Trabalho Prático 1

Sistema de controle de operação de salas-de-aula

Execução: **Individual**
Data de entrega: **22 de abril de 2024 até 23h:59min**

INTRODUÇÃO

Ondas de calor extremo vêm tomando conta de todo o mundo, inclusive aqui no Brasil, um país tropical já acostumado com o calor intenso que atinge a maior parte de suas regiões. No entanto, o aquecimento global e outros fenômenos climáticos tendem a tornar o calor ainda mais intenso, trazendo diversos efeitos à saúde e bem-estar das pessoas (Ecomax, 2023). Belo Horizonte foi a capital brasileira que mais esquentou no país em 2023, de acordo com o Centro Nacional de Monitoramento e Alerta de Desastres Naturais (Cemaden), do governo federal. A cidade chegou a registrar 4,2ºC acima da média. Este calor prejudica o desempenho de alunos em ambientes de ensino ao dificultar o raciocínio, a lógica e a aprendizagem dos alunos. O desenvolvimento de tecnologias IoT em Educação (Internet of Things in Education) permite aprimorar a coleta e análise de dados ao fornecer aos administradores escolares dados mais precisos sobre o ambiente educacional, desde o desempenho dos alunos até as condições ambientais das salas de aula.

As soluções IoT oferecem aos educadores maneiras mais eficientes de gerenciar suas salas de aula, interagir com os alunos e medir o progresso dos alunos. Além disso, ao conectar dispositivos físicos a redes, essas soluções coletam dados para análise posterior, a fim de compreender melhor o desempenho dos alunos e a situação das salas de aula. Os prédios e as salas de aula inteligentes estão se tornando cada vez mais populares nas instituições educacionais. Com o uso de sensores e outros dispositivos habilitados para IoT, essas salas podem ajustar a iluminação, a temperatura e diferentes configurações com base nas informações coletadas pelos sensores. Isso permite que os educadores criem um ambiente de aprendizagem ideal que aumenta o conforto e a concentração.

Uma empresa de controle e automação decidiu ingressar no segmento de monitoramento e de controle climático em universidades, onde necessita desenvolver um projeto piloto que consiste em criar um controle de informações climáticas de um centro de atividades didáticas capaz de registrar e monitorar dados sobre as salas de aula através de uma Unidade de Controle (UC) centralizada, responsável por todo o fluxo de informações e controles no sistema **(o servidor)** e de uma Unidade de Monitoramento (UM) cuja função consiste em cadastrar salas de aulas, coletar dados de temperatura, umidade do ar e estados de ventiladores das salas de aula através de sensores **(o cliente)** e retornar ao servidor por meio de protocolos de comunicação e da Internet. Assim, a Figura 1 ilustra a comunicação entre cada uma das entidades do sistema (sensores, cliente e servidor) operando em um pavimento em ambiente educacional.

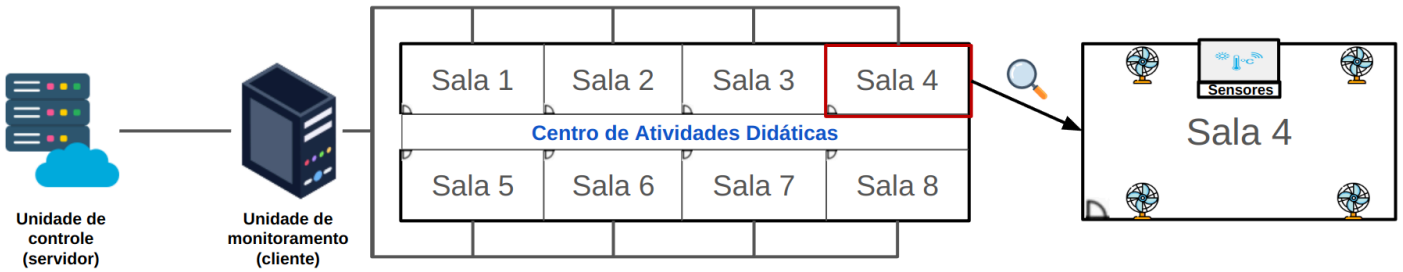

Figura 1 - Exemplo de comunicação entre as entidades do sistema

O programa cliente coleta dados de sensores instalados nas salas de aula para aferir a temperatura, a umidade do ar e os estados de **quatro** ventiladores, os quais são descritos na **Tabela I,** e envia os valores ao servidor que por sua vez armazena as informações em sua base de dados. Para que as mudanças aconteçam, o programa servidor espera receber os comandos do cliente, o qual informa os estados das salas de aula monitoradas pelos sensores, de acordo com o exemplo abaixo:

Tabela I - Dados coletados pelos sensores

ID da sala = [0,7]	Temperatura = [0,40] ◦C	Umidade do ar = [0,100] %	Estado de ventiladores = [ID][0 = Defeituoso, 1 = Desligado, 2 = Ligado]
0	27 ◦C	80 %	10, 20, 31, 40*
1	35 ◦C	70 %	11, 22, 32, 41**
2	18 ◦C	97 %	11, 21, 31, 41

* Ventiladores com IDs 1, 2 e 4 estão com defeito (0), ventilador com ID 3 está desligado (1)
** Ventiladores com IDs 1 e 4 estão desligados (1), ventiladores com IDs 2 e 3 estão ligados (2)

Diante do exposto, neste trabalho, você será responsável por desenvolver um **sistema em rede cliente-servidor** para simular a interação entre a **Unidade de Controle - UC (o servidor)** e a **Unidade de Monitoramento - UM (o cliente)**. A UC deve atender às seguintes solicitações da UM:

1. **Cadastrar Sala de Aula:** Instancia uma sala de aula e atribui um ID.
2. **Ligar Sensores:** Inicia coleta de dados dos sensores de uma sala de aula.
3. **Desligar Sensores:** Encerra a coleta de dados dos sensores de uma sala de aula.
4. **Atualizar Informações de Sensores:** Altera os valores de dados dos sensores de uma sala de aula.
5. **Consultar Informações de Sala de Aula:** Informa os atuais valores dos sensores de uma sala de aula.
6. **Consultar Tabela de Salas de Aulas:** Informa os valores armazenados dos sensores de todas as salas de aula.

Cada uma dessas solicitações correspondem a uma mensagem enviada pela UM à UC. Neste trabalho, **você deve implementar os seis tipos de mensagens propostas, bem como as mensagens de erro e confirmação** que serão especificadas nas próximas seções. Você desenvolverá dois (2) programas para um sistema simples de troca de mensagens utilizando apenas as interfaces da API de *sockets* POSIX e a comunicação via protocolo TCP. **Toda conexão deve utilizar a interface de *sockets* na linguagem C.**

Os objetivos gerais deste trabalho são:

1. Implementar UC (o servidor) utilizando a interface de *sockets* na linguagem C;
2. Implementar UM (o cliente) utilizando a interface de *sockets* na linguagem C;
3. Escrever o relatório.

PROTOCOLO

O protocolo de aplicação deverá funcionar sobre o protocolo TCP. Isso implica que as mensagens serão entregues sobre um canal de bytes com garantias de entrega em ordem, mas é sua responsabilidade implementar as especificações das mensagens e as funcionalidades tanto do servidor quanto do cliente.

A UC e a UM trocam mensagens curtas de até 500 bytes utilizando o TCP. As mensagens carregam textos codificados segundo a tabela ASCII. Apenas letras, números e espaços podem ser transmitidos. Caracteres acentuados e especiais não devem ser transmitidos.

ESPECIFICAÇÃO DAS MENSAGENS

Esta seção especifica as mensagens utilizadas na comunicação, bem como as mensagens de erro e confirmação. Nas tabelas abaixo, as células em "–" correspondem aos campos que não precisam ser definidos nas mensagens. As colunas Descrição e Exemplo não fazem parte da estrutura da mensagem.

Estrutura das Mensagens			
Action	**Info**	**Descrição**	**Exemplo**
CAD_REQ	**sala_id**	Mensagem de solicitação de cadastrar sala de aula	CAD_REQ **0**
INI_REQ	**sala_id** temp umid estados_vent	Mensagem de solicitação de iniciar sensores de sala de aula	INI_REQ **0** 27 80 10 20 31 40
DES_REQ	**sala_id**	Mensagem de solicitação de desligar sensores de sala de aula	DES_REQ **1**
ALT_REQ	**sala_id** temp umid estados_vent	Mensagem de solicitação de alteração de valores de sensores de sala de aula	ALT_REQ **0** 31 87 11 22 30 40
SAL_REQ	**sala_id**	Mensagem de solicitação de informações de sensores de sala de aula	SEN_REQ **2**
SAL_RES	**sala_id$_0$** temp$_0$ umid$_0$ estados_vent$_0$	Mensagem de resposta de informações de sensores de sala de aula	sala **0:** 31 87 11 22 30 40
CAD_REQ	–	Mensagem de solicitação de valores de sensores de todas as salas de aula	VAL_REQ
CAD_RES	**sala_id$_1$** temp$_1$ umid$_1$ estados_vent$_1$ **sala_id$_2$** temp$_2$ umid$_2$ estados_vent$_2$ …	Mensagem de resposta de valores de sensores	salas: **1** (31 87 11 22 30 40) **2** (35 70 11 22 32 41)

