Centro Universitário de Brasília – UNICEUB
Faculdade de Ciências Exatas e Tecnologia – FAET
Engenharia de Computação

Monitoração On-Line de ambientes pelo telefone celular utilizando streaming de vídeo

Brasília – DF
2007

Centro Universitário de Brasília - UNICEUB
Faculdade de Ciências Exatas e Tecnologia – FAET
Engenharia de computação

Monitoração On-Line de ambientes pelo telefone celular utilizando streaming de vídeo

por

Vinicius André Pinto
20016440 – FAET

Trabalho Final de Graduação

Prof. Msc Francisco Javier de Obaldía.Díaz
Orientador

Brasília/DF – Dezembro de 2007

Agradecimentos

A minha família, especialmente a minha esposa e filha pela paciência prestada durante este percurso.

Aos meus pais pelo esforço dispensado em minha educação me dando a oportunidade de chegar até aqui.

Aos meus irmãos, irmã e amigos pelo incentivo.

Á Deus por estar sempre em meu coração.

RESUMO

Nos dias de hoje, o termo segurança, se tornou uma grande preocupação para as pessoas e empresas em geral. Neste contexto, a monitoração de ambientes em tempo real, utilizando as facilidades de locomoção, oferecida pelas redes sem fio, GSM/GPRS, Wi-Fi e tão logo WCDMA/HSDPA, se torna uma grande aliada aos sistemas de segurança, pois permite que sejam visualizados todos os eventos de um ambiente no momento em que estão ocorrendo, estando-se em qualquer lugar do planeta onde haja disponibilidade destas redes. Desta forma é possível realizar ações de verificação de status e ações preventivas, visando à manutenção da integridade dos ambientes monitorados, ao alcança da mão.

Neste trabalho será apresentado o desenvolvimento e a implementação de um sistema de monitoração e vigilância de ambientes. O mesmo irá disponibilizar na tela de um telefone celular, “tempo real”, as imagens geradas por uma webcam conectada a um servidor de imagens (streamings de vídeo). O servidor analisará as imagens recebidas da câmera e irá determinar a existência ou não de movimentos. Havendo movimento o servidor enviará uma mensagem de texto para números de telefones celulares pré-determinados, avisando que o usuário deve se conectar ao sistema. Será desenvolvida uma aplicação que rodará no aparelho celular e que será responsável por conectar o tocador de streams do aparelho (player de streams), a um servidor de imagens, e apresentará na tela do aparelho as imagens capturadas na webcam.

Será possível, também, se conectar ao servidor de imagens através de qualquer PC conectado a internet, e monitorar o ambiente.

O resultado deste projeto será o protótipo de uma ferramenta que pode auxiliar a sistemas e equipes de vigilância e segurança, trazendo mobilidade, e praticidade aos processos de monitoração e vigilância. Embora, não contemple o áudio, a obtenção das imagens em tempo real mostra a viabilidade de se realizar a monitoração com o sistema aqui proposto.

Palavras-Chave: imagens, Webcam. Servidor, Vigilância, GSM, GPRS, Telefone Celular.

ABSTRACT

On today, the term safety, has became a major concern in general for the people, and companies. In this context, the surveillance of environments in real time, using the facilities of locomotion, offered by wireless networks, GSM/GPRS, Wi-Fi and once WCDMA / HSDPA it turns a big one allied to safety's systems it turns a big one allied to safety's systems, because it allows all to be visualized of the events of an environments, when they are occurring. Anywhere on the planet where there is availability of these networks. Allowing this way to accomplish actions of status verification and preventive actions, seeking the maintenance of the integrity of the monitored environments, the reach of the hand.

In this work it will be presented the development and implementation of a monitoring and surveillance of environments system. The same will make available in the screen of a cellular telephone, in "real time", the images generated by a webcam connected on an images a server (video streaming). The server will analyze the received images from the camera and it will determine the existence or not, of movements. Having movement the server will send a text message for numbers of pre determined cellular telephones, informing that the user may have to connect to the system. Will be developed an application that will run in the cellular and that it will be responsible for connecting it on an images server, and it will present on the screen of the cellular, the images captured in the webcam.

Will be possible too, connects to the images server through any PC connected to the internet, to monitor the environment.

The result of this project will be the prototype of a tool that will help to Surveillance and safety systems teams, bringing mobility, and practical, to Monitoring and surveillance processes. Although, not including audio, the taking of images in real time show the feasibility of performing the monitoring with the system proposed here.

Keywords: Webcam, Server, surveillance GSM, GPRS, Cellular Telephony.

SUMARIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIMBOLOS

3GPP	**3rd Generation Partnership Project**
AAC-LC	**Advanced Audio Coding - Low Complexity**
Ad-Hoc	**Rede sem topologia determinada - Autoconfigurável**
AMR-NB	**Adaptive Multi-Rate Narrow Band**
AP	**Access Point - Ponto de Acesso**
API	**Application Programming Interface**
AUC	**Authentication Centre**
BSC	**Base Station Controller**
BSS	**Base Station System**
BTS	**Base Transmission System**
CODEC	**Codificador e Decodificador**
Core CS	**Nucleo da rede de Voz - Circuit Switch**
Core OS	**Núcleo da rede de dados - Packet Switch**
DSS	**Darwing Streaming Server - Servidor de Streaming**
EDGE	**Enhanced General Packet Radio Service**
FMS	**Flash Media Server - Servidor de Streaming padrão FLV**
FTP	**File Tranfer Protocol**
GPRS	**General Packet Radio Service**
GSM	**Global System Mobile**
HLR	**Home Location register**
hotspot	**Célula de acesso para redes Wi-Fi**
HSDPA	**High-Speed Downlink Packet Access**
HSPA	**High-Speed Packet Access**
HTTP	**HyperText Transfer Protocol**
IN	**Inteligent Network**
IP	**Internet protocol**
IPv4	**Internet protocol versão 4**
IPv6	**Internet protocol versão 6**
LAN	**Local access network**
LBS	**Location-Based Services**
MGW	**Media Gateway for Mobile Networks**
MSC	**Mobile Switching Center**
MSS	**Mobile Soft Switch**
NSS	**Network & Switching Subsystem**
OSS	**Operation and support system**
QoS	**Quality of Service**
RFC	**Request for Comments**

RNC	**Radio Network Controller**
RSVP	**Resource Reservation Protocol**
RTMP	**Real Time Messaging Protocol**
RTP	**Real Time Transport Protocol**
RTSP	**Real Time Streaming Protocol**
SDP	**Session Description Protocol**
SMTP	**Simple Mail Transfer Protocol**
STA	**Wireless Station**
TCP	**Transmission Control Protocol**
TDMA	**Time Division Multiple Access**
UDP	**User Datagram Protocol**
UMTS	**Universal Mobile Telecommunication System**
USB	**Universal Serial Bus**
UTRAN	**UMTS Terrestrial Radio Access Network**
VAS	**Value Agregate Services**
VLR	**Visitor Location Register**
WAP	**Wireless Aplication Protocol**
WCDMA	**Wideband Code Division Multiple Access**
Wi-Fi	**Wireless fidelity**
XML	**eXtensible Markup Language**
buffer	**memória temporária utilizada para escrita e leitura de dados**
delay sensitive	**Sensibilidade ao atraso**
Handsets	**Aparelhos telefônicos portáteis**
interarrival jitter	**estimativa da variância estatística de chegadas subseqüentes de pacotes RTP de dados**
timestamping	**Campo do pacote RTP com iformções sobre tempoe sequência dos pacotes**
WebCam	**Câmera usada em serviços Internet ound.**

Capítulo 1 – Introdução

Nos dias de hoje, o termo segurança, se tornou uma grande preocupação para as pessoas e empresas em geral. Neste contexto, a monitoração de ambientes em tempo real, se torna uma grande aliada dos sistemas de segurança, pois permite que sejam visualizados todos os eventos de um ambiente no momento em que estão ocorrendo. Desta forma é possível realizar ações de verificação de status e ações preventivas, visando à manutenção da integridade dos ambientes monitorados.

O foco deste trabalho se concentra em demonstrar a utilização prática de streaming de vídeo em sistemas de segurança, visualizados através de dispositivos moveis com suporte a redes GSM/GPRS, WCDMA/HSDPA e Wi-Fi.

A seguir serão apresentados os motivos, objetivos e a estrutura que esta monografia segue, assim como o detalhamento das bases teóricas e, os conceitos utilizados como diretrizes no desenvolvimento deste trabalho.

1.1 MOTIVAÇÃO

Desenvolver mecanismos que possam proporcionar uma vida mais segura e prazerosa, sempre foi uma grande motivação para a comunidade científica e para a indústria, por ser uma forma de oferecer benefícios às comunidades e a melhoria do bem comum.

Atualmente, o crescimento da violência nas grandes cidades tem aumentado, ainda mais, a necessidade de investimentos na área de segurança.

Um dos gastos que estão entre os que mais crescem no orçamento familiar é o destinado a evitar roubos e assaltos cujo custo já começa a rivalizar com as mensalidades das escolas particulares e dos planos de saúde privados. Segundo levantamento do Instituto Brasileiro de Planejamento Tributário (IBPT), famílias com renda mensal entre R$ 3 mil e R$ 10 mil gastaram, em

2006, 4,31% da renda bruta anual familiar, o terceiro maior custo, atrás apenas da educação (13,22%) e da saúde (10,87%) [Securitech, 2007].

Este cenário leva a acreditar que o desenvolvimento de sistemas de segurança pessoal e empresarial é um negócio em grande expansão, criando espaço para diversas existentes.

Em função desta necessidade surgiu a idéia de desenvolver um sistema de segurança que possa oferecer um diferencial em relação aos sistemas já existentes, tanto para sistemas pessoais como empresariais.

A procura por diferenciais levou a idéia de trazer “mobilidade” para os sistemas de segurança, pois hoje a maioria dos dispositivos de segurança não tem soluções com mobilidade, e o grande desenvolvimento das redes móveis (GSM/GPRS, WCDMA/HSDPA e Wi-Fi), tem impulsionado o surgimento de aplicações que utilizam os recursos destas redes. Um dos recursos que estão na “mira” das empresas responsáveis pela oferta de serviços GSM/GPRS/EDGE, WCDMA/HSDPA e Wi-FI, é o de utilização de Multimídia (Imagem e Som) nos dispositivos móveis (Handsets), pois a capacidade das redes de utilizar estes recursos ainda não está totalmente explorada por usuários e provedores de serviços [Teleco, 2006].

Unindo estas duas realidades surgiu a possibilidade de desenvolver um sistema de segurança aliado à mobilidade, ou seja, uma forma do usuário verificar a integridade de seu patrimônio, sem precisar se deslocar, necessitando apenas de um aparelho celular com suporte a Multimídia de redes GSM/GPRS/EDGE ou WCDMA/HSDPA ou Wi-Fi ou qualquer combinação destas tecnologias.

O resultado destas idéias ganhou nome e se tornou o sistema de “Monitoração On Line de ambientes pelo telefone celular utilizando streaming de vídeo”. Este sistema propõe a possibilidade de monitorar ambientes, através de um telefone celular, onde será possível receber as imagens geradas por uma câmera ou webcam conectada a um computador, em um ambiente distante, na tela do telefone em modo “On Line”.

A idéia sugerida como projeto, tem aplicabilidade em diversas áreas, de segurança e vigilância, e além do sistema On Line de monitoração de ambientes, podemos citar como exemplo as seguintes aplicações:

- Sistemas de segurança privada para condomínios e empresas usando Handhelds com suporte Wi-Fi, e com sistema de envio de mensagens dedicado.
- Sistemas de monitoração de crianças à distância.
- Sistemas de monitoração de animais de estimação.
- Sistemas de monitoração de pessoas enfermas ou com necessidades especiais.
- Sistemas de monitoração com reconhecimento de face.
- Sistemas de monitoração com acompanhamento do movimento dos objetos ou pessoas.
- Sistema de monitoração de sistemas elétricos de alta capacidade, utilizando câmeras térmicas ou infravermelha.

Esta diversidade de aplicações possíveis de serem exploradas utilizando os recursos do projeto sugerido aumenta a possibilidade de aproveitamento do sistema, ou parte dele, em aplicações de cunho comercial.

1.2 OBJETIVOS

O principal objetivo é especificar e desenvolver o protótipo de um sistema que permita monitorar qualquer ambiente, através de um telefone celular, onde será possível receber as imagens geradas por uma câmera ou webcam conectada a um computador, na tela do telefone em modo “On Line”.

O sistema proposto oferece uma ferramenta de baixo custo que pode ser usada de forma a monitorar ambientes residenciais, em qualquer lugar onde exista estrutura de redes GSM/GPRS/EDGE, WCDMA/HSDPA ou Wi-Fi, ou como ferramenta de suporte a equipes de segurança empresarial, podendo no futuro ser conectado a sistema de segurança que não possuam a capacidade de oferecer mobilidade.

Para este trabalho será desenvolvido um protótipo composto de um computador rodando um servidor de media, onde será instalado um software, que será desenvolvido para o sistema. Este software terá a capacidade de

monitorar uma webcam, conectada neste computador, e de analisar as imagens a procura de movimentos. Quando detectado o movimento, o software enviará uma mensagem para o telefone(s) previamente cadastrado(s) no sistema informando ao usuário a existência de movimento no ambiente monitorado.

Ao mesmo tempo o software disponibilizará no servidor de media, as imagens capturadas pela webcam, em formato possível de ser visualizado pelo telefone celular através de uma rede GSM/WCDMA ou Wi-Fi, ou através de qualquer computador conectado a internet.

Para o desenvolvimento deste sistema serão utilizadas várias ferramentas de software e recursos de hardware e da telefonia móvel, com o objetivo de demonstrar a possibilidade de utilização dos recursos multimídia em sistemas de segurança.

1.3 ESTRUTURA DO TRABALHO

Neste trabalho será apresentado, no Capítulo 2, um breve referencial teórico e técnico de todas as tecnologias utilizadas na busca de uma solução para atingir os objetivos propostos acima, procurando sempre explicitar o motivo da adoção de uma ou outra tecnologia, no desenvolvimento do sistema.

No Capítulo 3, será mostrado todo o desenvolvimento da idéia do projeto, a proposta em seus detalhes, todas as topologias e premissas escolhidas para serem adotadas no desenvolvimento das partes e na integração do sistema. Também, será mostrada a execução da construção de cada módulo do projeto, dentro das condições de elaboração adotadas anteriormente, assim como todas as limitações enfrentadas, elaboração das interfaces, descrição de cada etapa, seguida da implementação do modelo de protótipo adotado.

No Capítulo 4, serão apresentados os resultados da implementação do protótipo sugerido para o projeto, os testes realizados, os resultados alcançados, as comparações feitas e as melhorias implementadas em função da análise dos resultados obtidos com as simulações.

No capítulo 5, finalmente, as conclusões e sugestões para a evolução do sistema e futuras implementações.

Capítulo 2 – Tecnologias de Redes e de Desenvolvimento utilizadas no Projeto

2.1 INTRODUÇÃO

Neste capítulo serão apresentadas as tecnologias e sistemas que fazem parte da solução do projeto, bem como uma descrição das justificativas pela escolha de cada uma delas.

As principais ferramentas utilizadas no desenvolvimento do projeto aqui apresentado têm como referencial teórico algumas tecnologias que apresentamos nas próximas seções.

2.2 MULTIMEDIA NETWORKING

As redes de computadores foram desenvolvidas para conectar computadores em diferentes locais com o intuito de estabelecer comunicação e compartilhar dados. Inicialmente, a maioria dos dados que trafegavam nas redes era textual; hoje, com o avanço da multimídia e das tecnologias de rede, transmitir informação multimídia vem tornando-se um aspecto indispensável na Internet [LIU,2000].

Nos últimos anos testemunhamos um explosivo avanço no desenvolvimento de aplicações de rede que transmitem e recebem conteúdo multimídia pela Internet. Novas aplicações, também conhecidas como aplicações de mídia contínua, como entretenimento, telefonia IP, rádio pela Internet, sites multimídia, teleconferência, jogos interativos, mundos virtuais, ensino à distância e outros, são anunciadas diariamente [ROSS,2001].

Para os desenvolvedores, *multimedia networking* significa desenvolver uma infra-estrutura de hardware e software para suportar o transporte de informação multimídia na rede, podendo esta encontrar-se no contexto de uma rede local ou na Internet [LIU,2000].

2.2.1 APLICAÇÕES MULTIMIDIA DE REDE

Os esforços exigidos para aplicações desse tipo diferem significantemente das tradicionais aplicações orientadas a dado como Web (texto/imagem), e-mail, FTP, etc. Em particular, aplicações multimídia possuem as seguintes características que as diferem das demais aplicações de rede:

a) **considerações com o tempo**: aplicações multimídia são altamente sensíveis a atrasos (*delay sensitive*) na transmissão e às variações que podem ocorrer nesses atrasos (*interarrival jitter*);

b) **tolerância à perda**: ocasionais perdas podem apenas causar ocasionais falhas na exibição do vídeo, e tais perdas podem ser parciais ou totalmente camufladas;

Estas diferenças sugerem que uma arquitetura de rede projetada inicialmente para comunicação confiável de dados possa não ser adequada para suportar aplicações multimídia. Assim, inúmeros esforços têm sido feitos para permitir que a arquitetura Internet possa oferecer suporte aos serviços exigidos por esse novo tipo de aplicação [ROSS,2001].

Prover serviços de vídeo consiste em um sistema especializado de distribuição multimídia cujo propósito é a coleção, armazenamento, distribuição e apresentação de imagens em movimento. Existem hoje duas técnicas utilizadas para fornecer serviços de vídeo [TEC,2001]:

b.1) **download and play**: esta técnica requer que o arquivo de vídeo seja completamente transferido para o cliente antes de ser usado ou visualizado. Vídeos podem ser transferidos como arquivos binários ou através de mensagens de e-mail. O tempo necessário para a transferência de grandes arquivos pode ser o principal aspecto a ser considerado quando o usuário está esperando para iniciar a visualização, porém, o armazenamento local do mesmo permite ao usuário visualizar o vídeo sempre que desejar. Os principais formados de arquivo de vídeo usados são AVI ou ActiveMovie para Windows e QuickTime para Mac ou Windows;

b.2) **streaming**: nesta técnica o sinal de vídeo é transmitido ao cliente e sua apresentação inicia-se após uma momentânea espera para armazenamento dos dados em um *buffer*. Nesta forma de transmitir vídeo não é preciso fazer o download prévio do arquivo, o micro vai recebendo as

informações continuamente enquanto mostra ao usuário. Esta técnica reduz o tempo de início da exibição e também elimina a necessidade de armazenamento local do arquivo. Transmissões eficazes desses sinais de vídeo através de redes com baixa largura de banda requerem uma alta taxa de compressão de dados para garantir a qualidade visual da apresentação.

A Internet possui hoje uma grande variedade de aplicações multimídia das quais três estão voltadas à transmissão de vídeo em tempo real na rede.

Essas três variedades são descritas a seguir.

2.2.2 STREAMING DE VÍDEO ARMAZENADO

Neste tipo de aplicação, clientes requisitam arquivos de vídeo que estão armazenados em servidores. Nesse caso, o conteúdo multimídia é, pré-gravado e armazenado em um servidor. Como resultado, o usuário pode controlar o vídeo mostrado à distância com funções similares às disponíveis em um videocassete. Para esse tipo de aplicação, a transmissão de conteúdo multimídia só acontecerá sob a demanda do cliente, podendo existir vários clientes conectados ao servidor simultaneamente; cada um visualizando um conteúdo diferente.

2.2.3 STREAMING DE VÍDEO AO VIVO

Este tipo de aplicação é similar à tradicional transmissão de rádio e televisão, conhecida como *broadcast*, onde o cliente assume uma posição passiva e não controla quando o *stream* começa ou termina. A única diferença está no fato dessa transmissão ser feita através da Internet. Tais aplicações permitem ao usuário receber um sinal de rádio ou televisão ao vivo que foi emitido de qualquer parte do mundo. Neste caso, como o *streaming* de vídeo não é armazenado em um servidor, o cliente não pode controlar a exibição da mídia. Neste tipo de transmissão podem existir muitos clientes recebendo o mesmo conteúdo simultaneamente com a sua distribuição ocorrendo de duas formas:

a) **unicast**: é uma conexão ponto-a-ponto entre o cliente e o servidor, onde cada cliente recebe seu próprio *stream* do servidor. Dessa forma, cada

usuário conectado ao *stream* tem sua própria conexão e os dados vêm diretamente do servidor;

b) **multicast**: ocorre quando o conteúdo é transmitido sobre uma rede com suporte à *multicast*, onde todos os clientes na rede compartilham o mesmo *stream*. Assim, preserva-se largura de banda, podendo ser extremamente útil para redes locais com baixa largura de banda.

2.2.4 VIDEO INTERATIVO EM TEMPO REAL

Esse tipo de aplicação permite às pessoas utilizar áudio e vídeo para comunicar-se em tempo real. Como exemplos de aplicações interativas em tempo real, temos softwares de telefonia e vídeo conferência na Internet, onde dois ou mais usuários podem se comunicar oral e visualmente.

Aplicações desse tipo envolvem muitos indivíduos ou grupos de indivíduos em uma espécie de diálogo. O objetivo é não manter uma simples conversa bilateral, mas suportar uma reunião entre dois ou mais participantes remotamente [FLU,1995].

2.2.5 LARGURA DE BANDA E ARMAZENAMENTO PARA STREAMINGS DE VIDEO

O tamanho médio para armazenamento de uma streaming de video, é calculado em função do tempo de execução do video e da largura de banda pela seguinte fórmula (para um único usuário e arquivo):

Tamanho do Arquivo (Mbyte) = Tempo de Exibição (Segundos) X Taxa de Compressão (kbit/s) / 8.388,608

(desde que: 1Mbyte = 8 x 1.048.576 bits = 8.388,608 Kilobits)

Exemplo Real:

Uma hora de video codificado a 300 Kbits/s, (esta é uma taxa típica para videos codificados com janelas de 320x240 pixels) terá:

(3.600 s x 300 kbit/s) / 8,388.608 = 128.7 Mbyte de tamanho para armazenagem.

Se este arquivo de vídeo é armazenado em um servidor de streaming fornecendo tráfego “on-demand”, e se este stream está sendo visualizado por 1.000 usuários, serão necessários:

300Kbits x 1.000 = 300.000 Kbits/s = 300 Mbits/s de largura de banda para distribuir o filme entre os usuários, o que é equivalente a 135 Gbits por hora.

2.3 STREAMING DE VÍDEO NA INTERNET

A Internet não é naturalmente adequada à transmissão de informação em tempo real. Para executar multimídia sobre a Internet, muitas questões precisam ser respondidas, [LIU,2000]. Assim:

a) multimídia indica intenso tráfego de dados. O hardware atual da internet não oferece largura de banda suficiente;

b) aplicações multimídia estão geralmente relacionadas com *multicast*, ou seja, o mesmo fluxo de dados, não múltiplas cópias, é enviado a um grupo de receptores. Por exemplo, uma transmissão de vídeo ao vivo pode ser enviada a milhares de clientes. Os protocolos desenvolvidos para aplicações multimídia precisam considerar o *multicast* para reduzir o tráfego;

c) o preço para agregar recursos de rede aos atualmente existentes torna-se impraticável. Aplicações em tempo real requerem largura de banda, então, precisa haver alguns mecanismos para que essas aplicações reservem os recursos necessários ao longo da rota de transmissão;

d) a Internet é uma rede de transmissão de pacotes que são encaminhados independentemente através de redes compartilhadas. As tecnologias atuais não podem garantir que dados de tempo real não irão encontrar seu destino sem serem desordenados. Alguns novos protocolos de transporte precisam ser usados para garantir que os dados de áudio e vídeo sejam mostrados continuamente, na ordem correta e em sincronismo;

e) é necessário que existam algumas operações padrão para as aplicações gerenciarem o transporte e apresentação de dados multimídia.

Existem hoje inúmeras discussões sobre como a Internet poderia atuar para melhor acomodar o tráfego multimídia com todas as suas implicações. Por um lado, alguns pesquisadores argumentam que não é necessário fazer

modificações nos serviços e protocolos atuais, mas sim, adicionar mais largura de banda às conexões. Opostos a esse ponto de vista, outros defendem a idéia que a adição de largura de banda pode ser dispendiosa e, em breve surgiriam novas aplicações multimídia que consumiriam essa banda extra [ROSS,2001].

Enfocando outra abordagem, alguns pesquisadores aconselham mudanças fundamentais na Internet, de forma que as aplicações possam reservar explicitamente largura de banda em uma conexão. Tais estudiosos acreditam que, se as aplicações indicarem o tráfego que pretendem realizar durante a comunicação, ao mesmo tempo em que a rede efetuar o policiamento dessas conexões a fim de verificar seu consumo real, o aproveitamento de largura da banda existente será maior. Tais mecanismos, quando combinados, requerem novos e complexos softwares nos *hosts* e roteadores, assim como novos tipos de serviços. [ROSS,2001]

2.4 PROTOCOLOS DE TEMPO REAL

Um grupo de pesquisa da IETF conhecido por *Integrated Services Working Group* desenvolveu um avançado modelo de serviço para a Internet chamado *Integrated Services* de tempo real que pode ser visto através do RFC 1633. Esses serviços irão desde habilitar redes IP a fornecer serviços de qualidade para aplicações multimídia, [LIU,2000].

Formado por protocolos como o **RSVP** (*Resource ReServation Protocol* – Protocolo de Reserva de Recursos), juntamente com o **RTP** (*Real-Time Transport Protocol* – Protocolo de Transporte em Tempo Real), **RTCP** (*Real-Time Control Protocol* – Protocolo de Controle de Tempo Real) e o **RTSP** (*Real-Time Streaming Protocol* – Protocolo de Fluxo Contínuo em Tempo Real), o RFC 1633 fornece fundamentos para serviços de tempo real. Sua proposta trata de permitir a configuração e o gerenciamento das aplicações tradicionais e multimídia em uma única infra-estrutura.

Atualmente existe também o protocolo **RTMP** (*Real Time Message Protocol* – Protocolo de mensagens em tempo real), desenvolvido pela *Adobe*, para a plataforma FLASH MEDIA SERVER, para uso em streamings de audio e video na internet.

2.4.1 – REAL TIME STREAMING PROTOCOL (RTSP)

De acordo com [SCH,1998], o RTSP é um protocolo do nível da aplicação para controle dos dados transmitidos na rede com propriedades de tempo real. Ele provê uma arquitetura para habilitar o controle da transmissão de conteúdo multimídia, como por exemplo, vídeo sob demanda.

A descrição do RTSP, definida no RFC 2326 (disponível em http://www.ietf.org,18/05/2007), especifica as seguintes características para o protocolo:

a) estabelece e controlam fluxos de mídia contínua como áudio e vídeo, atuando como um controle remoto de rede para servidores multimídia, permitindo ao cliente realizar operações similares àquelas disponíveis em videocassetes, como: pausa, avanço, retorno e interrupção de uma apresentação;

b) suporta as seguintes operações:

- recuperação de mídia de um servidor: um cliente pode requisitar uma apresentação a um servidor via HTTP ou outros métodos;
- convite de um servidor a uma conferência: um servidor de mídia pode ser “convidado” a participar de uma conferência existente, tanto para apresentar quanto para gravar o conteúdo multimídia;
- adição de uma mídia a uma apresentação existente: particularmente para

mídia ao vivo. Interessante quando o servidor deseja informar ao cliente a chegada de um novo conteúdo;

c) sua sintaxe e operação são intencionalmente similares ao HTTP/1.1 assim, a infra-estrutura existente pode ser reutilizada. Porém, introduz alguns métodos novos e seu próprio identificador de protocolo;

d) novos métodos e parâmetros podem ser facilmente acrescentados;

e) mensagens de RTSP podem ser transportadas tanto via UDP quanto TCP;

f) reutiliza mecanismos de segurança da WEB (como mecanismos de autenticação HTTP) ou mecanismos da camada de transporte e de rede;

g) o cliente pode negociar o método de transporte (TCP ou UDP) que precisar para processar um *stream* de mídia contínua;

h) se algumas características básicas estiverem desabilitadas, o cliente pode determinar o que não está implementado e apresentar ao usuário a interface mais apropriada, desabilitando os mecanismos não disponíveis.

As seguintes tarefas não são realizadas pelo RTSP:

a) não define métodos de compressão de áudio e vídeo;

b) não define como o sinal é encapsulado em pacotes para transmissão em uma rede. O encapsulamento para *streaming media* pode ser realizado pelo protocolo RTP ou um protocolo proprietário;

c) não restringe como a mídia será transportada. Pode ser via UDP ou TCP;

d) não restringe como um reprodutor de mídia armazena o sinal de vídeo em um *buffer*. O vídeo pode ser mostrado assim que chegar ao cliente ou com o atraso de alguns segundos ou ainda pode ser transferido totalmente antes de ser exibido.

A Figura 2.1 apresenta um browser requisitando uma apresentação a um servidor.

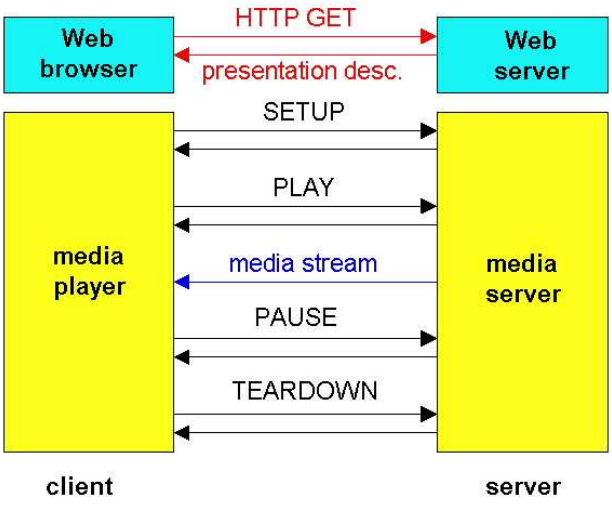


Figura 2.1 - Interação entre cliente e servidor usando RTSP [ROSS,2001]

Neste exemplo, o servidor Web encapsula a descrição do arquivo de apresentação em uma resposta HTTP e envia a mensagem ao browser solicitante. Quando o browser recebe a resposta, invoca o reprodutor de mídia

(comumente conhecido por *media player*) baseado no tipo do conteúdo retornado pelo servidor Web. Em seguida, o reprodutor e servidor de mídia trocam uma série de mensagens RTSP entre si. Requisições como PLAY e PAUSE enviadas pelo reprodutor permitem ao usuário controlar a exibição da mídia.

Cada sessão iniciada com uma requisição RTSP SETUP possui um identificador fornecido pelo servidor em uma resposta RTSP SETUP. O cliente repete o identificador de sessão para cada requisição até fechar a sessão com uma requisição TEARDOWN.

Na figura 2.2 se pode ver um exemplo simplificado de uma sessão RTSP entre o cliente (**C:**) e o servidor (**S:**).

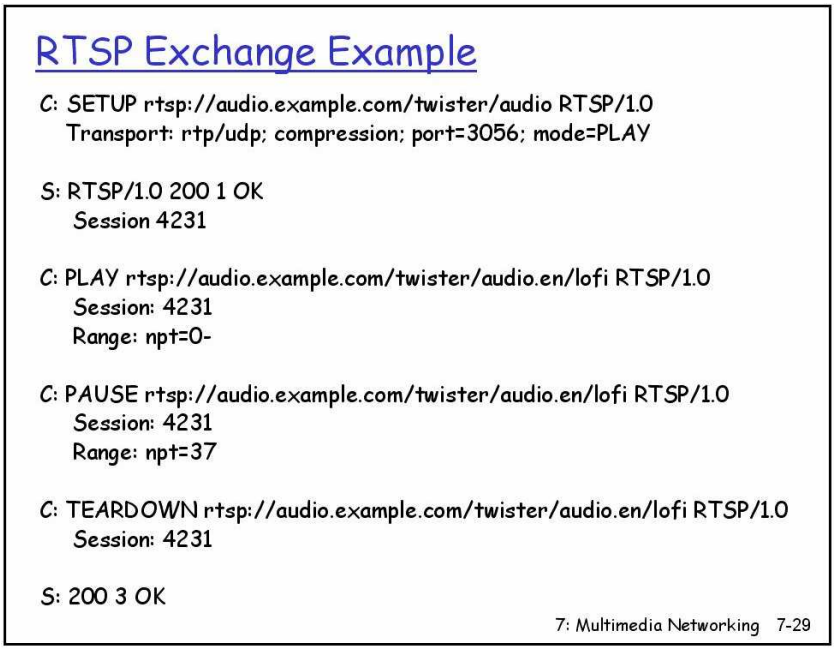


Figura 2.2 – Troca de Mensagens RTSP entre cliente e servidor [ROSS,2001].

2.4.2 REAL-TIME PROTOCOL (RTP)

De acordo com [SCH,1996], o RTP é um protocolo de transporte de rede fim-a-fim que oferece funções para aplicações transmitirem dados em tempo real, como áudio e vídeo, através de serviços de rede *unicast* ou *multicast*. O

RTP não realiza reserva de recursos nem fornece garantia de qualidade (*QoS – Quality of Service*) para serviços de tempo real. O transporte de dados é acrescido por um protocolo de controle, o RTCP, que permite o monitoramento da entrega dos dados até em grandes redes *multicast* além de fornecer um serviço mínimo de identificação.

Segundo [ROSS,2001], o RTP pode ser utilizado para o transporte de formatos comuns como PCM ou GSM para som e MPEG1 e MPEG2 (descrito na seção 3.5) para vídeo, além de poder ser utilizado para transportar outros formatos proprietários. O RTP também é descrito na arquitetura de protocolos H.323 para uso em aplicações de vídeo interativo em tempo real como áudio e vídeo conferência.

Como já foi citado na seção 2.3, a Internet é uma rede de transmissão de pacotes que são encaminhados independentemente através de redes compartilhadas, não garantindo a chegada dos pacotes ao destino na mesma ordem que foram enviados. Porém, aplicações multimídia requerem apropriada temporização na transmissão e exibição dos dados. O RTP fornece registro do tempo (*timestamping*), numeração seqüencial e outros mecanismos para tratar dessas dificuldades.

Para [LIU,2000], *timestamping* é a informação mais importante para aplicações de tempo real. O transmissor define o seu valor de acordo com o instante em que o primeiro *byte* da primeira amostra do pacote é gerada. Este valor é acrescido do tempo percorrido por um pacote, de acordo com o número de amostras que o formam. Depois de receber os pacotes, o receptor utiliza o *timestamping* para reconstruir a ordem de tempo original e exibir a informação na ordem correta.

O UDP não entrega os pacotes em ordem de tempo. Assim, números seqüenciais são usados para colocá-los na ordem correta, além de serem usados para detecção da perda dos mesmos. Basicamente, o RTP é um protocolo que funciona sobre o UDP, onde partes da informação multimídia gerada por codificadores são encapsuladas em pacotes RTP. No lado da aplicação responsável pelo envio da informação, cada pacote RTP é encapsulado em segmentos UDP para então ser encaminhado às camadas inferiores da arquitetura Internet. Da parte do receptor, pacotes RTP chegam à aplicação através de *sockets* UDP. É de responsabilidade da aplicação extrair

a informação multimídia dos pacotes [ROSS,2001], a figura 2.3 mostra um pacote RTP encapsulado em UDP e posteriormente em IP.

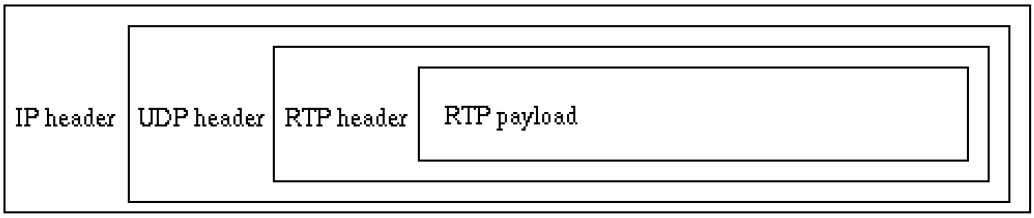


Figura 2.3 - Pacote de dados RTP no pacote IP.[LIU,2000]

Da perspectiva do desenvolvedor, o RTP faz parte da camada de aplicação, ou seja, o encapsulamento e extração de pacotes RTP em *sockets* UDP são de responsabilidade do desenvolvedor na implementação da aplicação, a figura 2.4 mostra a pilha de protocolos, onde podemos perceber que o RTP está diretamente ligado a camada de aplicação e aos pacotes UDP.

