

# Trabalho Prático 1

Sistema de controle de Smart Grids

**Execução: Individual**

**Data de entrega: 11 de outubro de 2023 até 23h:59min**

[Introdução](#)

[Protocolo](#)

    Especificação das Mensagens

    Descrição das Funcionalidades

[Implementação](#)

    Execução

[Avaliação](#)

    Entrega

    Prazo de Entrega

    Dicas e Cuidados

[Exemplos de execução](#)

# INTRODUÇÃO

*Smart grid* consiste em um sistema baseado em comunicação e tecnologia da informação adequado para a geração, fornecimento e consumo de energia. Esse sistema trata-se de redes inteligentes que são incorporadas às usinas a fim de recolher e administrar dados e, com base nestas informações coletadas, controlá-las com eficácia. Para tal, as smart grids utilizam do fluxo bidirecional de informações, com intuito de formar um sistema automatizado, amplamente distribuído e disponível de novas funcionalidades. Estas aplicabilidades indicadas são: *controle, competência operacional, resiliência da rede e uma melhor integração de tecnologias renováveis*.

O Sistema de Supervisão e Controle (SCADA), ao ser incorporado a *smart grid*, possui o encargo de efetuar as coletas, supervisão e administração dos dados. Os dados obtidos geralmente referem-se a valores de medidas e status dos diversos componentes da rede, tornando o sistema uma parte fundamental do setor elétrico, mediante a sua capacidade de cobrir grandes áreas e executar comunicações em tempo real. O SCADA é amplamente utilizado para supervisionar e monitorar continuamente infraestruturas críticas, como redes de distribuição de água, usinas de geração e distribuição de eletricidade, refinarias de petróleo, usinas nucleares e sistemas de transporte público.

Uma empresa de controle e automação decidiu ingressar no segmento de controle em *Smart Grids* e necessita desenvolver um projeto piloto que consiste em criar um controle de informações de rede capaz de gerenciar o estado de uma rede elétrica através de uma unidade de controle centralizada, responsável por todo o fluxo de informações e controles no sistema, a Unidade Terminal Principal, do inglês *Main Terminal Unit* - MTU (**o servidor**) e de uma Unidade Terminal Remota, do inglês *Remote Terminal Unit* - RTU (**o cliente**) cuja função é coletar dados dos dispositivos em campo, como sensores, e retornar à MTU por meio de protocolos de comunicação. Esta comunicação ocorre através da Internet. A Figura 1 ilustra a comunicação entre cada uma das entidades do projeto (sensores, RTU e MTU).

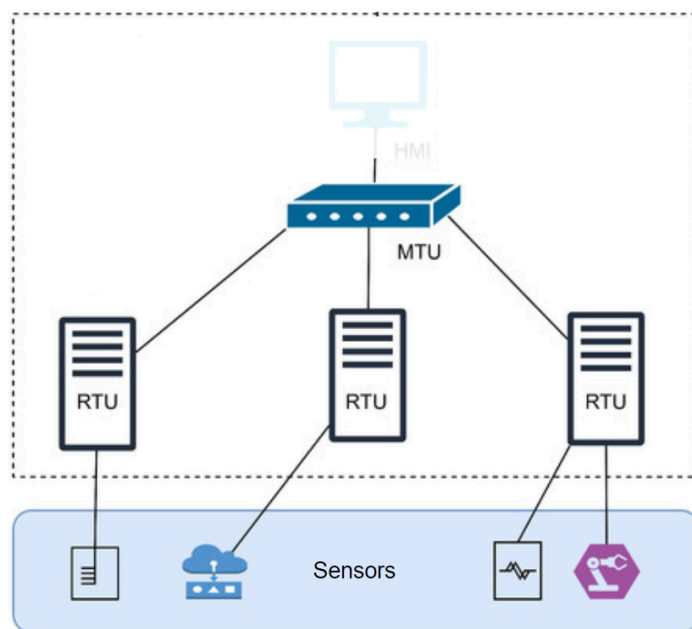


Figura 1 - Exemplo de comunicação entre as entidades do projeto

O programa cliente (RTU) coleta dados de corrente elétrica, de tensão elétrica e de eficiência energética de sensores, os quais são descritos na **Tabela I** e envia os valores para o servidor (MTU) que por sua vez calcula e armazena a potência elétrica junto com a eficiência energética. Para que as mudanças aconteçam, o programa servidor espera receber os comandos do cliente, o qual informa os estados dos dispositivos monitorados pelos sensores, por exemplo:

