Projeto Final da Fase 2: Etapa 2 – Arquitetura e Modelagem

https://github.com/EmbarcaTech-2025/projeto-final-antonio_almeida/tree/main

Arquitetura IoT com Raspberry Pi Pico, Servidor Ubuntu e Cloudflare para Ingestão Segura de Dados em Datalake na Nuvem

Aluno: *Antonio Carlos Ferreira de Almeida*

Data: 08/08/2025

Sumário

Índice de figuras

Índice de Tabelas

a) Escopo do projeto

1 – Apresentação do projeto.

Arquitetura IoT com Raspberry Pi Pico, Servidor Ubuntu e Cloudflare para Ingestão Segura de Dados em Datalake na Nuvem

Este projeto propõe uma arquitetura IoT voltada para **coleta estruturada de dados sensoriais** e envio seguro para um **datalake na nuvem**, permitindo análises avançadas e geração de inteligência.

A solução integra em 3 etapas:

- **Raspberry Pi Pico** rodando servidor HTTP simplificado com **lwIP** para microcontroladores, captando dados de sensores e atribuindo *metadata* (timestamp, ID de origem, localização, etc.);
- **Servidor Ubuntu headless** na mesma rede local, atuando como gateway, proxy reverso e ponto de consolidação;
- **Cloudflare Tunnel** garantindo que a comunicação com a nuvem seja segura, confiável e protegida contra ataques.

Título do projeto: - DataHarbor IoT.

Inspiração:
A ideia veio da analogia com sistemas físicos de armazenamento e transporte (como portos marítimos) e a necessidade de um "ponto de coleta local" para dados IoT, antes deles seguirem para a nuvem — um lugar confiável e estruturado para "ancorar" e organizar esses dados.

Além disso, "Harbor" também remete a segurança, proteção e estabilidade — tudo que queremos para o armazenamento e gerenciamento de dados sensíveis em IoT.

2 – Motivação.

O **Raspberry Pi Pico** é um dispositivo de baixo custo e consumo, capaz de interagir com sensores e atuadores de forma eficiente, mas não projetado para exposição direta à internet devido a restrições de segurança e processamento.

Um **servidor Ubuntu** — especialmente em hardware robusto como **Intel Xeon 28 núcleos e 64 GB de RAM** — oferece recursos para:

- Gerenciamento seguro de conexões;
- Aplicação de firewall e autenticação;
- Roteamento inteligente e pré-processamento de dados;
- Integração direta com APIs e serviços de nuvem para armazenamento massivo.

O **Cloudflare Tunnel** atua como elo seguro entre o servidor local e o provedor de nuvem, evitando a exposição de IPs reais e agregando funcionalidades como SSL/TLS, mitigação de DDoS e otimização de tráfego.

3 – Arquitetura Proposta

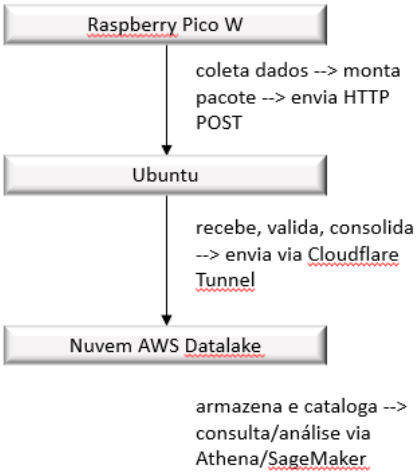


Figura 1:Arquitetura.

1. Raspberry Pi Pico

- Captura dados de múltiplos sensores;
- Enriquecimento dos dados com *metadata* (data/hora, ID, origem, contexto);
- Servidor HTTP embutido para envio dos dados ao Ubuntu via rede local;
- Processamento mínimo, priorizando baixa latência de envio.

