PROVA PRÁTICA # 1
CLIENTE/SERVIDOR

Vilmey Francisco Romano Filho - 11/0021380

Programa de Graduação em Engenharia Eletrônica, Faculdade do Gama
Universidade de Brasília
Gama, DF, Brasil
email: vilmeyr@gmail.com

1. OBJETIVO

Esta prova prático tem o objetivo de exemplificar e reunir técnicas aprendidas em sala de aula, como threds, sockets, pipes dentre outras ferramentas de programação. Nesta prova damos ênfase em sockets e programação em rede utilizando o protocolo TCP/IP para a elaboração de um cliente, capaz de receber requerimentos do usuário e um servidor capaz de se comunicar com um arduino (via UART).

2. INTRODUÇÃO

O microcontrolador ATmega possui muitas ferramentas úteis ao engenheiro, este controlador pode se communicar com outros dispositivos atravéz de protocolo de comunicação serial UART(Universal asynchronous receiver/transmitter). Para a comunicação entre o servidor (Raspberry Pi) e o microcontrolator esse protocolo se mostra eficiente, pois ambos não tem que compartilhar um sinal de clock. Uma vez estabelecida essa comunicação inicial o servidor deve ser capaz de receber requerimentos do cliente e tratá-los, pegando informações sobre temperatura, ligar ou desligar o dispositivos elétricos e controlar um sistema de alarme, junto ao controlador, e enviar a resposta ao cliente.
Já o código do cliente deve receber comandos do usuário via teminal e fazer a requisição junto ao servidor, posteriomente atualizar os dados mostrados ao usuário.
Sockets foram utilizados para criar o códigos tanto do cliente quanto do servidor, e a comunicação adotada foi o protocolo TCP/IP o qual é orientado a conexão assim os pacotes enviados são recebidos ordenadamente, evitando overhead e simplificando a troca de informações.

3. ESPECIFICAÇÃO

3.1. Cliente

O código do cliente possui uma interface gráfica rudimentar no terminal na qual oferece ao usuário as opções apresentadas nas tabelas [1 e 2]. A escolha das funções foi dividida em setores de temoeratura, presença, portas e janelas, ar-condicionado, lâmpadas e alarme.

Ao digitar o comando corretamente o cliente se conecta ao servidor na porta especifica e envia dados de acionameto de equipameto ou faz um requerimento de dados de temperatura e presença.

O protocolo de comunicação entre o cliente e servidor funciona da seguinte maneira. Todos os dados enviados ao servidor são do tipo inteiro. E a resposta do servidor depende do tipo de *request*, para temperatura, *float* e para estatus (ligado/desligado) e presença o servidor retorna um *char*.

Código	Sensor	Localização	Mensagem de Retorno
0xA1	Temperatura	1 = Externa 2 = Interna da Sala 3 = Interna do Quarto	float (4 bytes)
0xA2	Presença	1 = Entrada Principal 2 = Entrada de serviço 3 = Garagem	char (1 byte) - onde: 1 = presença detectada 0 = não há presença
0xA3	Portas / Janelas	1 = Porta principal 2 = Porta de serviço 3 = Porta da garagem 4 = Janela da sala 5 = Janela do Quarto 6 = Janela da cozinha	char (1 byte) - onde: 1 = fechado 0 = aberto

Fig. 1. Comandos disponiveis para envio e tipo de retorno.

Código	Dispositivo	Localização	Mensagem de Retorno
0xB1	Ar-condicionado	1 = Sala 2 = Quarto	char (1 byte) onde: 1 = Liga 2 = Desliga
0xB2	Lâmpada	1 = Entrada principal (externa) 2 = Entrada de serviço (externa) 3 = Garagem 4 = Sala 5 = Quarto 6 = Cozinha	char (1 byte) onde: 1 = Liga 2 = Desliga
0xB3	Alarme	1 = Setor 1 (Sensores de Presença) 2 = Setor 2 (Sensores de Portas/ Janelas)	char (1 byte) onde: 1 = Liga 2 = Desliga

Fig. 2. Comandos para envio e tipo de retorno.

3.2. Servidor

O servidor é responsável por receber os requerimentos do cliente em uma porta específica (alarme 4034, demais 8034), enviar e receber os dados do microcontrolador via UART. É importante que o servidor respeite os limites de velocidade de comunicação (9600 bauds). Após a comunicação com controlador o servidor pode enviar mensagem de resposta ao cliente, esta mensagem pode ser uma temperatura ou uma resposta a requisição de ligar ou desligar dispositivos.