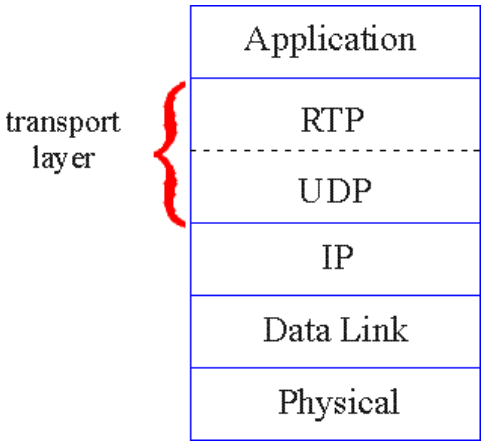


Figura 2.4 - O Protocolo RTP faz parte da camada de aplicação [ROSS,2001].

Em um pacote, o dado RTP é conhecido por *payload* (carga útil) e sua interpretação pela aplicação é determinada pelo campo *payload type*, que pode assumir um dos tipos padrão, definidos no RFC 1890, ou, então, utilizar um formato proprietário. A tabela 2.1 descreve alguns dos formatos padrão para áudio e vídeo:

Tabela 2.1 – Numero do payload para alguns formatos de vídeo [RFC,2007].

Número do payload	Formato de áudio	Formato de vídeo
3	GSM	–
9	G.722	–
14	Áudio MPEG	–
15	G.728	–
26	–	Motion JPEG
31	–	H.261
32	–	Vídeo MPEG 1
33	–	Vídeo MPEG 2

2.4.3 REAL-TIME CONTROL PROTOCOL (RTCP)

O RFC 1889 também especifica o RTCP como sendo um protocolo que aplicações multimídia de rede possam usar em conjunto com o RTP.

Em uma sessão RTP, pacotes RTCP são transmitidos entre os participantes daquela sessão, contendo relatórios estatísticos que podem ser úteis para a aplicação. Tais estatísticas incluem: número de pacotes enviados, número de pacotes perdidos e variações nos atrasos durante a transmissão. A especificação do RTP não indica o que a aplicação deve fazer com tais informações; esta tarefa fica a encargo do desenvolvedor [ROSS,2001].

Para determinar os participantes de uma sessão, o RTCP pode usufruir de mecanismos de distribuição como IP Multicast, ou, os receptores dos pacotes RTP podem manter uma tabela dos participantes, anotando as origens dos *streams*, disponíveis no campo SSRC (*synchronization source)* e as origens do *payload*, encontradas no campo CSRC (*contributing source*) contidos nos pacotes RTP recebidos.

2.4.3.1 TIPOS DE PACOTES RTCP

Para cada *stream* que um receptor recebe em uma sessão RTP, é gerado um relatório de recepção agregado a pacotes RTCP. O pacote é então enviado a todos os participantes daquela sessão (ROSS2001).

Relatórios de recepção incluem diversos campos, dos quais os mais importantes são listados abaixo:

a) a origem do *stream* de RTP para o qual o relatório está sendo gerado;

b) a fração de pacotes perdidos dentro do *stream*. Cada receptor calcula o número de pacotes RTP perdidos dividido pelo número de pacotes RTP enviados como parte de um *stream*;

c) o último número seqüencial (*sequence number*) recebido no *stream* de pacotes;

d) a variação nos atrasos, que é calculada como sendo a média de tempo de chegada entre sucessivos pacotes no *stream* de RTP.

Para cada *stream* de RTP que o remetente está transmitindo, são criados e transmitidos relatórios RTCP. Esses pacotes incluem informação sobre o *stream* de RTP, incluindo:

a) a origem do *stream* de RTP;

b) o *timestamp* e tempo real do pacote RTP mais recentemente gerado no *stream*;

c) o número de pacotes enviados no *stream*;

d) o número de bytes enviados no *stream*.

Relatórios do remetente podem ser utilizados para sincronizar diferentes *streams* de mídia dentro de uma sessão RTP. Por exemplo, uma aplicação de videoconferência pode gerar dois *streams* separados, um para vídeo e outro para áudio. Os valores do *timestamp* nesses pacotes estão ligados com o momento da amostra de áudio e vídeo e não com o tempo real em que os sinais foram gerados. Como cada relatório do remetente contém o *timestamp* do pacote RTP e o tempo real em que o pacote foi criado, o receptor pode usar essa associação para sincronizar a exibição do áudio e vídeo [ROS2001].

Para cada *stream* RTP que o remetente está transmitindo, também são criados pacotes com a descrição da origem dos mesmos. Estes pacotes contêm informações como endereço de e-mail, nome e a aplicação geradora do *stream*. Ele também inclui a origem do RTP associado [ROSS,2001].

2.4.4 RESOURCE RESERVATION PROTOCOL (RSVP)

RSVP é um protocolo de controle que permite aos receptores requisitar uma determinada qualidade de serviço para seu fluxo de dados. Aplicações de tempo real utilizam o RSVP para reservar a quantidade necessária de recursos

e roteadores ao longo do caminho de transmissão, a fim de obter disponibilidade da largura de banda necessária para a transmissão [LIU,2000]. A figura 2.5 ilustra o esquema base deste protocolo.

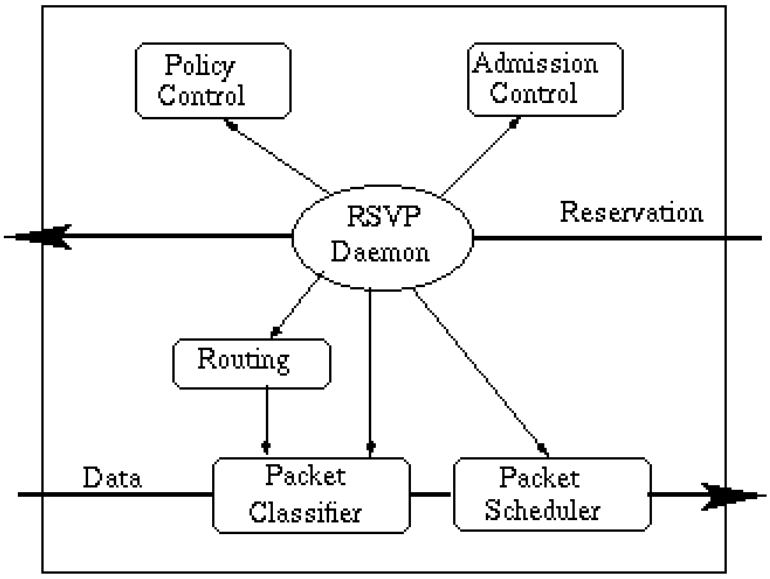


Figura 2.5 – Reserva de um nó da rede no caminho do fluxo de dados [LIU,2000]

A seguir, são descritos os elementos que compõem o esquema da Figura 2.5 [LIU2000]:

a) **Policy Control**: trata das permissões da entidade que efetuou o pedido. Nomeadamente administração, autenticidade, controle de acesso, pagamento de serviços, etc.;

b) **Admission Control**: verifica os recursos físicos da rede, decidindo se o pedido pode ou não ser atendido;

c) **RSVP Daemon**: analisa os resultados dos blocos anteriores. Caso um deles falhe, envia uma notificação de erro à entidade que efetuou o pedido. Caso contrário, estabelece os parâmetros dos blocos *packet classifie*r e *packet scheduler* de forma a atender ao pedido;

d) **Packet scheduler**: ordena os vários pacotes a serem transmitidos;

e) **Packet classifier**: atribui aos vários pacotes de informação diferentes categorias de QoS correspondente.[LIU,2000] descreve as seguintes características para o RSVP:

a) fluxos RSVP são *simplex*: RSVP distingue remetentes de receptores. Mesmo que, um *host* possa atuar tanto como remetente quanto como receptor, uma reserva RSVP somente aloca recursos para *streams* em uma direção;

b) suporta *unicast* e *multicast*: desde que a reserva seja iniciada pelo receptor e seu estado seja leve, RSVP pode facilmente manipular mudanças em membros e rotas. Um *host* pode enviar mensagens IGMP (*Internet Group Management Protocol*) para inscrever-se em um grupo *multicast*;

c) o RSVP é orientado ao receptor e manipula receptores heterogêneos: em grupos heterogêneos de *multicast*, receptores têm diferentes capacidades e níveis de QoS. Os remetentes podem dividir o tráfego em muitos fluxos, cada um é um fluxo RSVP com diferente nível de QoS.

d) compatibilidade: esforços têm sido feitos para executar o RSVP tanto sobre o IPv4 (protocolo IP versão 4, atualmente em uso) quanto sobre o IPv6 (futura versão desenvolvida para o protocolo IP).

2.5 ENVIANDO MULTIMÍDIA DE UM SERVIDOR DE STREAMING PARA UMA APLICAÇÃO

Uma das formas de fornecer vídeo pela Internet se dá através dos servidores de *streaming*, que tratam de enviar sinais de vídeo para reprodutores de mídia. Este servidor pode ser proprietário, como aqueles comercializados pela RealNetworks (Real ou Helix Media Server), Microsoft (Media Server), Adobe (FMS2) ou um servidor de *streaming* de domínio público [ROSS,2001]. Com um servidor de *streaming*, áudio e vídeo podem ser enviados sobre UDP (preferencialmente, em relação ao TCP) utilizando protocolos da camada de aplicação, mais adequados do que *streaming* via servidor HTTP.

Esta arquitetura requer dois servidores, como mostra a Figura 2.6. Um servidor de HTTP para servir às páginas Web; e um servidor de *streaming*, para servir aos arquivos de vídeo. Os dois servidores podem executar em um

mesmo sistema ou em dois sistemas distintos. Uma simples arquitetura para servidor de *streaming* descreve os seguintes passos:

a) um navegador Web faz uma requisição HTTP para obter um arquivo de descrição da apresentação, conhecido também por arquivo *metafile*.

b) o servidor Web retorna o arquivo utilizado ao navegador para identificar o reprodutor de mídia adequado para exibir o vídeo;

c) o arquivo é repassado para o reprodutor contendo informações referentes à localização do vídeo no servidor de *streaming*;

d) o reprodutor faz a requisição do arquivo diretamente ao servidor de *streaming*. A partir deste momento, servidor e reprodutor interagem diretamente utilizando seus próprios protocolos e pode permitir uma interação maior do usuário com o *stream* de vídeo.

A arquitetura demonstrada na Figura 2.6 possui várias opções para entregar o vídeo do servidor para os clientes (*media players*).

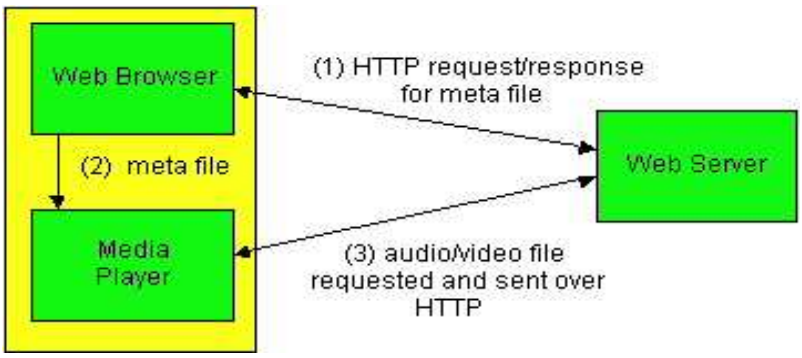


Figura 2.6 – Streaming de um servidor para um reprodutor de mídia [ROSS,2001].

Algumas delas são descritas a seguir:

a) o vídeo é transportado sobre o UDP a uma taxa constante, igual àquela com que o receptor apresenta o conteúdo (esta taxa é equivalente à taxa de codificação do vídeo);

b) similar à anterior, nesta opção o receptor atrasa a exibição por 2-5 segundos para eliminar a variação nos atrasos em pacotes do mesmo *stream*, que podem ocorrer em função da rede. Esta tarefa é realizada pelo cliente, armazenando o vídeo recebido em um *buffer*. Uma vez que o cliente guardou alguns segundos da mídia, ele começa a “descarregar”

o *buffer*. Novamente, a taxa de preenchimento do *buffer* é igual à taxa com que é descarregado, exceto na ocorrência da perda de pacotes;

c) processo semelhante ao descrito, porém, neste caso, a mídia é transportada sobre o TCP. Como o TCP retransmite pacotes perdidos, desta maneira pode-se oferecer uma qualidade melhor do que com o UDP. Por outro lado, a taxa de preenchimento do *buffer* varia conforme a perda de pacotes, o que pode ocasionar um esvaziamento do *buffer*, resultando indesejáveis pausas na exibição do vídeo no cliente.

2.6 CODIFICAÇÃO E COMPRESSÃO DE VÍDEO

Antes da transmissão do vídeo através de um computador na rede, ele precisa ser digitalizado e comprimido. A necessidade por digitalização é óbvia: os computadores transmitem bits na rede, assim, toda informação precisa ser representada como uma seqüência de bits. A compressão é importante porque o vídeo não comprimido consome uma quantidade muito grande de armazenamento e largura de banda.

Um vídeo é uma seqüência de imagens, normalmente exibidas à uma taxa constante, por exemplo a 24-30 imagens por segundo. Uma imagem digital não comprimida consiste em uma matriz de pontos, sendo cada ponto codificado em um número de bits que representam luminosidade e cor [ROS2001].

Existem dois tipos de redundância em vídeo, os quais podem ser explorados para compressão:

a) **espacial**: existente na imagem fornecida. Por exemplo, uma imagem que consiste de muitos espaços em branco pode ser eficientemente comprimida;

b) **temporal**: consiste na repetição da imagem numa imagem subseqüente. Se por exemplo, uma imagem e a subseqüente imagem forem exatamente iguais, não há razão para re-codificar a imagem, é mais eficiente simplesmente indicar durante a codificação que a imagem subseqüente é igual a anterior.

O padrão de compressão conhecido por MPEG (*Motion Pictures Experts Group*) é, sem dúvida, a técnica de compressão mais popular. Este inclui o MPEG 1 para vídeo com qualidade de CD-ROM (1.5 Mbps), MPEG 2 para vídeo com qualidade de DVD (3-6 Mbps), e MPEG 4 para compressão de vídeo orientada a objetos. O padrão de compressão H.261 que faz parte da arquitetura de protocolos H.323 também é muito popular na Internet. Existem ainda inúmeros padrões de compressão proprietários.

2.7 STREAMING EM REDES MÓVEIS

O Streaming de Vídeo em dispositivos móveis é usada na transmissão de dados em tempo-real, de um servidor para um cliente, onde o cliente decodifica e toca, ou apresenta, os dados conforme eles vão sendo recebidos. A Streaming de Vídeo proporciona ao usuário a capacidade de visualizar a imagem, imediatamente após um curto período de armazenamento *(Buffering).* Uma vantagem da Streaming, comparada à visualização de arquivos pré-armazenados no dispositivo, é que nenhuma informação é armazenada permanentemente no dispositivo móvel cliente. Neste caso o usuário final não pode retroceder o vídeo nem enviá-lo a outro usuário, o conteúdo deve ser "recarregado" cada vez que o usuário quiser vê-lo, isto favorece muito os Provedores de Serviço Móvel [NOKIA V3.0 ,2005].

A largura de banda disponível para transferência de conexões de Streaming deve variar de acordo com o tipo de tecnologia de acesso da rede móvel usada (Provedor GSM/WCDMA/HSDPA etc), e da configuração de rede do provedor de serviço,.se o conteúdo for apenas disponibilizado dentro do domínio do provedor de serviços, diretrizes de autorização são aplicadas no lado deste provedor [NOKIA V3.0 ,2005].

A figura 2.7 mostra de maneira simplificada, o processo padrão, recomendado pela Nokia, para geração e distribuição de streaming de vídeo para redes móveis.

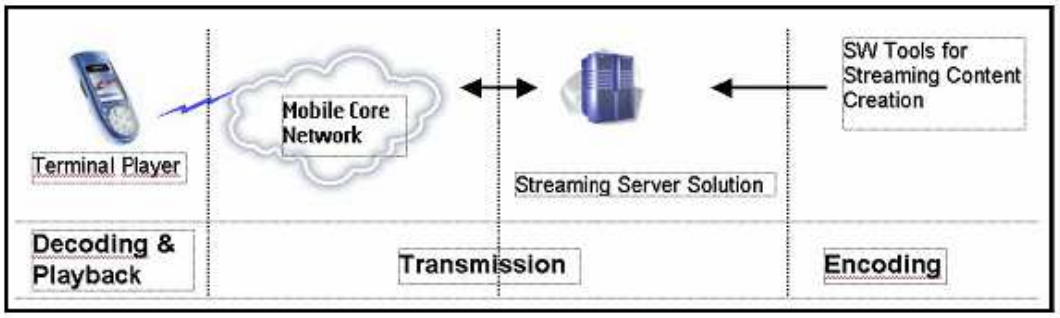


Figura 2 7 – Streaming em Redes Móveis [NOKIA V3.0, 2005].

Para o desenvolvimento deste projeto serão usados os protocolos RTP/RTSP para envio de streaming, do sistema de captação de imagens para o servidor de streaming, bem como do servidor de streaming para o dispositivo móvel. O protocolo RTP/RTSP é um dos mais difundidos para aplicações e áudio e vídeo na internet.

O processo de geração de streaming recomendado pela NOKIA também é utilizado neste projeto, as funções de codificação e transmissão de pacotes RTP demonstradas na figura 2.7, são basicamente executadas pelo codec HMPROD, que será citado no próximo capítulo, este processo é recomendado pela maioria dos fabricantes de sistemas de streaming.

2.8 REDES GSM E UMTS

2.8.1 INTRODUÇÃO

A finalidade deste item é expor de forma simplificada os diferentes sistemas de uma rede GSM/UMTS bem como definir alguns conceitos como NSS, BSS, e GPRS que serão utilizados posteriormente no desenvolvimento do projeto.

Basicamente os sistemas GSM e UMTS, na atual release 4 do 3GPP, compartilham dos mesmos Subsistemas NSS (Core PS e CS), OSS e VAS,

havendo diferença apenas na tecnologia de acesso. No GSM é utilizado o Subsistema BSS, e para o UMTS o UTRAN.

Os Subsistemas básicos para as duas tecnologias são:

- Core NSS (Network System Switch) ou *Circuit Switch* - É formado pelos elementos MSC/MSS, MGW, HLR/VLR AUC, atualmente as Plataformas de Valor agregado (VAS) e de Rede Inteligente (IN) também fazem parte do NSS. É responsável pela, comutação, roteamento, e trânsito para o tráfego de usuários, contém o banco de dados e as funções de gerenciamento da rede, e disponibilizam os serviços de rede inteligente (Pré Pago), e Valor agregado como Short Message, Voice Mail, WAP, Localização (LBS), Streaming de vídeo, e outros

- OSS (Operation and suport system) – São os sistemas de monitoração e controle da rede GSM e da rede UMTS.

Para rede GSM

- BSS (Base Station System), – Composto pelas BSC's e BTS's e pelas MS (Mobile Stations – Aparelhos Móveis).

Na rede GSM a tecnologia de acesso é baseada em TDMA (Time Division Multiple Access).

Para rede UMTS

- UTRAN (Terrestrial Radio Access Network) – Composto pelas RNCs, Node B e UE (User Equipament – Aparelhos Móveis).

Na rede UTRAN a tecnologia de acesso é baseada em WCDMA (Wideband Code Division Multiple Access).

- Core GPRS (General Packet Radio Services) - é a rede de tranporte de dados do GSM e do UMTS e funciona paralelo a rede de voz, e não depende dela, ou seja, voz e dados nas redes GSM/GPRS/EDGE e UMTS/HSPA utilizam diferentes recursos de hardware tornando as duas redes independentes.

Nas redes GSM utiliza-se tecnologia EDGE (Enhanced GPRS) na rede de acesso para elevar as taxas de transporte de dados até valores próximos de 384Kbps. Nas redes UTRAN a taxa máxima de transferência de dados pode chegar a valores próximos de 2Mbps, mas utilizando tecnologias de acesso como HSPA (High Speed Packet Access) este valor pode chegar a 14,4Mbps (teórico) [Evolium, 2000].

A figura 2.8, abaixo mostra os subsistemas no contexto das redes GSM e UTRAN e os elementos básicos da rede GPRS [GSM,1998

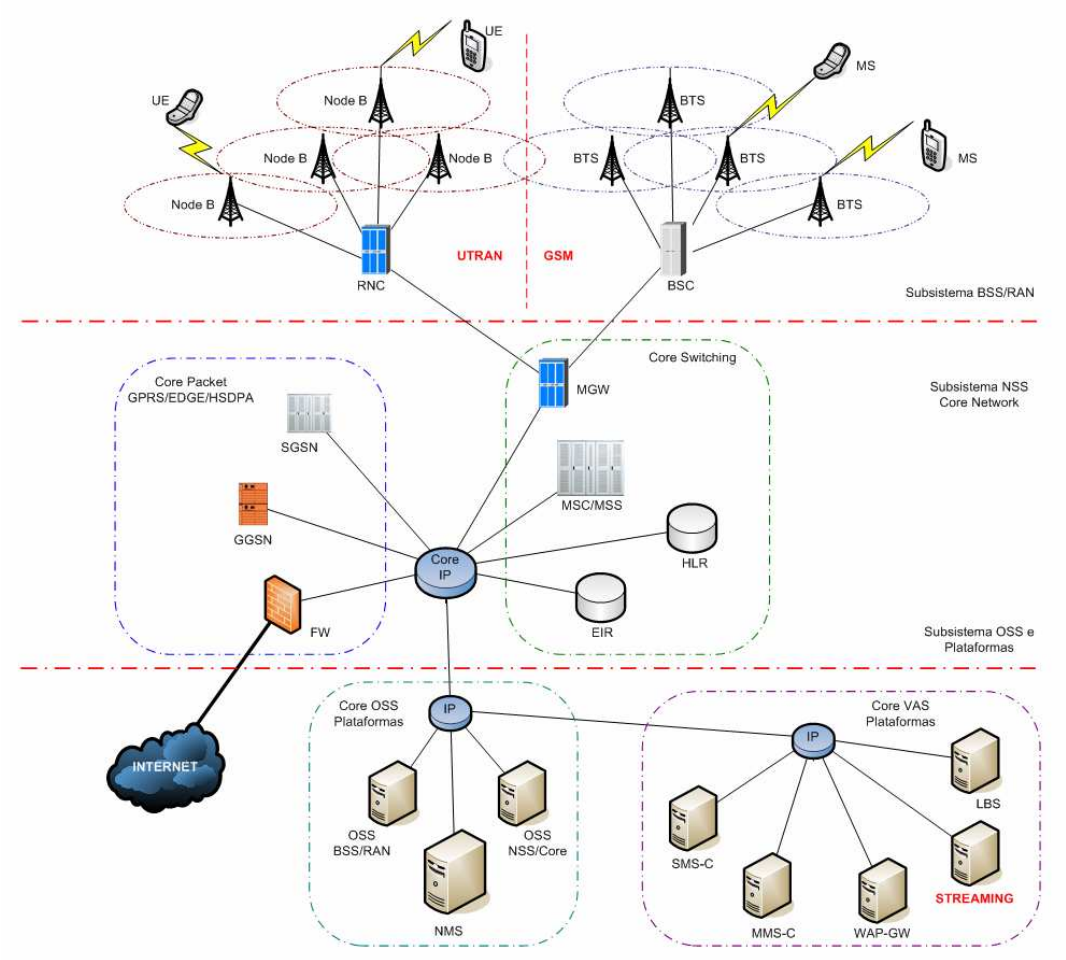


Figura 2.8 – Subsistemas das redes GSM e UTRAN [Evolium, 2000].

2.9 REDES WI-FI

Wi-Fi Foi uma marca licenciada originalmente pela Wi-Fi Alliance para descrever a tecnologia de redes sem fio embarcadas (WLAN) baseadas no padrão IEEE 802.11. O termo Wi-Fi foi escolhido como uma brincadeira com o termo "Hi-Fi" e pensa-se geralmente que é uma abreviatura para **wi**reless **fi**delity, no entanto a Wi-Fi Alliance não reconhece isso. Comumente o termo Wi-Fi é entendido como uma tecnologia de interconexão entre dispositivos sem fio, usando o protocolo IEEE 802.11.[www.wi-fi.org,2007]

O padrão Wi-Fi opera em faixas de freqüências que não necessitam de licença para instalação e/ou operação. Este fato as torna atrativas. No entanto, para uso comercial no Brasil é necessária licença da Agência Nacional de Telecomunicações (Anatel).

Para se ter acesso à internet através de rede Wi-Fi deve-se estar no raio de ação de um ponto de acesso (normalmente conhecido por hotspot) ou local público onde opere rede sem fios, deve-se usar um dispositivo móvel, como NoteBook, Tablet PC, Handheld ou Celular com suporte a Wi-Fi.

Hoje,muitas operadoras de telefonia estão investindo pesado no Wi-Fi, para ganhos empresariais.

O Hotspot Wi-Fi existe para estabelecer um ponto de acesso para conexão à internet. O ponto de acesso transmite o sinal sem fio numa pequena distância – cerca de 100 metros. Quando um periférico que permite Wi-Fi, como um Pocket PC, encontra um hotspot, o periférico pode na mesma hora conectar na rede sem fio. Muitos hotspots estão localizados em lugares que são acessíveis ao público, como aeroportos, cafés, hotéis e livrarias. Muitas casas e escritórios também têm redes Wi-Fi. Enquanto alguns hotspots são gratuitos, a maioria das redes públicas é suportada por Provedores de Serviços de Internet (Internet Service Provider - ISPs) que cobram uma taxa dos usuários para conectar.

Atualmente praticamente todos os computadores portáteis vêm de fábrica com dispositivos para rede sem fio no padrão Wi-Fi (802.11b, a, g ou n).

O que antes era acessório está se tornando item obrigatório, principalmente devido ao fato da redução do custo de fabricação.

2.9.1 OPERAÇÃO DE REDES Wi-Fi

Redes deste tipo operam em um dos dois modos disponíveis – ad-hoc (sem infra-estrutura) ou infra-estruturado. O padrão IEEE define o modo ad-hoc como Independent Basic Service Set (IBSS), e o modo infra-estruturado como Basic Service Set (BSS).

No modo ad-hoc cada estação pode se comunicar diretamente com outras estações na rede, como exemplifica a figura 2.9. O modo ad-hoc foi projetado de forma que apenas as estações que se encontrem dentro do alcance de transmissão (mesma célula) umas das outras possam se comunicar. Se uma das estações (STA A) quiser se comunicar com outra fora de seu alcance (STA B), uma terceira estação (STA C) dentro do alcance de STA A e STA B, deve ser utilizada como gateway e fazer o roteamento.

Já no modo infra-estruturado, cada cliente envia todas as suas mensagens para uma estação central, o Ponto de Acesso ou Access Point (AP). Esta estação central funciona como uma ethernet bridge, repassando as mensagens para a rede apropriada, ou seja, uma rede cabeada, ou para a própria rede sem fio. A figura 2.10 mostra este tipo de topologia.

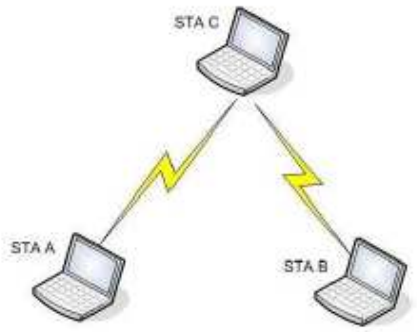


Figura 2.9: Exemplo de redes sem fio adotando topologia ad-hoc.

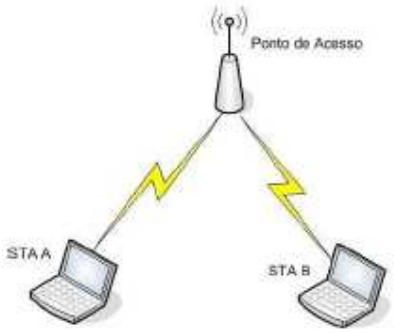


Figura 2.10: Exemplo de redes sem fio adotando modo infraestruturado, onde um AP é utilizado na comunicação.

Antes de trocarem dados, STAs e APs devem estabelecer um relacionamento, ou uma associação. Apenas depois que a associação for estabelecida as duas estações podem trocar dados. No caso de uma rede com infra-estruturada, as STAs se associam com o AP. O processo de associação possui dois passos, envolvendo três dos seguintes estados:

1. Não autenticado e não associado,
2. Autenticado e não associado, e
3. Autenticado e associado.

Para mudar de estados, as partes interessadas na comunicação trocam mensagens denominadas quadros de gerenciamento, ou management frames.

No entanto, uma STA precisa saber se existe algum AP dentro do alcance de seu rádio e a qual AP ela deve se associar. Assim, todos os APs transmitem um quadro de gerenciamento chamado beacon em intervalos fixos de tempo. Para se associar a um AP e se unir a uma BSS, uma estação procura escutar beacons para identificar APs dentro do alcance de seu rádio. Então a STA seleciona a qual BSS ela deseja se unir. Em seguida, AP e STA trocam diversas mensagens de gerenciamento com o objetivo de realizar uma autenticação mútua. Dois mecanismos de autenticação são previstos no padrão, mas não serão abordados neste trabalho, pois fogem ao tema principal do mesmo.

Após uma autenticação bem sucedida, a STA passa para o segundo estado, autenticado e não associado. Passar do segundo estado para o terceiro, autenticado e associado, requer que a STA envie um pedido de

associação ao AP e este, em seguida, responda com a aceitação. Uma vez que esse processo ocorre, a STA passa a ser um ponto (peer) da rede sem fio e pode, então, transmitir quadros através da mesma.

O projeto proposto utiliza as redes GSM e Wi-Fi, como tecnologias de acesso dos dispositivos móveis a internet, o objetivo é demonstrar que a utilização do sistema não depende da tecnologia móvel utilizada no aceso, e que o sistema se adapta ao conceito de convergência de tecnologias de acesso a serviços de dados móveis.

2.10 SERVIDOR DE STREAMING

Os servidores de streaming são softwares que tem a capacidade de disponibilizar para os dispositivos clientes, streamings de áudio e/ou vídeo, em modo armazenado ou ao vivo.

Modo armazenado: O arquivo acessado pelo cliente encontra-se armazenado no disco do servidor de streaming.

Modo ao vivo: A streaming é gerada por Hardware ou Software, é enviada ao servidor de streaming, que disponibiliza o conteúdo aos clientes que solicitarem.

A grande vantagem dos servidores de streaming é permitir que o software cliente disponibilize o conteúdo para o usuário sem a necessidade de armazenar o arquivo em disco, e disponibilizar em alguns casos ajuste de banda entre servidor e cliente, para se adaptarem a banda disponibilizada.

A figura 2.11 mostra a utilização de um servidor de streaming recebendo uma streaming ao vivo, que pode ser de áudio ou vídeo, e disponibilizando-a para dispositivos na internet que tem a capacidade de se conectar a servidores RTP/RTSP. A figura destaca que a conexão ao servidor pode ser feita por clientes com acesso móvel ou fixo.

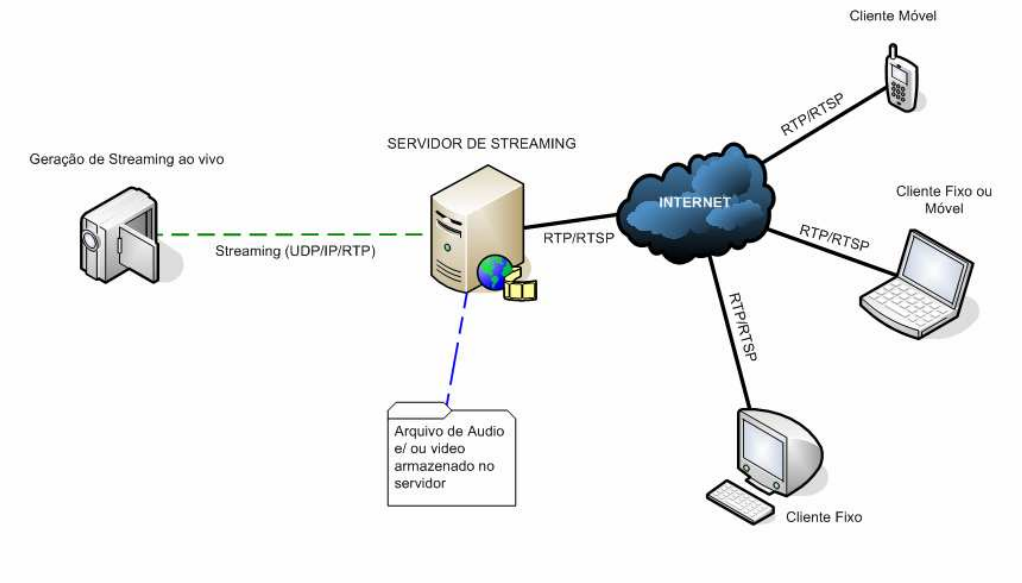


Figura 2.11: Servidor de Streaming[QTSS, 2006].

2.10.1 DARWING STREAMING SERVER

O *Darwin Streaming Server* é um servidor, dirigido por evento, com o código fonte aberto, desenvolvido pela Apple. Funciona como um conjunto de processos que executam os protocolos padrões de *streming media*, RTSP, RTP e RTCP. O Darwin opera fazendo um broadcast do conteúdo à todos clientes que solicitarem o arquivo, ou servindo sob demanda.

O *Darwin Streaming Server*, é uma versão livre do produto comercial *QuickTime Streaming Server*, da Apple. Esse servidor pode trabalhar com vídeos no formato MPEG-4 ou H263 e áudio MP3, AMR-NB (*Adaptive Multi Rate – Narrow Band*) ou AAC-LC (*Advanced Audio Coding – Low Complexity*).

2.11 LINGUAGEM C++ E BUILDER

A linguagem C é uma poderosa linguagem de programação, que permite o desenvolvimento de sistemas que não podem ser feitos em outras linguagens (links com outras linguagens, acesso direto ao Hardware, etc.). A grande importância do C++ sobre o C é o fato de que a mesma suporta a programação orientada a objetos, que tem se tornado essencial para o desenvolvimento de grandes aplicações.

O C++ Builder é um ambiente de desenvolvimento projetado pela Borland e tem como característica principal a alta produtividade na criação de aplicativos. Utiliza o conceito de orientação a objetos e RAD (Rapid Application Development). Neste tipo de ambiente, grande parte do código é gerado automaticamente, principalmente as rotinas de interface (criação de janelas, botões, controle do mouse, cursor etc.). Restando ao programador apenas a tarefa de inserir as linhas do programa referente à solução do problema (a definição das estruturas, cálculos necessários etc.),
que neste caso deve ser escrito através da linguagem C, ou mais especificamente da linguagem C++. É interessante observar que, já há algum tempo, este ambiente é bastante utilizado tanto na área acadêmica quanto comercial (por exemplo: desenvolvimento de softwares de várias naturezas, de sistemas operacionais, sistemas gráficos, etc) [Mateus,2000].

2.11.1 C++ BUILDER COMPONENTES (CLASSES) - VIDEOLAB

O VideoLab é um conjunto de componentes (CLASSES), para C++ Builder, desenvolvidos para o processamento de som e vídeo. Estes componentes permitem manipulações complexas de audio e vídeo em alta performance, utilizando poucas linhas de código de programa. A biblioteca suporta a antiga Win32 API (Video for Windows - VFW, VCM, WaveAPI, ACM), e os recentes DirectShow e DMO. Os componentes permitem a interação

entre estas APIs. O desenvolvedor pode escolher utilizar qualquer uma delas, usar todas, ou pode trocar de uma para outra em qualquer momento do desenvolvimento. Os componentes encobrem as complexidades destas tecnologias, e faz o desenvolvimento se tornar mais intuitivo. Inclui diversos componentes como: Vídeo Players e Video Loggers (Win32 e DicerctX/DirectShow) Video/Audio Capture e Video/Audio Output (Win32 e DicerctX/DirectShow) Filtros, transformações Geométricas Histogramas, Video Mixers, Direct Media Objects (DMO) Compressores/Decompressores, FreeFrame, Scopes e Waterfall.

A Biblioteca Video Lab é mantida pela MITOV Software (www.mitov.com), e distribuída gratuitamente para uso não comercial.

2.12 DIRECTX/DIRECTSHOW

O DirectX é um conjunto de interfaces de programação de aplicativo (APIs) de baixo nível que oferece suporte de multimídia acelerada por hardware de alto desempenho aos programas Windows [DirectX 9.0 SDK, 2004].

O DirectX permite que o programa determine facilmente os recursos de hardware do computador e defina os parâmetros do programa que devem ser correspondidos. Isso permite que os programas de software multimídia sejam executados em qualquer computador baseado no Windows com hardwares e drivers compatíveis com o DirectX e assegura que os programas de multimídia tirem proveito do hardware de alto desempenho [DirectX 9.0 SDK, 2004].

O DirectX contém um conjunto de APIs que fornece acesso aos recursos avançados de hardware de alto desempenho, como processadores de aceleração de elementos gráficos 3D e placas de som. Essas APIs controlam funções de nível baixo, inclusive a aceleração de elementos gráficos bidimensionais (2D), o suporte a dispositivos de entrada, como joysticks, teclados e mouses, além do controle da mixagem e da saída de som.

