

## IMPLEMENTAÇÃO DE CRIPTOGRAFIA EM REDE DE COMUNICAÇÃO IOT

### IMPLEMENTATION ENCRYPTION IN IOT COMMUNICATION NETWORK

André Giovanni Castaldin, Fatec Ourinhos, [andre.castaldin@fatecourinhos.edu.br](mailto:andre.castaldin@fatecourinhos.edu.br)  
Matheus Felipe Ribeiro Fabricio, Fatec Ourinhos, [matheus.fabricio@fatec.sp.gov.br](mailto:matheus.fabricio@fatec.sp.gov.br)  
Maykon Douglas Alves, Fatec Ourinhos, [maykon.alves@fatec.sp.gov.br](mailto:maykon.alves@fatec.sp.gov.br)  
Rafael Marques Mansano, Fatec Ourinhos, [rafael.mansano@fatec.sp.gov.br](mailto:rafael.mansano@fatec.sp.gov.br)

#### Resumo

A Internet das Coisas (IoT) é uma tecnologia que vem crescendo mundo à fora e sendo cada vez mais utilizada em vários cenários, como casas inteligentes, fábricas e vários outros setores. No entanto, devido à rápida implementação dessa tecnologia e aos diversos ambientes que utilizam dessa aplicação, muitos desses dispositivos IoT acabaram sendo implementados com uma baixa segurança, com poucos recursos, sem autenticações e uso de protocolos vulneráveis. O objetivo desse artigo é apresentar uma solução com criptografia para ser possível evitar a disseminação de falsos dados em uma rede IoT e mitigar a exposição de dados abertos que trafegam por meio de query strings em um sistema de controle de umidade e temperatura. Para esse objetivo, foi proposto a implementação de criptografia para que seja possível ocultar os dados que serão capturados pelo DHT11 e estarão em trânsito quando disparados do ESP32 para o banco de dados e site Web. Esse trabalho apresenta uma solução para melhorar a segurança em um dispositivo IoT, garantindo a proteção dos dados de temperatura e umidade coletados, que manterá a integridade e confidencialidade das informações dessa rede.

**Palavras-chave:** Criptografia, IOT, ESP32 e Arduino.

#### Abstract

*The Internet of Things (IoT) is a technology that has been growing worldwide and is increasingly being utilized in various scenarios, such as smart homes, factories, and various other sectors. However, due to the rapid implementation of this technology and the diverse environments that use this application, many of these IoT devices ended up being implemented with low security, limited resources, without authentication, and the use of vulnerable protocols. The objective of this article is to present a solution with encryption to prevent the spread of false data in an IoT network and mitigate the exposure of open data that travels through query strings in a humidity and temperature control system. For this purpose, the implementation of encryption was proposed to hide the data that will be captured by the DHT11 and will be in transit when sent from the ESP32 to the database and website. This work presents a solution to enhance security in an IoT device, ensuring the protection of collected temperature and humidity data, which will maintain the integrity and confidentiality of information in this network.*

**Keywords:** Encryption, IOT, ESP32, Arduino.

## 1 – Introdução

Com o rápido desenvolvimento recente da tecnologia de hardware e de rede, a tecnologia da *Internet of Things* (IoT), traduzido como Internet das Coisas está sendo aplicada em todas as partes de nossas vidas. Essa tecnologia é utilizada em áreas como casas, fábricas e cidades inteligentes, além de ser usada em setores como transporte, onde sensores são instalados em veículos e estradas e conectados através de redes. De acordo com Vailshery (2022), o número de dispositivos IoT presentes no mundo será superior a 29 bilhões até 2030. No entanto, devido a rápida implementação e diversidade das necessidades do mercado, muitos dispositivos IoT foram implantados com recursos de segurança fracos ou inexistentes, tornando-os vulneráveis a ataques maliciosos. Cada dispositivo inteligente precisa de proteção e manutenção na medida em que essas vulnerabilidades são detectadas.

Como exemplos de ameaças de segurança em dispositivos IoT, temos a falta de autenticação, o que pode permitir que terceiros não autorizados acessem facilmente esses serviços e dispositivos. Algumas das ameaças incluem serviços como *SHODAN* (mecanismo de busca online de dispositivos conectados à Internet criado em 2009 por John Matherly), onde é possível monitorar aparelhos como, por exemplo, câmeras de segurança, roteadores e sensores de monitoramento e controle de temperatura. Também o acesso interno por serviço vulnerável como, por exemplo, o protocolo de acesso remoto *Telnet*, onde terceiros não autorizados podem adentrar a dispositivos IoT devido à abertura de portas de serviço desnecessárias ou vulneráveis, além das portas de acesso para fornecer serviços de IoT. Como o Telnet não oferece suporte à comunicação criptografada, é possível que um invasor obtenha facilmente comandos e seus resultados por meio de *sniffing*, técnica utilizada para capturar dados sendo transmitidos em tempo real dentro de uma rede, como, por exemplo, credenciais de usuários.

