SENAI

Serviço Nacional de Aprendizagem Industrial

PELO FUTURO DO TRABALHO

Situação de Aprendizagem

Montagem Smart 4.0

Alunos: Alessandro Bento, Cesar Glufke, Daniel Barros, Jefferson Barros

Sumário

1. INTRODUÇÃO

Este documento apresenta o funcionamento dos sistemas de sensores de análise da etapa de montagem da Smart 4.0. Um protótipo com fins de aprendizagem na matéria de Modelagem de Sistemas do curso de Analise e Desenvolvimento Sistemas.

Composto por 4 sensores módulos sensor Lm393 de obstáculo infravermelho e um módulo ESP32 e uma protoboard, que detectam a presença da base no ponto de montagem e as possíveis lâminas nas laterais. As informações seram monstradas ainda em uma dashboard do sistema Tago.IO, ao lado dos dados provenientes do sistema xxx coletados com o Node-red.

2. REDES

Tago.IO é uma plataforma web que se destina ao monitoramento de ambientes por meio de dispositivos IoT (Internet das Coisas). A plataforma é totalmente baseada em nuvem (cloud) e é projetada para facilitar o desenvolvimento e implementação ágeis de soluções de IoT no mercado. Ela oferece diversas funcionalidades para suportar uma ampla gama de aplicações em vários setores, incluindo automação industrial, irrigação inteligente, localização interna de depósitos, composição, refrigeração, telemática e outros.

Algumas das características e capacidades típicas da plataforma Tago.IO podem incluir:

Monitoramento Remoto:
Capacidade de monitorar e coletar dados de dispositivos IoT em tempo real, permitindo o acompanhamento remoto de condições e eventos.

Análise de Dados:
Ferramentas para análise de dados coletados, permitindo a extração de insights e informações úteis a partir das informações geradas pelos dispositivos IoT.

Desenvolvimento Ágil:
Recursos que facilitam o desenvolvimento rápido de soluções de IoT, acelerando o ciclo de vida do projeto.

Conectividade:
Suporte para diversos dispositivos IoT e protocolos de comunicação, permitindo a integração de uma ampla variedade de dispositivos na plataforma.

Personalização:
Flexibilidade para personalizar e adaptar a plataforma de acordo com as necessidades específicas de diferentes aplicações e setores.

Setores Específicos:
Foco em atender a necessidades específicas de setores como automação industrial, agricultura (irrigação inteligente), logística (localização interna de depósitos), composição, refrigeração e telemática, entre outros.

Em resumo, a Tago.IO parece ser uma plataforma abrangente que fornece recursos para simplificar o desenvolvimento e a implementação de soluções baseadas em IoT em diversos cenários e setores.

2.1. COMO FUNCIONA O TAGO.IO

Registro e Configuração:
Os usuários começam registrando-se na plataforma Tago.IO. Após o registro, eles configuram sua conta, definem parâmetros e personalizam as configurações de acordo com as necessidades do projeto.

Conexão de Dispositivos IoT:
A plataforma oferece suporte para uma variedade de dispositivos IoT e protocolos de comunicação.
Os dispositivos IoT são conectados à plataforma Tago.IO, geralmente por meio de APIs, SDKs ou integrações específicas.

Coleta de Dados:
Os dispositivos IoT conectados começam a enviar dados para a plataforma. Esses dados podem incluir informações ambientais, leituras de sensores, status de dispositivos, etc.
A plataforma é projetada para coletar e armazenar esses dados de maneira eficiente.

Armazenamento e Gerenciamento de Dados:
Os dados coletados são armazenados na nuvem, permitindo fácil acesso e recuperação.
A plataforma fornece ferramentas para gerenciar grandes volumes de dados, incluindo recursos de armazenamento em série, bancos de dados, etc.

Análise de Dados:
A Tago.IO oferece ferramentas analíticas para processar e analisar os dados coletados.
Os usuários podem criar dashboards personalizados, gráficos e relatórios para extrair insights úteis dos dados.

Automação e Controle:
Com base nos dados e análises, os usuários podem implementar automações e controles. Por exemplo, acionar ações com base em determinadas condições ou alertar usuários sobre eventos críticos.

Integrações e Extensões:
A plataforma Tago.IO geralmente suporta integrações com outras ferramentas e serviços, permitindo uma maior flexibilidade.
Pode haver extensões ou aplicativos disponíveis para estender as funcionalidades da plataforma.

Gerenciamento Remoto:
Os usuários podem monitorar e gerenciar remotamente dispositivos e dados por meio da interface da plataforma, que geralmente é acessada por meio de um navegador web.

