PROTÓTIPO DE SISTEMA DE MONITORAMENTO REMOTO UTILIZANDO TCP/IP SOBRE ETHERNET (802.3)

Ariberto Montibeller Junior
Orientando

Prof. Miguel Alexandre Wisintainer
Orientador

Roteiro

- Introdução e Objetivos
- Fundamentação Teórica
- Especificação
- Implementação
- Apresentação do Protótipo
- Conclusão

Introdução

- Segurança
- Como se proteger?
- Sistemas atuais
- Proposta

Objetivos

- disponibilizar um hardware para captura de dados dos sensores e da câmera;
- estabelecer comunicação entre o hardware e o microcomputador através da rede ethernet (802.3);
- disponibilizar um software no PC para recebimento e envio dos dados para o dispositivo;
- disponibilizar um software no PC para gerenciar o dispositivo e exibir os dados e imagens recebidas.

Rede TCP/IP

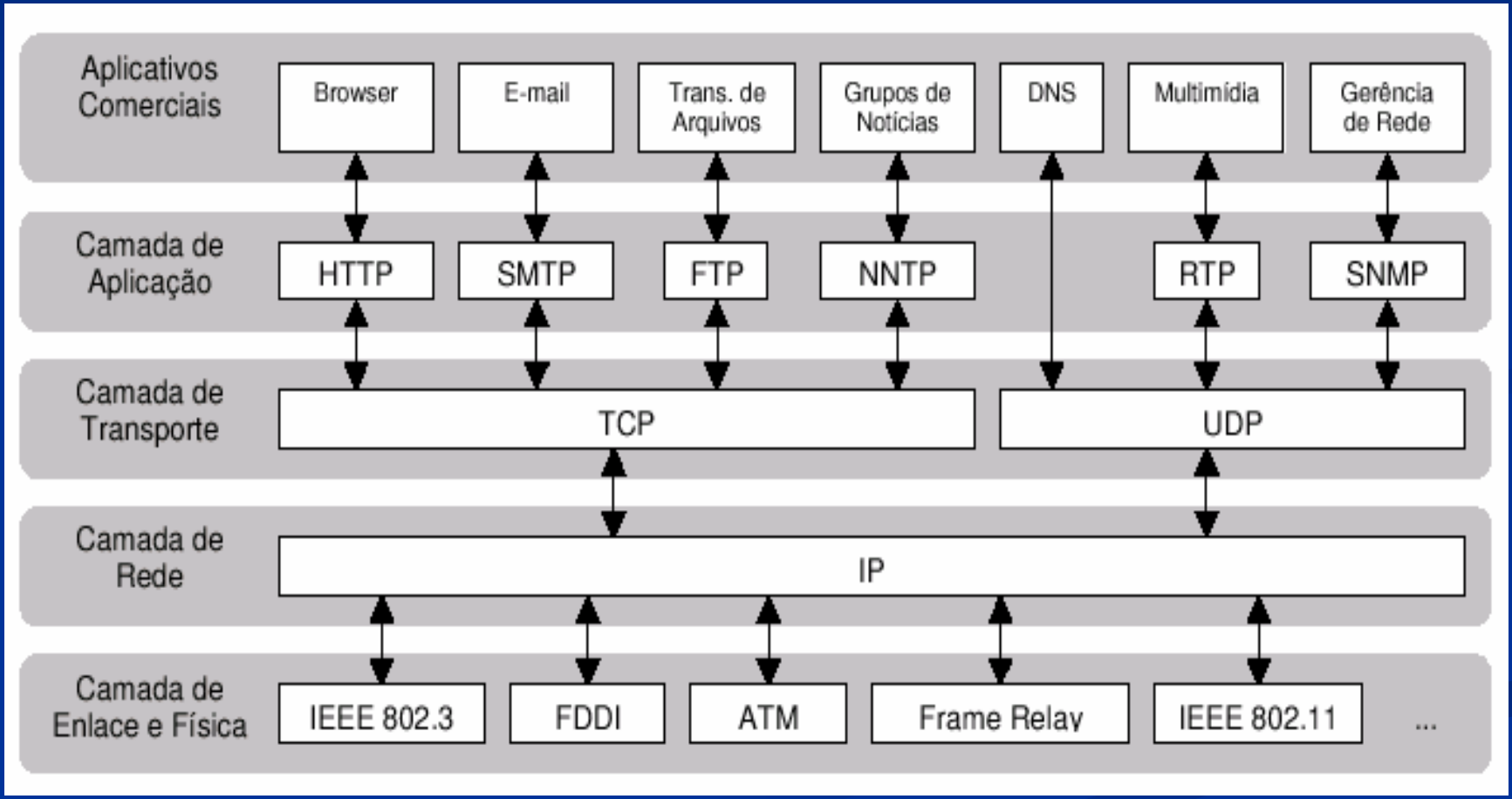

Aplicativos Comerciais
Browser
E-mail
Trans. de Arquivos
Grupos de Notícias
DNS
Multimídia
Gerência de Rede
Camada de Aplicação
HTTP
SMTP
FTP
NNTP
RTP
SNMP
Camada de Transporte
TCP
UDP
Camada de Rede
IP
Camada de Enlace e Física
IEEE 802.3
FDDI
ATM
Frame Relay
IEEE 802.11
...

Imagem digital

Imagem de tons
(ou cores) contínuas

amostragem

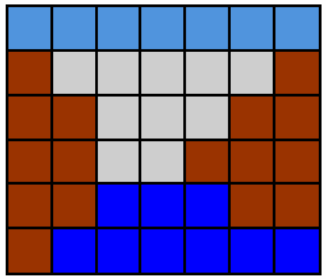


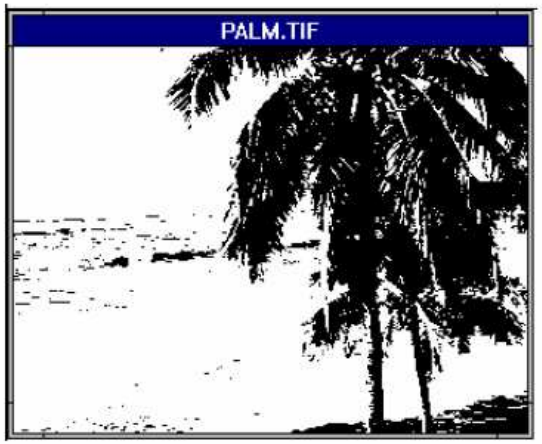
Imagem amostrada

quantização

55	55	55	55	55	55	55
55	20	22	23	45	55	55
55	55	10	09	11	55	55
55	55	43	42	70	55	55
55	55	28	76	22	55	55
55	55	55	55	55	55	55

Imagem amostrada
e quantizada

Imagem digital



(1bit - 2 int.)