2. Servidor Ubuntu Headless (Intel Xeon)

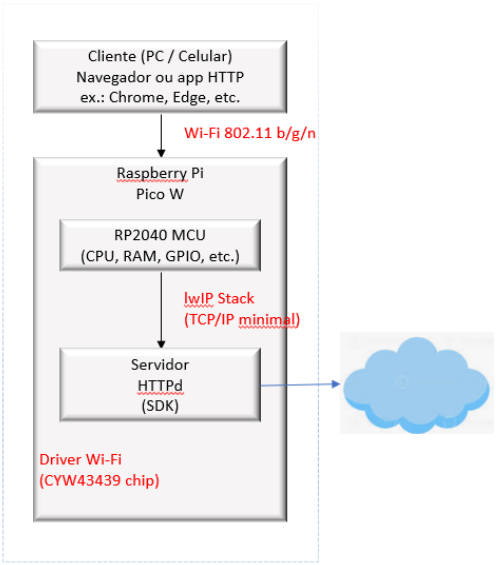
- Atua como **gateway** e **proxy reverso**;
- Recebe dados de múltiplos Picos, consolida e valida pacotes;
- Encaminha dados para a nuvem via APIs seguras;
- Garante resiliência e alta disponibilidade;
- Pode executar *batch processing* inicial ou compactação antes do envio.

3. Cloudflare Tunnel para receber URL pré-assinada (https://meu-bucket.s3.amazonaws.com/dados.json)

- Protege a entrada de dados na nuvem;
- Evita exposição direta do servidor e IP;
- Garante criptografia ponta-a-ponta;
- Facilita escalabilidade global.

4. Datalake na Nuvem

- Armazena dados brutos e enriquecidos;
- **Amazon S3** como repositório central.
- Dados guardados em **formato otimizado** (Parquet, ORC, JSON) para reduzir custo e acelerar leitura.
- Particionamento por **ano/mês/dia** e **tipo de dado** para consultas rápidas.
- **AWS Glue Data Catalog** para criar uma visão organizada dos dados brutos.
- Permite que qualquer *center of excellence* saiba **o que** existe no lago e **como filtrar**.
- **Centros de Competência** que desejam criar inteligência através de dados usam **Athena** (consulta SQL no S3), **EMR/Spark** ou **SageMaker** para análises.
- Cada área extrai apenas os dados relevantes, sem poluir ou duplicar o lago.
- Para cobrança por uso, o mais comum é implementar uma camada de serviço que monitore o uso (ex.: logs do CloudTrail, CloudWatch, ou métricas customizadas). Com esses dados é possível criar **faturamento via sistema externo** (ex.: Stripe, PayPal, etc).



4 – Objetivo Estratégico.

Não se trata de servir páginas a usuários finais, mas de construir um **pipeline robusto e seguro de ingestão de dados IoT**.
O foco é criar uma **fonte única de verdade** (*single source of truth*) para os dados captados, permitindo que áreas especializadas possam:

1. Filtrar apenas os dados relevantes à sua atuação;
2. Realizar análises históricas e preditivas;
3. Alimentar modelos de machine learning e sistemas de decisão.

5 – Benefícios da Arquitetura.

- **Segurança** — O Pico nunca é exposto diretamente à internet;
- **Escalabilidade** — O servidor Ubuntu pode receber dados de centenas de dispositivos simultaneamente;
- **Manutenção Simplificada** — Logs e atualizações centralizados;
- **Alta Disponibilidade** — Cloudflare mantém o sistema acessível mesmo sob ataque;
- **Inteligência de Dados** — O datalake permite análises complexas e descobertas a partir de dados brutos consolidados.

6 – Protocolo de comunicação.

Se faz necessário a criação de um **protocolo de comunicação simples** que funcione para todos tipos de dados no **Raspberry Pi Pico** (em C), com um **pacote fixo** para facilitar o envio e o parsing, tanto para grandezas físicas (temperatura, pressão, umidade) quanto para estados lógicos (ligado/desligado).

Objetivo

- Ter **um único formato de pacote** para qualquer tipo de dado.
- Incluir **identificação do tipo de dado**.
- Possuir **checagem de erro** simples (CRC ou soma).
- Ter tamanho **fixo** para parsing rápido no Pico.

b) Especificação do hardware.

1 – Diagrama em blocos.

