UAI.PY – A INTERNET DAS COISAS NA REATRIBUIÇÃO DE VALOR

Matheus Martins de Sousa[1], André Luiz Vicente Silva[2], Alex Medeiros de Carvalho[3]

[1] matheus.martins@estudante.iftm.edu.br, [2] andreluiz.vicente@ufu.br, [3] alex.carvalho@ufu.br

[1] INSTITUTO FEDERAL DO TRIÂNGULO MINEIRO – *campus* UBERLANDIA
[2] UNIVERSIDADE FEDERAL DE UBERLÂNDIA
[3] ESCOLA DE EDUCAÇÃO BASICA – UFU
Uberlândia – MG

Categoria: ARTIGO SUPERIOR / MULTIMÍDIA

Resumo: A crescente demanda por equipamentos com maior poder computacional, associado à obsolescência natural, resulta em uma pressão para o descarte de diversos componentes eletrônicos. Ainda, a presença de produtos apreendidos dada sua origem pirata e/ou utilização para atividades envolvendo pirataria, eleva ainda mais a quantidade de lixo eletrônico gerado. Diante dessa perspectiva, o presente artigo busca detalhar a concepção de uma solução proposta para o projeto Além do Horizonte, da Receita Federal do Brasil. O objetivo é conferir novas utilidades aos equipamentos TVBox apreendidos em operações realizadas pelo órgão governamental, totalizando mais de 100.000 aparelhos. Para atingir tal fim, as TVBox passam por um processo de descaracterização e recebem uma distribuição Linux. Em sequência, são adicionados ferramentas e algoritmos que a tornam capaz de receber informações de projetos baseados na plataforma Arduino, ordená-las e transmitir os dados para a nuvem, constituindo uma solução de Internet das Coisas (IoT). Mediante essa abordagem, é viável potencializar projetos desenvolvidos no âmbito científico e educacional, aproveitando as placas TVBox de forma renovada e, por conseguinte, reduzindo o volume de lixo eletrônico gerado.

Palavras Chaves: TVBox, IoT, Lixo eletrônico, Arduino.

Abstract: *The increasing demand for equipment with greater computational power, coupled with natural obsolescence, results in pressure for the disposal of various electronic components. Additionally, the presence of seized products due to their counterfeit origin and/or use in activities involving piracy further exacerbates the amount of electronic waste generated. In light of this perspective, the present article aims to detail the conception of a proposed solution for the "Beyond the Horizon" project by the Brazilian Federal Revenue. The goal is to confer new utilities to TVBox equipment seized in operations conducted by the governmental agency, totaling over 100,000 devices. In order to achieve this, the TVBox devices undergo a process of discharacterization and are equipped with a Linux distribution. Subsequently, tools and algorithms are added that make it capable of receiving information from projects based on the Arduino platform, sorting this data, and transmitting it to the cloud, constituting an Internet of Things (IoT) solution. Through this approach, it is feasible to enhance projects developed in both scientific and educational realms, leveraging the TVBox boards in a renewed manner and consequently reducing the volume of electronic waste generated.*

Keywords: *TVBox, Internet of Things, Eletronic waste, Arduino.*

1 INTRODUÇÃO

O desenvolvimento técnico e tecnológico não trás a sociedade apenas bonificações de sua escalada, mas também é reponsável por alavancar a emergência de desafios globais, tal qual o descarte e destinação adequada de lixos eletrônicos. Em 2019 (FORT V. et al., 2021), calcula-se que 82,6% de todo o lixo eletrônico gerado, correspondente a 44,3 milhões de toneladas, não foram descartados de forma adequada. Franz e Silva (2022) destacam o impacto ambiental diante do incorreto destino dado ao lixo eletrônico que, pela presença de substâncias tóxicas e metais pesados, são fontes de contaminação para o ambiente e geram riscos a saúde humana e animal.

Alinhando-se á essa prerrogativa, surge o projeto “Além do Horizonte”, uma iniciativa promovida pela Receita Federal do Brasil, e executada por instituições federais de ensino superior, cujo objetivo é conferir destinação adequada á equipamentos eletrônicos TVBox apreendidos em operações de combate à pirataria (ANATEL, 2022). Esses dispositivos são originalmente dotados de um sistema operacional Android capaz de acessar canais televisivos que exigem assinatura, sem concessão da transmissora. É estimado que existam mais de 1,4 milhão de equipamentos TVBox em circulação no pais. Em operações anteriores ao projeto, a Receita Federal apreendeu e destruiu mais 97 mil aparelhos do tipo (BRASIL, 2021).