Em resumo, a Tago.IO funciona como uma ponte entre dispositivos IoT e usuários, facilitando a coleta, armazenamento, análise e gestão de dados gerados por esses dispositivos, com o objetivo de criar soluções eficazes e personalizadas para diversos cenários.

2.2. MQTT

O MQTT, ou Message Queuing Telemetry Transport, é um protocolo de comunicação projetado para facilitar a troca de mensagens entre dispositivos em uma rede, comumente utilizado em cenários de Internet das Coisas (IoT). Ele opera com base nos princípios do modelo de publicação/assinatura, que difere do modelo cliente-servidor tradicional.

Funcionamento Geral do MQTT:

Conexão:
Um cliente MQTT se conecta a um intermediário chamado Broker.
Após a conexão, o cliente pode publicar mensagens, assinar tópicos ou fazer ambas as operações.

Publicação:
Clientes publicam mensagens contendo um tópico e dados formatados em bytes.
Por exemplo, uma lâmpada inteligente pode publicar "on" no tópico "livingroom/light".

Assinatura:
Clientes enviam mensagens SUBSCRIBE ao Broker para receber mensagens sobre tópicos específicos.
Por exemplo, um aplicativo de casa inteligente assina o tópico "light" para contar quantas luzes estão acesas.

Broker:
O Broker atua como intermediário, filtrando e distribuindo mensagens aos assinantes apropriados.

Tópicos do MQTT:
Tópicos são palavras-chave hierárquicas usadas pelo Broker para filtrar mensagens.
Organizados de forma semelhante a diretórios em um sistema de arquivos.

Exemplo:
"livingroom/light" em um sistema de casa inteligente de vários andares.

Qualidades de Serviço (QoS) no MQTT:

QoS 0 - No máximo uma vez:
Mensagem enviada apenas uma vez, sem feedback.
Rápido, mas menos seguro, pois a mensagem pode ser perdida.

QoS 1 - Pelo menos uma vez:
Mensagem entregue pelo menos uma vez com feedback (PUBACK).
Pode resultar em envios repetidos até receber feedback.

QoS 2 - Exatamente uma vez:
Mensagem entregue exatamente uma vez com um processo de quatro partes (PUBLISH, PUBREC, PUBREL, PUBCOMP).
Mais seguro, mas envolve armazenamento temporário da mensagem.

O MQTT proporciona flexibilidade e eficiência na comunicação entre dispositivos, sendo especialmente útil em cenários onde o desacoplamento entre emissores e receptores é essencial, como na Internet das Coisas.

2.3. ENDEREÇO IP (Internet Protocol)

Definição Geral:
Um código numérico atribuído a cada dispositivo conectado a uma rede.
Funciona como um identificador exclusivo para cada dispositivo.

Unicidade:
Cada dispositivo em uma rede possui um endereço IP único.
Isso permite a identificação e comunicação entre dispositivos na rede.

Tipos de Endereços IP:

IP Interno (LAN):
Cada dispositivo em uma rede local (LAN) possui um endereço IP interno.
Exemplo: Se uma rede doméstica tem cinco dispositivos, cada um terá um IP interno único na rede local.

IP Externo (WAN):
O dispositivo também pode ter um IP externo, atribuído pela operadora de Internet.
Esse IP é visível para a Internet e funciona como o endereço da rede quando o dispositivo se comunica fora da rede local.

Dualidade de IPs:

Rede Interna:
Cada dispositivo em uma rede local (como sua casa) tem um endereço IP interno.
Esses IPs são atribuídos localmente pelo roteador da rede doméstica.

Rede Externa (Internet):
O roteador da casa, por sua vez, tem um IP externo fornecido pela operadora de Internet.
Este IP é o endereço da rede quando os dispositivos se comunicam fora da rede local.

Importância:

Identificação:
Os endereços IP são essenciais para identificar e rotear dados entre dispositivos em uma rede.

Comunicação:
Facilitam a comunicação entre dispositivos dentro da rede local e permitem a conexão à Internet.

Em resumo, o endereço IP é fundamental para a comunicação em redes, proporcionando a cada dispositivo uma identificação única que facilita o roteamento eficiente de dados. A dualidade de IPs (interno e externo) é especialmente relevante em ambientes domésticos, onde a rede interna é conectada à Internet por meio de um roteador.

Quais são as diferenças entre IP dinâmico e IP estático/fixo?

IP Dinâmico:

Atribuição Automática:
Os endereços IP dinâmicos são atribuídos automaticamente por um servidor DHCP (Dynamic Host Configuration Protocol).
Quando um dispositivo se conecta à rede, o DHCP atribui um endereço IP disponível temporário.

