadas. O formato dos bancos de dados relacionais também permite que ações como obtenção, atualização e agregamento de dados sejam facilitadas.

Para lidar com a administração do MySQL através da *Web*, o 000webhost oferece a ferramenta phpMyAdmin (Figura 22), um *software* grátis, escrito em PHP, que é utilizado para gerenciar as tabelas e dados disponíveis. Com esta ferramenta, é possível alterar as tabelas, colunas, relacionamentos, índices, usuários e permissões dos bancos de dados através de sua interface de usuário, mantendo também a habilidade de realizar as operações através de declarações SQL (PHPMYADMIN, 2006).

Figura 22 – Aparência do phpMyAdmin



Fonte: Elaborada pelo autor.

4 Desenvolvimento

Neste capítulo, é apresentada uma visão geral do projeto, sua arquitetura, detalhes de cada uma de suas partes e como as ferramentas citadas foram utilizadas na implementação.

4.1 Design do sistema

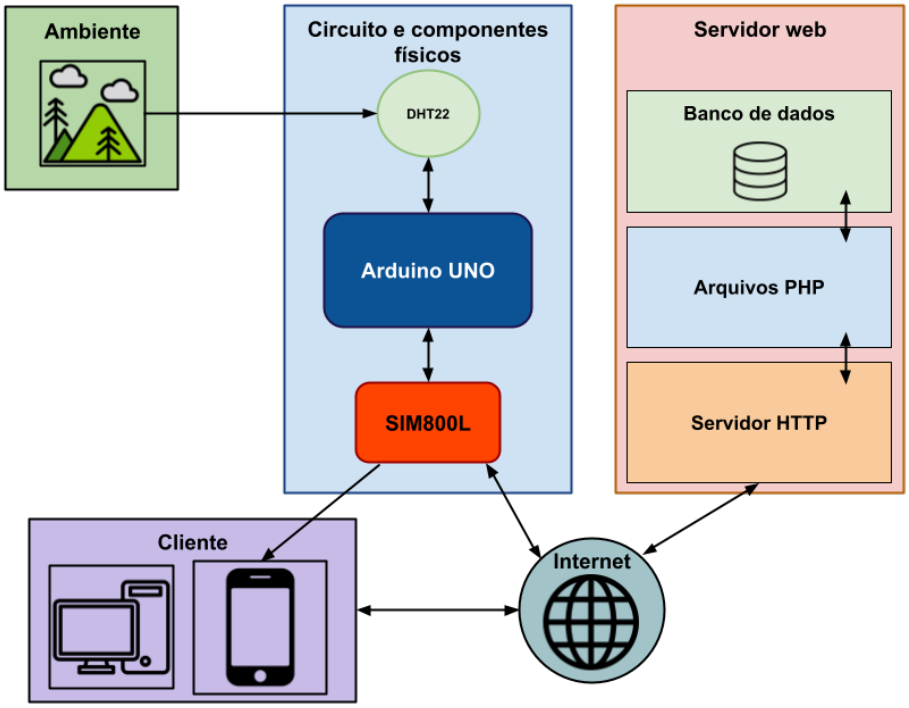
O sistema desenvolvido é separado em três aspectos principais seu esquema de arquitetura está ilustrado na Figura 23.

O primeiro aspecto se refere ao circuito eletrônico físico e seus componentes, que interagem entre si. Este sistema eletrônico funciona como a interface entre o ambiente físico, cujos parâmetros meteorológicos deseja-se ler; e o servidor da *web*, que recebe, armazena e auxilia na exibição das informações obtidas através de páginas da *web*, acessadas pelo usuário em seu aparelho. Nesta etapa, os sensores digitais do sistema eletrônico são responsáveis por realizar a leitura dos parâmetros de temperatura e umidade relativa do ar, e, a pedido da placa Arduino UNO, transmitir estas informações a ele, que serão recebidas e analisadas pelo microcomputador. O sistema eletrônico é explicado detalhadamente na seção 4.2.

Após a leitura dos parâmetros, o Arduino pode, por meio de comandos AT e o controle do módulo SIM800L, enviar mensagens de alerta SMS diretamente ao MS do usuário registrado, e/ou transmitir os dados obtidos a um servidor da *web*, através do protocolo HTTP para realizar as solicitações, e do *script* de *back-end* para armazenar as informações em um banco de dados na *web*. O processo de transmissão de dados para o servidor *web* e a estrutura do banco de dados são explicados minuciosamente na seção 4.4.

Além dos alertas imediatos de SMS, o intuito do projeto também é criar uma maneira de visualização dos dados de forma assíncrona ou em tempo real, sem precisar bombardear o aparelho móvel do usuário com mensagens instantâneas de texto. Para este propósito, foi criado um *website* que exibe os parâmetros lidos em tempo real; e também dá a opção ao usuário de buscar por histórico de leituras anteriores, com informações adicionais de data e horário. Como citado no Capítulo 3, através do serviço de *web hosting*, o *site* está disponível na *Internet*, sem restrição local; seu *back-end* foi criado utilizando *scripts* da linguagem PHP; já a parte visual e de exibição das páginas foi criado utilizando HTML e CSS. O desenvolvimento do *back-end* e do visual das páginas *web* é descrito na seção 4.5.

Figura 23 – Esquema de arquitetura do sistema de monitoramento



Fonte: Elaborada pelo autor.

4.2 Circuito eletrônico

Nesta seção, é explicada, em detalhes, a forma como os componentes são conectados no circuito eletrônico.

A conexão de pinos entre o DHT22 e o Arduino UNO é configurada da seguinte maneira:

- O pino de de alimentação de 5V do Arduino é conectado ao VCC do DHT22;
- O GND do Arduino é conectado ao GND do DHT22;
- O pino de transmissão de dados bidirecional do DHT22 é conectado ao pino digital 7 do Arduino.

Na *protoboard* (placa auxiliar de montagem), é colocado um resistor de *pull-up* entre o 5V e o pino de dados, para garantir que não haja a existência de valores aleatórios ou ruído na leitura do pino (SPARKFUN.COM, 2010).

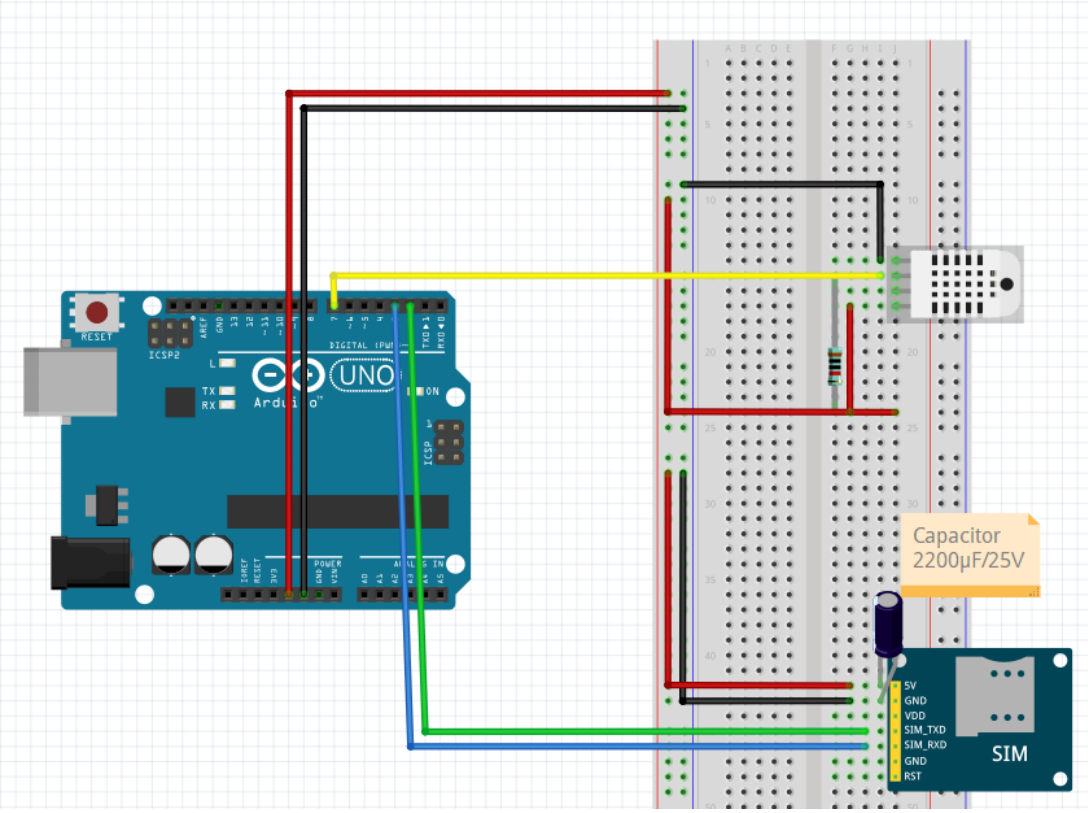
A conexão dos pinos entre o SIM800L e o Arduino UNO se dá da seguinte forma:

- Os pinos digitais 2 e 3 do Arduino são ligados, respectivamente, aos pinos SIM_TXD e SIM_RXD do SIM800L;
- O pino 5V do Arduino é ligado ao 5V do SIM800L;
- O pino GND do Arduino é ligado ao GND do SIM800L.

Entre os fios *jumper* de alimentação (5V e GND), na *protoboard*, é colocado um capacitor eletrolítico de 2200µF/25V, a fim de atingir o pico de corrente necessário às ações do módulo SIM800L, de modo que ele não reinicie.

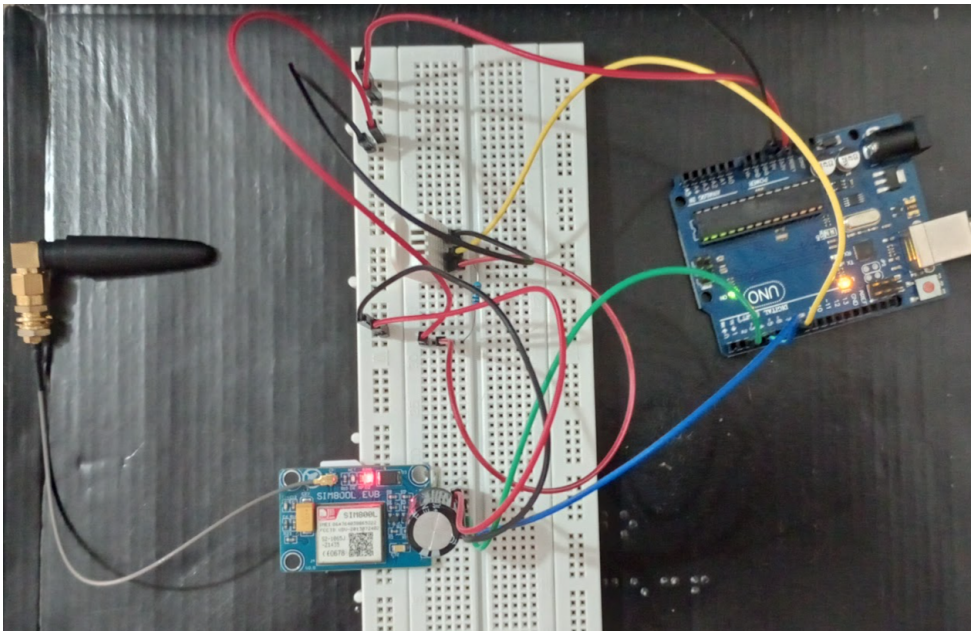
A esquemática do circuito eletrônico e a fotografia que exibe a ligação dos componentes são mostradas, respectivamente, nas Figuras 24 e 25.

Figura 24 – Esquemática do circuito eletrônico



Fonte: Elaborada pelo autor.

Figura 25 – Fotografia do circuito eletrônico



Fonte: Elaborada pelo autor.

4.3 Leitura e controle dos componentes

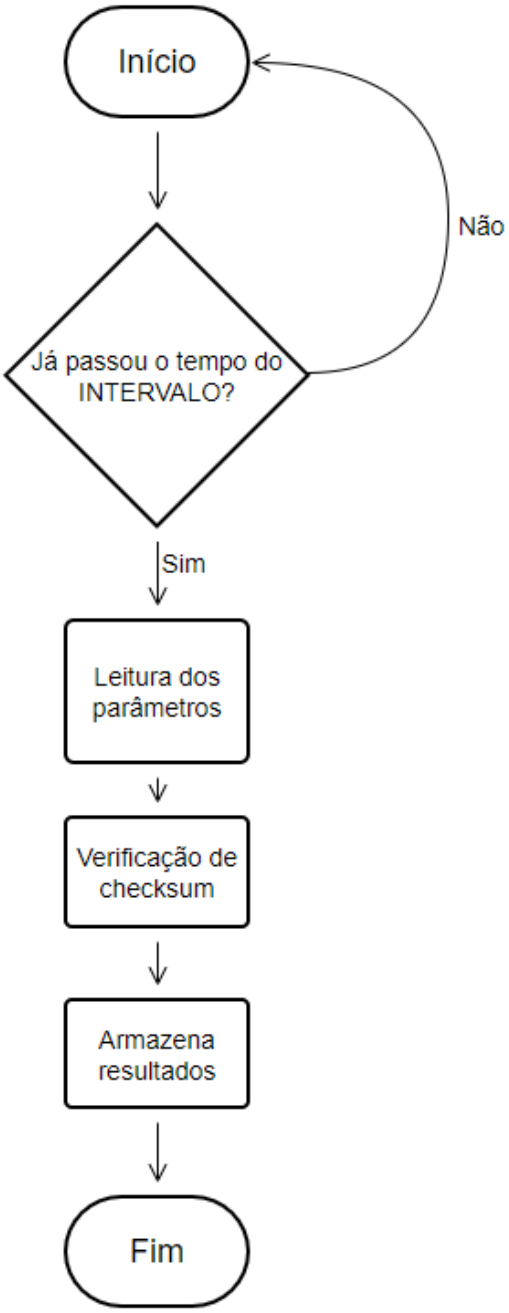
Para realizar a ação de leitura do pino 7, é utilizada a biblioteca *dht.h* (TILLAART, 2018). Através dela, é possível instanciar um objeto (*sensorDHT*) da classe *dht22*, utilizar o método de leitura (*read22()*) e armazenar os valores de estado da leitura, temperatura e umidade relativa do ar.

Ao realizar a leitura do pino, obtém-se um valor de *checksum*. Faz-se a análise deste valor para verificar se houve algum tipo de erro na leitura. Caso o estado (*Status*) da leitura seja diferente de *OK*, isso significa que há algum problema com o DHT22 ou sua conexão com o Arduino, a informação lida não é transmitida para a Internet, e o usuário é notificado através de mensagem SMS.

Caso o *Status* da ação de leitura do pino seja *OK*, os valores das variáveis *humidity* e *temperatura* do objeto *sensorDHT* são atribuídos às variáveis globais *Umidade* e *Temperatura*, respectivamente.

A fim de não haver necessidade de realizar medições constantes, a leitura é feita com uma certa frequência, especificada através da constante *INTERVALO*, que pode ser alterada baseado na necessidade do usuário. O fluxo do algoritmo da ação de leitura, escrito na função *leitura_dht22()* pode ser observado na Figura 26.

Figura 26 – Algoritmo de leitura do DHT22



Fonte: Elaborada pelo autor.

