FELIPE OLIVEIRA CHIARINI

APLICAÇÃO DE SERVIÇOS DE NUVEM NO MONITORAMENTO REMOTO DE SENSORES

LAVRAS - MG
2021

FELIPE OLIVEIRA CHIARINI

APLICAÇÃO DE SERVIÇOS DE NUVEM NO MONITORAMENTO REMOTO DE SENSORES

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação na modalidade de Concepção Básica, para a obtenção do título de Bacharel.

Prof. Dr. Arthur de Miranda Neto
Orientador

LAVRAS – MG
2021

FELIPE OLIVEIRA CHIARINI

APLICAÇÃO DE SERVIÇOS DE NUVEM NO MONITORAMENTO REMOTO DE SENSORES

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação na modalidade de Concepção Básica, para a obtenção do título de Bacharel.

APROVADO em 29 de novembro de 2021
Prof. Dr. Arthur de Miranda Neto - UFLA
Prof. Dr. Danton Diego Ferreira - UFLA
Prof. Dr. Leonardo Silveira Paiva - UFLA

Prof. Dr. Arthur de Miranda Neto
Orientador

LAVRAS – MG
2021

AGRADECIMENTOS

Agradeço a todos aqueles que auxiliaram de alguma forma no desenvolvimento deste trabalho.

RESUMO

Neste trabalho é apresentado um estudo da aplicação de serviços de nuvem no monitoramento remoto de sensores. Para isso, propõe-se a utilização de uma plataforma de nuvem sob demanda, *Amazon Web Services* (AWS), como principal ferramenta para a realização da transferência, armazenamento, manipulação e recuperação dos valores medidos. São apresentadas duas estruturas de fluxos de dados, utilizando-se para armazenamento dos dados o *Amazon DynamoDB* e o *Amazon Simple Storage Service* (*Amazon S3*), devido à redução de custos e benefícios que estes serviços podem oferecer. Para a execução dos testes, são utilizados o microcontrolador ESP8266 NodeMcu ESP-12 e o sensor de Umidade e Temperatura DHT11. Os dados dos sensores são enviados para a nuvem por meio do *AWS IoT Core*, o *DynamoDB* é utilizado para o armazenamento dos dados recentes, o *AWS Lambda* para a inserção dos dados de longo prazo no *Amazon Simple Storage Service*, tornando possível a exibição dessas informações para o usuário final através de uma aplicação *web*.

Palavras-chave: Monitoramento; Sensores; Engenharia; Automação; Internet das coisas.

ABSTRACT

This work presents a study of the application of using cloud services for remote monitoring of sensors. For this, it proposes the use of an on-demand cloud platform, Amazon Web Services (AWS), as the main tool for carrying out the transfer, storage, manipulation, and retrieval of measured values. Two data stream structures are presented, using Amazon DynamoDB, and Amazon Simple Storage Service (Amazon S3) for data storage, due to the reduction in costs and benefits that these services can offer. To carry out the tests, the ESP8266 NodeMcu ESP-12 microcontroller and the DHT11 Humidity and Temperature sensor are used. Sensor data is send to the cloud through AWS IoT Core, DynamoDB is used to store recent data, AWS Lambda is used to insertion of long-term data into the Amazon Simple Storage Service, making it possible to display this information to the end user through a web application.

Keywords: Monitoring; Sensors; Engineering; Automation; Internet of Things.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AWS	*Amazon Web Services*
CLP	Controlador Lógico Programável
HTTP	*Hypertext Transfer Protocol*
IAM	*Identity and Access Management*
IoT	*Internet of Things*
JSON	*JavaScript Object Notation*
RCU	*Unidade de capacidade de leitura*
S3	*Simple Storage Service*
TTL	*Time To Live*
URL	*Uniform Resource Locator*
WCU	Unidade de capacidade de gravação

SUMÁRIO

1 INTRODUÇÃO

1.1 Identificação do problema

Sensores são o meio de comunicação entre os mundos físico e digital. Em muitos casos, a partir da extração dos dados por um dispositivo, é necessária a transmissão para outro receptor, como *smartphones*, microcontroladores, Controladores Lógicos Programáveis (CLPs), dentre outros. Isso torna possíveis ações como, a visualização dos dados pelo usuário para a tomada de decisões ou o controle de processos industriais.

Quando um dispositivo precisa receber informações e não se encontra próximo à fonte dos dados, a transmissão torna-se um desafio, ao qual a *Internet of Things* (IoT) é o meio para a solução. A IoT possibilita a interligação entre dispositivos e usuários através da *internet*.

A IoT tem ganhado destaque, visto que, possibilita a conexão entre processos, pessoas e objetos do cotidiano; tanto no ambiente doméstico quanto industrial. Toda essa tecnologia é impulsionada por avanços na conectividade, sensores de baixo custo, algoritmos de inteligência artificial e, principalmente, plataformas de nuvem (ORACLE, c2021).

No entanto, o desenvolvimento desta tecnologia traz desafios: escalabilidade, grande complexidade de desenvolvimento e garantia da segurança da informação (KRANENBURG, 2012).

Por outro lado, os serviços de nuvem promovem uma estrutura robusta para aplicações de monitoramento remoto de sensores, de modo que, os impactos causados pelos problemas mencionados sejam reduzidos.

1.2 Objetivos

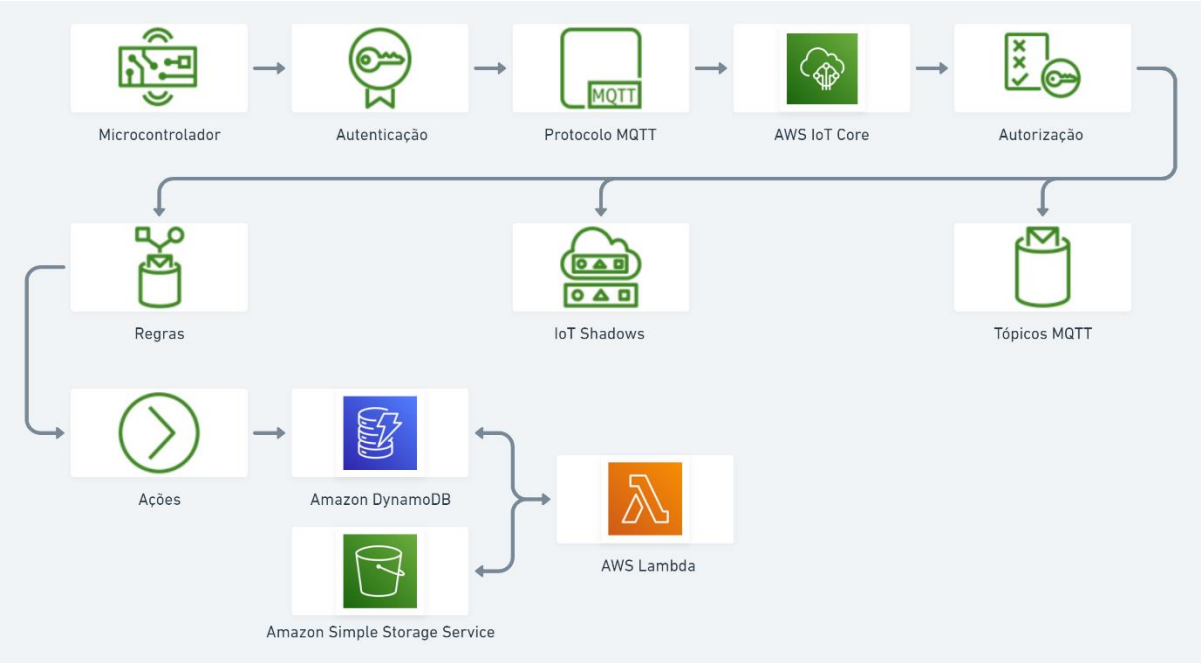
Este trabalho tem como objetivo principal, a análise da aplicação de serviços de nuvem no monitoramento remoto de sensores, através da compreensão das principais funcionalidades e suas precificações, além da comparação de duas alternativas para o fluxo de dados.

Como objetivo secundário, a implementação do monitoramento remoto de um sensor de temperatura, a partir de serviços de nuvem, para a construção dos fluxos de coleta, transmissão, manipulação, armazenamento e leitura dos dados.

2 FUNDAMENTAÇÃO

Este tópico aborda os princípios de funcionamento do protocolo de comunicação entre o microcontrolador e o serviço de nuvem, bem como as principais ferramentas voltadas para a aplicação de monitoramento remoto de sensores da Amazon Web Services (AWS). O fluxo de consumo dessas ferramentas é apresentado na Figura 1.

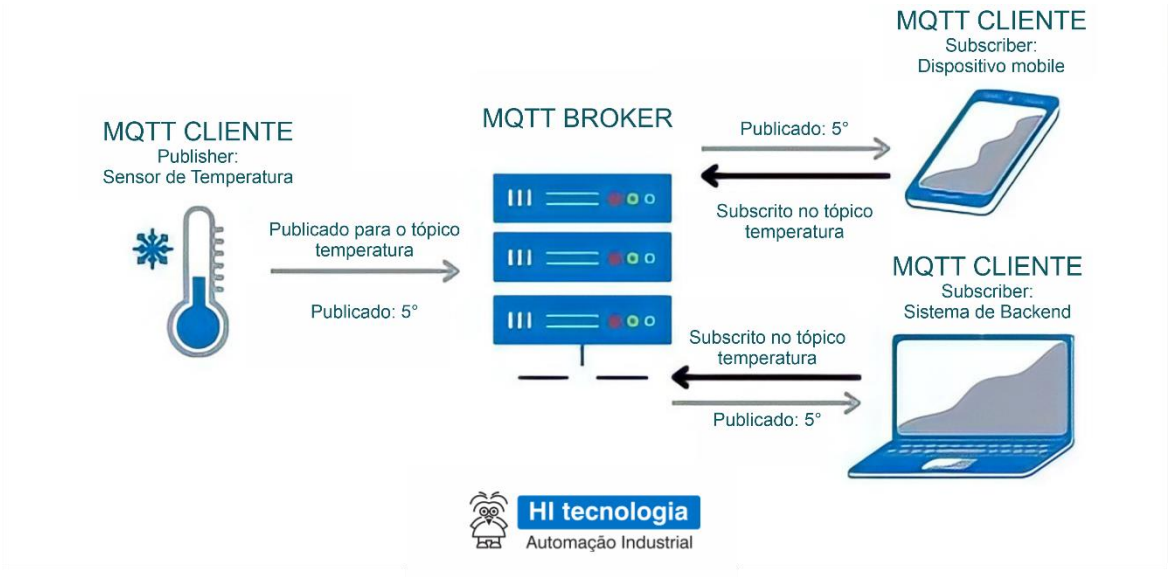
Figura 1 - Fluxo de consumo das ferramentas aplicadas ao monitoramento remoto de sensores.