Os componentes que constituem o DirectX são:

- Microsoft DirectDraw
- Microsoft Direct3D
- Microsoft DirectSound
- Microsoft DirectMusic
- Microsoft DirectInput
- Microsoft DirectPlay
- Microsoft DirectShow

Para este projeto foram utilizados os recursos da API DirectShow do DirectX, os recursos do DirectShow foram controlados pelos componentes VideoLab do C++ Builder, citados no item 2.11, evitando assim a manipulação de componentes COM (Component Object Model) no código C++ desenvolvido. O uso das CLASSES VideoLab, oferecem uma interface muito mais intuitiva para a manipulação dos componentes DirectShow [DirectX 9.0 SDK, 2004].

2.12.1 DIRECTSHOW

A API Microsoft DirectShow fornece captura de alta qualidade e reprodução de arquivos multimídia localizados no seu computador e em servidores da Internet. O DirectShow oferece suporte a uma ampla variedade de formatos de áudio e vídeo, inclusive arquivos no formato ASF (Advanced Streaming Format), AVI (Audio-Video Interleaved), DV (Digital Video), MPEG (Motion Picture Experts Group), MP3 (MPEG Audio Layer-3), WMA/WMV (Windows Media Audio/Video) e WAV. O DirectShow permite a captura de vídeo (Placas de Captura,Webcams, etc), reprodução de DVD, edição e mixagem de vídeo, decodificação de vídeo acelerada por hardware e ajuste de sinais analógicos de difusão e de televisão digitais [DirectX 9.0 SDK, 2004].

A figura 2.12 mostra um exemplo do funcionamento da API DirectShow em aplicações multimídia.

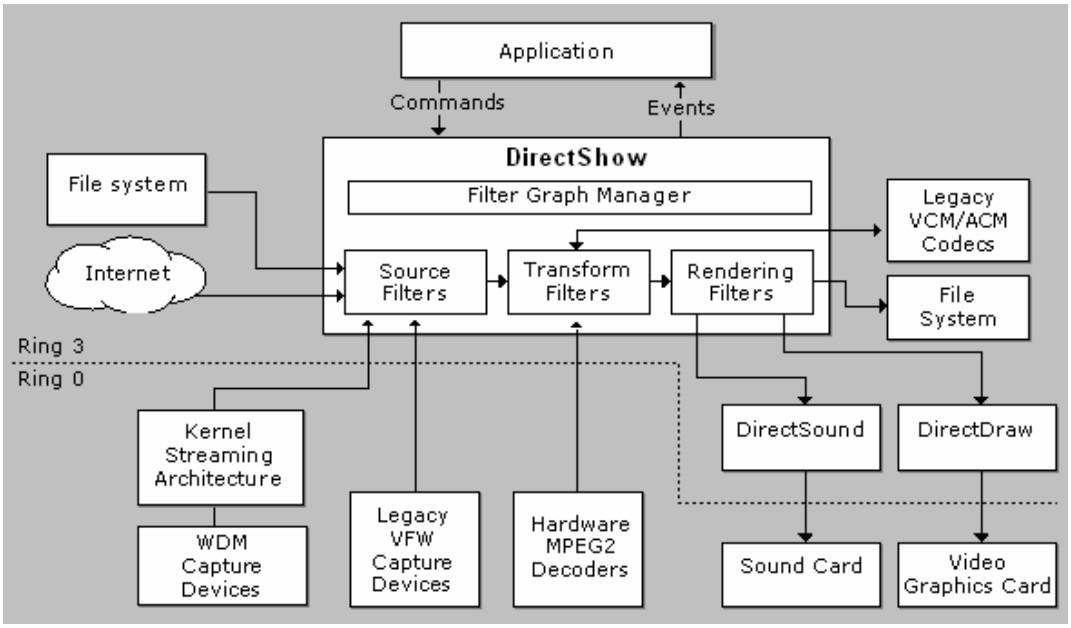


Figura 2.12: Estrutura do DirectShow [DirectX 9.0 SDK, 2004].

2.13 O 3GPP E CODECS

O *3rd Generation Partnership Project* (3GPP), é um acordo de colaboração que foi estabelecido em dezembro de 1998. Este acordo reuniu várias entidades de padronização da área de telecomunicações mundial que são conhecidos por *"Organizational Partners"* (Socios Organizacionais).

Os atuais "sócios organizacionais" são: ARIB, CCSA, ETSI, ATIS, TTA, e TTC [3GPP,2007].

O escopo original do 3GPP era produzir Especificações Técnicas globalmente aplicáveis e Relatórios Técnicos para a 3ª Geração de Sistema Móvel baseado no *Core* (Núcleo da rede) da rede GSM e nas tecnologias de acesso que ele suporta (ex., Universal Terrestrial Radio Access – UTRA, ambos os modos, Frequency Division Duplex – FDD, e Time Division Duplex - TDD). O escopo foi logo seguidamente alterado para incluir a manutenção e o desenvolvimento do sistema GSM (Global System for Mobile communication) Especificações e Relatórios Técnicos evoluíram as tecnologias de acesso de rádio (ex: General Packet Radio Service – GPRS, e aumentaram as taxas de Dados para Evolução do GSM - EDGE) [3GPP,2007].

O 3GPP também estabelece padrões para criação, envio e reprodução de arquivos multimídia (vídeos) em telefones celulares e outros aparelhos wireless (sem fio) GSM e WCDMA. O tipo de arquivo padronizado pelo 3GPP chama-se, “3gp” [3GPP,2007].

Este tipo de arquivo é uma versão mais simples do "ISO 14496-1 (MPEG-4) Media Format", muito similar aos arquivos MOV do Apple Quicktime. Assim como arquivos AVI ou MPEG, o 3GP pode ser codificado em mais de um tipo de codec - para vídeo pode-se usar MPEG-4 ou H.263 e para áudio os formatos AMR ou AAC-LC [3GPP,2007].

A tendência dos aparelhos de 3ª geração (nova conexão de alta velocidade, baseada na tecnologia WCDMA) é usar a dupla MPEG-4/AAC-LC. Aparelhos como os Nokia n90, n93 e n95 já são capazes de gravar vídeos com codec MPEG-4/AAC-LC de 352 x 288, qualidade similar a fitas de vídeo-cassete [3GPP,2007] .

A tabela 2.2 mostra os parâmetros de três tipos de Codecs recomendados pelo 3GPP para formar arquivos 3gp, na tabela podemos verificar quais parâmetros podem ou não, ser habilitados, e também a faixa de configuração de alguns parâmetros.

Tabela 2.2: Recomendações 3GPP – parâmetros de Codec

Parametros	Codec		
	H.263	MPEG-4	H.264/AVC
Interlace coding? (Y/N)	Y	Y	Y
Progressive coding? (Y/N)	Y	Y	Y
Optimized Bitrate Range (bits/sec)	>= 10k	>= 10k	>= 10k
Min. Picutre Size (width, height)	16x16	16x16	16x16
Max. Picture Size (width, height)	2048x1152	64kx64k	4096x2048
Variable aspect ratio? (Y/N)	Y	Y	Y
Variable frame rate? (Y/N)	Y	Y	Y
4:2:0 Chrominance? (Y/N)	Y	Y	Y
4:2:2 Chrominance? (Y/N)	N	Y	N
4:4:4 Chrominance? (Y/N)	N	Y	N
Motion Comp. (Y/N)	Y	Y	Y
Transform coding (Y/N)	Y	Y	Y
Compression capability (subjective)	4	4	5
Scalable Bitrate? (Y/N)	Y	Y	Y
Embedded Scalability? (Y/N)	Y	Y	Y

Na tabela 2.3 podemos verificar a recomendação de codecs de áudio, e vídeo, para a maior parte das releases publicadas pelo 3GPP para as redes sem fio, em função de alguns serviços básicos de dados.

Tabela 2.3 Exemplos de arquivos 3gp – Release, Conteúdo [3GPP,2007].

Conformance	Suffix	Brand	Compatible brands	Example content
MMS and download: Files shall contain one or more of the brands 3gp4, 3gp5, 3gp6 and 3gp7. It is good practice to include compatible brands of earlier releases to enable legacy players to play the files.				
Release 4	.3gp	3gp4	3gp4	H.263 and AMR
Release 5, 4	.3gp	3gp5	3gp5, 3gp4, isom	H.263 and AMR
Release 6, 5, 4	.3gp	3gp6	3gp6, 3gp5, 3gp4, isom	H.263 and AMR
Release 7, 6, 5, 4	.3gp	3gp7	3gp7, 3gp6, 3gp5, 3gp4, isom	H.263 and AMR
Release 6, 5, 4	.3gp	3gp6	3gp6, 3gp5, 3gp4, isom	H.263, AMR and Timed text
Release 6, 5	.3gp	3gp6	3gp6, 3gp5, isom	Timed text
Release 6	.3gp	3gp6	3gp6, isom	H.264 (AVC) and AMR
Release 6	.3gp	3gp6	3gp6, isom	fragmented H.263 and AMR
Release 7	.3gp	3gp7	3gp7, isom	DIMS and AMR
Progressive download and MMS				
Release 6, 5, 4	.3gp	3gr6	3gr6, 3gp6, 3gp5, 3gp4, isom	H.263
Release 6, 5, 4	.3gp	3gr6	3gr6, 3gp6, 3gp5, 3gp4, isom	interleaved H.263 and AMR
Release 6	.3gp	3gr6	3gr6, 3gp6, isom	fragmented and interleaved H.263 and AMR
Release 6	.3gp	3gr6	3gr6, 3gp6, avc1	interleaved H.264 (AVC) and AMR
Streaming servers: Some files may in principle also be used for MMS or download.				
Release 6	.3gp	3gs6	3gs6, isom	AMR and hint track
Release 6	.3gp	3gs6	3gs6, isom	2 tracks H.263 and 2 hint tracks
Release 6, 5, 4	.3gp	3gs6	3gs6, 3gp6, 3gp5, 3gp4, isom	H.263, AMR and hint tracks
Extended presentations:				
Release 7, 6	.3gp	3ge7	3ge7, 3ge6, iso2	SMIL, AMR and JPEG images
Release 7	.3gp	3ge7	3ge7, iso2	DIMS, AMR, H.264 (AVC) and JPEG images
General purpose: Files that are not yet suitable for MMS, download or PSS streaming servers.				
Release 6	.3gp	3gg6	3gg6, isom	4 tracks H.263 (and no hint tracks)
Release 6	.3gp	3gg6	3gg6, isom	2 tracks H.263, 3 tracks AMR
3GP file, also conforming to MP4				
Release 4, 5 and MP4	.3gp	3gp5	3gp5, 3gp4, mp42, isom	MPEG-4 video
MP4 file, also conforming to 3GP				
Release 5 and MP4	.mp4	mp42	mp42, 3gp5, isom	MPEG-4 video and AAC

2.13.1 CODEC HMPROD

HMPROD é um CODEC e gerador de streaming RTP, desenvolvido pela Helix, para ser usado em linha de comando do Windows, ele recebe arquivos ou streamings em sua entrada e os converte conforme configurado em seu arquivo de configuração (.xml), que é passado como argumento durante a inicialização. O HMPROD é compatível com os Codecs de Vídeo MPEG-4 SP (Simple Profile) e H263, e com os Codecs de Audio AMR-NB, MP3 e AAC, e gera streamings RTP/RTSP para ser enviada a servidores de streaming, o objetivo deste codec, é criar streamings compatíveis com o padrão 3gp (3GPP), para gerar conteúdo em servidores de *Mobile Streaming* de redes móveis.

Este codec não é distribuído pelo fabricante, mas pode ser encontrado no pacote Helix Mobile Producer que possui uma versão gratuita com limitação para geração de conteúdo.

Na figura 2.13 é mostrado um exemplo de utilização do codec HMPROD, através da linha de comandos do Windows, onde ele apresenta uma versão resumida dos argumentos que podem ser utilizados com o codec para configurá-lo.

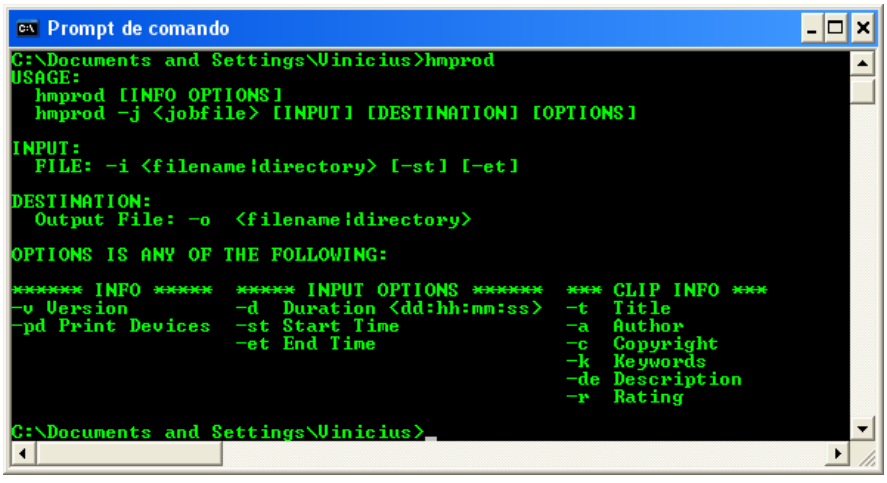


Figura 2.13: Codec HMPROD – Linha de Comando.

Na figura 2.14, é mostrado um arquivo de configuração em formato XML, usado para configurar o HMPROD, este arquivo é passado como argumento de linha de comando para Codec, e nele estão os dados de configuração do codec e da streaming gerada pelo HMPROD, como largura de banda e IP e porta UDP do servidor de streaming

```
<job build="547" exportType="3gppv5" version="11.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <clipInformation>
        <title />                       //Metadados não usados
        <author />                      //Metadados não usados
        <copyright />                   //Metadados não usados
        <mediaKeywords />               //Metadados não usados
        <description />                 //Metadados não usados
        <mediaRating>MPAA:G</mediaRating>  //Motion Picture Association of America:G = Geral (todas idades)
    </clipInformation>
    <inputs>
        <input xsi:type="captureInput">
        <audioDeviceID>0</audioDeviceID>  //Dispositivo de audio padrão do PC
        <videoDeviceID>0</videoDeviceID>  //Dispositivo de video padrão do PC
        <videoDeviceWidth>320</videoDeviceWidth>  //Largura do Video Capturado
        <videoDeviceHeight>240</videoDeviceHeight> //Altura do Video Capturado
        <prefilters>
            <audioPreFilters />
            <videoPreFilters>
                <enableResize>true</enableResize> //Habilita codec a Modifiar altura e largura do Video
                <resizeWidth>176</resizeWidth>    //Nova Largura do Video
                <resizeHeight>144</resizeHeight>  //Nova Altura do Video
            </videoPreFilters>
        </prefilters>
        </input>
    </inputs>
    <outputs>
        <broadcastOutput xsi:type="rtp">                                    //Tipo de envio de pacotes
        <sdpFileName>C:\Program Files\VigiaMovel-50K.sdp</sdpFileName>      //Nome e Path do Arquivo SDP
        <destinationAddress>127.0.0.1</destinationAddress>                  //Destino dos pacotes RTP (servidor)
        <destinationPort>50000</destinationPort>                            //Porta de destino no servidor
        <TTL>3</TTL>                                                        //Time to Live dos Pacotes
        </broadcastOutput>
    </outputs>
    <encodingParameters>                            // Controle da taxa de largura de banda da Streaming
        <rateControlMode>cbr</rateControlMode>      //CBR = Constant Bit Rate
        <numberOfPass>2</numberOfPass>              //Valido apenas para Codificão de arquivos para "live" é ignorado
        <encodeVideo>true</encodeVideo>             //Habilita codificação de Video
        <encodeAudio>true</encodeAudio>             //Habilita Codificação de Audio
        <exportSettings>
            <progressiveDownload>false</progressiveDownload> //Profile de download progressivo desativado
            <hinted>true</hinted>                            //Incluir duração habilitado
            <MTUSize>1400</MTUSize>                          //Max Transfer Unit - Só vale para MPEG-4
        </exportSettings>
        <audiences>
            <audience>
                <version>11.0</version>    //Versão do Codec
                <audienceName>50k EDGE Voice</audienceName> //Nome do Perfil Criado
                <audienceAvgBitRate>50000</audienceAvgBitRate> //Largura de Streaming Desejada
                <information />
                <audienceMaxBitRate>60000</audienceMaxBitRate> //Só vale se VBR Habilitado - Não está
                <videoEncoder xsi:type="mpeg4sp">              //Tipo do Codec MPEG-4 Standard Profile
                    <level>Automatic</level>     //Nivel do Codec - "AutoMatic" - ele se ajusta
                    <keyFramePeriodInMs>5000</keyFramePeriodInMs> //Tempo Máximo entre quadros - ms
                    <maxFrameRate>10.000000</maxFrameRate>        //Quadros por segundo (de 0,5 a 30)
                    <encodingComplexity>high</encodingComplexity> //Melhor Qualidade - Mais Lento
                    <videoMode>normal</videoMode>   //Padrão - Faz um balanço entre Frame Rate e Sharpness
                </videoEncoder>
                <audioEncoder xsi:type="amrnb">   //Codec de Audio AMR-Narrow Band
                    <bitRate>12200</bitRate> //Taxa de Bits para a Codificação de Audio
                    <useDTX>false</useDTX>   //Dedtecção de Silencio - Se houver Silencio não Codifica
                </audioEncoder>
            </audience>
        </audiences>
    </encodingParameters>
</job>
```

Figura 2.14: HMPROD – Arquivo .xml para Confiduração do Codec.

Grande parte das tecnologias mostradas neste capítulo serão utilizadas no desenvolvimento do protótipo deste projeto, no próximo capítulo serão apresentados detalhes da aplicação destas tecnologias no protótipo, demonstrando deste modo quais foram aplicadas ao projeto.

Capítulo 3 – Arquitetura e Implementação do Sistema de Monitoração On-Line

3.1 INTRODUÇÃO

A utilização de streaming de vídeo em sistemas de vigilância, e a facilidade em visualizar esta streaming em dispositivos portáteis, usando redes móveis, é, até certo ponto, pouco explorada pelos sistemas de segurança, encontrados no mercado, mas aos poucos começam a surgir novidades com tecnologias semelhantes.

Nas próximas seções serão apresentadas as especificações que compõem os módulos do protótipo proposto para este sistema de vigilância, bem como a integração entre as varias tecnologias apresentadas.

Este projeto procura desenvolver de forma prática, um protótipo de sistema de vigilância, que permita testar e validar as diversas facilidades que podem ser incorporadas nos sistemas residenciais e industriais, utilizando streaming de vídeo e redes móveis.

De maneira simplificada o funcionamento do protótipo proposto é o seguinte: Um software rodando em um computador com sistema operacional Windows XP e conectado a uma WebCam USB, monitora o movimento nas imagens que são geradas pela webcam. Este software controla um CODEC, que tem a função de converter estas imagens em formatos compatíveis com redes moveis (3gp, mp4), e gera uma streaming RTP/RTSP que é enviada para um servidor de streaming localizado no mesmo computador, mas pode ser utilizado opcionalmente um servidor externo, localizado na internet ou na própria rede LAN do computador que gera as imagens.

Na figura 3.1 é mostrado de maneira esquemática o funcionamento do sistema.

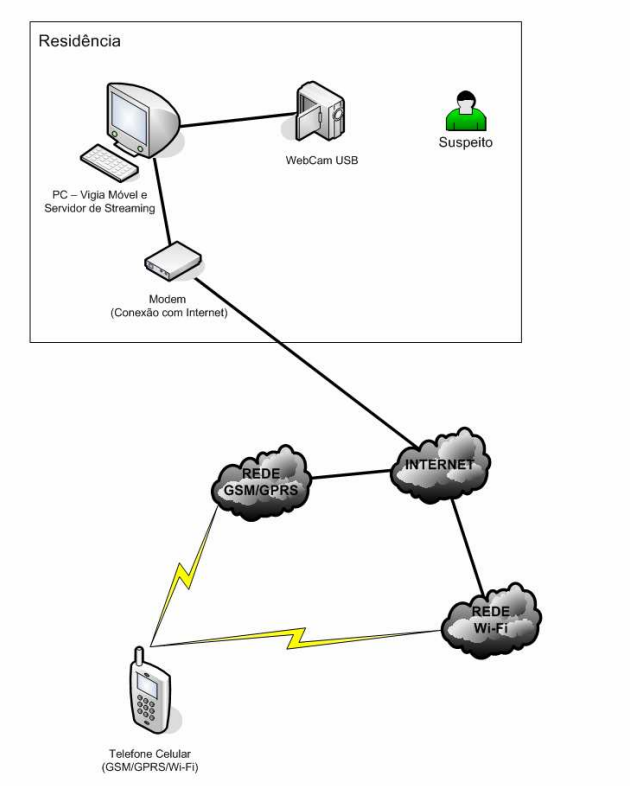


Figura 3.1 – Diagrama esquemático do projeto

Quando acontece um movimento na imagem monitorada pelo software, ele dispara o envio de uma mensagem SMS para um determinado telefone móvel pré-estabelecido, avisando que foi detectado algum movimento na imagem. Se o movimento continuar o sistema irá enviar outra mensagem, 3 minutos após o envio da mensagem anterior, e assim por diante.

Ao receber esta mensagem o dispositivo móvel poderá acionar uma aplicação que tem o objetivo de direcionar o tocador de streaming do dispositivo, para a streaming que está sendo gerada pelo servidor. Com isto o móvel poderá se conectar via GSM/GPRS, WCDMA/HSDPA, ou Wi-Fi (depende de configuração do dispositivo), ao servidor e visualizar na tela do aparelho as imagens que estão sendo geradas pela Webcam conectada ao Computador. O software que controla a Webcam também irá armazenar as imagens dos movimentos detectados, em um arquivo mpeg em seu próprio disco para posterior verificação.

Para o desenvolvimento do projeto o sistema foi dividido em duas partes:

- **Vigia Móvel - Central de Vigilância:** Software que roda no PC.
  - Captura de Imagens.
  - Controla a detecção de movimentos.
  - Controla o envio de SMS para o móvel
  - Controla o codec que gera a streaming em formato 3gp ou mp4
  - Armazena os movimentos detectados em um arquivo no disco

- **Vigia Móvel – Vigilante:** Software que roda no dispositivo móvel.
  - Direciona o tocador de streaming do móvel para o servidor de streaming conectado a Central de vigilância.

3.2 CENTRAL DE VIGILÂNCIA

A central de vigilância é um software desenvolvido em C++, utilizando como plataforma de desenvolvimento o Compilador *Borland Developer Studio 2006*. Para chegar até o protótipo final, foram utilizados vários componentes disponíveis no Borland, que viabilizam o desenvolvimento rápido de aplicações multimídia em plataformas Windows, utilizando os recursos da API DirectShow da Microsoft.

A figura 3.2 mostra a estrutura do software dentro do contexto do projeto, as partes assinaladas em verde foram desenvolvidas utilizando os recursos do Borland C++.

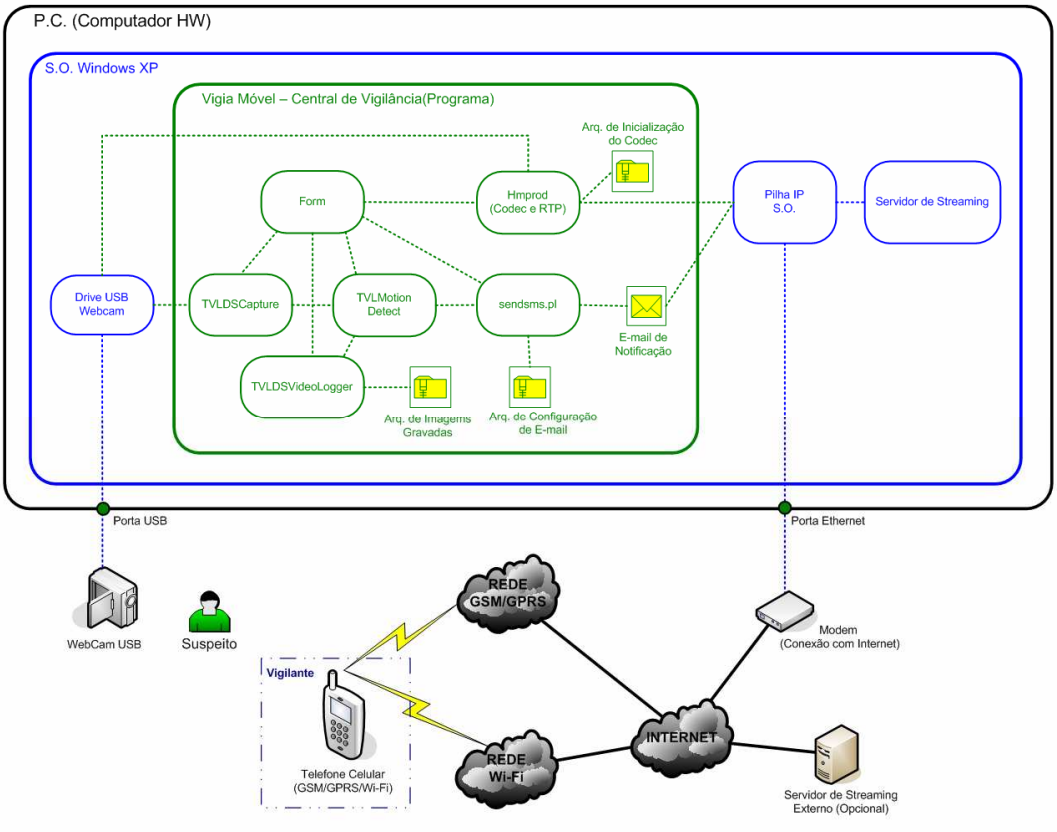


Figura 3.2 – Estrutura do sistema Vigia Móvel - Central de Vigilância

A Central de vigilância é composta basicamente de seis componentes:

- Form : Monitora os eventos de interação entre os componentes
- TVLDSCapture: Componente da Biblioteca Vision Lab que captura as imagens da Webcam.
- TVLMotionDetect: Componente da biblioteca Vision Lab que Analiza a imagem recebida por TVLDSCapture, em busca de movimentos.
- TVLDSVideoLogger: Componente da Biblioteca Vision Lab que grava as imagens de movimentos detectados TVLMotionDetect
- Sendsms.pl: Componente dabiblioteca Indy, que implementa um cliente de e-mails.
- HMPROD: Codec 3gp e gerador de Streaming RTP/RTSP para enviar imagems ao servidor DSS (Darwing Streaming Server)

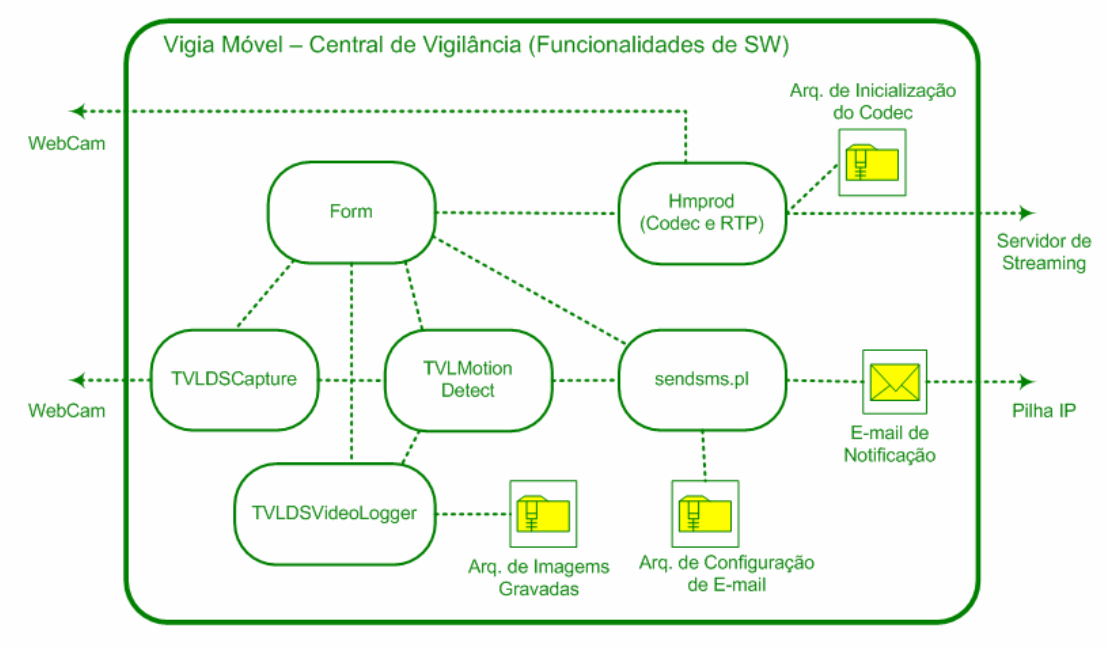


Figura 3.3 – Blocos do Software - Central de Vigilância

3.2.1 CAPTURA DE IMAGENS

A captura de imagens da Webcam é feita utilizando um componente da biblioteca do Builder C++, chamado TVLDSCapture. Este componente utiliza funcionalidades da API DirectShow da Microsoft para capturar imagens e enviar a outros componentes, a API DirectShow utiliza o conceito de filtros, sendo que cada componente do Borland Builder funciona como um Filtro DirectShow para o tratamento de informações multimídia.

O Componente TVLDSCapture possui suas propriedades, métodos e evento. Na figura 3.4 podemos ver o componente TVLDSCapture inserido no projeto Vigia Móvel e a lista de suas propriedades.

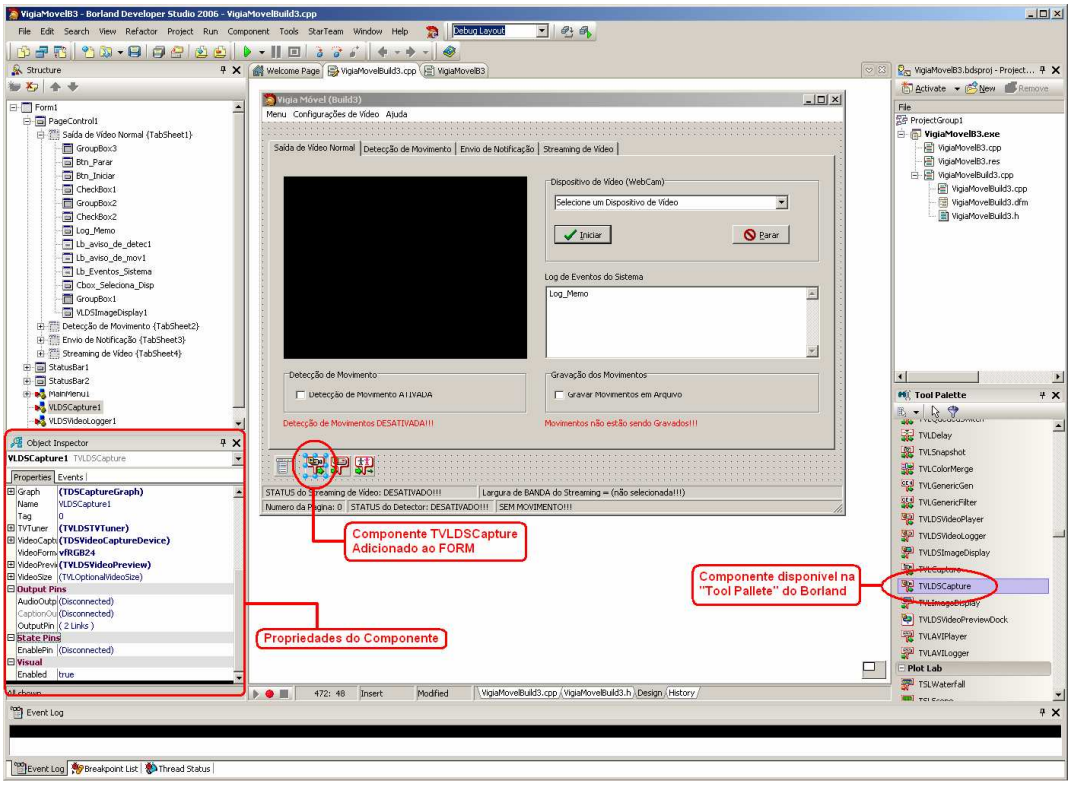
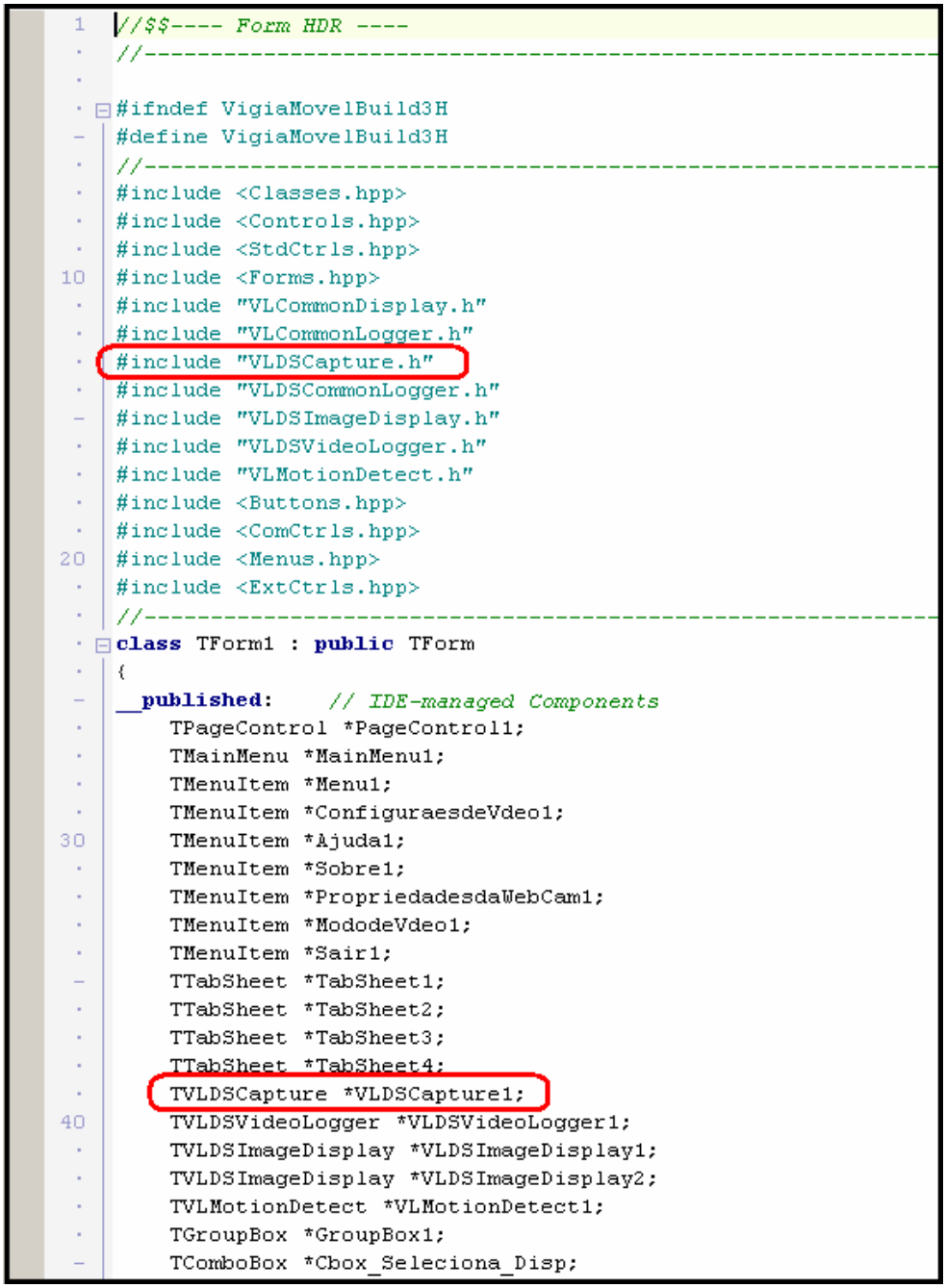


Figura 3.4 – Componente TVLDSCapture - Central de Vigilância.