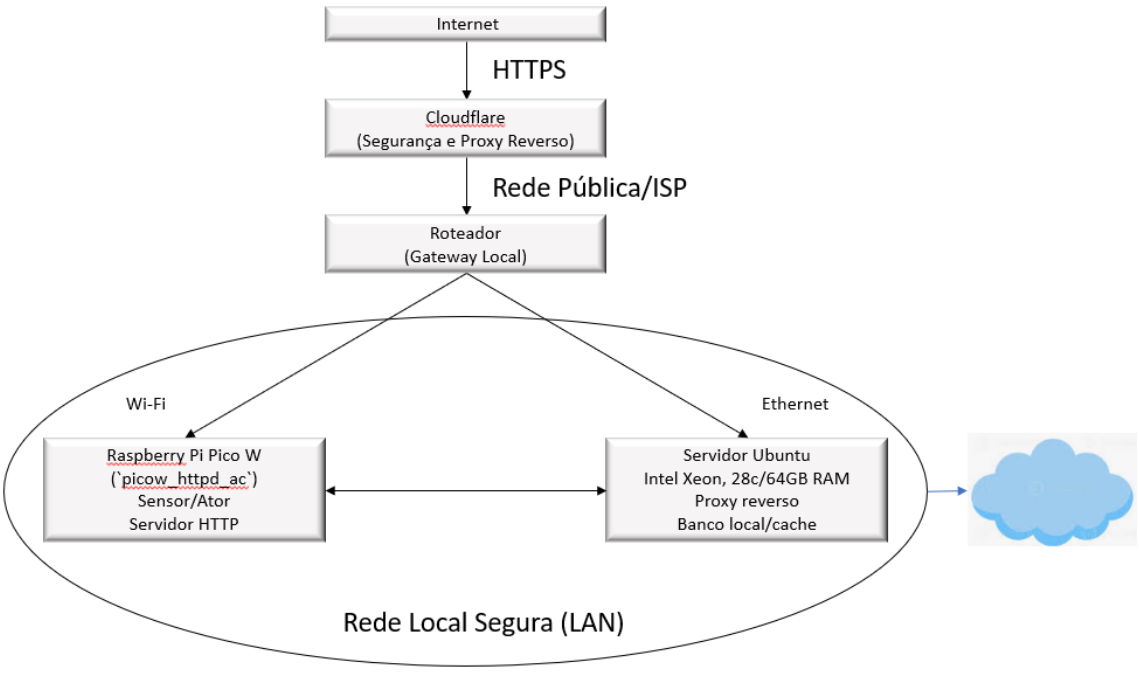


Figura 2: Diagrama em Blocos.

2 – Função de cada bloco.

1. Inicialização do Dispositivo Raspberry Pi Pico;
2. Captura de Dados no Pico;
3. Envio dos Dados para o Servidor Ubuntu (via HTTP local);
4. Recepção e Validação dos Dados no Ubuntu;
5. Pré-processamento e Consolidação (Opcional);
6. Comunicação Segura via Cloudflare Tunnel;
7. Armazenamento no Datalake AWS;
8. Consulta e Análise dos Dados (Centros de Competência) (Opcional);
9. Monitoramento, Cobrança e Gestão (Opcional).

3 – Configuração de cada bloco.
Sistema de Controle (Raspberry Pi Pico W) Bitdoglab

1. Inicialização do Dispositivo Raspberry Pi Pico
   - ✓ Iniciar servidor HTTP simplificado (lwIP).
   - ✓ Inicializar sensores e interfaces de captura.
   - ✓ Definir metadados fixos (ID dispositivo, localização, etc).
   - ✓ Aguardar evento de coleta periódica ou acionamento externo.
2. Captura de Dados no Pico
   - ✓ Ler sensores físicos (grandezas físicas: temperatura, umidade, etc).
   - ✓ Montar pacote de dados com:
   - ✓ Valores dos sensores;
   - ✓ Timestamp atual;
   - ✓ Metadados de origem.
   - ✓ Armazenar pacote temporariamente para envio.
3. Envio dos Dados para o Servidor Ubuntu (via HTTP local)
   - ✓ Pico realiza requisição HTTP POST para servidor Ubuntu na rede local.
   - ✓ Dados são enviados no corpo da requisição, formato JSON ou outro definido.
   - ✓ Esperar resposta de confirmação do Ubuntu.
4. Recepção e Validação dos Dados no Ubuntu
   - ✓ Receber requisição HTTP.
   - ✓ Validar integridade e formato do pacote.
   - ✓ Aplicar autenticação e regras de segurança.
   - ✓ Armazenar dados temporariamente localmente (cache ou banco local).
5. Pré-processamento e Consolidação (Opcional)
   - ✓ Agregar dados recebidos de múltiplos dispositivos.
   - ✓ Executar compressão ou filtragem simples.
   - ✓ Preparar pacote para envio seguro à nuvem.