2.1 *Message Queue Telemetry Transport* (MQTT)

Message Queue Telemetry Transport (MQTT) é um protocolo de comunicação entre dispositivos, projetado para exigir um baixo consumo de banda. O funcionamento do MQTT é baseado em tópicos, nos quais os *Publishers* inserem informações, que são posteriormente transmitidas a todos os *Subscribers* por meio de um broker. O fluxo é apresentado na Figura 2, em que um sensor de temperatura envia suas medições para dois clientes (AMAZON..., c2021a).

Figura 2 – Protocolo MQTT.



Fonte: Fernandes (2021).

2.2 Amazon Web Services

A AWS é uma plataforma de nuvem lançada oficialmente em 2006, voltada inicialmente para prover serviços *online* para sistemas cliente-servidor. Com o seu crescimento, foram acrescidas funcionalidades para o desenvolvimento de aplicações para a IoT, possibilitando a troca de informações entre diversos dispositivos em tempo real, armazenamento de informações e acesso a ferramentas para a manipulação e operação dos dados (AMAZON..., c2021b).

2.2.1 Autenticação

A autenticação refere-se à identificação do dispositivo. Este é um fator de extrema importância, pois, reconhece e diferencia um dispositivo do outro; tratando-o de forma única.

Para a realização desta função, o uso de certificados é extremamente útil. A AWS possibilita a geração de certificados X.509, um formato padrão para certificados de chave pública. Este certificado inclui uma chave pública, uma chave privada, assinatura digital, informações sobre a identidade associada ao certificado e sua emissão de autoridade de certificação (SSL..., c2021).

2.2.2 Conexão

A conexão da fonte de dados com a AWS é realizada pelo *AWS IoT Core*, com o protocolo de comunicação MQTT, o que possibilita o envio e recebimento de informações, além do consumo de ferramentas da AWS.

Através dos tópicos específicos do MQTT, os serviços da AWS são consumidos, logo, para acionar qualquer funcionalidade, é necessário um envio de mensagem, que tem um custo. É possível reduzir os custos através do recurso *basic ingest*, que otimiza o fluxo de dados removendo intermediários do processo (AMAZON..., c2021d).

2.2.3 Autorização

Uma vez que o usuário é autenticado, é importante um mecanismo de autorização para definir o que o usuário tem ou não permissão. Para isso, é utilizada a ferramenta *Identity and Access Management* (IAM). Ela permite definir políticas de acesso, especificando as operações que o usuário pode executar.

2.2.4 *Device Shadow*

O serviço *Device Shadow* é um recurso voltado para a IoT. Ele permite armazenar variáveis na nuvem referentes a um dispositivo. Este recurso pode ser utilizado, por exemplo, para acionar *leds* em uma interface com o usuário, armazenar a posição de um motor, dentre outras funções (AMAZON..., c2021e).

2.2.5 Regras e ações

As regras e ações permitem a interação do dispositivo com os serviços da AWS. Quando acionadas, as regras extraem os dados enviados e, se necessário, manipulam e acrescentam informações, o que desencadeia uma ou mais ações. Estas ações podem, por exemplo, inserir dados em uma tabela do *DynamoDB*, ativar uma função *Lambda*, publicar mensagens em um tópico MQTT, armazenar dados no *Amazon S3*, dentre outras opções.

2.2.6 *DynamoDB*

O *DynamoDB* é um banco de dados NoSQL de chave-valor totalmente gerenciado, com grande desempenho, segurança e escalabilidade. Recomendado

para aplicativos móveis, sites, jogos e principalmente para a IoT. Através das regras e ações, apresentadas no tópico 2.5.5, é possível enviar dados dos sensores para o *DynamoDB* (AMAZON..., c2021f).

O funcionamento do *DynamoDB* é baseado em três modos de capacidade. O modo de capacidade sob demanda, em que o consumo não é predefinido e é cobrado apenas pelo uso. O modo de capacidade provisionada, em que são determinadas as taxas de leitura e escrita por hora, com um custo fixo. O modo de capacidade reservada, em que as capacidades de leituras e escritas podem ser reservadas com uma antecedência de 1 a 3 anos.

Existem diferentes categorias de leituras no *DynamoDB*: leituras eventualmente consistentes, na qual a resposta pode não refletir os resultados de operações concluídas recentemente, porém com menor latência; leituras fortemente consistentes, em que os dados são retornados atualizados, mas podem ocorrer problemas de desempenho e custos mais altos (AMAZON..., c2021g).

Outra abordagem é a leitura e escrita transacional, que permite o agrupamento de várias ações para serem realizadas em uma única chamada, reduzindo custos. (AMAZON..., c2021h).

É possível realizar *backups* pontuais enviados para o *Amazon S3*, para isso é feita uma cópia dos dados do *DynamoDB* no momento da solicitação, possibilitando a recuperação dos dados em caso de necessidade da aplicação.

2.2.7 *Amazon S3*

O *Amazon S3* é um serviço de armazenamento de objetos. Esta é uma ferramenta útil para a IoT, pois, armazena uma grande quantidade de dados, nos formatos de arquivos, a um custo inferior (AMAZON..., c2021j).

Existem diferentes classes de armazenamento para o *Amazon S3.* Elas possibilitam, por exemplo, que itens que não são acessados com grande frequência sejam armazenados por um menor custo, mas com menos disponibilidade para o usuário. As classes existentes e suas descrições são apresentadas a seguir:

- *S3 Standard*: Armazenamento de uso geral, com grande uso para todos os tipos de dados acessados com frequência.

- *S3 Intelligent-Tiering*: É realizado o monitoramento dos padrões de acesso dos dados, gerando economias de custo automáticas para os dados. Porém, para dados menores que 128KB, esse gerenciamento não é aplicado e também não são cobradas as taxas de monitoramento e automação.
- *S3 Standard - Infrequent Access*: Armazenamento de dados com retenção de longa duração, com baixa frequência de uso e necessidade de rápido acesso.
- *S3 One Zone - Infrequent Access*: Armazenamento de dados recriáveis, com baixa frequência de uso e que demandam rápido acesso.
- *S3 Glacier*: Armazenamento de backups e arquivos com retenção de longo prazo, com recuperação entre 1 minuto e 12 horas.
- *S3 Glacier Deep Archive*: Armazenamento de backups e arquivos com retenção de longo prazo, com baixo uso e recuperação em até 12 horas.

2.2.8 *AWS Lambda*

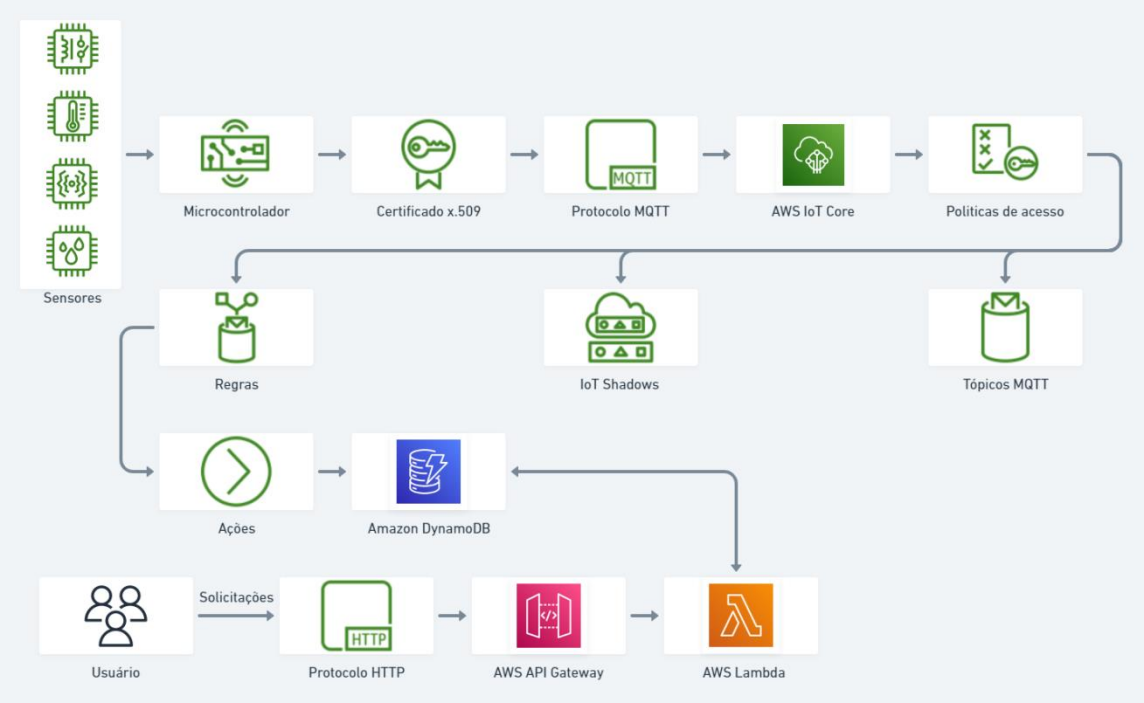
O *AWS Lambda* é um serviço de computação em nuvem sem servidor. Ele permite a execução de códigos em diversas linguagens, sem a necessidade do gerenciamento de servidores. Sua execução é baseada em solicitações ou eventos de entrada, como um novo arquivo no *Amazon S3* ou uma ação do *AWS IoT Core* (AMAZON..., c2021l). A conexão do usuário com uma função Lambda é feita através da ferramenta *API Gateway.* Também é possível a execução agendada de funções através do *AWS CloudWatch.*

3 SOLUÇÕES PROPOSTAS

Baseado nas funcionalidades da AWS apresentadas no tópico 2.2 deste trabalho, a seguir, duas alternativas para o uso de serviços de nuvem são aplicadas no monitoramento remoto de sensores. Assim, são detalhados os fluxos dos processos de coleta, transmissão, manipulação, armazenamento e leitura dos dados.