Visando colaborar com o enfrentamento á esse desafio, o presente trabalho têm em sua essência, a prerrogativa de demonstrar mecanismos que permitem evitar o descarte das TVBox apreendidas, atribuindo um novo uso para esses equipamentos.

A ideia proposta têm inspiração na ascensão das aplicações que se integram ao ecossistema de internet das coisas (IoT – *Internet of things*). IoT é definido por uma estrutura capaz de fundir os domínios físico e virtual, e recorre ao uso da internet como mecanismo de comunicação e transmissão de informações. Para isso, são utilizados protocolos capazes de gerir, ordenar e manipular os dados sem a interferência humana (S. Balaji et al., 2019).

Portanto, espera-se evitar a destruição de uma parcela ou totalidade desses equipamentos apreendidos, reduzindo o lixo eletrônico gerado por meio da consolidação de técnicas que permitem colocá-los como ferramentas para instaurar ou ampliar soluções que integrem um sistema IoT.

2 O TRABALHO PROPOSTO

O presente artigo tem suas prerrogativas na ideia de atribuir um uso alternativo ao hardware presente em equipamentos TVBox. Diante dessa oportunidade, foi desenvolvido um algoritmo capaz de receber informações emitidas pela porta serial de uma plataforma Arduino, conectado via USB 2.0 da TVBox, e enviá-las para um servidor Web. Para isso, foi necessário substituir o software que acompanha nativamente as TVBox, implementar uma série de algoritmos desenvolvidos para conectar a TVBOX à internet de forma segura. Ainda foi necessário modelar um projeto na plataforma arduíno que se alinha com o viés acadêmico de uma das instiuições que acomodam o projeto, dando origem ao equipamento nomeado como "UAI.py"

3 MATERIAIS E MÉTODOS

O trabalho foi desenvolvido na parceria entre o Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro (IFTM), *campus* Uberlândia e o Laboratório de Robótica e Inteligência Artificial (RIVED) pertencente á Universidade Federal de Uberlândia. A execução de todas as etapas descritas a seguir ocorreu durante o período de fevereiro de 2022 á agosto de 2023.

O desenvolvimento do projeto pode ser ordenado, de forma não linear á execução, em três etapas: (1) estabelecimento das informações a serem extraídas pelo ambiente físico; (2) preparação da TVBox para a aquisição do algoritmo; (3) Elaboração do algoritmo de conexão entre os domínios.

3.1 Sensoriamento

Para a validação da ideia como um projeto capaz de interligar os domínios físico e virtual, foi desenvolvida uma solução com potencial uso no âmbito do IFTM *campus* Uberlândia. Dada as raízes da instituição como uma escola agrotécnica, foi projetado um equipamento capaz de estimar algumas informações climáticas, com o intúito de auxiliar o manejo da irrigação em platações do *campi* com maior exatidão.

3.1.1 Confecção de um pluviômetro

Por meio da análise de um pluviômetro comercial tipo báscula, pertencente ao IFTM *campus* Uberlândia, foi desenvolvido um produto análogo, com modelo apresentado na figura 01.

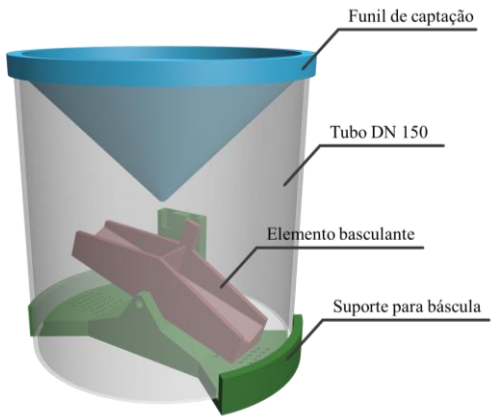

Figura 01 – Visão 3D do pluviômetro desenvolvido

O princípio de funcionamento desse tipo de equipamento é descrito por Blainski et al., (2012). Em eventos pluviométricos, a água é coletada pelo funil, que conduz o volume para um dos dois compartimentos da báscula, alinhado com o centro do funil. À medida que esse compartimento se enche, haverá uma força peso exercida sobre a peça, que induz ao movimento em torno do eixo de rotação, permitindo que água armazenada seja despejada para fora e o outro compartimento se alinhe com o centro do funil.

De posse do volume necessário para que um evento de movimento da báscula ocorra, torna-se possível estimar a altura pluviométrica, por meio da equação 1.

$$L_h = \frac{n_h \times V}{A_c} \times 10 \qquad \text{(Eq.1)}$$

Onde,

- L_h – altura pluviométrica durante o perído *h*, em mm;
- n_h – número absoluto de "basculadas" registradas durante o período "h";
- V – volume por basculada, em cm^3;
- A_c – área de coleta de chuva do pluviômetro, em cm^2;

Para quantificar a variável "n_h", na porção central da báscula, deslocado positivamente no eixo Z, foi instalado um ímã, que acompanha o movimento do elemento basculante. No suporte da báscula, foram instalados dois sensores hall, sensíveis ao campo magnético gerado, alinhando-se em cada um dos dois extremos de movimento da báscula.

