# PROJETO INTEGRADOR: CRUZAMENTO 4.0

## FASE 1: LEVANTAMENTO DE REQUISITOS

### 1.1. Visão Geral

O objetivo deste projeto é desenvolver um semáforo inteligente capaz de adaptar seus tempos de sinalização com base em condições climáticas (chuva) e fluxo de veículos, utilizando sensores e comunicação via protocolo MQTT.

### 1.2. Requisitos Funcionais (RF)

* **(Must Have) [RF001] Controle de Tempo por Fluxo:** O sistema deve detectar alta densidade de veículos e aumentar o tempo do sinal verde para melhorar o fluxo.
* **(Must Have) [RF002] Monitoramento Climático:** O sistema deve identificar chuva através de sensores de umidade.
* **(Must Have) [RF003] Adaptação à Chuva:** Em caso de chuva forte detectada, o tempo do sinal amarelo deve ser aumentado para garantir maior segurança na frenagem.
* **(Must Have) [RF004] Modo de Segurança:** Caso haja falha nos sensores, o semáforo deve entrar automaticamente no modo "pisca amarelo" (intermitente).
* **(Must Have) [RF005] Operação Local (Contingência):** O algoritmo deve continuar funcionando localmente mesmo se a conexão com a internet ou com o Broker MQTT cair.
* **(Should Have) [RF006] Armazenamento de Dados:** O sistema deve armazenar histórico de fluxo em vetores/matrizes para análise.
* **(Must Have) [RF007] Interface de Saída:** Simulação dos LEDs através de saídas visuais no console ou dashboard.

### 1.3. Requisitos Não Funcionais (RNF)

* **(Must Have) [RNF001] Conectividade:** Uso do protocolo MQTT (porta 1883) ou TCP/IP.
* **(Must Have) [RNF002] Hardware:** Compatibilidade com microcontroladores ESP32 ou Arduino.
* **(Should Have) [RNF003] Código Limpo:** O código deve seguir padrões de *Clean Code* (nomes claros, funções pequenas).
* **(Must Have) [RNF004] Segurança de Rede:** Implementação de Firewall bloqueando portas não utilizadas.
* **(Should Have) [RNF005] Controle de Acesso:** Diferenciação de usuários (Admin vs. Operador) com senhas fortes.

## FASE 2: ARQUITETURA DE REDES E IOT

### 2.1. Diagrama da Arquitetura de Rede

Diagrama

O conteúdo gerado por IA pode estar incorreto.

### 2.2. Lista de Equipamentos e Justificativa

Para garantir a comunicação entre os sensores, o controlador (ESP32) e o servidor, foram definidos os seguintes equipamentos:

### 1. Access Point (Ponto de Acesso)

* **Função:** Criar a rede sem fio (Wi-Fi) para conectar os dispositivos móveis e IoT.
* **Justificativa no Projeto:** Essencial para permitir a conexão do controlador **ESP32** e dos sensores sem fio à rede principal. Como o semáforo fica na rua (ambiente externo), o Access Point atua como a "ponte" que converte os sinais de rádio do ESP32 para a rede cabeada que vai até o servidor.

### 2. Switch

* **Função:** Interconectar dispositivos cabeados dentro de uma mesma rede local (LAN), encaminhando dados para o destino correto usando endereços MAC.
* **Justificativa no Projeto:** Necessário para conectar fisicamente o *Access Point*, o *Servidor Local* e eventuais terminais de manutenção, garantindo estabilidade e alta velocidade na troca de dados interna, sem depender do Wi-Fi para o tráfego pesado da rede.

### 3. Roteador

* **Função:** Gerenciar o tráfego entre diferentes redes (ex: LAN e WAN) e atribuir endereços IP (via DHCP) aos dispositivos.
* **Justificativa no Projeto:** É o responsável por organizar a rede lógica, garantindo que o ESP32 e o Servidor tenham IPs válidos para se comunicarem. Além disso, permite que o sistema tenha uma saída para a internet (se necessário para atualizações ou relatórios remotos) ou segmente a rede do semáforo da rede pública da prefeitura.

### 4. Firewall

* **Função:** Filtrar o tráfego de entrada e saída com base em regras de segurança, bloqueando acessos não autorizados.
* **Justificativa no Projeto:** Crítico para a segurança do sistema, pois o cenário original menciona "falhas de segurança". O firewall deve ser configurado para bloquear todas as conexões suspeitas e permitir **apenas** o tráfego necessário, como a porta **1883** (protocolo MQTT) para a comunicação dos sensores, impedindo ataques externos ao controlador do semáforo.

### 5. Servidor Local

* **Função:** Computador ou hardware dedicado que hospeda as aplicações, banco de dados e serviços de processamento central.
* **Justificativa no Projeto:** Peça fundamental para cumprir o requisito de que o "algoritmo deve rodar localmente se a internet cair". Ele hospedará o **Broker MQTT** (que recebe os dados dos sensores), o banco de dados de tráfego e o próprio algoritmo de controle do semáforo, garantindo que o cruzamento continue funcionando mesmo sem conexão externa.