4. IMPLEMENTAÇÃO

4.1. Cliente

A tela apresentada ao usuário é apresentado abaixo na figura[3]. O programa captura as entradas do teclado numérico para prosseguir no menu de opções.

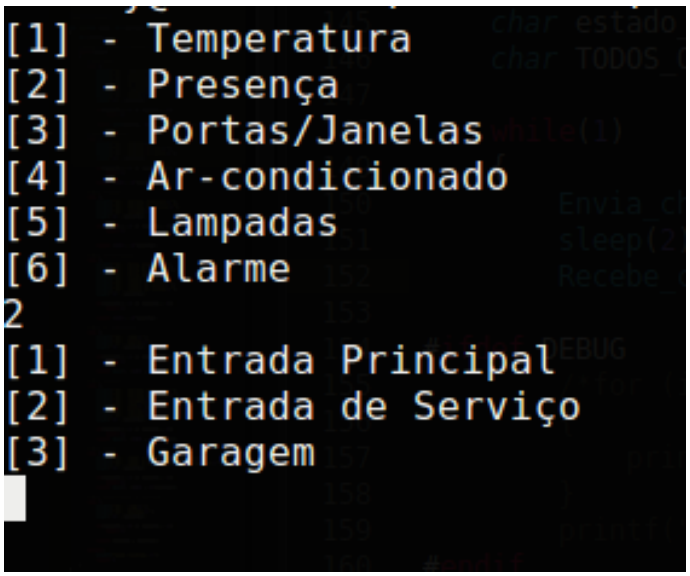

Fig. 3. Tela de comando.

A leitura do teclado se encontra dentro da função menu, que é responsável tembém por imprimir na tela as opções. Se o comando for válido chamamos a função *open_socket* com o numero da porta específica e a opção desejada.

E finalmente, o comando digitado pelo usuário é enviado ao servidor, a resposta do servidor é armazenada em uma variável de tipo especifico a requisição.

4.2. Servidor

Na finção *Log_data* criamos um arquivo no formato .txt que armazena a data, horário, IP do cliente conectado e operação requisitada (HH:MM:SS - DD/MM/AAAA - Cliente $< IP >$; Setor $< 1 - 6 >$; Dispositivo $< X >$; Estado $< 0 - 1 >$). Foi necessário adicionar o header *¡time.h¿* para que a *struct* responsável pela data e hora funcionasse adequadamente.

A configuração da UART figura[4] é um dos passos mais importantes do processo, pois sem ela não é possivel se comunicar eficientemente com o controlador. Após termos a nossa UART devidamente configurada, criamos uma thread de servidor para cada porta, pois cada porta terá diferentes tipos de serviços.

```
//Configura os parametros da comunicacao UART
void Uart_Config(int uart0_filestream)
{
    struct termios options;
    tcgetattr(uart0_filestream, &options);
    options.c_cflag = B9600 | CS8 | CLOCAL | CREAD; // Set baud rate
    options.c_iflag = IGNPAR;
    options.c_oflag = 0;
    options.c_lflag = 0;
    tcflush(uart0_filestream, TCIFLUSH);
    tcsetattr(uart0_filestream, TCSANOW, &options);
}
```

Fig. 4. Configuração UART.

Os servidores ficarão esperando em suas respectivas portas pela conexão do cliente, e uma vez que o cliente se conecte, pegamos o seu IP que se encontra dentro do banco de informações do cliente (struct). Esta informaçao será adicionada ao log do sistema, juntamente com o comando recebido.

Na função TrataClienteTCP revebemos o pacote de informações do cliente, o separamos em Setor, qual Dispositvo, e Estado (ligado, desligado). Chamamos a função enviaArduino que fará a comunicação com o controlador, e se for o caso enviará a resposta ao cliente por meio das funçãoes Envia_char_cliente e Envia_float_cliente.

Quando o alarme é setado a thread Verifica_sensor varre os sensores do setor específico (setor 1 e/ou 2), e caso detecte a presença enquanto o alarme estiver acionado, um alarme é tocado com a função play_alarm_music. O alarme somente será desligado quando o cliente enviar o comando desabilitando a função.