Ao se inserir um componente no projeto, automaticamente é criado um "link" no código C++ (arquivo .h) para o arquivo que contém as definições da classe do componente TVLDSCapture (arquivo .h ou .hpp), e também uma função para a definição de uma instância do componente na memória do sistema (ponteiro) é inserido no código, na figura 3.5 podemos visualizar estas definições no arquivo VigiaMovel.h.



```
//$$---- Form HDR ----
//---------------------------------------------------------------------------

#ifndef VigiaMovelBuild3H
#define VigiaMovelBuild3H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "VLCommonDisplay.h"
#include "VLCommonLogger.h"
#include "VLDSCapture.h"
#include "VLDSCommonLogger.h"
#include "VLDSImageDisplay.h"
#include "VLDSVideoLogger.h"
#include "VLMotionDetect.h"
#include <Buttons.hpp>
#include <ComCtrls.hpp>
#include <Menus.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:    // IDE-managed Components
    TPageControl *PageControl1;
    TMainMenu *MainMenu1;
    TMenuItem *Menu1;
    TMenuItem *ConfiguraesdeVdeo1;
    TMenuItem *Ajuda1;
    TMenuItem *Sobre1;
    TMenuItem *PropriedadesdaWebCam1;
    TMenuItem *MododeVdeo1;
    TMenuItem *Sair1;
    TTabSheet *TabSheet1;
    TTabSheet *TabSheet2;
    TTabSheet *TabSheet3;
    TTabSheet *TabSheet4;
    TVLDSCapture *VLDSCapture1;
    TVLDSVideoLogger *VLDSVideoLogger1;
    TVLDSImageDisplay *VLDSImageDisplay1;
    TVLDSImageDisplay *VLDSImageDisplay2;
    TVLMotionDetect *VLMotionDetect1;
    TGroupBox *GroupBox1;
    TComboBox *Cbox_Seleciona_Disp;
```

Figura 3.5 – Definições do componente TVLDSCapture – arquivo: “VigiaMovel.h”.

Para que o componente TVLDSCapture possa capturar imagens de um dispositivo, é necessário passar o nome do dispositivo, no formato ASCII, como argumento da propriedade “DeviceName”.

Para consultar os dispositivos de vídeo disponíveis no computador o componente oferece a função “GetDeviceList”, que retorna uma lista do nome, em formato ASCII, dos dispositivos como webcam e placas de captura de vídeo.

A figura 3.6 mostra os fragmentos de código usados para obter a lista de dispositivos de vídeo do computador, e o código usado para selecionar na lista, o nome escolhido para ser o dispositivo de vídeo padrão do componente TVLDSCapture.

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FormShow(TObject *Sender)
{

     VLDSCapture1->VideoCaptureDevice->GetDeviceList(Lista1);

     Cbox_Seleciona_Disp->Items->AddStrings(Lista1);
  }
//---------------------------------------------------------------------------
void __fastcall TForm1::Cbox_Seleciona_DispChange(TObject *Sender)
{
  nomeDisp=Cbox_Seleciona_Disp->Items->Strings[Cbox_Seleciona_Disp->ItemIndex];
  VLDSCapture1->VideoCaptureDevice->DeviceName = nomeDisp;
  VLDSCapture1->Open();
  VLDSCapture1->VideoSize->Width = 320;
  VLDSCapture1->VideoSize->Height = 240;
}

//---------------------------------------------------------------------------
```

Figura 3.6 – Obtenção e definição do dispositivo de vídeo utilizado pelo componente.

Na figura 3.6 também é possível visualizar as definições de largura e altura, definidos para a captura de imagens, o valor de 320x240 foi escolhido em função das especificações da maioria das webcams USB que oferecem melhor desempenho nessa dimensão, mesmo sabendo-se que a maioria pode trabalhar com valores maiores. O Valor de 320x240 também se adaptou melhor na formatação da interface do aplicativo.

A função “Open”, apresentada no código, tem o objetivo de preparar o dispositivo para a captura de imagens, ou seja, já conecta o componente ao “driver” do dispositivo.

Para iniciar a captura de imagens, primeiramente é definido o ponto de entrega das imagens capturadas pelo componente, segundo a estrutura da aplicação, as imagens capturadas, devem ser encaminhadas a um componente de detecção de movimentos, e a um componente de visualização de imagens.

Na figura 3.7 é mostrado o código usado para iniciar a captura de imagens pela webcam, este código faz parte da definição de um evento “On Click” de um Botão, ou seja, quando o botão INICIAR é pressionado a função *VLDSCapture1->Start();*, é chamada e a captura de imagens se inicia.

```
220  //---------------------------------------------------------------------------

     void __fastcall TForm1::Btn_IniciarClick(TObject *Sender)
     {
       VLDSVideoLogger1->Enabled = True;

       VLDSCapture1->Start();


     }
     //---------------------------------------------------------------------------
```

Figura 3.7 – Função para iniciar captura de vídeo, no evento OnClick do Botão Iniciar.

Os pontos de conexão do componente TVLDSCapture com outros componentes podem ser definidos através da propriedade “Output Pin”. Esta propriedade pode ser definida utilizando o ambiente gráfico do Borland Builder, efetuando um Click no painel da propriedade.

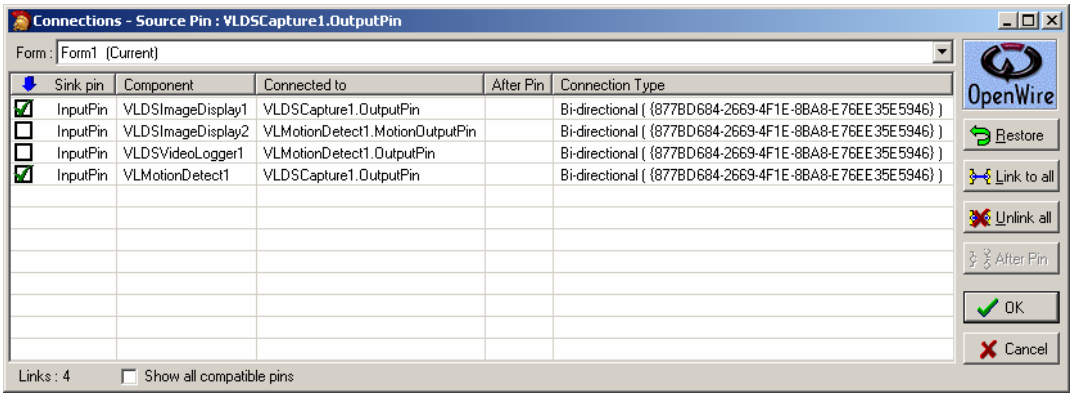


Figura 3.8 – Configuração do ponto de saída da imagem para o componente TVLDSCapture.

Conforme mostra a figura 3.8, o ponto de saída do componente TVLDSCapture foi conectado ao ponto de entrada do componente TVLDSImageDisplay, que tem a função de mostrar a imagem capturada pela webcam, e ao componente TVLMotionDetect, que tem a função de analisar a presença de movimentos na imagem.

Na figura 3.9 é possível visualizar a localização gráfica de todos os componentes que interagem com o componente TVLDSCapture.

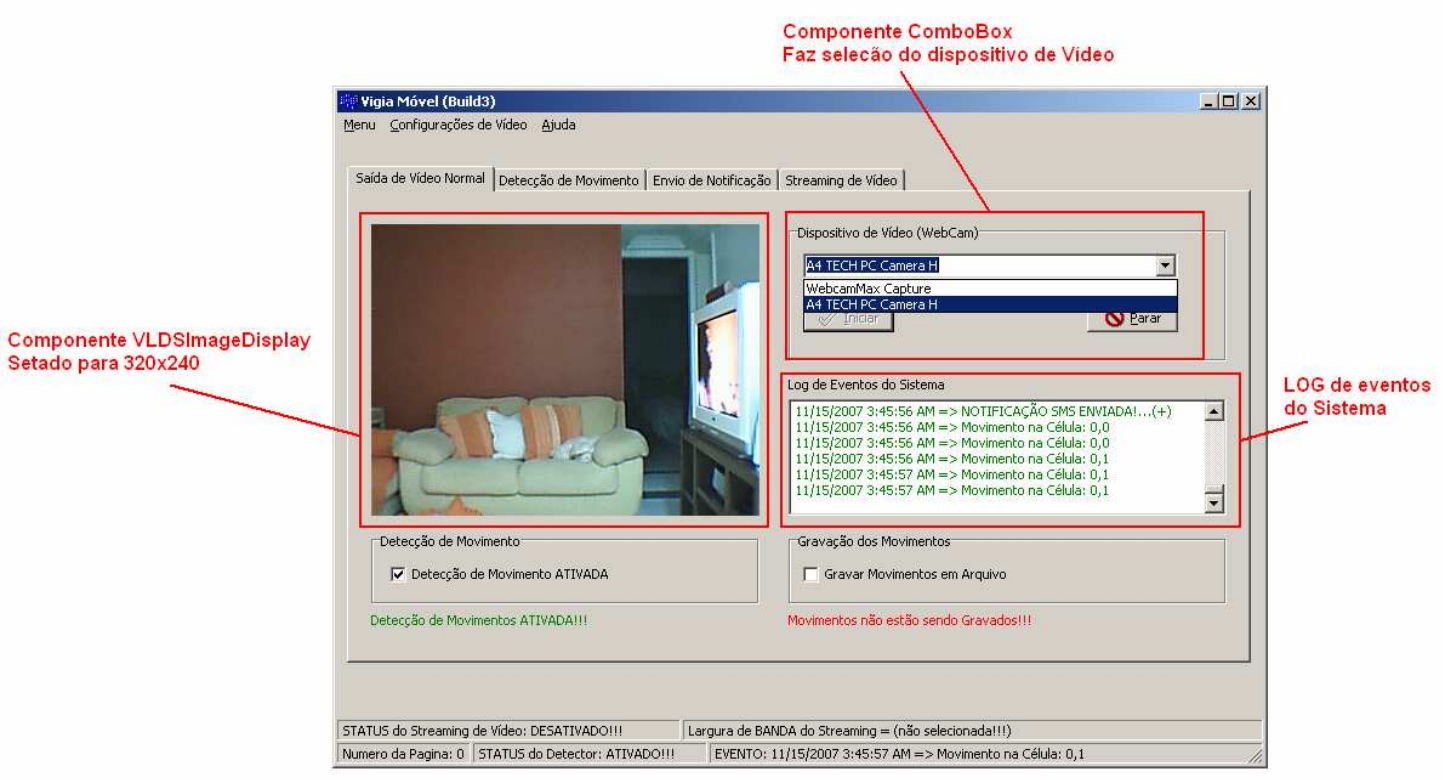


Figura 3.9 – Localização dos componentes que interagem com TVLDSCapture

3.2.2 DETECÇÃO DE MOVIMENTOS

Para realizar a detecção dos movimentos nas imagens capturadas pela webcam foi utilizado o componente TVLMotionDetect, da Biblioteca *VisionLab* do Borland Builder, este componente utiliza o método da “Diferença entre 2 quadros sucessivos”, para verificação dos movimentos [Mitov,2007]

Neste método, verifica-se para cada quadro a variação de Pixels entre duas imagens consecutivas.

No caso da variação dos pixels exceder a percentagem de movimento definida pelo usuário, é detectado uma intrusão, e o Evento “OnMotionDetected” é acionado pelo componente.

O componente TVLMotionDetect recebe em sua entrada as imagens capturadas pelo componente TVLDSCapture, após analisadas pelo algoritmo as imagens são enviadas para duas saídas diferentes. Para a saída “OutputPin”, é enviado o último quadro com movimentos detectados, desta forma ele recebe sempre a última imagem com movimento, já a saída “MotionOutputPin”, envia todos os quadros de entrada mostrando apenas os pixels diferentes entre cada quadro subseqüente, em tonalidade branca.

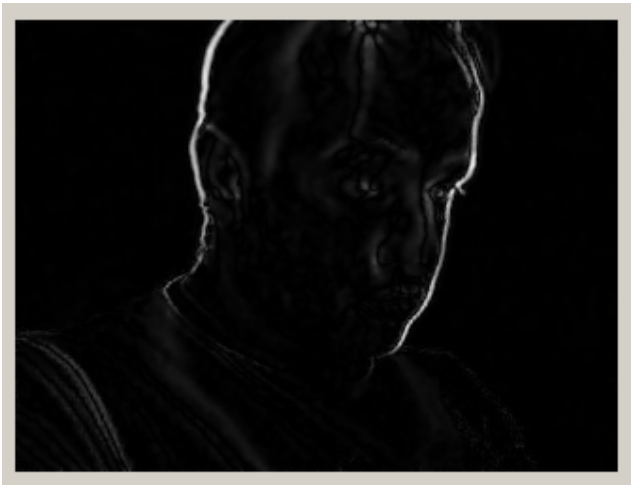


Figura 3.10 – Imagem da diferença entre quadros subseqüentes: apenas os pixels com diferença de tonalidade entre os quadros são apresentados na tonalidade branca.

O componente TVLMotionDetect utiliza o algoritmo chamado de “Método da diferença entre 2 quadros sucessivos”, para fazer a avaliação de presença de movimentos nas imagens [Mitov,2007].

Neste método, calcula-se para cada quadro a diferença entre este e o anterior, o qual é tomado como referência. Assim, se uma imagem for expressa como apresentado em (1).

$$I = f(x, y, t) \qquad (1)$$

Onde: I é um valor representando a intensidade luminosa no ponto de coordenadas (x,y) no instante t, tem-se que, como mostrado em (2),

$$\Delta I = \left| f(x, y, t_2) - f(x, y, t_1) \right| \qquad (2)$$

Onde: ΔI é o módulo da diferença entre a imagem no instante t_2 (imagem atual) e a imagem no instante t_1 (imagem anterior).

Em um ambiente controlado, com muito pouca variação, a diferença entre os dois quadros sucessivos terá valores iguais ou muito próximos a zero nas regiões onde existe pouca ou nenhuma variação na intensidade luminosa (ou seja, no cenário de fundo que permaneceu estático), e valores diferentes de zero nas regiões onde objetos estão em movimento. Pode-se considerar que variações, dentro de certa faixa de tamanho (extensão no plano) e proporções, correspondam a uma pessoa em movimento e, fora desta faixa, simples objetos.

Para este projeto a saída “OutputPin” foi conectada a um componente TVLDSVideoLogger que efetua a função de gravar as imagens em arquivos, e a saída “MotionOutputPin”, foi conectada em um componente TVLImageDisplay (Fig. 3.11), para possibilitar a visualização da diferença entre cada quadro pelo usuário do sistema.

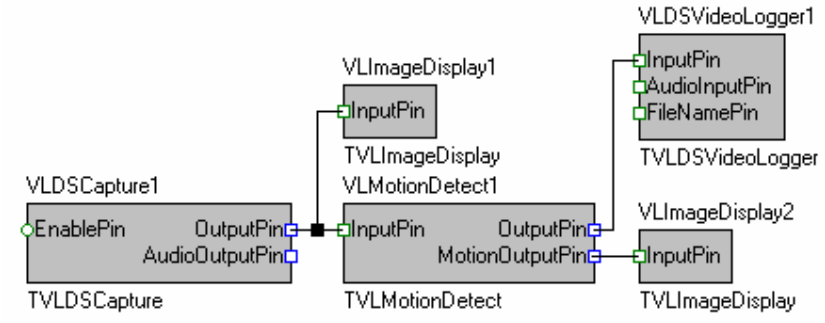


Figura 3.11 – Conexões entre o componente MotionDetect, e os demais componentes.

Na figura 3.11 é mostrado a configuração de portas de entrada e saída para a imagem entre os componentes do sistema, na figura 3.12 podemos ver também como ficam as tabela de conexões do componente TVLMotionDetect.

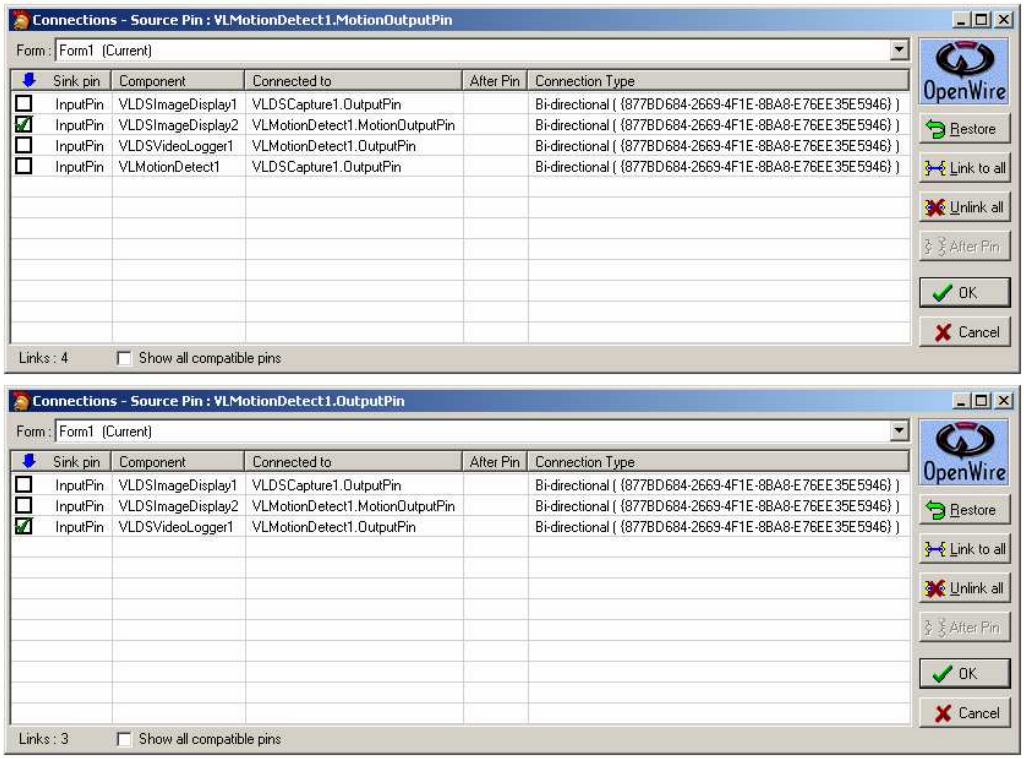


Figura 3.12 – Tabela de Conexões para MotionOutputPin e OutputPin.

O Componente TVLMotionDetect, possui a propriedade “MotionGrid” que permite determinar um número de células no qual uma imagem pode ser dividida (no máximo 20x20), e deste maneira podemos determinar diferentes sensibilidades para cada célula ou região da imagem. Desta forma pode-se configurar uma sensibilidade menor para uma região da imagem onde possa haver um movimento aleatório, como uma cortina, um vaso de flores ou um ventilador, e também uma sensibilidade maior em uma região estratégica como uma porta, ou uma janela.

Para este projeto foi determinado que o “Motion Grid” tenha uma dimensão de 2x2, ou seja, a análise de sensibilidade a movimentos, pode ser diferente para cada quadrante da imagem.

Os Níveis de ajuste de sensibilidade podem variar de “0” a “9”:

- “0” = detecção desligada;
- “9” = sensibilidade máxima (Dispara evento OnMotionDetect).

Na inicialização do programa o evento “OnShow” do “Form” principal é chamado. Dentro da definição deste evento, foi inserido as propriedades que definem a configuração inicial do “MotionGrid”.

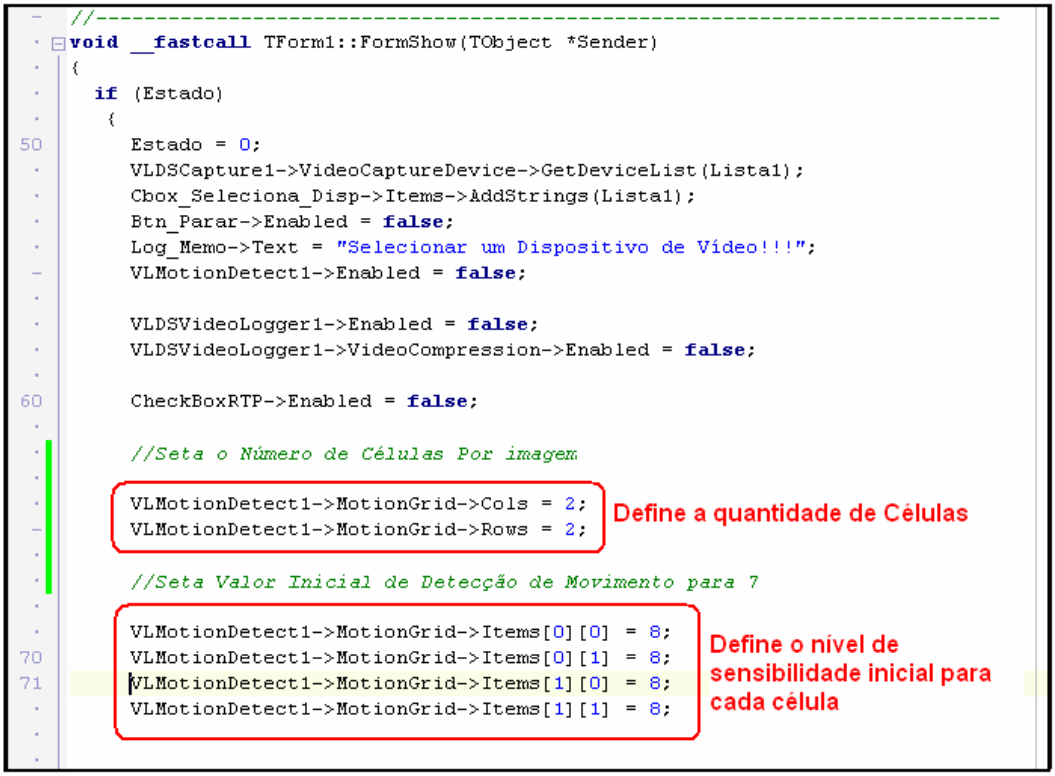


Figura 3.13 – Definição inicial do “MotionGrid”, no evento OnShow do Form1

A figura 3.13 mostra os métodos que definem as propriedades iniciais do componente “TVLMotionGrid” para detecção de movimentos em até quatro células da imagem.

É possível ver as funções que definem o número de Linhas (Rows) e colunas (Cols), e as que definem a sensibilidade inicial de cada célula (Nível 8).

Para tornar o sistema mais flexível, o ajuste de sensibilidade das células pode ser feito a qualquer momento em tempo de execução do programa utilizando um componente botão (Atualizar) que faz uma chamada aos métodos “MotionGrid->Items[x][y]”, onde “x” é a posição da linha, e “y” e a posição da coluna, que recebe como argumento um inteiro entre “0” e “9” , a figura 3.14 mostra esta funcionalidade

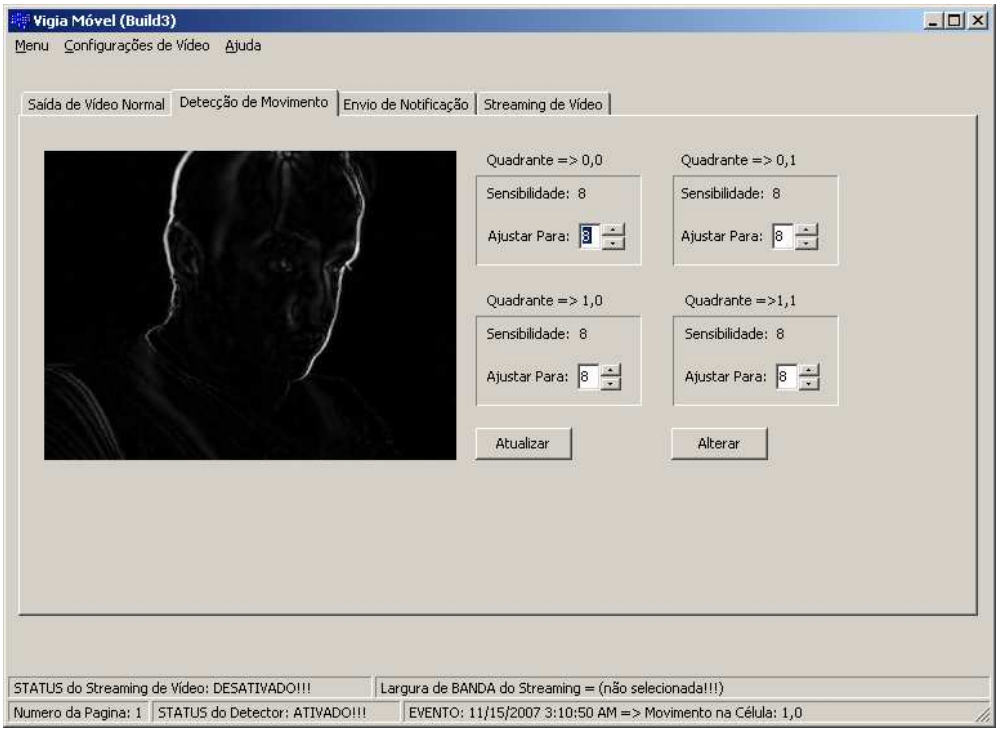


Figura 3.14 – Ajuste de sensibilidade da detecção de movimento, dividido em 4 áreas.

Na figura 3.15 é mostrado o código usado para implementar o ajuste de sensibilidade.

```
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Btn_Atualiza_SensClick(TObject *Sender)
{
   LCell00_Valor->Caption = VLMotionDetect1->MotionGrid->Items[0][0];
   LCell01_Valor->Caption = VLMotionDetect1->MotionGrid->Items[0][1];
   LCell10_Valor->Caption = VLMotionDetect1->MotionGrid->Items[1][0];
   LCell11_Valor->Caption = VLMotionDetect1->MotionGrid->Items[1][1];
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
   int ce00, ce01, ce10, ce11;

   //Converte String do EditCell para Inteiro a ser aplicado no MotionGrid
   ce00 = StrToInt(EditCell00->Text[1]);
   ce01 = StrToInt(EditCell01->Text[1]);
   ce10 = StrToInt(EditCell10->Text[1]);
   ce11 = StrToInt(EditCell11->Text[1]);
   VLMotionDetect1->MotionGrid->Items[0][0] = ce00;
   VLMotionDetect1->MotionGrid->Items[0][1] = ce01;
   VLMotionDetect1->MotionGrid->Items[1][0] = ce10;
   VLMotionDetect1->MotionGrid->Items[1][1] = ce11;

   //Atualiza Valores de Sensibilidade na Janela
   LCell00_Valor->Caption = VLMotionDetect1->MotionGrid->Items[0][0];
   LCell01_Valor->Caption = VLMotionDetect1->MotionGrid->Items[0][1];
   LCell10_Valor->Caption = VLMotionDetect1->MotionGrid->Items[1][0];
   LCell11_Valor->Caption = VLMotionDetect1->MotionGrid->Items[1][1];

   }
//---------------------------------------------------------------------------
```

Figura 3.15 – Código usado para ajuste de sensibilidade da detecção de movimento

No código da figura 3.15 é mostrada a função para o evento “OnClick” do botão “Atualizar”, que verifica os valores configurados para sensibilidade de cada célula nas propriedades *“MotionGrid->Items[x][y]”*, também é mostrada a função que converte a String digitada pelo usuário, em Inteiro para ser inserido como novo valor de sensibilidade da célula, onde são permitidos valores entre “0” e “9” somente, para inserir o novo valor também é usado o método *“MotionGrid->Items[x][y]”*.

O Componente TVLMotionDetect, possue uma propriedade que habilita ou não o componente a executar a detecção de movimentos, esta propriedade é uma variável Booleana, quando seu valor for FALSE, indica ao componente que ele não vai verificar a existência de movimentos nas imagens entregues

em sua entrada, quando o valor for TRUE, o componente deve verificar a existência de movimentos.

Esta propriedade foi utilizada pelo sistema para adicionar controle à detecção de movimentos, ou seja, desta forma o usuário pode “LIGAR” ou “DESLIGAR” a detecção conforme sua necessidade.

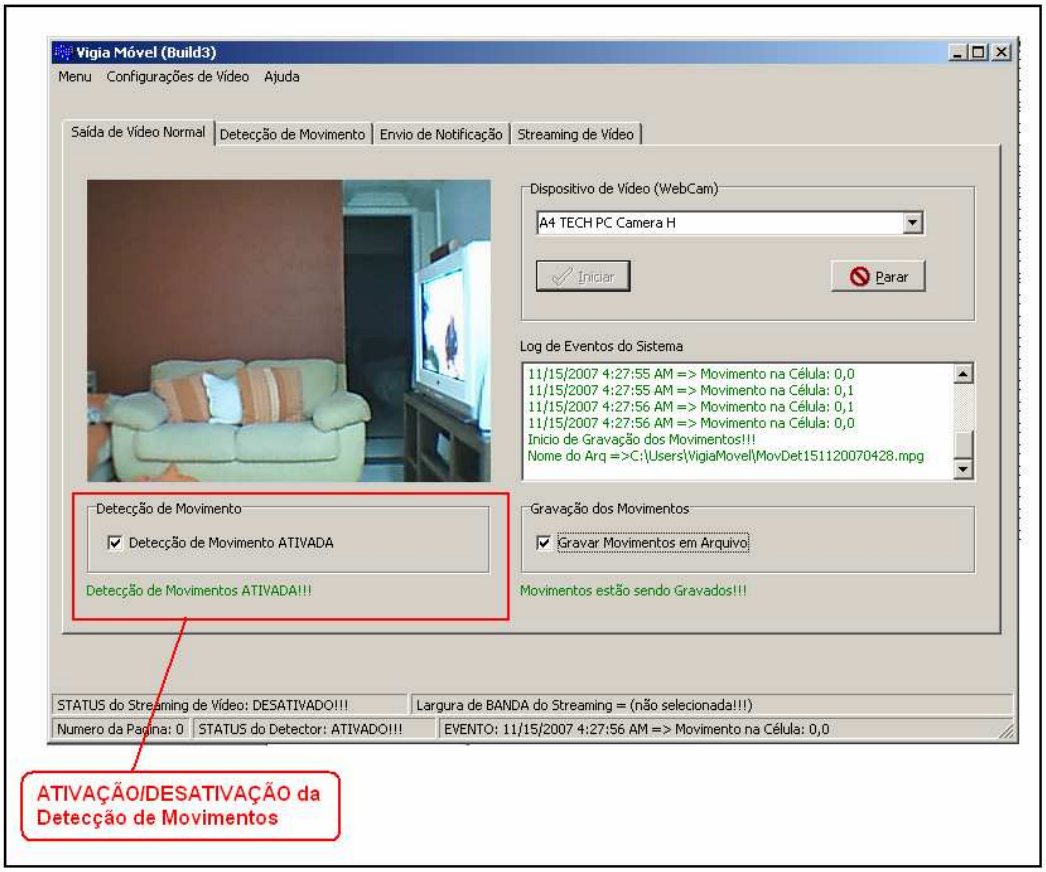


Figura 3.16 – CheckBox usado para Ativar/Desativar a detecção de movimentos.

Na figura 3.16 é mostrado o código usado para implementar o “CheckBox” que Ativa ou Desativa o detector de movimentos.

```
//---------------------------------------------------------------------------

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  if (CheckBox1->Checked)
      {
        VLMotionDetect1->Enabled = true;
        StatusBar1->Panels->Items[1]->Text = " STATUS do Detector: ATIVADO!!!";
        Log_Memo->Lines->Add(AnsiString ("Detector de Movimentos ATIVADO!!!" ));
        Lb_aviso_de_detec1->Font->Color = clGreen;
        Lb_aviso_de_detec1->Caption = "Detecção de Movimentos ATIVADA!!!";
      }
  else if (!CheckBox1->Checked)
      {
        VLMotionDetect1->Enabled = false;
        VLDSImageDisplay2->Clear();
        StatusBar1->Panels->Items[1]->Text = " STATUS do Detector: DESATIVADO!!!";
        StatusBar1->Panels->Items[2]->Text = " SEM MOVIMENTO!!!";
        Log_Memo->Lines->Add(AnsiString ("Detector de Movimentos DESATIVADO!!!" ));
        Lb_aviso_de_detec1->Font->Color = clRed;
        Lb_aviso_de_detec1->Caption = "Detecção de Movimentos DESATIVADA!!!";

      }
}
//---------------------------------------------------------------------------
```

Figura 3.17 – Código usado para Ativar/Desativar a detecção de movimentos.

O evento OnClick do CheckBox1, dispara a função “CheckBoxClick”, que avalia o status do CheckBox. Se ele estiver com a propriedade “Enabled” armazenada como TRUE ela habilita o detector de movimentos, se estiver FALSE ela desabilita o detector de movimentos, no código existem também funções usadas para enviar mensagens ao LOG do sistema informando o status do detector de movimentos.

3.2.3 ENVIO DE SMS E E-MAIL

A Central de vigilância tem a função de enviar mensagens de SMS e e-mail para destinos pré-estabelecidos, para aumentar a segurança e garantir que o usuário seja informado sobre os movimentos detectados, o sistema pode enviar mensagens de SMS para até dois telefones celulares GSM, e uma caixa postal de e-mail, a versão atual pode enviar SMS para três operadoras, CLARO, VIVO, e TIM, mas podem ser adicionadas outras operadoras.

O envio de SMS através de Internet para telefones celulares é semelhante para todas as operadoras, as Plataformas de SMSC (Shor Message Center), comunicam-se com a rede GSM através do protocolo SMPP (Short Message Peer To Peer), mas por motivos de segurança (Ataques, Spam) nenhuma operadora disponibiliza a Interface SMPP, em modo aberto na Internet. O procedimento utilizado é conectar as plataformas de SMSC a servidores de E-mail (SMTP) que possuem interfaces SMPP do lado interno da rede da operadora. Estes servidores de e-mail possuem uma conta para cada número de telefone celular, e desta forma, os e-mails destinados para cada número de telefone e são convertidos em SMS, ou em MMS (no caso de mensagens com anexo) , e encaminhados ao SMSC da operadora.

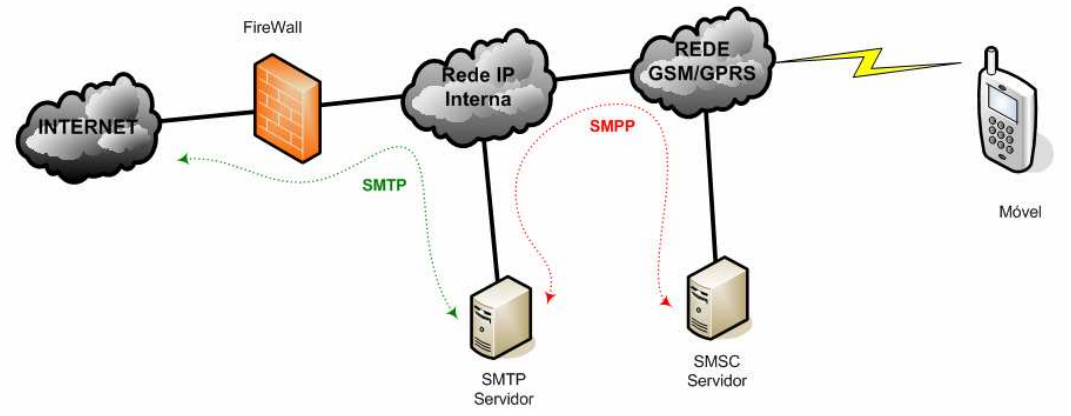


Figura 3.18 – Topologia para envio de SMS para celulares, a partir da Internet.

É possível também enviar SMS através de páginas Web, das operadoras, mas para automatizar este processo é necessário desenvolver uma interface XML mais complexa, com a ajuda de leitores OCR para códigos

CAPTCHA, (Completely Automated Public Turing test to tell Computers and Humans Apart), que é extremamente complexo e pesado, e os algoritmos atuais não conseguem garantir a decodificação das Imagens CAPTCHA [Wikipedia,2007].

Para este projeto foi usado o método de enviar mensagens via SMTP por um servidor de E-mail, que encaminha estas mensagens para o servidor de e-mails da operadora em questão, para este processo funcionar poderiam ser utilizados duas técnicas:

Implementar o servidor SMTP e o Cliente SMTP no sistema.

Implementar somente o cliente SMTP no sistema e utilizar qualquer servidor SMTP disponível na internet.

Com o objetivo de tornar o protótipo mais simples e funcional foi escolhido implementar apenas o cliente SMTP no sistema, pois desta forma, há mais flexibilidade para o sistema, e diminuição da complexidade do código.

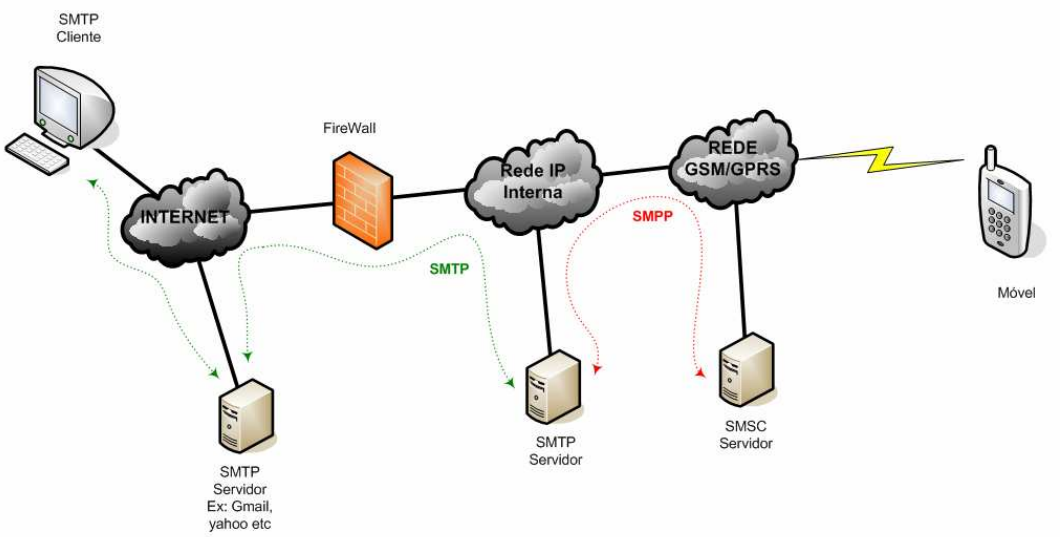


Figura 3.19 – Topologia para envio de SMS usada pela aplicação – Vigia Móvel.

Usando apenas o cliente SMTP, o sistema pode ser configurado com qualquer conta de e-mail SMTP. Para o protótipo foi utilizado um algoritmo que se conecta usando contas do Gmail, que é um serviço grátis e bastante confiável em relação à disponibilidade e entrega de mensagens. Futuramente será desenvolvida uma interface para se conectar a qualquer serviço SMTP.

Para o envio das mensagens de SMS e e-mail foi desenvolvido um programa em linguagem Perl, chamado *sendsms.pl,* localizado no diretório padrão da aplicação, que é chamado pelo programa principal em C++, toda vez que as mensagens devem ser enviadas.

A figura 3.20 abaixo, mostra o programa *sendsms.pl*, escrito em PERL para o envio de mensagens SMS usando o pacote “Webamail::Gmail” de PERL.

C:\USERS\VigiaMovel\sendsms.pl*

