UNIVERSIDADE FEDERAL DE SANTA CATARINA
Centro de Ciências, Tecnologias e Saúde - CTS

TESTE DE ENVIO DE ARQUIVOS EM DIFERENTES REDES SEM FIO

ESTUDO PRELIMINAR

DISCIPLINA
DEC7563 - Redes sem Fios

DOCENTE
Roberto Rodrigues Filho, Ph.D.

DISCENTES
Jose Norberto Fagundes Isaias (19202785)
Vinícius Souza Capistrano (18204884)

SUMÁRIO

SUMÁRIO **2**
INTRODUÇÃO **3**
Objetivo Geral 3
Objetivos Específicos 3
Justificativa 4
Prazo Estimado 4
Benefícios Esperados 4
PREMISSAS **5**
RESTRIÇÕES **6**
ANÁLISE E GESTÃO DE RISCOS **7**
ESCOPO **8**
Configuração dos Dispositivos 8
Cenários de Teste - Ambiente Controlado 8
Tipos e Tamanhos de Arquivos 8
PROCEDIMENTO DE ENVIO DE ARQUIVOS **9**
ESTRUTURA DO CÓDIGO **10**
Código do Cliente Wi-Fi (Remetente): 11
Código do Cliente Wi-Fi (Receptor): 12
Código do Cliente Bluetooth (Remetente): 13
Código do Cliente Bluetooth (Receptor): 14
ENCERRAMENTO **15**

INTRODUÇÃO

Com a crescente demanda por conectividade e a expansão do uso de dispositivos móveis e IoT, o desenvolvimento de sistemas de comunicação sem fio robustos e eficientes tornou-se essencial. No entanto, o planejamento e a implementação de redes sem fio eficazes nem sempre acompanharam a rápida evolução tecnológica. O surgimento de plataformas como o ESP32 e Arduino, capazes de operar com diferentes tecnologias como Wi-Fi e Bluetooth, destaca a necessidade de uma análise comparativa detalhada dessas tecnologias sob diversas condições de uso.

Na prática, a transferência de arquivos em ambientes sem fio pode ser desafiadora, enfrentando problemas de latência, taxa de erro e limitações de alcance, especialmente em ambientes urbanos densos onde o espectro de RF é frequentemente saturado. Neste contexto, nosso software é desenvolvido com o objetivo de maximizar a eficiência das transferências de arquivos através de diferentes tecnologias sem fio, permitindo uma avaliação precisa da viabilidade técnica e prática de cada uma delas em variadas condições ambientais.

Este projeto pretende fornecer insights valiosos sobre as capacidades e limitações de Wi-Fi e Bluetooth em contextos práticos de uso, utilizando dispositivos de hardware comuns e acessíveis. Com esse estudo, esperamos identificar a tecnologia mais adequada para diferentes tipos de aplicação, baseados na eficiência, confiabilidade e praticidade em cenários de uso real.

Objetivo Geral

Desenvolver uma solução robusta para avaliar e comparar a performance de diferentes tecnologias de comunicação sem fio (Wi-Fi e Bluetooth) no envio de arquivos em variadas condições ambientais. Esse estudo visa identificar a tecnologia mais eficiente e confiável para a transferência de arquivos em ambientes urbanos e rurais, contribuindo para melhorar a conectividade e a comunicação em dispositivos móveis e IoT.

Objetivos Específicos

Os objetivos específicos deste projeto são detalhados a seguir:

- **Desenvolvimento de um sistema de avaliação:** Criar uma aplicação integrada utilizando ESP32 para Wi-Fi e Bluetooth, que permita o envio e recebimento de arquivos de diversos formatos e tamanhos sob diferentes condições de teste.

- **Análise comparativa das tecnologias:** Comparar as taxas de transferência, a latência e as taxas de erro das diferentes tecnologias em diversos ambientes, para determinar qual tecnologia é mais adequada para aplicativos específicos baseados em eficiência e confiabilidade.