Alteração Periódica:
Os dispositivos podem receber endereços IP diferentes cada vez que se conectam à rede.
O endereço IP é geralmente temporário e pode mudar ao longo do tempo.

Facilidade de Gerenciamento:
Facilita a administração de uma grande quantidade de dispositivos em uma rede, pois os endereços são alocados dinamicamente conforme necessário.

Ideal para Redes Dinâmicas:
Recomendado para ambientes onde a configuração dos dispositivos pode mudar com frequência, como em redes domésticas e ambientes corporativos.

IP Estático/Fixo:

Atribuição Manual:
Os endereços IP estáticos são configurados manualmente, geralmente pelo administrador da rede.
Cada dispositivo tem um endereço IP fixo que não muda, a menos que seja reconfigurado manualmente.

Estabilidade de Endereço:
O endereço IP permanece constante, independentemente de quantas vezes o dispositivo se conecta ou reconecta à rede.

Controle Preciso:
Oferece maior controle sobre a atribuição de endereços IP, pois cada dispositivo é designado com um endereço específico.

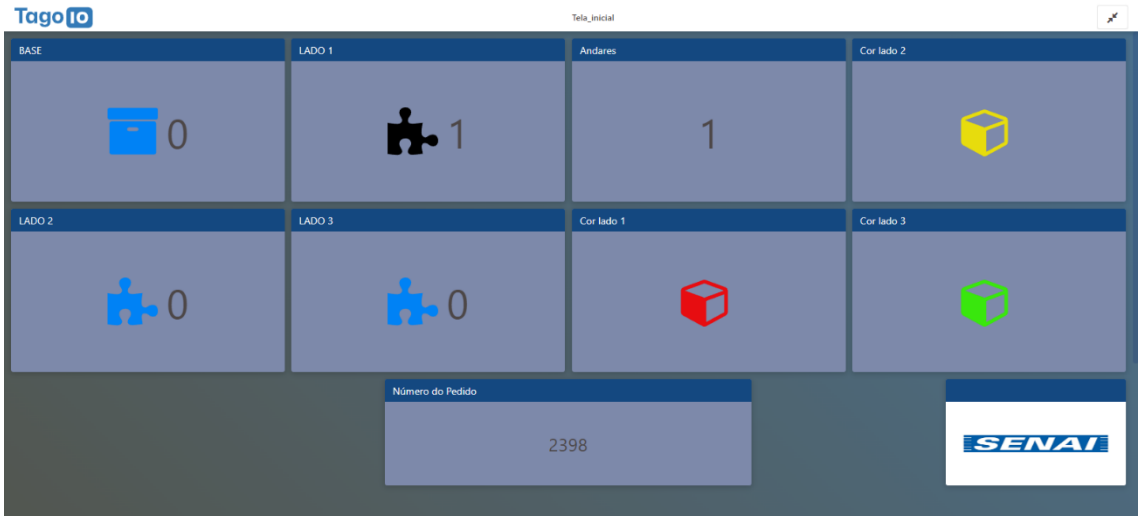
Recomendado para Servidores e Dispositivos Críticos:
É comum em servidores, roteadores, impressoras e outros dispositivos que precisam ter um endereço IP fixo para fácil identificação e acesso constante.

3. DASHBOARD

O dashboard é um painel de informações que facilita a interpretação e o acompanhamento de indicadores importantes.

Há alguns benefícios na utilização de uma dashboard, como:

- Atualização e acompanhamento dos indicadores;
- Centralização das informações;
- Identificação de padrões;
- Facilidade de comunicação;
- Otimização;



4. DESCRIÇÃO GERAL DO SISTEMA DESENVOLVIDO

Tendo como área de trabalho o modulo de montagem do sistema Smart 4.0 o objetivo foi através da implementação de um sistema de sensores externa, analisar o correto posicionamento das lâminas na base, e possibilitando a visualização através das dashboard, via web.