Dados os exemplos acima, o objetivo deste artigo é apresentar solução eficaz que permita a transmissão de dados, como temperatura e umidade, de forma criptografada para posterior armazenamento, evitando a exposição dos dados que trafegam por meio de *query string* (parâmetros e informações, algumas vezes sensíveis, que são passados pela URL do navegador para o servidor da aplicação requisitada). Sem a proteção necessária, um invasor pode interceptar e visualizar os dados durante sua transição na rede.

## 2 - Referencial Teórico

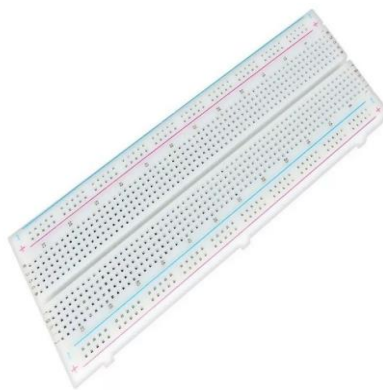
### 2.1 Tecnologias

Para a elaboração deste trabalho, foram utilizadas as seguintes ferramentas e tecnologias para a coleta, armazenamento e análise de dados de temperatura e umidade em um ambiente específico, com o objetivo de garantir a segurança dos dados durante todo o processo de transmissão. Essas ferramentas foram escolhidas por serem acessíveis e eficientes para a realização do projeto, permitindo a coleta, armazenamento e exibição de dados precisos e confiáveis.

#### 2.1.1 Protoboard 830 Pontos

A figura 1 representa a *protoboard*, também conhecida como *Breadboard*, Placa de Ensaio ou Matriz de Contato, é uma placa com furos e conexões pré-definidas, que visa auxiliar a montagem de teste de circuitos eletrônicos experimentais de forma simples e ágil. (MAKERHERO,2023). Foi utilizada neste trabalho para construir circuitos de teste e comunicação entre o ESP32 e o sensor DHT11.

Figura 1 - Protoboard 830 pontos



Fonte: MAKERHERO (2023)

#### 2.1.2 Sensor DHT11

O item representado na figura 2 é o sensor DHT11 que é capaz de fazer a medição de temperatura entre 0° e 50° Celsius e umidade entre 20% e 90% com faixa de precisão de 2° Celsius para temperatura e 5% para umidade. (ARDUINO&CIA,2013). Sua função neste trabalho é de medição das temperaturas ambientes seu envio para o ESP32.

**Figura 2 - Sensor DHT11**

Fonte: ARDUINO&amp;CIA (2013)

### 2.1.3 Microcontrolador ESP32

O ESP32, mostrado na figura 3 é um microcontrolador de baixo custo e consumo de energia além de possuir Bluetooth e tecnologia *Wireless* (ESPRESSIFSYSTEMS,2023). Foi utilizado neste trabalho para coletar informações recebidas pelo sensor DHT11, analisá-las e enviá-las para o banco de dados.

**Figura 3 - Microcontrolador ESP32**

Fonte: VIDADESILICIO (2023)

### 2.1.4 Sistema de Gerenciamento de dados MySQL

A figura 4 refere-se ao logo do sistema de gerenciamento de dados MySQL. O MySQL é um sistema de gerenciamento de banco de dados relacional de código aberto amplamente utilizado, conhecido por sua confiabilidade e desempenho. Ele organiza dados

em tabelas usando SQL, é altamente escalável e possui uma comunidade ativa de desenvolvedores. Amplamente empregado em aplicativos da web e sistemas empresariais, o MySQL oferece uma gama de recursos avançados e é compatível com várias linguagens de programação, tornando-se uma escolha popular para armazenar e gerenciar dados. Foi utilizado em nosso trabalho para fazer o armazenamento de informações coletadas e enviadas pelo ESP32.

**Figura 4 – logo MySQL**



Fonte: MYSQL (2023)

### **2.1.5 Protocolos HTTP e HTTPS**

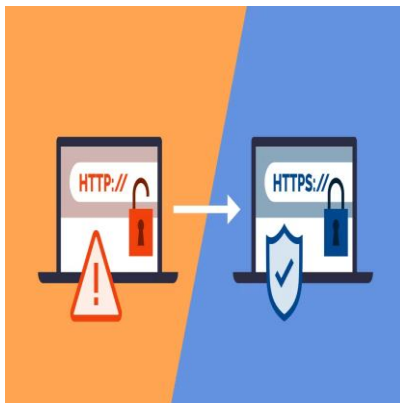
Na figura 6, é mostrado uma imagem apresentando a principal diferença entre o protocolo HTTP e HTTPS, a segurança. O protocolo HTTP, sigla de *Hyper Text Transfer Protocol*, traduzido para Protocolo de Transferência de Hipertexto é um protocolo que possibilita a transferência de dados pela internet, principalmente por páginas na internet e dados entre um servidor web e um navegador. Funciona sem criptografia, ou seja, toda informação enviada pode ser acessada e alterada por terceiros que tenham acesso. (Hostgator, 2021)

A sua versão segura, isto é, com criptografia é o protocolo HTTPS, de *Hyper Text Transfer Protocol Secure*, traduzido como Protocolo de Transferência de Hipertexto Seguro. Por possuir proteção criptográfica, as informações transmitidas desde a origem são protegidas de acesso e alteração por terceiros. (Hostgator, 2021).