Quando a báscula está pendente para um dos lados, o sensor hall alocado no mesmo lado sofrerá influência do ímã. Portanto, a mudança do sensor excitado, é o indicativo do movimento da báscula, proveniente da coleta de água pelo funil, representando uma adição á variável "n_h".

3.1.2 Informações coletadas

Para receber essa lógica e monitorar os sensores hall, faz-se uso da plataforma Arduino Uno R3, baseada no microcontrolador ATmega328. As informações coletadas incluem a umidade e temperatura do ar, estimadas por meio de um sensor DHT 11, e o número de basculadas efetuadas, informação obtida por meio de dois sensores de efeito hall, modelo 49e. A programação concebida e implementada no Arduíno é responsável por mensurar, em intervalos de 60 segundos, a temperatura e umidade instânea e calcular, com base no número de basculadas registradas, a altura pluviométrica. Esses dados são então impressos em formato JSON, por meio da porta serial.

3.2 Preparo da TVBox: Descaracterização

Os equipamentos TVBox utilizados na execução do projeto foram cedidos pela Receita Federal do Brasil ao IFTM *campus* Uberlândia, como parte do projeto "Além do Horizonte". Diante dos diferentes modelos de equipamentos disponibilizados, foi realizado um levantamento técnico com testes de estresse para avaliar quais possuiam hardware com maior capacidade computacional. Em posse dos resultados, foram selecionados os modelos TX6-p e TX9. Ambas compartilham algumas especificações, incluindo:

- SoC: Amlogic S905w;
- Arquitetura: ARMv8-A (64-bit);
- Processador: 4X ARM Cortex-A53 @ 2GHz
- Memória RAM: 2 GB (DDR3)
- Memória Interna: 32/64 GB

A descaracterização é o processo obrigatório para uso das TVBox disponibilizadas, que consiste em retirar quaisquer

ferramentas de software que permitam o acesso aos canais televisivos de forma pirata. Perante á essa necessidade, foi realizada uma pesquisa a fim de verificar sistemas operacionais (SO) compatíveis com o microchip presente nas placas e que permitisse a execução das etapas seguintes.

O Armbian® é uma distribuição computacional que permite a construção de *kernels* Linux com suporte a microprocessadores de arquitetura ARM. Existe uma comunidade de pessoas que atualmente contribuem na execução de *kernels* com base no armbian e que sejam compatíveis com os chips presentes nas TVBox comercializadas (AMLOGIC et al., 2023).

A instalação do *kernel* linux na TVBox é um processo dividido em duas partes:

3.2.1 Preparo das imagens linux

Com base em Amlogic et al. (2023), foram selecionados os *kernels* com melhor desempenho computacional para as TVBox utilizadas, descritos na tabela 01.

Tabela 1 – Imagens linux utilizadas.

Placa	Imagem	Versão
TX6-p	Armbian 20.05.3 Arm-64 buster	5.7.0
TX9	Armbian 20.10 Arm-64 bullseye	5.9.0

As imagens eram copiadas para um cartão micro-sd, que também recebia um arquivo de software *u-boot* modificado para cada uma das placas. O arquivo *u-boot* atua como um carregador de inicialização primário, responsável por empacotar as instruções para a correta inicialização do *kernel* do sistema operacional no dispositivo.

3.2.2 Boot e gravação na memória interna

Em sequência, a TVBox é inicializada com o cartão microSD inserido, permitindo o *boot* com a imagem Linux escolhida. Após inicialização, é acessado o super-usuário para realizar uma modificação do arquivo “install-aw.sh”. Nessa etapa, altera-se a origem do SO da memória “eMMC” para a presente no cartão microSD. Esse procedimento permite sobregravar o ardroid da TVBox com a imagem Amrbian e efetivar o processo de descaracterização.

3.3 O algoritmo da UAI.py

Para ser possível a conexão entre o ambiente físico e o armazenamento dos dados na internet, é necessário uma série de conexões lógicas que acontecem virtualmente tanto em ambiente local, com os dados recebidos do ambiente físico através da do que nomeamos por UAI.py (dispositivo descaracterizado), quanto no ambiente remoto, através de um servidor exposto de forma autenticável e segura na internet.