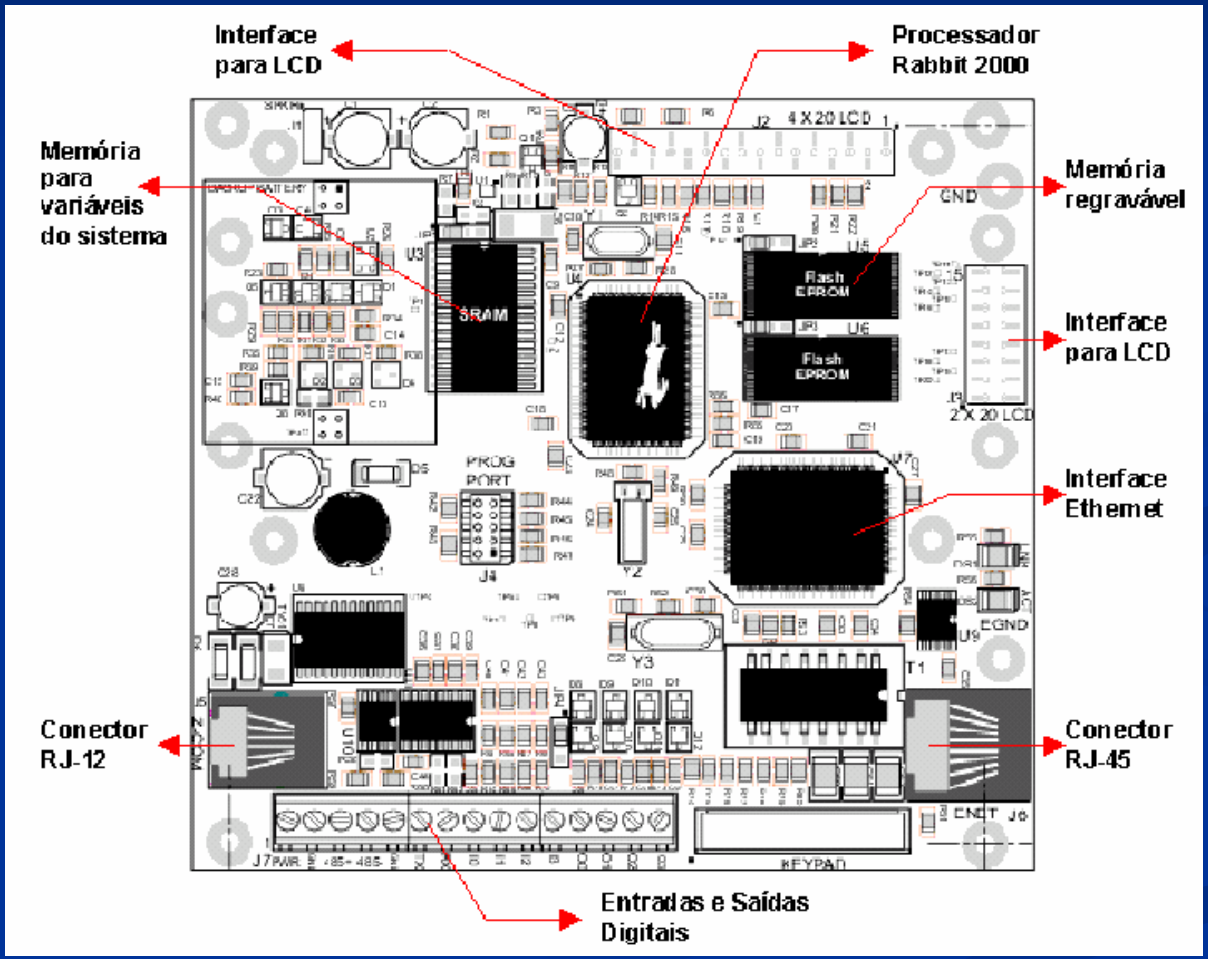
(4 bits - 16 int.)

(8 bits - 256 int.)

Sensores

- sensor magnético ou *reed-switch*;
- sensor capacitivo
- sensor indutivo
- sensor óptico
- sensor de pressão ou chave fim de curso
- *encoder*
- CCD

Kit rabbit 2000 TCP/IP



Kit rabbit 2000 TCP/IP

Bibliotecas

- tcpip.lib
- math.lib
- rs232.lib
- rtclock.lib
- xmem.lib
- sysio.lib

Trabalhos correlatos

- protótipo de um sistema coletor de dados microcontrolado conectado a uma rede TCP/IP (Vasques, 2003);
- protótipo de um hardware para controle de freqüência acadêmica (Silva, 2002);
- protótipo de hardware e software para captura e visualização de imagens compartilhadas via interface digital serial diferencial balanceada (Santos, 2002);
- construção de um protótipo (hardware e software) para segurança predial através de monitoração via câmera digital e *display* gráfico (Merege Neto, 2004).