Utiliza os protocolos de segurança SSL (*Secure Sockets Layer*) e TLS (*Transport Layer Security*), que tem como função estabelecer conexões seguras na Internet, garantindo a privacidade, autenticidade e integridade dos dados transmitidos entre um cliente e um

servidor. É amplamente usado para navegação segura na *Web* e em várias outras aplicações, esses protocolos desempenham um papel fundamental na segurança *online*. (Hostgator, 2021)

**Figura 6 - HTTP e HTTPS**



Fonte: HOSTGATOR (2021)

### 2.1.6 PHP

O logo mostrado na figura 7 é referente a linguagem de programação para desenvolvimento web PHP, acrônimo de *Hypertext Preprocessor*, em português brasileiro Pré-processador de Hipertexto é utilizada para a criação e desenvolvimento dos códigos necessários para transferência de dados para o banco de dados. (PHP, 2023)

**Figura 7 - Logo PHP**



Fonte: PHP (2023)

### 2.1.8 Arduino IDE

A figura 8 é referente a IDE (*Integrated Development Environment*), traduzido como Ambiente de Desenvolvimento Integrado da Arduino é um editor para desenvolvimento de

códigos com o objetivo de conectar e fazer a comunicação com o *hardware* de dispositivos, como ESP32. (Arduino, 2023)

**Figura 8 – Logo Arduino**



Fonte: ARDUINO (2023)

### 2.1.9 – XAMPP

O XAMPP, representado pela figura 9, é um pacote com os principais servidores de código aberto do mercado, incluindo FTP, banco de dados MySQL e Apache, além das linguagens PHP e Perl. É extremamente prático para criar um servidor web local para testes. É usado principalmente por desenvolvedores *Web* para criar e testar sites e aplicativos localmente antes de implantá-los em um ambiente de produção. (APACHEFRIENDS, 2023). Foi utilizado neste projeto pois uma de suas ferramentas, o *phpMyAdmin*, é de extrema utilidade para gerenciar o banco de dados *MySQL* utilizado para armazenar os dados captados pelo ESP32.

**Figura 9 – Logo XAMPP**



Fonte: APACHEFRIENDS (2023)

### 2.1.10 – WIRESHARK

O *Wireshark*, representado pela figura 10, é um analisador de pacotes de rede. É uma ferramenta capaz de capturar, analisar e visualizar o tráfego de rede em tempo real. É usado por profissionais de rede e segurança para solucionar problemas, monitorar a atividade da rede e detectar problemas de segurança. (WIRESHARK, 2023). Foi utilizado para capturar a transferência de pacotes entre o ESP32 e o banco de dados, sendo possível, no caso de aplicações sem criptografia, visualizar informações de temperatura e umidade.

Figura 10 – Logo WIRESHARK



Fonte: WIRESHARK (2023)

## 2.2 Revisão da Literatura

Vários artigos nos apresentam a necessidade de implementar medidas de monitoramento e segurança em rede com dispositivos inteligentes. No trabalho de Paula (2020) o Módulo de Alertas do *Smart Place* tem como o objetivo monitorar dados para detectar falhas em um sistema *Smart Place* (sistema de IoT que tem por finalidade diminuir o consumo de energia elétrica por meio do gerenciamento eficiente dos aparelhos) e notificar os desenvolvedores e usuários de forma automática detalhando o problema detectado, sendo seu diferencial a possibilidade do usuário através de uma interface gráfica reportar um problema que está ocorrendo e acompanhar o feedback dos administradores desde a geração do alerta até sua resolução. O Módulo de Alertas foi arquitetado em três elementos principais: Dispositivos de *hardware*, infraestrutura de *middleware* e plataforma *Web*.



Para Silva (2020) as redes IoT vem ganhando força em todo o mundo por seus benefícios e facilidades, porém as vulnerabilidades que ele possui não são bem abordadas. Os dispositivos IoT quando interligados em uma rede são muito vulneráveis, pois apresentam uma gama de problemas que podem ser explorados, por exemplo o uso da rede TCP/IP que o torna vulnerável a possíveis ataques devido a miniaturização que dificulta a construção de mecanismos complexos de segurança. Com essas dificuldades, o sistema fica vulnerável a ataques de *ransomware*, um software malicioso que, quando dentro de um dispositivo, sequestra todos os dados armazenados e se alastra por toda a rede na qual o dispositivo está conectado. O criminoso responsável então pede um resgate das informações em dinheiro ou criptomoedas.

