UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE ITAPAJÉ

**UNIVERSIDADE FEDERAL DO CEARÁ**

**CAMPUS ITAPAJÉ**

**JARDINS DE ANITA**

# CHAT SIMULTÂNEO UTILIZANDO ARQUITETURA CLIENTE/SERVIDOR COM RASPBERRY PI

**PROF. JUAN SEBASTIAN TOQUICA ARENAS**

**ALUNOS:**
**ALLAN MICHEL ROCHA DOS SANTOS,**
**ANTONIO FILIPE SOUSA SILVA,**
**PEDRO JONNATHAN MATOS DE SOUSA**

**ITAPAJÉ , CEARÁ , JUNHO DE 2023**

## LISTA DE FIGURAS

**SUMÁRIO**

# 1. RESUMO

O objetivo deste trabalho é implementar sistemas distribuídos e embarcados usando o Raspberry Pi 3B, visando estabelecer uma comunicação cliente/servidor. Por meio desse projeto, busca-se adquirir conhecimento e compreensão sobre o funcionamento dessa comunicação e explorar as capacidades do Raspberry Pi como servidor nesse contexto. Essa iniciativa permitirá aprofundar o entendimento sobre o tema e estudar o comportamento e o desempenho do Raspberry Pi como servidor.

**Palavras Chave:** Raspberry, Cliente/Servidor, Sistema distribuídos, Sistema embarcados ;

# 2. INTRODUÇÃO

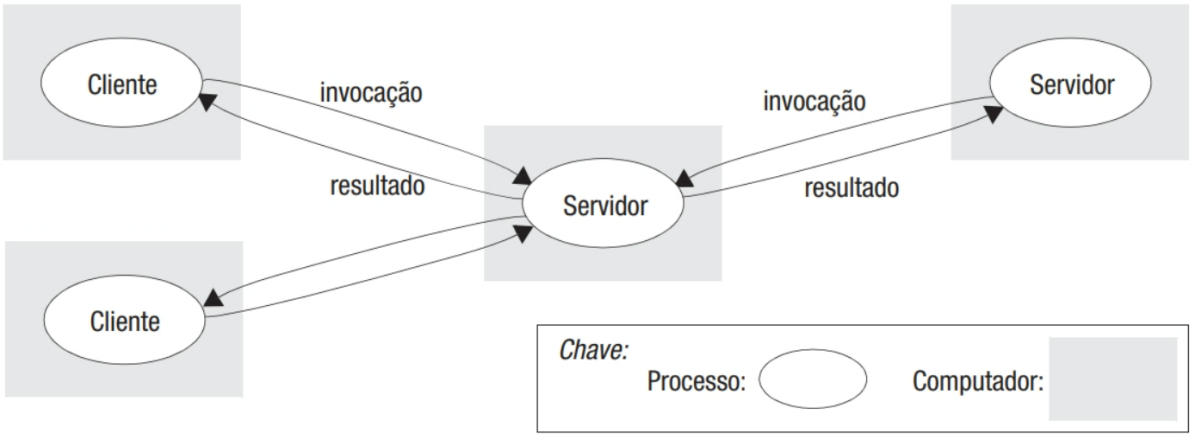
## 2.1. ASPECTOS FUNDAMENTAIS DE SISTEMAS DISTRIBUÍDOS

No âmbito da computação, com a grande miríade de informação e redes interligadas entre si, torna-se essencial a introdução aos sistemas distribuídos. Neste contexto, um sistema distribuído é aquele em que os componentes localizados em computadores conectados em rede se comunicam e coordenam suas ações exclusivamente por meio de troca de mensagens. Essa definição leva a três características especialmente importantes dos sistemas distribuídos: concorrência entre os componentes, ausência de um relógio global compartilhado e a possibilidade de falhas em componentes independentes. (COULOURIS, George; DOLLIMORE, JEAN; KINDERBERG, Tim; et al. , 2013)

### 2.1.1. COMUNICAÇÃO CLIENTE-SERVIDOR

Coulouris et al. (2013) pontuam que nesta rede, um servidor central gerencia e controla os recursos compartilhados, enquanto os clientes se conectam ao servidor para acessá-los. A comunicação entre eles é baseada em trocas de mensagens, em que o cliente envia solicitações e o servidor responde com as informações solicitadas. Em uma rede cliente-servidor, o servidor pode ser um computador dedicado, que é projetado especificamente para gerenciar e fornecer serviços aos clientes, enquanto os clientes podem ser dispositivos de computação de diferentes tipos, como desktops, laptops, tablets e smartphones. Esse modelo é adequado para redes em que é necessário um alto grau de centralização e controle, como em empresas ou organizações.