5. CONCLUSÕES

A prova prática foi contrutiva no tocante a parte de comunicação entre dispositivos, pois a mesma englobou diferentes protocolos e dispositivos. Esta prova foi uma aplicação prática do conhecimento e se mostrou muito interessante para futuras aplicações em sistemas de monitoramento remoto de qualquer tipo, sendo limitada apenas pelo hardware do dispositivo utilizado.

6. ANEXO - CÓDIGOS

Os códigos abaixo possuem comentários que ajudam a compreensão do trabalho desenvolvido na comunicação entre servidor, cliente e microcontrolador.

```c
//Compilação//
//      gcc cliente_tcp.c -o cliente_tcp

//Execução//
// ./cliente_tcp <ip do servidor>

//-----------------------------------------------------------------------------------------------------------------
//Includes do programa//
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <pthread.h>
#include <termios.h>
#include <fcntl.h>

//-----------------------------------------------------------------------------------------------------------------
//Variáveis globais//
unsigned short PORTA_ALARME = 4034;
unsigned short PORTA_GERAL  = 8034;


//-----------------------------------------------------------------------------------------------------------------
//Menu do Cliente//

int menu(int opcao, int recursao)
{
        int escolha;

        //Opções de Escolha do Cliente//
        switch(opcao)
        {
                //Menu geral
                case 0:
                        printf("[1] - Temperatura\n");
                        printf("[2] - Presença\n");
                        printf("[3] - Portas/Janelas\n");
                        printf("[4] - Ar-condicionado\n");
                        printf("[5] - Lampadas\n");
                        printf("[6] - Alarme\n");
                        scanf("%d", &escolha);

                        if((escolha < 1) || (escolha > 6))
                                return 0;

                        return menu(escolha, 0);                        //Retorna a propria função

                //Menu temperatura
                case 1:
                        printf("[1] - Externa\n");
                        printf("[2] - Interna da Sala\n");
                        printf("[3] - Interna do quarto\n");
                        scanf("%d", &escolha);

                        if((escolha < 1) || (escolha > 3))
                                return 0;

                        return (1000 + 10*escolha);

                //Menu Presenca
                case 2:
                        printf("[1] - Entrada Principal\n");
                        printf("[2] - Entrada de Serviço\n");
                        printf("[3] - Garagem\n");
                        scanf("%d", &escolha);
```

```
68
69                          if((escolha < 1) || (escolha > 3))
70                                  return 0;
71
72
73                          return (2000 + 10*escolha);
74
75                  //Menu Portas/Janelas
76                  case 3:
77                          printf("[1] - Porta principal\n");
78                          printf("[2] - Porta de serviço\n");
79                          printf("[3] - Porta garagem\n");
80                          printf("[4] - Janela da sala\n");
81                          printf("[5] - Janela do quarto\n");
82                          printf("[6] - Janela da cozinha\n");
83                          scanf("%d", &escolha);
84
85                          if((escolha < 1) || (escolha > 6))
86                                  return 0;
87
88                          return (3000 + 10*escolha);
89
90                  //Menu Ar-condicionado
91                  case 4:
92                          printf("[1] - Sala\n");
93                          printf("[2] - Quarto\n");
94                          scanf("%d", &escolha);
95
96                          if((escolha < 1) || (escolha > 2))
97                                  return 0;
98
99                          return menu(7, (4000 + 10*escolha));            //Retorna o menu, na opção
    estado (Ligado/Desligado)
100
101                 //Menu Lampadas
102                 case 5:
103                         printf("[1] - Entrada principal\n");
104                         printf("[2] - Entrada de serviço\n");
105                         printf("[3] - Garagem\n");
106                         printf("[4] - Sala\n");
107                         printf("[5] - Quarto\n");
108                         printf("[6] - Cozinha\n");
109                         scanf("%d", &escolha);
110
111                         if((escolha < 1) || (escolha > 6))
112                                 return 0;
113                         return menu(7, (5000 + 10*escolha));
114
115                 //Menu Alarme
116                 case 6:
117                         printf("[1] - Setor 1 (Sensores de Presença)\n");
118                         printf("[2] - Setor 2 (Sensores de Portas/ Janelas)\n");
119                         scanf("%d", &escolha);
120
121                         if((escolha < 1) || (escolha > 2))
122                                 return 0;
123                         return menu(7, (6000 + 10*escolha));
124
125                 //Menu Estado Ligado ou Desligado