3.1 Alternativa 1: *DynamoDB* para armazenamento

Figura 3 – Fluxograma da alternativa 1.



Fonte: Chiarini (2021).

A Figura 3 apresenta o fluxo da alternativa 1. O processo é iniciado com a coleta dos dados através de sensores pelo microcontrolador, que contém um certificado x.509. Assim, é feita a conexão do dispositivo com o AWS IoT Core através do protocolo MQTT.

Com isso, o dispositivo autorizado envia os dados coletados para uma regra, que acrescenta duas informações: usuário e *timestamp,* que se refere ao momento em que a medição é registrada. Em seguida, é disparada uma ação, que armazena os valores no *DynamoDB*.

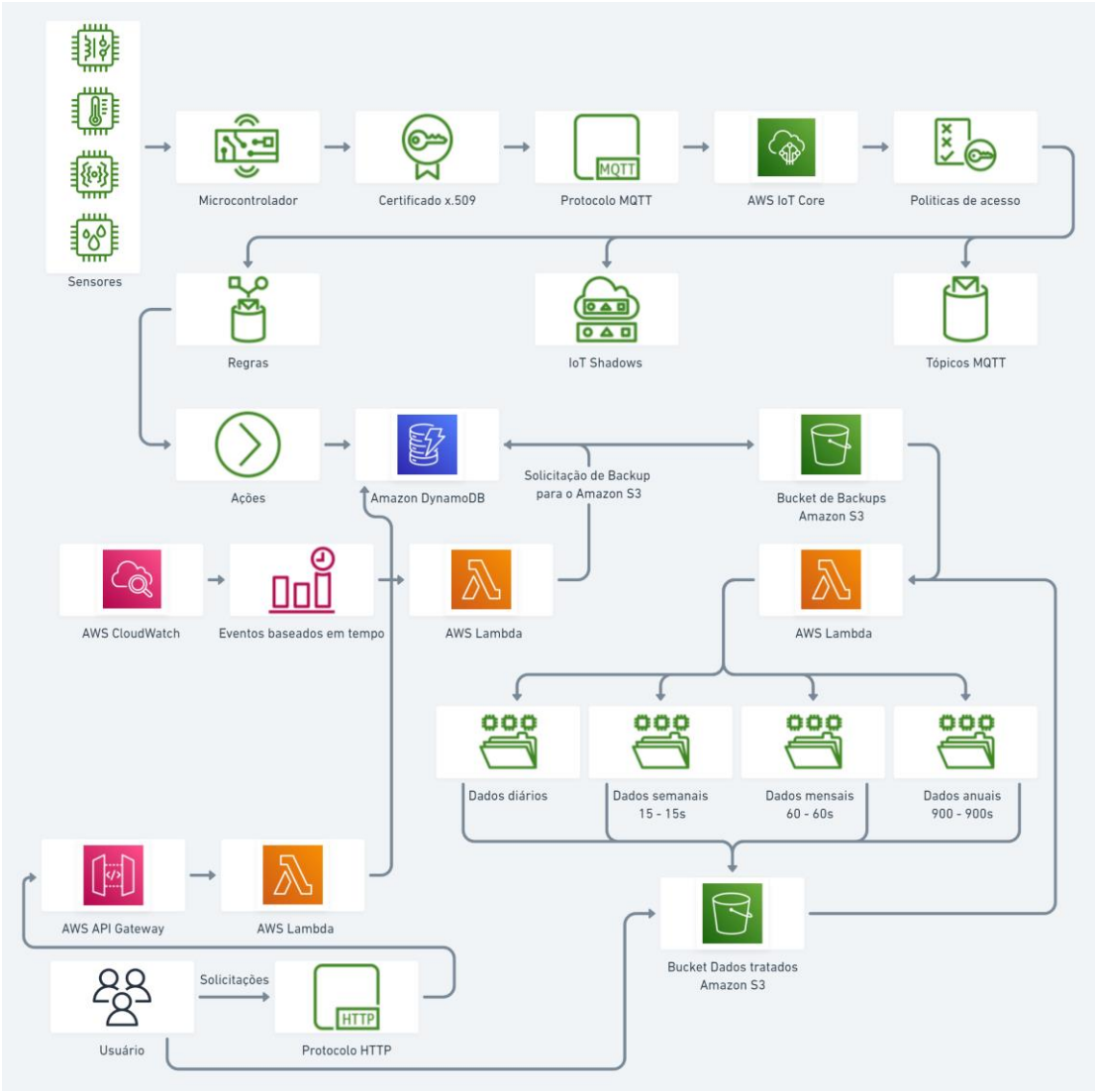
Com os dados armazenados, no momento em que um usuário realiza uma solicitação de leitura, as informações da solicitação são enviadas por meio do

Hypertext Transfer Protocol (HTTP) para o *AWS API Gateway*, que aciona uma função *Lambda* responsável pela leitura dos dados no *DynamoDB*.

Existem duas formas de leitura, a de dados completos, que retorna todas as medidas, e de dados resumidos. A função *Lambda* consulta todos os dados do período solicitado, realiza a compactação, retorna para o usuário, e armazena em uma tabela auxiliar. Deste modo, quando ocorre uma solicitação das mesmas informações, não é necessário a execução de todos os processos.

3.2 Alternativa 2: DynamoDB e Amazon S3 para armazenamento

Figura 4 – Fluxograma da alternativa 2.



Fonte: Chiarini (2021).

As operações realizadas pela alternativa 2 são apresentadas na Figura 4. Para esta alternativa, as operações iniciais são iguais às da alternativa 1. Mudanças ocorrem a partir do momento em que os dados são armazenados no *DynamoDB*. Em razão disso, é incluído o atributo *Time To Live* (TTL), no qual provoca a exclusão automática dos dados após uma hora.

Paralelamente, de hora em hora, antes da exclusão dos itens, é acionada uma função Lambda pelo *CloudWatch,* que solicita um *backup* pontual do *DynamoDB*, armazenando os dados em um *bucket* do *Amazon S3*.

No momento que os dados são inseridos no *bucket,* uma função *Lambda* é acionada, transformando os dados da hora anterior em 4 conjuntos: dados completos diários, e outros três, semanais, mensais e anuais, os quais são feitas as médias dos dados em conjuntos de tempo de 15 segundos, 60 segundos e 900 segundos respectivamente. Essas informações são armazenadas em um *bucket* para dados tratados no *Amazon S3*, no formato *JavaScript Object Notation* (JSON).

Para ter acesso aos dados, o usuário realiza uma solicitação de leitura pelo protocolo HTTP para o AWS API Gateway, que aciona uma função *Lambda*. As solicitações de acesso aos dados da última hora são consultadas no *DynamoDB.* Para os dados históricos, o *Lambda* gera o *Uniform Resource Locator* (URL) de acesso temporário, utilizado para a leitura dos dados diretamente no *Amazon S3*.

4 LEVANTAMENTO DOS CUSTOS

Para a análise dos aspectos econômicos da aplicação de monitoramento remoto, utilizando serviços de nuvem, são estimados os custos mensais, 31 dias, em um cenário onde um microcontrolador, conectado sempre ao sistema de rede, envia informações de um sensor a cada segundo, com apenas uma única medida numérica.

Para o cenário analisado, foram consideradas 744 horas de conexão, totalizando 2.678.400 envios, sendo que cada dado é composto de 50 bytes, salvos no *DynamoDB*. Após um processo de compactação de dados, o novo tamanho associado é de 20 bytes, salvos no *Amazon S3*.

4.1 Custos da *Amazon Web Services*

Na AWS, a definição dos custos tem grande relação com a região utilizada para o projeto. As informações de custos apresentadas a seguir levam em consideração a região leste dos Estados Unidos da América (Norte da Virgínia). A escolha da região influência nos resultados, pois, impacta diretamente na latência de conexão do usuário com a ferramenta.

4.1.1 Conexão

O custo da conectividade é de 0,08 USD por milhão de minutos de conexão. Assim, para um dispositivo ativo durante 365 dias, 24 horas por dia, o custo total aproximado é de 0,042 USD (AMAZON..., c2021c).

Além disso, o custo de envio de mensagens segue a definição:

-Até 1 bilhão de mensagens: 1,00 USD (por milhão de mensagens).

-Próximos 4 bilhões de mensagens: 0,80 USD (por milhão de mensagens).

-Mais de 5 bilhões de mensagens: 0,70 USD (por milhão de mensagens).

4.1.2 Device Shadow

O recurso *Device Shadow* tem o custo de 1,25 USD por milhão de operações, onde uma operação é medida em incrementos de 1 KB. Logo, uma atualização de um registro de 1,5 KB é medida como duas operações (AMAZON..., c2021c).

4.1.3 Regras e ações

Os custos para os serviços de regras e ações são de 0,15 USD por milhão de acionamentos. Com isso, o envio de uma informação recebida em um tópico MQTT, para o DynamoDB, através do acionamento de uma regra e uma ação, tem o custo unitário de 0,0000003 USD. (AMAZON..., c2021c).