Mensagens de Erro e Confirmação			
Type	**Payload**	**Descrição**	**Exemplo**
ERROR	Código	Mensagem de erro transmitida do Servidor para Cliente. O campo payload deve informar o código de erro. Abaixo apresenta o código de cada mensagem: 01 : sala inválida 02 : sala já existe 03 : sala inexistente 04 : sensores inválidos 05 : sensores já instalados 06 : sensores não instalados	ERROR 01
OK	Código	Mensagem de confirmação transmitida do servidor para cliente. O campo payload deve informar a mensagem de confirmação. Abaixo apresenta o código de cada mensagem: 01 : sala instanciada com sucesso 02 : sensores inicializados com sucesso 03 : sensores desligados com sucesso 04 : informações atualizadas com sucesso	OK 02

DESCRIÇÃO DAS FUNCIONALIDADES

Esta seção descreve o fluxo de mensagens transmitidas entre a UC e a UM resultante de cada uma das seis funcionalidades da aplicação a fim de gerenciar os sensores na rede elétrica.

1) Cadastrar Sala de Aula

1. A UM recebe comando via teclado
   register **sala_id**
   para o cadastro da sala **sala_id**. A UM verifica se esse valor está de acordo com o padrão de entrada especificado na **TABELA I**.
   1.1. Em caso negativo, a UM imprime a mensagem de erro código 01 (vide *Especificação das Mensagens*) e cancela a ação.
   1.2. Em caso positivo, a UM envia a mensagem **CAD_REQ** para a UC.
2. A UC recebe a solicitação e verifica se a sala existe na **TABELA I**.
   2.1. Em caso positivo, a UC responde com mensagem de erro código 02.
      2.1.1. A UM recebe código de erro e imprime sua descrição na tela.
   2.2. Em caso negativo, a UC adiciona a sala à **TABELA I,** atribui **-1** às informações dos sensores e responde a mensagem de confirmação código 01.
      2.2.1. A UM recebe o código de confirmação e imprime sua descrição na tela.

2) Ligar Sensores

1. A UM recebe comando via teclado
   init file **nome_arquivo**
   init info **sala_id** temp umid estados_vent

para a iniciação dos sensores na sala **sala_id** de valores $temp_{id}$ $umid_{id}$ $estados\_vent_{id}$. A UM verifica se esses valores estão de acordo com o padrão de entrada especificado na **TABELA I**.

1.1. Em caso negativo, a UM imprime, nesta ordem de prioridade, a mensagem de erro código 01 (vide *Especificação das Mensagens*) caso **sala_id** esteja fora do padrão de entrada ou imprime a mensagem de erro código 04 caso os valores dos sensores estejam fora dos padrões de entrada, e cancela a ação.

1.2. Em caso positivo, a UM envia a mensagem **INI_REQ** para a UC.

2. A UC recebe a solicitação e verifica se a sala existe na **TABELA I**.

2.1. Em caso negativo, responde com mensagem de erro código 03.

2.1.1. A UM recebe código de erro e imprime sua descrição na tela.

2.2. Em caso positivo, a UC verifica se os sensores estão instalados na **TABELA I.**

2.2.1. Em caso positivo, responde com mensagem de erro código 05.

2.2.1.1. A UM recebe código de erro e imprime sua descrição na tela.

2.2.2. Em caso negativo, a UC adiciona as informações dos sensores à **TABELA I** e responde a mensagem de confirmação código 02.

2.2.2.1. A UM recebe código de confirmação e imprime sua descrição na tela.

3) Desligar Sensores

1. A UM recebe comando via teclado

shutdown **sala_id**

para o desligamento dos sensores da sala **sala_id**. Para isso, a UM envia a mensagem **DES_REQ** para a UC.