Figura 1 - Representação onde os clientes chamam o servidor




Fonte: COULOURIS, George; DOLLIMORE, JEAN; KINDERBERG, Tim; et al. (2013)

### 2.1.2. PROTOCOLO TCP/IP

O TCP/IP é um conjunto de protocolos de comunicação usado para conectar dispositivos em redes de computadores. Ele consiste em dois protocolos principais: o TCP, que garante a entrega confiável e ordenada dos dados, e o IP, responsável pelo endereçamento e roteamento dos pacotes na rede. O TCP/IP opera em camadas, seguindo o modelo OSI. As principais camadas, como vistas na figura 2, são: aplicação, transporte, rede e interface de rede. Cada camada desempenha funções específicas para permitir a comunicação entre dispositivos conectados. Esse conjunto de protocolos é essencial para a comunicação em redes modernas, tanto na Internet quanto em redes locais. Ele garante a transmissão confiável e eficiente dos dados, permitindo que dispositivos se comuniquem e compartilhem informações de forma padronizada. O TCP/IP é amplamente adotado e suportado por diferentes sistemas operacionais e dispositivos de rede, tornando-se o padrão dominante para a comunicação em redes de computadores. (COULOURIS, George; DOLLIMORE, JEAN; KINDERBERG, Tim; et al. , 2013)

Figura 2 - Exemplos de protocolos OSI

| *Camada* | *Descrição* | *Exemplos* |
|---|---|---|
| Aplicativo | Protocolos projetados para atender aos requisitos de comunicação de aplicativos específicos, frequentemente definindo uma interface para um serviço. | HTTP, FTP, SMTP, CORBA IIOP |
| Apresentação | Os protocolos deste nível transmitem dados em uma representação de rede independente das usadas em cada computador, as quais podem diferir. A criptografia, se exigida, também é feita nesta camada. | Segurança TLS, CORBA Data Rep. |
| Sessão | Neste nível são realizadas a confiabilidade e a adaptação, como a detecção de falhas e a recuperação automática. | SIP |
| Transporte | Este é o nível mais baixo em que mensagens (em vez de pacotes) são manipuladas. As mensagens são endereçadas para portas de comunicação associadas aos processos. Os protocolos desta camada podem ser orientados a conexão ou não. | TCP, UDP |
| Rede | Transfere pacotes de dados entre computadores em uma rede específica. Em uma rede de longa distância, ou em redes interligadas, isso envolve a geração de uma rota passando por roteadores. Em uma única rede local, nenhum roteamento é exigido. | IP, circuitos virtuais ATM |
| Enlace | Responsável pela transmissão de pacotes entre nós que estão diretamente conectados por um enlace físico. Em uma rede de longa distância, a transmissão é feita entre pares de roteadores ou entre roteadores e *hosts*. Em uma rede local, ela é feita entre qualquer par de *hosts*. | MAC Ethernet, transferência de célula ATM, PPP |
| Física | São os circuitos e o *hardware* que materializam a rede. Essa camada transmite sequências de dados binários através de sinalização analógica, usando modulação em amplitude ou em frequência de sinais elétricos (em circuitos a cabo), sinais luminosos (em circuitos de fibra óptica) ou outros sinais eletromagnéticos (em circuitos de rádio e micro-ondas). | Sinalização de banda-base Ethernet, ISDN |

Fonte: COULOURIS, George; DOLLIMORE, JEAN; KINDERBERG, Tim; et al. (2013)

### 2.1.3. SOCKETS

Socket é uma abstração de software que permite a comunicação entre processos através de uma rede. Em termos simples, um socket é um ponto final de comunicação que possibilita o envio e o recebimento de dados entre dispositivos conectados em uma rede. No contexto de programação, os sockets são utilizados para estabelecer conexões e trocar informações entre um cliente e um servidor. Eles permitem que aplicações desenvolvidas em diferentes linguagens de programação se comuniquem de forma padronizada, seguindo os protocolos de rede estabelecidos. Através do uso de sockets, é possível criar aplicativos que sejam capazes de enviar e receber dados, seja em forma de texto, arquivos,

comandos ou qualquer outro tipo de informação, através de uma rede.