4.1.4 *DynamoDB*

O custo na modalidade sob demanda do DynamoDB é de 1,25 USD por milhão de unidades de solicitação de gravação, sendo que uma unidade comporta até 1 KB. Para as leituras sob demanda o custo é de 0,25 USD por milhão de unidades de solicitação de leitura, comportando até 4 KB por unidade (AMAZON..., c2021q).

A capacidade provisionada é cobrada com base no custo por hora do uso das unidades de capacidade de gravação (WCU) e unidades de capacidade de leitura (RCU). Para a capacidade reservada é cobrada uma taxa adiantada para a compra de blocos de 100 WCUs ou 100 RCUs. Ambas as precificações são apresentadas nas Tabelas 1 e 2.

Tabela 1 – Custos da capacidade provisionada.

Tipo de transferência provisionada	Custo por hora
Unidade de capacidade de gravação (WCU)	0,00065 USD por WCU
Unidade de capacidade de leitura (RCU)	0,00013 USD por RCU

Fonte: AMAZON... (c2021i).

Tabela 2 – Custos da capacidade reservada.

Compromisso mensal	Antecipado: 1 ano	Por hora: 1 ano	Antecipado: 3 anos	Por hora: 3 anos
100 unidades de capacidade de gravação	150,00 USD	0,0128 USD	180,00 USD	0,0081 USD
100 unidades de capacidade de leitura	30,00 USD	0,0025 USD	36,00 USD	0,0016 USD

Fonte: AMAZON... (c2021i).

Para o armazenamento, os primeiros 25 GB armazenados são gratuitos e, a partir de então, é cobrado 0,25 USD por GB por mês para os GB seguintes. A

exportação programada para o *Amazon S3* tem o custo de 0,1 USD por GB exportado (AMAZON..., c2021q).

4.1.5 Amazon S3

O *Amazon S3* tem diferentes custos para cada classe de armazenamento, eles são apresentados na Tabela 3.

Tabela 3 – Custos de armazenamento no *Amazon S3.*

	Custo do armazenamento
S3 Standard	
Primeiros 50 TB/mês	0,023 USD por GB
Próximos 450 TB/mês	0,022 USD por GB
Mais de 500 TB/mês	0,021 USD por GB
S3 Intelligent-Tiering	
Camada de acesso frequente, primeiros 50 TB/mês	0,023 USD por GB
Camada de acesso frequente, próximos 450 TB/mês	0,022 USD por GB
Camada de acesso frequente, mais de 500 TB/mês	0,021 USD por GB
Camada de acesso infrequente, todo o armazenamento/mês	0,0125 USD por GB
Nível de acesso de arquivamento, todo o armazenamento/mês	0,004 USD por GB
Nível de acesso de arquivamento profundo, todo o armazenamento/mês	0,00099 USD por GB
Monitoramento e automação, todo o armazenamento/mês (objetos > 128 KB)	0,0025 USD por 1.000 objetos
S3 Standard - Infrequent Access	
Todo o armazenamento/mês	0,0125 USD por GB
S3 One Zone - Infrequent Access	
Todo o armazenamento/mês	0,01 USD por GB
S3 Glacier	
Todo o armazenamento/mês	0,004 USD por GB
S3 Glacier Deep Archive	
Todo o armazenamento/mês	0,00099 USD por GB

Fonte: AMAZON... (c2021k).

As operações realizadas no *Amazon S3* para armazenar e acessar dados variam conforme a classe, com exceção das solicitações de *LIST*, que seguem os

mesmos custos do *S3 Standard*. Além disso, os itens armazenados no *S3 Standard - Infrequent Access, S3 One Zone - Infrequent Access, S3 Glacier e S3 Glacier Deep Archive,* necessitam da operação de recuperação para possibilitar as demais solicitações apresentadas na Tabela 4.

Tabela 4 – Custo das solicitações e recuperações de dados no Amazon S3

	Solicitações *PUT, COPY, POST, LIST* (por 1.000 solicitações)	Solicitações *GET, SELECT* e todas as outras (por 1.000 solicitações)	Solicitações de transição de ciclo de vida para (por 1.000 solicitações)	Solicitações de recuperação de dados (por 1.000 solicitações)	Recuperações de dados (por GB)
S3 Standard	0,005 USD	0,0004 USD	-	-	-
S3 Intelligent Tiering	0,005 USD	0,0004 USD	0,01 USD	-	-
Acesso de arquivamento, padrão	-	-	-	-	-
Acesso de arquivamento, em massa	-	-	-	-	-
Acesso de arquivamento, expresso	-	-	-	10,00 USD	0,03 USD
Acesso de arquivamento profundo, padrão	-	-	-	-	-
Acesso de arquivamento profundo, em massa	-	-	-	-	-
S3 Standard – Infrequent Access	0,01 USD	0,001 USD	0,01 USD	-	0,01 USD
S3 One Zone – Infrequent Access	0,01 USD	0,001 USD	0,01 USD	-	0,01 USD
S3 Glacier	0,03 USD	0,0004 USD	0,03 USD	-	-
Express	-	-	-	10,00 USD	0,03 USD
Standard	-	-	-	0,05 USD	0,01 USD
Em massa	-	-	-	0,025 USD	0,0025 USD
Unidades de capacidade provisionadas	-	-	-	-	100,00 USD por unidade
S3 Glacier Deep Archive	0,05 USD	0,0004 USD	0,05 USD	-	-
Standard	-	-	-	0,10 USD	0,02 USD
Em massa	-	-	-	0,025 USD	0,0025 USD

Fonte: AMAZON... (c2021k).

4.1.6 AWS Lambda

Os custos dos serviços relacionados ao *AWS Lambda* são: para o *API Gateway,* 1,00 USD por milhão de requisições, até as primeiras 300 milhões de requisições. A partir disso, 0,90 USD por milhão de requisições (AMAZON..., c2021m); para *o Amazon CloudWatch*, o custo é de 1,00 USD por milhão de eventos (AMAZON..., c2021n).

O *AWS Lambda* tem o custo baseado no número de solicitações, na duração da execução e nos recursos alocados. Assim, o custo é de 0,20 USD por 1 milhão de solicitações. A Tabela 5 é apresenta o custo por milissegundo de execução.

Tabela 5 – Custo do *AWS Lambda*

Memória (MB)	Custo por 1 milissegundo
128	0,0000000021 USD
512	0,0000000083 USD
1.024	0,0000000167 USD
1536	0,0000000250 USD
2048	0,0000000333 USD
3072	0,0000000500 USD
4096	0,0000000667 USD
5120	0,0000000833 USD
6144	0,0000001000 USD
7168	0,0000001167 USD
8192	0,0000001333 USD
9216	0,0000001500 USD
10.240	0,0000001667 USD

Fonte: AMAZON... (c2021o).

4.1.7 Transferência de dados para ambientes externos

A AWS cobra pela largura de banda transferida para ambientes externos pelos serviços como *Amazon S3, DynamoDB*, Funções *Lambdas*, dentre outros. Não há cobrança para dados transferidos entre funcionalidades na mesma região. Os custos da transferência de dados para ambientes externos são apresentados na Tabela 6.

Tabela 6 – Custo da transferência de dados para ambientes externos.

Transferência de dados	Definição de custo
Todas as transferências de dados para dentro	0,00 USD por GB
Transferência de dados para fora	
Até 1 GB/mês	0,00 USD por GB
Próximos 10 TB/mês	0,09 USD por GB
Próximos 40 TB/mês	0,085 USD por GB
Próximos 100 TB/mês	0,07 USD por GB
Maior que 150 TB/mês	0,05 USD por GB

Fonte: AMAZON... (c2021p).

4.2 Custos da alternativa 1: *DynamoDB* para armazenamento

O custo para a alternativa 1 está agrupado em 4 principais pontos: estabelecimento da conexão, envio dos dados, armazenamento e leitura da informação.

4.2.1 Estabelecimento da conexão

A conexão contínua do dispositivo com o *AWS IoT Core,* através do protocolo MQTT, tem seu custo por hora de uso, ele é apresentado na Tabela 7.

Tabela 7 – Custo de conexão com o *AWS IoT Core.*

Total de horas de conexão mensal	744 horas
Custo por milhão de minutos	0,08 USD
Custo total	0,0035712 USD

Fonte: Chiarini (2021).

4.2.2 Envio dos dados

Com o dispositivo conectado, os dados são enviados para um tópico MQTT, que através das regras e ações são armazenados no DynamoDB. Para redução dos custos, pode-se utilizar o recurso *basic ingest,* eliminando os custos de envio de

mensagens para o consumo de funcionalidades. O custo total do envio dos dados é apresentado na Tabela 8.

Tabela 8 – Custo do envio dos dados.

	Sem o *basic ingest*	Com o *basic ingest*
Total de mensagens	2.678.400	2.678.400
Custo por milhão de mensagens	1 USD	0 USD
Custo por milhão de regras acionadas	0,15 USD	0,15 USD
Custo por milhão de ações acionadas	0,15 USD	0,15 USD
Custo total	3,48192 USD	0,80352 USD

Fonte: Chiarini (2021).

Logo, com a utilização do recurso *basic ingest* para o consumo das funcionalidades, o custo total é reduzido em 2,68 USD.

4.2.3 Armazenamento da informação

Todos os dados são armazenados no *DynamoDB* com 3 atributos: ID do usuário, *timestamp* e valor da medição. Para a capacidade provisionada é considerada 1 WCU e para a capacidade reservada é aproximado o custo de compra de apenas uma WCU no bloco de 100 unidades. Com isso, são apresentados na Tabela 9 os custos para as inserções dos dados, e nas Tabelas 10 e 11, os custos de armazenamento.

Tabela 9 – Custos de inserção dos dados no *DynamoDB*.