Requisitos

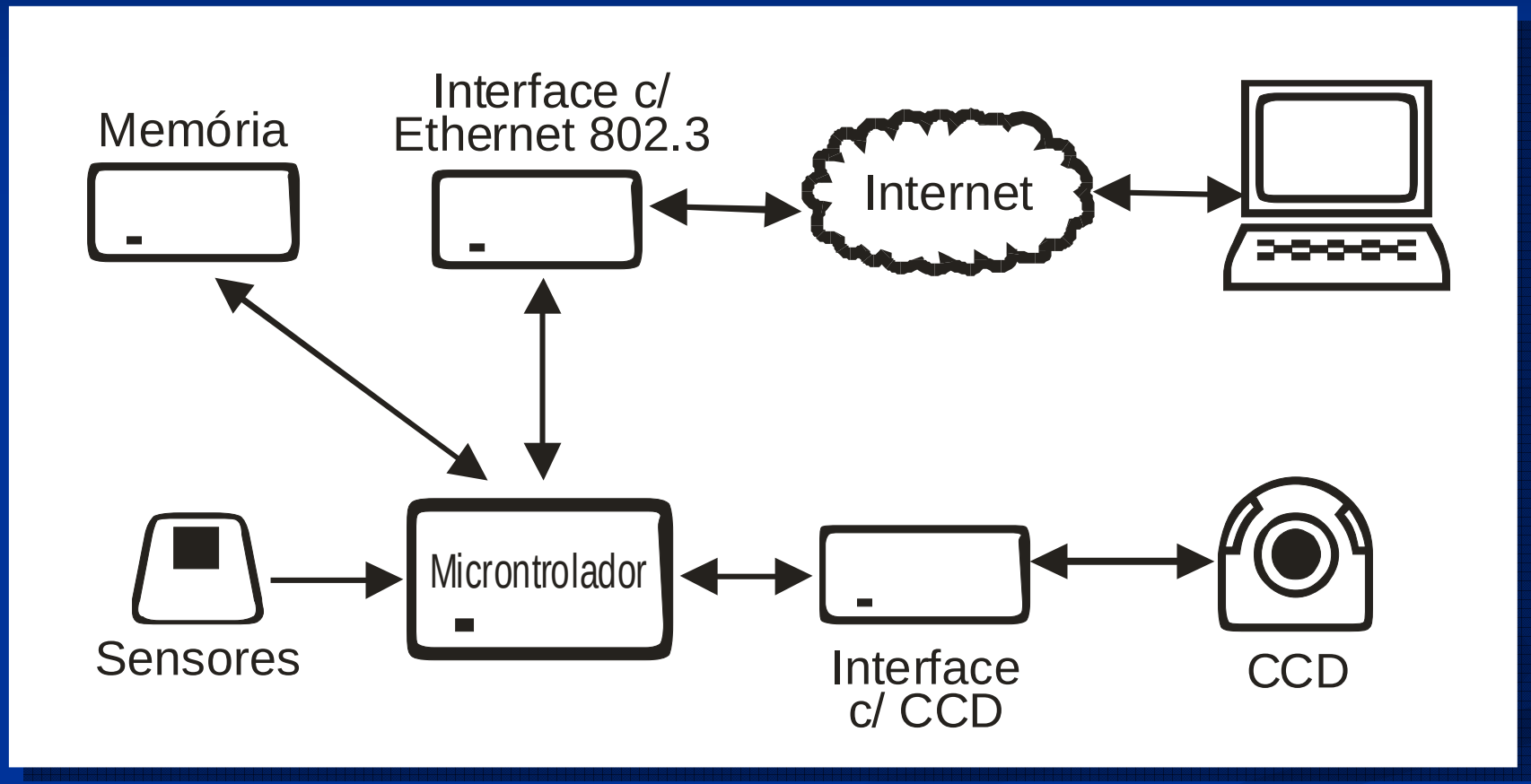
O hardware deve:

- coletar e armazenar os dados captados pela câmera (RF);
- conectar-se à rede ethernet para comunicação com o microcomputador (RF);
- enviar alerta quando ocorrer um imprevisto no ambiente que está sendo monitorado (RF).

O software deve:

- gerenciar o dispositivo (RF);
- receber os dados enviados pelo dispositivo através de uma rede ethernet (RF);
- possibilitar a visualização das imagens captadas pelo dispositivo, assim como os dados dos demais sensores (RF);
- ser independente de plataforma e sistema operacional (RNF).

Especificação do hardware



Especificação do software embarcado

- Cliente TCP/IP
- *Monothread*
- Diagramas de estados e atividades (UML)