3.3.1 Domínio Virtual Local

Segundo Couloris, G.; Dollimore, J.; Kindberg, T (2000), autores do livro "Distributed Systems: Concepts and Design", o termo "ambiente local" se refere à uma área física ou geográfica em que um conjunto de dispositivos operam e se interagem. É um espaço em que os componentes estão fisicamente próximos uns aos outros e em virtude disto, permite a comunicação e a cooperação entre esses componentes com uma menor latência. Define-se por latência o atraso ou o período de tempo que leva para uma ação ou um evento acontecer após ter sido iniciado (TANEMBAUM, A.; FEAMSTER, N.; WETHERALL, D., 2011).

Com base nesses princípios, o ambiente local tem a importante responsabilidade de gerenciar os dados captados pelo arduino e enviá-los para a nuvem, prezando por uma baixa latência. Para isso, o dispositivo descaracterizado opera como um gerenciador local, orquestrando e distribuindo os dados captados no ambiente físico. A partir da figura 02, é possível visualizar a sequência de processos que ocorrem no ambiente local, tendo como maestro o dispositivo UAI.py.

Para realizar a leitura do arduíno, foi desenvolvido um algoritmo em JavaScript responsável por ler a porta serial com os dados enviados pelo arduino em formado *JSON (JavaScript Object Notation)*. A partir desta leitura, o script endereça os dados captados para um servidor hospedado na própria UAI.py, serviço este que opera sobre o protocolo *HTTP (Hypertext Transfer Protocol)* e pode ser acessado pela porta 3000. Um fator importante da arquitetura é que o script e o servidor express são sistemas completamente independentes e cooperam entre si para que os dados sejam armezenados localmente de forma distribuída, onde ambos são executados em paralelo a nível de execução dentro da UAI.py.

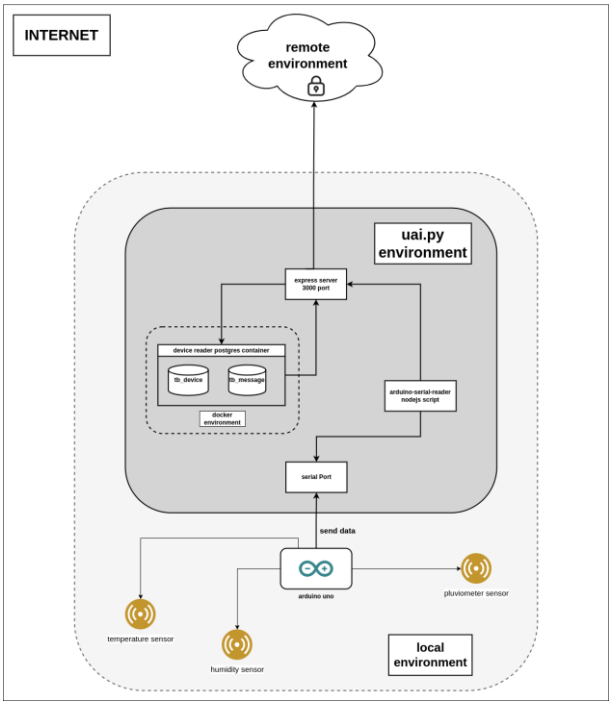

Figura 02 – Representação arquitetural do ambiente local hospedado pela uai.py

A partir do momento em que os dados chagaram na porta 3000, o servidor Express operado através do NodeJs processa a requisição *HTTP* tendo como base o corpo da requisição, onde se é identificado os dados lidos pelo arduino em formato *JSON* e a identificação do dispositivo que enviou os dados (neste caso o dispositivo foi o arduino). Após a leitura do corpo da requisição, o servidor estabelece uma conexão com um banco de dados local para salvar as informações recebidas. Cabe ressaltar que este banco de dados local é hospedado através de um container postgres, o que possibilita que esse banco de dados seja executado em um ambiente independente e isolado dentro da UAI.py (BURNS B., 2018).

Após escrever os dados no container postgres, o servidor opera o envio desses para o ambiente remoto. Para isso, ele faz a identificação dos dados que precisam ser enviados e realiza uma

requisição *HTTP* para o servidor remoto, responsável por armazenar agrupamentos de dados advindos de diferentes dispositivos UAI.py. Na figura 01, há a representação de um cadeado associado ao o servidor, representando que apenas dispositivos autenticados podem registrar dados naquele ambiente. Portanto, o dispositivo UAI.py precisa sempre enviar sua assinatura digital para a requisição ser válida, caso contrário, o servidor recusará a requisição.