- **Uma placa ESP32:**
  O com código desenvolvido no IDE do Arduino, linguagem C++. Recebendo as informações dos sensores de presença infra vermelho, e tendo como saída o envio da leitura para o sistema Taigo.IO. A placa ESP32 é alimentação via USB. E é um microcontrolador usado em diversos projetos de IoT (Internet das Coisas), robótica, automação residencial e outros projetos que envolvem conexão com a internet. Ele é composto basicamente por um processador Xtensa Dual-Core de 32 bits, uma porta micro-USB para alimentação e programação, e um conversor USB serial integrado, além de já possuir WiFi nativo;

- **Bibliotecas ESP32:**
  Arduino json, espmqttclient e arduino esp32board;

- **Módulo sensor Lm393 de o obstáculo infravermelho:**

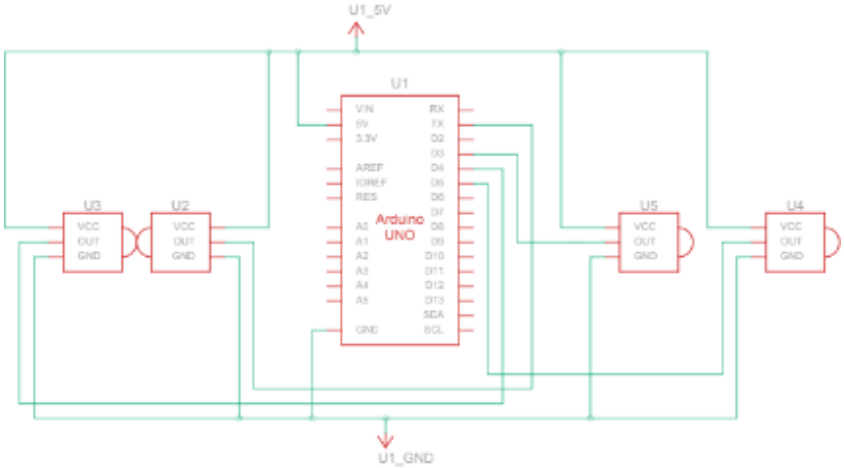
- **Uma protoboard** de 400 pontos interligadas, que servirá como matriz de contatos para a construção dos circuitos;

- **Um cabo de alimentação USB**, para alimentação do sistema e comunicação do NodeMCU com o computador;

- **Cabos jumper** macho-macho para fazer as conexões entre os componentes eletrônicos na protoboard;

- **Dois resistores** que servirão para ajudar na corrente que ligará os LED's;

4.1. ESQUEMA ELÉTRICO ARDUINO



Na figura acima podemos ver o esquema elétrico dos sensores e Arduino:

- **VCC** fornece energia para o módulo. Deve ser conectado no pino 5V da placa de desenvolvimento utilizada;
- **GND** é o pino de aterramento / neutro, e precisa ser conectado ao pino GND da placa;
- **D23 D26 D32 D33** pinos que recebem a leitura digital dos sensores infravermelhos;

