Trabalho Prático 1

Sistema controle de dispositivos em Smart Homes

Execução: **Individual**
Data de entrega: ~~**17**~~ **23 de outubro de 2022 até 23h:59min**

INTRODUÇÃO

O desenvolvimento de tecnologias IoT (Internet of Things) está promovendo grandes oportunidades em inovações no mercado. Dentre os segmentos mais beneficiados nestes avanços estão as aplicações em Smart Homes (Casas “Inteligentes”), em que sistemas em rede gerenciam o “estado” das residências a fim de adaptá-las da melhor forma ao contexto dos moradores. Tecnicamente, estes sistemas monitoram e controlam os ambientes domésticos através de sensores e atuadores a fim de controlar diferentes dispositivos conectados em rede como, por exemplo, televisão, lâmpadas, ar condicionados, portas e etc. Desta forma, podemos notar a importância e necessidade das redes de computadores na automação em Smart Homes, tendo em vista que os dispositivos IoT necessitam “conversar” através de um infraestrutura de comunicação para adaptar o estado das residências aos diferentes perfis de usuários.

Uma empresa de controle e automação está ingressando no segmento de automação em Smart Homes e necessita desenvolver um projeto piloto que consiste em criar uma aplicação de rede capaz de adaptar o estado de uma residência através do envio de comandos de uma Aplicação **(o cliente)** para uma Central de Controle **(o servidor)** através da Internet a qual será responsável por monitorar e controlar os dispositivos (lâmpadas, TVs, geladeira, etc) da residência. A Figura 1 ilustra a comunicação entre cada uma das entidades do projeto (dispositivos, aplicação de controle e Aplicação).

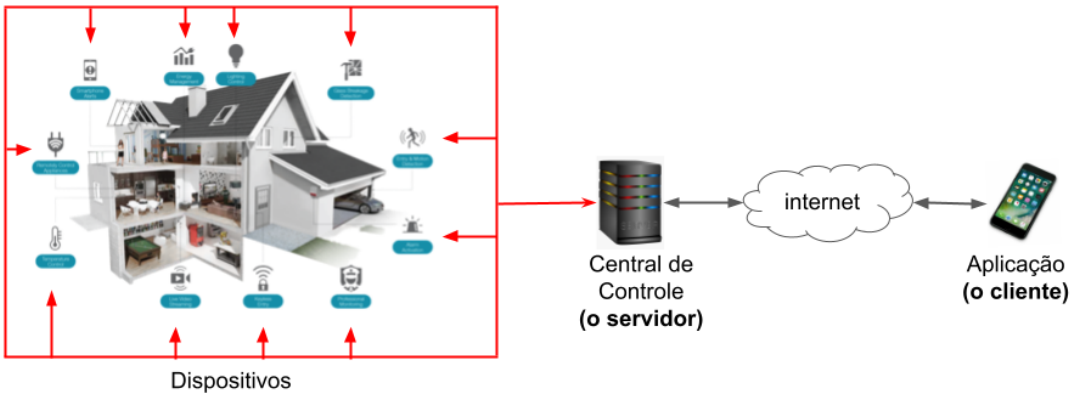

Figura 1 - Exemplo de comunicação de entre as entidades do projeto

O servidor manipula cinco dispositivos da residência por meio da mudança dos valores dos seus estados, os quais são descritos na **Tabela I**. Para que as mudanças aconteçam, o servidor espera receber os comandos da aplicação cliente remota, a qual informa os estados dos dispositivos em diferentes locais da residência. A **Tabela II** apresenta os locais da residência que são controlados pelo servidor.

ID	Dispositivo	Estados
1	Ar Condicionado	Ligado=<Não = 0 | Sim = 1> Temperatura = [0, 40]
2	TV	Ligado=<Não = 0 | Sim = 1> Canal = [1, 150]
3	Lâmpada	Ligado=<Não = 0 | Sim = 1> Local = <Local Id>
4	Porta	Aberto=<Não = 0 | Sim = 1> Trancado=<Não = 0 | Sim = 1>
5	Câmera	Ligado=<Não = 0 | Sim = 1> Ângulo= [0, 359]

Tabela I - Dispositivos e seus respectivos estados

ID	Local
1	Quarto
2	Suite
3	Sala
4	Cozinha
5	Banheiro

Tabela II - Identificação dos locais da residência

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

1. **Instalar Dispositivo:** Vincula dispositivo em um dos locais da residência.
2. **Remover Dispositivo:** Desvincula dispositivo de um local da residência.
3. **Alterar Estado de Dispositivo:** Altera o estado de um dispositivo de um local.
4. **Consultar Estado de Dispositivo:** Informa o estado de um dispositivo de um local.
5. **Consultar Estado do Local:** Informa os estados dos dispositivos correntes em um local da residência.