3.3.2 Domínio Virtual Remoto

De acordo com Couloris, G.; Dollimore, J.; Kindberg, T (2000), define-se por "ambiente remoto" uma área física ou geográfica em que os componentes do sistema estão fisicamente distantes uns dos outros, exigindo que a transmissão de dados seja realizado através de redes, como a Internet. Em virtude disso, ambientes remotos geralmente envolvem uma maior latência na comunicação devido sua distância geográfica.

Com base neste princípio, o ambiente remoto tem a importante responsabilidade de gerenciar com segurança e autenticidade todos os dados recebidos de diferentes dispositivos ao redor do mundo e armazena-los de maneira permanente em uma máquina exposta para a internet. Para isso, foi desenvolvido um servidor baseado em NodeJs hospedado de forma segura na nuvem capaz de receber requisições *HTTP* pela internet e se comunicar com um banco de dados postgres, também hospedado de forma segura na nuvem. Visando garantir a autenticidade de dados e requisições recebidos, o servidor autentica cada requisição com base nas credenciais de acesso criptografadas em um token (Figura 03).

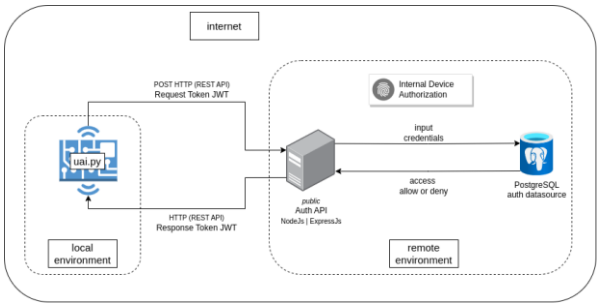

Figura 03 – Autenticação entre a UAI.py e servidor remoto

O processo de autenticação das mensagens enviadas pela UAI.py para o servido remoto (Figura 03), inicia com uma requisição por parte da UAI.py via servidor, complartilhando suas credenciais (identificação e senha), até a validação dessas pelo banco de dados. Em casos de credenciais inválidas, não é gerado um token de requisição e consequentemente, o acesso aos dados é negado. Caso os dados sejam válidos, é retornado um token de acesso no corpo da resposta da requisição. A partir do acesso, a UAI.py armazena o token no banco de dados local e o reaproveita em todas as suas requisições de envio dos dados para o ambiente remoto, processo este que pode ser visualizado na Figura 04.

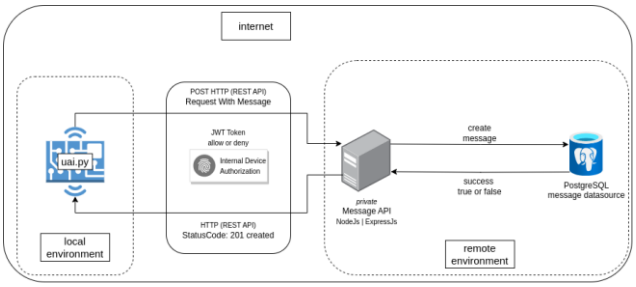

Figura 04 – Representação arquitetural descrevendo uma requisição autenticada da uai.py para um servidor remoto

Na figura 04, "*message*" é o nome atribuído a um conjunto de dados coletados pelo arduíno e enviados para a UAI.py. A partir do disparo de dados pela UAI.py, junto ao token de acesso, há a validação desse token pelo servidor, que persiste os dados recebidos na requisição em um banco de dados postgres. Caso a UAI.py realize uma requisição enviando um lote contendo vários dados, o que pode acontecer em casos de interrupção na conexão de rede entre o ambiente local e o remoto, o servidor possui a inteligência para realizar o processamento e armazenar os dados em lote, respeitando a ordem na qual os dados foram gerados e armazenados localmente.

Por fim, para acessar os dados armazenados no servidor remoto, é necessário realizar uma requisição para uma rota pública disponível no servidor. Por se tratar de uma rota pública, o servidor não exige token de acesso e como consequência, qualquer pessoa com acesso a internet pode ler os dados enviados pelo arduino e orquestrados pela UAI.py (Figura 05).