	Capacidade sob demanda	Capacidade provisionada	Capacidade reservada 1 ano	Capacidade reservada 3 anos
Custo de gravação por milhão	1,25 USD	0 USD	0 USD	0 USD
Custo por hora	0 USD	0,00065 USD	0,000128 USD	0,000081 USD
Custo total mensal	3,348 USD	0,4836 USD	0,095232 USD	0,060264 USD

Fonte: Chiarini (2021).

Tabela 10 – Custo de armazenamento dos dados completos no *DynamoDB*.

Mês	Armazenamento médio mensal (GB)	Custo mensal (USD)	Mês	Armazenamento médio mensal (GB)	Custo mensal (USD)
1	0,06236136	0,01559034	13	0,810697675	0,202674419
2	0,124722719	0,03118068	14	0,873059034	0,218264759
3	0,187084079	0,04677102	15	0,935420394	0,233855098
4	0,249445438	0,06236136	16	0,997781754	0,249445438
5	0,311806798	0,077951699	17	1,060143113	0,265035778
6	0,374168158	0,093542039	18	1,122504473	0,280626118
7	0,436529517	0,109132379	19	1,184865832	0,296216458
8	0,498890877	0,124722719	20	1,247227192	0,311806798
9	0,561252236	0,140313059	21	1,309588552	0,327397138
10	0,623613596	0,155903399	22	1,371949911	0,342987478
11	0,685974956	0,171493739	23	1,434311271	0,358577818
12	0,748336315	0,187084079	24	1,49667263	0,374168158

Fonte: Chiarini (2021).

Tabela 11 – Custo máximo do armazenamento dos dados resumidos no *DynamoDB*.

Mês	Armazenamento médio mensal (GB)	Custo mensal (USD)	Mês	Armazenamento médio mensal (GB)	Custo mensal (USD)
1	0,005263299	0,001315825	13	0,068422884	0,017105721
2	0,010526597	0,002631649	14	0,073686182	0,018421546
3	0,015789896	0,003947474	15	0,078949481	0,01973737
4	0,021053195	0,005263299	16	0,08421278	0,021053195
5	0,026316494	0,006579123	17	0,089476079	0,02236902
6	0,031579792	0,007894948	18	0,094739377	0,023684844
7	0,036843091	0,009210773	19	0,100002676	0,025000669
8	0,04210639	0,010526597	20	0,105265975	0,026316494
9	0,047369689	0,011842422	21	0,110529274	0,027632318
10	0,052632987	0,013158247	22	0,115792572	0,028948143
11	0,057896286	0,014474072	23	0,121055871	0,030263968
12	0,063159585	0,015789896	24	0,12631917	0,031579792

Fonte: Chiarini (2021).

4.2.4 Leitura das informações

Considerando que a frequência de leitura de dados pode variar de acordo com as necessidades de cada projeto, o custo de leitura é medido conforme a demanda, conforme apresentado na Tabela 12.

Tabela 12 – Custo por leitura no *DynamoDB*.

Intervalo de dados	Peso estimado (MB)	Custo de leitura sob demanda (USD)	*API Gateway* (USD)	*AWS Lambda* (USD)	Transferência de dados (USD)	Custo total por consulta (USD)
1 hora	0,1716613	0,0000055	0,000001	0,0000027	0,0000151	0,0000206
1 dia	4,1198730	0,0001318	0,000001	0,0000052	0,0003620	0,0004940
7 dias	28,839111	0,0009228	0,000001	0,0000168	0,0025347	0,0034575
1 mês	127,71606	0,0040869	0,000001	0,0001247	0,0112250	0,0153120
1 ano	1503,7536	0,0481201	0,000001	0,0049952	0,1321659	0,1802860

Fonte: Chiarini (2021).

4.3 Custos da alternativa 2*: DynamoDB e Amazon S3* para armazenamento

A alternativa 2 possui os mesmos processos iniciais descritos para a alternativa 1, logo, o estabelecimento da conexão e a transferência de dados apresentam o mesmo custo.

4.3.1 Armazenamento da informação

O *DynamoDB* é responsável por armazenar os dados da última hora, para isso, todos os dados são inseridos com o atributo TTL, sendo removido pelo *DynamoDB* automaticamente conforme a especificação, não incorrendo custos de remoção.

O custo de armazenamento do *DynamoDB* é fixo, pois, em média, haverá sempre 60 minutos de dados armazenados. Com isso, o custo de armazenamento é de aproximadamente 0,0000419 USD.

Para o envio dos dados históricos para o *Amazon S3*, de hora em hora, é acionado uma função *Lambda,* que solicita um *backup* pontual do *DynamoDB* para o Amazon S3, gerando um custo de 0,1 USD por GB de dados.

O custo das funções *Lambdas* é baseado no tempo de execução de 300ms e 512 MB de memória. Além disso, a classe de armazenamento do *Amazon S3* tem relação com as especificações do projeto. Para as análises apresentadas neste

documento, foi considerado o *S3 Standard*. A partir de um processo de compactação, os arquivos passam a conter aproximadamente 20 bytes por item.

Os custos referentes ao armazenamento dos dados são apresentados nas Tabelas 13 e 14.

Tabela 13 – Custo do fluxo de armazenamento dos dados.

Processo	Custo
Ativações do *CloudWatch*	0,000744 USD
Função *Lambda* de solicitação *backup*	0,00200136 USD
Backup do *DynamoDB*	0,012472272 USD
Função *Lambdas* conversão dos dados	0,00200136 USD
Ações de leitura *Amazon S3*	0,001488 USD
Ações de delete *Amazon S3*	0,001488 USD
Ações de gravação *Amazon S3*	0,0186 USD
Total	0,038794992 USD

Fonte: Chiarini (2021).

Tabela 14 – Custo do armazenamento no *Amazon S3*.

Mês	Armazenamento (GB)	Custo mensal (USD)	Mês	Armazenamento (GB)	Custo mensal (USD)
1	0,02705045	0,00062216	13	0,351655856	0,008088085
2	0,054100901	0,001244321	14	0,378706306	0,008710245
3	0,081151351	0,001866481	15	0,405756757	0,009332405
4	0,108201802	0,002488641	16	0,432807207	0,009954566
5	0,135252252	0,003110802	17	0,459857658	0,010576726
6	0,162302703	0,003732962	18	0,486908108	0,011198886
7	0,189353153	0,004355123	19	0,513958558	0,011821047
8	0,216403604	0,004977283	20	0,541009009	0,012443207
9	0,243454054	0,005599443	21	0,568059459	0,013065368
10	0,270504504	0,006221604	22	0,59510991	0,013687528
11	0,297554955	0,006843764	23	0,62216036	0,014309688
12	0,324605405	0,007465924	24	0,649210811	0,014931849

Fonte: Chiarini (2021).

4.3.2 Leitura das informações

A leitura dos dados tem uma abordagem diferente da alternativa 1. Neste caso, o usuário realiza a leitura dos últimos dados diretamente do *DynamoDB*, incorrendo nos mesmos custos do modelo anterior, 0,0000205806 USD por leitura. Para dados

diários, semanais, mensais e anuais, é feita a leitura dos arquivos JSONs armazenados no *Amazon S3*.

Para realizar essa leitura, o usuário obtém uma URL de acesso por meio de uma função *Lambda*, cujo custo é apresentado na Tabela 15. A leitura dos dados completos ocorre pela consulta de vários arquivos diários no *Amazon S3*, incorrendo nos custos apresentados na Tabela 16.

Tabela 15 – Custo por leitura no *Amazon S3*: dados granulares.

Intervalo de dados granulares	*API Gateway* (USD)	*AWS Lambda* (USD)	Peso estimado (GB)	Solicitação de leitura (USD)	Transferência de dados (USD)	Custo total por consulta (USD)
1 dia	0,000001	0,0000027	0,0016093	0,0000004	0,0001448	0,0001489
7 dias (15 - 15s)	0,000001	0,0000027	0,0007510	0,0000004	6,759E-05	7,168E-05
1 mês (60 - 60s)	0,000001	0,0000027	0,0008315	0,0000004	7,483E-05	7,892E-05
1 ano (900 - 900s)	0,000001	0,0000027	0,0006527	0,0000004	5,874E-05	6,283E-05

Fonte: Chiarini (2021).

Tabela 16 – Custo por leitura no *Amazon S3*: dados completos.

Intervalo de dados completos	*API Gateway* (USD)	*AWS Lambda* (USD)	Peso estimado (GB)	Solicitação de leitura (USD)	Transferência de dados (USD)	Custo total por consulta (USD)
1 dia	0,000001	0,0000027	0,0016093	0,0000004	0,0001448	0,0001489
7 dias	0,000001	0,0000027	0,0112653	0,0000028	0,0010139	0,0010204
1 mês	0,000001	0,0000027	0,0498891	0,0000124	0,0044900	0,0045061
1 ano	0,000001	0,0000027	0,5874038	0,000146	0,0528663	0,0530160

Fonte: Chiarini (2021).

5 LEVANTAMENTO DOS BENEFÍCIOS

As alternativas apresentadas no Capítulo 3, em razão do uso de serviços de nuvem, oferecem vantagens ao desenvolvimento e escalabilidade do projeto. Independentemente do número de dispositivos e usuários, conforme acontece o aumento da demanda, toda a infraestrutura necessária para a sustentação das funcionalidades é escalada automaticamente, evitando problemas de indisponibilidade. Com isso, seu custo ocorre apenas pelos serviços consumidos, não sendo efetuado nenhum pagamento pelas estruturas não utilizadas.

5.1 Alternativa 1: *DynamoDB* para armazenamento

O armazenamento dos dados no *DynamoDB* oferece uma baixa latência de leitura e escrita, oferecendo a possibilidade de leitura, edição e remoção de dados em pontos específicos. Isso possibilita, por exemplo, a leitura de uma única medição, incorrendo em custos mais baixos.