4.2. DIAGRAMAS

DIAGRAMA DE CLASSE

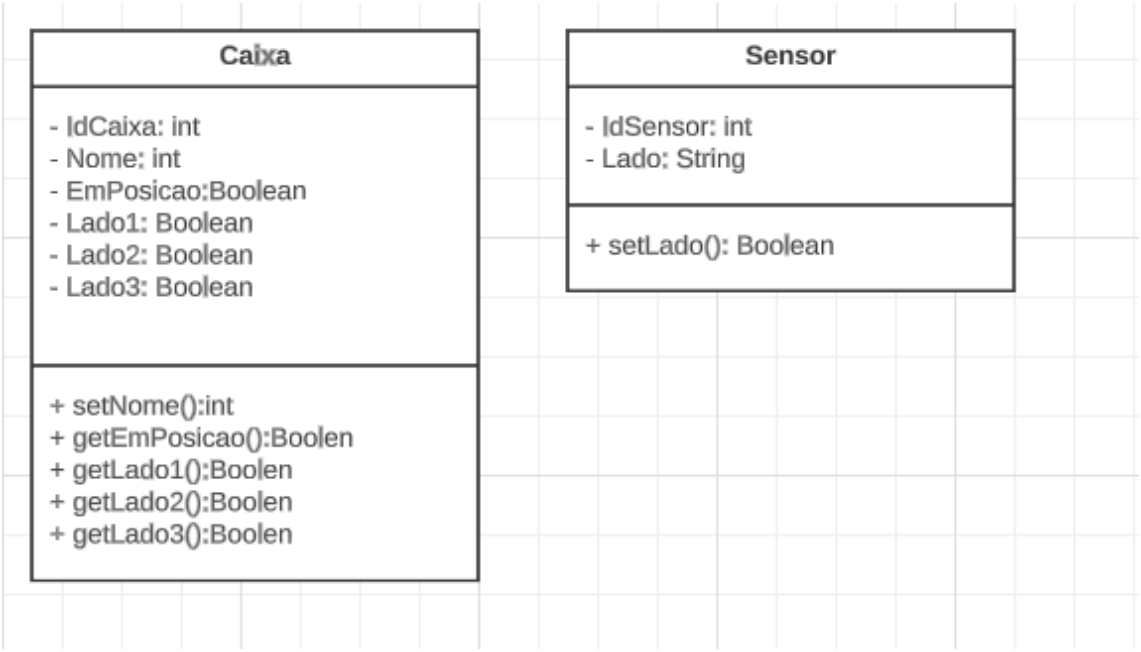
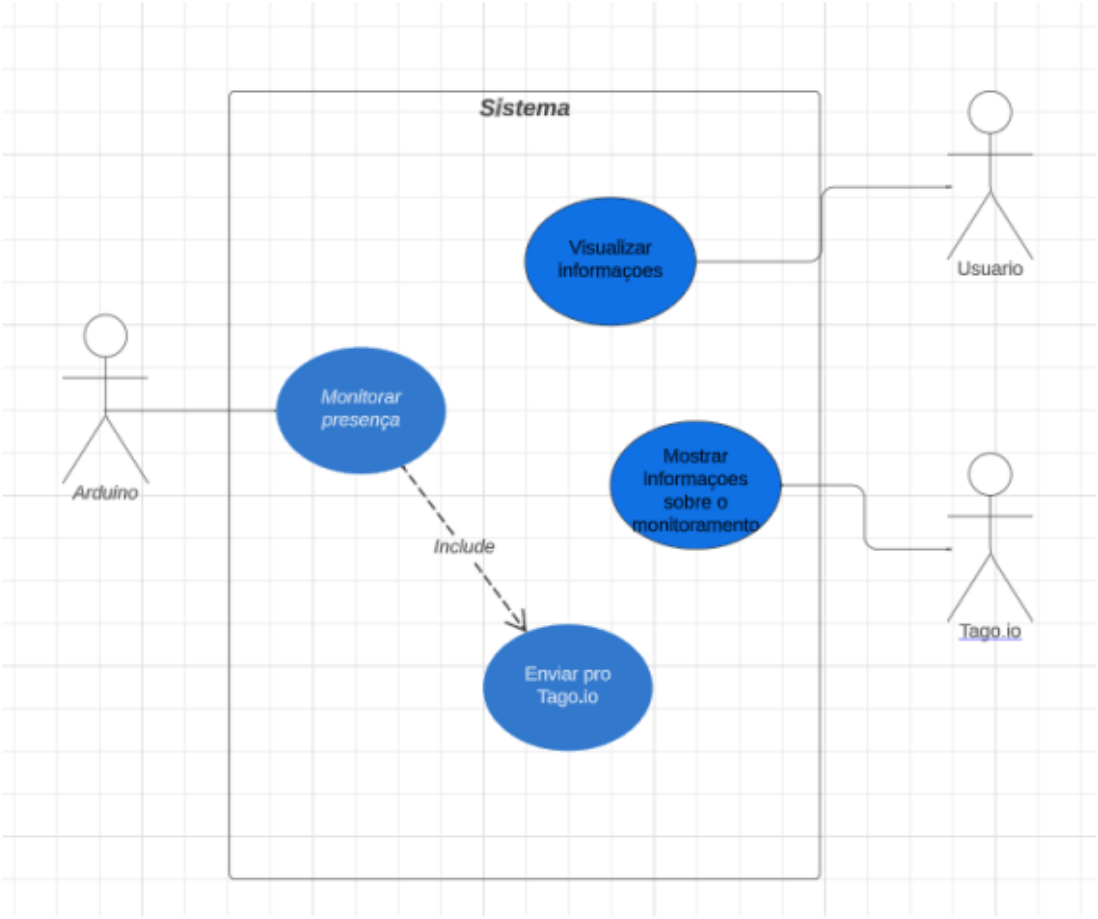


DIAGRAMA DE CASO DE USO



Cenário típico:

1 - Começa quando os sensores leem os sinais;

2 - O ESP32 capta esses sinais e envia para uma plataforma; 3 - A plataforma Tago.io exibe os valores em uma dashboard; 4 - O ESP32 verifica se o sinal 0 (não tem local) ou 1 (possui local);

Fluxo alternativo:

1 - No passo 4, caso o sinal esteja dentro do esperado, recebemos um sinal no Tago.io demonstrando a presença ou não da parede da caixa;

2 - O Tago.io sempre mostra as variáveis existentes porem só altera valores com o recebimento de informação através do sensor, sendo feito a conferencia a cada segundo;