Para o alerta de SMS ser enviado a um número de celular registrado (variável *numeroTelefone*), foram definidas constantes de mínimo e máximo para temperatura e umidade do ar, que podem ser alteradas no código, de acordo com o interesse do usuário. Após o término da ação de leitura do pino de dados do DHT22, é chamada a função *checaLeitura()*, que verifica se a temperatura ou umidade estão com valores abaixo ou acima dos máximos e mínimos definidos, além de verificar se houve erro de *checksum* (Trecho do código na Figura 27). Caso alguma das condições seja verdadeira, é chamada a função *enviaSMS()*, que será responsável

por propriamente enviar a mensagem alerta de SMS ao número de telefone definido, juntamente com o tipo de alerta (temperatura, umidade, ou erro de *checksum* do módulo DHT22). Para não gerar um bombardeamento constante de mensagens, foi definida uma constante de quantidade máxima de avisos (*QUANTIDADE_AVISOS*), que representa o máximo de alertas de SMS que pode ser enviados para cada parâmetro. O trecho do código com as definições de constantes pode ser visto na Figura 28.

Figura 27 – Trecho de código de verificação de erro

```
//Verificação de erro
switch (check)
{
case DHTLIB_OK:
    Status = "OK";
    break;
case DHTLIB_ERROR_CHECKSUM:
    Status = "Erro de checksum";
    break;
case DHTLIB_ERROR_TIMEOUT:
    Status = "Erro de timeout";
    break;
case DHTLIB_ERROR_CONNECT:
    Status = "Erro de conexão";
    break;
case DHTLIB_ERROR_ACK_L:
    Status = "Erro de ACK baixo";
    break;
case DHTLIB_ERROR_ACK_H:
    Status = "Erro de ACK alto";
    break;
default:
    Status = "Erro desconhecido";
    break;
}
```

Fonte: Elaborada pelo autor.

Figura 28 – Definição de constantes

```
//Declaração de constantes para possível alerta
#define TEMP_MIN  10              //Define temperatura mínima
#define TEMP_MAX  50              //Define temperatura máxima
#define UMI_MIN 15                //Define umidade mínima
#define UMI_MAX 80                //Define umidade máxima
#define QUANTIDADE_AVISOS 5       //Define quantidade de alertas a serem enviados
```

Fonte: Elaborada pelo autor.

Após a checagem da leitura com *Status OK*, o Arduino deve comandar o módulo SIM800L através de comandos AT para realizar as transmissões de dados através da *Internet*.

Para que o Arduino transmita os comandos AT ao SIM800L, é utilizada a biblioteca *SoftwareSerial.h*, que permite a comunicação serial nos pinos digitais do arduino, utilizando

software para replicar esta funcionalidade (ARDUINO.CC, 2022). Através dela, instancia-se o objeto *sim800l*, configurando o pino 2 como aquele que receberá os dados seriais; e o pino 3 como aquele que transmitirá os dados seriais. O objeto serial *sim800l* é iniciado na *baud rate* de 9600.

A função utilizada para transmitir os comandos do Arduino para o SIM800L, através da comunicação serial, é o *enviar_serial()*, que necessita da *String* do comando de parâmetro. O objeto serial instanciado envia a *String* de comando AT ao SIM800L através do método *println*. Logo após, o Arduino ficará aguardando e lendo (por um tempo especificado) a resposta do SIM800L sendo enviada serialmente, e exibe no monitor serial. Dependendo da situação, esta resposta pode enviar um *status* bem-sucedido, alguma requisição, ou possível erro. O trecho de código da função está presente na Figura 29.

Figura 29 – Função para enviar dados ao serial do SIM800L

```
//Função para enviar dados ao serial do sim800l
void enviar_serial(String comando) {
  //Envia o comando ao serial do sim800l
  sim800l.println(comando);
  //Configura o timestamp para frequência de leitura
  long tempoespera = millis();
  //4 segundos de espera para leitura
  while (tempoespera + 4000 > millis()) {
    //Loop enquanto o sim800l estiver disponível para leitura
    while (sim800l.available()) {
      //Encaminha para o serial o que foi lido do sim800l
      Serial.write(sim800l.read());
    }
  }
  Serial.println();
}
```

Fonte: Elaborada pelo autor.

4.4 Envio e armazenamento de dados

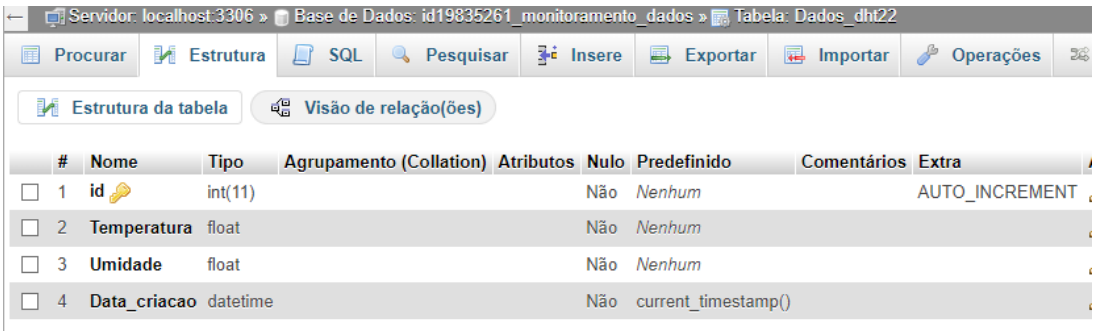
Após a obtenção da leitura dos parâmetros através da interface entre o DHT22 e o Arduino, deseja-se transmitir os dados obtidos para um *web server* e armazená-los em seu banco de dados.

O banco de dados foi criado utilizando a ferramenta phpMyAdmin, citada na subseção 3.2.3, e se trata da tabela *Dados_dht22* (Figura 30). Ela contém os seguintes campos e tipos:

- **id**, a chave primária da tabela, é do tipo *int* e é configurada para autoincrementar;
- **Temperatura**, do tipo *float*;

- **Umidade**, do tipo *float*;
- **Data_criacao**, do tipo *datetime*, predefinido para sempre ter o valor da data e horário em que a tupla foi gerada.

Figura 30 – Estrutura da tabela Dados_dht22



#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Comentários	Extra
1	id	int(11)			Não	*Nenhum*		AUTO_INCREMENT
2	Temperatura	float			Não	*Nenhum*		
3	Umidade	float			Não	*Nenhum*		
4	Data_criacao	datetime			Não	current_timestamp()		

Fonte: Elaborada pelo autor.

Para realizar a transmissão de dados, o SIM800L precisa ser configurado para o modo GPRS. Para alcançar este objetivo, o SIM800L ativa o contexto de PDP; escolhe o identificador do portador; determina o tipo de conexão para GPRS; e informa o APN, junto com informações de usuário e senha, se necessário. O código da função *configure_gprs()*, referente a este algoritmo, está presente na Figura 31.

Figura 31 – Função para configurar o SIM800L no modo GPRS

```
void configure_gprs() {
  Serial.println(" --- CONFIG GPRS --- ");
  //Configurações para o portador do token baseado em IP
  //Contype se refere ao tipo de conexão à internet, configurado para GPRS
  send_to_serial("AT+SAPBR=3,1,Contype,GPRS");
  //Configura a string de Access Point Name
  send_to_serial("AT+SAPBR=3,1,APN," + apn);
  //Configuração de usuário e senha para a APN, caso seja necessário
  if (apn_u != "") {
    send_to_serial("AT+SAPBR=3,1,USER," + apn_u);
  }
  if (apn_p != "") {
    send_to_serial("AT+SAPBR=3,1,PWD," + apn_p);
  }
}
```

Fonte: Elaborada pelo autor.

Após configurado o modo GPRS, para enviar os dados ao servidor *web*, é necessário utilizar o método POST do HTTP. Esta solicitação de POST é enviada através de um formulário HTML, e resultará em uma mudança no lado do servidor. O tipo do conteúdo (como os dados do formulário são codificados para serem enviados ao servidor) também deve ser especificado. Neste caso, o tipo de formulário utilizado é o "*application/x-www-form-urlencoded*",

em que as chaves e os valores são codificados em tuplas, separadas pelo caractere "*&*", com um símbolo "=" entre a chave e o valor (MOZILLA.ORG, 2022a).

No *back-end* do servidor, ao receber uma requisição HTTP de POST, será necessário identificar os dados sendo transmitidos através do formulário e inseri-los na tabela *Dados_dht22*. Esta ação é feita através do *script config.php*, escrito na linguagem PHP.