126                 case 7:
127                         printf("[1] - Liga\n");
128                         printf("[0] - Desliga\n");
129                         scanf("%d", &escolha);
130
131                         if((escolha < 0) || (escolha > 1))
132                                 return 0;
133
```

```
134                             return (escolha + recursao);
135
136                     default:
137                             printf("Opção Inválida.\n");
138                             return 0;
139             }
140     }
141
142     //-------------------------------------------------------------------------------------------------------
143     //Funções de envio e recebimento de dados//
144
145     //Recebe float do servidor
146     void receive_float(int *clienteSocket)
147     {
148             float temperatura;
149
150             if((recv(*clienteSocket, &temperatura, sizeof(float), 0)) < 0)
151                     printf("Erro no recv()\n");
152
153             printf("Temperatura %f\n", temperatura);
154
155             close(*clienteSocket);
156     }
157
158     //Recebe char do servidor
159     void receive_char(int *clienteSocket)
160     {
161             char mensagem;
162
163             if((recv(*clienteSocket, &mensagem, sizeof(char), 0)) < 0)
164                     printf("Erro no recv()\n");
165
166             printf("Estado %c\n", mensagem);
167
168             close(*clienteSocket);
169     }
170
171     //Envia um request de para o servidor
172     void request(int *clienteSocket, int mensagem)
173     {
174             int bytesRecebidos;
175             int buffer;
176
177             if(send(*clienteSocket, &mensagem, sizeof(int), 0) != sizeof(int))
178                     printf("Erro no envio: numero de bytes enviados diferente do esperado\n");
179
180     }
181
182
183     //-------------------------------------------------------------------------------------------------------
184     //Abertura do socket//
185     void socket_open(int *clienteSocket, struct sockaddr_in *servidorAddr, char *IP_Servidor, unsigned
        short servidorPorta)
186     {
187             // Criar Socket
188             if((*clienteSocket = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
189                     printf("Erro no socket()\n");
190
191             // Construir struct sockaddr_in
192             memset(servidorAddr, 0, sizeof(*servidorAddr)); // Zerando a estrutura de dados
193             servidorAddr->sin_family = AF_INET;
194             servidorAddr->sin_addr.s_addr = inet_addr(IP_Servidor);
195             servidorAddr->sin_port = htons(servidorPorta);
196
197             // Connect
198             if(connect(*clienteSocket, (struct sockaddr *) servidorAddr, sizeof(*servidorAddr)) < 0)
199                     printf("Erro no connect()\n");
```

```
200
201  }
202
203
204  //-----------------------------------------------------------------------------------------------------------------
205  //Função principal//
206
207  int main(int argc, char *argv[]) {
208
209          //Variaveis locais//
210          int clienteSocket;
211          int escolha;
212          struct sockaddr_in servidorAddr;
213          unsigned short servidorPorta = PORTA_GERAL;
214          char *IP_Servidor;
215
216          if ((argc < 2) | (argc > 3))
217          {
218                  printf("Uso: %s <IP do Servidor>\n", argv[0]);
219                  exit(1);
220          }
221
222          //Ip via linha de comando
223          IP_Servidor = argv[1];
224
225
226          while(1)
227          {
228                  if((escolha = menu(0, 0)) == 0)
229                          printf("Escolha invalida\n");
230
231
232                  else
233                  {
234                          system("clear");
235
236                          //Define a porta a ser conectado//
237                          if(escolha > 6000)
238                                  servidorPorta = PORTA_ALARME;
239
240                          else
241                                  servidorPorta = PORTA_GERAL;
242
243                          //Mostra a porta utilizada e a opção espolhida
244                          printf("|Escolha %d, Porta %d|\n", escolha, servidorPorta);
245
246                          //Abrimos a conexão
247                          socket_open(&clienteSocket, &servidorAddr, IP_Servidor, servidorPorta);
248
249                          //Enviamos um dado do menu
250                          request(&clienteSocket, escolha);
251
252                          //Recebemos o retorno do sensor, de acordo com o dado requerido
253                          if((escolha > 1000) && (escolha < 2000))
254                                  receive_float(&clienteSocket);
255
256                          if((escolha > 2000) && (escolha < 4000))
257                                  receive_char(&clienteSocket);
258
259                          //Fechamos a conexão com o servidor
260                          close(clienteSocket);
261
262                  }
263
264          }
265
266          exit(0);
```

```
267 }
```

```c
/*Compilação*/
//      gcc servidor_tcp.c play_alarm.c -o servidor_tcp