DIAGRAMA DE ESTADOS

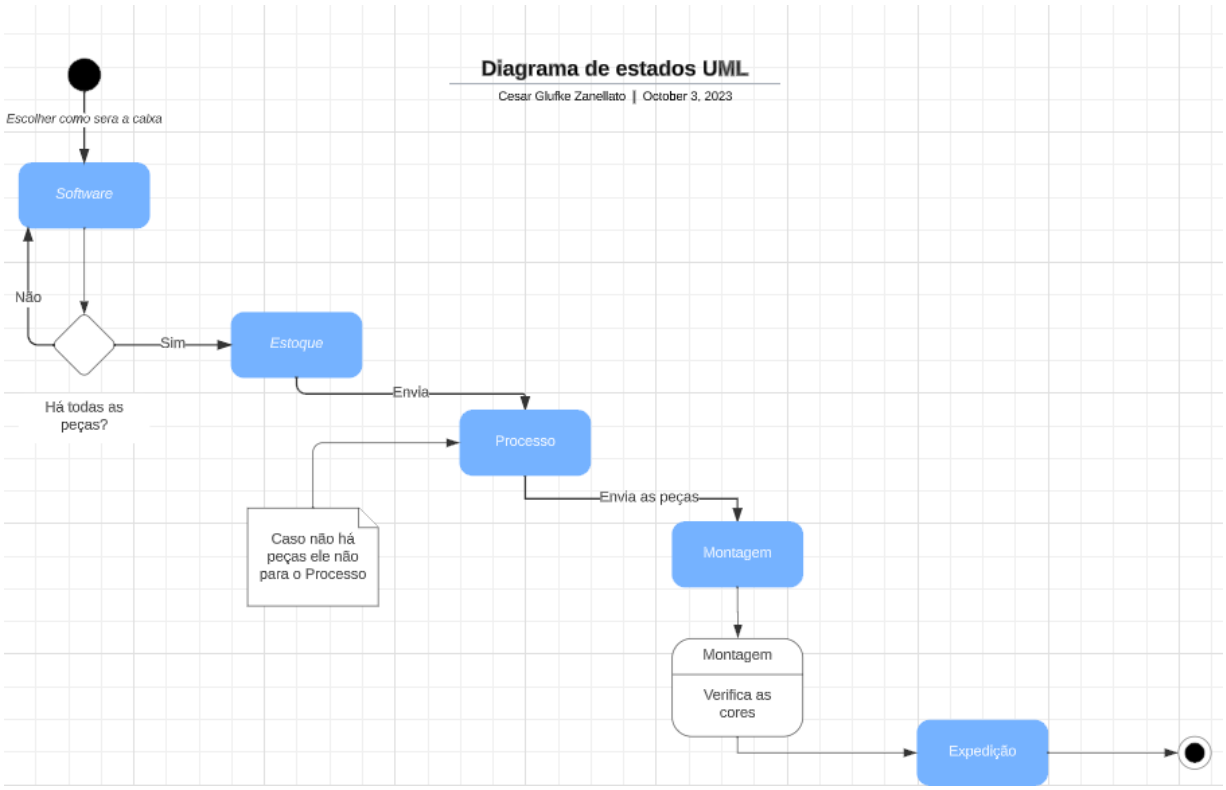


DIAGRAMA DE SEQUÊNCIA

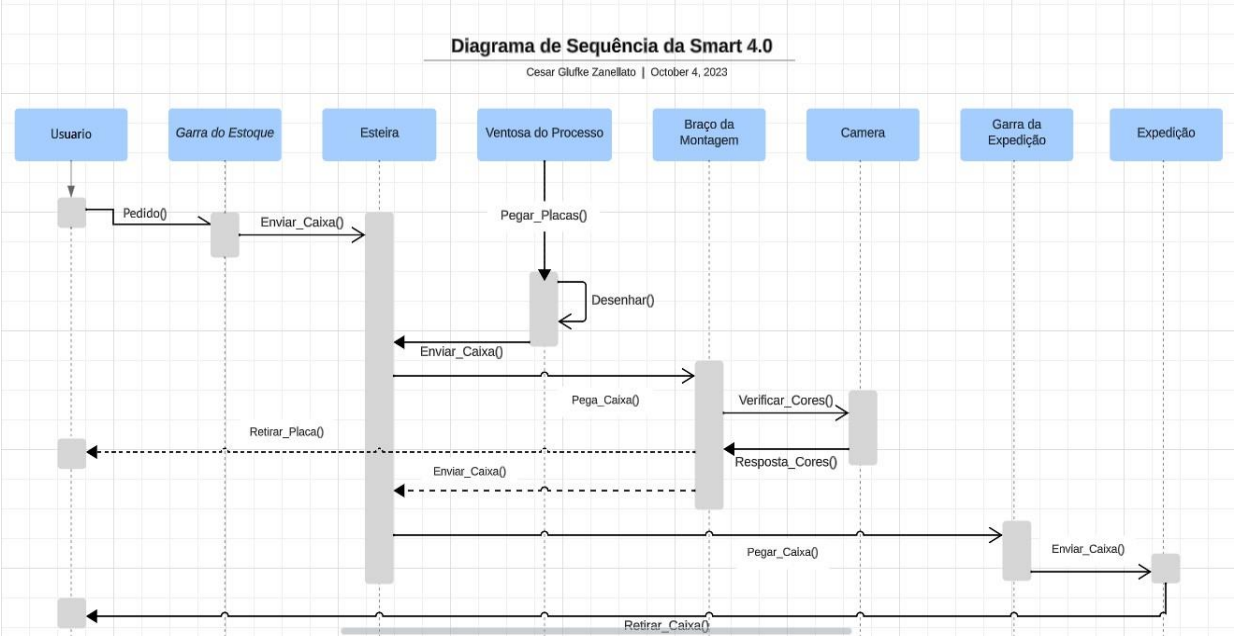
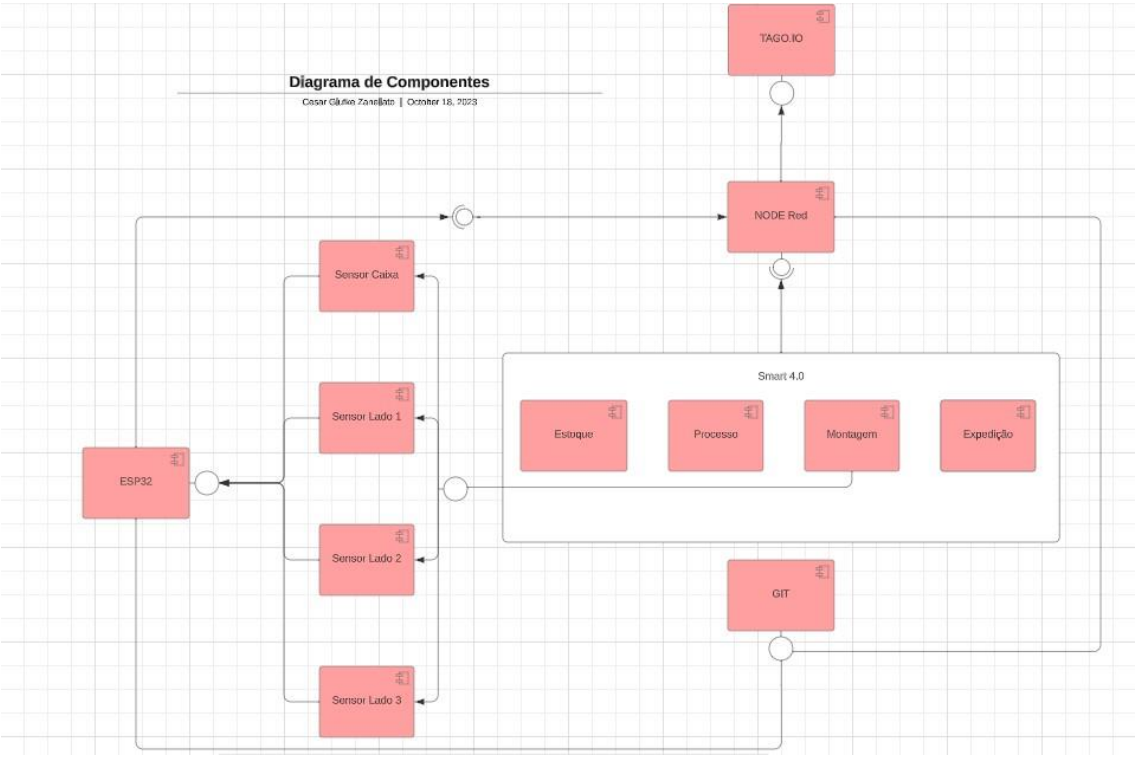


DIAGRAMA DE COMPONENTES



5. REQUESITOS DO SISTEMA

REQUISITOS FUNCIONAIS

Verificar a existência da lâmina no local e da caixa principal, possibilitando identificar se ela está no local ou não.

REQUISITOS NÃO FUNCIONAIS

Verificar a distância da lâmina e o sensor, possibilitando verificar se esta dentro ou fora da caixa.