### 2.1.4. LOG

Os logs de sistema são registros detalhados de eventos e atividades em um sistema de computador. Eles desempenham um papel crucial na monitoração, análise de desempenho, detecção de problemas e investigação de ameaças de segurança. Ao configurar e analisar corretamente os logs do sistema, é possível obter informações valiosas para garantir a integridade e a segurança do sistema.

### 2.1.5. THREADS

Threads são unidades básicas de execução dentro de um programa. Elas permitem a execução simultânea de diferentes partes do programa dentro de um único processo, melhorando a eficiência e o desempenho. No entanto, é necessário tomar precauções para evitar problemas de concorrência ao trabalhar com threads.

### 2.1.6. PYTHON

Python é uma linguagem de programação de alto nível, amplamente utilizada em diversas áreas de desenvolvimento de software. Ela se destaca por sua simplicidade, legibilidade e pela vasta quantidade de bibliotecas e frameworks disponíveis. No contexto de comunicação através de sockets, Python oferece uma biblioteca padrão chamada "socket" que facilita a implementação de funcionalidades de rede. Essa biblioteca fornece uma interface simples e abrangente para a criação de sockets, envio e recebimento de dados, além de suportar diferentes protocolos de rede, como TCP/IP e UDP. Através da biblioteca "socket" do Python, é possível desenvolver tanto clientes quanto servidores que utilizam sockets para estabelecer conexões, trocar dados e fornecer serviços de rede. Essa facilidade de implementação torna Python uma escolha popular para o desenvolvimento de aplicativos de rede, desde simples trocas de mensagens até aplicações complexas em tempo real.

## 2.2. ASPECTOS FUNDAMENTAIS SOBRE SISTEMAS EMBARCADOS

Os sistemas embarcados são sistemas computacionais dedicados a realizar tarefas específicas em dispositivos eletrônicos. Eles são construídos com hardware e software personalizados para atender aos requisitos do dispositivo. Esses sistemas operam com recursos limitados e são otimizados para um desempenho eficiente. Os sistemas embarcados podem operar em tempo real, respondendo a eventos e executando tarefas dentro de prazos específicos. Eles estão se tornando cada vez mais conectados em rede, permitindo a comunicação e a troca de informações com outros dispositivos. É crucial realizar testes e validação rigorosos para garantir a confiabilidade e a segurança desses sistemas. Além disso, eles são projetados para ter um ciclo de vida prolongado,

requerendo manutenção, atualizações e suporte contínuos. Os sistemas embarcados desempenham um papel fundamental na automação, controle e comunicação em diversos setores e dispositivos eletrônicos. (CERQUEIRA, Marcos V B.; MASCHIETTO, Luis G.; ZANIN, Aline; et al., 2021)

### 2.2.1. MICROPROCESSADORES

Os microprocessadores são dispositivos eletrônicos que funcionam como a unidade central de processamento (CPU) em um computador. Eles são responsáveis por executar as instruções e processar os dados em um sistema computacional. Os microprocessadores são compostos por milhões de transistores integrados em um único chip de silício. Eles contêm unidades de controle, unidades aritméticas e lógicas, registradores e memória cache, que permitem realizar operações lógicas, aritméticas e de controle. Esses dispositivos são projetados para executar instruções em sequência e em alta velocidade. Eles interpretam e executam instruções básicas, como adição, subtração, multiplicação, divisão, comparação e desvio, permitindo a execução de programas e o processamento de dados. Os microprocessadores evoluíram significativamente desde a sua criação, com o aumento da velocidade, aprimoramento da eficiência energética e o desenvolvimento de arquiteturas mais avançadas. Hoje em dia, encontramos microprocessadores em uma ampla variedade de dispositivos, desde computadores pessoais e smartphones até eletrodomésticos, veículos e dispositivos de Internet das Coisas (IoT). Esses componentes são fundamentais para o funcionamento de sistemas computacionais modernos, fornecendo o poder de processamento necessário para executar tarefas complexas em um curto espaço de tempo. Sua evolução contínua impulsiona o avanço da tecnologia e o desenvolvimento de novas aplicações em diversos setores da indústria e da vida cotidiana. (CERQUEIRA, Marcos V B.; MASCHIETTO, Luis G.; ZANIN, Aline; et al., 2021)

### 2.2.2. ARQUITETURA RISC

Isso resulta em A arquitetura RISC (Reduced Instruction Set Computing) é um paradigma de projeto de processadores que se baseia na simplicidade e eficiência. Ao contrário da arquitetura CISC (Complex Instruction Set Computing), a RISC utiliza um conjunto reduzido de instruções altamente otimizadas. processadores mais rápidos e eficientes em termos de energia, utilizados em dispositivos eletrônicos diversos, impulsionando o avanço tecnológico.