/*Execução*/
// ./servidor_tcp

//-----------------------------------------------------------------------------------------------------------------
/*Includes do programa*/
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <termios.h>
#include <fcntl.h>
#include <time.h>
#include <signal.h>
#include "play_alarm.h"

//-----------------------------------------------------------------------------------------------------------------
/*Defines*/
#define DEVICE_FILE             "/dev/ttyAMA0"
#define DEBUG


//-----------------------------------------------------------------------------------------------------------------
/*Variáveis globais*/
char alarm_setor[] = {0, 0};
pid_t alarm_pid;
int uart0_filestream = -1;
unsigned short porta_4000 = 4034;
unsigned short porta_8080 = 8034;


//-----------------------------------------------------------------------------------------------------------------
/*Função de Log do sistema*/

void Log_data(char *clienteAddr, int setor, int dispositivo, int estado)
{
        FILE *fd;                                                       //Descritor do arquivo

    time_t tempo;                                               //Struct da biblioteca time.h
        struct tm *data;
        time(&tempo);
        data=localtime(&tempo);

        fd = fopen("Client_log.txt", "a+");     //Abre o arquivo para incluir

        //HH:MM:SS - DD/MM/AAAA - Cliente <IP>; Setor <1-6>; Dispositivo <X>; Estado <0-1>
        fprintf(fd, "%d:%d:%d - %d/%d/%d",(*data).tm_hour,(*data).tm_min,(*data).tm_sec,
(*data).tm_mday,(*data).tm_mon+1,(*data).tm_year+1900);
        fprintf(fd, " - Cliente %s; Setor %d; Dispositivo %d; Estado %d\n", clienteAddr, setor,
dispositivo, estado);

        fclose(fd);                                                     //Fecha o arquivo
}


//-----------------------------------------------------------------------------------------------------------------
/* Comunicação UART*/

//Configura os parametros da comunicacao UART
void Uart_Config(int uart0_filestream)
{
        struct termios options;
```

```
66          tcgetattr(uart0_filestream, &options);
67          options.c_cflag = B9600 | CS8 | CLOCAL | CREAD; // Set baud rate
68          options.c_iflag = IGNPAR;
69          options.c_oflag = 0;
70          options.c_lflag = 0;
71          tcflush(uart0_filestream, TCIFLUSH);
72          tcsetattr(uart0_filestream, TCSANOW, &options);
73  }
74
75  //Recebimento de float via UART
76  void Recebe_float (int uart0_filestream, float *retorno)
77  {
78          float temperatura;
79          uart0_filestream = open(DEVICE_FILE, O_RDWR | O_NOCTTY |O_NDELAY);
80
81          int rx_length = read(uart0_filestream, &temperatura, sizeof(float));
82                  if (rx_length < 0)
83                          perror("Falha na leitura");
84
85                  else if (rx_length == 0)
86                          printf("Nenhum dado disponivel\n");
87
88                  else
89                          printf("\n|%i bytes lidos|", rx_length);
90
91
92          close(uart0_filestream);
93
94  #ifdef DEBUG
95
96          printf("\n\nRetorno float: %f\n\n\n", temperatura);
97          *retorno = temperatura;
98
99  #endif
100 }
101
102 //Recebimento de char via UART
103 void Recebe_char (char * retorno, int tamanho, int uart0_filestream)
104 {
105         int i;
106
107         uart0_filestream = open(DEVICE_FILE, O_RDWR | O_NOCTTY |O_NDELAY);
108         int rx_length = read(uart0_filestream, retorno, tamanho);
109
110         if (rx_length < 0)
111                 perror("Falha na leitura");
112
113         else if (rx_length == 0)
114                 printf("Nenhum dado disponivel\n");
115
116         else
117                 printf("| %i bytes lidos |", rx_length);
118
119         close(uart0_filestream);
120
121 }
122
123 //Envio de char via UART
124 void Envia_char (char *Comando, int tamanho, int uart0_filestream)
125 {
126
127 #ifdef DEBUG
128         printf("Comando : %X %d %X\n", Comando[0], Comando[1], Comando[2]);
129 #endif
130
131         uart0_filestream = open(DEVICE_FILE, O_RDWR | O_NOCTTY |O_NDELAY);
132         int tx_length = write(uart0_filestream, Comando, tamanho);
```

```
133                 if (tx_length != tamanho)
134                         printf("Erro na transmissao - UART TX\n");
135
136         close(uart0_filestream);
137 }
138
139 //-------------------------------------------------------------------------------------------------------------
140 /*Thead de verificação dos sensores de presença, portas e janelas*/
141
142 void *Verifica_sensor()
143 {
144         int i;
145         char estado_sensores[9];
146         char TODOS_OS_SENSORES = 0xA4;
147
148         while(1)
149         {
150                 Envia_char (&TODOS_OS_SENSORES, 1, uart0_filestream);
151                 sleep(2);
152                 Recebe_char (estado_sensores, 9, uart0_filestream);
153
154 #ifdef DEBUG
155                 /*for (i = 0; i < 9; i++)
156                 {
157                         printf("S[%d] %d, ",i, estado_sensores[i] );
158                 }
159                 printf("\n");*/
160 #endif
161
162                 /*Monitoramento do Setor 1*/