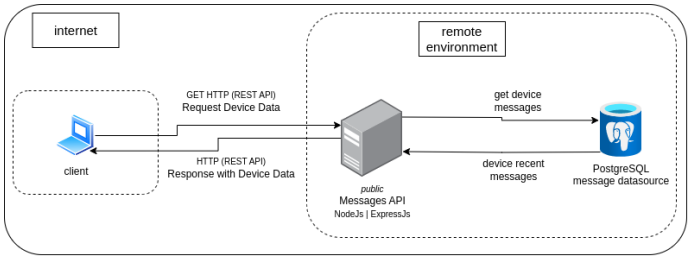

Figura 05 – Fluxo de requisição dos dados armazenados

A série de processos representados na figura 05 se desenvolve da esquerda para a direita, desde a requisição realizada pelo client (representado na imagem como um notebook) solicitando os dados, a leitura do banco de dados postgres pelo servidor, que retorna todos os dados registrados. Caso a requisição seja realizada com sucesso, o servidor retorna para o client (para o usuário) todos os registros recentes de um dispositivo, finalizando todo o fluxo. Desde o momento onde o arduino gerou os dados, até o momento onde eles foram persistidos remotamente de forma segura e autenticável, sendo disponibilizados para qualquer pessoa e em qualquer lugar do mundo desde que tenha acesso a internet.

4 RESULTADOS E DISCUSSÃO

Após todo o processo de design, modelagem e desenvolvimento, foi extraído do projeto uma série de resultados quantitativos e qualitativos a respeito de sua operação. Todo o código, endereços de acesso e documentação do que foi realizado está disponibilizado publicamente através de um repositório no github chamado UAI.py (*github.com/andrelvicent-zup/uai.py*). Através deste repositório, é possível que qualquer pessoa do mundo, com acesso a internet, realize um "clone" do código, reaproveitando toda a estrutura e arquitetura desenvolvida com a necessidade de pequenas adaptações simples para suas especificidades, como definir a porta na qual será realizada a leitura do arduino ou a URL de destino do ambiente remoto.

Além disso, através da junção de todas as etapas do processo de publicação da mensagem, que será descrito na figura 6, é possível visualizar todo o fluxo desde o momento onde os dados são captados por sensores até o momento onde os dados são consultados por algum client no final do processo. Dentro da imagem, é possível visualizar também a segregação de subredes, onde na parte inferior podemos observar em destaque o ambiente local, onde a uaipy realiza sua operação em conjunto com o arduino, e na parte superior, dentro de uma VPC (Virtual Private Cloud), um servidor remoto exposto de forma segura na

internet responsável por processar, autenticar, e armazenar as informações das mensagens enviadas pela uai.py.

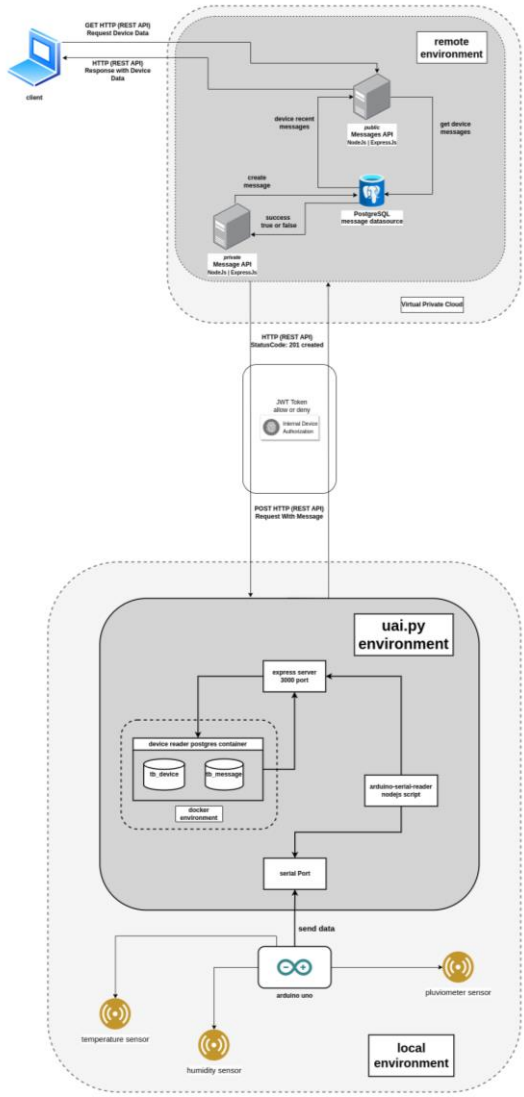

Figura 06 – Arquitetura estrutural do processo de publicação e leitura em uma abordagem expandida

O protótipo foi submetido á um teste de estresse para coleta de dados quantitativos acerca da estabilidade e confiabilidade da modelagem arquitetural aqui propostos. A UAI.py ficou responsável por receber dados de altura pluviométrica, temperatura e umidade diretamente do arduino e enviar para o servidor remoto com intervalos de 1 minutos, durante 12 horas. Como resultado deste experimento, tivemos ao todo 720 requisições de envio de mensagens da UAI.py para o servidor remoto, totalizando 721 registros armazenados.