Cada uma dessas solicitações correspondem a uma mensagem enviada pela Aplicação à Central de Controle. 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 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 Central de Controle (o servidor) utilizando a interface de *sockets* na linguagem C;
2. Implementar Aplicação (o cliente) utilizando a interface de *sockets* na linguagem C;
3. Escrever o relatório.

PROTOCOLO

O protocolo de aplicação deverá funcionar sobre 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 Central de Controle e a Aplicação trocam mensagens curtas de até 500 bytes utilizando o protocolo 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					
Type	**Local Id**	**Device Id**	**Device State**	**Descrição**	**Exemplo**
INS_REQ	LocId	DevId	$value_1$ $value_2$ ...	Mensagem de solicitação de instalação de dispositivo	INS_REQ 2 2 1 30
REM_REQ	LocId	DevId	–	Mensagem de solicitação de remoção de dispositivo	REM_REQ 2 2

CH_REQ	LocId	DevId	$value_1$ $value_2$ …	Mensagem de solicitação de alteração de estado de dispositivo	CH_REQ 2 2 0
DEV_REQ	LocId	DevId	–	Mensagem de solicitação de estado de dispositivo	DEV_REQ 2 2
DEV_RES	–	–	$value_1$ $value_2$ …	Mensagem de resposta de estado de dispositivo	DEV_RES 1 30
LOC_REQ	LocId	–	–	Mensagem de solicitação do estado do local	LOC_REQ 3
LOC_RES	–	–	**$devId_1$** $value_1$ $value_2$ … **$devId_2$** $value_1$ $value_2$ …	Mensagem de resposta do estado do local	DEV_RES **2** 1 30 **1** 1 22

Mensagens de Erro e Confirmação			
Typo	**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 : device not installed 02 : no devices 03 : invalid device 04 : invalid local	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 : successful installation 02 : successful removal 03 : successful change	OK 02

Descrição das Funcionalidades

Esta seção descreve o fluxo de mensagens transmitidas entre a Central de Controle e a Aplicação resultante de cada uma das cinco funcionalidades da aplicação a fim de gerenciar os dispositivos na rede da residência.

1) Instalar Dispositivo

1. A Aplicação recebe comando via teclado
   install **devId** in **locId**: $valor_1$ $valor_2$ $valor_3$ …
   para a instalação do dispositivo **devId** de estado $valor_1$ $valor_2$ $valor_3$ … no local **LocId**. Para isso, a Aplicação envia a mensagem **INS_REQ** para a Central de Controle.
2. A Central de Controle recebe solicitação e verifica se dispositivo existe em **TABELA I**.

2.1. Em caso negativo, a Central de Controle responde com mensagem de erro código 03 (vide *Especificação das Mensagens*).
2.1.1. Aplicação de controle recebe código de erro e imprime sua descrição em tela.
2.2. Em caso positivo, a Central de Controle instala o dispositivo no respectivo local e responde mensagem de confirmação código 01.
2.2.1. Aplicação de controle recebe código de confirmação e imprime sua descrição em tela.

2) Remover Dispositivo

1. A Aplicação recebe comando via teclado
remove **devId** in **locId**
para a remoção do dispositivo **devId** no local **LocId**. Para isso, a Aplicação envia a mensagem **REM_REQ** para a Central de Controle.
2. A Central de Controle recebe solicitação e verifica se dispositivo existe em **TABELA I**.
2.1. Em caso negativo, a Central de Controle responde com mensagem de erro código 03.
2.1.1. Aplicação recebe código de erro e imprime sua descrição em tela.
2.2. Em caso positivo, a Central de Controle verifica se dispositivo encontra-se instalado no respectivo local.
2.2.1. Em caso negativo, a Central de Controle responde com mensagem de erro código 01.
2.2.1.1. Aplicação recebe código de erro e imprime sua descrição em tela.
2.2.2. Em caso positivo, a Central de Controle remove o dispositivo e responde com mensagem de confirmação código 02.
2.2.2.1. Aplicação recebe código de confirmação e imprime sua descrição em tela.