163                 if(alarm_setor[0] == 1)
164                 {
165                         for (i = 0; i < 3; i++)
166                         {
167                                 if(estado_sensores[i] != 0)
168                                 {
169                                         //Toca Alarme
170                                         printf("Toca o ALARME !!!\n");
171                                         play_alarm_music(&alarm_setor[0]);
172                                 }
173                                 else
174                                         sleep(2);
175
176                         }
177                 }
178
179                 /*Monitoramento do Setor 2*/
180                 if(alarm_setor[1] == 1)
181                 {
182                         for (i = 3; i < 9; i++)
183                         {
184                                 if(estado_sensores[i] != 0)
185                                 {
186                                         //Toca Alarme
187                                         printf("Toca o ALARME !!!\n");
188                                         play_alarm_music(&alarm_setor[1]);
189                                 }
190                                 else
191                                         sleep(2);
192
193                         }
194                 }
195         }
196 }
197
198
199 //-------------------------------------------------------------------------------------------------------------
```

```
200  /*Funções de envio de dados ao cliente*/
201
202  void Envia_float_cliente(int socketCliente, struct sockaddr_in clienteAddr, float temperatura) {
203
204          if(send(socketCliente, &temperatura, sizeof(float), 0) != sizeof(float))
205                  printf("Erro no envio - send()\n");
206
207  }
208
209  void Envia_char_cliente(int socketCliente, struct sockaddr_in clienteAddr, char mensagem) {
210
211          if(send(socketCliente, &mensagem, sizeof(char), 0) != sizeof(char))
212                          printf("Erro no envio - send()\n");
213
214  }
215
216
217  //-----------------------------------------------------------------------------------------------------------
218  /*Tratamento da informação recebida do cliente*/
219
220
221  void enviaArduino(char setor, char dispositivo, char estado, int socketCliente, struct sockaddr_in
     clienteAddr)
222  {
223          //Declaração das variaveis
224          char SETOR_TEMPERATURA  = 0xA1;
225          char SETOR_PRESENCA             = 0xA2;
226          char SETOR_PORTAS               = 0xA3;
227          char SETOR_AC                   = 0xB1;
228          char SETOR_LAMPADAS             = 0xB2;
229          char SETOR_ALARME               = 0xB3;
230          char comando[3];
231          char resposta;
232          float temperatura;
233
234          //Divisão dos dispositivos em setores
235          switch(setor)
236          {
237                  //Menu temperatura
238                  case 1:
239                          //Passando parâmetros para comando
240                          comando[0] = SETOR_TEMPERATURA;
241                          comando[1] = dispositivo;
242                          comando[2] = estado;
243
244                          //Enviando o comando para o arduino
245                          Envia_char (comando, 2, uart0_filestream);
246
247                          //Recebe resposta
248                          Recebe_float(uart0_filestream, &temperatura);
249
250                          //Envia resposta ao cliente
251                          Envia_float_cliente(socketCliente, clienteAddr, temperatura);
252
253  #ifdef DEBUG
254                          switch(dispositivo)
255                          {
256                                  case 1:
257                                          printf("Temperatura - Externa\n");
258                                          break;
259
260                                  case 2:
261                                          printf("Temperatura - Interna da Sala\n");
262                                          break;
263
264                                  case 3:
265                                          printf("Temperatura - Interna do quarto\n");
```

```
266                                             break;
267                             }
268     #endif
269                             break;
270
271                     //Menu Presenca
272                     case 2:
273
274                             comando[0] = SETOR_PRESENCA;
275                             comando[1] = dispositivo;
276                             comando[2] = estado;
277                             Envia_char (comando, 2, uart0_filestream);
278                             Recebe_char(&resposta, 1, uart0_filestream);
279                             Envia_float_cliente(socketCliente, clienteAddr, resposta);
280
281     #ifdef DEBUG
282                             switch(dispositivo)
283                             {
284                                     case 1:
285                                             printf("Presença - Entrada Principal\n");
286                                             break;