**Tabela I - Dados coletados pelos sensores**

ID	Corrente = [0,10] A	Tensão = [0,150] V	Eficiência Energética = [0,100] %
0	5 A	100 V	30%
1	2 A	30 V	100%
2	7 A	11 V	70%

Diante do exposto, neste trabalho, você será responsável por desenvolver um **sistema em rede cliente-servidor** para simular a interação entre a **Unidade Terminal Principal - MTU (o servidor)** e a **Unidade Terminal Remota - RTU (o cliente)**. A MTU deve atender às seguintes solicitações da RTU:

1. **Ligar Sensor:** Inicia coleta de dados de um sensor.
2. **Desligar Sensor:** Encerra a coleta de dados de um sensor.
3. **Atualizar Informações de Sensor:** Altera os valores de dados de um sensor.
4. **Consultar Informações de Sensor:** Informa os atuais valores de um sensor.
5. **Consultar Tabela de Sensores:** Informa os valores armazenados de todos os sensores.

Cada uma dessas solicitações correspondem a uma mensagem enviada pela RTU à MTU. Neste trabalho, **você deve implementar os cinco 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 MTU (o servidor) utilizando a interface de sockets na linguagem C;
2. Implementar RTU (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 MTU e a RTU 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	Sensor info	Descrição	Exemplo
INS_REQ	<b>sensorId</b> cor ten efic_energ	Mensagem de solicitação de iniciar sensor	INS_REQ 1 2 30 100
REM_REQ	<b>sensorId</b>	Mensagem de solicitação de desligar sensor	REM_REQ 2
CH_REQ	<b>sensorId</b> cor ten efic_energ	Mensagem de solicitação de alteração de valores de sensor	CH_REQ 2 6 30 80
SEN_REQ	<b>sensorId</b>	Mensagem de solicitação de informações de sensor	SEN_REQ 2
SEN_RES	<b>sensorId</b> <sub>1</sub> pot <sub>1</sub> efic_ener <sub>1</sub>	Mensagem de resposta de informações de sensor	<b>sensor 0</b> : 500 30
VAL_REQ	–	Mensagem de solicitação de valores de sensores	VAL_REQ
VAL_RES	<b>sensorId</b> <sub>1</sub> pot <sub>1</sub> efic_ener <sub>1</sub> <b>sensorId</b> <sub>2</sub> pot <sub>2</sub> efic_ener <sub>2</sub> ...	Mensagem de resposta de valores de sensores	<b>sensors</b> : 1 (60 100) 2 (77 70)

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 : sensor not installed	ERROR 01

		02 : no sensors 03 : invalid sensor 04 : sensor already exists	
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 : successful installation 02 : successful removal 03 : successful change	OK 02

## DESCRIÇÃO DAS FUNCIONALIDADES

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

### 1) Ligar Sensor

1. A RTU recebe comando via teclado  

```
install file nome_arquivo
```

```
install param sensorId cor ten effic_energ
```

para a instalação do sensor **sensorId** de valores **corrente<sub>id</sub>**, **tensao<sub>id</sub>**, **eficiencia\_energetica<sub>id</sub>**. A RTU verifica se esses valores estão de acordo com o padrão de entrada especificado na **TABELA I**.
  - 1.1. Em caso negativo, a RTU imprime a mensagem de erro código 03 (vide *Especificação das Mensagens*) e cancela a ação.
  - 1.2. Em caso positivo, a RTU envia a mensagem **INS\_REQ** para a MTU.
2. A MTU recebe solicitação e verifica se o sensor existe em **TABELA I**.
  - 2.1. Em caso positivo, a MTU responde com mensagem de erro código 04.
    - 2.1.1. A RTU recebe código de erro e imprime sua descrição em tela.
  - 2.2. Em caso negativo, a MTU calcula a potência elétrica ( $P = U \cdot i$ ), adiciona o sensor à **TABELA I** e responde a mensagem de confirmação código 01.
    - 2.2.1. A RTU recebe código de confirmação e imprime sua descrição em tela.

### 2) Desligar Sensor

1. A RTU recebe comando via teclado  

```
remove sensorId
```

para a remoção do sensor **sensorId**. Para isso, a RTU envia a mensagem **REM\_REQ** para a MTU.
2. A MTU recebe solicitação e verifica se o sensor existe em **TABELA I**.
  - 2.1. Em caso negativo, a MTU responde com mensagem de erro código 01.
    - 2.1.1. A RTU recebe código de erro e imprime sua descrição em tela.
  - 2.2. Em caso positivo, a MTU remove o sensor e responde com mensagem de confirmação código 02.
    - 2.2.1. A RTU recebe código de confirmação e imprime sua descrição em tela.