Justificativa

Com o avanço contínuo da tecnologia digital e o aumento da interconectividade impulsionado pela Internet das Coisas (IoT), a necessidade de transferir dados de forma rápida e confiável entre dispositivos torna-se crucial. Em um mundo onde a quantidade de dados gerados e consumidos cresce exponencialmente, é essencial desenvolver sistemas de comunicação que possam lidar com essas demandas de forma eficiente. Além disso, a diversidade de ambientes de rede e a variação das condições de uso exigem soluções adaptáveis e robustas, fazendo deste projeto uma resposta direta a essas necessidades.

Prazo Estimado

O prazo estimado para o desenvolvimento completo e a finalização deste protótipo é de 10 (dez) semanas. Esse período inclui planejamento, desenvolvimento, testes intensivos e análise dos resultados.

Benefícios Esperados

A implementação deste protótipo espera trazer uma série de melhorias significativas na eficiência das comunicações sem fio. Os benefícios incluem:

- **Melhoria na Eficiência de Transmissão de Dados:** Identificação das soluções mais eficazes, reduzindo a latência e as taxas de erro em transferências de arquivos.
- **Adaptação às Diferentes Condições Ambientais:** Desenvolvimento de sistemas mais resilientes e adaptáveis, capazes de operar eficientemente em ambientes urbanos densos e outras condições desafiadoras.
- **Contribuição para o Avanço da IoT:** Melhoria da confiabilidade das redes sem fio, contribuindo para o crescimento e a eficácia da Internet das Coisas.

Esses avanços são fundamentais não apenas para melhorar a experiência do usuário final, mas também para apoiar o desenvolvimento de novas aplicações e serviços em um ambiente urbano cada vez mais digitalizado e conectado.

PREMISSAS

As premissas para o projeto estão relacionadas ao desenvolvimento e implementação tanto do software de teste quanto da infraestrutura de rede necessária, sendo elas:

- **Compatibilidade de Tecnologias:** O sistema será capaz de operar com Wi-Fi e Bluetooth utilizando plataformas como ESP32.
- **Dispositivos de Teste:** Todos os dispositivos utilizados para testar a transferência de arquivos deverão ter capacidades de comunicação compatíveis com as tecnologias testadas**.**
- **Presença de Especialistas:** Durante os testes, técnicos especializados estarão disponíveis para resolver qualquer problema técnico que possa surgir.
- **Condições Ambientais Controladas:** Os testes serão realizados em uma variedade de ambientes para garantir resultados consistentes e confiáveis.
- **Capacitação Técnica:** Os usuários envolvidos nos testes deverão ter conhecimento técnico suficiente para operar os sistemas e analisar os resultados.

RESTRIÇÕES

Para o funcionamento completo do sistema de teste, as restrições do projeto estão relatadas abaixo:

- **Infraestrutura de Hardware:** Serão necessários dispositivos específicos como ESP32, além de computadores para coleta e análise de dados.
- **Orçamento do Projeto:** Vamos utilizar os módulos de ESP32 disponíveis no laboratório, eliminando a necessidade de aquisição adicional de componentes de hardware e software.
- **Proteção de Dados:** Todos os dados coletados durante os testes devem ser acessíveis apenas aos gestores do projeto e protegidos conforme as normas de privacidade e segurança de dados.
- **Limitações Técnicas:** Devido à natureza experimental do projeto, algumas tecnologias de comunicação podem não alcançar a eficiência máxima esperada inicialmente.