6. RESULTADOS

```
//exemplo base
//referência: https://raw.githubusercontent.com/vinicius0082/esp32-
tagoIO/main/exemplos/mpu_esp32_mqtt_tagoIO

#include <ArduinoJson.h>
#include "EspMQTTClient.h"

//pinos de entrada e saída
const int pino_botao = 25; //input
const int pino_lado1 = 33; //input
const int pino_lado2 = 32; //input
const int pino_lado3 = 35; //input
const int pino_led = 26; //output

//variáveis para Json
char json_btn[100];
char json_lado1[100];
char json_lado2[100];
char json_lado3[100];

//variáveis internas
int valor_btn;
int valor_lado1;
int valor_lado2;
int valor_lado3;

//configurações da conexão MQTT
EspMQTTClient client
(
  "FIESC_IOT", //nome da sua rede Wi-Fi
  "C6qnM4ag81", //senha da sua rede Wi-Fi
  "mqtt.tago.io",  // MQTT Broker server ip padrão da tago
  "Token",   // username
  "19a56446-0f07-46fa-ad11-91c6e97c6f60",   // Código do Token
  "TestClient",     // Client name that uniquely identify your device
  1883              // The MQTT port, default to 1883. this line can be
omitted
);

//configuração dos pinos
void setup()
{
```

```
  Serial.begin(115200);
  //configura os pinos como entrada
  pinMode(pino_botao, INPUT);
  pinMode(pino_lado1, INPUT);
  pinMode(pino_lado2, INPUT);
  pinMode(pino_lado3, INPUT);
  pinMode(pino_led, OUTPUT);

  //ativa pull-up no pino de entrada
  digitalWrite(pino_botao, HIGH);
  digitalWrite(pino_lado1, HIGH);
  digitalWrite(pino_lado2, HIGH);
  digitalWrite(pino_lado3, HIGH);
}

void leitura_sinais()
{
  valor_btn = digitalRead(pino_botao);
  valor_lado1 = digitalRead(pino_lado1);
  valor_lado2 = digitalRead(pino_lado2);
  valor_lado3 = digitalRead(pino_lado3);
}

void converte_json()
{
  StaticJsonDocument<300> sjson_btn;

  sjson_btn["variable"] = "botao";
  sjson_btn["value"] = valor_btn;
  serializeJson(sjson_btn, json_btn);

  StaticJsonDocument<300> sjson_lado1;

  sjson_lado1["variable"] = "lado1";
  sjson_lado1["value"] = valor_lado1;
  serializeJson(sjson_lado1, json_lado1);

  StaticJsonDocument<300> sjson_lado2;

  sjson_lado2["variable"] = "lado2";
  sjson_lado2["value"] = valor_lado2;
  serializeJson(sjson_lado2, json_lado2);

  StaticJsonDocument<300> sjson_lado3;

  sjson_lado3["variable"] = "lado3";
  sjson_lado3["value"] = valor_lado3;
```

```
  serializeJson(sjson_lado3, json_lado3);
}

void envia_msg()
{ // Você pode ativar a sinalização de retenção definindo o terceiro
parâmetro como true
  client.publish("node/btn", json_btn);
  client.publish("node/btn", json_lado1);
  client.publish("node/lado2e3", json_lado2);
  client.publish("node/lado2e3", json_lado3);
}

//loop do programa
void loop()
{
  leitura_sinais();
  converte_json();
  envia_msg();

  delay(1000);

  client.loop();
}

void processa_msg(const String payload)
{
  StaticJsonDocument<300> msg;

  DeserializationError err = deserializeJson(msg, payload);
  if (err) {
    Serial.print(F("deserializeJson() failed with code "));
    Serial.println(err.f_str());
  }
  Serial.print("var:");
  String var = msg["variable"];
  Serial.println(var);
  if(var == "status")
  {
    Serial.print("value:");
    String val = msg["value"];
    Serial.println(val);
    if(val == "on")
      digitalWrite(pino_led, LOW);
    else
      digitalWrite(pino_led, HIGH);
  }
}
```

```
// Esta função é chamada uma vez que tudo está conectado (Wifi e MQTT)
// ATENÇÃO: VOCÊ DEVE IMPLEMENTÁ-LA SE ESTIVER USANDO EspMQTTClient
void onConnectionEstablished()
{
   client.subscribe("node/status", [] (const String &payload)  {
   Serial.println(payload);
   processa_msg(payload);
  });
}
```

7. REFERÊNCIAS

Código: https://raw.githubusercontent.com/vinicius0082/esp32-tagoIO/main/exemplos/mpu_esp32_mqtt_tagoIO