6. Comunicação Segura via Cloudflare Tunnel

   - ✓ Estabelecer conexão segura e criptografada com a nuvem.
   - ✓ Transmitir dados para o serviço AWS (S3/Datalake) via APIs protegidas.
   - ✓ Confirmar recebimento e sucesso da operação.

7. Armazenamento no Datalake AWS

   - ✓ Receber dados no bucket S3 (armazenamento em formatos otimizados).
   - ✓ Atualizar catálogo de dados no AWS Glue.
   - ✓ Monitorar e registrar logs de acesso e consumo.

8. Consulta e Análise dos Dados (Centros de Competência) (Opcional)

   - ✓ Usuários ou sistemas especializados acessam dados via Athena, EMR ou SageMaker.
   - ✓ Realizam consultas específicas e análises preditivas.
   - ✓ Geram insights e relatórios a partir do datalake.

9. Monitoramento, Cobrança e Gestão (Opcional)

   - ✓ Monitorar volume de dados acessados e transferidos.
   - ✓ Gerar relatórios de uso e custos.
   - ✓ Executar cobrança via sistema externo integrado.

4 – Formato do Pacote (8 bytes).

Byte	Campo	Descrição
0	**STX**	Início do pacote (0x7E)
1	**ID**	Código do sensor/dispositivo (0–255)
2	**Tipo**	Tipo de dado (0=temp, 1=pressão, 2=umidade, 3=ligado/desligado)
3-4	**Valor**	Dado em **inteiro de 16 bits** (escala fixa para todos)
5	**Flags**	Bits extras (ex.: ligado/desligado, unidade de medida)
6	**CRC**	Soma dos bytes 1–5 (mod 256)
7	**ETX**	Fim do pacote (0x7F)

Figura 3: Estrutura de Pacote.

- **Convenção dos Dados:**
  - **Temperatura**: Valor × 100 (ex.: 25,37°C → 2537)
  - **Pressão**: Valor × 10 (ex.: 1013,2 hPa → 10132)
  - **Umidade**: Valor × 100 (ex.: 65,4% → 6540)
  - **Ligado/Desligado**: Valor 1 = ligado, 0 = desligado
- **Tamanho:**
  - Todos usam **int16_t**, assim o parsing é o mesmo.
  -
- **Exemplo de Pacote:**

- [0] 0x7E (STX)
- [1] 0x02 (ID = 2)
- [2] 0x00 (Tipo = temperatura)
- [3] 0x09 (Valor alto = 2537 >> 8)
- [4] 0xE9 (Valor baixo = 2537 & 0xFF)
- [5] 0x00 (Flags = não usado)
- [6] 0xF4 (CRC = (0x02+0x00+0x09+0xE9+0x00) % 256)
- [7] 0x7F (ETX)

- **Estrutura:**

```
typedef struct {
    uint8_t start;
    uint8_t id;
    uint8_t type;
    int16_t value;
    uint8_t flags;
    uint8_t crc;
    uint8_t end;
} DataPacket;
```

- **Cálculo CRC:**

```
uint8_t calc_crc(uint8_t id, uint8_t type, int16_t value, uint8_t flags) {
    uint8_t sum = id + type + ((value >> 8) & 0xFF) + (value & 0xFF) + flags;
    return sum % 256;
}
```

- **Construtor:**

```
DataPacket create_packet(uint8_t id, uint8_t type, int16_t value, uint8_t flags) {
    DataPacket p;
    p.start = STX;
    p.id = id;
    p.type = type;
    p.value = value;
    p.flags = flags;
    p.crc = calc_crc(p.id, p.type, p.value, p.flags);
    p.end = ETX;
    return p;
}
```