Diagrama de estados

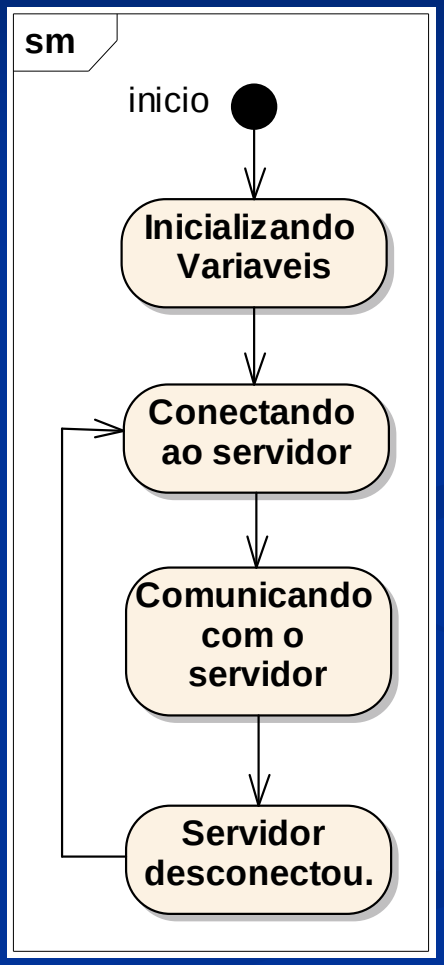
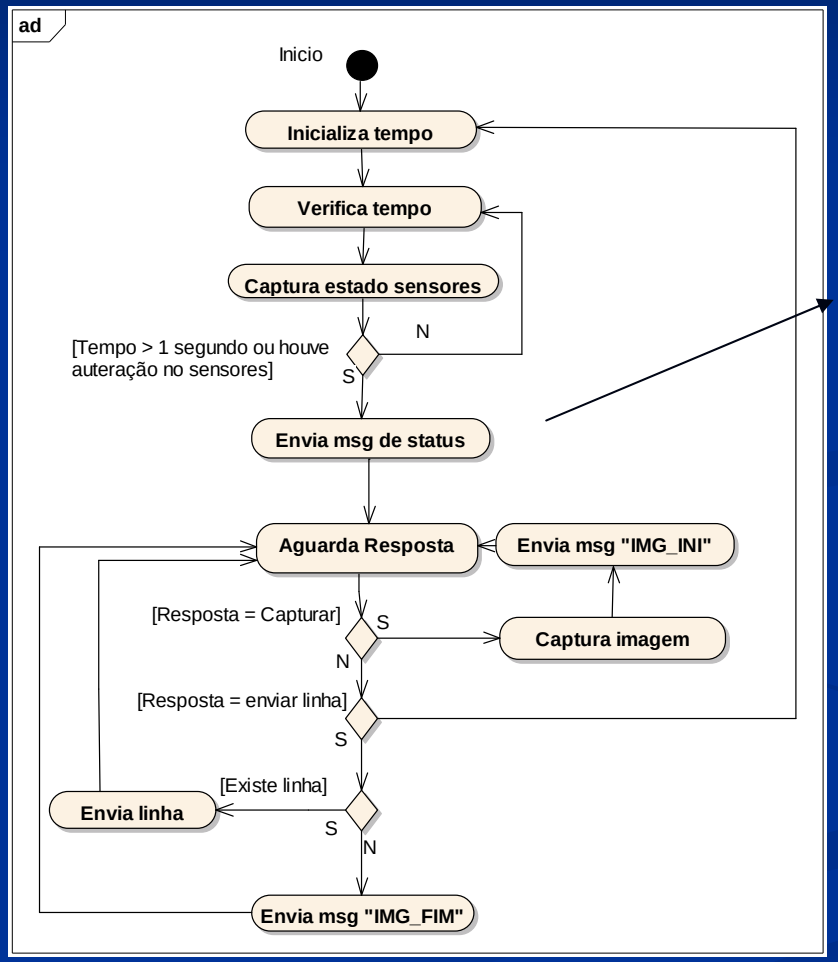
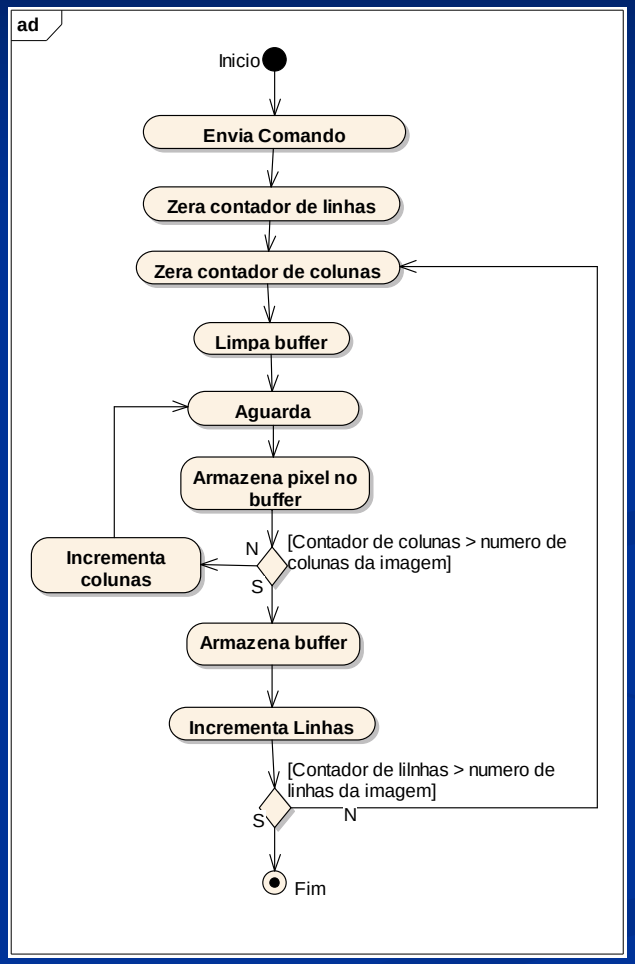