```
#!/usr/bin/perl

## Declarando Pacote Perl ###################
use Mail::Webmail::Gmail;
## Abrindo arquivo ##########################
open (SOURCE1 ,"/cygdrive/c/USERS/VigiaMovel/emaildados.txt");
## Carrendo arquivo em array na memória #####################
my @linha = <SOURCE1>;
## Fechando Arquivo ###########################################
close SOURCE1;
## Elimando caracteres de nova linha ##########################
chomp($linha[0]);
chomp($linha[1]);
chomp($linha[2]);
chomp($linha[3]);
chomp($linha[4]);
chomp($linha[5]);

##  Eliminando caracteres não usados ############################
my $usuario=substr($linha[0],5,-1);
my $senha=substr($linha[1],5,-1);
my $conta=substr($linha[2],6,-1);
my $movel1=substr($linha[3],3,-1);
my $movel2=substr($linha[4],3,-1);
my $email=substr($linha[5],3,-1);
## Alinhando destinos em uma lista ##############################
my $email_dest = [$movel1,$movel2,$email];


## Mensagem de Alerta ##

$mensagem = "!!! ALERTA !!! Foi detectado Movimento em Sua Webcam!!!Conecte-se ao Vigia Movel !!!";

## Login no Gmail ##

my $gmail = Mail::Webmail::Gmail->new(username => $usuario, password => $senha);

## Enviando as Notificacoes ##

$gmail->send_message( to => $email_dest , subject => 'Vigia Movel', msgbody => $mensagem );


## Verificando Erros ##

if ( $gmail->error() )
   {
      print $gmail->error_msg();
   }
################################################################################
############### ARQUIVO COM LOG DE MENSAGEMS ###################################
################################################################################

$data = `/bin/date +%d/%m/%y-%k:%M:%S`;
chomp($data);

open (SOURCE, ">>c:/Users/VigiaMovel/msglog.log");

print SOURCE "================================================================================\n";
print SOURCE ">>$data\n";
print SOURCE ">>Enviado SMS para $movel1 !!!\n";
print SOURCE ">>Enviado SMS para $movel2 !!!\n";
print SOURCE ">>Enviado E-mail para conta $email !!!\n";
print SOURCE ">>Mensagems Enviadas pela conta: $conta !!!\n";
print SOURCE "================================================================================\n";
print SOURCE "\n";

close SOURCE;
```

Figura 3.20 – Código do Programa sendsms.pl.

Este programa usa um pacote Perl Chamado “Webmail::Gmail”, que implementa o protocolo SSL para se conectar ao servidor do Gmail.

O programa *sendsms.pl*, faz a leitura de um arquivo texto *(emaildados.txt),* localizado no diretório padrão da aplicação, que contem as informações de conta de e-mail SMTP, e destinos SMS e E-mail que devem receber as mensagens, ele também acessa um arquivo de LOG *(smslog.log),* também localizado no diretório padrão do sistema, onde são armazenadas informações sobre as mensagens enviadas, como horário de envio e destinos. Este arquivo serve para monitorar o funcionamento do sistema.

Figura 3.21 – Arquivo emaildados.txt com dados para envio de mensagens.

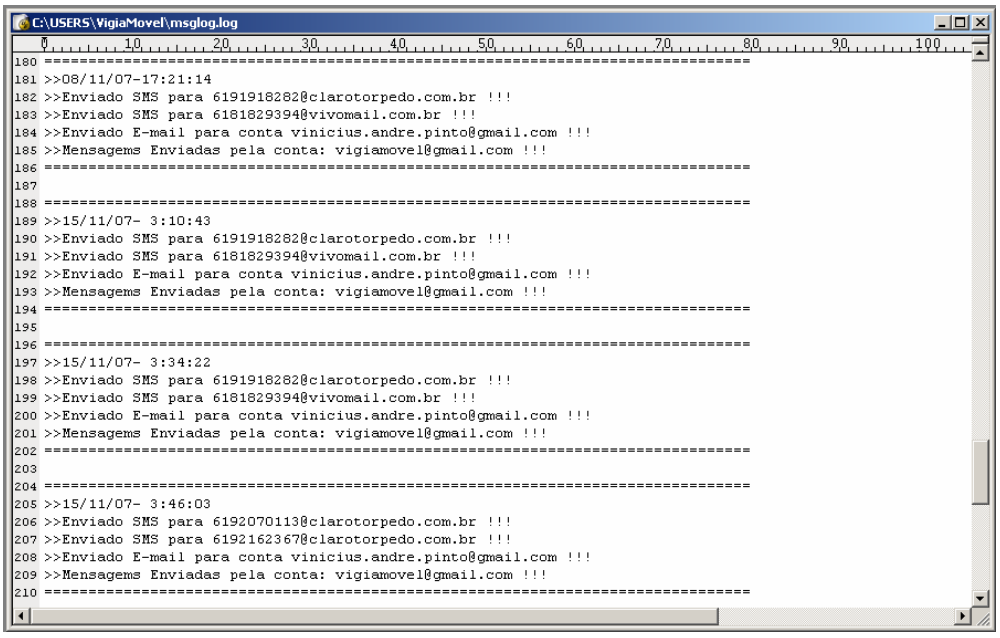


Figura 3.22 – Arquivo smslog.log contém informações sobre o envio das mensagens.

Na figura 3.21 é possível verificar ao dados de conta do Gmail usado no protótipo, foi criada uma conta especialmente para o sistema, e os destinos de SMS e e-mail para onde devem ser enviadas as mensagens.

Para tornar o programa mais flexível, foi criada uma interface na qual os destinos de SMS e e-mail podem ser alterados pelo usuário, esta interface faz parte do programa principal, e espera como entrada que o usuário digite o numero do telefone principal para envio de SMS junto com seu código DDD sem o zero (exemplo 6199998888), e escolha a operadora, em um ComboBox. Neste protótipo foram utilizadas três operadoras, mas podem ser acrescentadas outras o mesmo deve ser feito para um numero de telefone secundário, e para uma conta de e-mail qualquer.

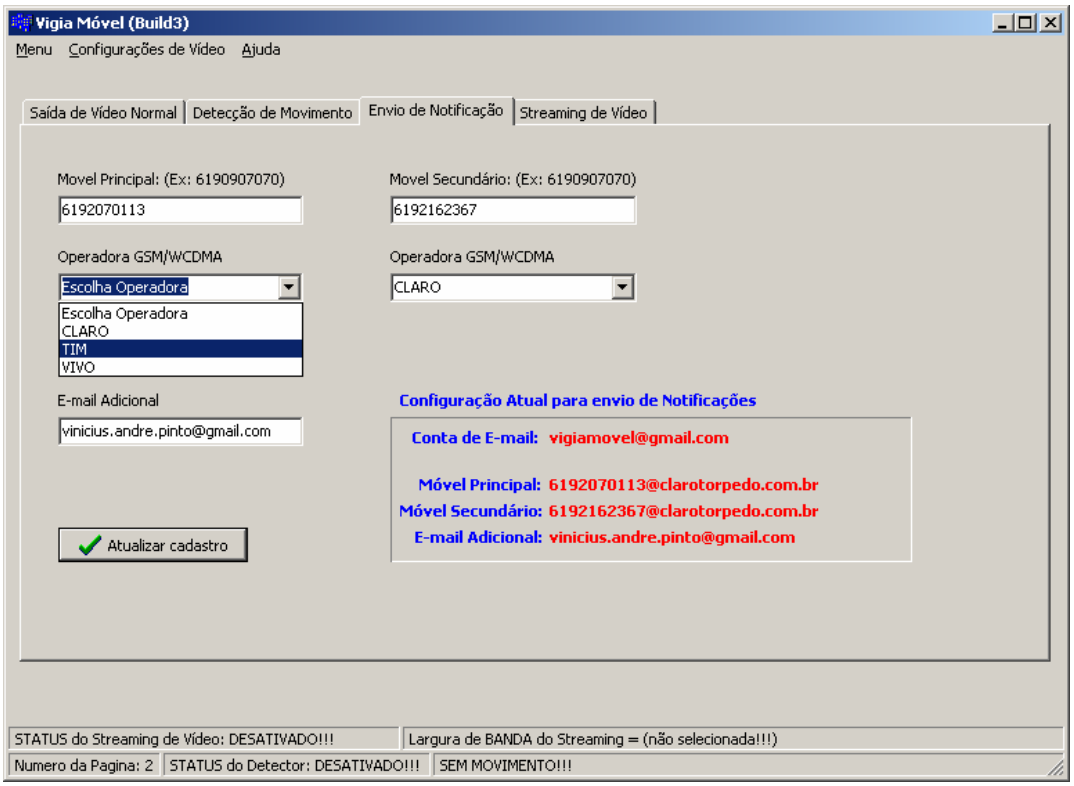


Figura 3.23 – Interface para configuração dos destinos para envio de SMS e E-mail.

Ao pressionar a tecla “Atualizar Cadastro”, o sistema “abre” o arquivo *emaildados.txt*, e atualiza os campos de destino para que ele possam ser lidos corretamente pelo programa *“sendsms.pl”*, e também atualiza a tela da interface com os novos dados inseridos.

```
//---------------------------------------------------------------------------

void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{

    char linha1[128], linha2[128], linha3[128], linha4[128], linha5[128], linha6[128];
    AnsiString linha12, linha22, linha32, linha42, linha52, linha62;
    AnsiString Op1, Op2, Dom1, Dom2, NovaL4, NovaL5, NovaL6;

     ifstream ler_arquivo("C:\\Users\\VigiaMovel\\emaildados.txt");

     ler_arquivo.getline(linha1, sizeof(linha1));
     ler_arquivo.getline(linha2, sizeof(linha2));
     ler_arquivo.getline(linha3, sizeof(linha3));
     ler_arquivo.getline(linha4, sizeof(linha4));
     ler_arquivo.getline(linha5, sizeof(linha5));
     ler_arquivo.getline(linha6, sizeof(linha6));

     ler_arquivo.close();

     linha42=linha4; linha52=linha5; linha62=linha6;

     //Define Operadoras

     Op1=Cbox_Operadora1->Items->Strings[Cbox_Operadora1->ItemIndex];
     Op2=Cbox_Operadora2->Items->Strings[Cbox_Operadora2->ItemIndex];

     if (Op1 == "CLARO") { Dom1 = "@clarotorpedo.com.br"; }
     else if (Op1 == "TIM") { Dom1 = "@tim.com.br"; }
     else if (Op1 == "VIVO") { Dom1 = "@vivomail.com.br"; }
     else {  Dom1 = "@operadora.com.br"; }

     if (Op2 == "CLARO") { Dom2 = "@clarotorpedo.com.br"; }
     else if (Op2 == "TIM") { Dom2 = "@tim.com.br"; }
     else if (Op2 == "VIVO") { Dom2 = "@vivomail.com.br"; }
     else {  Dom2 = "@operadora.com.br"; }

     NovaL4 = "MP=" + Ed_MSISDN1->Text + Dom1;
     NovaL5 = "MS=" + Ed_MSISDN2->Text + Dom2;
     NovaL6 = "EA=" + Ed_Email_Adic->Text;

    //Remonta arquivo

     char *L4final, *L5final, *L6final;
     //Converte AnsiString to *char...
     L4final=NovaL4.c_str();
     L5final=NovaL5.c_str();
     L6final=NovaL6.c_str();

     ofstream escreve_arquivo("C:\\Users\\VigiaMovel\\emaildados.txt");

     escreve_arquivo <<  linha1  << endl;
     escreve_arquivo <<  linha2  << endl;
     escreve_arquivo <<  linha3  << endl;
     escreve_arquivo <<  L4final << endl;
     escreve_arquivo <<  L5final << endl;
     escreve_arquivo <<  L6final << endl;

     escreve_arquivo.close();

     //Verifica arquivo...

     ifstream verifica_arquivo("C:\\Users\\VigiaMovel\\emaildados.txt");

     verifica_arquivo.getline(linha1, sizeof(linha1));
     verifica_arquivo.getline(linha2, sizeof(linha2));
     verifica_arquivo.getline(linha3, sizeof(linha3));
     verifica_arquivo.getline(linha4, sizeof(linha4));
     verifica_arquivo.getline(linha5, sizeof(linha5));
     verifica_arquivo.getline(linha6, sizeof(linha6));

     verifica_arquivo.close();

     linha32=linha3; linha42=linha4; linha52=linha5; linha62=linha6;

     linha32 = linha32.SubString(7,linha32.Length());
     linha42 = linha42.SubString(4,linha42.Length());
     linha52 = linha52.SubString(4,linha52.Length());
     linha62 = linha62.SubString(4,linha62.Length());

     Lb_Conta_Atual->Caption = linha32;
     Lb_Mp_Atual->Caption = linha42;
     Lb_Ms_Atual->Caption = linha52;
     Lb_Ea_Atual->Caption = linha62;
}
//---------------------------------------------------------------------------
```

Figura 3.24 – Código criado para atualizar o arquivo maildados.txt.

A figura 3.24, mostra o código em C++ para implementar a atualização do arquivo maildados.txt pela interface do programa, o código avalia a operadora escolhida no ComboBox, e em função da escolha, é gerado uma string de caracteres, com o domínio correto para onde deve ser enviado o SMS.

O envio de SMS e E-mails, é controlado por um evento que pertence ao componente *"TVLMotionDetect"*, chamado *"OnMotionDetect"*. Este evento é chamado toda vez que o componente detecta um movimento, este evento sempre que é chamado executa a função *"VLMotionDetect1MotionDetect"*, sendo que esta função atualiza o Log do sistema e faz uma chamada a *sendsms.pl*, para enviar as mensagens SMS e o E-mail, cadastrados em *maildados.txt,* conforme mostrado no código da figura 3.25.

```
//---------------------------------------------------------------------------

void __fastcall TForm1::VLMotionDetect1MotionDetect(TObject *Sender,
      int AMaxValue, TPoint &ACell)
{
  //Atualiza Log de Movimentos!!!

  Log_Memo->Lines->Add(AnsiString(DateTimeToStr(Now()) +
                       " => Movimento na Célula: ") + ACell.x + "," + ACell.y);
   StatusBar1->Panels->Items[2]->Text = " EVENTO: " + AnsiString(DateTimeToStr(Now()) +
                       " => Movimento na Célula: ") + ACell.x + "," + ACell.y;

////////////////////////////////////////////////////////////////////////////////

  //Envio de  Notificação SMS e EMAIL...

  if (gatilho)
   {
    gatilho=0;

    //Enviando Mensagem

    ShellExecute(Handle,"open", "C:\\cygwin\\bin\\perl", "C:\\Users\\VigiaMovel\\sendsms.pl", NULL, SW_HIDE);
    hora_envio = time(NULL);
    Log_Memo->Lines->Add(AnsiString(DateTimeToStr(Now()) + " => NOTIFICAÇÃO SMS ENVIADA!... "));
   }
  if (gatilho == 0)
   {
     hora_atual = time(NULL);
     dif = difftime(hora_atual, hora_envio);
     if(dif > 90)
      {
       ShellExecute(Handle,"open", "C:\\cygwin\\bin\\perl", "C:\\Users\\VigiaMovel\\sendsms.pl", NULL, SW_HIDE);
       hora_envio = time(NULL);
       Log_Memo->Lines->Add(AnsiString(DateTimeToStr(Now()) + " => NOTIFICAÇÃO SMS ENVIADA!...(+)"));
      }
   }



}
//---------------------------------------------------------------------------
```

Figura 3.25 – Código criado para enviar SMS e E-mail em função do Evento "OnMotionDetect".

Para que não haja o envio repetitivo de e-mail e SMS para os destinos indicados, o Código envia um primeiro e-mail, e só enviará outro 3mim depois, os Movimentos detectados neste período não vão fazer chamadas ao programa "*sendsms.pl*". Esta implementação é feita, capturando-se sempre o horário de envio da última notificação, e então, para o próximo evento de

movimento, a hora atual é comparada com a do envio anterior, se o tempo for superior a 90s, a mensagem é enviada, caso contrário nada é executado, desta forma criou-se uma forma de evitar e-mail desnecessários sendo enviados aos destinos. Contudo este parâmetro pode ser configurado para valores maiores ou menores de tempo, em função da preferência do usuário.

3.2.4 GERAÇÃO DE STREAMING E ENVIO DE PACOTES RTP

A aplicação Vigia Móvel, tem a capacidade de converter as imagens captadas pela Webcam para o formato mp4, e gerar uma streaming RTP/RTSP para enviar estes pacotes ao servidor de streaming, que pode estar na própria máquina (127.0.0.1) ou localizado em qualquer parte da Internet.

A padronização das redes móveis pelo 3GPP estabeleceu que o formato de mídia padrão para transmissão de streaming em redes móveis é o 3gp ou mp4, isto porque o formato 3gp é praticamente o mesmo formato mp4 com algumas mudanças para compressão de áudio (AMR e AAC). Como a transmissão de áudio não está contemplada neste projeto, a diferença entre os dois formatos quase não se apresenta.

Para efetuar a compressão das imagens, segundo o 3GPP podemos usar dois tipos de codecs, o “H263” e o “MPEG4 Standard Profile”. Por se tratar de um codec mais moderno e eficiente para este projeto usaremos apenas o Codec “MPEG4 SP”.

As técnicas de programação utilizadas para desenvolver codificadores de imagens são muito sofisticadas, e envolvem estudo aprofundado sobre imagens e algoritmos de programação. Existem muitas teses de Mestrado e Doutorado nesta área.

Como o escopo do projeto está focado na utilização de streamings de vídeo para sistemas de vigilância, foi descartada a hipótese de desenvolver um codec próprio para o sistema. Desta forma foi utilizado no projeto um codec fabricado pela Helix (empresa especializada em Streaming para Internet e sistemas Móveis), chamado HMPROD. Este Codec trouxe duas soluções práticas para o projeto, pois além de gerar as imagens em formato mp4, ele também gera os pacotes RTP/RTSP e os envia para um servidor de destino.

O Codec HMPROD precisa ser configurado, para gerar a streaming no padrão e largura de banda desejada, e para enviá-la ao endereço correto.

Foi considerado neste projeto que o servidor de streamings, poderia estar localizado tanto dentro do PC que está rodando a aplicação Vigia Móvel, como também, localizado remotamente na internet, mas não ao mesmo tempo.

Na figura 3.26 é mostrado a topologia do sistema quando usado em configuração de servidor de interno, e na figura 3.27 é mostrado a topologia do sistema quando usado em configuração de servidor de streaming externo.

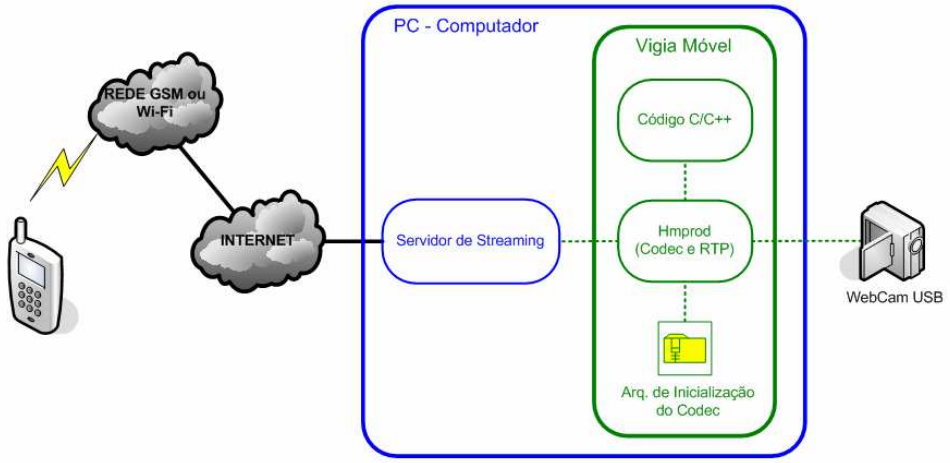


Figura 3.26 – Codec HMPROD, no Vigia Móvel, configurado para servidor Interno.

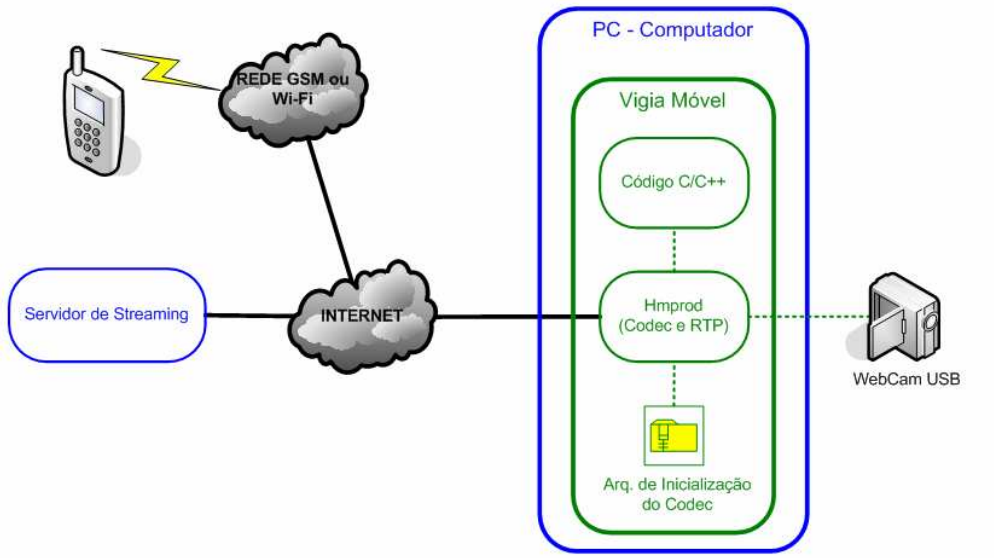


Figura 3.27 – Codec HMPROD, no Vigia Móvel, configurado para servidor Externo.

Na inicialização do Codec, é passado como argumento, o caminho para o arquivo de configuração que contém todos os dados necessários para configurá-lo, este arquivo deve estar no formato XML e deve conter vários parâmetros, como:

- Banda da streaming
- Endereço IP e porta de destino da Streaming RTP/RTSP
- Codec Utilizado (MPEG4)
- Dimensões da imagem
- Nome do arquivo SDP
- Dispositivo de Vídeo
- Dispositivo de áudio
- MTU, TTL e outros.

O Codec escolhido pode ser configurado para gerar streamings de Vídeo com áudio, mas para sistemas de vigilância o mais importante são as imagens. Portanto o protótipo implementado não faz captura nem codificação do áudio.

Futuramente o protótipo pode ser adaptado para incluir áudio na streaming.

A Figura 3.28 mostra um arquivo de configuração para o codec HMPROP. Este arquivo configura o codec para receber imagens do dispositivo de vídeo padrão, e gera uma streaming de vídeo de 50Kbps, usando padrão MPEG4 SP, com banda fixa (CBR) e enviando pacotes RTP para o endereço do servidor de streming 200.169.119.105 na porta UDP 50000.

O Processo de geração de streaming para envio a servidores exige que o codificador monte um arquivo com extensão SDP *(Session Description Protocol)*. Este arquivo contém informações sobre a streaming gerada, como nome e banda, tipo de pacotes, bem como detalhes de como os clientes de streaming devem se configurar para receber os pacotes RTP.

Portanto sempre que a configuração do codec é alterada um novo arquivo SDP deve ser criado. Este arquivo SDP deve ser armazenado no diretório padrão do servidor de Streaming, pois é por ele, que os clientes se conectam ao servidor.

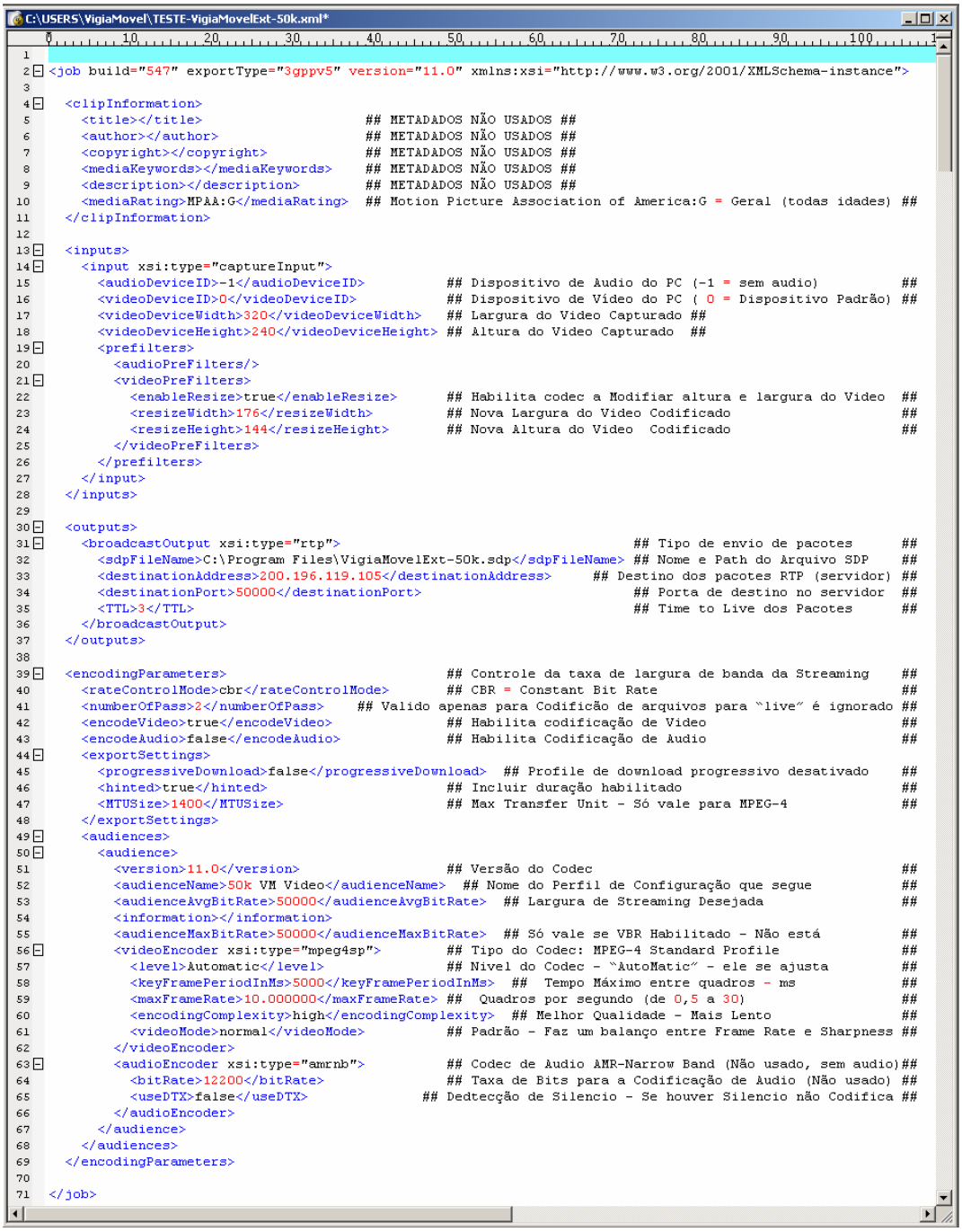


Figura 3.28 – Arquivo XML com configuração do codec para gerar streming de 50kbps.

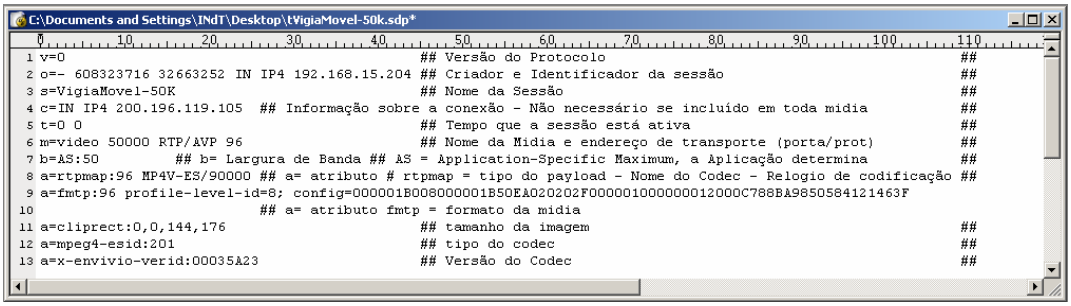


Figura 3.29 – Arquivo SDP Gerado para a configuração de streaming da figura 3.28 (50kbps/MPEG4 SP).

Como mencionado no capítulo 2, o servidor de streaming usado no projeto é o Darwing streaming Server (DSS), que é um servidor de código aberto, desenvolvido pela Apple, similar ao Quick Time Streaming Server, que é um produto comercial da Apple.

Em função disto sempre que as configurações de streaming do codec são alteradas no arquivo XML deve-se gerar um novo arquivo SDP e copiá-lo para o diretório padrão do servidor de streaming, que para Sistemas operacionais Windows é: *C:\Program Files\Darwin Streaming Server\Movies* e para Sistemas operacional LINUX é: */usr/local/movies.*

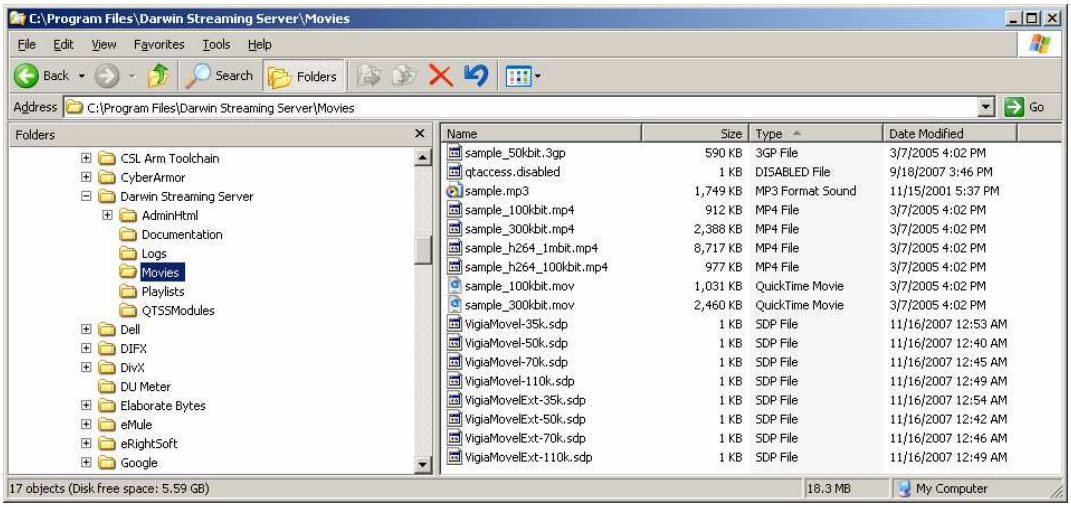


Figura 3.30 – Diretório onde são armazenados os arquivos SDP no servidor DSS WinXP.

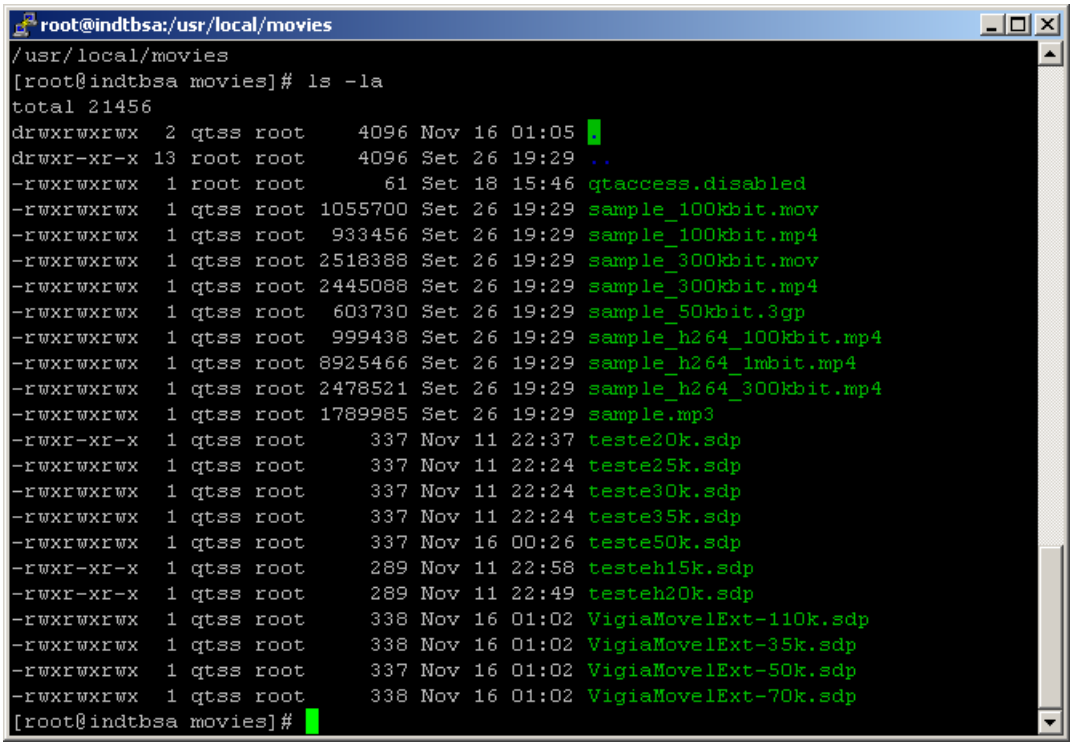


Figura 3.31 – Diretório onde são armazenados os arquivos SDP no servidor DSS LINUX.

Quando a aplicação usa servidor Interno (no próprio PC) o servidor usado será o DSS para Windows, quando usar servidor externo (Internet) pode-se ter a opção de um servidor rodando LINUX ou Windows.

Para evitar erros na criação e no armazenamento em diretório correto dos arquivos SDP, a aplicação não possibilita o usuário escolher a largura de banda e qualquer outro parâmetro do CODEC, pelo menos na versão atual. Desta forma o usuário precisa apenas escolher entre as larguras de banda disponíveis na interface gráfica ou escolher entre servidor interno ou externo (ver figura. 3.32).

Outro fator que levou a não disponibilizar as alterações de parâmetros e banda da Streaming, é que em função da rede de acesso usada pelo móvel, (Wi-Fi, ou GSM ou WCDMA), as larguras de banda fora do ajuste correto podem ocasionar perda de conexão com o servidor de streaming.

As redes de acesso móvel possuem características de largura de banda variável, em função do tráfego cursado nas estações, portanto em regiões de alta densidade de tráfego pode haver diminuição na largura de banda efetiva disponível para cada usuário, e isto afeta o tráfego de streaming de dados.

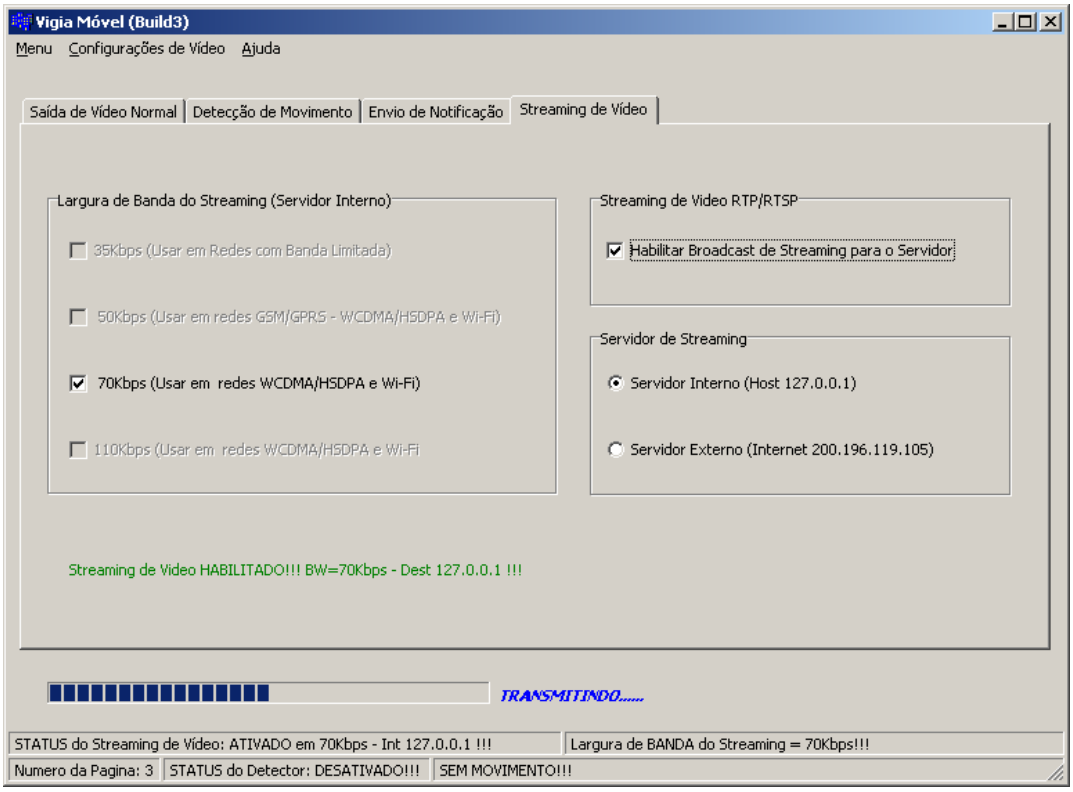


Figura 3.32 – Interface de configuração da Streaming e envio de pacotes RTP/RTSP.

Na interface de configuração de streaming da aplicação temos algumas opções de banda para streaming de vídeo em função das redes de acesso, e a opção de servidor de streaming (Interno ou externo), como pode ser visto na figura 3.32. Para chegar aos valores de banda disponível como opções do sistema, foram feitos alguns testes baseados nas sugestões da documentação da sobre Streaming de Vídeo para Dispositivos Móveis da NOKIA (Vídeo Streaming on Móbile Devices - V3.0), que é o maior fabricante de dispositivos móveis do mundo e grande incentivador do uso de multimídia em celulares.

A tabela 3.1, mostra os valores de largura de banda sugeridos pela Nokia para streaming de vídeo em redes móveis em função da tecnologia de acesso. Os valores apresentados fazem parte da documentação para utilização de multimídia em redes móveis publicado pela NOKIA, no site www.forum.nokia.com .

Tabela 3.1 – Banda sugerida pela documentação da NOKIA para streaming de vídeo em função da tecnologia de acesso.

Configuration Number	Network	Configuration	Available Bandwidth, kbit/s	Suggested Bit Rate for Streamable Content, kbit/s
1**	HSCSD	2 TS DL	28.8	20
2**	HSCSD	3 TS DL	43.2	35
3	GPRS	2 TS DL (CS 2)	26.8	20
4	GPRS	3 TS DL (CS 2)	40.2	28
5	GPRS	4 TS DL (CS 2)	53.6	38
6	EGPRS	2 TS (MCS 6)	59.2	42
7	EGPRS	3 TS (MCS 6)	88.8	63
8	EGPRS	4 TS (MCS 6)	118.4	84.2
9	WCDMA	GBR=MBR=64	64	47
10	WCDMA	GBR=MBR=128	128	92

** Connection method could be ISDN V.110, ISDN V.120, or normal modem type

Baseado nas recomendações da tabela 3.1 acima, foi pré-configurado no sistema Vigia Móvel a banda de 35Kbps para as tecnologias HSCSD e GPRS (máximo de 35Kbps e máximo de 38Kbps respectivamente) ambas as tecnologias são pouco usadas em redes móveis, 50Kbps e 70Kbps para Sistemas EGPRS, e 110Kbps para sistemas WCDMA e sistemas Wi-Fi.

Para implementar as opções de banda (35k, 50k, 70k e 110k) e os dois modos de localização do servidor (local e remoto) foram montados 6 arquivos de configuração XML para o codec, estes arquivos foram armazenados no diretório padrão da aplicação (C:\Users\VigiaMovel), a tabela 3.2 mostra o nome dos arquivos e configuração relacionada. A figura 3.33 mostra o diretório padrão do sistema com os arquivos XML armazenados.

Tabela 3.2 – Arquivos de configuração para o codificador HMPROD

Nome do Arquivo	Banda	Codec	IP destino	Porta UDP	Servidor
VigiaMovel-35k.xml	35kbps	MPEG4	127.0.0.1	50000	Local
VigiaMovel-50k.xml	50kbps	MPEG4	127.0.0.1	50000	Local
VigiaMovel-70k.xml	70kbps	MPEG4	127.0.0.1	50000	Local
VigiaMovel-110k.xml	110kbps	MPEG4	127.0.0.1	50000	Local
VigiaMovelExt-35k.xml	35kbps	MPEG4	200.196.119.105	50000	Remoto
VigiaMovelExt-50k.xml	50kbps	MPEG4	200.196.119.105	50000	Remoto
VigiaMovelExt-70k.xml	70kbps	MPEG4	200.196.119.105	50000	Remoto
VigiaMovelExt-110k.xml	110kbps	MPEG4	200.196.119.105	50000	Remoto

Diretório padrão: C:\Users\VigiaMovel

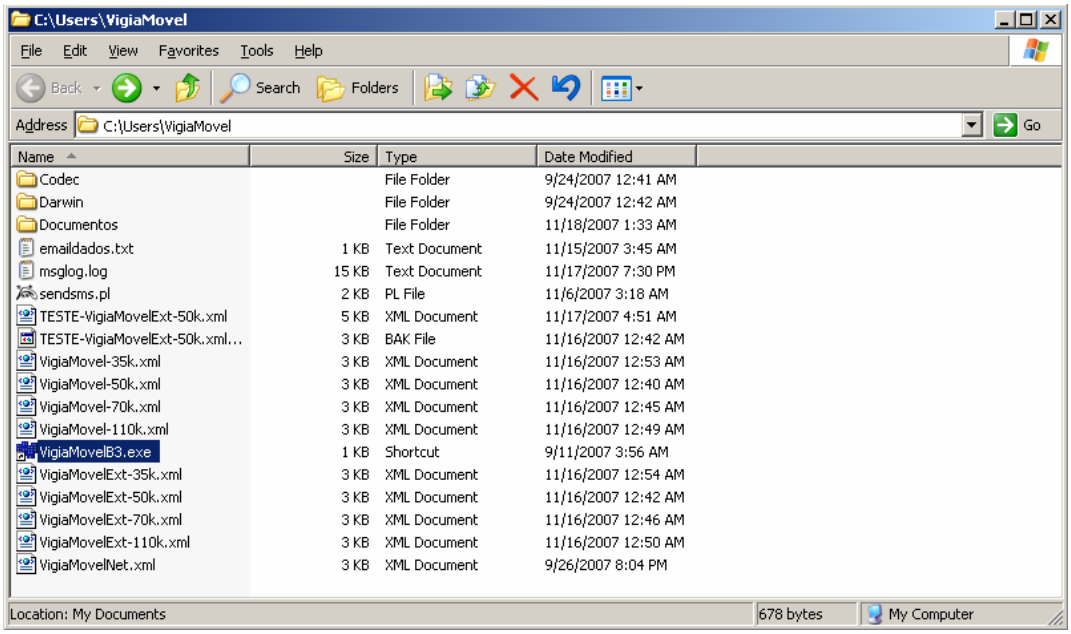


Figura 3.33 – Diretório padrão do Vigia Móvel com os arquivos XML.

Para que as escolhas do usuário na interface gráfica sejam traduzidas de forma correta pelo sistema foi criada uma série de “if e else” no código C/C++, que faz uma chamada a função ShellExecute, esta função inicia o codificador HMPROD, lhe fornecendo como argumento o arquivo que satisfizer as escolhas do usuário.

A chamada de ShellExecute cria um processo na máquina chamado HMPROD, conforme mostrado na figura 3.35, onde podemos ver o gerenciador de processos do Windows, mostrando o processo “hmprod.exe”.