Devido ao conjunto de instruções menores, os processadores RISC costumam ser mais baratos e fisicamente menores em comparação aos processadores CISC. Essa característica torna a arquitetura RISC mais adequada para dispositivos com recursos limitados ou aplicações específicas que não exigem

um poder de processamento avançado. (CERQUEIRA, Marcos V B.; MASCHIETTO, Luis G.; ZANIN, Aline; et al., 2021)

### 2.2.3. ARQUITETURA ARM

A arquitetura ARM (Advanced RISC Machine) é uma arquitetura de processador amplamente utilizada em dispositivos móveis, sistemas embarcados e outros dispositivos de baixo consumo de energia. Esta arquitetura foi desenvolvida pela empresa britânica ARM Holdings e tem sido um dos principais impulsionadores da inovação tecnológica nas últimas décadas; E ao longo dos anos, acabou licenciando sua arquitetura para várias empresas de semicondutores, permitindo que elas projetaram e fabricaram seus próprios chips baseados na arquitetura ARM(como Texas Instruments, Qualcomm, Nvidia, Apple e Samsung).

A história da arquitetura ARM remonta aos anos 1980, quando a empresa Acorn Computers Ltd. começou a desenvolver um novo computador doméstico conhecido como Acorn Archimedes. Para acompanhar esse projeto, a Acorn percebeu a necessidade de uma arquitetura de processador eficiente e econômica que pudesse competir com os processadores Intel e Motorola disponíveis na época. Como resultado, a empresa estabeleceu a divisão ARM em 1990, juntamente com a Apple Computer e a VSLI Technology, que se concentraram exclusivamente neste projeto.

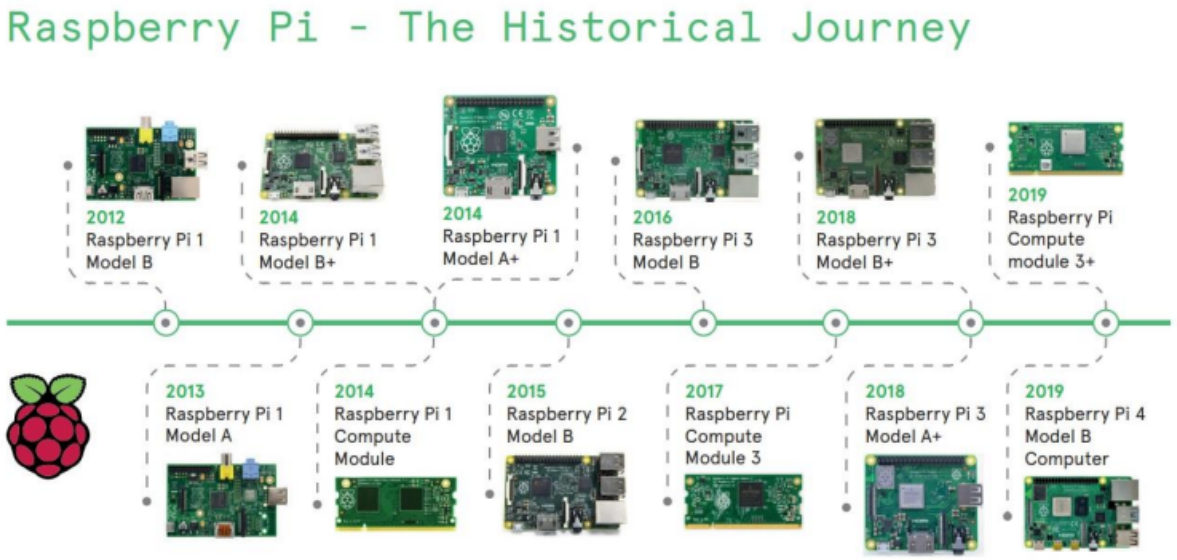
Uma das principais características da arquitetura ARM é sua natureza RISC (Reduced Instruction Set Computing), Essa abordagem permitiu que os processadores ARM se destacassem em dispositivos móveis e sistemas embarcados, onde a eficiência energética é crucial. (UPTON; HALLFARCREE, 2016). Sendo uma das aplicações mais notáveis em sistemas embarcados, o Raspberry Pi, um computador de placa única que se tornou um fenômeno no campo da computação de baixo custo e educação tecnológica.