O *script* contém as informações de servidor, nome, usuário e senha do banco de dados; utiliza o método *$_GET* do PHP para coletar os dados submetidos através do formulário HTML (PHP.NET, 2015a); cria a conexão com o banco de dados utilizando as informações dadas; checa se há conexão e insere os dados fornecidos dentro dos campos correspondentes na tabela. Código do *config.php* disponível no Apêndice B.

A fim de solicitar o HTTP POST através do SIM800L, é preciso enviar os seguintes comandos AT (SIMCOM, 2015):

- **AT+SAPBR=1,1**, ativa o modo aberto do serviço de GPRS (*open bearer*);
- **AT+SAPBR=2,1**, ativa o modo de consulta do serviço de GPRS (*query bearer*) e deve retornar o endereço de IP do portador;
- **AT+HTTPINIT**, inicializa o serviço de HTTP do módulo;
- **AT+HTTPPARA=CID,1**, configura o parâmetro do identificador do perfil do portador como o escolhido anteriormente (1);
- **AT+HTTPPARA=URL, sendURL**, define a URL que será requisitada, em que a variável *sendURL* se trata do endereço do *script* de servidor *config.php*, somado com as combinações de chaves "Temperatura" e "Umidade" com seus valores lidos pelo DHT22, respectivamente;
- **AT+HTTPPARA=CONTENT,application/x-www-form-urlencoded**, determina o tipo de conteúdo no cabeçalho HTTP como sendo "*x-www-form-urlencoded*", citado anteriormente;
- **AT+HTTPDATA=192,5000**, determina o tamanho dos dados (em *bytes*) do conteúdo que será requisitado para POST, seguido da quantidade de tempo de espera (em ms) do comando, até *timeout*;
- **AT+HTTPACTION=1**, realiza a solicitação do método de POST HTTP e deve retornar o código de *status* HTTP. Neste caso, deve retornar 200, representando *status OK*;
- **AT+HTTPREAD**, lê a resposta retornada pelo servidor. Neste caso, deve informar os valores dos dados enviados;

- **AT+HTTPTERM**, finaliza o modo HTTP do módulo;
- **AT+SAPBR=0,1**, fecha o modo GPRS do portador identificado.

O envio dos comandos AT e suas respostas, por parte do módulo SIM800L, são apresentadas nas Figuras 32, 33 e 34.

Fluxograma da função *loop()* do Arduino é ilustrado na Figura 35, e seu código completo está disponível no Apêndice A.

Figura 32 – Envios e respostas de comandos AT para configurações de IP

```
19:18:07.452 ->  --- Start GPRS & HTTP ---
19:18:07.499 -> AT+SAPBR=1,1
19:18:07.963 -> OK
19:18:11.448 ->
19:18:11.494 -> AT+SAPBR=2,1
19:18:11.494 -> +SAPBR: 1,1,"10.224.67.254"
19:18:11.540 ->
19:18:11.540 -> OK
```

Fonte: Elaborada pelo autor.

Figura 33 – Envios e respostas de comandos AT para configurações de HTTP

```
19:18:15.488 -> AT+HTTPINIT
19:18:15.535 -> OK
19:18:19.495 ->
19:18:19.540 -> AT+HTTPPARA=CID,1
19:18:19.540 -> OK
19:18:23.538 ->
19:18:23.630 -> AT+HTTPPARA=URL,http://gsm-sensor-monitoring.000webhostapp.com/config.php?Temperatura=29.20&Umidade=61.70
19:18:23.722 -> OK
19:18:27.615 ->
19:18:27.708 -> AT+HTTPPARA=CONTENT,application/x-www-form-urlencoded
19:18:27.755 -> OK
19:18:31.697 ->
19:18:31.697 -> AT+HTTPDATA=192,5000
19:18:31.743 -> DOWNLOAD
19:18:35.695 ->
19:18:36.345 ->
19:18:36.345 -> OK
```

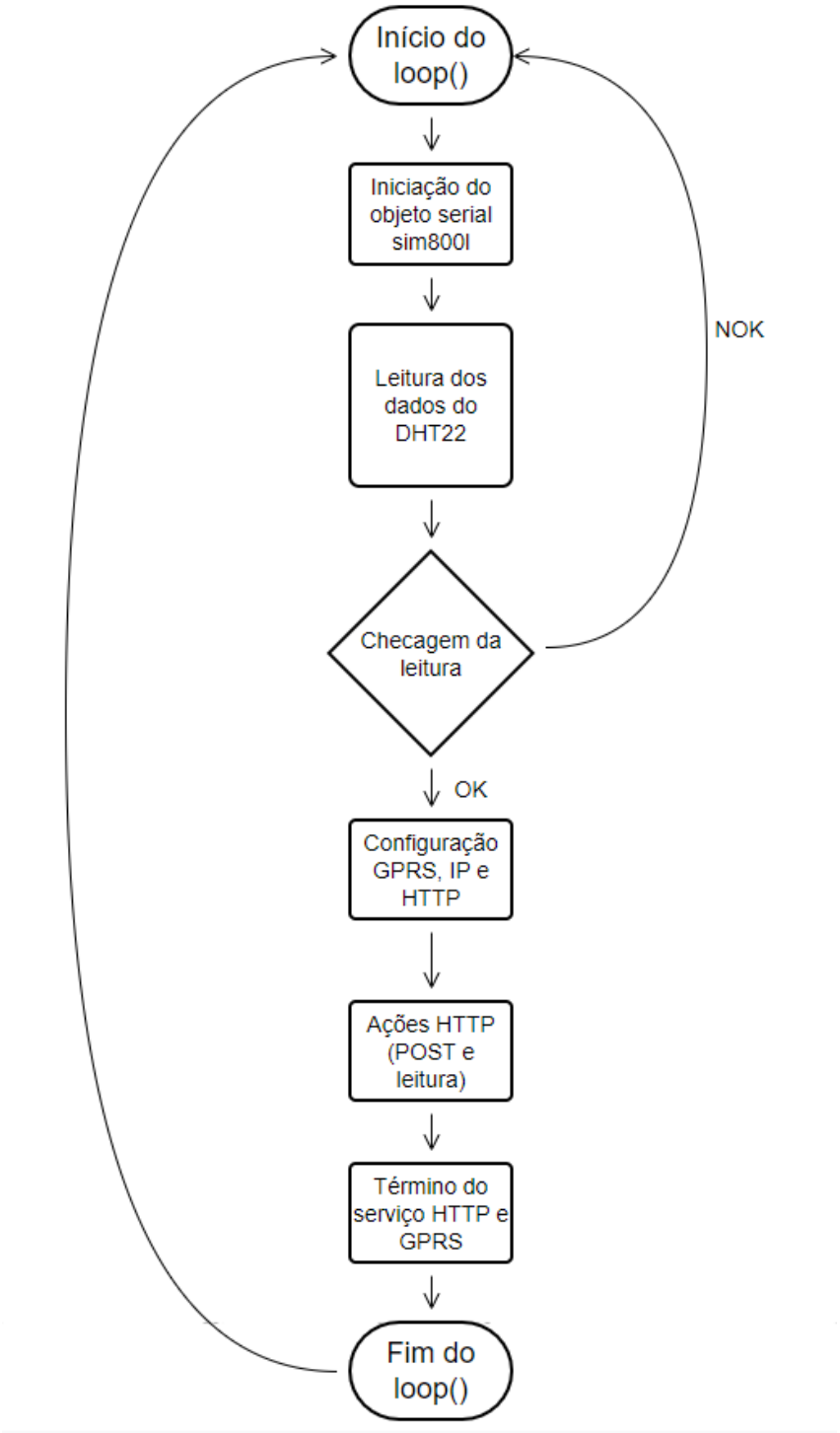
Fonte: Elaborada pelo autor.