2. A UC recebe solicitação e verifica se a sala existe na **TABELA I**.

2.1. Em caso negativo, a UC responde com mensagem de erro código 03.

2.1.1. A UM recebe código de erro e imprime sua descrição na tela.

2.2. Em caso positivo, a UC verifica se os sensores da sala **sala_id** estão instalados.

2.2.1. Em caso negativo, responde com mensagem de erro código 06.

2.2.1.1. A UM recebe código de erro e imprime sua descrição na tela.

2.2.2. Em caso positivo, a UC coloca **-1** nas informações dos sensores da sala **sala_id** e responde com mensagem de confirmação código 03.

2.2.2.1. A UM recebe código de confirmação e imprime sua descrição na tela.

4) Atualizar Informações de Sensores

1. A UM recebe comando via teclado

update file **nome_arquivo**
update info **sala_id** temp umid estados_vent

para a alteração dos valores dos sensores na sala **sala_id** de valores $temp_{id}$ $umid_{id}$ $estados\_vent_{id}$. A UM verifica se esses valores estão de acordo com o padrão de entrada especificado na **TABELA I**.

1.1. Em caso negativo, a UM imprime, nesta ordem de prioridade, a mensagem de erro código 01 (vide *Especificação das Mensagens*) caso **sala_id** esteja fora do padrão de entrada ou imprime a mensagem de erro código 04 caso os valores dos sensores estejam fora dos padrões de entrada, e cancela a ação.

1.2. Em caso positivo, a UM envia a mensagem **ALT_REQ** para a UC.

2. A UC recebe a solicitação e verifica se a sala existe na **TABELA I**.

2.1. Em caso negativo, responde com mensagem de erro código 03.

2.1.1. A UM recebe código de erro e imprime sua descrição na tela.

2.2. Em caso positivo, a UC verifica se os sensores estão instalados na **TABELA I.**

2.2.1. Em caso negativo, responde com mensagem de erro código 06.

2.2.1.1. A UM recebe código de erro e imprime sua descrição na tela.

2.2.2. Em caso positivo, a UC atualiza as informações dos sensores na **TABELA I** e responde a mensagem de confirmação código 04.

2.2.2.1. A UM recebe código de confirmação e imprime sua descrição na tela.

5) Consultar Informações de Sala de Aula

1. A UM recebe comando via teclado

   load info **sala_id**

   para mostrar os valores dos sensores da sala **sala_id**. Para isso, a UM envia a mensagem **SAL_REQ** para a UC.
2. A UC recebe solicitação e verifica se a sala existe na **TABELA I**.
   2.1. Em caso negativo, a UC responde com mensagem de erro código 03.
      2.1.1. A UM recebe código de erro e imprime sua descrição na tela.
   2.2. Em caso positivo, a UC verifica se os sensores estão instalados na **TABELA I.**
      2.2.1. Em caso negativo, responde com mensagem de erro código 06.
         2.2.1.1. A UM recebe código de erro e imprime sua descrição na tela.
      2.2.2. Em caso positivo, a UC responde a UM com os valores atuais dos sensores da sala **sala_id** por meio da mensagem **SAL_RES**.
         2.2.2.1. A UM recebe mensagem e imprime na tela:

         sala **sala_id:** $temp_{id}$ $umid_{id}$ $estados\_vent_{id}$

6) Consultar Tabela de Salas de Aulas

1. A UM recebe comando via teclado

   load rooms

   para mostrar os valores de sensores de todas as salas de aula instanciadas. Para isso, a UM envia a mensagem **CAD_REQ** para a UC.
2. A UC recebe a solicitação e verifica se existem salas na **Tabela I**.
   a. Em caso negativo, a UC responde com mensagem de erro código 03.
      i. UM recebe código de erro e imprime sua descrição na tela.
   b. Em caso positivo, a UC responde com os valores dos sensores instalados nas salas de aula por meio da mensagem **CAD_RES**.
      i. UM recebe mensagem e imprime na tela:

      salas: **sala_id$_0$** ($temp_0$ $umid_0$ $estados\_vent_0$) **sala_id$_2$** ($temp_2$ $umid_2$ $estados\_vent_2$) …

IMPLEMENTAÇÃO

Pequenos detalhes devem ser observados no desenvolvimento de cada programa que fará parte do sistema. É importante observar que o protocolo é simples e único (o cliente sempre tem que enviar a mensagem codificada para o servidor e vice-versa, de modo que o correto entendimento da mensagem deve ser feito por todos os programas).

Como mencionado anteriormente, o protocolo de transporte será o TCP, criado com **[socket(AF_INET, SOCK_STREAM, 0)]** ou com **[socket(AF_INET6, SOCK_STREAM, 0)]**, a fim de utilizar tanto os protocolos de redes IPv4 quanto o IPv6. O programador deve usar as funções *send* e *recv* para enviar e receber mensagens. O aluno deve implementar tanto uma versão do servidor (UC) quanto uma versão do cliente (UM).