ANÁLISE E GESTÃO DE RISCOS

Risco	Descrição	Impacto	Probabilidade	Mitigação	Plano de Contingência
Falhas de Hardware	Dispositivos ESP32 podem falhar durante os testes.	Interrupção dos testes, necessidade de substituição de componentes.	Média	- Testar todos os dispositivos antes dos testes principais. - Ter dispositivos de backup prontos para substituição.	Substituir imediatamente o dispositivo falho por um backup.
Bugs de Software	Erros no código podem causar falhas na comunicação.	Necessidade de correção de código, atrasos nos testes.	Alta	- Revisar e testar o código frequentemente. - Utilizar controle de versão (Git) para gerenciar o desenvolvimento.	Identificar e corrigir rapidamente o problema no código.
Fatores Ambientais	Interferências de sinal, obstáculos físicos e variações climáticas.	Resultados inconsistentes ou não representativos.	Média	- Realizar testes em ambientes controlados sempre que possível. - Documentar condições de teste detalhadamente.	Repetir os testes em um ambiente controlado se necessário.
Gestão do Tempo	Atrasos no cronograma devido a imprevistos.	Não cumprimento dos prazos.	Média	- Criar um cronograma detalhado com margens de segurança. - Realizar reuniões regulares para monitorar o progresso.	Reavaliar prioridades e ajustar o cronograma conforme necessário.

ESCOPO

O projeto focará no desenvolvimento de um sistema de avaliação para testar a performance das tecnologias Wi-Fi e Bluetooth usando dispositivos ESP32. A análise será baseada em códigos de teste executados na Arduino IDE, sem interfaces de usuário complexas além das necessárias para configurar e executar os testes.

Configuração dos Dispositivos

- **ESP32:** Dois ESP32 para cada tecnologia (Wi-Fi e Bluetooth). Um atuará como cliente (remetente) e outro como servidor (receptor).
- **Fonte de Alimentação:** Cada ESP32 deve estar conectado a uma fonte de alimentação estável.
- **Sistema de Arquivos:** Utilizar SPIFFS (SPI Flash File System) no ESP32 para armazenar os arquivos a serem enviados.

Cenários de Teste - Ambiente Controlado

- **Distância:** Configure os dispositivos ESP32 a diferentes distâncias (por exemplo, 1m, 5m, 10m) para testar a influência da distância na performance.
- **Obstáculos:** Realize testes em linha de visão direta e com obstáculos (paredes, móveis) para simular diferentes condições ambientais.
- **Ambiente Urbano e Rural:** Realize testes em áreas com alta densidade de dispositivos (urbano) e áreas com menor interferência de sinal (rural).

Tipos e Tamanhos de Arquivos

- **Pequeno**
  - Texto: 10 KB (ex: .txt)
  - Imagem: 100 KB (ex: .jpg, .png)
  - Áudio: 250 KB (ex: .mp3, .wav)

- **Médio**
  - Texto: 500 KB (ex: .pdf, .docx)
  - Imagem: 1 MB (ex: .jpg, .png)
  - Áudio: 2 MB (ex: .mp3, .wav)
  - Vídeo: 5 MB (ex: .mp4, .avi)

- **Grande**
  - Texto: 5 MB (ex: .pdf, .docx)
  - Imagem: 10 MB (ex: .jpg, .png)
  - Áudio: 10 MB (ex: .mp3, .wav)
  - Vídeo: 20 MB (ex: .mp4, .avi)

PROCEDIMENTO DE ENVIO DE ARQUIVOS

Preparação dos Dispositivos:

1. Carregue o código no ESP32 cliente e servidor.
2. Certifique-se de que ambos os dispositivos estão configurados corretamente para a rede Wi-Fi ou conexão Bluetooth.

Inicialização:

1. Inicie o servidor ESP32.
2. Conecte o cliente ESP32 ao servidor.

Envio de Arquivos:

1. No cliente, selecione o arquivo a ser enviado (pequeno, médio ou grande).
2. Inicie o envio do arquivo.
3. No servidor, receba o arquivo e salve no SPIFFS.

Coleta de Dados:

1. Registre o tempo de início e término da transferência para calcular a latência e a taxa de transferência.
2. Compare checksums ou hashes dos arquivos enviados e recebidos para calcular a taxa de erro.

Repetição do Teste:

1. Repita o processo para diferentes tamanhos de arquivos e diferentes distâncias.
2. Realize múltiplos testes para garantir consistência e confiabilidade dos dados.