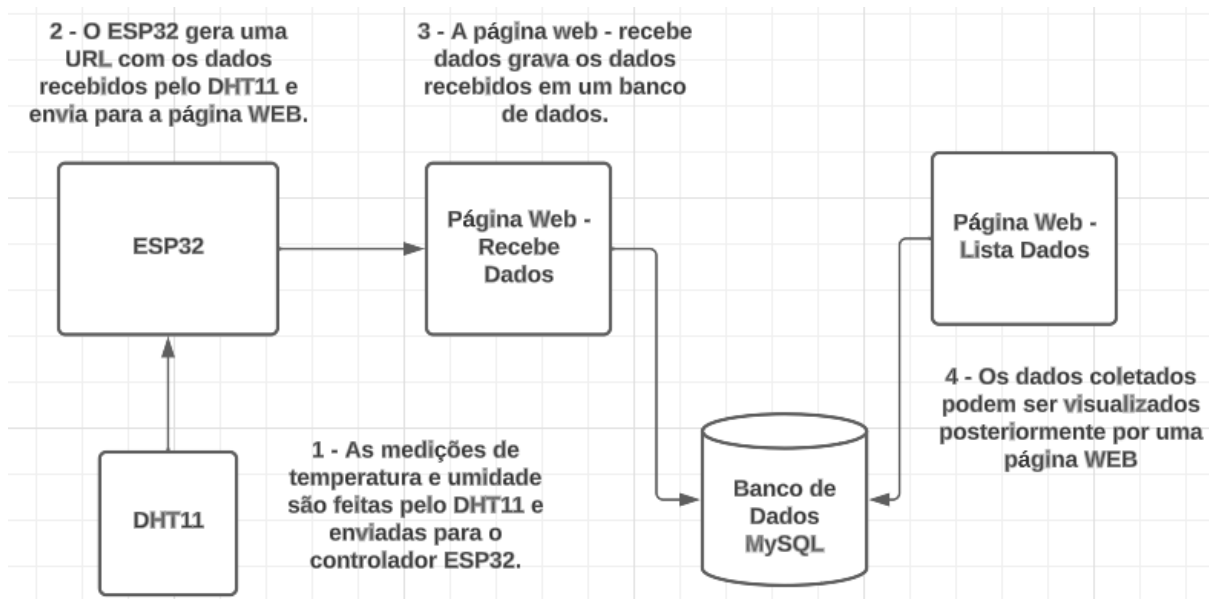
### 3 – Metodologia

O objetivo principal deste trabalho é coletar, armazenar e analisar os dados de temperatura e umidade em um ambiente específico e garantir a segurança dos dados, com o uso de criptografia para proteger as informações durante todo o processo de transmissão de informações, evitando vulnerabilidades que poderiam ser exploradas por invasores, como captura de pacotes com informações sensíveis através da *query string* devido a falta de criptografia.

Para tal, utilizamos um sensor DHT11 para captar dados de temperatura e umidade em um ambiente. O sensor DHT11 é capaz de fornecer leituras precisas dessas grandezas. Esses dados serão coletados por um microcontrolador ESP32, que possui processamento adequado para esse propósito. Após a coleta dos dados, eles são enviados para um banco de dados dentro de uma conexão segura com o protocolo de segurança aplicado em sites (HTTPS) para garantir a navegação segura através do padrão de criptografia SSL/TLS (*Secure Sockets Layer/Transport Layer Security*) que usa criptografia RSA (*Rivest-Shamir-Adleman*), um algoritmo assimétrico, ou seja, que possui um par de chaves únicas chamadas pública e privada. A chave pública é usada para criptografar informações de forma segura, enquanto a chave privada é usada para descriptografar essas informações. A combinação das duas chaves permite a comunicação segura e a proteção de dados confidenciais na internet e em outros sistemas que utilizam criptografia assimétrica.

Dessa forma, o projeto oferece uma solução automatizada para monitorar as condições de temperatura e umidade em um ambiente específico. A criptografia dos dados garante a segurança das informações coletadas, dificultando assim o acesso as informações e modificações por um possível atacante, ficando mais claro no diagrama apresentado na figura 11.

**Figura 11 – Diagrama das etapas do projeto**



Fonte: Elaborado pelos Autores

### 3.1 - Montagem do Ambiente sem Criptografia

A montagem do ambiente foi realizada em 3 passos, onde o primeiro passo foi a criação do banco de dados, o segundo a criação da página *web* utilizando a linguagem de programação PHP, o terceiro a criação do script no ESP32 utilizando a linguagem de programação Arduino e por fim como quarto e último passo a conexão dos serviços.