- **Apoio:**

```c
void send_packet(DataPacket *p) {
    uint8_t *ptr = (uint8_t *)p;
    for (int i = 0; i < sizeof(DataPacket); i++) {
        putchar(ptr[i]); // Pode ser substituído por UART, SPI, etc.
    }
}
```

- **Parsing no Receptor:**

  O receptor lê byte a byte:

  1. Espera STX;
  2. Lê os próximos 6 bytes;
  3. Confere ETX;
  4. Calcula CRC e valida;
  5. Converte value de volta para a grandeza original.

5 – Lista de materiais (uso de sensores limitado para demonstração)**.**

Quantidade	***Descrição***
01	**Raspberry Pi Pico W - BitDogLab** (com Wi-Fi integrado)
01	**Cabo micro-USB** (para alimentação e programação)
01	**Temperatura e Umidade** – Sensor SHT31-D ou BME280 (I²C)
01	**Pressão Atmosférica** – BMP280 ou BME280 (I²C)
01	**Luminosidade** – BH1750 (I²C)
01	**Qualidade do Ar (VOC/CO_2)** – CCS811 ou SGP30 (I²C)
01	**Módulo Relé 5V (1 canal) – 1 unidade (para ligar/desligar uma carga simulada)**
01	**Acelerômetro/Inclinação** – MPU6050 (I²C)
01	**Sensor Magnético / Bússola** – HMC5883L ou QMC5883 (I²C)
01	**Protoboard**
...	**Cabos jumper macho-fêmea e macho-macho**
01	**Fonte de alimentação 5V USB**
...	**Ferramentas e Acessórios**

Tabela 1: Lista de Materiais.

7 – Circuito completo do hardware.

- Todas as referências com relação a pinagem...

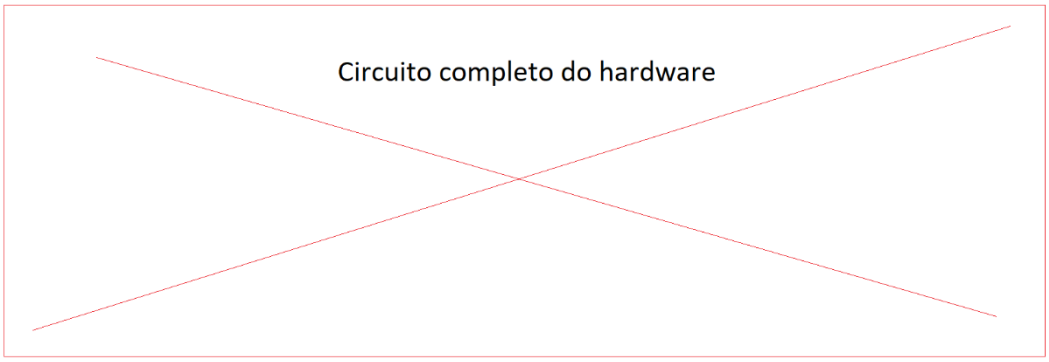


Figura 4: Circuito completo.

c) Especificação do firmware/software.

1 – Blocos funcionais.

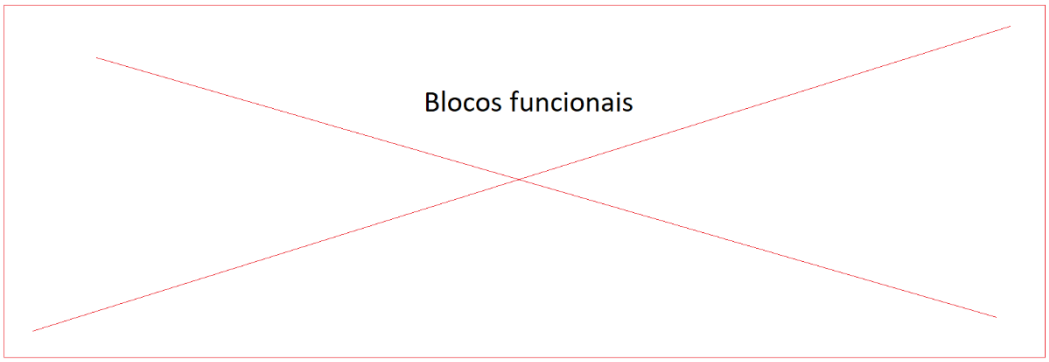


Figura 5: Blocos Funcionais.