### 2.2.4. RASPBERRY PI

O Raspberry Pi é um microcomputador completo, com seus componentes em uma única placa lógica. Há o processador, a memória RAM e a placa de vídeo impressos, e entradas USB, HDMI, áudio e vídeo composto, para câmera e telas LCD e uma GPIO, com pinos I/O de múltiplo propósito. A alimentação é feita através de uma porta microUSB, que permite usar fontes de energia de telefones celulares. O RPi pode ser utilizado como um computador normal, com teclado, mouse, monitor (TVs de tubo inclusas), fonte e um cartão microSD com o sistema

e programas (UPTON; HALLFARCREE, 2016). No mercado, são encontrados vários modelos com uma miríade de especificações diferentes no tocante a sua capacidade de processamento, memória e demais componentes já citados, como mostrado na figura 3.

Figura 3 - Jornada Histórica do Raspberry PI



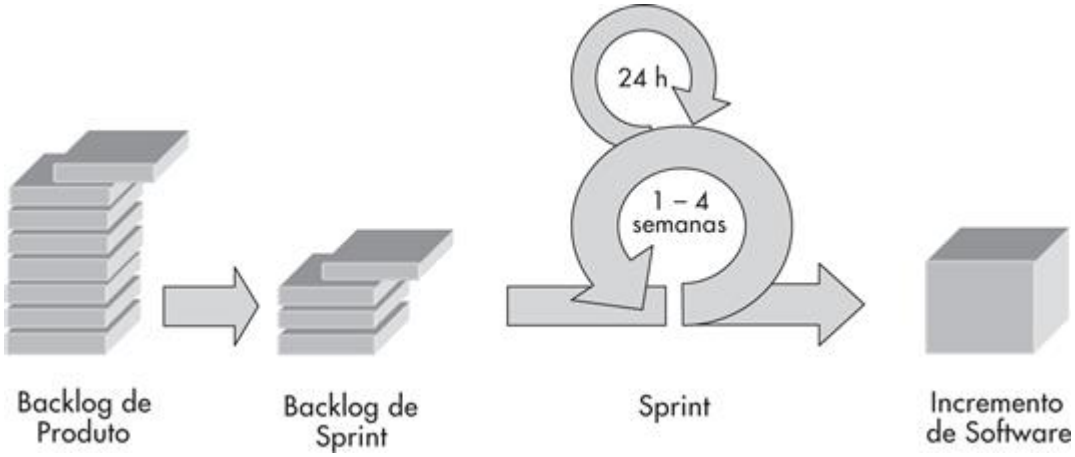

Fonte: Element14 (2012)

## 3. METODOLOGIA

Primordialmente, o primeiro trabalho adotou como forma de gerenciamento de projeto, a metodologia ágil Scrum, que trabalha de forma incremental e iterativa. Nele, o processo, como visto na figura 4, é dividido em dois ciclos principais. O primeiro ciclo é chamado de Sprint e dura de 1 a 4 semanas. Durante esse período, uma funcionalidade específica é desenvolvida e entregue aos envolvidos. Essa funcionalidade é definida a partir das necessidades do cliente, que são organizadas em uma lista chamada Backlog de Produto. Essa lista é discutida e priorizada para criar o Backlog de Sprint, que são as tarefas a serem realizadas durante o ciclo. Durante um Sprint, não são permitidas mudanças. Ao final de um Sprint, há uma reunião de revisão do que foi feito com os envolvidos. Esse ciclo se repete para o próximo Sprint. O segundo ciclo acontece dentro de um Sprint e envolve reuniões diárias, com duração de 24 horas, para acompanhar o progresso do dia anterior, identificar problemas e priorizar as tarefas do dia (HIRAMA, Kechi, 2011).

Com esta técnica, foi implementada de forma adaptada cerimônias, onde eram realizadas sprints semanalmente com o Professor Juan Sebastian Toquica Arenas, atuando de forma condizente a um Product Owner e organização das tarefas registradas no Trello.