Diagrama de atividades troca de mensagens



id	status
10	4

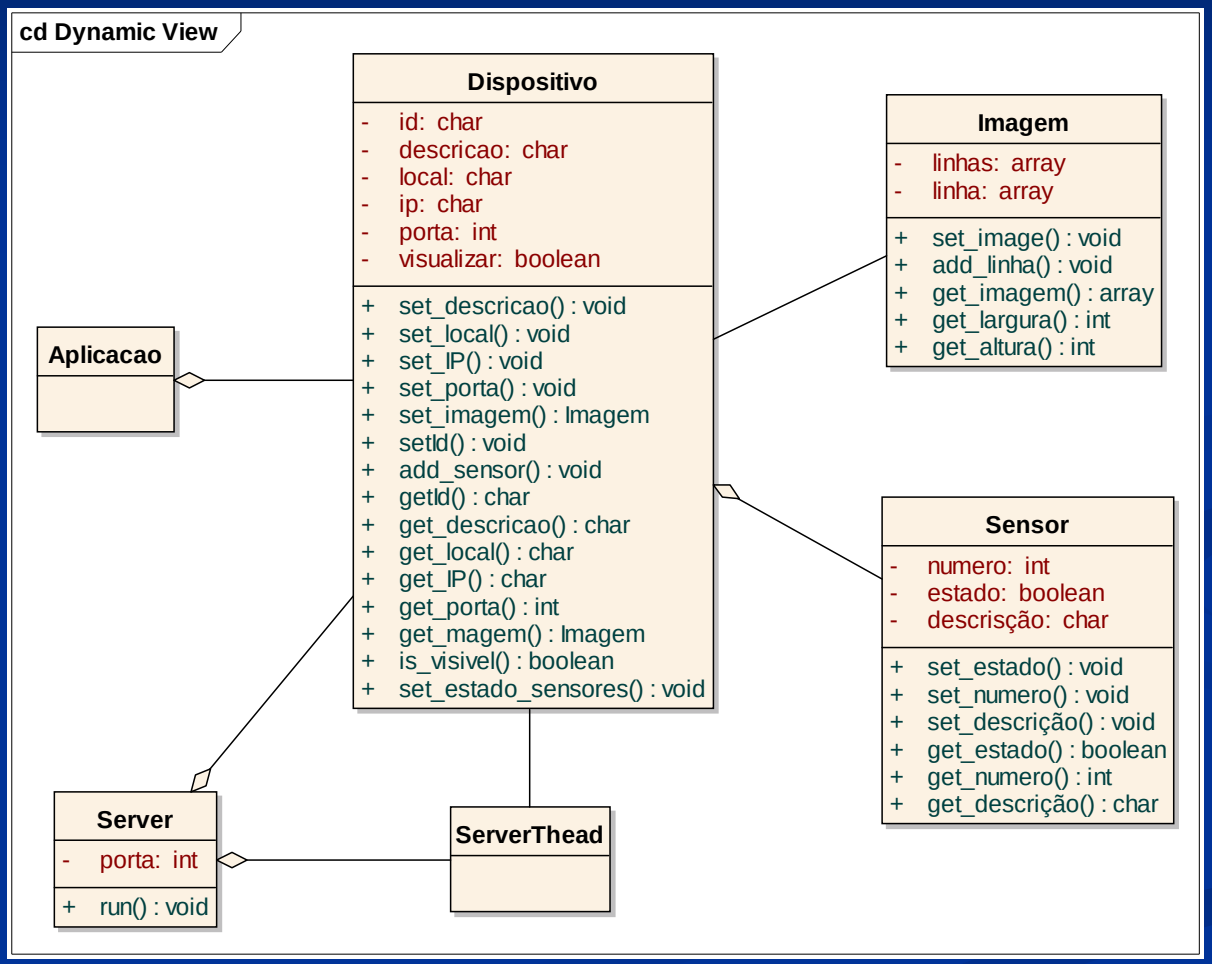
Captura da imagem



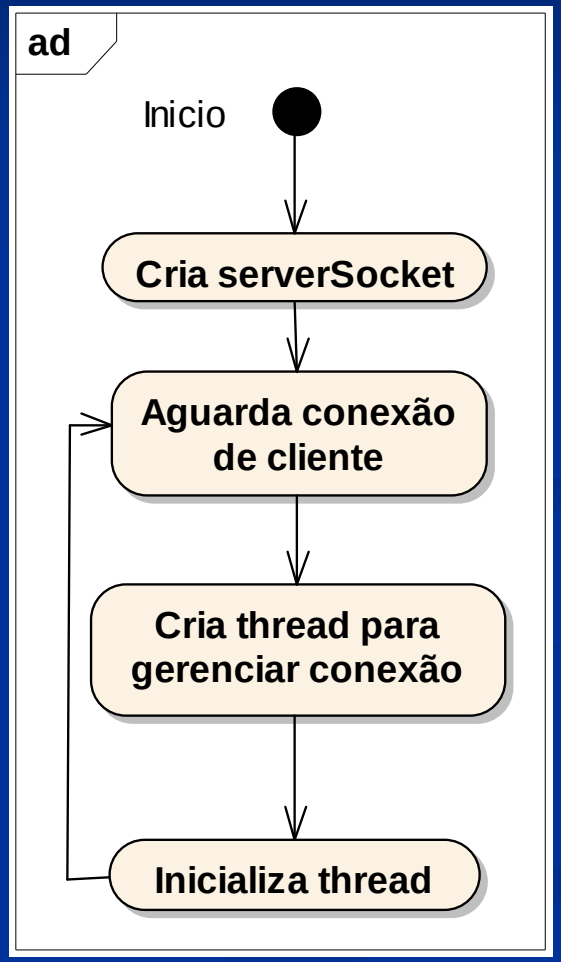
Especificação do software para PC

- Orientado a objetos
- Duas camadas
  - Camada da aplicação
  - Camada servidor
- Classe Dispositivo