Figura 34 – Envios e respostas de comandos AT para ações de HTTP

```
19:18:39.743 -> AT+HTTPACTION=1
19:18:39.743 -> OK
19:18:42.471 ->
19:18:42.471 -> +HTTPACTION: 1,200,20
19:18:43.730 ->
19:18:43.776 -> AT+HTTPREAD
19:18:43.776 -> +HTTPREAD: 20
19:18:43.776 -> Recebido: 29.2061.70
19:18:43.824 -> OK
19:18:47.727 ->
19:18:47.774 -> AT+HTTPTERM
19:18:47.774 -> OK
19:18:51.756 ->
19:18:51.802 -> AT+SAPBR=0,1
19:18:52.312 -> OK
```

Fonte: Elaborada pelo autor.

Figura 35 – Fluxograma do loop() do Arduino



Fonte: Elaborada pelo autor.

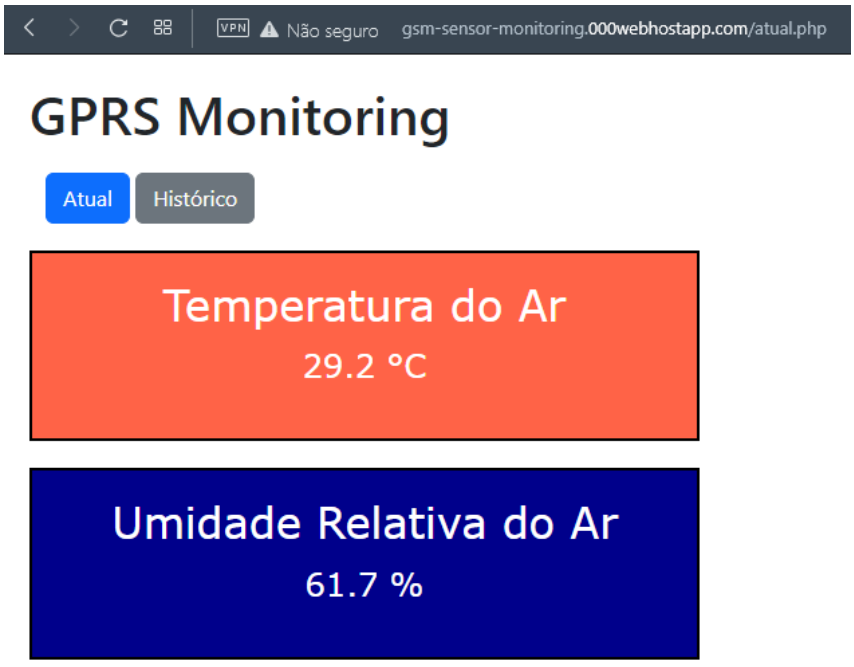
4.5 Exibição de dados

Foram criadas duas páginas *web* para exibição dos dados que são transferidos e armazenados com o intuito de que o usuário possa visualizar os dados de forma assíncrona, mas que tenha a oportunidade também de checar dados anteriores para análise própria.

Para a implementação de *containers*, botões, *inputs* de data e visuais, foi utilizada a *framework* de desenvolvimento *web Bootstrap*. Esta *framework* foi criada com o objetivo de facilitar o processo de criação de *websites* ao providenciar uma coletânea de *templates* de *designs*. Ele consiste de HTML, CSS e *scripts* em *JavaScript* para implementar diversas funções e componentes (JORDANA, 2021).

A página principal (Figura 36), *atual.php*, mostra os últimos dados lidos e enviados de temperatura e umidade relativa do ar. Ela possui dois botões, um deles referencia a própria página e o outro transfere para a página seguinte, *historico.php*. A página é construída em HTML e CSS para estruturação e visual, utilizando a *framework Bootstrap*. Para buscar os dados mais recentes dos parâmetros ambientais, a página conta com um *script* em PHP (*back-end*) para se conectar ao banco de dados, realizar a busca pelas informações desejadas e exibi-las na tela. O código completo do *script* está disponível no Apêndice C.

Figura 36 – Visual da página de dados mais recentes

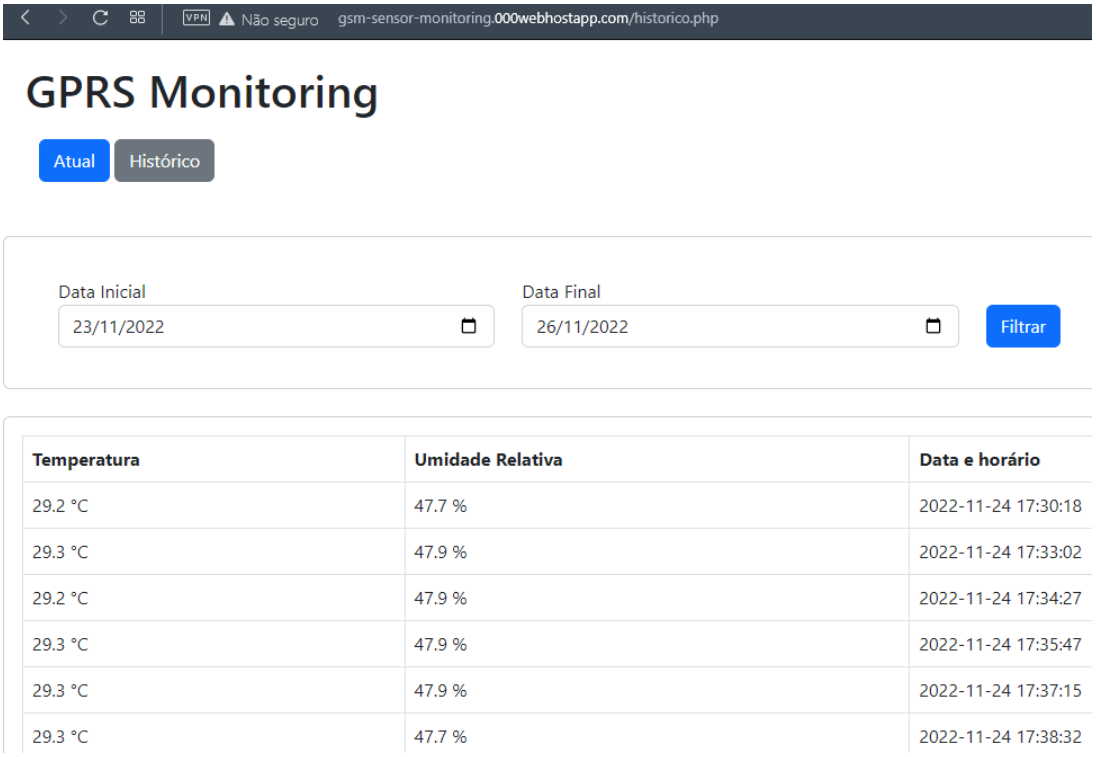


Fonte: Elaborada pelo autor.

A segunda página (Figura 37), *historico.php*, tem o propósito de exibir dados históricos, baseado no *input* do usuário. Ao receber os *inputs* de "Data Inicial" e "Data Final" através de

formulários, a página deve buscar, no banco de dados, apenas os dados obtidos na faixa de data especificada e exibi-los na tela, juntamente com a data e horário da leitura. Caso não existam dados disponíveis na faixa de tempo especificada pelo usuário, a página o deve informar. Os formulários de data e botões da página foram implementados através do *Bootstrap*; já a parte de conexão ao banco de dados, busca por faixa de data, e exibição das linhas da tabela foi feita utilizando *scripts* em PHP. O código completo do *script* está disponível no Apêndice D.

Figura 37 – Visual da página de dados históricos



Fonte: Elaborada pelo autor.

5 Conclusão

Considerando a importância do monitoramento ambiental remoto para um ecossistema, mesmo quando não há conexão de rede *Wi-Fi*, foi proposto desenvolver um sistema de IoT que realiza medições físicas de parâmetros de temperatura e umidade relativa do ar, transmita essas informações para um servidor *web* através de rede GPRS, exiba as informações lidas através de um *front-end* de páginas *web*, e alerte o usuário por SMS caso os valores lidos ultrapassem limites definidos.