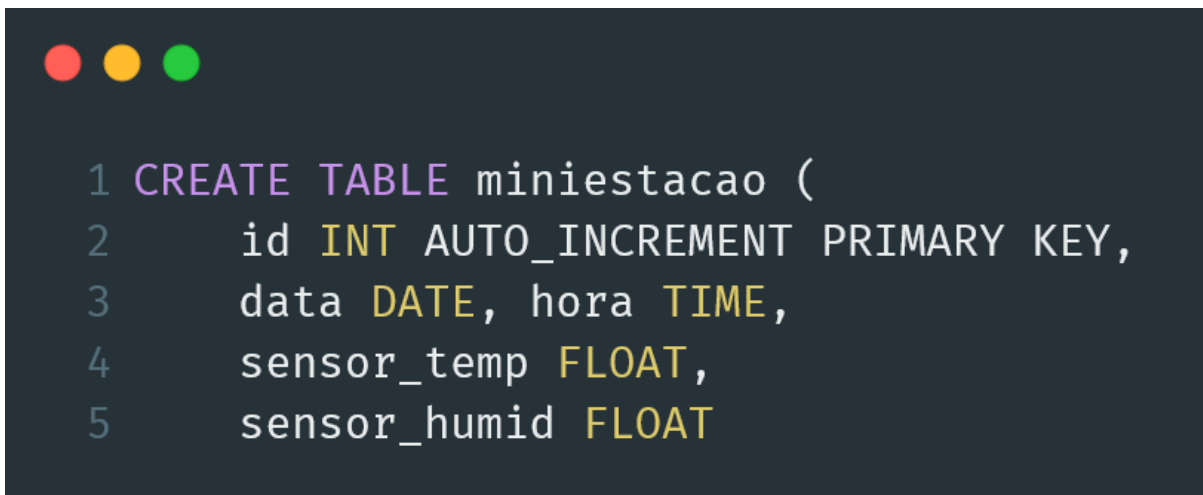
### 3.2 - Criação do Banco de Dados

Para criação e manipulação foi utilizado o serviço de banco de dados MySQL do XAMPP que utiliza o phpMyAdmin, um administrador de bancos de dados em MySQL.

A criação do Banco de Dados foi feita pela interface gráfica, enquanto a tabela e os campos foram por meio da linguagem SQL, segundo apresentado na figura 12.

Com o banco de dados criado, é possível receber as informações capturadas pelo sensor de temperatura DHT11 e transmitidas pelo microcontrolador ESP32, além de ficarem disponíveis para análise.

Figura 12 – Código em SQL para criação do banco de dados



```
1 CREATE TABLE miniestacao (  
2     id INT AUTO_INCREMENT PRIMARY KEY,  
3     data DATE, hora TIME,  
4     sensor_temp FLOAT,  
5     sensor_humid FLOAT
```

Fonte: Elaborado pelos autores

### 3.3 – Criação do Servidor Web

Foi utilizado o PHP para criação do servidor Web onde foi hospedado no serviço apache do XAMPP. O servidor é dividido em três páginas PHP, sendo a primeira o “index.php”, acessada quando conectam-se ao site. A segunda é receptora dos dados enviados pelo ESP32, denominada “recebe\_dados\_get.php” e a terceira faz a listagem dos dados, chamada “listar\_valores.php”. Como nesse caso o processo mais relevante é a página que irá receber os dados e gravar no nosso Banco de Dados, será apresentado somente os trechos de seu código.

Figura 13 – Código em PHP para recebimento de dados.

```
1 $temp=test_input($_GET["sensor_temp"]);
2 $humid=test_input($_GET["sensor_humid"])
3 $dataAtual=date("Y-M-D");
4 $horaAtual=date('H:i:s');
```

Fonte: Elaborado pelos autores

Na figura 13 é abordado as variáveis (temp, humid) que capturam os dados enviados na *QueryString* pelo ESP32 através do método *get*, enquanto dataAtual e horaAtual é gerada pelo próprio sistema enquanto a figura 14 se refere ao código responsável pela gravação dos dados no banco de dados.

Figura 14 – Código em PHP para gravação de dados recebidos.

```
1 $sql="INSERT INTO miniestacao (data,hora,sensor_temp,sensor_humid)
2 VALUES ('$dataAtual','$horaAtual','$temp','$humid')",
```

Fonte: Elaborado pelos autores

### 3.4 – Utilização de dados captados pelo Microcontrolador ESP 32

Figura 15 – Código Arduino para coleta de dados de temperatura e umidade

```
1 bool getTemperatura() {
2   TempAndHumidity newValues = dht.getTempAndHumidity();
3   if (dht.getStatus()≠0){
4     Serial.println("DHT11 error status:" + String(dht.getStatusString()));
5     return false;
6   }
7   sensor_temp=newValues.temperature;
8   sensor_humid=newValues.humidity;
9   return true;
10 }
```

Fonte: Elaborado pelos autores.

Figura 16 – Código Arduino para conexão

```
1 if ((WiFi.status() = WL_CONNECTED)) {
2   HTTPClient http;
3   String minhaURL = "http://";
4   minhaURL.concat("192.68.5.10");
5   minhaURL.concat("/projeto/recebe_dados_get.php?sensor_temp=");
6   minhaURL.concat(String(sensor_temp));
7   minhaURL.concat("&sensor_humid=");
8   minhaURL.concat(String(sensor_humid));
9   http.begin(minhaURL);
10  int httpCode = http.GET();
11  if (httpCode > 0) {
12    if (httpCode == HTTP_CODE_OK) {
13      String resultadoDaPaginaWeb = http.getString();
14      USE_SERIAL.println("=====");
15      USE_SERIAL.println(".....Resultado Recebido da página.....");
16      USE_SERIAL.println(resultadoDaPaginaWeb);
17      USE_SERIAL.println("*****Final do Conteudo recebido *****");
18    }
19  }
20  http.end();
21 } else {
22   USE_SERIAL.println("[HTTP] Sem conexão Wi-Fil, reinicie o ESP32 ou verifique o nome e senha da Rede WiFi \r\n");
23 }
```

Fonte: Elaborado pelos autores

Na figura 15 é obtido do sensor DHT11 a temperatura e umidade através do método `dht.getTempAndHumidity()` e na figura 16 é verificada a conexão via *wireless* do ESP32 e os

dados são enviados para a página “recebe\_dados\_get.php”, caso não haja a conexão, é disparada uma mensagem de erro.