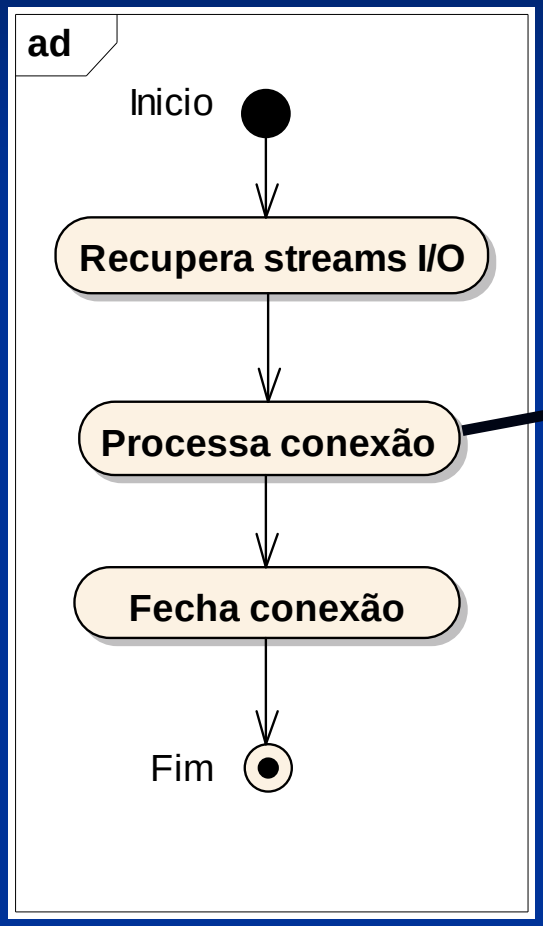
Diagrama de classe

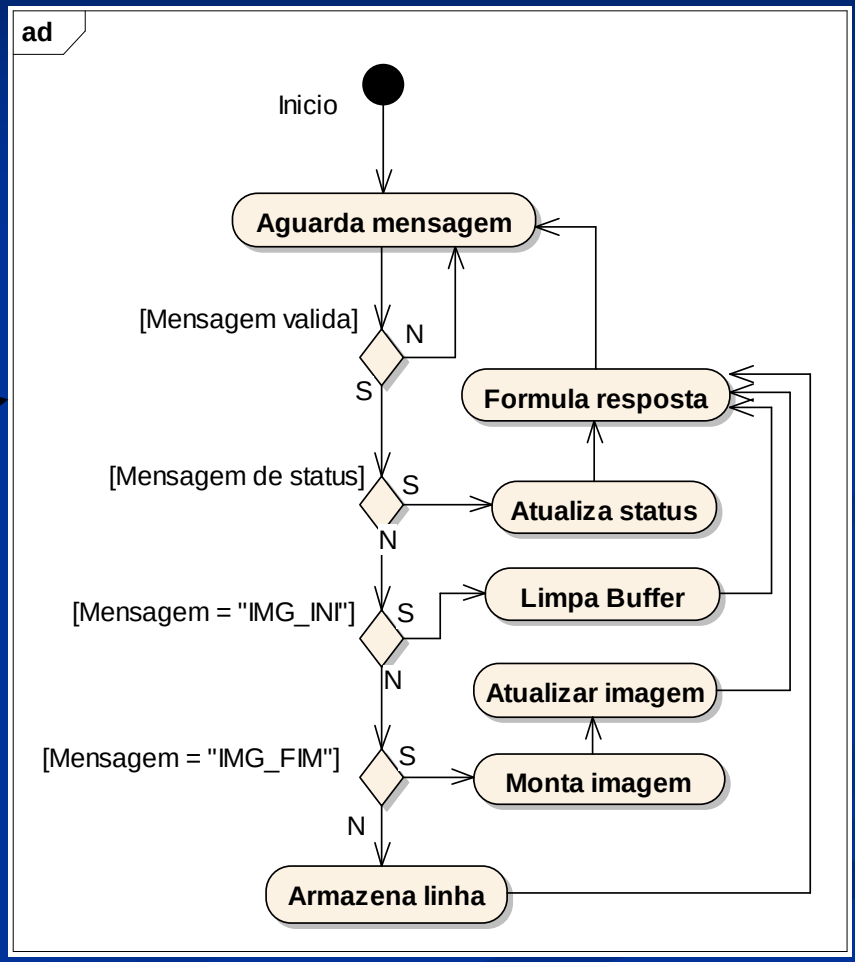


Digrama de atividades do server

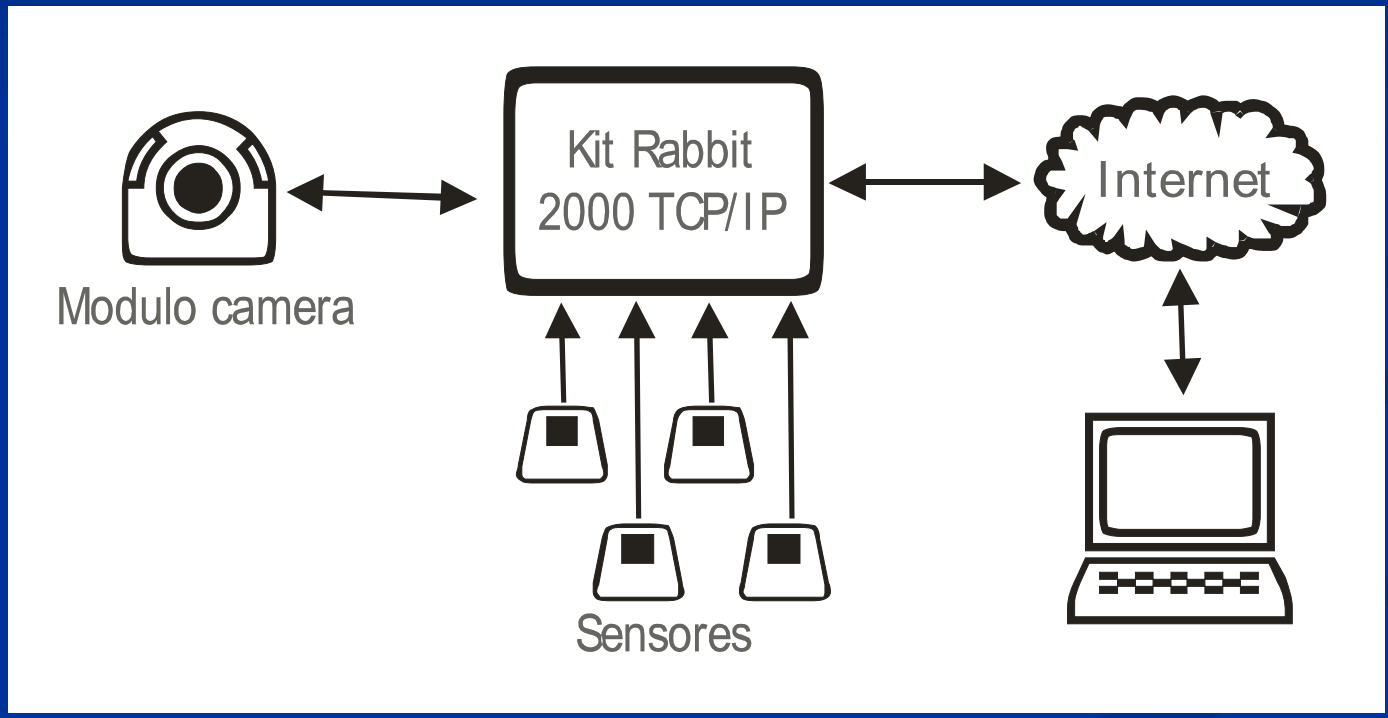


Thread tratamento conexão dos clientes





Hardware

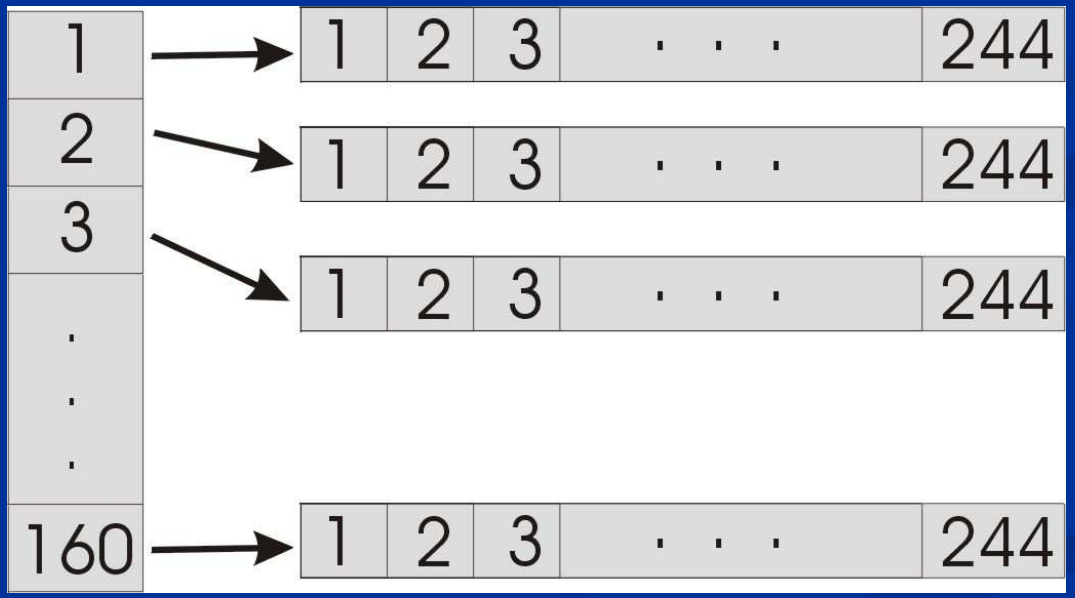


Definição de variáveis

```
#define MY_IP_ADDRESS          "10.0.0.200"      // Endereço IP do dispositivo
#define MY_NETMASK             "255.255.255.0"   // Mascara de rede
#define MY_GATEWAY             "10.0.0.2"        // Endereço IP do Gateway
//#define TCPCONFIG 5          // Obter IP via servidor DHCP
#use "dcrtcp.lib“

#define DEST                   "10.0.0.2"        // Definição do IP do Servidor
#define PORT                   5000              // Porta do servidor

#define CINBUFSIZE 15          // Define tamanho do buffer da porta serial
#define COUTBUFSIZE 15

#ifndef _232BAUD               // Define a velocidade da comunicação serial
#define _232BAUD 57600
#endif

tcp_Socket socket;             // Variável do socket
static char Buf[244];          // Array do buffer de leitura temporário
static unsigned long linhas[160];   // Array que guarda ponteiro da linhas alocadas
```

Alocação de memória

```
for (i = 0; i < 160; i++){
      linhas[i] = xalloc(244);
}
```



Loop principal do software

```
Sock_init();
if( 0L == (destIP = resolve(DEST))  ) { exit(2); }
sock_err:
tcp_open(&socket,0,destIP,PORT,NULL);

Do {
      sock_wait_input(&socket, 20, NULL, &status);                    // define time out de recebimento de mensagem
      bytes_read = sock_fastread(&socket, buffer, sizeof(buffer)-1);  // aguarda recebimento de alguma mensagem

      if(bytes_read>0) {
           buffer[bytes_read] = '\0';                  // Insere character de fim de palavra
           if (buffer[0] == 'K'){
               waitfor( DelayMs(500) );               // Aguarda meio segunda
               str_tmp = monta_str_status()           /* Chama função que monta string contendo ID concatenado com o estado dos sensores*/
               sock_write(&socket, str_tmp, 14);      // Envia retorno da função
            }
            if (buffer[0] == 'C'){
               captura_imagem();                      //Chama função de captura da imagem
               sock_write(&socket, "IMG_INI", 7);  // Envia msg de inicio da imagem