287
288                                     case 2:
289                                             printf("Presença - Entrada de Serviço\n");
290                                             break;
291
292                                     case 3:
293                                             printf("Presença - Garagem\n");
294                                             break;
295
296                             }
297     #endif
298                             break;
299
300                     //Menu Portas/Janelas
301                     case 3:
302
303                             comando[0] = SETOR_PORTAS;
304                             comando[1] = dispositivo;
305                             comando[2] = estado;
306                             Envia_char (comando, 2, uart0_filestream);
307                             Recebe_char(&resposta, 1, uart0_filestream);
308                             Envia_float_cliente(socketCliente, clienteAddr, resposta);
309
310     #ifdef DEBUG
311                             switch(dispositivo)
312                             {
313
314                                     case 1:
315                                             printf("Estado - Porta principal\n");
316                                             break;
317
318                                     case 2:
319                                             printf("Estado - Porta de serviço\n");
320                                             break;
321
322                                     case 3:
323                                             printf("Estado - Porta garagem\n");
324                                             break;
325
326                                     case 4:
327                                             printf("Estado - Janela da sala\n");
328                                             break;
329
330                                     case 5:
331                                             printf("Estado - Janela do quarto\n");
332                                             break;
```

```
333
334                                     case 6:
335                                             printf("Estado - Janela da cozinha\n");
336                                             break;
337                             }
338     #endif
339                             break;
340
341                     //Menu Ar-condicionado
342                     case 4:
343
344                             comando[0] = SETOR_AC;
345                             comando[1] = dispositivo;
346                             comando[2] = estado;
347                             Envia_char (comando, 3, uart0_filestream);
348
349     #ifdef DEBUG
350                             switch(dispositivo)
351                             {
352                                     case 1:
353                                             printf("AC - Sala - %d\n", estado);
354                                             break;
355
356                                     case 2:
357                                             printf("AC - Quarto - %d\n", estado);
358                                             break;
359                             }
360     #endif
361                             break;
362
363                     //Menu Lampadas
364                     case 5:
365
366                             comando[0] = SETOR_LAMPADAS;
367                             comando[1] = dispositivo;
368                             comando[2] = estado;
369                             Envia_char (comando, 3, uart0_filestream);
370
371     #ifdef DEBUG
372                             switch(dispositivo)
373                             {
374                                     case 1:
375                                             printf("Lampada - Entrada principal - %d\n", estado);
376                                             break;
377
378                                     case 2:
379                                             printf("Lampada - Entrada de serviço - %d\n", estado);
380                                             break;
381
382                                     case 3:
383                                             printf("Lampada - Garagem - %d\n", estado);
384                                             break;
385
386                                     case 4:
387                                             printf("Lampada - Sala - %d\n", estado);
388                                             break;
389
390                                     case 5:
391                                             printf("Lampada - Quarto - %d\n", estado);
392                                             break;
393
394                                     case 6:
395                                             printf("Lampada - Cozinha - %d\n", estado);
396                                             break;
397                             }
398     #endif
399                             break;
```

```
400
401                 //Menu Alarme
402                 case 6:
403
404                         comando[0] = SETOR_ALARME;
405                         comando[1] = dispositivo;
406                         comando[2] = estado;