Para alcançar os resultados, o desenvolvimento do sistema proposto foi dividido em três partes principais: primeiramente, foi feita uma análise do IoT e os benefícios do monitoramento remoto, estudo dos padrões de rede GSM/GPRS e análise dos componentes eletrônicos. Na segunda parte, o circuito físico de monitoramento foi implementado. O Arduino foi ligado ao sensor e módulo de rede, e códigos foram escritos para ler parâmetros, configurar transmissão de dados, transmitir pela Internet e enviar alertas de SMS. Um banco de dados *online* e um servidor *web* também foram criados para armazenar os dados enviados. A terceira parte foi o desenvolvimento do front-end com duas páginas web para consulta dos dados lidos em tempo real ou históricos com opções de filtragem por data.

Pode-se concluir que as propostas iniciais foram alcançadas, pois o sistema é capaz de medir os parâmetros físicos do ambiente e transmitir para a *web* através da tecnologia de rede GPRS, com a opção de alertar o usuário através da função SMS da rede GSM. Também é possível visualizar as leituras de forma assíncrona através das páginas hospedadas na *web*, com a opção de ver dados atuais ou históricos.

Os resultados obtidos são satisfatórios. A leitura dos dados pelo sensor é precisa, e sua análise garante que dados com erros não sejam transmitidos para o servidor. O usuário pode ser alertado rapidamente através da função de SMS, mas também pode acompanhar as leituras de forma assíncrona através das páginas *web*. Entretanto, sistema poderia ser mais robusto com a adição de mais sensores; otimização na montagem dos componentes físicos no circuito, focando na portabilidade e persistência; maior quantidade de sistemas físicos para melhor cobertura de leitura; e a opção de visualização de histórico através de gráficos, para uma análise mais completa pelo usuário.

Referências

AMIN, A.; KHAN, M. N. A. A survey of gsm technology to control remote devices. *International Journal of u- and e-Service, Science and Technology*, v. 7, p. 153–162, dez. 2014.

ANATEL. *Anatel - Telefonia Móvel*. 2022. https://informacoes.anatel.gov.br/paineis/acessos/telefonia-movel. Acesso em: 20/12/2022.

AOSONG. *AM2302 SIP Packaged Temperature and humidity sensor-sensor-temperature and humidity-Guangzhou aosong electronic co., ltd*. 2018a. http://www.aosong.com/en/products-22.html. Acesso em 26/12/2022.

AOSONG. *Temperature and humidity module*. [S.l.], 2018b.

ARDUINO.CC. *SoftwareSerial Library*. 2022. https://docs.arduino.cc/learn/built-in-libraries/software-serial. Acesso em 27/12/2022.

AVELAR, E. *ESP32 Lab 01 – Display OLED, sensor DHT22 e hora via NTP service*. 2018. https://easytromlabs.com/esp32/esp32-lab-01-display-oled-sensor-dht22-e-hora-via-ntp-service/. Acesso em 26/12/2022.

BARMPOUTIS, P.; PAPAIOANNOU, P.; DIMITROPOULOS, K.; GRAMMALIDIS, N. A review on early forest fire detection systems using optical remote sensing. *Sensors*, Basel, Switzerland, v. 20, 2020.

BODIC, G. L. *Mobile Messaging Technologies and Services: SMS, EMS and MMS*. Chichester, England: [s.n.], 2005.

CIRANI, S.; FERRARI, G.; PICONE, M.; VELTRI, L. *Internet of things: Architectures, protocols and standards*. [S.l.: s.n.], 2018.

DEJAN. *DHT11 DHT22 sensors temperature and humidity tutorial using Arduino*. 2016. https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/. Acesso em 25/12/2022.

DOMANTAS, G. *What Is Web Hosting – Web Hosting Explained for Beginners*. 2022. https://www.hostinger.com/tutorials/what-is-web-hosting/. Acesso em 26/12/2022.

DUCKETT, J. *HTML and CSS: Design and Build Websites*. [S.l.: s.n.], 2011.

EVANS, M.; NOBLE, J.; HOCHENBAUM, J. *Arduino in Action*. [S.l.]: Manning Publications, 2013.

FANG, Y.-Y.; CHEN, X.-J. Design and simulation of uart serial communication module based on vhdl. In: IEEE (Ed.). [S.l.: s.n.], 2011. (2011 3rd International Workshop on Intelligent Systems and Applications).

GEORGIEV, T.; GEORGIEVA, E.; SMIRIKAROV, A. M-learning: A new stage of e-learning. In: PRESS, A. (Ed.). New York, New York, USA: [s.n.], 2004. (Proceedings of the 5th international conference on Computer systems and technologies - CompSysTech '04).

GOURLEY, D.; TOTTY, B.; SAYER, M.; AGGARWAL, A.; REDDY, S. *HTTP: The Definitive Guide*. [S.l.: s.n.], 2002.

GROOT, W. J. de; GOLDAMMER, J. G.; KEENAN, T.; BRADY, M. A.; LYNHAM, T. J.; JUSTICE, C. O.; CSISZAR, I. A.; O'LOUGHLIN, K. Developing a global early warning system for wildland fire. *Forest ecology and management*, v. 234, p. S10, 2006.

HEINE, G. *GSM Networks: Protocols, terminology and implementation*. Norwood, MA, USA: [s.n.], 1999.

HOSTINGER. *000Webhost.com*. 2009. https://www.000webhost.com. Acesso em 26/12/2022.

JORDANA, A. *What is Bootstrap?* 2021.

KUKUSHKIN, A. *Introduction to Mobile Network Engineering: GSM, 3G-WCDMA, LTE and the Road to 5G*. Nashville, TN, USA: [s.n.], 2018.

LAN, T.; WANG, N. Future internet: The internet of things. In: . [S.l.: s.n.], 2010. (2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)).

LOHCHAB, V.; KUMAR, M.; SURYAN, G.; GAUTAM, V.; DAS, R. K. A review of iot based smart farm monitoring. *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, abr. 2018.

MCROBERTS, M. *Beginning Arduino*. [S.l.]: Apress, 2010.

MOZILLA.ORG. *POST*. 2022a. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST. Acesso em 28/12/2022.

MOZILLA.ORG. *What is a web server?* 2022b. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server. Acesso em 26/12/2022.

MOZILLA.ORG. *What is CSS?* 2022c. https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS. Acesso em 26/12/2022.

MULLA, A.; BAVISKAR, J.; KHARE, S.; KAZI, F. The wireless technologies for smart grid communication: A review. *2015 Fifth International Conference on Communication Systems and Network Technologies*, abr. 2015.

MYSQL.COM. *What is MySQL?* 2012. https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. Acesso em 26/12/2022.

NETTIGO. *SIM800L GSM / GRPS module*. 2016. https://nettigo.eu/products/sim800l-gsm-grps-module. Acesso em: 25/12/2022.

PHPMYADMIN. *phpMyAdmin*. 2006. https://www.phpmyadmin.net. Acesso em 26/12/2022.

PHP.NET. *SQL Injection*. 2006. https://www.php.net/manual/en/security.database.sql-injection.php. Acesso em 26/12/2022.

PHP.NET. *$_GET*. 2015a. https://www.php.net/manual/en/reserved.variables.get.php. Acesso em 28/12/2022.

PHP.NET. *What is PHP?* 2015b. https://www.php.net/manual/en/intro-whatis.php. Acesso em 26/12/2022.

SIMCOM. *SIM800L Hardware Design V1.00*. [S.l.], 2013.

SIMCOM. *SIM800 Series AT Command Manual V1.09*. [S.l.], 2015.

SPARKFUN.COM. *Pull-up resistors*. 2010. https://learn.sparkfun.com/tutorials/pull-up-resistors/all. Acesso em 27/12/2022.

TILLAART, R. *Arduino playground - DHTLib*. 2018. https://playground.arduino.cc/Main/DHTLib/. Acesso em 27/12/2022.

YAMAZOE, N.; SHIMIZU, Y. Humidity sensors: Principles and applications. *Sensors and actuators*, out. 1986.