              printf("iniciando transmição\n");
               i = 0;                                 // Inicia contador de linhas enviadas
            }
            if ((buffer[0] == 'N') && (i<160 )){
                xmem2root(Buf, linhas[i], 244);       // Recupera linha alocada dinamicamente
               sock_write(&socket,Buf,244);           // Envia linha
                i++;
            }
            if ((buffer[0] == 'N') && (i == 160)){
               sock_write(&socket,"IMG_FIM",7);  // Envia msg de fim da imagem
               printf("Fim da Transmição\n");
                i++;
            }
     }
} while(tcp_tick(&socket));     // Sai do loop se perder a conexão
```

Função de captura da imagem

```
Void captura_imagem(){
   serCputc('E');       // Envia character ‘E’ via serial para interface da camera
    for (j=0;j<161;j++){                                  // percorre 161 linhas esperadas
         for (i=0;i<361;i++) {                            // percorre 361 pixels de cada linha
               while ((Pxl=serCgetc()) == -1);            // Agurada dado valido
               if  ( i < 244)
                 Buf[i] = Pxl;                            // concatena pixel do buff
            if ((i == 244) && (j < 160))
                 root2xmem(linhas[j], Buf, 244);          // aloca buffer na memoria
       }
    }
}
```

Imagem

```
class Imagem extends JLabel {

    private BufferedImage img;

    public void setImagem(BufferedImage   img){
        this.img = img;
    }

    public void paint(Graphics g) {
         super.paint(g);
         g.drawImage(img, 0, 0, 244, 160, null);
    }
}
```

Imagem

```
BufferedImage img = new BufferedImage( ((ArrayList)imagem.get(1)).size(),
                                                    imagem.size(),
                                                   BufferedImage.TYPE_INT_RGB );
Color c;
int pixelInt;
for (int i = 0; i < imagem.size(); i++ ){
      ArrayList linha = (ArrayList)imagem.get(i);
      for(int j = 0; j < linha.size(); j++){
            try{
                pixelInt = Integer.parseInt( linha.get(j).toString() ) ;
                c = new Color( pixelInt, pixelInt, pixelInt);
            }catch( Exception e ){
                c = new Color( 125, 125, 125);
            }
            img.setRGB(j, i, c.getRGB() );
       }
}
```