5.2 Alternativa 2: *DynamoDB* e *Amazon S3* para armazenamento

Na alternativa 2, uma vez que dados históricos são armazenados no *Amazon S3,* há um menor custo de armazenamento em comparação ao *DynamoDB na* alternativa 1. Além disso, para reduzir ainda mais os custos, é possível utilizar outras classes de armazenamento do *Amazon S3,* de acordo com os requisitos do projeto.

Durante a leitura dos dados, com exceção do custo de transferência de dados, que é o mesmo para o *DynamoDB* e o *Amazon S3*, o custo para a leitura do *Amazon S3* é fixo, enquanto no *DynamoDB* varia de acordo com o tamanho da consulta. Logo, para a leitura de grandes volumes de dados, como é o caso comum em aplicações de monitoramento de sensores, o *Amazon S3* tem um custo inferior.

6 ANÁLISE ECONÔMICA

A partir das alternativas propostas, a seguir seus custos são apresentados. A Tabela 17 apresenta os custos de conexão, envio, gravação e armazenamento dos dados.

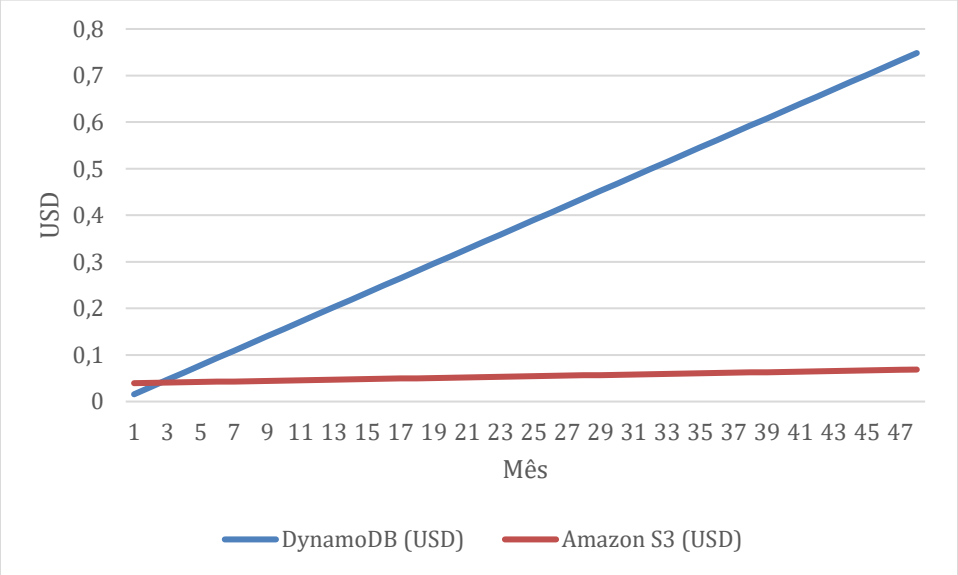
Tabela 17 – Custo da conexão, transferência e armazenamento das informações de um dispositivo, considerando o envio de dados a cada segundo.

		Alternativa 1 (USD)	Alternativa 2 (USD)
Conexão		0,00005952	0,00005952
Envio das informações		0,80352	0,80352
Gravação no DynamoDB		0,4836	0,4836
Armazenamento	DynamoDB	0,01559034	0,0000419
	Transferência e conversão dos dados	0	0,038794992
	Amazon S3	0	0,00062216
Total		1,30628154	1,330150252

Fonte: Chiarini (2021).

Na Figura 5, são apresentadas as curvas de custo de armazenamento do *DynamoDB* e do *Amazon S3*. Além disso, destaca-se que com a adição de variáveis ou o aumento da frequência de coleta, a diferença de custo entre os dois serviços de armazenamento aumenta.

Figura 5 – Custo de armazenamento no *DynamoDB* e no *Amazon S3* mensal.



Fonte: Chiarini (2021).

A partir dos custos de leitura dos dados apresentados nas Tabelas 18 e 19, verifica-se que, o Amazon S3 tem uma diferença de custo de leitura de aproximadamente 70% em relação ao *DynamoDB* sob demanda. Porém, o *DynamoDB* permite a leitura, edição e remoção de dados em pontos específicos, incorrendo em um baixo custo.

O *DynamoDB* pode também ter uma redução dos custos, através do uso da capacidade provisionada ou reservada.

Tabela 18 – Custo por leitura de dados completos.

	Alternativa 1 (USD)	Alternativa 2 (USD)
Diário	0,000493934	0,000148929
Semanal	0,003457539	0,001020365
Mensal	0,015311959	0,004506108
Anual	0,180285966	0,05301603

Fonte: Chiarini (2021).

Tabela 19 – Custo por leitura de dado resumido.

	Alternativa 1 (USD)	Alternativa 2 (USD)
Diário	0,000500114	0,000148929
Semanal	0,000236683	0,0000716817
Mensal	0,000261379	0,0000789236
Anual	0,000206498	0,0000628304

Fonte: Chiarini (2021).

Para obter os custos de leitura, é considerado a estimativa apresentada na Tabela 20.

Tabela 20 – Leituras estimadas durante o período de um mês.

	Completos	Resumidos
Última hora	200	0
Diário	120	0
Semanal	15	50
Mensal	5	25
Anual	2	10

Fonte: Chiarini (2021).

Com isso, o custo final de leitura para a alternativa 1 é de 0,57 USD, totalizando um custo final de 1,88 USD para o uso do sistema no primeiro mês, enquanto que para a alternativa 2, é de 0,17 USD, com o custo final de 1,50 USD.

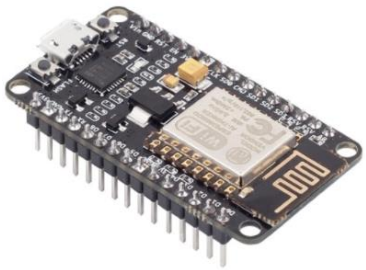
7 IMPLEMENTAÇÃO DO SISTEMA

O projeto de implementação apresentado a seguir é referente a alternativa 2. A escolha pela alternativa 2 leva em consideração a sua maior complexidade, além dela oferecer um menor custo de operação.

7.1 Extração das informações

A primeira etapa se inicia na coleta dos dados. Para isso, foi utilizado o microcontrolador ESP8266 NodeMcu ESP-12, exibido na Figura 6. Ele possui interface de conexão com redes wireless, possibilitando a troca de informações com serviços de nuvem. É uma solução de baixo custo, possui diferentes portas de conexão para sensores e demais componentes.

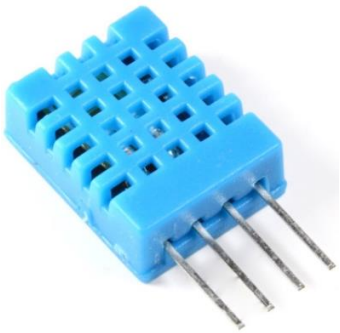
Figura 6 – Módulo WiFi ESP8266 NodeMcu ESP-12.



Fonte: FilipeFlop (2021a).

O sensor DHT11 utilizado é apresentado na Figura 7.

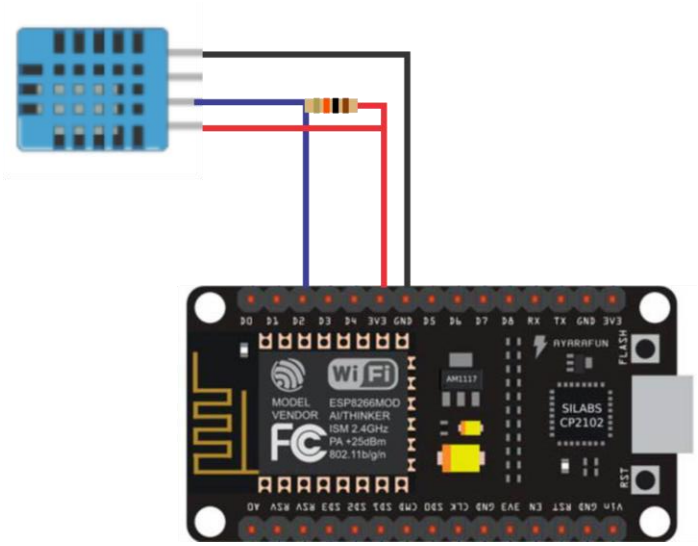
Figura 7 – Sensor de Umidade e Temperatura DHT11.



Fonte: FilipeFlop (2021b).

A Figura 8 apresenta o circuito necessário para a leitura da temperatura pelo microcontrolador com o sensor DHT11.

Figura 8 – Circuito elétrico.



Fonte: Fonte: Chiarini (2021).

7.2 Configurações do dispositivo

Com o dispositivo configurado no *AWS IoT Core*, é criado uma *shadow* para armazenar informações. Para atender os requisitos de segurança, são configuradas políticas de nível de acesso ao dispositivo. Essas configurações são apresentadas nas Figuras 9, 10 e 11.

Figura 9 – Política de conexão do dispositivo.

```
{
    "Effect": "Allow",
    "Action": "iot:Connect",
    "Resource": "*ARN*:client/${iot:Connection.Thing.ThingName}"
}
```

Fonte: Chiarini (2021).

Figura 10 – Política de acionamento da regra.

```
{
    "Effect": "Allow",
    "Action": "iot:Publish",
    "Resource": "*ARN*:topic/$aws/rules/EspToDynamoDB"
}
```

Fonte: Chiarini (2021).

Figura 11 – Política de interações com os *shadows* do dispositivo.

```
{
    "Effect": "Allow",
    "Action": "iot:*",
    "Resource":
"*ARN*:topic/$aws/things/${iot:Connection.Thing.ThingName}/shadow/*"
}
```

Fonte: Chiarini (2021).