```
//---------------------------------------------------------------------------


void __fastcall TForm1::CheckBoxRTPClick(TObject *Sender)
{
  TimerProgBar->Enabled = true;
  LabelTx->Visible = true;
  LabelTx->Caption = "TRANSMITINDO.";

  if (CheckBoxRTP->Checked)
  {
    if (CheckBox35k->Checked)
    {
     if (RadioButtonServInt->Checked)
     {
      ShellExecute(Handle,
      "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
      "-j C:\\Users\\VigiaMovel\\VigiaMovel-35k.xml", NULL, SW_SHOWMINNOACTIVE);
      StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Video: ATIVADO em 35Kbps - Int 127.0.0.1 !!!";
      Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 35Kbps - Dest 127.0.0.1 !!!" ));
      LbRtpStatus->Font->Color = clGreen;
      LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=35Kbps - Dest 127.0.0.1 !!!";
     }
     else if (RadioButtonServExt->Checked)
     {
      ShellExecute(Handle,
      "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
      "-j C:\\Users\\VigiaMovel\\VigiaMovelExt-35k.xml", NULL, SW_SHOWMINNOACTIVE);
      StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Video: ATIVADO em 35Kbps - Ext 200.196.119.105 !!!";
      Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 35Kbps - Dest 200.196.119.105 !!!" ));
      LbRtpStatus->Font->Color = clGreen;
      LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=35Kbps - Dest 200.196.119.105 !!!";
     }
    }
    else if (CheckBox50k->Checked)
    {
     if (RadioButtonServInt->Checked)
     {
      ShellExecute(Handle,
      "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
      "-j C:\\Users\\VigiaMovel\\VigiaMovel-50k.xml", NULL, SW_SHOWMINNOACTIVE);
      StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Video: ATIVADO em 50Kbps - Int 127.0.0.1 !!!";
      Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 50Kbps - Dest 127.0.0.1 !!!" ));
      LbRtpStatus->Font->Color = clGreen;
      LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=50Kbps - Dest 127.0.0.1 !!!";
     }
     else if (RadioButtonServExt->Checked)
     {
      ShellExecute(Handle,
      "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
      "-j C:\\Users\\VigiaMovel\\VigiaMovelExt-50k.xml", NULL, SW_SHOWMINNOACTIVE);
      StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Video: ATIVADO em 50Kbps - Ext 200.196.119.105 !!!";
```

Figura 3.34 – Mostra parte do código usado para acionar o codec HMPROD.

Na figura 3.34 podemos ver a função ShellExecute, e o arquivo XML declarado como argumento da função, a função abre o arquivo, e faz a leitura dos parâmetros para configurar o Codec, iniciando-o em seguida.

Após a chamada o funcionamento do Codec passa a ser controlado pelo processo HMPROD. Se o usuário solicitar o desligamento do Codec através do CheckBoxRTP na interface gráfica (mesma que o iniciou), o processo é automaticamente “Terminado” pela função “TerminateProcess” Esta função procura pelo processo e lhe envia um sinal para terminar. A figura 3.35 mostra o processo rodando, e na figura 3.36 podemos ver o código usado para implementar a função que termina o processo “hmprod.exe”.

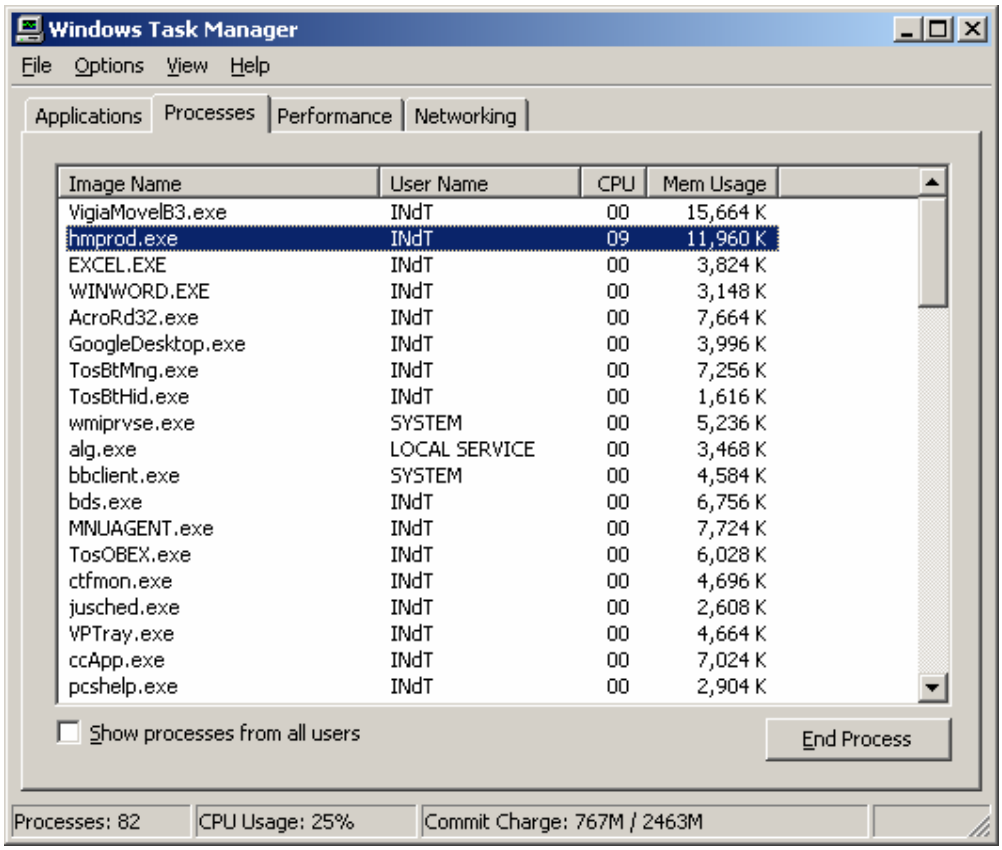


Figura 3.35 – Lista de Processos do Windows pode-se ver o hmprod.exe.

```
    HWND hwndHMPROD;
    if ((hwndHMPROD = FindWindow(NULL,
        "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe")) != NULL)
    {
       //DWORD -> 32-bit unsigned integer
       DWORD ID;
       //UINT -> Unsigned INT
       UINT ExitCode = 1;
       HANDLE hdl;

       GetWindowThreadProcessId(hwndHMPROD, &ID);
       hdl = OpenProcess(PROCESS_ALL_ACCESS,false, ID);

       TerminateProcess(hdl, ExitCode);
       StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: DESATIVADO!!!";
       Log_Memo->Lines->Add(AnsiString ("TERMINANDO o ENVIO de PACOTES RTP!!!" ));
       LbRtpStatus->Font->Color = clRed;
       LbRtpStatus->Caption = "Streaming de Video DESABILITADA!!!";
    }
    else
    {
       ShowMessage("Impossivel Interromper o CODEC hmprod.exe, favor terminar manualmente!!");
    }
  }
}
//---------------------------------------------------------------------------
```

Figura 3.36 – Código usado para terminar o processo hmprod.exe.

3.2.5 GRAVAÇÃO DE IMAGENS COM MOVIMENTOS

A aplicação foi desenvolvida para gravar as imagens em que são detectados movimentos. O objetivo é armazenar estas imagens para que o usuário possa conferir posteriormente o que ocorreu no ambiente.

Para efetuar a gravação das imagens foi usado um componente TVLDSVideoLogger. Este componente pode gravar as imagens recebidas em sua porta InputPin em vários formatos de arquivos de mídia diferente, dependendo dos Codecs disponíveis no sistema operacional.

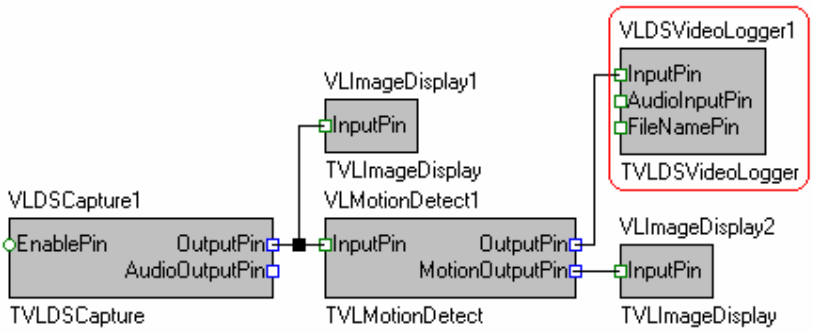


Figura 3.37 – Componente VideoLogger conectado a o componente MotionDetect.[Mitov,2007]

Como explicado anteriormente, o componente MotionDetect, possue uma saída chamada OutputPin que exporta os quadros com movimento das imagens monitoradas. Esta saída é conectada à entrada do componente VídeoLogger que comprime a imagem no formato desejado e grava em um arquivo pré-estabelecido. A configuração do VideoLogger pode ser feita diretamente pela janela de propriedades do C++ Builder.

As duas propriedades necessárias para o funcionamento do componente são “InputPin” e “Compressions”. A primeira especifica de onde vêm as informações a serem gravadas (no caso MotionDetect) e a segunda especifica o Compressor (Codec) usado para comprimir, as informações antes de serem gravadas.

O Componente possui também a propriedade booleana “Enable”, que determina quando as informações devem ser gravadas, quando o valor for “true” ele inicia a gravação e quando “false”, são paradas as gravações.

O nome do arquivo também deve ser passado para o componente com o caminho completo, mas o nome do arquivo pode ser passado em tempo de execução Para a aplicação deste projeto, toda vez que o componente inicia a gravação de um arquivo acionado pela propriedade “enable”, é criado um nome de arquivo que contém a data e horário em que ele foi iniciado. Esta implementação serve para armazenar diferentes arquivos caso a gravação de arquivos seja acionada ou terminada os arquivos ficam armazenados no diretório padrão do sistema. A figura 3.38 mostra o quadro de propriedades do componente “TVLDSVideoLogger”, no C++ Builder.

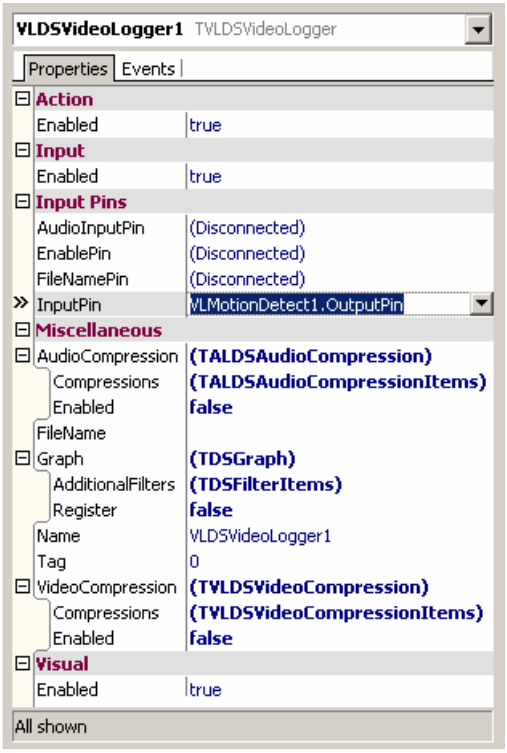


Figura 3.38 – Propriedades do componente VideoLogger.

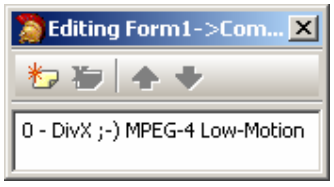


Figura 3.39 – Propriedade Compressions com o nome do compressor utilizado.

A figura 3.39 mostra qual o compressor usado para gravar os arquivos de imagens. O componente aceita somente os compressores disponíveis no sistema Operacional neste caso foi escolhido o compressor “MPEG4-Low-Motion”, que codifica os arquivos em formato MP4, apropriado para imagens com pouco movimento. Este Codec costuma estar disponível como parte do pacote de codecs do WinXp.

```
//---------------------------------------------------------------------------

void __fastcall TForm1::CheckBox2Click(TObject *Sender)
{
  if (CheckBox2->Checked)
  {
   iAno=Now().FormatString("yyyy");
   iMes=Now().FormatString("mm");
   iDia=Now().FormatString("dd");
   iHora=Now().FormatString("hh");
   iMinuto=Now().FormatString("nn");
   DataFormatada= iDia+iMes+iAno+iHora+iMinuto;
   ArqComp = "C:\\Users\\VigiaMovel\\MovDet"+DataFormatada+".mpg";
   VLDSVideoLogger1->FileName = ArqComp;
   VLDSVideoLogger1->VideoCompression->Enabled = true;
   VLDSVideoLogger1->Enabled = true;
   //ShowMessage(ArqComp);
   Log_Memo->Lines->Add(AnsiString ("Inicio de Gravação dos Movimentos!!!" ));
   Log_Memo->Lines->Add(AnsiString ("Nome do Arq =>" )+ ArqComp);
   Lb_aviso_de_mov1->Font->Color = clGreen;
   Lb_aviso_de_mov1->Caption = "Movimentos estão sendo Gravados!!!";


   }
   else if (!CheckBox2->Checked)
   {
    VLDSVideoLogger1->Enabled = false;
    VLDSVideoLogger1->VideoCompression->Enabled = false;
    Log_Memo->Lines->Add(AnsiString ("Fim de Gravação dos Movimentos!!!" ));
    Lb_aviso_de_mov1->Font->Color = clRed;
    Lb_aviso_de_mov1->Caption = "Movimentos não estão sendo Gravados!!!";
   }
}