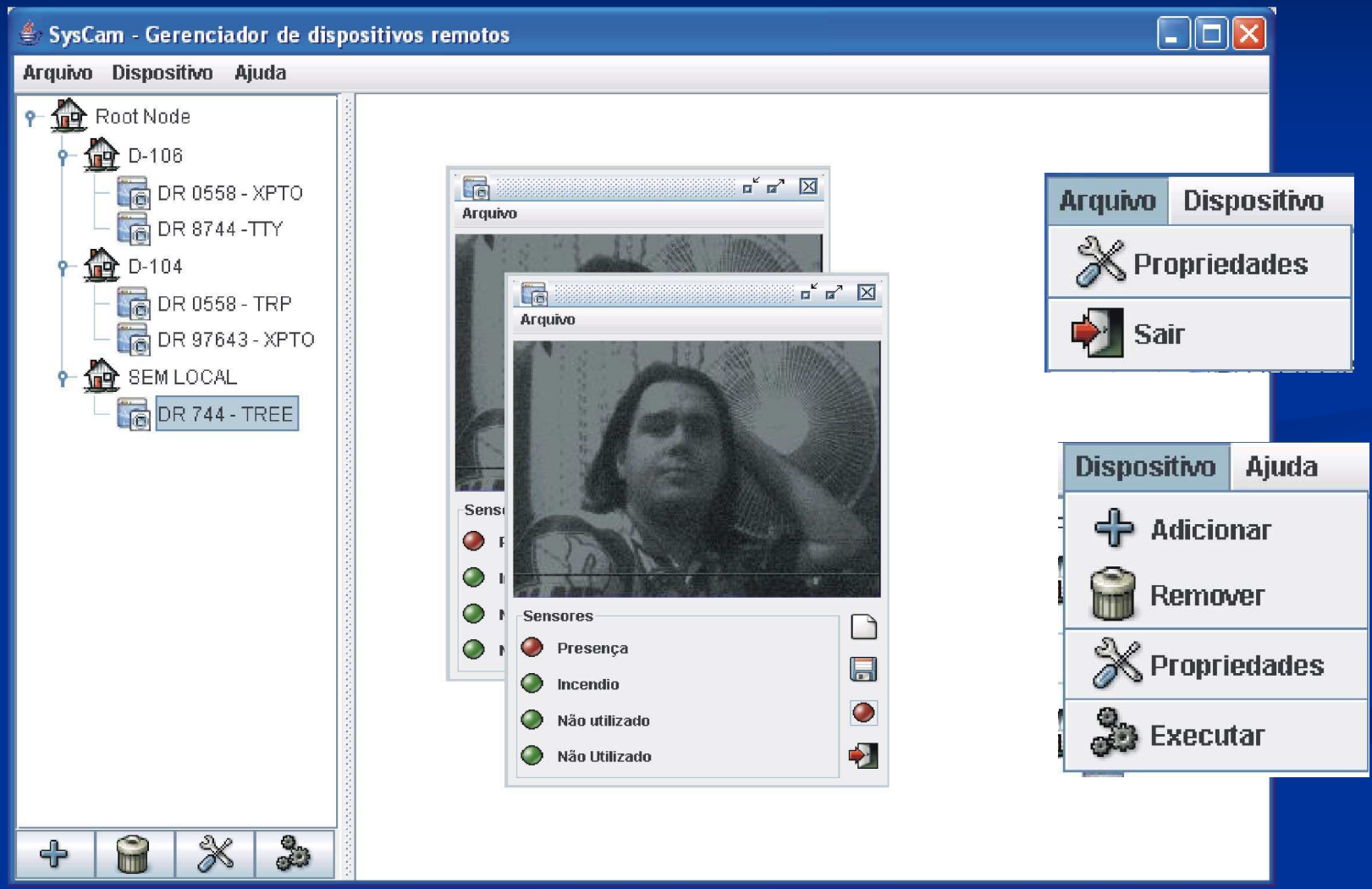
Thread principal do sevidor

```
public class Server implements Runnable {
   private static int porta = 5000;
   private HashMap dispositivos;

   Server(HashMap d){
         this.dispositivos = d;
   }
   Server(int porta, HashMap d){
         this.dispositivos = d;
         this.porta = porta;
   }
   public void run() {
         try{
               ServerSocket listener = new ServerSocket(porta);
               Socket socket;
               while(true){
                     socket = listener.accept();
                     ServerThead conn = new ServerThead(socket, this.dispositivos);
                     (new Thread(conn) ).start();
               }
         } catch (IOException ioe) {
                System.out.println("IOException on socket listen: " + ioe);
         }
    }
}
```

Tratamento conexão com cliente

```
int i = input.read(aux);
msg = new String(aux, 0, i);
if ( msg.length() == 14 ){
   id = getId(msg);
   estadoSensores = getEstado(msg);
   if ( !dispositivos.containsKey( id ) ){
            DispositivoRemoto disp = new DispositivoRemoto(connection.getInetAddress().toString(), connection.getPort() );
            dispositivos.put( id, disp );
   }
   if ( ( (DispositivoRemoto)dispositivos.get(id) ).isVisivel() ){
            output.write( "C".getBytes() );
   }else{
            output.write( "K".getBytes() );
   }
   ( (DispositivoRemoto)dispositivos.get(id) ).setEstadoSensores(estadoSensores);
   output.flush();
}
if ( msg.equals("IMG_INI") ) {
    output.write( "N".getBytes() );
   output.flush();
}
if ( msg.equals("IMG_FIM") ) {
   ((DispositivoRemoto)dispositivos.get(id)).setImagem(imagem);
   output.write( "K".getBytes() );
   output.flush();
}
ArrayList linha = new ArrayList();
for (int ind = 0; ind < msg.length(); ind++ ){
      linha.add( new Integer( aux[ind] ) );
}
imagem.add( linha );
output.write( "N".getBytes() );
output.flush();
```

Tela principal



Propriedades do dispositivo

Dispositivo

Resultados e discussão

- Uso de interface de controle para câmera.
  - Vantagens
  - Desvantagens
- Restrição, apresentou problemas quando utilizado JVM 1.5

Conclusões

- todos os objetivos propostos foram alcançados;
- aplicações que utilizam rede TCP/IP ;
- as ferramentas utilizadas mostraram-se apropriadas para o desenvolvimento do protótipo.

Extensões

- eliminar interface de controle da câmera, aumentando a velocidade de captura das imagens;
- utilizar um CCD colorido e de maior resolução;
- aumentando a velocidade da captura, ao invés de exibir e gravar imagens estáticas, fotos, exibir e gravar vídeos;
- utilizar servo motores para direcionar a câmera;
- mostras mais informações dos sensores, não somente dois estados, por exemplo, ser captada a temperatura do local e demonstrar em escala de graus Celsius;
- criptografar os dados que trafegam pela rede;
- tornar e *thread* do servidor um software independente para que vários programas possam exibir os dados simultâneos, uma sugestão seria utilizar Java RMI.