Ao longo do intervalo de testes, a internet caiu durante 7 minutos, e neste momento, a uai.py não conseguiu enviar os dados para a nuvem imediatamente, conseguindo enviar apenas na próxima tentativa, onde ela enviou tanto o novo registro obtido pelo arduino, quanto o antigo registro que ainda não estava sincronizado remotamente, comprovando a eficiência de nosso retry orquestrado pelo servidor local, e hospedado na UAI.py em casos onde ocorre quedas de internet.

Outro serviço que também se demonstrou eficiente foi o servidor remoto, que obteve 100% de disponibilidade ao longo dos testes, isso significa que todas as vezes em que ele foi acionado, ele estava disponível, ou seja, o servidor não caiu em nenhum momento. Na figura 07, podemos observar um log retirado da UAI.py durante os primeiros minutos da bateria de testes e nele, demonstrando o servidor local em funcionamento, enviando os dados para o servidor remoto.

```
[18:29:21.624] INFO (217250): data storage:
    statusCode: 201
    serverResponse: {
      "message": "Created"
    }
[18:29:31.686] INFO (217250): data storage:
    statusCode: 201
    serverResponse: {
      "message": "Created"
    }
[18:29:41.755] INFO (217250): data storage:
    statusCode: 201
    serverResponse: {
      "message": "Created"
    }
[18:29:51.818] INFO (217250): data storage:
    statusCode: 201
    serverResponse: {
      "message": "Created"
    }
[18:30:01.867] INFO (217250): data storage:
    statusCode: 201
    serverResponse: {
      "message": "Created"
    }
[18:30:11.936] INFO (217250): data storage:
    statusCode: 201
    serverResponse: {
      "message": "Created"
    }
[18:30:21.992] INFO (217250): data storage:
    statusCode: 201
    serverResponse: {
      "message": "Created"
    }
[18:30:32.059] INFO (217250): data storage:
    statusCode: 201
    serverResponse: {
      "message": "Created"
    }
[18:30:42.122] INFO (217250): data storage:
    statusCode: 201
    serverResponse: {
      "message": "Created"
    }
```

Figura 07 – Log do servidor local

Quatro horas após o início dos testes, foi realizada uma validação para compreender se o arduino estava enviando os dados corretamente para o servidor local. Após validar que ele estava funcionando perfeitamente, também foi registrado o script realizando a leitura da porta serial do arduino e enviando para o servidor local (Figura 08).

```
[23:26:01.395] INFO (315514): serial port open on: /dev/ttyACM0
[23:26:12.996] INFO (315514): message publish to local server:
    statusCode: 201
    localServerResponse: {
      "sucess": true
    }
[23:26:23.003] INFO (315514): message publish to local server:
    statusCode: 201
    localServerResponse: {
      "sucess": true
    }
[23:26:33.010] INFO (315514): message publish to local server:
    statusCode: 201
    localServerResponse: {
      "sucess": true
    }
[23:26:43.022] INFO (315514): message publish to local server:
    statusCode: 201
    localServerResponse: {
      "sucess": true
    }
[23:26:53.029] INFO (315514): message publish to local server:
    statusCode: 201
    localServerResponse: {
      "sucess": true
    }
[23:27:03.036] INFO (315514): message publish to local server:
    statusCode: 201
    localServerResponse: {
      "sucess": true
    }
[23:27:13.044] INFO (315514): message publish to local server:
    statusCode: 201
    localServerResponse: {
      "sucess": true
    }
[23:27:23.055] INFO (315514): message publish to local server:
    statusCode: 201
    localServerResponse: {
      "sucess": true
    }
[23:27:33.062] INFO (315514): message publish to local server:
    statusCode: 201
    localServerResponse: {
      "sucess": true
    }
```

Figura 08 – Log do script responsável receber informações da porta serial

Um ponto importante a ressaltar, é que habilitamos a duração do token de acesso neste primeiro momento para durar 10 dias. Portanto, a partir do momento em que a UAI.py realizou a primeira autenticação com o servidor remoto, ela armazenou o token que durou durante todo o desenvolvimento, sendo enviado em cada uma de suas requisições.