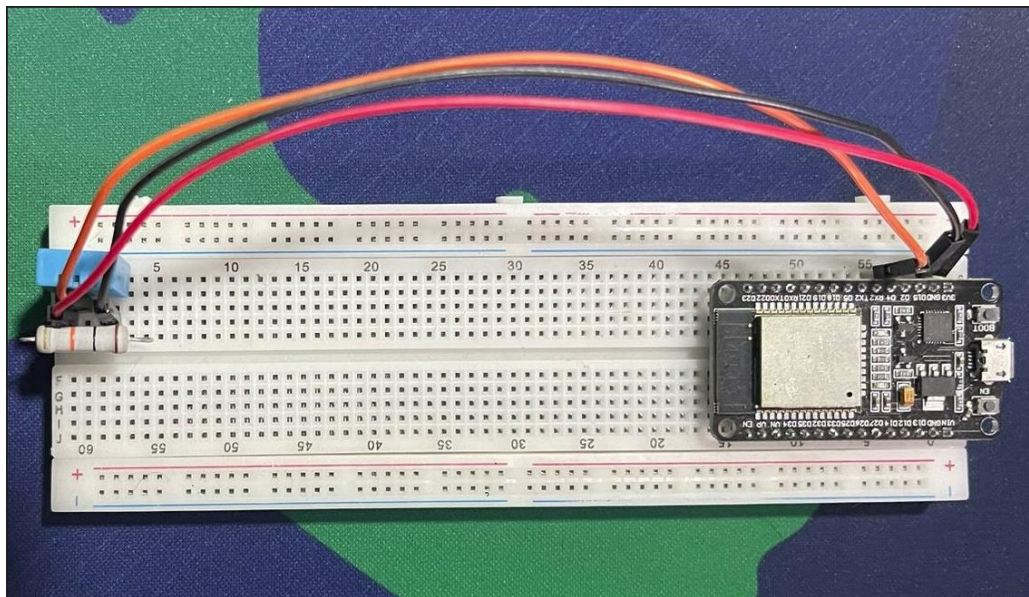
### 3.5 – Implementação da Criptografia

Para a criptografia foi contratado um plano na *HostGator*, uma plataforma de hospedagem para sites que oferece um domínio e um certificado SSL/TLS. A mudança significativa é na *url* gerada pelo ESP32, onde anteriormente era passado o IP local do XAMPP e agora passa a ser o domínio do site, que responde pelo protocolo HTTPS, tornando a conexão criptografada durante a transição das informações.

## 4 - Resultados e Discussões

O ambiente foi montado em um espaço fechado com Ar-condicionado que serviu para manter a temperatura do ambiente estável conforme mostrado na figura 17.