1. Dispositivo IoT: Raspberry Pi Pico

Função:

- Captura dados sensoriais (grandezas físicas como: temperatura, umidade, etc).
- Anexa metadados: timestamp, ID do dispositivo, localização, contexto da coleta.
- Executa servidor HTTP leve usando a stack lwIP para comunicação local.
- Envia pacotes estruturados para o gateway Ubuntu via rede local.
- Prioriza baixo consumo e latência, sem processamento pesado.

Interfaces:

- Sensores físicos (GPIO, I2C, SPI).
- Rede local (Wi-Fi ou Ethernet via módulo externo).

---

2. Gateway Local: Servidor Ubuntu Headless (Intel Xeon 28 núcleos, 64 GB RAM)

Função:

- Recebe dados de múltiplos dispositivos Pico via HTTP.
- Consolida e valida pacotes recebidos, assegurando integridade.
- Executa proxy reverso, roteamento e controle de acesso.
- Processa (opcionalmente) dados localmente: filtragem, compactação, pré-análise.
- Garante segurança: firewall, autenticação, monitoramento.
- Encaminha os dados para a nuvem por meio de APIs seguras e túnel Cloudflare.
- Gerencia alta disponibilidade e resiliência do sistema local.

Interfaces:

- Rede local interna.
- Cloudflare Tunnel para comunicação externa segura.

---

3. Camada de Segurança e Rede: Cloudflare Tunnel

Função:

- Estabelece um túnel seguro e criptografado entre o servidor Ubuntu e a nuvem.
- Oculta IPs reais e previne ataques (DDoS, exploits, etc).
- Fornece SSL/TLS automático para comunicação segura ponta a ponta.
- Otimiza a performance e escalabilidade global do serviço.

Interfaces:

- Rede pública da Internet para a nuvem AWS.
- Rede local via servidor Ubuntu.

---

4. Armazenamento e Processamento na Nuvem: Datalake AWS (Amazon S3 + Glue + Athena + SageMaker)

Função:

- Armazenamento centralizado de dados brutos e enriquecidos em formatos otimizados (Parquet, ORC, JSON).
- Catalogação dos dados com AWS Glue para facilitar busca e organização.
- Particionamento eficiente para consultas rápidas (por tempo, tipo, origem).
- Consultas analíticas via Athena (SQL sobre S3).
- Processamento avançado e machine learning com EMR/Spark e SageMaker.

- Suporte a múltiplos centros de competência que acessam dados filtrados conforme interesse.
- Monitoramento do uso de dados para faturamento externo se necessário.

Interfaces:

- Cloudflare para entrada segura de dados.
- APIs internas para consulta e análise dos dados.

---

5. Camada de Monetização e Controle de Acesso (Opcional)

Função:

- Monitorar acessos e consumo do datalake via logs (CloudTrail, CloudWatch).
- Integrar sistema de faturamento externo (ex.: Stripe, PayPal) baseado no consumo real.
- Gerenciar autenticação e autorização para usuários e sistemas externos.

2 – Descrição das funcionalidades.

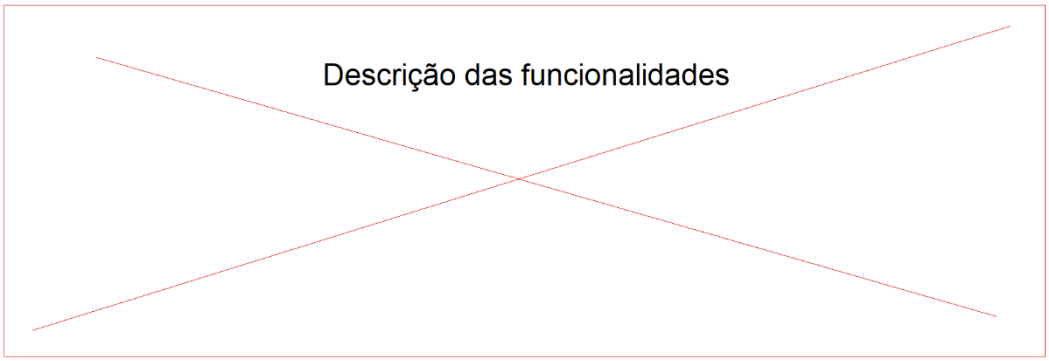


Figura 6: Descrição das Funcionalidades.