//---------------------------------------------------------------------------
```

Figura 3.40 – Código usado para iniciar a gravação e gerar o nome dos arquivos.

O código da figura 3.40 foi implementado dentro da função chamada pelo evento “OnClick” do CheckBox e usado para selecionar o inicio da gravação na interface gráfica. O código habilita o inicio da gravação e determina o nome do arquivo onde as imagens serão gravadas. Além disso, a data e a hora de inicio são acrescentados ao nome do arquivo da seguinte maneira:

Nome do arquivo: MovDet+dia+mês+ano+hora+minuto.mpg

Ex: data 17/11/2007 hora 03:37am

Nome de Arquivo resultante: MovDet171120070337.mpg.

3.2.6 VISUALIZAÇÃO DE STREAMING PELO MÓVEL

Para visualizar as streamings geradas pela central de vigilância, os dispositivos móveis necessitam apenas de um tocador de streamings MP4 a maioria dos dispositivos disponíveis no mercado dispõem deste recurso.

O tocador de streaming deve ser direcionado para o servidor de streaming através da URL + nome do arquivo SDP pelo browser do aparelho, que posteriormente irá encaminhar a execução da streaming para o tocador do móvel, como mostrado nas figuras 3.41 e 3.42.

Figura 3.41 – Iniciando a visualização de streming de 50kbps externa em um NOKIA N77.

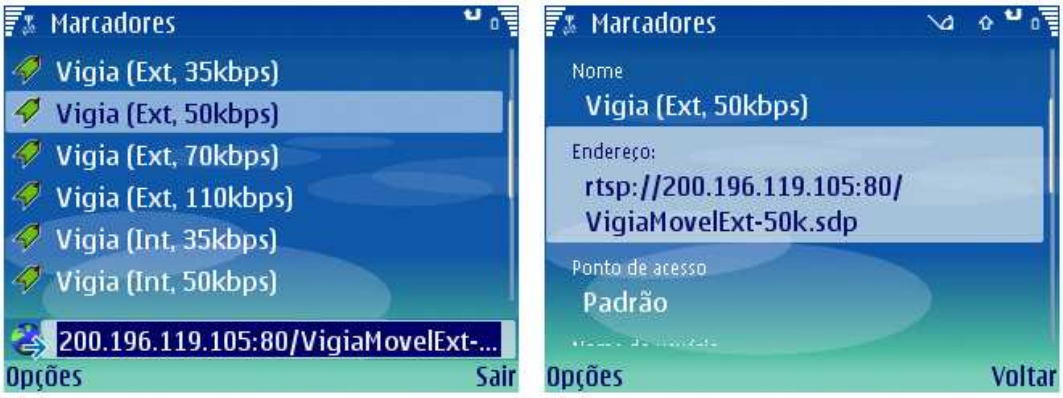


Figura 3.42 – Iniciando a visualização de streming de 50kbps externa em um NOKIA E61.

O procedimento descrito acima é um pouco trabalhoso, pois exige que o usuário digite a URL completa para a streaming, e normalmente é difícil lembrar estes tipos de endereços. Para facilitar a utilização dos aparelhos existe a opção de guardar as URLs em uma lista de favoritos, mas isto pode não ser a melhor solução para o caso de ter que mudar de aparelho.

Para solucionar estes problemas foi desenvolvida uma pequena aplicação em Flash Lite 1.1, que incorpora as URLs da aplicação VigiaMóvel, e as disponibiliza em forma de Botões que podem ser acionados pelo usuário. Esta aplicação facilita a utilização do dispositivo móvel pois não exige que o usuário tenha conhecimento das URLs usadas e facilita a troca de aparelhos, pois pode ser instalado em qualquer dispositivo móvel com suporte a Flash Lite.

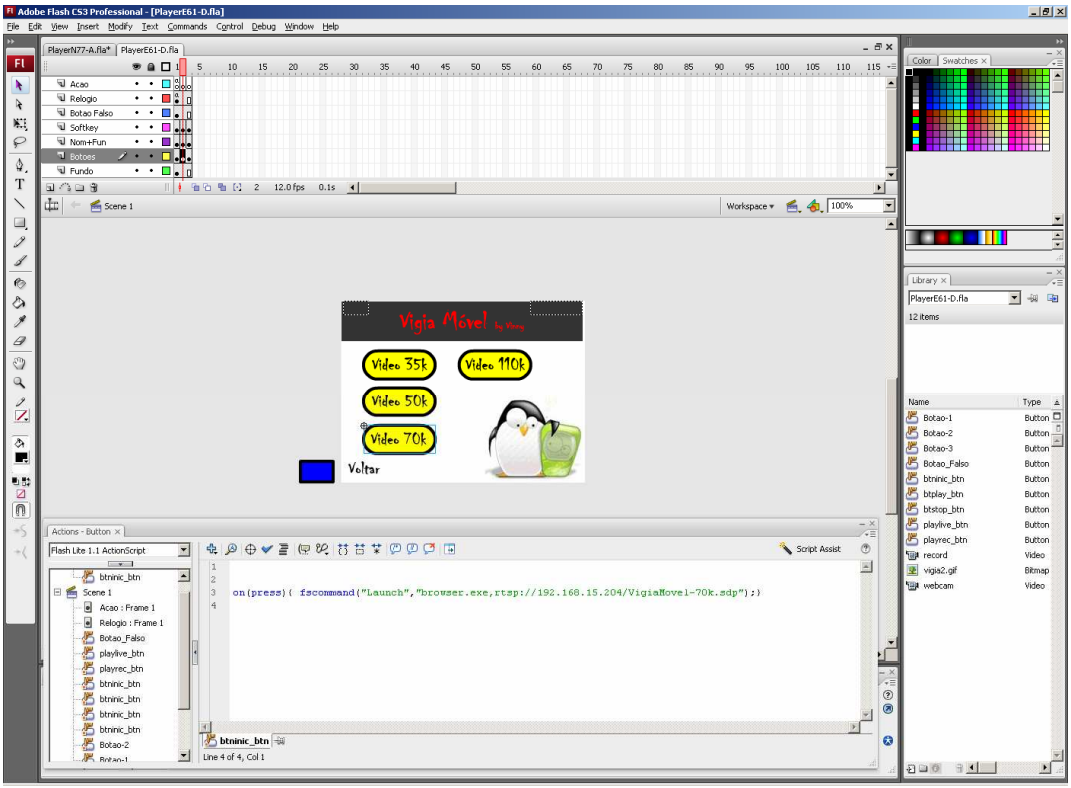


Figura 3.43 – Flash CS3 ambiente onde foi desenvolvido cliente para o móvel.

Na figura 3.43 é mostrado o ambiente de programação Flash CS3.0 onde foi desenvolvido o Cliente em Flash Lite que direciona as URLs para a o tocador de stream do móvel.

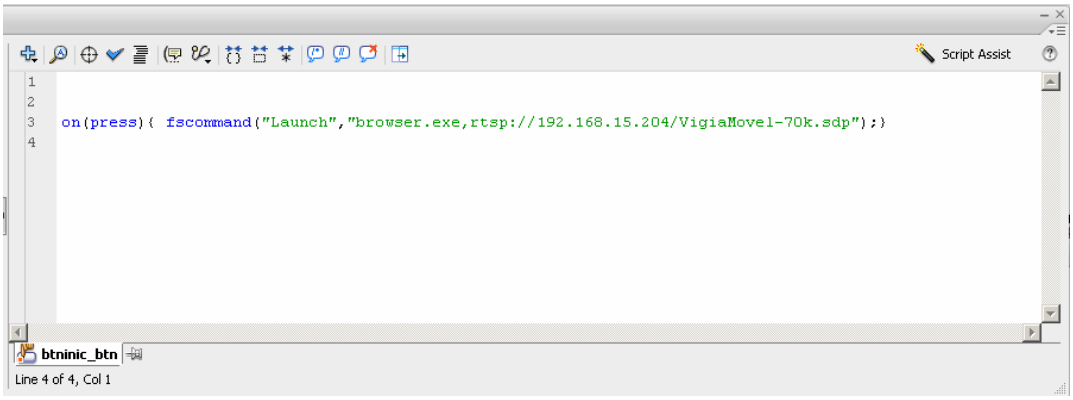


Figura 3.44 – Código Action Script usado no cliente do móvel.

Na figura 3.44 é possível ver o código Action Script usado para direcionar o tocador de streaming do dispositivo móvel ao servidor de streamings. Este código faz uma chamada ao browser do dispositivo, passando como argumento a URL com o endereço do servidor de streaming e o nome do arquivo SDP da streaming desejada.

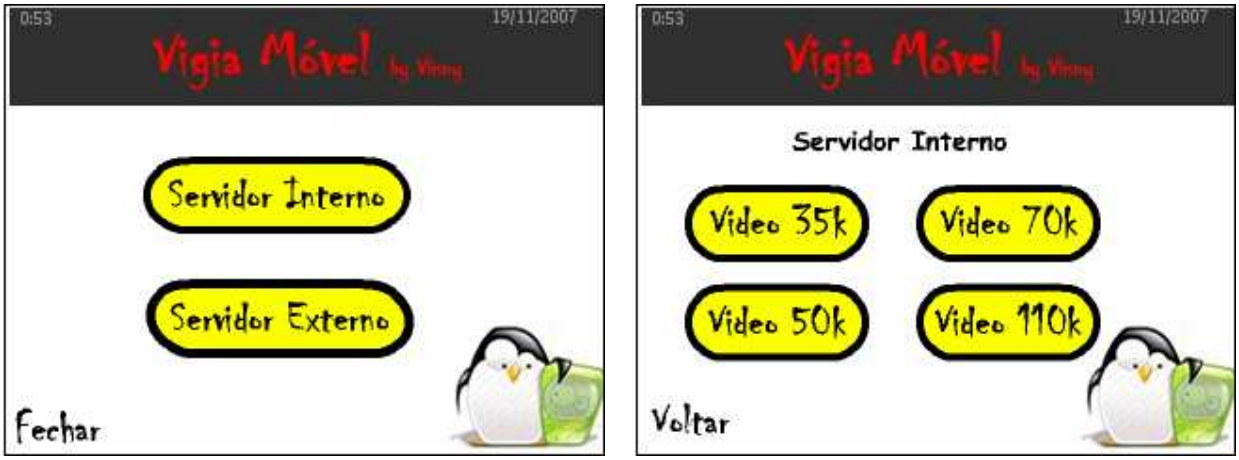


Figura 3.45 – Interface gráfica do software cliente.

Na Figura 3.45 podemos ver a interface gráfica do software cliente para o dispositivo móvel. Nele existem atalhos para as streamings, em todas as opções oferecidas pela aplicação Vigia Móvel, tanto para servidores locais como para servidores externos.

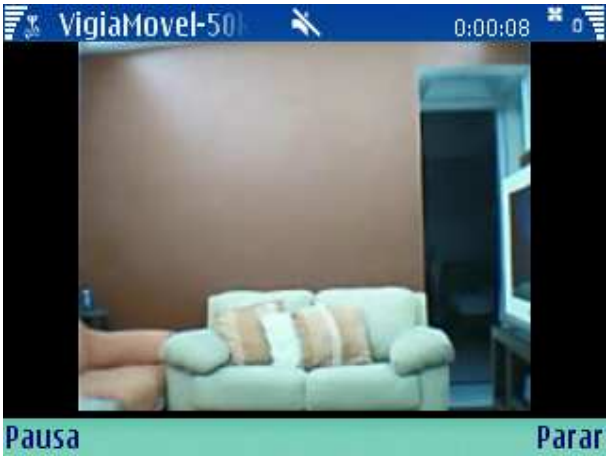


Figura 3.46 – Imagem apresentada pelo tocador de streaming do móvel, direcionada pelo software cliente.

Capítulo 4 – Principais resultados obtidos com a implementação do Projeto

4.1 INTRODUÇÃO

O Objetivo desta seção é apresentar os resultados obtidos através dos testes efetuados com o protótipo desenvolvido para este projeto.

Os fatores mais importantes na realização dos testes foram as configurações de rede IP para o servidor da central de vigilância, para o dispositivo móvel, bem como os testes de atraso entre a imagem capturada e a streaming recebida pelo dispositivo móvel.

4.2 TOPOLOGIA DOS TESTES

Inicialmente o projeto proposto não contemplava a utilização de sistemas Wi-Fi, mas problemas iniciais de restrição ao tráfego de acesso aos clientes de banda larga doméstica, levaram a considerar a utilização destas redes. Assim, a utilização dos sistemas Wi-Fi foi considerada para validar o sistema, antes de chegar ao objetivo final que eram as redes GSM e WCDMA.

Com a rede Wi-Fi foi possível efetuar um teste em um ambiente isolado sem nenhum tipo de restrição por parte da operadora que dá acesso ao dispositivo móvel, nem por parte da operadora de banda larga conectada a Central de Vigilância. Porém para isto, foi necessário encontrar um dispositivo móvel com interface Wi-Fi, além da interface GSM/WCDMA. Para satisfazer esta necessidade foi utilizado o aparelho NOKIA E61, que atende aos requisitos por possuir suporte as redes de acesso GSM/WCDMA e Wi-Fi. A figura 4.1 mostra o aparelho Nokia E61 utilizado na implementação e demonstração do projeto.

Figura 4.1 – Nokia E61 – GSM/WCDMA/Wi-Fi

Para simular uma rede com características similares à internet foi utilizado um roteador Wireless da Linksys WRTP54G, este roteador possui 4 portas LAN 10/100Mbits, Wi-Fi A/B/G, 1 porta Wan e 2 portas SIP. Este roteador, mostrado na Figura 3.42 é utilizado na implementação e demonstração do projeto.

Figura 4.2 – Linksys WRTP54G – LAN/WAN/Wi-Fi.

A topologia usada para este teste considera que o Roteador Wireless simula uma rede, representando a Internet, e que o Computador rodando a aplicação, e o dispositivo móvel, estão conectados a ele, através de Wi-Fi, conforme a figura 4.3.

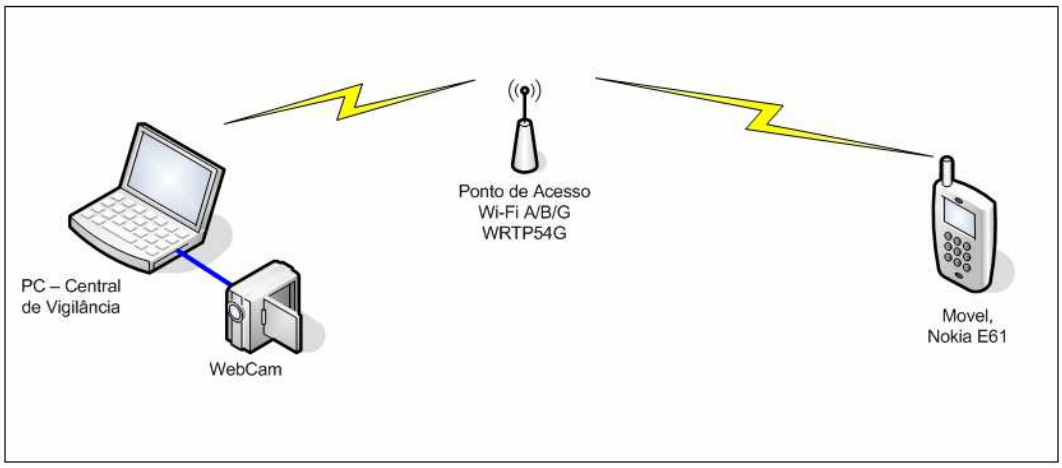


Figura 4.3 – Topologia do primeiro teste – rede Wi-Fi.

Utilizando a configuração da figura 4.3 foi possível gerar a streaming com o software desenvolvido, e conectar-se a esta streaming através do servidor de streamings, utilizando os atalhos do software desenvolvido para o dispositivo móvel.

Para o PC rodando o servidor foi necessário configurá-lo com IP fixo, pois a rede simulada não possuía um DNS para fazer resolução de nomes em IPs.

Depois deste resultado positivo, foi implementada uma nova topologia, baseada no ambiente “alvo” da aplicação. O Software Vigia Móvel, e o servidor de streaming, estava rodando em um PC, o qual foi conectado a Internet através de uma conexão de banda larga do provedor “Virtua”, com 2MBps de banda. O dispositivo móvel foi conectado a internet através de uma conexão GPRS/EGPRS da operadora CLARO, conforme mostrado na figura 4.4.

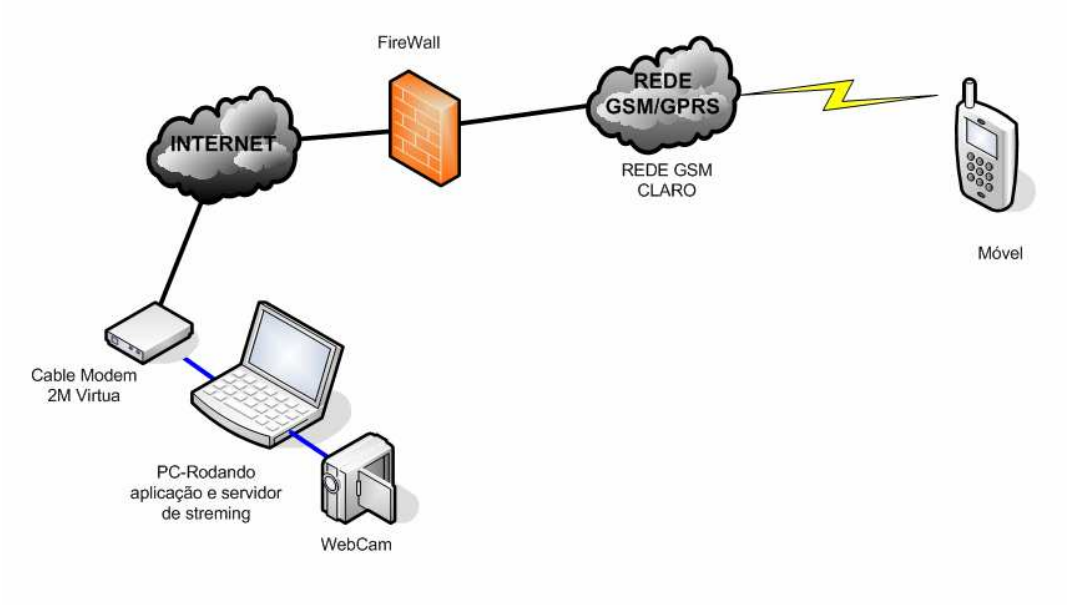


Figura 4.4 – Topologia do segundo teste – rede GSM.

Foram efetuadas várias tentativas de acesso, sem resultado, o dispositivo móvel acusava erro na conexão com o servidor, conforme mostrado na figura 4.5.

Figura 4.5 – Mensagem de erro após tentativas de conexão ao servidor.

Esta mensagem de erro caracteriza que o dispositivo móvel não conseguiu alcançar o servidor de streaming, portanto foi verificado o estado

das portas RTP/RTSP (TCP 554 e 7070, e UDP 6970, 6971) no PC, com o comando “netstat –a” aplicado na linha de comando do Windows XP.

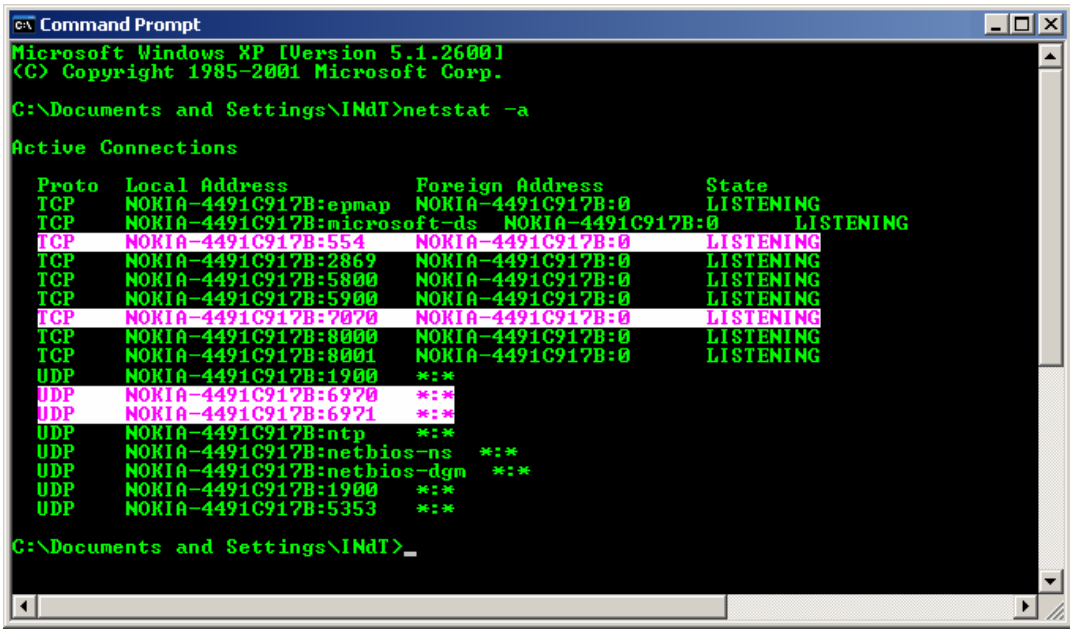


Figura 4.6 – Situação das portas RTP/RTSP na interface do PC.

Conforme mostrado na figura 4.6 foi constatado que as portas RTP/RTSP estavam disponíveis para acesso dos clientes ao PC. Esta constatação levou a suspeita de que o provedor de acesso estava barrando estas portas para acesso externo. Portanto para verificar esta suspeita foi feito uma varredura das portas TCP/UDP no IP do Link banda larga do Virtua, utilizando um site que fornece serviços de varredura de portas (http://www.t1shopper.com/tools/port-scanner/), e foi comprovado que as portas estavam bloqueadas no circuito Banda Larga da Vírtua.

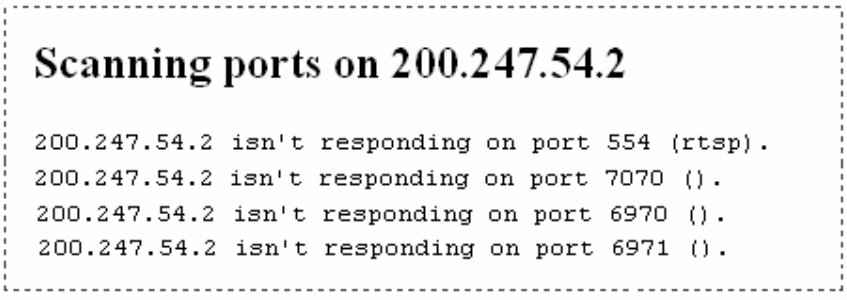


Figura 4.7 – Resultado da varredura das portas RTP/RTSP no circuito de Banda larga.

Como pode ser visto na figura 4.7 as portas RTP/RTSP não respondiam a solicitações de varredura, para garantir o resultado do teste foi feita uma consulta à equipe de suporte técnico do provedor de Banda Larga, que informou que estas portas e outras portas TCP/UDP e todo tráfego com direção em sentido entrante ao PC conectado ao circuito de banda larga do Vírtua estavam sendo bloqueados pelo provedor. Isto com o objetivo de evitar o uso comercial do circuito visto que o contrato do mesmo se destinava para uso doméstico. Para sanar este problema seria necessário, alterar o circuito para a modalidade comercial, mediante um aumento no custo mensal do serviço pelo prazo mínimo de 1 ano.

Para solucionar este problema o teste foi transferido para outra localidade onde havia disponibilidade de um circuito banda larga de 600Kbps com IP fixo e sem nenhuma restrição do provedor. Esta modalidade de circuito é destinada ao uso por empresas, ou aplicações específicas.

A topologia utilizada para o teste foi a mesma da figura 4.4 trocando apenas o circuito de internet doméstico de 2Mbps da Vírtua, por um circuito comercial de 600Kbps da LinkExpress.

Após esta mudança foram executados vários testes de conexão com diferentes bandas de streaming entre 35Kbps e 110Kbps e todos obtiveram sucesso na exibição da streaming utilizando os recursos da rede GSM/EGPRS para o acesso do dispositivo móvel.

Em função do problema encontrado com o link doméstico, foi idealizado um terceiro teste para a aplicação, onde o servidor de streaming estivesse localizado em um lugar diferente do software de geração da streaming. O objetivo deste teste seria demonstrar a possibilidade de existir um “serviço” de servidores de streaming na Internet, prontos para receber as streamings de uma ou mais aplicações do Vigia Móvel pela Internet e deste modo proporcionar aos usuários a capacidade de visualizar estas imagens, resolvendo assim o problema de bloqueio de portas apresentado pelos circuitos domésticos dos provedores de internet.

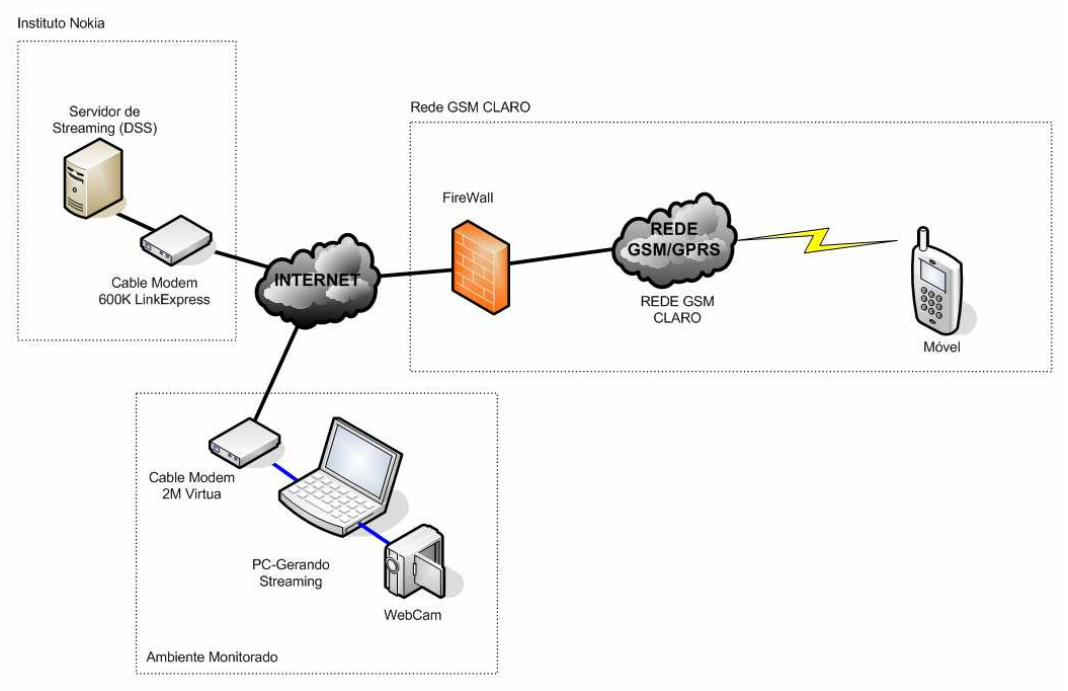


Figura 4.8 – Topologia utilizado no teste utilizando servidor de streaming localizado distante da geração de imagens.

A figura 4.8 mostra a topologia utilizada no teste com o servidor localizado distante da geração imagens. Neste teste foi possível acessar o servidor de streamings com o dispositivo móvel, e o acesso às imagens se mostrou igual ao encontrado nas configurações anteriores.

As topologias testadas usando acesso pela rede GSM foram testadas também, utilizando conexão Wi-Fi, e todos os acesso foram bem sucedidos, utilizando larguras de banda entre 35kbps e 110Kbps para a geração de streamings

É importante ressaltar que os testes pela rede GSM foram feitos utilizando a rede da operadora CLARO. Esta operadora exige que os usuários que utilizam aplicações de streaming de vídeo, utilizem uma configuração específica para o ponto de acesso do dispositivo móvel. Para estas aplicações a operadora faz uso de um ponto de acesso chamado “streaming.claro.com.br”, nas configurações de APN (Access Point Name do aparelho). É provável que esta APN utilize políticas de QoS que privilegiam a utilização de streaming pela

rede GPRS/EGPRS da operadora, mas por motivos de falta de informações por parte da operadora não é possível comprovar este fato.

Foram executados testes utilizando outras APNs (exemplo: claro.com.br) e foi detectado perda de conexão e falha da transmissão das streamings, o que caracteriza que a APN *streaming.claro.com.br* possui configurações de rede apropriadas para o uso de streaming de vídeo.

É importante ressaltar que cada operadora pode ter sua própria configuração de APN e rede GPRS/EGPRS, incluindo configurações e políticas específicas para este tipo de aplicação. E estes dados devem estar disponíveis para os usuários em sites e documentos das operadoras, relativos a utilização de serviços de dados.

Figura 4.9 – Escolha e configuração do ponto de acesso para utilização de streamings de vídeo na rede da operadora CLARO, utilizando o dispositivo móvel NOKIA E61.

A figura 4.9 mostra as telas de configurações para ponto de acesso em um aparelho Nokia E61, nela podemos ver a string que caracteriza o ponto de acesso (streming.claro.com.br).

4.3 ATRASO ENTRE IMAGENS GERADAS E IMAGENS RECEBIDAS

Durante os testes para validar a as topologias utilizadas pelo sistema foi constatado que havia um atraso entre as imagens geradas e as imagens recebidas pelo móvel. Conforme exposto no capitulo 2 seção 2.2.1, as aplicações de streaming de vídeo possuem alta sensibilidade a atrasos. Estes atrasos resultam do tempo gasto pelo codificador para gerar a streaming somado ao tempo gasto pelos pacotes UDP, até chegarem ao destino.

Para realizar as medições de tempo foi usado um cronômetro, que era disparado ao se gerar um movimento em frente a WebCam. Quando este movimento era percebido no dispositivo móvel o cronômetro era parado. Com esta rotina foi possível medir o atraso da streaming em relação a imagem da WebCam para todas as bandas sugeridas no projeto.

Tabela 4.1 – Tabela comparativa dos atrasos da streaming em diferentes bandas

	ATRASO (s)	
BANDA	Low	High
35Kbps	20	16
50Kbps	20	18
70Kbps	18	16
110Kbps	18	15

OBS: Low e High são campos de configuração de complexidade do codec MPEG4

Como pode ser visto na tabela 4.1 acima, os atrasos não têm grande, variação em função da banda utilizada, a tabela mostra os tempos medidos para as diferentes bandas, a latência das redes tem influência neste valor.

Para verificar a influencia da codificação no atraso das streamings, foi alterado o campo “COMPLEXIDADE do CODEC”, dos arquivos XML de configuração do HMPROD de “HIGH” para “LOW” e pode-se constatar que ocasionava um aumento no atraso. A etapa de codificação mostra ter influencia neste atraso.

Depois dos testes todos os arquivos XML de configuração do codec foram configurados para o valor HIGH, que apresentava menos atraso.

4.4 ESTIMATIVAS DE CUSTO DO SISTEMA

O sistema Vigia Móvel foi desenvolvido utilizando dispositivos facilmente encontrados em computadores pessoais, e de baixo custo.

Para que o sistema possa ser utilizado tanto em redes GSM,como em redes Wi-Fi, são necessários os seguintes equipamentos

- Computador pessoal (Desktop) rodando Windows XP.
- WebCam comum com resolução de no mínimo 320x240 Pixels e 30fps
- Ponto de Acesso ou roteador Wireless
- Acesso a internet banda Larga (se for usado em redes GSM/EGPRS) com no mínimo 300Kbps
- Dispositivo Móvel com suporte a Wi-Fi, e/ou redes GSM, e/ou WCDMA.

A tabela 4.2 mostra os preços médios encontrados no mercado para os dispositivos utilizados no projeto do Vigia Móvel.

Tabela 4.2 – Tabela de preços médios para os componentes usados no sistema.

Equipamentos	Preço *
Computador Pentium 4 3.0GHz 512MB 80GB DVD-RW NOVA	R$ 800,00
Webcam DSB-C120 - D-Link	R$ 49,90
Ponto de acesso Wi-Fi	R$ 190,00
Dispositivo Móvel NOKIA E61	R$ 740,00
Banda larga (Turbo 250 - Mensal)	R$ 59,90
Total	R$ 1.839,80

*Preços Médios encontrados no mercado em pesquisa feita em lojas pela Internet

Deve ser levado em consideração que o sistema não exige a utilização de um computador reservado, e portanto, os custos citados podem ser compartilhados com os custos do sistema de computação pessoal, residencial ou corporativo.

Para usuários de vídeo conferência na Web, os equipamentos citados acima fazem parte do sistema usual de computação, e portanto o custo seria apenas o de instalação e configuração do sistema.

Capítulo 5 – CONCLUSÃO

O Estudo apresentado neste projeto tem grande importância na utilização de novas tecnologias para o desenvolvimento de sistemas cada vez mais confiáveis de segurança, pública e privada. A utilização de streaming de vídeo em dispositivos móveis, como ferramenta de vigilância, oferece uma maior flexibilidade aos sistemas de segurança.

A maior parte dos sistemas de vigilância de ambientes é baseada em técnicas de circuitos fechados de TV, sem nenhuma opção de mobilidade para as equipes de vigilância, e para os proprietários. O uso de streaming em dispositivos móveis oferece uma alternativa de melhorar a eficiência na prevenção de perdas materiais e pessoais, por que possibilita uma maior interação do usuário com o ambiente monitorado, devido à praticidade do uso dos dispositivos móveis.

O protótipo apresentado neste projeto foi capaz de demonstrar que com a utilização de dispositivos de hardware, facilmente encontrados em computadores pessoais, como câmeras USB, e conexão banda larga com a Internet, aliadas à sistemas de software que integram tecnologias como streaming de vídeo, detecção de movimentos, e envio de notificações, é viável desenvolver aplicações de vigilância confiáveis e de fácil acesso.

O projeto também demonstra sucesso na utilização de diversas tecnologias de acesso aos dispositivos móveis, como GSM, WCDMA e Wi-Fi, comprovando a possibilidade de convergência entre sistemas que suportam conexões de acesso TCP/IP.

Durante o projeto foram encontradas diversas dificuldades para implementação do sistema. Primeiramente foi escolhido a linguagem Flash 8 para o desenvolvimento da central de vigilância, mas durante o desenvolvimento do código foi constatado que os recursos necessários para controlar adequadamente os Codecs não estavam disponíveis na linguagem, exigindo que o código fosse reescrito em C++, fato este que impactou diretamente no cronograma do projeto.

Da mesma forma foram encontrados vários problemas para estabelecer a comunicação entre os dispositivos GSM/EGPRS e o servidor de streaming, em função disto, foi tomada a decisão de utilizar-se a tecnologia Wi-Fi, como um cenário de testes altamente controlável, criando-se assim um modelo de conexão móvel-servidor, que após ser colocado em funcionamento de maneira satisfatória, foi transferido para a rede GSM/EGPRS, o que facilitou a solução dos problemas de conexão apresentados no início dos testes.

Em projetos futuros a utilização de câmeras IP, que integram a função de geração de streaming, tende a facilitar o desenvolvimento de sistemas deste gênero, pois repassam para o dispositivo de captura a função de codificação das imagens. A utilização de algoritmos mais complexos para a detecção de movimentos e, possivelmente, até detecção de “face”, poderá trazer uma diminuição significativa de falsas notificações de intrusão, pois podem selecionar e distinguir, indivíduos e objetos. Há ainda a possibilidade de criar serviços de fornecimento de streaming na internet, onde a aplicação apresentada faria parte de um sistema monitoração compartilhado por vários usuários na internet, podendo-se criar grupos com acesso a determinadas streamings, através de autenticação de usuários para o recebimento das imagens.

E como foi citado anteriormente o sistema pode ser adaptado a sistemas de vigilância e controle de crianças, animais de estimação, pessoas enfermas, deficientes reconhecimentos de face, acompanhamento de objetos em movimento, câmeras térmicas, infravermelhas, e outras soluções que necessitem de mobilidade no uso de streaming de vídeo e até áudio.

BIBLIOGRAFIA

ALCATEL TD, OMC-R Administration Guide B8 – Evolium 2000.

ALCATEL TD, OMC-R Command Mode B8 – Evolium 2000.

ALCATEL UNIVERSITY, Maintenance of The Evolium BTS B4/B5/B6/B7 – São Paulo: Alcatel ,2000.

Barbatana, Fabrício Eras Manzi , FLASH LITE 2 - Crie Aplicativos e Games para Celulares, 2006

DEITEL, H. M. et al., Perl Como Programar. Tradução Carlos Arthur Lang Lisboa. Porto Alegre: Bookman, 2002.

FLUCKIGER, François , Understanding networked multimedia: applications and technology. Londres: Prentice Hall, 1995.

GSM System Survey - Ericsson Radio Systems – 1998.

Jamsa, Kris e Klander, Lars , Programando em C/C++ - A Bíblia, 1999

Liu, Chunlei , Multimedia Over IP: RSVP, RTP, RTCP, RTSP, 2000

Medeiros, Fernando Ventura , FLASH PROFESSIONAL 8 - Fundamentos e Aplicações, 2005

Mateus, César Augusto , C++ BUILDER 5 - Guia Prático, 2000
Jamsa, Kris , Aprendendo C++, 1999

NOKIA , Video And Streaming In Nokia Devices v3.0, 2005.

Ross, K. W. e Kurose, J F , Computer Networking, A Top Down Approach Featuring the Internet, 3rd edition, 2004

SCHULZRINNE, H et al , RTP: a transport protocol for real-time applications. [S.l], jan. 1996.

TECNOLOGY GUIDES: White papers for IT professionals. Real Time video on the Internet. [S.l], 2001. Disponível em: http://www.techguide.com/titles/rtvideo.shtml

APÊNDICE – CODIGO FONTE

CODIGO FONTE DA CENTRAL DE VIGILÂNCIA

Basicamente o a Central de Vigilância possui os seguintes arquivos:

VigiaMovelB3.cpp
VigiaMovelBuild3.cpp
VigiaMovelBuild3.h
VigiaMovelBuild3.dfm
sendsms.pl
emaildados.txt
msglog.log
VigiaMovel-35k.xml
VigiaMovel-50k.xml
VigiaMovel-70k.xml
VigiaMovel-110k.xml
VigiaMovelExt-35k.xml
VigiaMovelExt-50k.xml
VigiaMovelExt-70k.xml
VigiaMovelExt-110k.xml

CODIGO FONTE DO VIGILANTE

Basicamente o Vigilante possui o seguinte arquivo:

PlayerE61-E.fla

CODIGO FONTE DA CENTRAL DE VIGILÂNCIA

VigiaMovelB3.cpp

```
//$$---- EXE CPP ----
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("VigiaMovelBuild3.cpp", Form1);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
      try
      {
            Application->Initialize();
            Application->CreateForm(__classid(TForm1), &Form1);
            Application->Run();
      }
      catch (Exception &exception)
      {
            Application->ShowException(&exception);
      }
      catch (...)
      {
            try
            {
                  throw Exception("");
            }
            catch (Exception &exception)
            {
                  Application->ShowException(&exception);
            }
      }
      return 0;
}
//---------------------------------------------------------------------------
```

VigiaMovelBuild3.cpp

```
//$$---- Form CPP ----
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include <iostream.h>
#include <fstream.h>
#include <time.h>

#include "VigiaMovelBuild3.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "VLCommonDisplay"
#pragma link "VLCommonLogger"
#pragma link "VLDSCapture"
#pragma link "VLDSCommonLogger"
#pragma link "VLDSImageDisplay"
#pragma link "VLDSVideoLogger"
#pragma link "VLMotionDetect"
#pragma link "VLDSVideoCompressor"
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------------

//####### Variáveis Globais ########

static int Estado = 1;
int i = 0, nDisp, gatilho;
AnsiString   iAno,iMes,iDia,iHora,iMinuto,DataFormatada,ArqComp;
AnsiString nomeDisp;
int LbTxL;
TStringList *Lista1 = new TStringList;

//Variáveis Globais, para Envio de Notificações
time_t hora_envio, hora_atual;
double dif;



//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormShow(TObject *Sender)
{
  if (Estado)
   {
       Estado = 0;
       VLDSCapture1->VideoCaptureDevice->GetDeviceList(Lista1);
       Cbox_Seleciona_Disp->Items->AddStrings(Lista1);
       Btn_Parar->Enabled = false;
       Log_Memo->Text = "Selecionar um Dispositivo de Vídeo!!!";
       VLMotionDetect1->Enabled = false;
```

```
VLDSVideoLogger1->Enabled = false;
VLDSVideoLogger1->VideoCompression->Enabled = false;

CheckBoxRTP->Enabled = false;
RadioButtonServInt->Checked = true;
TimerProgBar->Enabled = false;
LabelTx->Visible = false;

//Seta o Número de Células Por imagem

//VLMotionDetect1->MotionGrid->Cols = 2;
//VLMotionDetect1->MotionGrid->Rows = 2;

//Seta Valor Inicial de Detecção de Movimento para 7

VLMotionDetect1->MotionGrid->Items[0][0] = 8;
VLMotionDetect1->MotionGrid->Items[0][1] = 8;
VLMotionDetect1->MotionGrid->Items[1][0] = 8;
VLMotionDetect1->MotionGrid->Items[1][1] = 8;

//Atualizar Valores na Janela de Movimento ao Abrir aplicação

LCell00_Valor->Caption = VLMotionDetect1->MotionGrid->Items[0][0];
LCell01_Valor->Caption = VLMotionDetect1->MotionGrid->Items[0][1];
LCell10_Valor->Caption = VLMotionDetect1->MotionGrid->Items[1][0];
LCell11_Valor->Caption = VLMotionDetect1->MotionGrid->Items[1][1];

//Verificando dados no arquivo "emaildados.txt"

char user[128], pass[128], conta[128], MP[128], MS[128], EA[128];
AnsiString user2, pass2, conta2, MP2, MS2, EA2;

ifstream arq_entrada("C:\\Users\\VigiaMovel\\emaildados.txt");

arq_entrada.getline(user, sizeof(user));
arq_entrada.getline(pass, sizeof(pass));
arq_entrada.getline(conta, sizeof(conta));
arq_entrada.getline(MP, sizeof(MP));
arq_entrada.getline(MS, sizeof(MS));
arq_entrada.getline(EA, sizeof(EA));

arq_entrada.close();

user2=user;
pass2=pass;
conta2=conta;
MP2=MP;
MS2=MS;
EA2=EA;

conta2 = conta2.SubString(7,conta2.Length());
MP2 = MP2.SubString(4,MP2.Length());
MS2 = MS2.SubString(4,MS2.Length());
EA2 = EA2.SubString(4,EA2.Length());

Lb_Conta_Atual->Caption = conta2;
Lb_Mp_Atual->Caption = MP2;
Lb_Ms_Atual->Caption = MS2;
Lb_Ea_Atual->Caption = EA2;
```

```
       //Atualiza Valores na Janela...

       MP2 = MP2.SubString(1,10);
       MS2 = MS2.SubString(1,10);

       Ed_MSISDN1->Text = MP2;
       Ed_MSISDN2->Text = MS2;
       Ed_Email_Adic->Text = EA2;

       //Variavel que inicializa Tempo entre Notificações

       gatilho = 1;


   }
}

//---------------------------------------------------------------------------
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
  if (CheckBoxRTP->Checked)
  {
   HWND hwndHMPROD;
   if ((hwndHMPROD = FindWindow(NULL,
         "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe")) != NULL)
   {
        //DWORD -> 32-bit unsigned integer
        DWORD ID;
        //UINT -> Unsigned INT
        UINT ExitCode = 1;
        HANDLE hdl;

        GetWindowThreadProcessId(hwndHMPROD, &ID);
        hdl = OpenProcess(PROCESS_ALL_ACCESS,false, ID);

        TerminateProcess(hdl, ExitCode);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: DESATIVADO!!!";
        Log_Memo->Lines->Add(AnsiString ("TERMINANDO o ENVIO de PACOTES RTP a 110Kbps!!!" ));
        LbRtpStatus->Font->Color = clRed;
        LbRtpStatus->Caption = "Streaming de Video DESABILITADA!!!";
   }
   else
   {
        ShowMessage("Inpossivel Interromper o CODEC hmprod.exe, favor terminar manualmente!!");
   }
  }
}

//---------------------------------------------------------------------------


void __fastcall TForm1::Cbox_Seleciona_DispChange(TObject *Sender)
{
  nomeDisp=Cbox_Seleciona_Disp->Items->Strings[Cbox_Seleciona_Disp->ItemIndex];
  VLDSCapture1->VideoCaptureDevice->DeviceName = nomeDisp;
  VLDSCapture1->Open();
  VLDSCapture1->VideoSize->Width = 320;
  VLDSCapture1->VideoSize->Height = 240;
  Log_Memo->Font->Color = clBlue;
  Log_Memo->Lines->Add(AnsiString ("Selecionado =>")+ nomeDisp);
```

```
  //ShowMessage("Nome Do Dispositivo" + nomeDisp);
}

//---------------------------------------------------------------------------

void __fastcall TForm1::PropriedadesdaWebCam1Click(TObject *Sender)
{
  VLDSCapture1->ShowVideoDialog(cdVideoCapture);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::MododeVdeo1Click(TObject *Sender)
{
  VLDSCapture1->ShowVideoDialog(cdVideoCapturePin);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Sair1Click(TObject *Sender)
{
  if (CheckBoxRTP->Checked)
  {
   HWND hwndHMPROD;
   if ((hwndHMPROD = FindWindow(NULL,
         "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe")) != NULL)
   {
        //DWORD -> 32-bit unsigned integer
        DWORD ID;
        //UINT -> Unsigned INT
        UINT ExitCode = 1;
        HANDLE hdl;

        GetWindowThreadProcessId(hwndHMPROD, &ID);
        hdl = OpenProcess(PROCESS_ALL_ACCESS,false, ID);

        TerminateProcess(hdl, ExitCode);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: DESATIVADO!!!";
        Log_Memo->Lines->Add(AnsiString ("TERMINANDO o ENVIO de PACOTES RTP a 110Kbps!!!" ));
        LbRtpStatus->Font->Color = clRed;
        LbRtpStatus->Caption = "Streaming de Video DESABILITADA!!!";
   }
   else
   {
        ShowMessage("Inpossivel Interromper o CODEC hmprod.exe, favor terminar manualmente!!");
   }
  }

  Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Btn_IniciarClick(TObject *Sender)
{
  Btn_Iniciar->Enabled = false;
  Btn_Parar->Enabled = true;

  //nomeDisp = Cbox_Seleciona_Disp->Items->Strings[Cbox_Seleciona_Disp->ItemIndex];
  //VLDSCapture1->VideoCaptureDevice->DeviceName = nomeDisp;
  //ShowMessage("Nome do Dispositivo=" + nomeDisp);

  VLDSVideoLogger1->Enabled = True;
  VLDSCapture1->Start();
```

```
  Log_Memo->Font->Color = clGreen;
  Log_Memo->Lines->Add(AnsiString ("Iniciado Captura =>" )+ nomeDisp);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Btn_PararClick(TObject *Sender)
{
  Btn_Iniciar->Enabled = true;
  Btn_Parar->Enabled = false;
  VLDSVideoLogger1->Enabled = False;
  VLDSCapture1->Stop();
  VLDSImageDisplay1->Clear();
  Log_Memo->Font->Color = clRed;
  Log_Memo->Lines->Add(AnsiString ("Interrompido Captura =>" )+ nomeDisp);
  StatusBar1->Panels->Items[2]->Text = " SEM MOVIMENTO!!!";
}
//---------------------------------------------------------------------------

void __fastcall TForm1::VLMotionDetect1MotionDetect(TObject *Sender,
        int AMaxValue, TPoint &ACell)
{
  //Atualiza Log de Movimentos!!!