Figura 4 - Processo Scrum




Fonte: HIRAMA, Kechi (2011)

Neste trabalho, utilizamos uma abordagem baseada em pesquisa bibliográfica e implementação de código para desenvolver o chat simultâneo. Para embasar nosso estudo, é compreender os conceitos essenciais, utilizamos o livro "Distributed Systems" de Andrew S. Tanenbaum como uma fonte de referência sobre sistemas distribuídos. No processo de implementação do chat simultâneo, utilizamos o livro "Distributed Systems: Concepts and Design" de George Coulouris como uma fonte de pesquisa para entender os princípios fundamentais dos sistemas distribuídos.

A implementação do código foi realizada utilizando a linguagem de programação Python, que oferece suporte robusto para o desenvolvimento de aplicativos de rede. A documentação oficial do Python foi uma valiosa fonte de referência para a implementação das funcionalidades necessárias. Para garantir a simultaneidade no chat, adotamos a biblioteca threading do Python. Essa biblioteca nos permitiu criar threads independentes para cada cliente, permitindo que eles se conectem e interajam simultaneamente no chat. Através do uso da biblioteca threading, conseguimos gerenciar várias conexões e mensagens de forma eficiente. A criação das conexões entre o servidor e os clientes foi realizada utilizando a biblioteca socket do Python. Essa biblioteca fornece uma interface para criar e gerenciar sockets, permitindo a comunicação bidirecional entre os dispositivos conectados à rede. Utilizamos uma abordagem iterativa no desenvolvimento do chat simultâneo, realizando testes e ajustes à medida que progredimos.

À medida em que encontrávamos desafios e obstáculos, utilizamos a literatura de referência mencionada anteriormente e a documentação do Python para encontrar soluções adequadas.

**Revisão Bibliográfica:**

- Foi realizada uma revisão da literatura sobre sistemas distribuídos, sistemas embarcados e a arquitetura cliente/servidor.
- Foi estudado os principais conceitos, protocolos e tecnologias relacionados ao tema do

estudo.
- Analisamos estudos de caso e projetos anteriores que tenham utilizado o Raspberry Pi 3B para sistemas distribuídos.

**Configuração do Ambiente:**
- Foi instalado e configurado o sistema operacional Raspbian no Raspberry Pi 3B.
- Estabeleceu-se as configurações de rede Wi-Fi, para permitir a conexão com os clientes.

**Desenvolvimento do Servidor:**
- Foi implementado a lógica do servidor no Raspberry Pi 3B utilizando Python.
- Foi definido as funcionalidades do servidor, incluindo o gerenciamento de conexões e o processamento de requisições dos clientes.
- Foi desenvolvido a lógica de armazenamento e processamento dos dados recebidos.

**Desenvolvimento dos Clientes:**
- Foi desenvolvido a lógica dos clientes, que podem ser dispositivos como PCs, como sistemas operacionais Linux e Windows.
- Foi definido as funcionalidades dos clientes, como o envio de requisições e o processamento dos dados recebidos do servidor.

**Análise de Resultados:**
- Foi analisado e interpretado os resultados dos testes e avaliações.
- Foi avaliado a eficácia do sistema distribuído implementado e a qualidade da comunicação cliente/servidor.
- Foi identificado vantagens, limitações e oportunidades de aprimoramento do sistema desenvolvido.

**Documentação e Relatório Final:**
- Foi documentado todo o processo de desenvolvimento, incluindo a arquitetura do sistema, os protocolos utilizados e as configurações do Raspberry Pi 3B.

## 4. CONCLUSÃO

Neste artigo, foi explorado a criação de um chat simultâneo utilizando a biblioteca threading, em conjunto com a linguagem de programação Python e o protocolo TCP/IP por meio de sockets. Além disso, utilizamos o Raspberry Pi como servidor e notebooks como clientes para estabelecer a comunicação. A implementação desse chat simultâneo demonstrou a flexibilidade e poder do Python no desenvolvimento de aplicativos de rede. Com o auxílio da biblioteca threading, conseguimos criar uma aplicação capaz de lidar com várias conexões simultâneas, permitindo que os clientes se comuniquem entre si de forma eficiente.