7.3 Inserção dos dados no DynamoDB

No *DynamoDB,* é criada uma tabela para armazenar os dados, com a chave de partição referente ao dispositivo que contém a informação e a chave de classificação o *timestamp*.

Para a comunicação do dispositivo com o *DynamoDB*, é criada uma regra no *AWS IoT Core*, apresentada na Figura 12, que acrescenta o nome do dispositivo, o *timestamp* e o atributo TTL, para que os itens sejam removidos automaticamente.

Figura 12 – Instrução da consulta da regra.

```
SELECT *, clientid() AS id, timestamp() AS timestamp,
(timestamp()+3600000)/1000 AS TTL
```

Fonte: Chiarini (2021).

7.4 Funções *Lambdas*

As Funções *Lambdas* são responsáveis pela manipulação e transferência dos dados entre o *DynamoDB* e o *Amazon S3*. A função responsável pelo *backup* pontual do *DynamoDB*, apresentada na Figura 13, é ativada de hora em hora através do *AWS CloudWatch*.

Figura 13 – Função *Lambda* para acionamento do *backup* do *DynamoDB*.

```
import boto3
import os
ddbRegion = os.environ['AWS_DEFAULT_REGION']
client = boto3.client('dynamodb', region_name=ddbRegion)

def lambda_handler(event, context):
    try:
        response = client.export_table_to_point_in_time(
            TableArn='**ARN**',
            S3Bucket='**Bucket**',
            ExportFormat='DYNAMODB_JSON'
        )
        return("Ok")

    except Exception as e:
        return(e)
```

Fonte: Chiarini (2021).

No momento em que o *backup* é concluído, as medições da última hora são inseridas no *Amazon S3*, acionando a função *Lambda* responsável pelas conversões e armazenamento das informações tratadas dentro do *Amazon S3*. A codificação desta função *Lambda* é apresentada no Apêndice A.

7.5 Integração do microcontrolador com a *AWS IoT Core*

Com todas as etapas anteriores concluídas, é possível que o microcontrolador se integre às funcionalidades da AWS, para isso, o código do Apêndice B faz com que o microcontrolador realize o envio das medições.

7.6 Interface com o usuário

Um dos principais pontos do projeto de monitoramento remoto é a obtenção das informações pelo usuário final. Nos tópicos anteriores, é tratada toda a arquitetura necessária para o armazenamento dos dados, com isso, o usuário final pode extrair as informações do DynamoDB e de arquivos armazenados no Amazon S3.

Entretanto, não é algo prático para o usuário obter esses dados sem uma interface, de modo a visualizar as informações de forma simples e amigável.

Para a leitura dos dados do *DynamoDB* na interface, é utilizada a função *Lambda*, apresentada na Figura 14, que realiza a extração dos dados recentes no *DynamoDB*. Para a comunicação do usuário com a função *Lambda,* é criada uma porta de comunicação HTTP pelo *API Gateway*.

Figura 14 – Função *Lambda* para leitura dos dados.

```
class DecimalEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, Decimal):
            return float(obj)
        return json.JSONEncoder.default(self, obj)
def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb',region_name='us-east-1')
    table = dynamodb.Table('EspSignals')
    response = table.scan()
    return {
        'statusCode': 200,
        'headers': {
            'Access-Control-Allow-Headers': 'Content-Type',
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': 'OPTIONS,POST,GET'
        },
        'body': json.dumps(response["Items"],cls = DecimalEncoder)
    }
```

Fonte: Chiarini (2021).

No lado do usuário, com Typescript e React, é possível recuperar os dados através da codificação da Figura 15.

Figura 15 – Recuperação dos dados do *DynamoDB* pela interface do usuário.

```
const api = axios.create();

api.get("**URL**").then((response) => {
        const data = response.data as Type;
        return(data);
      });
```

Fonte: Chiarini (2021).

A leitura dos dados do *Amazon S3* é feita diretamente pela interface do usuário, conforme apresentada a codificação na Figura 16.

Figura 16 – Recuperação dos dados do Amazon S3 pela interface do usuário.

```
import AWS from 'aws-sdk';
AWS.config.update({
  accessKeyId: '****',
  secretAccessKey: '****',
  region:'****',
});
export const s3 = new AWS.S3();
export const getS3Object = (src: string): Promise<string> => {
  return new Promise((resolve, reject) => {
    s3.getObject({
        Bucket:'****',
        Key: src,
      },
      (error: AWS.AWSError, data: AWS.S3.GetObjectOutput) => {
        if (error) {
          reject(error);
        } else{
        resolve(data.Body);
        }},);})
}
```

Fonte: Chiarini (2021).

8 CONCLUSÃO

Este trabalho apresentou análises do uso de serviços de nuvem para o monitoramento remoto de sensores, buscando simplificar o processo de desenvolvimento e garantir uma estrutura escalável, segura e com baixo custo.

Para se atingir uma compreensão da aplicação, foram estruturadas duas alternativas. Na primeira, foi utilizado o *DynamoDB* como fonte única de armazenamento, obtendo um custo estimado de 1,88 USD para o uso de um dispositivo no primeiro mês, com um crescimento mensal de aproximadamente 0,015 USD, devido ao armazenamento dos dados. Na segunda alternativa, o *DynamoDB* foi utilizado para armazenar os dados mais recentes, e o *Amazon S3* para os dados históricos, gerando um custo estimado de 1,50 USD para o uso de um dispositivo no primeiro mês e um crescimento mensal de 0,0006 USD.

A partir das análises foi possível verificar os impactos econômicos, em termos de custos, para as diferentes configurações selecionadas.

Uma das alternativas apresentadas, a segunda, foi implementada a fim de se verificar sua viabilidade técnica. Como conclusão, verificou-se que os serviços de nuvens são peças-chave para impulsionar a adoção de sistemas de monitoramento remoto.

REFERÊNCIAS

AMAZON WEB SERVICES. **MQTT**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021a. Disponível em: https://docs.aws.amazon.com/pt_br/iot/latest/developerguide/mqtt.html. Acesso em: 08 out. 2021.

AMAZON WEB SERVICES. **Computação em nuvem com a AWS**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021b. Disponível em: https://aws.amazon.com/pt/what-is-aws/. Acesso em: 18 set. 2021.

SSL.COM, Equipe de Suporte. **O que é um certificado X.509?.** *In:* Equipe de Suporte SSL.com. *[S.l.].* 23 set. 2019. Disponível em: https://www.ssl.com/pt/faqs/o-que-é-um-certificado-x-509/#. Acesso em: 18 set. 2021.

AMAZON WEB SERVICES. **Preço do AWS IoT Core**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021c. Disponível em: https://aws.amazon.com/pt/iot-core/pricing/. Acesso em: 18 set. 2021.

AMAZON WEB SERVICES. **Reducing messaging costs with basic ingest**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021d. Disponível em: https://docs.aws.amazon.com/iot/latest/developerguide/iot-basic-ingest.html/. Acesso em: 18 set. 2021.

AMAZON WEB SERVICES. **Serviço Device Shadow do AWS IoT.** *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021e. Disponível em: https://docs.aws.amazon.com/pt_br/iot/latest/developerguide/iot-device-shadows.html. Acesso em: 20 set. 2021.

AMAZON WEB SERVICES. **Rules for AWS IoT**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021. Disponível em: https://docs.aws.amazon.com/iot/latest/developerguide/iot-rules.html. Acesso em: 20 set. 2021.

AMAZON WEB SERVICES. **Amazon DynamoDB**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021f. Disponível em: https://aws.amazon.com/pt/dynamodb/. Acesso em: 20 set. 2021.

AMAZON WEB SERVICES. **Read Consistency**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021g. Disponível em: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadConsistency.html. Acesso em: 21 set. 2021.

AMAZON WEB SERVICES. **Amazon DynamoDB transactions**: Como ele funciona. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021h. Disponível em: https://docs.aws.amazon.com/pt_br/amazondynamodb/latest/developerguide/transaction-apis.html. Acesso em: 21 set. 2021.

AMAZON WEB SERVICES. **Definição do preço para capacidade provisionada**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021i. Disponível em: https://aws.amazon.com/pt/dynamodb/pricing/provisioned/. Acesso em: 21 set. 2021.

AMAZON WEB SERVICES. **Amazon S3**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021j. Disponível em: https://aws.amazon.com/pt/s3/. Acesso em: 22 set. 2021.

AMAZON WEB SERVICES. **Categorias de armazenamento do Amazon S3**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021. Disponível em: https://aws.amazon.com/pt/s3/storage-classes/. Acesso em: 08 out. 2021.

AMAZON WEB SERVICES. **Definição de preço do Amazon S3**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021k. Disponível em: https://aws.amazon.com/pt/s3/pricing/. Acesso em: 08 out. 2021.

AMAZON WEB SERVICES. **Restoring na archived object**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021. Disponível em:

https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html. Acesso em: 08 out. 2021.

AMAZON WEB SERVICES. **AWS Lambda**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021l. Disponível em: https://aws.amazon.com/pt/lambda/. Acesso em: 23 set. 2021.

AMAZON WEB SERVICES. **Preço do Amazon API Gateway**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021m. Disponível em: https://aws.amazon.com/pt/api-gateway/pricing/. Acesso em: 23 set. 2021

AMAZON WEB SERVICES. **Tutorial:** Schedule AWS Lambda Functions Using CloudWatch Events. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021n. Disponível em: https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/RunLambdaSchedule.html. Acesso em: 25 set. 2021

AMAZON WEB SERVICES. **Preço do Amazon CloudWatch.** *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021. Disponível em: https://aws.amazon.com/pt/cloudwatch/pricing/. Acesso em: 25 set. 2021

AMAZON WEB SERVICES. **Definição de preço do AWS Lambda.** *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021o. Disponível em: https://aws.amazon.com/pt/lambda/pricing/. Acesso em: 23 set. 2021

AMAZON WEB SERVICES. **Preço sob demanda do Amazon EC2.** *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021p. Disponível em: https://aws.amazon.com/pt/ec2/pricing/on-demand/#Data_Transfer. Acesso em: 08 out. 2021

AMAZON WEB SERVICES. **Definição do preço para capacidade sob demanda**. *[S.l.].* Amazon Web Services, Inc. and/or its affiliates, c2021q. Disponível em: https://aws.amazon.com/pt/dynamodb/pricing/on-demand/. Acesso em: 21 set. 2021.