### 3) Atualizar Informações de Sensor

1. A RTU recebe comando via teclado  

```
change file nome_arquivo
```

change param **sensorId** cor ten efic\_energ

para a alteração dos valores do sensor **sensorId** para **corrente<sub>id</sub>** **tensao<sub>id</sub>** **eficiencia\_energetica<sub>id</sub>**. A RTU verifica se esses valores estão de acordo com o padrão de entrada especificado na TABELA I.

- 1.1. Em caso negativo, a RTU imprime a mensagem de erro código 03 e cancela a ação.
- 1.2. Em caso positivo, a RTU envia a mensagem **CH\_REQ** para a MTU.
2. A MTU recebe solicitação e verifica se o sensor existe em TABELA I.
  - 2.1. Em caso negativo, a MTU responde com mensagem de erro código 01.
    - 2.1.1. A RTU recebe código de erro e imprime sua descrição em tela.
  - 2.2. Em caso positivo, a MTU calcula a potência elétrica ( $P = U \cdot i$ ), atualiza os valores do sensor e responde com mensagem de confirmação código 03.
    - 2.2.1. RTU recebe código de confirmação e imprime sua descrição em tela.

#### 4) Consultar Informações de Sensor

1. A RTU recebe comando via teclado  
**show value sensorId**  
para mostrar os valores do sensor **sensorId**. Para isso, a RTU envia a mensagem **SEN\_REQ** para a MTU.
2. A MTU recebe solicitação e verifica se o sensor existe em TABELA I.
  - 2.1. Em caso negativo, a MTU responde com mensagem de erro código 01.
    - 2.1.1. A RTU recebe código de erro e imprime sua descrição em tela.
  - 2.2. Em caso positivo, a MTU responde a RTU com os valores atuais do sensor por meio da mensagem **SEN\_RES**.
    - 2.2.1. A RTU recebe mensagem e imprime em tela:  
**sensor sensorId: pot<sub>id</sub>** **eficiencia\_energetica<sub>id</sub>**

#### 5) Consultar Tabela de Sensores

1. A RTU recebe comando via teclado  
**show values**  
para mostrar os valores de sensores instalados. Para isso, a RTU envia a mensagem **VAL\_REQ** para a MTU.
2. A MTU recebe a solicitação e verifica se existem sensores na Tabela I.
  - a. Em caso negativo, a MTU responde com mensagem de erro código 02.
    - i. RTU recebe código de erro e imprime sua descrição em tela.
  - b. Em caso positivo, a MTU responde com os valores dos sensores correntemente instalados por meio da mensagem **VAL\_RES**.
    - i. RTU recebe mensagem e imprime em tela:  
**sensors: sensorId<sub>1</sub> (pot<sub>1</sub>** **eficiencia\_energetica<sub>1</sub>) sensorId<sub>2</sub> (pot<sub>2</sub>**  
**eficiencia\_energetica<sub>2</sub>) ...**

## 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 (MTU) quanto uma versão do cliente (RTU).

### 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: “instal” em vez de “install”), 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 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 90900 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 90900

Terminal 2: ./client 127.0.0.1 90900

## 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**):

- Instalar sensor: **+2 pontos**
- Remover sensor: **+2 pontos**
- Alterar estado do sensor: **+2 pontos**
- Consultar valores do sensor: **+3 pontos**
- Consultar valores de sensores: **+3 pontos**
- Testar casos de mensagem inválida: **+2 pontos**
- 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 úteis. 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 (Instalação e Remoção)	Exemplo 2 (Atualizar Informações de Sensor)
<pre>install file file1.txt successful installation install param 2 7 11 70 successful installation remove 2 successful removal</pre>	<pre>install file file0.txt successful installation change param 0 6 93 32 successful change change file file0.txt successful change</pre>

Exemplo 3 (Consultar Informações de Sensor)	Exemplo 4 (Consultar Tabela de Sensores)
<pre>install param 0 5 90 30 successful installation install file file4.txt successful installation show value 0 sensor 0: 450 30 show value 4 sensor 4: 66 57</pre>	<pre>install param 2 7 11 70 successful installation install param 0 5 90 30 successful installation show values sensors: 2 (77 70) 0 (450 30)</pre>

Exemplo 5 (Tratamento de erros)	Exemplo 6 (Tratamento de erros)
<pre>install param 1 100 2 3 invalid sensor install param 0 1 2 invalid sensor show value 99 sensor not installed</pre>	<pre>show values no sensors</pre>

## 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.