407                         Envia_char (comando, 3, uart0_filestream);
408
409                         if(dispositivo == 1)
410                                 alarm_setor[0] = estado;
411
412                         if(dispositivo == 2)
413                                 alarm_setor[1] = estado;
414
415 #ifdef DEBUG
416                         switch(dispositivo)
417                         {
418                                 case 1:
419                                         printf("Alarme - Setor 1 (Sensores de Presença) - %d\n",
    estado);
420                                         break;
421
422                                 case 2:
423                                         printf("Alarme - Setor 2 (Sensores de Portas/ Janelas) - %d
    \n", estado);
424                                         break;
425                         }
426                         break;
427 #endif
428         }
429 }
430
431
432 void TrataClienteTCP(int socketCliente, struct sockaddr_in clienteAddr) {
433         int buffer;
434         int tamanhoRecebido;
435         int setor, dispositivo, estado;
436
437         if((tamanhoRecebido = recv(socketCliente, &buffer, sizeof(int), 0)) < 0)
438                 printf("Erro no recv()\n");
439
440
441         setor = buffer/1000;
442         dispositivo = (buffer - (setor*1000))/10;
443         estado = buffer - setor*1000 - dispositivo*10;
444
445         printf("Setor %d, Dispositivo %d, Estado %d\n", setor, dispositivo, estado);
446
447         Log_data(inet_ntoa(clienteAddr.sin_addr), setor, dispositivo, estado);
448         enviaArduino((char)setor, (char)dispositivo, (char)estado, socketCliente, clienteAddr);
449
450 }
451
452 void *server(void *porta)
453 {
454         unsigned short *servidorPorta = (unsigned short *) porta;
455         int servidorSocket;
456         int socketCliente;
457         struct sockaddr_in servidorAddr;
458         struct sockaddr_in clienteAddr;
459         unsigned int clienteLength;
460
461         // Abrir Socket
462         if((servidorSocket = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
463                 printf("falha no socker do Servidor\n");
464
```

```
465          // Montar a estrutura sockaddr_in
466          memset(&servidorAddr, 0, sizeof(servidorAddr)); // Zerando a estrutura de dados
467          servidorAddr.sin_family = AF_INET;
468          servidorAddr.sin_addr.s_addr = htonl(INADDR_ANY);
469          servidorAddr.sin_port = htons(*servidorPorta);
470
471          // Bind
472          if(bind(servidorSocket, (struct sockaddr *) &servidorAddr, sizeof(servidorAddr)) < 0)
473                  printf("Falha no Bind\n");
474
475          // Listen
476          if(listen(servidorSocket, 10) < 0)
477                  printf("Falha no Listen\n");
478
479          while(1) {
480
481                  printf("Esperando por clientes na porta %d\n", *servidorPorta);
482                  clienteLength = sizeof(clienteAddr);
483                  if((socketCliente = accept(servidorSocket, (struct sockaddr *) &clienteAddr,
     &clienteLength)) < 0)
484                          printf("Falha no Accept\n");
485
486                  printf("Conexão do Cliente %s\n", inet_ntoa(clienteAddr.sin_addr));
487
488                  TrataClienteTCP(socketCliente, clienteAddr);
489                  close(socketCliente);
490
491          }
492          close(servidorSocket);
493  }
494
495  int main(int argc, char *argv[]) {
496
497          pthread_t server_8080;
498          pthread_t server_4000;
499          pthread_t alarm_id;
500
501          Uart_Config(uart0_filestream);
502
503          pthread_create(&server_4000, NULL, &server, &porta_4000);
504          pthread_create(&server_8080, NULL, &server, &porta_8080);
505          pthread_create(&alarm_id, NULL, &Verifica_sensor, NULL);
506
507          pthread_join(server_8080, NULL);
508          pthread_join(server_4000, NULL);
509
510  }
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <string.h>
#include <errno.h>


void play_alarm_music( char *estado)
{
        int i, fd, size =0;
        char buffer[256], musica[256];

        FILE *audio_fd;
        FILE *stream;

        while(*estado == 1)
        {
                //Descritor para o arquivo de musica
                audio_fd = fopen("Tornado_Siren.wav", "r");
                if(!audio_fd)
                {
                        printf("Erro na abertura fopem");
                }

                //Abre um pipe entre o programa e o aplay
                stream = popen("aplay -t wav -", "w");
                if(!stream)
                {
                        printf("Erro na abertura do pipe\n");
                }

                //Fileno retorna um descritor da stream no formato int.
                fd = fileno(stream);

                //Executa arquivo ate o FEOF
                while(!feof(audio_fd))
                {
                        fread(buffer, sizeof(buffer), 1, audio_fd);
                        write(fd,buffer,sizeof(buffer));
                }

                //Fecha o pipe e o arquivo de audio
                pclose(stream);
                fclose(audio_fd);
        }
}
```