FILIPEFLOP**. Módulo WiFi ESP8266 NodeMcu ESP-12**. *[S.l.].* FilipeFlop, c2021a. Disponível em: https://www.filipeflop.com/produto/modulo-wifi-esp8266-nodemcu-esp-12/. Acesso em: 18 set. 2021.

FILIPEFLOP. **Sensor de Umidade e Temperatura DHT11.** *[S.l.].* FilipeFlop, c2021b. Disponível em: https://www.filipeflop.com/produto/sensor-de-umidade-e-temperatura-dht11/. Acesso em: 18 set. 2021.

FERNANDES, Natália. **O que é o protocolo MQTT?.** *In:* HI Tecnologia - Automação Industrial. *[S.l.].* 26 fev. 2021. Disponível em: https://www.hitecnologia.com.br/blog/o-que-e-protocolo-mqtt/. Acesso em: 12 out. 2021.

KRANENBURG, Van; BASSI. **IoT Challenges**. Communications in Mobile Computing 2012 1:9

ORACLE. **O que é IoT?.** *[S.l.].* Oracle, c2021. Disponível em: https://www.oracle.com/br/internet-of-things/what-is-iot/. Acesso em: 07 out. 2021.

APÊNDICE A – FUNÇÃO LAMBDA PARA CONVERSÕES DOS DADOS PARA O AMAZON S3

```
#%% Imports
from datetime import date, datetime, timedelta
import json
import urllib.parse
import boto3
import os
from gzip import GzipFile
from io import BytesIO
import pandas as pd
import numpy as np
from datetime import timezone, datetime, timedelta
import pytz

%## Configs
ddbRegion = os.environ['AWS_DEFAULT_REGION']
client = boto3.client('iot-data', region_name=ddbRegion)
s3 = boto3.client('s3', region_name=ddbRegion)
bucket = '**Bucket**'
bucketExport = "**Bucket**"

def lambda_handler(event, context):
    for record in event['Records']:
        try:
            #%% Read data
            key =
urllib.parse.unquote_plus(record['s3']['object']['key'])
            data = GzipFile(None,'rb', fileobj =
BytesIO(s3.get_object(Bucket = bucket, Key =
key)['Body'].read())).read().decode('utf-8')
            data =
data.replace('{"Item":','').replace('}\n',',').replace('"S":','').repla
ce('"N":','').replace(":{",':').replace('},',',')
            data = '[' + data[:-1] + ']'
            df = pd.read_json(data, convert_dates=True)
            data = None

            #%% Adjust interval
            endTime = datetime.now().replace(minute = 0, second = 0,
microsecond = 0)
            beginTime = endTime - timedelta(hours=1)
            df = df[df['timestamp'] >= beginTime]
```

```
            df = df[df['timestamp'] < endTime]

            #%% Variables
            id = df['id'][0]
            tzUser = "America/Sao_Paulo"

            #%% Adjust timestamp
            df['timestamp'] =
df['timestamp'].dt.tz_localize(timezone.utc)
            df['timestamp'] =
df['timestamp'].dt.tz_convert(pytz.timezone(tzUser))

            #%% Remove unsed variables
            df = df.drop(['id','TTL'], axis=1)

            #%% For in all ranges and valide if is a new item 0-day 1-
week 2-month 3-year
            now = datetime.now(pytz.timezone(tzUser))
            keys = list(df.keys())

            for x in range(4):
                newItem = False
                readFile = ''
                interval = '1S'

                if x == 0:
                    readFile = now.strftime('Day/%Y/%m/%d')
                    if now.hour == 0:
                        newItem = True

                elif x == 1:
                    interval = '15S'
                    readFile =
now.strftime('Week/%Y/'+str(now.isocalendar()[1]))
                    if now.weekday() == 0 and now.hour == 0:
                        newItem = True

                elif x == 2:
                    interval = '60S'
                    readFile = now.strftime('Month/%Y/%m')
                    if now.day == 1 and now.hour == 0:
                        newItem = True
```

```
                else:
                    interval = '900S'
                    readFile = now.strftime('Year/%Y')
                    if now.month == 1 and now.day == 1 and now.hour ==
0:
                        newItem = True

                #%% Agroup data by time specific
                dfCopy = df.copy()

                if x != 0:
                    dfCopy = dfCopy.groupby(pd.Grouper(key =
'timestamp', freq = interval)).mean()
                    dfCopy = dfCopy.dropna(axis = 0, how = 'all')
                    dfCopy = dfCopy.replace({np.nan: None})

                #%% Read data and list categories other data
                if not newItem:
                    try:
                        objPastData = json.loads(s3.get_object(Bucket =
bucketExport, Key =
(id+'/'+readFile+'.json'))['Body'].read().decode('utf-8'))
                        info = objPastData['indexes']
                        pastValues =
pd.DataFrame(objPastData['values']).set_axis(info,axis=1).replace({np.n
an: None})

                        #%% Add None(null) for undata variables
                        diff = list(set(info) - set(keys))
                        dfCopy[diff] = None

                        #%% If has new data add None(null) in past data
                        pastDiff = list(set(keys) - set(info))
                        pastValues[pastDiff] = None

                    except Exception as e:
                        print(e)
                        newItem = True

                #%% Fix index
                if x != 0:
                    dfCopy['timestamp'] = dfCopy.index.astype(str)
                else:
```

```
                    dfCopy['timestamp'] =
dfCopy['timestamp'].astype(str)

                #%% Agroup data
                if not newItem:
                    dfCopy = pd.concat([pastValues, dfCopy])

                #%% Converting to JSON
                if not newItem:
                    for x in keys:
                        if not x in info:
                            info.append(x)
                    dfCopy = dfCopy[info]
                    indexes = info
                else:
                    dfCopy = dfCopy[keys]
                    indexes = keys

                filename = id+'/'+readFile+'.json'

                exportData = {}
                exportData['indexes'] = indexes
                exportData['values'] = dfCopy.values.tolist()
                exportData = bytes(json.dumps(exportData).encode('UTF-
8'))

                s3.put_object(Body=exportData, Bucket=bucketExport,
Key=filename)

        except Exception as e:
            print(e)

    return('Ok')
```

Fonte: Chiarini (2021).

APÊNDICE B – INTEGRAÇÃO DO MICROCONTROLADOR

```
#include "FS.h"
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <DHT.h>

const char *ssid="*******";
const char *password="********";

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org");

#define DHTPIN 4
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);
float temperature;

const  char* AWS_endpoint = "*********" ;

const char* topicSignal = "$aws/rules/**Topic**";

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

}

WiFiClientSecure espClient;
PubSubClient client(AWS_endpoint, 8883, callback, espClient); //set
MQTT port number to 8883 as per //standard

#define BUFFER_LEN 266
long lastMsg = 0;
char msg[BUFFER_LEN];
int value = 0;
```

```
byte mac[6];
char mac_Id[18];

void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network

  espClient.setBufferSizes(4096,4096);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  timeClient.begin();

  while(!timeClient.update()){
    timeClient.forceUpdate();
  }

  espClient.setX509Time(timeClient.getEpochTime());

}


void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect(clientId)) {
      Serial.println("connected");
```

```
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      char buf[256];
      espClient.getLastSSLError(buf,256);
      Serial.print("WiFiClientSecure SSL error: ");
      Serial.println(buf);

      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}


void setup() {

  Serial.begin(115200);
  Serial.setDebugOutput(true);
  delay(50);
  dht.begin();
  pinMode(LED_BUILTIN, OUTPUT);

  setup_wifi();
  delay(1000);

  if (!SPIFFS.begin()) {
    Serial.println("Failed to mount file system");
    return;
  }

  Serial.print("Heap: "); Serial.println(ESP.getFreeHeap());

  // Load certificate file
  File cert = SPIFFS.open("/cert.der", "r"); //replace cert.crt with
your uploaded file name
  if (!cert) {
    Serial.println("Failed to open cert file");
  }
  else
    Serial.println("Success to open cert file");
```

```
  delay(1000);

  if (espClient.loadCertificate(cert))
    Serial.println("cert loaded");
  else
    Serial.println("cert not loaded");


  // Load private key file
  File private_key = SPIFFS.open("/private.der", "r"); //replace
private with your uploaded file name
  if (!private_key) {
    Serial.println("Failed to open private cert file");
  }
  else
    Serial.println("Success to open private cert file");

  delay(1000);

  if (espClient.loadPrivateKey(private_key))
    Serial.println("private key loaded");
  else
    Serial.println("private key not loaded");


   // Load CA file
  File ca = SPIFFS.open("/ca.der", "r");
  if (!ca) {
    Serial.println("Failed to open ca ");
  }
  else
    Serial.println("Success to open ca");

    delay(1000);

    if(espClient.loadCACert(ca))
      Serial.println("ca loaded");
    else
      Serial.println("ca failed");
      Serial.print("Heap: "); Serial.println(ESP.getFreeHeap());
  }
```

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 1000) {
    lastMsg = now;

    temperature = dht.readTemperature();
    snprintf (msg, BUFFER_LEN, "{\"temp\" : %f}",temperature);
    Serial.print("Publish message: ");
    Serial.println(msg);
    client.publish(topicSignal, msg);

    Serial.print("Heap: "); Serial.println(ESP.getFreeHeap());
  }
  digitalWrite(LED_BUILTIN, HIGH);
  delay(100);
  digitalWrite(LED_BUILTIN, LOW);
  delay(100);
}
```

Fonte: Chiarini (2021).