Apêndices

APÊNDICE A – Código do Arduino

```
/*********************************************************************
 *
 * Código para leitura dos parâmetros do DHT22 e enviar os dados para um
    web server
 *
 * Autor: Samuel Cabral
 * RA: 181026171
 *
*********************************************************************/


//Biblioteca para permitir comunicação serial nos pinos digitais do
   Arduino
#include <SoftwareSerial.h>
//Biblioteca para realizar a leitura do sensor
#include <dht.h>

//Definindo pino de dados do sensor
#define PIN_SENSOR 7
//Intervalo para frequencia de leitura (em ms)
#define INTERVALO 2000

//Variável para auxiliar na frequência de leitura do sensor
unsigned long delayIntervalo;
//Instancia objeto da classe DHT
dht sensorDHT;

//Variáveis float para armazenar umidade e temperatura do ar
float Umidade, Temperatura;
//Variável para armazenar o status da leitura
String Status;
//Variável para armazenar contagem de erro
int ContagemErro;
//Variável para armazenar contagem de alertas de temperatura
int ContagemAlertaTemp;
//Variável para armazenar contagem de alertas de temperatura
int ContagemAlertaUmi;

//Declaração de constantes para possível alerta
#define TEMP_MIN  10                //Define temperatura mínima
#define TEMP_MAX  50                //Define temperatura máxima
#define UMI_MIN 15                  //Define umidade mínima
#define UMI_MAX 80                  //Define umidade máxima
```

```
#define QUANTIDADE_AVISOS 5         //Define quantidade de alertas a
    serem enviados

//Configura o APN (Access Point Name)
String apn = "internet";
//APN Username (Caso necessário)
String apn_u = "";
//APN Password (Caso necessário)
String apn_p = "";
//URL do servidor
String url = "http://gsm-sensor-monitoring.000webhostapp.com/config.php
    ";
//Define o número de celular que irá receber as mensagens SMS
String numeroTelefone = "+XXXXXXXXXXXXX";

SoftwareSerial sim800l(2,3);        //Instancia objeto serial sim800l,
    com TX no pino 2 e RX no pino 3

//Função de configuração
void setup()
{
  //Configura a taxa de dados em bits por segundo (9600) para transmissã
   o de dados serial
  Serial.begin(9600);
  //Configura a taxa de dados em bits por segundo (9600) para transmissã
   o de dados serial do sim800l
  sim800l.begin(9600);
  //Aguarda 5 segundos
  delay(5000);
  while (sim800l.available()) {
    //Encaminha para o serial o que foi lido do sim800l
    Serial.write(sim800l.read());
  }
  //Aguarda dois segundos
  delay(2000);
  //Variável de contagem de erro e alerta inicia com zero
  ContagemErro = ContagemAlertaTemp = ContagemAlertaUmi = 0;
}

//Função que fica em execução constantemente
void loop() {
  //Faz a leitura dos dados do DHT22
  leitura_dht22();
  //Analisa a leitura do DHT22
  checaLeitura();
  if (Status = "OK")  {
    //Faz a configuração do modo GPRS do sim800l
```

```
      configure_gprs();
      //Envia os dados lidos ao servidor
85    send_to_server();
      }
    //Aguarda 2 segundos
    delay(2000);
89 }

  //Função para enviar dados lidos ao web servidor
  void send_to_server() {
93  //Define a string sendURL
    String sendURL = url+"?Temperatura="+String(Temperatura)+"&Umidade="+
     String(Umidade);
    Serial.println(" --- Start GPRS & HTTP --- ");
    //Ativa o GPRS
97  enviar_serial("AT+SAPBR=1,1");
    //Consulta a conexão, deve retornar um endereço IP
    enviar_serial("AT+SAPBR=2,1");
    //Ativa o modo HTTP
101 enviar_serial("AT+HTTPINIT");
    //Configura o identificador de perfil HTTP
    enviar_serial("AT+HTTPPARA=CID,1");
    //Define a URL que será acessada
105 enviar_serial("AT+HTTPPARA=URL," + sendURL);
    //Configura o conteúdo para parâmetro URL
    enviar_serial("AT+HTTPPARA=CONTENT,application/x-www-form-urlencoded")
     ;
    //Tamanho dos dados em byte para POST, e tempo em ms para o input
109 enviar_serial("AT+HTTPDATA=192,5000");
    //Envia para o serial os parâmetros lidos do DHT22
    enviar_serial(String(Temperatura)+String(Umidade));
    //Realiza a ação de POST dos dados
113 enviar_serial("AT+HTTPACTION=1");
    //Lê a resposta do servidor
    enviar_serial("AT+HTTPREAD");
    //Termina o serviço HTTP
117 enviar_serial("AT+HTTPTERM");
    //Finaliza o modo GPRS
    enviar_serial("AT+SAPBR=0,1");
  }
121
  //Função para configurar o sim800l no modo GPRS
  void configure_gprs() {
    Serial.println(" --- CONFIG GPRS --- ");
125 //Configurações para o portador do token baseado em IP
    //Contype se refere ao tipo de conexão à internet, configurado para
     GPRS
```

```
    enviar_serial("AT+SAPBR=3,1,Contype,GPRS");
    //Configura a string de Access Point Name
    enviar_serial("AT+SAPBR=3,1,APN," + apn);
    //Configuração de usuário e senha para a APN, caso seja necessário
    if (apn_u != "") {
      enviar_serial("AT+SAPBR=3,1,USER," + apn_u);
    }
    if (apn_p != "") {
      enviar_serial("AT+SAPBR=3,1,PWD," + apn_p);
    }
}

  //Função para enviar dados ao serial do sim800l
  void enviar_serial(String comando) {
    //Envia o comando ao serial do sim800l
    sim800l.println(comando);
    //Configura o timestamp para frequência de leitura
    long tempoespera = millis();
    //4 segundos de espera para leitura
    while (tempoespera + 4000 > millis()) {
      //Loop enquanto o sim800l estiver disponível para leitura
      while (sim800l.available()) {
        //Encaminha para o serial o que foi lido do sim800l
        Serial.write(sim800l.read());
      }
    }
    Serial.println();
  }

  void leitura_dht22(){
    if ((millis() - delayIntervalo) > INTERVALO) { //Se passou do tempo de
      intervalo

        //Leitura dos dados
        int check = sensorDHT.read22(PIN_SENSOR); //Faz a leitura do DHT22
      no pino 7

        //Verificação de erro
        switch (check)
        {
      case DHTLIB_OK:
            Status = "OK";
            break;
      case DHTLIB_ERROR_CHECKSUM:
            Status = "Erro de checksum";
            break;
      case DHTLIB_ERROR_TIMEOUT:
```

```
            Status = "Erro de timeout";
            break;
        case DHTLIB_ERROR_CONNECT:
            Status = "Erro de conexão";
            break;
        case DHTLIB_ERROR_ACK_L:
            Status = "Erro de ACK baixo";
            break;
        case DHTLIB_ERROR_ACK_H:
            Status = "Erro de ACK alto";
            break;
        default:
            Status = "Erro desconhecido";
            break;
        }

        //Dados lidos
        //Umidade relativa do ar
        Umidade = sensorDHT.humidity;
        //Temperatura do ar
        Temperatura = sensorDHT.temperature;
        //Reinicia a variável delayIntervalo considerando o tempo passado
        delayIntervalo = millis();
      }
  }

  void checaLeitura(){
    //Se a temperatura está abaixo ou acima da máxima e mínima definidas
    if (Temperatura <= TEMP_MIN || Temperatura >= TEMP_MAX){
      ContagemAlertaTemp++;
      if (ContagemAlertaTemp <= QUANTIDADE_AVISOS) enviaSMS("Temperatura",
      Temperatura);
      }
    else{
      ContagemAlertaTemp = 0;
    }
    //Se a umidade está abaixo ou acima da máxima e mínima definidas
    if (Umidade <= UMI_MIN || Umidade >= UMI_MAX){
      ContagemAlertaUmi++;
      if (ContagemAlertaUmi <= QUANTIDADE_AVISOS) enviaSMS("Umidade",
     Umidade);
      }
    else{
      ContagemAlertaUmi = 0;
    }
    //Se o Status da leitura for diferente do normal
    if (Status != "OK"){
```

```
217     ContagemErro++;
        if (ContagemErro <= QUANTIDADE_AVISOS) enviaSMS("Error", 0);
        }
      else{
221     ContagemErro = 0;
      }
    }

225 void enviaSMS(String Parametro, float Valor){
      //Configura o módulo para o modo SMS
      enviar_serial("AT+CMGF=1");
      //Enviar SMS para o número especificado
229   enviar_serial("AT+CMGS=\"" + numeroTelefone + "\"");

      if (Parametro != "Error")
        //Mensagem de alerta de parâmetro
233     enviar_serial("Foi detectado uma " + Parametro + " de " + Valor);
      else
        //Mensagem de alerta de erro do módulo DHT22
        enviar_serial("Foi detectado o seguinte erro na leitura:" + Status);
237   //Símbolo que representa o fim da mensagem de SMS
      enviar_serial((String)(char)26);
    }
```