Outros detalhes de implementação:

- As mensagens são terminadas com um caractere de quebra de linha `'\n'`. O caractere nulo `'\0'` para terminação de strings em C *não* deve ser enviado na rede.
- O cliente deve desconectar do servidor caso receba uma mensagem com um comando desconhecido (exemplo: "`ini`" em vez de "`init`"), mas não precisa retornar mensagem inválida.
- Para funcionamento do sistema de correção semi-automática (descrito abaixo), seu servidor deve fechar todas as conexões e terminar sua execução ao receber a mensagem **"`kill`"** a qualquer momento.

Limites:

- Cada mensagem possui no máximo 500 bytes.

Materiais para Consulta:

- Capítulos 2 e 3 do livro sobre programação com sockets disponibilizado no Moodle.
- Playlist de programação com sockets.

EXECUÇÃO

O **cliente** deve receber mensagens do teclado e imprimir as mensagens recebidas pelo servidor na tela. O **servidor** deve imprimir na saída padrão todas as mensagens recebidas dos clientes. **Não é necessário** que o servidor aceite mais de um cliente simultaneamente.

Seu servidor deve receber, **estritamente nessa ordem**, o tipo de endereço que será utilizado (**v4** para IPv4 ou **v6** para IPv6) e um número de porta na linha de comando especificando em qual porta ele vai receber conexões (Sugestão: utilize a porta 12345 para efeitos de padronização do trabalho). Seu cliente deve receber, **estritamente nessa ordem**, o endereço IP e a porta do servidor para estabelecimento da conexão. A seguir, um exemplo de execução de um cliente conectado com um servidor em dois terminais distintos:

```
Terminal 1: ./server v4 12345
Terminal 2: ./client 127.0.0.1 12345
```

AVALIAÇÃO

O trabalho deve ser realizado individualmente e **deve ser implementado na linguagem de programação C** utilizando somente a biblioteca padrão (interface POSIX de sockets de redes). Deve ser possível executar seu programa no sistema operacional **Linux** e **não deve utilizar bibliotecas Windows, como o winsock**. Seu programa deve interoperar com qualquer outro programa implementando o mesmo protocolo (você pode testar com as implementações dos seus colegas). Procure escrever seu código de maneira clara, com comentários pontuais e bem indentados. Isto facilita a correção dos monitores e tem impacto positivo na avaliação.

Correção Semi-automática

Seu servidor será corrigido de forma semi-automática por uma bateria de testes. Cada teste verifica uma funcionalidade específica do servidor. O seu servidor será testado por um cliente implementado pelo professor com funcionalidades adicionais para realização dos testes. Os testes avaliam a aderência do seu

servidor ao protocolo de comunicação inteiramente através dos dados trocados através da rede (a saída do seu servidor na tela, e.g., para depuração, não impacta os resultados dos testes).

Para a correção os seguintes testes serão realizados **(com IPv4 e IPv6)**:

- Cadastrar sala de aula: **+1 ponto**
- Ligar sensores: **+2 pontos**
- Desligar sensores: **+2 pontos**
- Alterar informações de sensores: **+2 pontos**
- Consultar informações de sala de aula: **+2 pontos**
- Consultar tabela de salas de aula: **+2 pontos**
- Testar casos de mensagem inválida: **+2 pontos**
- Testar casos de salas inválidos: **+1 ponto**
- Testar casos de sensores inválidos: **+1 ponto**
- Cliente envia kill para o servidor e encerrar a execução: **+1 ponto**

ENTREGA

Cada aluno deve entregar documentação em PDF de até 6 páginas, sem capa, utilizando fonte tamanho 10, e figuras de tamanho adequado ao tamanho da fonte. **Ele deve conter uma descrição da arquitetura adotada para o servidor, os refinamentos das ações identificadas no mesmo, as estruturas de dados utilizadas, as decisões de implementação não documentadas nesta especificação**. Como sugestão, considere incluir as seguintes seções no relatório: introdução, arquitetura, servidor, cliente, discussão e conclusão. O relatório deve ser entregue em formato PDF. A documentação corresponde a 20% dos pontos do trabalho (**+4 pontos)**, mas só será considerada para as funcionalidades implementadas corretamente.

Será utilizado um sistema para detecção de código repetido, portanto não é admitido cola de trabalhos.
Será adotada a média harmônica entre as notas da documentação e da execução, o que implica que a nota final será 0 se uma das partes não for entregue.