Figura 17 – ESP32 e sensor DHT11 conectados a protoboard

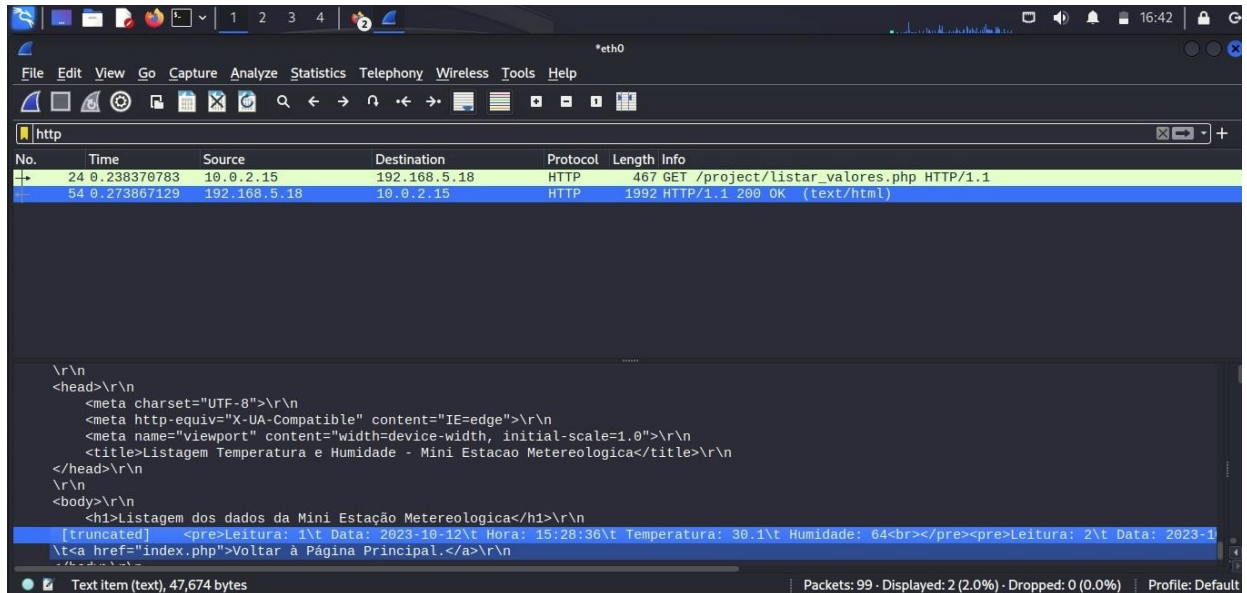


Fonte: Elaborado pelos autores

No primeiro cenário sem criptografia, realizando a transição pelo protocolo HTTP através da ferramenta *Wireshark*, foi possível visualizar os dados de temperatura, umidade e seus horários de leitura trafegando pela rede de maneira aberta como mostra a figura 18.

Como citado, essa é uma falha de segurança grave, pois informações trafegando sem nenhuma criptografia podem ser acessadas e até mesmo alteradas.

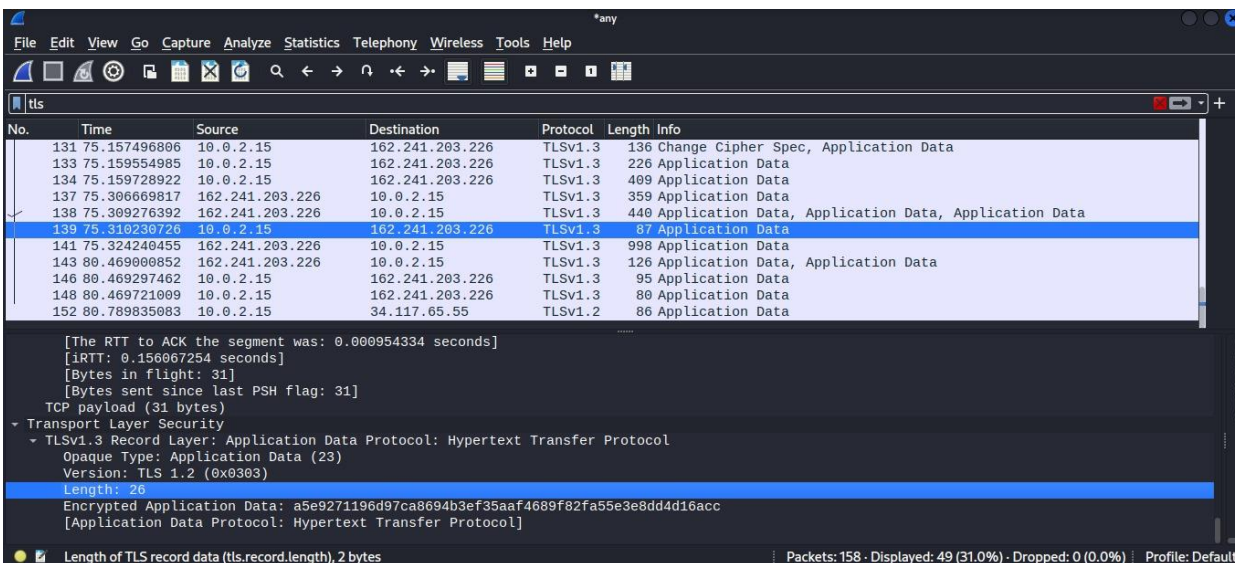
**Figura 18 – Scan realizado com Protocolo HTTP utilizando Wireshark**



Fonte: Elaborado pelos autores

Já no segundo cenário, com a implementação da criptografia e utilizando o protocolo HTTPS, foi executado um *SCAN* com a ferramenta *Wireshark* com o filtro TLS como apresentado na figura 19. As informações repassadas pelo sensor ESP32 para o banco de dados ficaram criptografadas e consequentemente ocultas, impedindo assim o acesso a qualquer dado transferido.

Figura 19 – Scan realizado com Protocolo HTTPS utilizando Wireshark



No.	Time	Source	Destination	Protocol	Length	Info
131	75.157496806	10.0.2.15	162.241.203.226	TLSv1.3	136	Change Cipher Spec, Application Data
133	75.159554985	10.0.2.15	162.241.203.226	TLSv1.3	226	Application Data
134	75.159728922	10.0.2.15	162.241.203.226	TLSv1.3	409	Application Data
137	75.306669817	162.241.203.226	10.0.2.15	TLSv1.3	359	Application Data
138	75.309276392	162.241.203.226	10.0.2.15	TLSv1.3	448	Application Data, Application Data, Application Data
139	75.310230726	10.0.2.15	162.241.203.226	TLSv1.3	87	Application Data
141	75.324240455	162.241.203.226	10.0.2.15	TLSv1.3	998	Application Data
143	80.469000852	162.241.203.226	10.0.2.15	TLSv1.3	126	Application Data, Application Data
146	80.469297462	10.0.2.15	162.241.203.226	TLSv1.3	95	Application Data
148	80.469721009	10.0.2.15	162.241.203.226	TLSv1.3	86	Application Data
152	80.789835083	10.0.2.15	34.117.65.55	TLSv1.2	86	Application Data