Análise dos Resultados:

1. Compile os dados de todas as transferências.
2. Analise as métricas (taxa de transferência, latência, taxa de erro, consumo de energia, estabilidade da conexão).
3. Compare os resultados entre Wi-Fi e Bluetooth.

ESTRUTURA DO CÓDIGO

O projeto é dividido em duas partes principais: código para o cliente (remetente) e código para o servidor (receptor). Ambos os códigos são fornecidos para Wi-Fi e Bluetooth.

Código do Cliente:
O código do cliente, ou remetente, é responsável por iniciar a conexão com o servidor e enviar os arquivos armazenados no sistema de arquivos SPIFFS do ESP32. Ele estabelece a conexão (via Wi-Fi ou Bluetooth), lê o arquivo a ser enviado e o transmite ao servidor, enquanto mede o tempo total de transferência.

Código do Servidor:
O código do servidor, ou receptor, aguarda conexões de clientes e recebe os arquivos enviados. Ele se conecta à rede (Wi-Fi ou Bluetooth), escuta por dados de entrada e grava os dados recebidos no SPIFFS, registrando o tempo total de recepção. O servidor assegura que os dados sejam armazenados corretamente para análise posterior.

Códigos disponíveis em https://github.com/VCapis/esp32_wifi_bt.git .

Código do Cliente Wi-Fi (Remetente):

```
#include <WiFi.h>
#include <SPIFFS.h>

// Define as credenciais da rede Wi-Fi
const char* ssid = "ESP_WIFI";
const char* password = "12345678";

// Cria um servidor Wi-Fi na porta 80
WiFiServer server(80);

void setup() {
  Serial.begin(115200); // Inicializa a comunicação serial
  WiFi.begin(ssid, password); // Conecta-se à rede Wi-Fi

  // Aguarda a conexão com o Wi-Fi
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  server.begin(); // Inicia o servidor
  SPIFFS.begin(); // Inicializa o sistema de arquivos SPIFFS
}

void loop() {
  WiFiClient client = server.available(); // Verifica se há um cliente conectado
  if (client) {
    File file = SPIFFS.open("/receivedfile.txt", FILE_WRITE); // Abre um arquivo para escrita no SPIFFS
    if (file) {
      unsigned long startTime = millis(); // Marca o tempo de início
      while (client.connected()) { // Enquanto o cliente estiver conectado
        if (client.available()) { // Se houver dados disponíveis do cliente
          file.write(client.read()); // Escreve os dados no arquivo
        }
      }
      unsigned long endTime = millis(); // Marca o tempo de término
      Serial.print("Received file in: ");
      Serial.print(endTime - startTime);
      Serial.println(" ms"); // Imprime o tempo total de recebimento do arquivo
      file.close(); // Fecha o arquivo
    }
    client.stop(); // Fecha a conexão com o cliente
  }
}
```

Código do Cliente Wi-Fi (Receptor):

```
include <WiFi.h>
include <SPIFFS.h>

/ Define as credenciais da rede Wi-Fi
onst char* ssid = "ESP_WIFI";
onst char* password = "12345678";
onst char* serverIP = "192.168.1.100"; // IP do servidor
onst int serverPort = 80; // Porta do servidor

oid setup() {
 Serial.begin(115200); // Inicializa a comunicação serial
 WiFi.begin(ssid, password); // Conecta-se à rede Wi-Fi

 // Aguarda a conexão com o Wi-Fi
 while (WiFi.status() != WL_CONNECTED) {
   delay(1000);
   Serial.println("Connecting to WiFi...");
 }
 Serial.println("Connected to WiFi");
 SPIFFS.begin(); // Inicializa o sistema de arquivos SPIFFS


oid loop() {
 WiFiClient client;
 if (client.connect(serverIP, serverPort)) { // Conecta-se ao servidor
   File file = SPIFFS.open("/testfile.txt", FILE_READ); // Abre um arquivo para leitura no SPIFFS
   if (file) {
     unsigned long startTime = millis(); // Marca o tempo de início
     while (file.available()) { // Enquanto houver dados disponíveis no arquivo
       client.write(file.read()); // Envia os dados ao servidor
     }
     unsigned long endTime = millis(); // Marca o tempo de término
     Serial.print("Transfer time: ");
     Serial.print(endTime - startTime);
     Serial.println(" ms"); // Imprime o tempo total de transferência do arquivo
     file.close(); // Fecha o arquivo
   }
   client.stop(); // Fecha a conexão com o servidor
 }
 delay(60000); // Aguarda 60 segundos antes de realizar outra transferência
```