3 – Definição das Variáveis e Constantes do Sistema.

Variável	***Tipo***	***Descrição***

Tabela 2: Variáveis do Sistema.

Constantes	Tipo	Descrição

Tabela 3: Constantes do Sistema.

4 – Fluxograma.

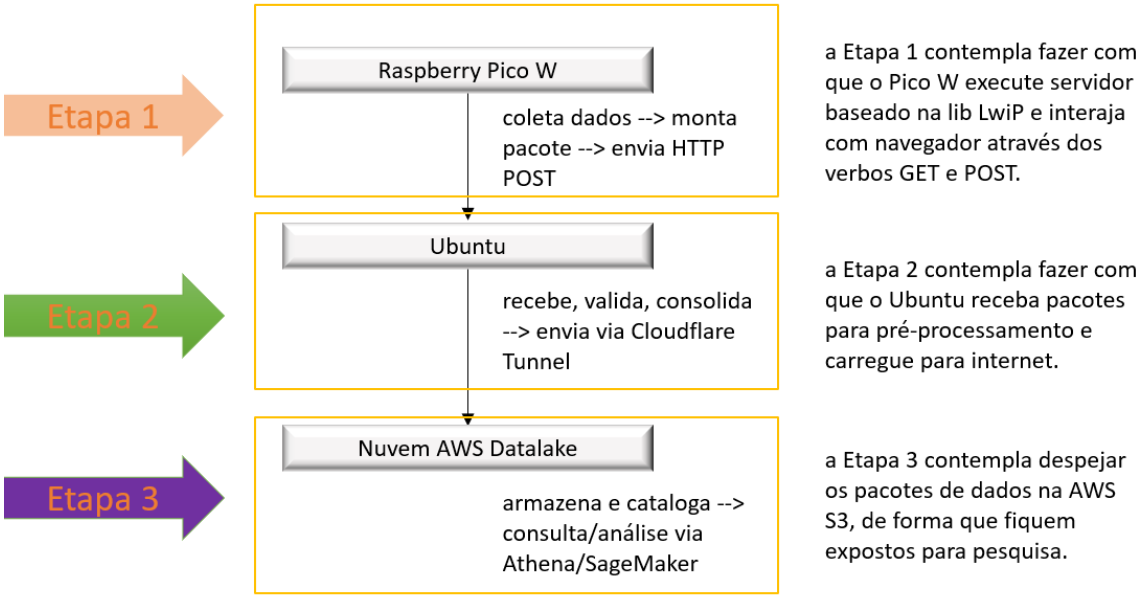


Figura 7: Fluxograma.

5 – Inicialização.

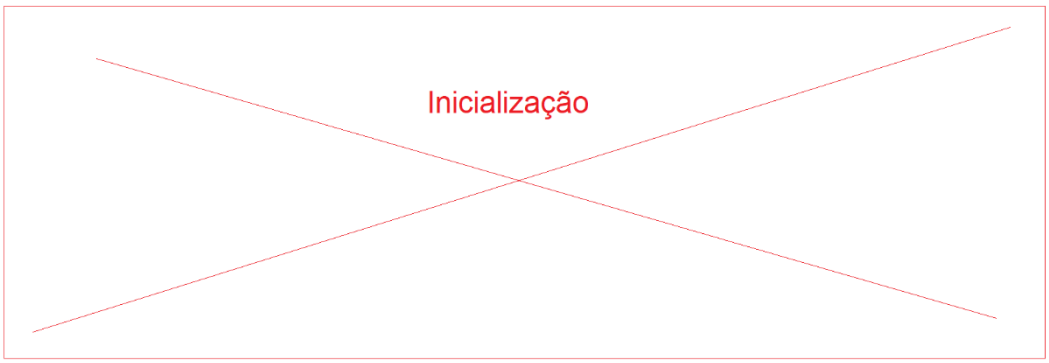


Figura 8: Inicialização do Software.