## FASE 3: SISTEMA OPERACIONAL E SEGURANÇA (PSI)

### 3.1. Escolha do Sistema Operacional

## 3.1.1. Comparativo de Sistemas Operacionais

Abaixo apresentamos a análise comparativa entre as duas principais opções de mercado para hospedar o servidor local do semáforo, considerando os critérios de custo, segurança e aderência ao ecossistema de Internet das Coisas (IoT).

|  |  |  |
| --- | --- | --- |
| **Item** | **Windows Server** | **Ubuntu Server** |
| **Custo** | **Alto.** Exige pagamento de licenças (por núcleo) e CALs de acesso de cliente. Custo recorrente elevado para um projeto público/MVP. | **Baixo / Gratuito.** Open Source (código aberto). Não há custo de licença para uso, o que reduz drasticamente o orçamento do projeto. |
| **Segurança** | **Média/Alta.** Possui boas ferramentas, mas é o alvo mais frequente de ataques e malwares. Exige atualizações frequentes que muitas vezes forçam reinicializações do sistema. | **Muito Alta.** Arquitetura baseada em permissões restritas (root). Menor superfície de ataque para vírus comuns. Atualizações de kernel (Livepatch) muitas vezes não exigem reinicialização. |
| **Suporte a IoT** | **Bom.** Focado em integração com nuvem (Azure IoT). Pode ser "pesado" para rodar brokers MQTT simples em hardware modesto. | **Excelente (Nativo).** É o padrão da indústria para gateways IoT. Suporte nativo e leve para ferramentas como Docker, Mosquitto MQTT, Node-RED e Python. |

## 3.1.2. Justificativa da Escolha

**Sistema Operacional Escolhido:** **Ubuntu Server (Linux)**

**Justificativa Técnica:**

A equipe optou pelo **Ubuntu Server** como a plataforma ideal para o servidor local do projeto "Cruzamento 4.0" pelos seguintes motivos estratégicos:

1. **Viabilidade Econômica (Custo):** Como se trata de um projeto para a prefeitura (serviço público) com foco em um MVP (Produto Mínimo Viável), a eliminação de custos com licenças de software permite investir mais em hardware (sensores e controladores) e redundância.
2. **Estabilidade e Desempenho:** O algoritmo do semáforo precisa rodar 24/7 sem interrupções. O Linux gerencia melhor os recursos de hardware (memória e processador) e não força reinicializações automáticas para atualizações, garantindo maior disponibilidade do serviço (SLA).
3. **Ecossistema IoT:** O Ubuntu Server possui integração nativa e facilitada com o protocolo **MQTT** (através do broker *Mosquitto*) e com a linguagem do algoritmo (Python ou C++), simplificando a configuração do ambiente de desenvolvimento e produção.
4. **Segurança:** A facilidade em configurar um Firewall (UFW) via terminal para bloquear todas as portas e liberar apenas a **1883 (MQTT)** e **22 (SSH)** torna o sistema extremamente robusto contra acessos não autorizados.

### 3.2. Configuração do Ambiente em Laboratório

Para garantir a organização e segurança no servidor Linux:

* **Gerenciamento de Usuários:** Foram criados usuários distintos.
  + admin: Acesso total (root/sudo) para configurações do sistema.
  + operador: Acesso restrito apenas para visualização de logs e status do semáforo.
* **Permissões de Arquivos:** Utilização do comando chmod para garantir que apenas o admin possa alterar os scripts de configuração do semáforo, enquanto o operador tem permissão apenas de leitura.

### 3.3. Política de Segurança da Informação (PSI)

Regras definidas para mitigar riscos no projeto Cruzamento 4.0:

1. **Controle de Acesso e Senhas:**
   * Todas as senhas devem ter no mínimo **8 caracteres**, combinando letras maiúsculas, minúsculas e números.
   * A troca de senhas deve ser feita a cada 90 dias.
   * Bloqueio de conta após 5 tentativas falhas de login.
2. **Backup e Recuperação:**
   * **Backup Diário:** Incremental (apenas arquivos modificados).
   * **Backup Semanal:** Completo (imagem do sistema/banco de dados).
   * Os backups devem ser armazenados em local externo ao servidor principal.
3. **Segurança Física e Lógica:**
   * Bloqueio de tela automático nas estações de trabalho após 5 minutos de inatividade.
   * Uso de Firewall configurado para negar todo tráfego de entrada, exceto a porta 1883 (MQTT) e 22 (SSH).
4. **Plano de Contingência:**
   * Se o sensor falhar, o sistema assume o "Modo Seguro" (amarelo intermitente).
   * Se a internet cair, o sistema opera em "Modo Local".

## FASE 4: ALGORITMO E LÓGICA DE PROGRAMAÇÃO

### 4.1. Fluxograma de Funcionamento

O algoritmo segue a lógica de leitura contínua de sensores para tomada de decisão.

### Diagrama O conteúdo gerado por IA pode estar incorreto.

### 4.2. Descrição da Lógica

1. **Início:** Inicialização das variáveis e conexão com o Wi-Fi/Broker.
2. **Leitura:** O sistema lê os dados dos sensores (Presença/Fluxo e Sensor de Chuva).
3. **Decisão de Chuva:**
   * Se Chuva == Verdadeiro: O tempo do sinal amarelo é incrementado (ex: de 3s para 5s) para evitar derrapagens.
4. **Decisão de Fluxo:**
   * Se Fluxo == Alto (muitos carros): O tempo do sinal verde da via principal é estendido.
   * Se Fluxo == Baixo: Mantém o tempo padrão.
5. **Verificação de Erros:**
   * A cada ciclo, verifica-se a integridade dos sensores. Se houver erro de leitura (null ou valores impossíveis), aciona o **Modo de Segurança** (Pisca Amarelo) e envia alerta ao servidor.
6. **Atuação:** Envia o comando para os LEDs (Vermelho, Amarelo, Verde) conforme o estado calculado.