3) Alterar Estado do Dispositivo

1. A Aplicação recebe comando via teclado
change **devId** in **locId**: $valor_1$ $valor_2$ $valor_3$ …
para a alteração do estado do dispositivo **devId** para $valor_1$ $valor_2$ $valor_3$ … no local **locId**. Para isso, a Aplicação envia a mensagem **CH_REQ** para a Central de Controle.
2. A Central de Controle recebe solicitação e verifica se dispositivo existe em **TABELA I**.
2.1. Em caso negativo, a Central de Controle responde com mensagem de erro código 03.
2.1.1. Aplicação recebe código de erro e imprime sua descrição em tela.
2.2. Em caso positivo, a Central de Controle verifica se dispositivo encontra-se instalado no respectivo local.
2.2.1. Em caso negativo, a Central de Controle responde com mensagem de erro código 01.
2.2.1.1. Aplicação recebe código de erro e imprime sua descrição em tela.
2.2.2. Em caso positivo, a Central de Controle altera o estado do dispositivo e responde com mensagem de confirmação código 03.
2.2.2.1. Aplicação recebe código de confirmação e imprime sua descrição em tela.

4) Consultar Estado de Dispositivo

1. A Aplicação recebe comando via teclado
show state **devId** in **locId**
para mostrar o estado do dispositivo **devId** no local **locId**. Para isso, a Aplicação envia a mensagem **DEV_REQ** para a Central de Controle.
2. A Central de Controle recebe solicitação e verifica se dispositivo existe em **TABELA I**.
2.1. Em caso negativo, a Central de Controle responde com mensagem de erro código 03.

2.1.1. Aplicação recebe código de erro e imprime sua descrição em tela.

2.2. Em caso positivo, a Central de Controle verifica se dispositivo encontra-se instalado no respectivo local.

2.2.1. Em caso negativo, a Central de Controle responde com mensagem de erro código 01.

2.2.1.1. Aplicação recebe código de erro e imprime sua descrição em tela.

2.2.2. Em caso positivo, a Central de Controle responde a Aplicação de Comando com o estado atual do dispositivo por meio da mensagem **DEV_RES**.

2.2.2.1. Aplicação recebe mensagem e imprime em tela:

device **devId** in **ComId**: $valor_1$ $valor_2$ $valor_3$ …

5) Consultar Estado do Local

1. A Aplicação recebe comando via teclado

   show state in **locId**

   para mostrar o estado do local **locId**. Para isso, a Aplicação envia a mensagem **LOC_REQ** para a Central de Controle.
2. A Central de Controle recebe a solicitação e verifica se o local existe na **Tabela II**.
   a. Em caso negativo, a Central de Controle responde com mensagem de erro código 04.
      i. Aplicação recebe código de erro e imprime sua descrição em tela.
3. Em caso positivo, a Central de Controle verifica se existem dispositivos instalados no respectivo local.
   a. Em caso negativo, a Central de Controle responde com mensagem de erro código 02.
      i. Aplicação recebe código de erro e imprime sua descrição em tela.
   b. Em caso positivo, a Central de Controle responde com os estados dos dispositivos correntemente instalados no local por meio da mensagem **LOC_RES**.
      i. Aplicação recebe mensagem e imprime em tela:

      local **locId**: **$devId_1$** ($valor_1$ $valor_2$ …) **$devId_2$** ($valor_1$ $valor_2$ …) …

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 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 (Central de Controle) quanto uma versão do cliente (Aplicação).

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 servidor deve desconectar do cliente 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 51511 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 51511
Terminal 2: ./client 127.0.0.1 51511
```

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 dispositivo: **+2 pontos**
- Remover dispositivo: **+2 pontos**
- Alterar estado do dispositivo: **+2 pontos**
- Consultar estado do dispositivo: **+3 pontos**

- Consultar estado de local: **+3 pontos**
- Testar casos de mensagem inválida: **+2 pontos**
- Testar casos de dispositivos ou local inválidos: **+1 pontos**
- Cliente envia kill para o servidor e encerrar a execução: **+1 pontos**

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 (Alterar Estado de Dispositivo)**
install 2 in 3: 1 35 successful installation install 6 in 4: 1 27 successful installation remove 2 in 3 successful removal	install 2 in 3: 1 35 successful installation change 2 in 3: 0 1 successful change

Exemplo 3 (Consultar Estado de Dispositivo)	**Exemplo 4 (Consultar Estado do Local)**
install 2 in 3: 1 35 successful installation install 6 in 4: 1 27 successful installation show state 2 in 3 device 2 in 3: 1 35 show state 2 in 3 device 6 in 4: 1 27	install 2 in 3: 1 35 successful installation install 6 in 3: 1 27 successful installation show state in 3 local 3: 2 (1 35) 6 (1 27)

Exemplo 5 (Tratamento de erros)	**Exemplo 4 (Tratamento de erros)**
install 6 in 3: 1 35 invalid device show state 2 in 3 device not installed	show state in 8 invalid local show state in 2 no devices