Cada aluno deve entregar, além da documentação, o **código fonte em C** e um **makefile** para compilação do programa. Instruções para submissão e compatibilidade com o sistema de correção semi-automática:

- O Makefile deve compilar o “client” e o “server”.
- Seu código deve ser compilado pelo comando “make” sem a necessidade de parâmetros adicionais.
- A entrega deve ser feita no formato ZIP, seguindo a nomenclatura: TP1_MATRICULA.zip
- O nome dos arquivos deve ser padronizado:
    - server.c
    - client.c
    - common.c, common.h (se houver)

PRAZO DE ENTREGA

Os trabalhos podem ser entregues até às 23:59 (vinte e três e cinquenta e nove) do dia especificado para a entrega.

Desconto de Nota por Atraso

Após a data e horário de entrega definido, os trabalhos já estarão sujeitos a penalidades em função dos total de dias de atraso. A fórmula para desconto por atraso na entrega do trabalho prático é:

$$desconto = d * 4$$

onde *d* é o atraso em dias corridos. Note que após 4 dias, o trabalho não deve ser mais entregue.

DICAS E CUIDADOS

- O guia de programação em rede do Beej (http://beej.us/guide/bgnet/) tem bons exemplos de como organizar um servidor
- Procure escrever seu código de maneira clara, com comentários pontuais e bem identado.
- Não se esqueça de **conferir se seu código não possui erros de compilação ou de execução**.

EXEMPLOS DE EXECUÇÃO

Esta seção apresenta alguns exemplos de execuções do sistema.

Exemplo 1 (Cadastrar, Ligar e Desligar)	**Exemplo 2 (Atualizar Informações de Sensores)**
register 0 sala instanciada com sucesso register 1 sala instanciada com sucesso init info 0 27 80 10 20 31 40 sensores inicializados com sucesso init file file1.txt sensores inicializados com sucesso shutdown 1 sensores desligados com sucesso	register 0 sala instanciada com sucesso init file file0.txt sensores inicializados com sucesso update info 0 31 87 11 22 30 40 informações atualizadas com sucesso update file file0.txt informações atualizadas com sucesso

Exemplo 3 (Consultar Informações de Sala de Aula)	**Exemplo 4 (Consultar Tabela de Salas de Aulas)**
register 0 sala instanciada com sucesso register 4 sala instanciada com sucesso init info 0 27 80 10 20 31 40 sensores inicializados com sucesso init file file4.txt sensores inicializados com sucesso load info 0 sala 0: 27 80 10 20 31 40 load info 4 sala 4: 40 35 12 22 32 42	register 4 sala instanciada com sucesso register 0 sala instanciada com sucesso register 8 sala instanciada com sucesso init info 0 27 80 10 20 31 40 sensores inicializados com sucesso init file file4.txt sensores inicializados com sucesso load rooms salas: 4 (40 35 12 22 32 42) 0 (27 80 10 20 31 40) 8 (-1 -1 -1 -1 -1 -1)

Exemplo 5 (Tratamento de erros)	Exemplo 6 (Tratamento de erros)
load rooms sala inexistente register 9 sala inválida register 0 sala instanciada com sucesso register 0 sala já existe init info 1 27 80 10 20 31 40 sala inexistente init info 0 85 80 10 20 31 40 sensores inválidos init info 0 27 80 10 20 31 sensores inválidos load info 99 sala inválida	shutdown 5 sala inexistente update info 1 27 80 10 20 31 40 sala inexistente update info 0 27 80 10 20 31 40 sensores não instalados load info 7 sala inexistente load info 0 sensores não instalados init info 0 27 80 10 20 31 40 sensores inicializados com sucesso init info 0 27 80 10 20 31 40 sensores já instalados

LEMBRETES

- Para usuários Windows, a ferramenta Windows Subsystems for Linux (WSL), nativa do sistema operacional, possibilita operar ambientes Linux no Windows. Seguem os links sobre a ferramenta e tutorial para instalação:
    - https://learn.microsoft.com/pt-br/windows/wsl/about
    - https://learn.microsoft.com/pt-br/windows/wsl/install
- Aos alunos que preferem operar diretamente em ambientes Linux, o ICEx possui duas salas (1006 e 1008) com computadores com sistema operacional Linux disponíveis a todos estudantes da UFMG. Essas salas ficam localizadas no térreo, à direita da portaria principal do ICEx.