APÊNDICE B – Código do *script* config.php

O código se refere ao *script* em PHP para obtenção dos dados enviados pelo método HTTP POST e seu armazenamento no banco de dados do servidor.

```
<?php
$servername = "localhost";
$dbname = "id19835261_monitoramento_dados";
$username = "id19835261_monitoring";
$password = "XXXXXXXXXXXXXXXX";

date_default_timezone_set('America/Sao_Paulo');

//Valores que serão enviados
$Temperatura = $_GET['Temperatura'];
$Umidade = $_GET['Umidade'];

echo 'Recebido: '.$Temperatura.$Umidade;

//Cria a conexão
$conn = new mysqli($servername, $username, $password, $dbname);

//Checar conexão
if ($conn->connect_error) {
  die("Falha de conexão: " . $conn->connect_error);
}

//Inserção de dados
$sql = "INSERT INTO Dados_dht22(Temperatura,Umidade) VALUES ('
   $Temperatura','$Umidade')";

//Verificação de conexão
if ($conn->query($sql) === TRUE) {
  return "Dados inseridos com sucesso!";
}
else {
  return "Erro: " . $sql . "<br>" . $conn->error;
}
$conn->close();

?>
```

APÊNDICE C – Código do *script* atual.php

O código se refere ao *script* da página da web que exibe os parâmetros lidos mais recentes.

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>GPRS Monitoring</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/
    bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/
    bootstrap.bundle.min.js"></script>
    <style>
      .temperatura {
        background-color: tomato;
        color: white;
        border: 2px solid black;
        margin: 20px;
        padding: 20px;
        font-family: verdana;
        max-width: 500px;
        text-align: center;
      }
      .umidade {
        background-color: darkblue;
        color: white;
        border: 2px solid black;
        margin: 20px;
        padding: 20px;
        font-family: verdana;
        max-width: 500px;
        text-align: center;
      }
    </style>
  </head>

  <!-- Script php para pegar os dados mais recentes da database -->
  <?php
    $servername = "localhost";
    $dbname = "id19835261_monitoramento_dados";
    $username = "id19835261_monitoring";
    $password = "XXXXXXXXXXXXXXXX";
```

```
      $conn = mysqli_connect($servername, $username, $password, $dbname);
41    $sql = "SELECT * FROM Dados_dht22 ORDER BY id DESC LIMIT 1";
      $query = $conn->query($sql);

      if($query->num_rows >0){
45      while($row = $query-> fetch_assoc()){
          $temperatura = $row['Temperatura'];
          $umidade = $row['Umidade'];
        }
49    }
      else{
        $temperatura = 'Nada';
        $umidade = 'Nada';
53    }
      $conn->close();
    ?>

57  <body>
      <h1 style="margin:20px"> GPRS Monitoring </h1>
      <div class="container mt-3" style="margin: 20px">
        <button type="button" class="btn btn-primary" onclick="location.
     href='http://gsm-sensor-monitoring.000webhostapp.com/atual.php'">
61        Atual
        </button>
        <button type="button" class="btn btn-secondary" onclick="location.
     href='http://gsm-sensor-monitoring.000webhostapp.com/historico.php'">
          Histórico
65      </button>
      </div>

      <div class="temperatura">
69      <h2>Temperatura do Ar</h2>
        <p style="font-size: 24px"><?php echo $temperatura; ?>  C </p>
      </div>

73    <div class="umidade">
        <h2>Umidade Relativa do Ar</h2>
        <p style="font-size: 24px"><?php echo $umidade; ?> %</p>
      </div>
77
    </body>
  </html>
```

APÊNDICE D – Código do *script* historico.php

O código se trata da exibição da página que apresenta os formulários para entrada de uma faixa de data, e exibição dos parâmetros lidos no período de tempo especificado pelo usuário.

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>GPRS Monitoring</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/
    bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/
    bootstrap.bundle.min.js"></script>
    <script src="jquery-3.6.1.min.js"></script>
  </head>

  <body>
    <h1 style="margin:20px"> GPRS Monitoring </h1>
    <div class="container mt-3" style="margin: 20px">
      <button type="button" class="btn btn-primary" onclick="location.
    href='http://gsm-sensor-monitoring.000webhostapp.com/atual.php'">
        Atual
      </button>
      <button type="button" class="btn btn-secondary" onclick="location.
    href='http://gsm-sensor-monitoring.000webhostapp.com/historico.php'">
        Histórico
      </button>
    </div>

    <div class="card mt-5">
      <div class="card-body">
        <form action="" method="GET">
          <div class="row" style="margin: 20px">
            <div class="col-md-4">
              <div class="form-group">
                <label>
                  Data Inicial
                </label>
                <input type="date" name="data_inicial" value="<?php if(
```

```
      isset($_GET['data_inicial'])){ echo $_GET['data_inicial']; } ?>"class
      ="form-control">
33              </div>
              </div>

              <div class="col-md-4">
37              <div class="form-group">
                  <label>
                    Data Final
                  </label>
41                <input type="date" name="data_final" value="<?php if(
      isset($_GET['data_final'])){ echo $_GET['data_final']; } ?>" class="
      form-control">
                </div>
              </div>

45            <div class="col-md-4" >
                <div class="form-group">
                  <br>
                  <button type="submit" class="btn btn-primary">
49                  Filtrar
                  </button>
                </div>
              </div>
53          </div>
          </form>
        </div>
       </div>
57
       <div class="card mt-4">
         <div class="card-body">
           <table class="table table-bordered">
61           <thead>
               <tr>
                 <th>Temperatura</th>
                 <th>Umidade Relativa</th>
65               <th>Data e horário</th>
               </tr>
             </thead>
             <tbody>
69
             <?php
               $servername = "localhost";
               $dbname = "id19835261_monitoramento_dados";
73             $username = "id19835261_monitoring";
               $password = "XXXXXXXXXXXXXXXX";
```

```
                $conn = mysqli_connect($servername, $username, $password,
      $dbname);
77
                if(isset($_GET['data_inicial']) && isset($_GET['data_final
      ']))){
                  $data_inicial = $_GET['data_inicial'];
                  $data_final = $_GET['data_final'];
81                $sql = "SELECT * FROM Dados_dht22 WHERE Data_criacao
      BETWEEN '$data_inicial' AND '$data_final'";
                  $query = $conn->query($sql);

                  if($query->num_rows >0){
85                  while($row = $query-> fetch_assoc()){
                      ?>
                        <tr>
                          <td><?php echo $row['Temperatura']."  C "; ?></td>
89                        <td><?php echo $row['Umidade']." %"; ?></td>
                          <td><?php echo $row['Data_criacao']; ?></td>
                        </tr>
                      <?php
93                  }
                  }
                  else{
                    echo "Sem dados encontrados neste período.";
97                }
                }
              ?>
              </tbody>
101         </table>
          </div>
        </div>

105   </body>
    </html>
```