A escolha do protocolo TCP/IP, através do uso de sockets, foi fundamental para garantir a confiabilidade e a entrega ordenada dos dados trocados entre os participantes do chat. Essa combinação ofereceu uma experiência de comunicação fluida, permitindo que os usuários compartilhassem mensagens de forma instantânea e eficaz. O Raspberry Pi, atuando como servidor, mostrou-se uma escolha versátil e acessível para hospedar a aplicação de chat. Sua capacidade de processamento e conectividade facilitou a criação de um ambiente de comunicação centralizado e confiável. Ao utilizar notebooks como clientes, foi possível aproveitar os recursos de interface gráfica e interação intuitiva oferecidos por esses dispositivos, proporcionando uma experiência amigável aos usuários durante as trocas de mensagens. Em resumo, a combinação da biblioteca threading, a linguagem Python e o protocolo TCP/IP utilizando sockets permitiu a criação de um chat simultâneo, onde o Raspberry Pi atuou como servidor e notebooks como clientes. Esse projeto exemplifica a capacidade de Python e o potencial dos dispositivos Raspberry Pi para desenvolver soluções de comunicação eficientes e interativas. Com o conhecimento adquirido, é possível explorar e expandir ainda mais as possibilidades de aplicativos de rede, proporcionando interações

colaborativas em tempo real.

## 5. REFERÊNCIAS


COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim; et al. Sistemas distribuídos. [Porto Alegre - RS]: Grupo A, 2013. E-book. ISBN 9788582600542. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788582600542/. Acesso em: 20 jun. 2023.

CERQUEIRA, Marcos V B.; MASCHIETTO, Luis G.; ZANIN, Aline; et al. Sistemas Operacionais Embarcados. [São Paulo - SP]: Grupo A, 2021. E-book. ISBN 9786556902616. Disponível em: https://app.minhabiblioteca.com.br/#/books/9786556902616/. Acesso em: 19 jun. 2023.

MASCHIETTO, Luis G.; MORAES, Diego Martins Polla de; ALVES, Nicolli Souza R.; et al. Desenvolvimento de Software com Metodologias Ágeis. [Porto Alegre - RS]: Grupo A, 2021. E-book. ISBN 9786556901824. Disponível em: https://app.minhabiblioteca.com.br/#/books/9786556901824/. Acesso em: 28 jun. 2023.

UPTON, Eben; HALFACREE, Gareth. Raspberry Pi user guide. John Wiley & Sons, 2016.

HIRAMA, Kechi. Engenharia de Software. [São Paulo - SP]: Grupo GEN, 2011. E-book. ISBN 9788595155404. Disponível em: https://app.minhabiblioteca.com.br/#/books/9788595155404/. Acesso em: 02 jul. 2023.

Python. A referência da Linguagem Python. Disponível em: <https://docs.python.org/pt-br/3.11/reference/index.html#reference-index>. Acesso em: 20 jun. 2023

Python. Socket - Low-Level networking interface. Disponível em: <https://docs.python.org/3.11/library/socket.html>. Acesso em: 20 jun. 2023

Python. Threading - Thread-based parallelism. Disponível em: <https://docs.python.org/3.11/library/threading.html>. Acesso em: 20 jun. 2023

Python. logging — Logging facility for Python. Disponível em: <https://docs.python.org/3/library/logging.html#module-logging>. Acesso em: 24 jun. 2023

Python. Tkinter — Python interface to Tcl/Tk. Disponível em: <https://docs.python.org/3.11/library/tkinter.html#module-tkinter>. Acesso em: 25 jun. 2023

Python. Select — Waiting for I/O completion. Disponível em: <https://docs.python.org/3.11/library/select.html#module-select>. Acesso em: 25 jun. 2023

Tecnoblog. O que é o Raspberry Pi?. Disponível em: <https://tecnoblog.net/responde/o-que-e-o-raspberry-pi/#:~:text=O%20que%20%C3%A9%20Raspberry%20Pi,I%2FO%20de%20m%C3%BAltiplo%20prop%C3%B3sito.>. Acesso em: 20 jun. 2023

Element14. 10 Years of Raspberry Pi - History of Raspberry Pi and element14 Community. Disponível em: <https://tecnoblog.net/responde/o-que-e-o-raspberry-pi/#:~:text=O%20que%20%C3%A9%20Raspberry%20Pi,I%2FO%20de%20m%C3%BAltiplo%20prop%C3%B3sito.>. Acesso em: 20 jun. 2023