6 – Configurações dos registros.

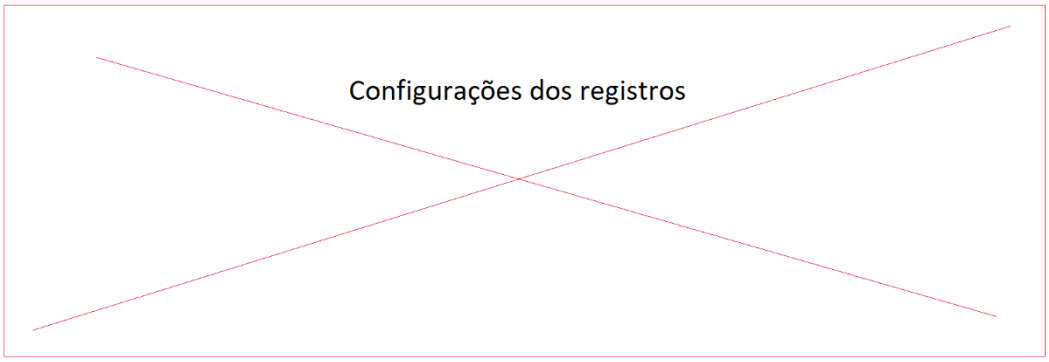


Figura 9: Configurações dos Registros

7 – Estrutura e formato dos dados.

- ✓ ***I2C (OLED SSD1306):***

  ....

- ✓ ***ADC:***

  ...

- ✓ ***OLED:***

  ...

- ✓ ***GPIO:***

  ...

8 – Organização da memória.

- Endereços de memória utilizados indiretamente:
  - ...
  - .

9 – Protocolo de comunicação.

- I2C (Inter-Integrated Circuit) é um ...

10 – Formato do pacote de dados.

- Pacote I2C ...

Figura 10: Exemplo de Pacote de dados.

d) Execução do projeto

1 – Metodologia.

- Basicamente, ...

2 – Testes de validação.

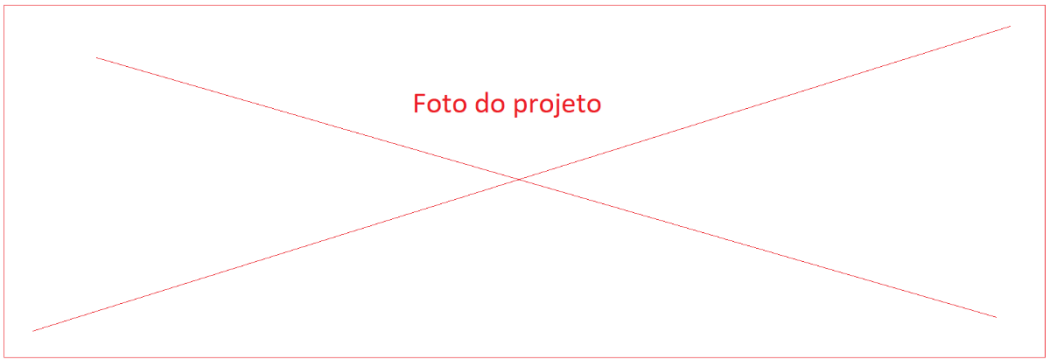
Abaixo encontram-se os testes e validações de software...

3 – Discussão dos Resultados.

- Objetivos alcançados:
  - ✓ **A Bitdoglab** foi configurada para controlar ...

4 – Link do Vídeo do projeto funcionando.

Link do YouTube, *não listado:*



e) Referência Bibliográficas

1 - https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html#pico-1-family

2 - https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf

3 - https://datasheets.raspberrypi.com/rp2040/hardware-design-with-rp2040.pdf

4- https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf

5 - https://datasheets.raspberrypi.com/picow/connecting-to-the-internet-with-pico-w.pdf