[The RTT to ACK the segment was: 0.000954334 seconds]  
 [iRTT: 0.150067254 seconds]  
 [Bytes in flight: 31]  
 [Bytes sent since last PSH flag: 31]  
 TCP payload (31 bytes)  
 - Transport Layer Security  
   - TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol  
     Opaque Type: Application Data (23)  
     Version: TLS 1.2 (0x0303)  
     Length: 26  
     Encrypted Application Data: a5e9271196d97ca8694b3ef35aaf4689f82fa55e3e8dd4d16acc  
     [Application Data Protocol: Hypertext Transfer Protocol]

Length of TLS record data (tls.record.length), 2 bytes

Packets: 158 · Displayed: 49 (31.0%) · Dropped: 0 (0.0%) · Profile: Default

Fonte: Elaborado pelos autores

## 5 – Considerações Finais

Com base nos resultados obtidos, podemos concluir que a implementação de criptografia através do uso do protocolo HTTPS que usa o padrão de criptografia SSL/TLS, tornou a transição dos dados gerados pelo ESP32 para o servidor web seguros. A solução apresentada neste artigo é capaz de coletar, armazenar e analisar os dados de temperatura e umidade em um ambiente específico e garantir a segurança dos dados, com o uso de criptografia para proteger as informações durante todo o processo de transmissão de informações, evitando vulnerabilidades que poderiam ser exploradas por invasores. Apesar dos resultados obtidos serem positivos, surge a necessidade de mitigar outra vulnerabilidade recorrente em redes de comunicação IoT, a falta de autenticação dos dispositivos, uma vez autenticado abre a possibilidade de enviar informações para servidor, sendo um ponto que pode ser explorado em um trabalho futuro. Portanto, é importante que as soluções de segurança em IoT sejam constantemente atualizadas e aprimoradas para garantir a proteção dos dados e a privacidade e, somado a essas medidas, também é necessário que os usuários estejam cientes dos riscos envolvidos na utilização de dispositivos IOT e adotem boas práticas de segurança, como a atualização regular de *firmwares* e a escolha de senhas fortes.



### Referências

APACHEFRIENDS. Apache Friends - Sobre. 2023. Disponível em: [https://www.apachefriends.org/pt\\_br/about.html](https://www.apachefriends.org/pt_br/about.html). Acesso em: 20 de setembro de 2023.

ARDUINO&CIA. Sensor de Umidade e Temperatura DHT11. 2013. Disponível em: <https://www.arduinoecia.com.br/sensor-de-umidade-e-temperatura-dht11/>. Acesso em: 08 de maio de 2023.

ESPRESSIFSYSTEMS. ESP32. 2023. Disponível em: <https://www.espressif.com/en/products/socs/esp32/>. Acesso em: 08 de maio de 2023.

HOSTGATOR. Protocolo HTTP: entenda o que é e para que serve! 2023. Disponível em: <https://www.hostgator.com.br/blog/o-que-e-protocolo-http/>. Acesso em: 20 de setembro de 2023.

MAKERHERO. Protoboard 830 furos. 2023. Disponível em: <https://www.makerhero.com/produto/protoboard-400-pontos/>. Acesso em: 16 de setembro de 2023.

MYSQL. MySQL Logo Downloads. 2023. Disponível em: <https://www.mysql.com/about/legal/logos.html>. Acesso em: 20 de setembro de 2023.

PAULA, A. F. Monitorando dados e gerenciando alertas em sistemas de iot. Universidade Federal do Rio Grande do Norte, Universidade Federal do Rio Grande do Norte, 2020.

PHP. Official PHP Logos. 2023. Disponível em: <https://www.php.net/download-logos.php>. Acesso em: 20 de setembro de 2023.

PHP. PHP General Information. 2023. Disponível em: <https://www.php.net/manual/en/faq.general.php>. Acesso em: 20 de setembro de 2023.

SILVA, D. P. Análise de métodos de detecção de ataques de ransomware em dispositivos iot em redes tcp/ip. Instituto Federal de Ciência e Tecnologia de Pernambuco, Instituto Federal de Ciência e Tecnologia de Pernambuco, 2020.

VAILSHERY, L. S. Number of IoT connected devices worldwide 2019-2021, with forecasts to 2030. 2022. Disponível em: <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>. Acesso em: 08 de maio de 2023.

VIDADESILICIO. ESP32 NodeMcu – DevKit v1. 2023. Disponível em: <https://www.vidadesilicio.com.br/wp-content/uploads/2021/09/2494-640x640.jpg>. Acesso em: 08 de maio de 2023.

WIRESHARK. Wireshark Frequently Asked Questions. 2023. Disponível em: [https://www.wireshark.org/faq.html#\\_what\\_is\\_wireshark](https://www.wireshark.org/faq.html#_what_is_wireshark). Acesso em: 20 de setembro de 2023.