Código do Cliente Bluetooth (Remetente):

```
#include "BluetoothSerial.h"
#include <SPIFFS.h>

// Cria uma instância da classe BluetoothSerial
BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200); // Inicializa a comunicação serial
  SerialBT.begin("ESP_BT_Rem"); // Inicia o Bluetooth com o nome "ESP_BT_Rem"
  Serial.println("Bluetooth Started");
  SPIFFS.begin(); // Inicializa o sistema de arquivos SPIFFS
}

void loop() {
  if (SerialBT.connected()) { // Verifica se há uma conexão Bluetooth ativa
    File file = SPIFFS.open("/testfile.txt", FILE_READ); // Abre um arquivo para leitura no SPIFFS
    if (file) {
      unsigned long startTime = millis(); // Marca o tempo de início
      while (file.available()) { // Enquanto houver dados disponíveis no arquivo
        SerialBT.write(file.read()); // Lê os dados do arquivo e os envia via Bluetooth
      }
      unsigned long endTime = millis(); // Marca o tempo de término
      Serial.print("Transfer time: ");
      Serial.print(endTime - startTime);
      Serial.println(" ms"); // Imprime o tempo total de transferência do arquivo
      file.close(); // Fecha o arquivo
    }
    delay(60000); // Aguarda 1 minuto antes de iniciar outro teste
  }
}
```

Código do Cliente Bluetooth (Receptor):

```
#include "BluetoothSerial.h"
#include <SPIFFS.h>

// Cria uma instância da classe BluetoothSerial
BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200); // Inicializa a comunicação serial
  SerialBT.begin("ESP_BT_Rec"); // Inicia o Bluetooth com o nome "ESP_BT_Rec"
  Serial.println("Bluetooth Started");
  SPIFFS.begin(); // Inicializa o sistema de arquivos SPIFFS
}

void loop() {
  if (SerialBT.available()) { // Verifica se há dados disponíveis via Bluetooth
    File file = SPIFFS.open("/receivedfile.txt", FILE_WRITE); // Abre um arquivo para escrita no SPIFFS
    if (file) {
      unsigned long startTime = millis(); // Marca o tempo de início
      while (SerialBT.available()) { // Enquanto houver dados disponíveis via Bluetooth
        file.write(SerialBT.read()); // Lê os dados do Bluetooth e os escreve no arquivo
      }
      unsigned long endTime = millis(); // Marca o tempo de término
      Serial.print("Received file in: ");
      Serial.print(endTime - startTime);
      Serial.println(" ms"); // Imprime o tempo total de recebimento do arquivo
      file.close(); // Fecha o arquivo
    }
  }
}
```

ENCERRAMENTO

O projeto será encerrado quando todos os requisitos forem atingidos e a entrega efetuada, bem como a validação do projeto por parte do cliente. Todos os objetivos específicos devem ser cumpridos, incluindo o desenvolvimento do sistema de avaliação, a realização dos testes de transferência de arquivos via Wi-Fi e Bluetooth, e a análise comparativa das tecnologias.

A validação final incluirá a verificação dos dados coletados e a confirmação de que os resultados atendem às expectativas delineadas no início do projeto. Também envolverá a entrega de toda a documentação necessária, incluindo os relatórios de análise e os códigos desenvolvidos.