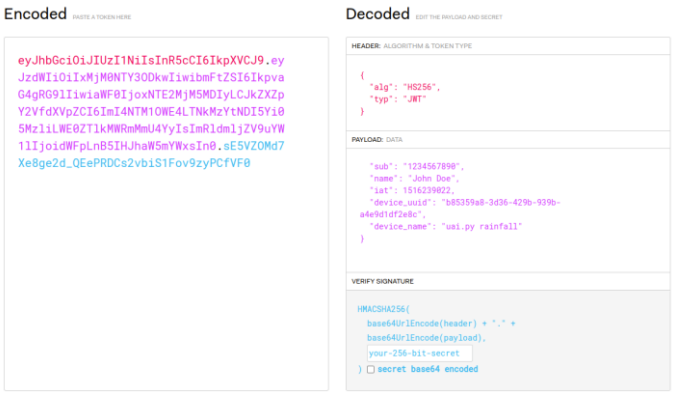

Figura 09 – Decodificação de token JWT

Na figura 09, quando analisada da esquerda para a direita, podemos perceber ao lado esquerdo o token de acesso utilizado pela UAI.py durante o experimento, já do lado direito, podemos observar suas informações decodificadas, dentre elas, está o objeto payload, com os atributos "device_uuid" e "device_name", utilizados pelo servidor remoto para identificar cada dispositivo autorizado à enviar dados para o armazenamento interno.

Contudo, através da análise dos dados quantitativos referente a performance e consistência do protótipo, conclui-se que o objetivo final do projeto foi atingido, pelo menos neste primeiro momento, cujo o objetivo era criar uma primeira versão funcional de um sistema distribuído e orquestrado, sobretudo, por um dispositivo TVBox descaracterizado apelidado carinhosamente por uai.py.

Como próximos passos, pretendemos extender a variedade de sensores que o arduino envia para a UAI.py, desta forma, podemos fazer com que um único arduino envie diversos dados diferente de um único projeto. Além de aumentar a quantidade de sensores acoplados no arduino, também temos a perspectiva de expandir o projeto para diferentes dispositivos enviando dados para o servidor local, desta forma, teríamos uma única uai.py respondendo a dois dispositivos diferentes, como por exemplo um arduino via porta serial e um ESP8266 enviando dados via wifi, ambos localmente.

5 CONCLUSÕES

À medida que concluímos este estudo, fica claro que a colaboração entre a Receita Federal, o Instituto Federal do Triângulo Mineiro e a Universidade Federal de Uberlândia não apenas trouxe resultados tangíveis, mas também lançou as bases para futuras iniciativas colaborativas. A interseção de conhecimento institucional, habilidades técnicas e experiência prática forneceu *insights* valiosos para a promoção contínua da inovação tecnológica e do progresso social. A integração através da UAI.py em projetos acadêmicos demonstra como as parcerias entre setores distintos podem criar soluções que beneficiam a sociedade como um todo, proporcionando uma visão promissora para o futuro da interdisciplinaridade e da aplicação prática do conhecimento.

6 AGRADECIMENTOS

Os autores agradecem a confiança da Receita Federal do Brasil nas instituições de ensino participantes do projeto, permitindo o desenvolvimento colaborativo e multuo da sociedade.

REFERÊNCIAS BIBLIOGRÁFICAS

ANATEL: Anatel colabora com Receita Federal no programa Além do Horizonte — Agência Nacional de Telecomunicações. Disponível: www.gov.br, Acesso em: agosto 2023.

AMLOGIC et al. github.com/ophub/amlogic-s9xxx-armbian Acesso em julho de 2023.

BALAJI, S. et al., IoT Technology, Applications and Challenges: A Contemporary Survey, 2019.

BLAINSKI, É et al. Estações hidrometeorológicas automáticas: recomendações técnicas para instalação. Florianópolis: Epagri, 2012.

BRASIL, Receita Federal e Associação Brasileira de Televisão por Assinatura destroem mais de 97 mil aparelhos de tv box piratas no Rio de Janeiro. Disponível em: https://www.gov.br/receitafederal/pt-br/assuntos/noticias/2021/maio/receita-federal-e-associacao-brasileira-de-televisao-por-assinatura-destroem-mais-de-97-mil-aparelhos-de-tv-box-piratas-no-rio-de-janeiro. 2021. Acesso em: Agosto de 2023.

BURNS, B. Designing Distributed Systems. ed. O'Reilly Media, 2018. 162p. ISBN10: 1491983647.

COULORIS, G.; DOLLIMORE, J.; KINDBERG, T. Distributed Systems: Concepts and Design, ed.3. Addison Wesley, agosto de 2000.

FRANZ, N. M.; SILVA, C. L. Waste Electrical and Electronic Equipment (WEEE): global and contemporary challenge to production chains and the urban environment. Gestão & Produção, 29, e6621. 2022 doi.org/10.1590/1806-9649-2022v29e6621.

FORT, V. et al. The global e-waste monitor 2020 Bonn: United Nations University (UNU). 120p., 2021.

TANEMBAUM, A.; FEAMSTER, N.; WETHERALL, D. Redes de computadores. ed. Pearson Prentice Hall, 2011; 582p. ISBN: 857605924X.