  Log_Memo->Lines->Add(AnsiString(DateTimeToStr(Now()) +
                                  " => Movimento na Célula: ") + ACell.x + "," + ACell.y);
   StatusBar1->Panels->Items[2]->Text = " EVENTO: " + AnsiString(DateTimeToStr(Now()) +
                                  " => Movimento na Célula: ") + ACell.x + "," + ACell.y;

//((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((

  //Envio de  Notificação SMS e EMAIL...

  if (gatilho)
   {
      gatilho=0;

      //Enviando Mensagem

      ShellExecute(Handle,"open", "C:\\cygwin\\bin\\perl", "C:\\Users\\VigiaMovel\\sendsms.pl",
NULL, SW_HIDE);
      hora_envio = time(NULL);
      Log_Memo->Lines->Add(AnsiString(DateTimeToStr(Now()) + " => NOTIFICAÇÃO SMS ENVIADA!...
"));
   }
  if (gatilho == 0)
   {
       hora_atual = time(NULL);
       dif = difftime(hora_atual, hora_envio);
       if(dif > 90)
        {
         ShellExecute(Handle,"open", "C:\\cygwin\\bin\\perl",
"C:\\Users\\VigiaMovel\\sendsms.pl", NULL, SW_HIDE);
         hora_envio = time(NULL);
         Log_Memo->Lines->Add(AnsiString(DateTimeToStr(Now()) + " => NOTIFICAÇÃO SMS
ENVIADA!...(+)"));
        }
   }



}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  if (CheckBox1->Checked)
        {
            VLMotionDetect1->Enabled = true;
            StatusBar1->Panels->Items[1]->Text = " STATUS do Detector: ATIVADO!!!";
            Log_Memo->Lines->Add(AnsiString ("Detector de Movimentos ATIVADO!!!" ));
            Lb_aviso_de_detec1->Font->Color = clGreen;
            Lb_aviso_de_detec1->Caption = "Detecção de Movimentos ATIVADA!!!";
        }
  else if (!CheckBox1->Checked)
        {
            VLMotionDetect1->Enabled = false;
            VLDSImageDisplay2->Clear();
            StatusBar1->Panels->Items[1]->Text = " STATUS do Detector: DESATIVADO!!!";
            StatusBar1->Panels->Items[2]->Text = " SEM MOVIMENTO!!!";
            Log_Memo->Lines->Add(AnsiString ("Detector de Movimentos DESATIVADO!!!" ));
            Lb_aviso_de_detec1->Font->Color = clRed;
            Lb_aviso_de_detec1->Caption = "Detecção de Movimentos DESATIVADA!!!";

        }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Btn_Atualiza_SensClick(TObject *Sender)
{
   LCell00_Valor->Caption = VLMotionDetect1->MotionGrid->Items[0][0];
   LCell01_Valor->Caption = VLMotionDetect1->MotionGrid->Items[0][1];
   LCell10_Valor->Caption = VLMotionDetect1->MotionGrid->Items[1][0];
   LCell11_Valor->Caption = VLMotionDetect1->MotionGrid->Items[1][1];
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
   int ce00, ce01, ce10, ce11;

   //Converte String do EditCell para Inteiro a ser aplicado no MotionGrid
   ce00 = StrToInt(EditCell00->Text[1]);
   ce01 = StrToInt(EditCell01->Text[1]);
   ce10 = StrToInt(EditCell10->Text[1]);
   ce11 = StrToInt(EditCell11->Text[1]);
   VLMotionDetect1->MotionGrid->Items[0][0] = ce00;
   VLMotionDetect1->MotionGrid->Items[0][1] = ce01;
   VLMotionDetect1->MotionGrid->Items[1][0] = ce10;
   VLMotionDetect1->MotionGrid->Items[1][1] = ce11;

   //Atualiza Valores de Sensibilidade na Janela
   LCell00_Valor->Caption = VLMotionDetect1->MotionGrid->Items[0][0];
   LCell01_Valor->Caption = VLMotionDetect1->MotionGrid->Items[0][1];
   LCell10_Valor->Caption = VLMotionDetect1->MotionGrid->Items[1][0];
   LCell11_Valor->Caption = VLMotionDetect1->MotionGrid->Items[1][1];

   }
//---------------------------------------------------------------------------

void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  char buf[3];
```

```
  StatusBar1->Panels->Items[0]->Text = " Numero da Pagina: " +
                    AnsiString(itoa(PageControl1->ActivePage->PageIndex, buf,10));

}
//---------------------------------------------------------------------------

void __fastcall TForm1::CheckBox2Click(TObject *Sender)
{
  if (CheckBox2->Checked)
  {
   iAno=Now().FormatString("yyyy");
   iMes=Now().FormatString("mm");
   iDia=Now().FormatString("dd");
   iHora=Now().FormatString("hh");
   iMinuto=Now().FormatString("nn");
   DataFormatada= iDia+iMes+iAno+iHora+iMinuto;
   ArqComp = "C:\\Users\\VigiaMovel\\MovDet"+DataFormatada+".mpg";
   VLDSVideoLogger1->FileName = ArqComp;
   VLDSVideoLogger1->VideoCompression->Enabled = true;
   VLDSVideoLogger1->Enabled = true;
   //ShowMessage(ArqComp);
   Log_Memo->Lines->Add(AnsiString ("Inicio de Gravação dos Movimentos!!!" ));
   Log_Memo->Lines->Add(AnsiString ("Nome do Arq =>" )+ ArqComp);
   Lb_aviso_de_mov1->Font->Color = clGreen;
   Lb_aviso_de_mov1->Caption = "Movimentos estão sendo Gravados!!!";


   }
   else if (!CheckBox2->Checked)
   {
      VLDSVideoLogger1->Enabled = false;
      VLDSVideoLogger1->VideoCompression->Enabled = false;
      Log_Memo->Lines->Add(AnsiString ("Fim de Gravação dos Movimentos!!!" ));
      Lb_aviso_de_mov1->Font->Color = clRed;
      Lb_aviso_de_mov1->Caption = "Movimentos não estão sendo Gravados!!!";
   }
}

//---------------------------------------------------------------------------

void __fastcall TForm1::CheckBox35kClick(TObject *Sender)
{
   if (CheckBox35k->Checked)
       {
             CheckBox50k->Enabled = false;
             CheckBox70k->Enabled = false;
             CheckBox110k->Enabled = false;
             CheckBoxRTP->Enabled = true;
             Log_Memo->Lines->Add(AnsiString ("Escolhido Streaming de 35Kbps!!!" ));
             StatusBar2->Panels->Items[1]->Text = " Largura de BANDA do Streaming = 35Kbps!!!";

       }
   else if (!CheckBox35k->Checked)
       {
             CheckBox50k->Enabled = true;
             CheckBox70k->Enabled = true;
             CheckBox110k->Enabled = true;
             CheckBoxRTP->Enabled = false;
             Log_Memo->Lines->Add(AnsiString ("Desabilitado Streaming de 35Kbps!!!" ));
             StatusBar2->Panels->Items[1]->Text = " Largura de BANDA do Streaming = (não
```

```
selecionada!!!)";
       }


}

//---------------------------------------------------------------------------

void __fastcall TForm1::CheckBox50kClick(TObject *Sender)
{
   if (CheckBox50k->Checked)
       {
             CheckBox35k->Enabled = false;
             CheckBox70k->Enabled = false;
             CheckBox110k->Enabled = false;
             CheckBoxRTP->Enabled = true;
             Log_Memo->Lines->Add(AnsiString ("Escolhido Streaming de 50Kbps!!!" ));
             StatusBar2->Panels->Items[1]->Text = " Largura de BANDA do Streaming = 50Kbps!!!";

       }
   else if (!CheckBox50k->Checked)
       {
             CheckBox35k->Enabled = true;
             CheckBox70k->Enabled = true;
             CheckBox110k->Enabled = true;
             CheckBoxRTP->Enabled = false;
             Log_Memo->Lines->Add(AnsiString ("Desabilitado Streaming de 50Kbps!!!" ));
             StatusBar2->Panels->Items[1]->Text = " Largura de BANDA do Streaming = (não
selecionada!!!)";
       }

}
//---------------------------------------------------------------------------


void __fastcall TForm1::CheckBox70kClick(TObject *Sender)
{
   if (CheckBox70k->Checked)
       {
             CheckBox35k->Enabled = false;
             CheckBox50k->Enabled = false;
             CheckBox110k->Enabled = false;
             CheckBoxRTP->Enabled = true;
             Log_Memo->Lines->Add(AnsiString ("Escolhido Streaming de 70Kbps!!!" ));
             StatusBar2->Panels->Items[1]->Text = " Largura de BANDA do Streaming = 70Kbps!!!";


       }
   else if (!CheckBox70k->Checked)
       {
             CheckBox35k->Enabled = true;
             CheckBox50k->Enabled = true;
             CheckBox110k->Enabled = true;
             CheckBoxRTP->Enabled = false;
             Log_Memo->Lines->Add(AnsiString ("Desabilitado Streaming de 70Kbps!!!" ));
             StatusBar2->Panels->Items[1]->Text = " Largura de BANDA do Streaming = (não
selecionada!!!)";
       }
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::CheckBox110kClick(TObject *Sender)
{
   if (CheckBox110k->Checked)
       {
             CheckBox35k->Enabled = false;
             CheckBox50k->Enabled = false;
             CheckBox70k->Enabled = false;
             CheckBoxRTP->Enabled = true;
             Log_Memo->Lines->Add(AnsiString ("Escolhido Streaming de 110Kbps!!!" ));
             StatusBar2->Panels->Items[1]->Text = " Largura de BANDA do Streaming = 110Kbps!!!";

       }
   else if (!CheckBox50k->Checked)
       {
             CheckBox35k->Enabled = true;
             CheckBox50k->Enabled = true;
             CheckBox70k->Enabled = true;
             CheckBoxRTP->Enabled = false;
             Log_Memo->Lines->Add(AnsiString ("Desabilitado Streaming de 110Kbps!!!" ));
             StatusBar2->Panels->Items[1]->Text = " Largura de BANDA do Streaming = (não
selecionada!!!)";
       }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::CheckBoxRTPClick(TObject *Sender)
{
  TimerProgBar->Enabled = true;
  LabelTx->Visible = true;
  LabelTx->Caption = "TRANSMITINDO.";

  if (CheckBoxRTP->Checked)
  {
      if (CheckBox35k->Checked)
      {
       if (RadioButtonServInt->Checked)
       {
        ShellExecute(Handle,
        "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
        "-j C:\\Users\\VigiaMovel\\VigiaMovel-35k.xml", NULL, SW_SHOWMINNOACTIVE);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: ATIVADO em 35Kbps -
Int 127.0.0.1 !!!";
        Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 35Kbps - Dest 127.0.0.1 !!!"
));
        LbRtpStatus->Font->Color = clGreen;
        LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=35Kbps - Dest 127.0.0.1
!!!";
       }
       else if (RadioButtonServExt->Checked)
       {
        ShellExecute(Handle,
        "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
        "-j C:\\Users\\VigiaMovel\\VigiaMovelExt-35k.xml", NULL, SW_SHOWMINNOACTIVE);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: ATIVADO em 35Kbps -
Ext 200.196.119.105 !!!";
        Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 35Kbps - Dest 200.196.119.105
!!!" ));
        LbRtpStatus->Font->Color = clGreen;
        LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=35Kbps - Dest
200.196.119.105 !!!";
```

```
       }
      }
      else if (CheckBox50k->Checked)
      {
       if (RadioButtonServInt->Checked)
       {
        ShellExecute(Handle,
        "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
        "-j C:\\Users\\VigiaMovel\\VigiaMovel-50k.xml", NULL, SW_SHOWMINNOACTIVE);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: ATIVADO em 50Kbps -
Int 127.0.0.1 !!!";
        Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 50Kbps - Dest 127.0.0.1 !!!"
));
        LbRtpStatus->Font->Color = clGreen;
        LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=50Kbps - Dest 127.0.0.1
!!!";
       }
       else if (RadioButtonServExt->Checked)
       {
        ShellExecute(Handle,
        "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
        "-j C:\\Users\\VigiaMovel\\VigiaMovelExt-50k.xml", NULL, SW_SHOWMINNOACTIVE);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: ATIVADO em 50Kbps -
Ext 200.196.119.105 !!!";
        Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 50Kbps - Dest 200.196.119.105
!!!" ));
        LbRtpStatus->Font->Color = clGreen;
        LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=50Kbps - Dest
200.196.119.105 !!!";
       }
      }
      else if (CheckBox70k->Checked)
      {
       if (RadioButtonServInt->Checked)
       {
        ShellExecute(Handle,
        "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
        "-j C:\\Users\\VigiaMovel\\VigiaMovel-70k.xml", NULL, SW_SHOWMINNOACTIVE);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: ATIVADO em 70Kbps -
Int 127.0.0.1 !!!";
        Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 70Kbps - Dest 127.0.0.1 !!!"
));
        LbRtpStatus->Font->Color = clGreen;
        LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=70Kbps - Dest 127.0.0.1
!!!";
       }
       else if (RadioButtonServExt->Checked)
       {
        ShellExecute(Handle,
        "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
        "-j C:\\Users\\VigiaMovel\\VigiaMovelExt-70k.xml", NULL, SW_SHOWMINNOACTIVE);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: ATIVADO em 70Kbps -
Ext 200.196.119.105 !!!";
        Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 70Kbps - Dest 200.196.119.105
!!!" ));
        LbRtpStatus->Font->Color = clGreen;
        LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=70Kbps - Dest
200.196.119.105 !!!";
       }
      }
      else if (CheckBox110k->Checked)
```

```
      {
       if (RadioButtonServInt->Checked)
       {
        ShellExecute(Handle,
        "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
        "-j C:\\Users\\VigiaMovel\\VigiaMovel-110k.xml", NULL, SW_SHOWMINNOACTIVE);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: ATIVADO em 110Kbps
- Int 127.0.0.1 !!!";
        Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 110Kbps - Dest 127.0.0.1 !!!"
));
        LbRtpStatus->Font->Color = clGreen;
        LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=110Kbps - Dest 127.0.0.1
!!!";
       }
       else if (RadioButtonServExt->Checked)
       {
        ShellExecute(Handle,
        "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
        "-j C:\\Users\\VigiaMovel\\VigiaMovelExt-110k.xml", NULL, SW_SHOWMINNOACTIVE);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: ATIVADO em 110Kbps
- Ext 200.196.119.105 !!!";
        Log_Memo->Lines->Add(AnsiString ("ENVIANDO PACOTES RTP a 110Kbps - Dest 200.196.119.105
!!!" ));
        LbRtpStatus->Font->Color = clGreen;
        LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=110Kbps - Dest
200.196.119.105 !!!";
       }
      }

      else
      {
        ShowMessage("Não foi escolhido a banda da Streaming!! Por Padrão será usado a de
50Kbps!!");
        ShellExecute(Handle,
        "open", "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe",
        "-j C:\\Users\\VigiaMovel\\VigiaMovel-50k.xml", NULL, SW_SHOWMINNOACTIVE);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: ATIVADO em
50Kbps!!!";
        Log_Memo->Lines->Add(AnsiString ("INICIADO o ENVIO de PACOTES RTP a 50Kbps!!!" ));
        LbRtpStatus->Font->Color = clGreen;
        LbRtpStatus->Caption = "Streaming de Video HABILITADO!!! BW=50Kbps!!!";
      }
  }
  else if (!CheckBoxRTP->Checked)
  {
   TimerProgBar->Enabled = false;
   ProgressBarStr->Position = 0;
   LabelTx->Visible = false;

   HWND hwndHMPROD;
   if ((hwndHMPROD = FindWindow(NULL,
         "C:\\Program Files\\Helix\\Helix Mobile Producer 11.0\\hmprod.exe")) != NULL)
   {
        //DWORD -> 32-bit unsigned integer
        DWORD ID;
        //UINT -> Unsigned INT
        UINT ExitCode = 1;
        HANDLE hdl;

        GetWindowThreadProcessId(hwndHMPROD, &ID);
        hdl = OpenProcess(PROCESS_ALL_ACCESS,false, ID);
```

```
        TerminateProcess(hdl, ExitCode);
        StatusBar2->Panels->Items[0]->Text = " STATUS do Streaming de Vídeo: DESATIVADO!!!";
        Log_Memo->Lines->Add(AnsiString ("TERMINANDO o ENVIO de PACOTES RTP!!!" ));
        LbRtpStatus->Font->Color = clRed;
        LbRtpStatus->Caption = "Streaming de Video DESABILITADA!!!";
    }
    else
    {
        ShowMessage("Impossivel Interromper o CODEC hmprod.exe, favor terminar manualmente!!");
    }
   }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{

      char linha1[128], linha2[128], linha3[128], linha4[128], linha5[128], linha6[128];
      AnsiString linha12, linha22, linha32, linha42, linha52, linha62;
      AnsiString Op1, Op2, Dom1, Dom2, NovaL4, NovaL5, NovaL6;

       ifstream ler_arquivo("C:\\Users\\VigiaMovel\\emaildados.txt");

       ler_arquivo.getline(linha1, sizeof(linha1));
       ler_arquivo.getline(linha2, sizeof(linha2));
       ler_arquivo.getline(linha3, sizeof(linha3));
       ler_arquivo.getline(linha4, sizeof(linha4));
       ler_arquivo.getline(linha5, sizeof(linha5));
       ler_arquivo.getline(linha6, sizeof(linha6));

       ler_arquivo.close();

       linha42=linha4; linha52=linha5; linha62=linha6;

       //Define Operadoras

       Op1=Cbox_Operadora1->Items->Strings[Cbox_Operadora1->ItemIndex];
       Op2=Cbox_Operadora2->Items->Strings[Cbox_Operadora2->ItemIndex];

       if (Op1 == "CLARO") { Dom1 = "@clarotorpedo.com.br"; }
       else if (Op1 == "TIM") { Dom1 = "@tim.com.br"; }
       else if (Op1 == "VIVO") { Dom1 = "@vivomail.com.br"; }
       else {  Dom1 = "@operadora.com.br"; }

       if (Op2 == "CLARO") { Dom2 = "@clarotorpedo.com.br"; }
       else if (Op2 == "TIM") { Dom2 = "@tim.com.br"; }
       else if (Op2 == "VIVO") { Dom2 = "@vivomail.com.br"; }
       else {  Dom2 = "@operadora.com.br"; }

       NovaL4 = "MP=" + Ed_MSISDN1->Text + Dom1;
       NovaL5 = "MS=" + Ed_MSISDN2->Text + Dom2;
       NovaL6 = "EA=" + Ed_Email_Adic->Text;

      //Remonta arquivo

      char *L4final, *L5final, *L6final;
      //Converte AnsiString to *char...
      L4final=NovaL4.c_str();
      L5final=NovaL5.c_str();
      L6final=NovaL6.c_str();
```

```
      ofstream escreve_arquivo("C:\\Users\\VigiaMovel\\emaildados.txt");

      escreve_arquivo <<  linha1  << endl;
      escreve_arquivo <<  linha2  << endl;
      escreve_arquivo <<  linha3  << endl;
      escreve_arquivo <<  L4final << endl;
      escreve_arquivo <<  L5final << endl;
      escreve_arquivo <<  L6final << endl;

      escreve_arquivo.close();

      //Verifica arquivo...
      ifstream verifica_arquivo("C:\\Users\\VigiaMovel\\emaildados.txt");

      verifica_arquivo.getline(linha1, sizeof(linha1));
      verifica_arquivo.getline(linha2, sizeof(linha2));
      verifica_arquivo.getline(linha3, sizeof(linha3));
      verifica_arquivo.getline(linha4, sizeof(linha4));
      verifica_arquivo.getline(linha5, sizeof(linha5));
      verifica_arquivo.getline(linha6, sizeof(linha6));

      verifica_arquivo.close();

      linha32=linha3; linha42=linha4; linha52=linha5; linha62=linha6;

      linha32 = linha32.SubString(7,linha32.Length());
      linha42 = linha42.SubString(4,linha42.Length());
      linha52 = linha52.SubString(4,linha52.Length());
      linha62 = linha62.SubString(4,linha62.Length());

      Lb_Conta_Atual->Caption = linha32;
      Lb_Mp_Atual->Caption = linha42;
      Lb_Ms_Atual->Caption = linha52;
      Lb_Ea_Atual->Caption = linha62;

}
//---------------------------------------------------------------------------
void __fastcall TForm1::TimerProgBarTimer(TObject *Sender)
{
   if (ProgressBarStr->Position >= 100)
      {
        ProgressBarStr->Position = 0;
        //ProgressBarStr->Position ++;
      }
   else
      {
        ProgressBarStr->Position = ProgressBarStr->Position + 10;
      }

   LbTxL = LabelTx->Caption.Length();
   if (LbTxL == 20)
      {
        LabelTx->Caption = "TRANSMITINDO.";
      }
   else
      {
         LabelTx->Caption = LabelTx->Caption + ".";
      }
}
//---------------------------------------------------------------------------
```

VigiaMovelBuild3.h

```
//$$---- Form HDR ----
//---------------------------------------------------------------------------

#ifndef VigiaMovelBuild3H
#define VigiaMovelBuild3H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "VLCommonDisplay.h"
#include "VLCommonLogger.h"
#include "VLDSCapture.h"
#include "VLDSCommonLogger.h"
#include "VLDSImageDisplay.h"
#include "VLDSVideoLogger.h"
#include "VLMotionDetect.h"
#include <Buttons.hpp>
#include <ComCtrls.hpp>
#include <Menus.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
      TPageControl *PageControl1;
      TMainMenu *MainMenu1;
      TMenuItem *Menu1;
      TMenuItem *ConfiguraesdeVdeo1;
      TMenuItem *Ajuda1;
      TMenuItem *Sobre1;
      TMenuItem *PropriedadesdaWebCam1;
      TMenuItem *MododeVdeo1;
      TMenuItem *Sair1;
      TTabSheet *TabSheet1;
      TTabSheet *TabSheet2;
      TTabSheet *TabSheet3;
      TTabSheet *TabSheet4;
      TVLDSCapture *VLDSCapture1;
      TVLDSVideoLogger *VLDSVideoLogger1;
      TVLDSImageDisplay *VLDSImageDisplay1;
      TVLDSImageDisplay *VLDSImageDisplay2;
      TVLMotionDetect *VLMotionDetect1;
      TGroupBox *GroupBox1;
      TComboBox *Cbox_Seleciona_Disp;
      TMemo *Log_Memo;
      TBitBtn *Btn_Iniciar;
      TBitBtn *Btn_Parar;
      TGroupBox *GroupBox2;
      TCheckBox *CheckBox1;
      TLabel *Lb_Eventos_Sistema;
      TUpDown *UpDownCell00;
      TEdit *EditCell00;
      TLabel *LCell00_Valor;
      TLabel *LCell00_Sens;
      TBevel *BevelCell00;
      TLabel *LCell00_Quad;
      TLabel *LCell00_Ajuste;
```

```
TLabel *LCell01_Quad;
TLabel *LCell01_Sens;
TLabel *LCell01_Valor;
TLabel *LCell01_Ajuste;
TEdit *EditCell01;
TUpDown *UpDownCell01;
TBevel *BevelCell01;
TBevel *BevelCell10;
TBevel *BevelCell11;
TLabel *LCell10_Quad;
TLabel *LCell10_Sens;
TLabel *LCell10_Valor;
TLabel *LCell10_Ajuste;
TEdit *EditCell10;
TUpDown *UpDownCell10;
TUpDown *UpDownCell11;
TEdit *EditCell11;
TLabel *LCell11_Quad;
TLabel *LCell11_Sens;
TLabel *LCell11_Valor;
TLabel *LCell11_Ajuste;
TButton *Btn_Atualiza_Sens;
TButton *Button2;
TStatusBar *StatusBar1;
TGroupBox *GroupBox3;
TCheckBox *CheckBox2;
TLabel *Lb_aviso_de_mov1;
TLabel *Lb_aviso_de_detec1;
TCheckBox *CheckBoxRTP;
TLabel *LbRtpStatus;
TLabel *Label5;
TEdit *Ed_MSISDN1;
TLabel *Label6;
TComboBox *Cbox_Operadora1;
TComboBox *Cbox_Operadora2;
TEdit *Ed_MSISDN2;
TLabel *Label7;
TLabel *Label8;
TBitBtn *BitBtn2;
TGroupBox *GroupBoxRTP;
TCheckBox *CheckBox50k;
TCheckBox *CheckBox70k;
TCheckBox *CheckBox110k;
TGroupBox *GroupBoxBW;
TStatusBar *StatusBar2;
TEdit *Ed_Email_Adic;
TLabel *Lb_Email_Adic;
TLabel *Lb_Conta;
TBevel *Bevel1;
TLabel *Lb_Mp;
TLabel *Lb_Ms;
TLabel *Lb_Ea;
TLabel *Lb_Conta_Atual;
TLabel *Lb_Mp_Atual;
TLabel *Lb_Ms_Atual;
TLabel *Lb_Ea_Atual;
TLabel *Lb_Contas_Atual;
TCheckBox *CheckBox35k;
TRadioGroup *RadioGroup1;
TRadioButton *RadioButtonServInt;
TRadioButton *RadioButtonServExt;
```

```
      TProgressBar *ProgressBarStr;
      TTimer *TimerProgBar;
      TLabel *LabelTx;
      void __fastcall FormShow(TObject *Sender);
      void __fastcall Cbox_Seleciona_DispChange(TObject *Sender);
      void __fastcall PropriedadesdaWebCam1Click(TObject *Sender);
      void __fastcall MododeVdeo1Click(TObject *Sender);
      void __fastcall Sair1Click(TObject *Sender);
      void __fastcall Btn_IniciarClick(TObject *Sender);
      void __fastcall Btn_PararClick(TObject *Sender);
      void __fastcall VLMotionDetect1MotionDetect(TObject *Sender, int AMaxValue,
          TPoint &ACell);
      void __fastcall CheckBox1Click(TObject *Sender);
      void __fastcall Btn_Atualiza_SensClick(TObject *Sender);
      void __fastcall Button2Click(TObject *Sender);
      void __fastcall PageControl1Change(TObject *Sender);
      void __fastcall CheckBox2Click(TObject *Sender);
      void __fastcall CheckBoxRTPClick(TObject *Sender);
      void __fastcall CheckBox50kClick(TObject *Sender);
      void __fastcall CheckBox70kClick(TObject *Sender);
      void __fastcall CheckBox110kClick(TObject *Sender);
      void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
      void __fastcall BitBtn2Click(TObject *Sender);
      void __fastcall CheckBox35kClick(TObject *Sender);
      void __fastcall TimerProgBarTimer(TObject *Sender);
      //FUnções Definidas por Mim!!!

      //AnsiString __fastcall VerificaEmails (AnsiString File

private:    // User declarations
public:           // User declarations
      __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

VigiaMovelBuild3.dfm

```
object Form1: TForm1
  Left = 0
  Top = 0
  Caption = 'Vigia M'#243'vel (Build3)'
  ClientHeight = 520
  ClientWidth = 763
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  Menu = MainMenu1
  OldCreateOrder = False
  OnClose = FormClose
  OnShow = FormShow
  PixelsPerInch = 96
  TextHeight = 13
  object LabelTx: TLabel
    Left = 354
    Top = 451
    Width = 90
    Height = 13
    Caption = 'TRANSMITINDO.'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clBlue
    Font.Height = -11
    Font.Name = 'Tahoma'
    Font.Style = [fsBold, fsItalic]
    ParentFont = False
    Visible = False
  end
  object PageControl1: TPageControl
    Left = 8
    Top = 24
    Width = 745
    Height = 401
    ActivePage = TabSheet4
    TabOrder = 0
    OnChange = PageControl1Change
    object TabSheet1: TTabSheet
      Caption = 'Sa'#237'da de V'#237'deo Normal'
      ExplicitHeight = 381
      object Lb_Eventos_Sistema: TLabel
        Left = 360
        Top = 149
        Width = 129
        Height = 13
        Caption = 'Log de Eventos do Sistema'
      end
      object Lb_aviso_de_mov1: TLabel
        Left = 360
        Top = 343
        Width = 201
        Height = 13
        Caption = 'Movimentos n'#227'o est'#227'o sendo Gravados!!!'
        Font.Charset = DEFAULT_CHARSET
        Font.Color = clRed
        Font.Height = -11
```

```
  Font.Name = 'Tahoma'
  Font.Style = []
  ParentFont = False
end
object Lb_aviso_de_detec1: TLabel
  Left = 16
  Top = 343
  Width = 198
  Height = 13
  Caption = 'Detec'#231#227'o de Movimentos DESATIVADA!!!'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clRed
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  ParentFont = False
end
object VLDSImageDisplay1: TVLDSImageDisplay
  Left = 16
  Top = 24
  Width = 320
  Height = 240
  Color = clDefault
  ParentColor = False
  InputPin.SourcePin = Form1.VLDSCapture1.OutputPin
  AutoClear = False
  FullScreen = False
  Graph.AdditionalFilters = <>
  Graph.Register = False
end
object GroupBox1: TGroupBox
  Left = 360
  Top = 24
  Width = 361
  Height = 113
  Caption = 'Dispositivo de V'#237'deo (WebCam)'
  TabOrder = 1
end
object Cbox_Seleciona_Disp: TComboBox
  Left = 372
  Top = 48
  Width = 309
  Height = 21
  ItemHeight = 13
  TabOrder = 2
  Text = 'Selecione um Dispositivo de V'#237'deo'
  OnChange = Cbox_Seleciona_DispChange
end
object Log_Memo: TMemo
  Left = 360
  Top = 168
  Width = 361
  Height = 96
  Lines.Strings = (
    'Log_Memo')
  ScrollBars = ssVertical
  TabOrder = 3
end
object Btn_Iniciar: TBitBtn
  Left = 372
  Top = 88
```

```
    Width = 75
    Height = 25
    Caption = '&Iniciar'
    TabOrder = 4
    OnClick = Btn_IniciarClick
    Kind = bkYes
  end
  object Btn_Parar: TBitBtn
    Left = 606
    Top = 88
    Width = 75
    Height = 25
    Caption = '&Parar'
    TabOrder = 5
    OnClick = Btn_PararClick
    Kind = bkNo
  end
  object GroupBox3: TGroupBox
    Left = 360
    Top = 278
    Width = 361
    Height = 59
    Caption = 'Grava'#231#227'o dos Movimentos'
    TabOrder = 6
  end
  object CheckBox2: TCheckBox
    Left = 372
    Top = 304
    Width = 185
    Height = 17
    Caption = 'Gravar Movimentos em Arquivo'
    TabOrder = 7
    OnClick = CheckBox2Click
  end
  object GroupBox2: TGroupBox
    Left = 16
    Top = 278
    Width = 320
    Height = 59
    Caption = 'Detec'#231#227'o de Movimento'
    TabOrder = 9
  end
  object CheckBox1: TCheckBox
    Left = 32
    Top = 304
    Width = 193
    Height = 17
    Caption = 'Detec'#231#227'o de Movimento ATIVADA'
    TabOrder = 8
    OnClick = CheckBox1Click
  end
end
object TabSheet2: TTabSheet
  Caption = 'Detec'#231#227'o de Movimento'
  ImageIndex = 1
  ExplicitHeight = 381
  object LCell00_Sens: TLabel
    Left = 360
    Top = 50
    Width = 65
    Height = 13
```

```
  Caption = 'Sensibilidade:'
end
object LCell00_Valor: TLabel
  Left = 431
  Top = 50
  Width = 6
  Height = 13
  Caption = '0'
end
object BevelCell00: TBevel
  Left = 351
  Top = 43
  Width = 129
  Height = 70
end
object LCell00_Quad: TLabel
  Left = 360
  Top = 24
  Width = 90
  Height = 13
  Caption = 'Quadrante => 0,0'
end
object LCell00_Ajuste: TLabel
  Left = 361
  Top = 83
  Width = 64
  Height = 13
  Caption = 'Ajustar Para:'
end
object LCell01_Quad: TLabel
  Left = 511
  Top = 24
  Width = 90
  Height = 13
  Caption = 'Quadrante => 0,1'
end
object LCell01_Sens: TLabel
  Left = 511
  Top = 50
  Width = 65
  Height = 13
  Caption = 'Sensibilidade:'
end
object LCell01_Valor: TLabel
  Left = 582
  Top = 50
  Width = 6
  Height = 13
  Caption = '0'
end
object LCell01_Ajuste: TLabel
  Left = 511
  Top = 83
  Width = 64
  Height = 13
  Caption = 'Ajustar Para:'
end
object BevelCell01: TBevel
  Left = 504
  Top = 43
  Width = 129
```

```
  Height = 70
end
object BevelCell10: TBevel
  Left = 351
  Top = 152
  Width = 129
  Height = 70
end
object BevelCell11: TBevel
  Left = 504
  Top = 152
  Width = 129
  Height = 70
end
object LCell10_Quad: TLabel
  Left = 360
  Top = 133
  Width = 90
  Height = 13
  Caption = 'Quadrante => 1,0'
end
object LCell10_Sens: TLabel
  Left = 360
  Top = 159
  Width = 65
  Height = 13
  Caption = 'Sensibilidade:'
end
object LCell10_Valor: TLabel
  Left = 432
  Top = 159
  Width = 6
  Height = 13
  Caption = '0'
end
object LCell10_Ajuste: TLabel
  Left = 360
  Top = 192
  Width = 64
  Height = 13
  Caption = 'Ajustar Para:'
end
object LCell11_Quad: TLabel
  Left = 514
  Top = 133
  Width = 87
  Height = 13
  Caption = 'Quadrante =>1,1'
end
object LCell11_Sens: TLabel
  Left = 514
  Top = 159
  Width = 65
  Height = 13
  Caption = 'Sensibilidade:'
end
object LCell11_Valor: TLabel
  Left = 585
  Top = 159
  Width = 6
  Height = 13
```

```
  Caption = '0'
end
object LCell11_Ajuste: TLabel
  Left = 514
  Top = 192
  Width = 64
  Height = 13
  Caption = 'Ajustar Para:'
end
object VLDSImageDisplay2: TVLDSImageDisplay
  Left = 16
  Top = 24
  Width = 320
  Height = 240
  Color = clDefault
  ParentColor = False
  InputPin.SourcePin = Form1.VLMotionDetect1.MotionOutputPin
  AutoClear = False
  FullScreen = False
  Graph.AdditionalFilters = <>
  Graph.Register = False
end
object UpDownCell00: TUpDown
  Left = 449
  Top = 80
  Width = 18
  Height = 21
  Associate = EditCell00
  Max = 9
  Position = 8
  TabOrder = 1
end
object EditCell00: TEdit
  Left = 431
  Top = 80
  Width = 18
  Height = 21
  MaxLength = 1
  TabOrder = 2
  Text = '8'
end
object EditCell01: TEdit
  Left = 581
  Top = 80
  Width = 18
  Height = 21
  MaxLength = 1
  TabOrder = 3
  Text = '8'
end
object UpDownCell01: TUpDown
  Left = 599
  Top = 80
  Width = 18
  Height = 21
  Associate = EditCell01
  Max = 9
  Position = 8
  TabOrder = 4
end
object EditCell10: TEdit
```

```
    Left = 430
    Top = 189
    Width = 18
    Height = 21
    MaxLength = 1
    TabOrder = 5
    Text = '8'
  end
  object UpDownCell10: TUpDown
    Left = 448
    Top = 189
    Width = 16
    Height = 21
    Associate = EditCell10
    Max = 9
    Position = 8
    TabOrder = 6
  end
  object UpDownCell11: TUpDown
    Left = 602
    Top = 189
    Width = 16
    Height = 21
    Associate = EditCell11
    Max = 9
    Position = 8
    TabOrder = 7
  end
  object EditCell11: TEdit
    Left = 584
    Top = 189
    Width = 18
    Height = 21
    MaxLength = 1
    TabOrder = 8
    Text = '8'
  end
  object Btn_Atualiza_Sens: TButton
    Left = 351
    Top = 239
    Width = 75
    Height = 25
    Caption = 'Atualizar'
    TabOrder = 9
    OnClick = Btn_Atualiza_SensClick
  end
  object Button2: TButton
    Left = 503
    Top = 239
    Width = 75
    Height = 25
    Caption = 'Alterar'
    TabOrder = 10
    OnClick = Button2Click
  end
end
object TabSheet3: TTabSheet
  Caption = 'Envio de Notifica'#231#227'o'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
```

```
Font.Name = 'Tahoma'
Font.Style = []
ImageIndex = 2
ParentFont = False
ExplicitHeight = 381
object Bevel1: TBevel
  Left = 264
  Top = 208
  Width = 377
  Height = 105
end
object Label5: TLabel
  Left = 24
  Top = 29
  Width = 164
  Height = 13
  Caption = 'Movel Principal: (Ex: 6190907070)'
end
object Label6: TLabel
  Left = 24
  Top = 85
  Width = 119
  Height = 13
  Caption = 'Operadora GSM/WCDMA'
end
object Label7: TLabel
  Left = 264
  Top = 85
  Width = 119
  Height = 13
  Caption = 'Operadora GSM/WCDMA'
end
object Label8: TLabel
  Left = 264
  Top = 29
  Width = 178
  Height = 13
  Caption = 'Movel Secund'#225'rio: (Ex: 6190907070)'
end
object Lb_Email_Adic: TLabel
  Left = 24
  Top = 189
  Width = 73
  Height = 13
  Caption = 'E-mail Adicional'
end
object Lb_Conta: TLabel
  Left = 280
  Top = 216
  Width = 91
  Height = 13
  Caption = 'Conta de E-mail:'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clBlue
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = [fsBold]
  ParentFont = False
  Transparent = False
end
object Lb_Mp: TLabel
```

```
  Left = 285
  Top = 250
  Width = 88
  Height = 13
  Caption = 'M'#243'vel Principal:'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clBlue
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = [fsBold]
  ParentFont = False
end
object Lb_Ms: TLabel
  Left = 271
  Top = 269
  Width = 103
  Height = 13
  Caption = 'M'#243'vel Secund'#225'rio:'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clBlue
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = [fsBold]
  ParentFont = False
end
object Lb_Ea: TLabel
  Left = 282
  Top = 288
  Width = 92
  Height = 13
  Caption = 'E-mail Adicional:'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clBlue
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = [fsBold]
  ParentFont = False
end
object Lb_Conta_Atual: TLabel
  Left = 379
  Top = 216
  Width = 100
  Height = 13
  Caption = 'nome@gmail.com'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clRed
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = [fsBold]
  ParentFont = False
end
object Lb_Mp_Atual: TLabel
  Left = 379
  Top = 250
  Width = 181
  Height = 13
  Caption = '6190908080@operadora.com.br'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clRed
  Font.Height = -11
  Font.Name = 'Tahoma'
```

```
    Font.Style = [fsBold]
    ParentFont = False
  end
  object Lb_Ms_Atual: TLabel
    Left = 379
    Top = 269
    Width = 181
    Height = 13
    Caption = '6180809090@operadora.com.br'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clRed
    Font.Height = -11
    Font.Name = 'Tahoma'
    Font.Style = [fsBold]
    ParentFont = False
  end
  object Lb_Ea_Atual: TLabel
    Left = 379
    Top = 288
    Width = 161
    Height = 13
    Caption = 'fulanodetal@dominio.com.br'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clRed
    Font.Height = -11
    Font.Name = 'Tahoma'
    Font.Style = [fsBold]
    ParentFont = False
  end
  object Lb_Contas_Atual: TLabel
    Left = 271
    Top = 189
    Width = 258
    Height = 13
    Caption = 'Configura'#231#227'o Atual para envio de Notifica'#231#245'es'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clBlue
    Font.Height = -11
    Font.Name = 'Tahoma'
    Font.Style = [fsBold]
    ParentFont = False
  end
  object Ed_MSISDN1: TEdit
    Left = 24
    Top = 48
    Width = 177
    Height = 21
    TabOrder = 0
    Text = '6199998888'
  end
  object Cbox_Operadora1: TComboBox
    Left = 24
    Top = 104
    Width = 177
    Height = 21
    ItemHeight = 13
    TabOrder = 1
    Text = 'Escolha Operadora'
    Items.Strings = (
      'Escolha Operadora'
      'CLARO'
```

```
      'TIM'
      'VIVO')
  end
  object Cbox_Operadora2: TComboBox
    Left = 264
    Top = 104
    Width = 178
    Height = 21
    ItemHeight = 13
    TabOrder = 2
    Text = 'Escolha Operadora'
    Items.Strings = (
      'Escolha Operadora'
      'CLARO'
      'TIM'
      'VIVO')
  end
  object Ed_MSISDN2: TEdit
    Left = 264
    Top = 48
    Width = 178
    Height = 21
    TabOrder = 3
    Text = '6188889999'
  end
  object BitBtn2: TBitBtn
    Left = 24
    Top = 288
    Width = 137
    Height = 25
    Caption = 'Atualizar cadastro'
    TabOrder = 4
    OnClick = BitBtn2Click
    Kind = bkOK
  end
  object Ed_Email_Adic: TEdit
    Left = 24
    Top = 208
    Width = 177
    Height = 21
    TabOrder = 5
    Text = 'fulano@dominio.com.br'
  end
end
object TabSheet4: TTabSheet
  Caption = 'Streaming de V'#237'deo'
  ImageIndex = 3
  ExplicitHeight = 381
  object LbRtpStatus: TLabel
    Left = 32
    Top = 312
    Width = 183
    Height = 13
    Caption = 'Streaming de Video DESABILITADA!!!!'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clRed
    Font.Height = -11
    Font.Name = 'Tahoma'
    Font.Style = []
    ParentFont = False
  end
```

```
object GroupBoxRTP: TGroupBox
  Left = 408
  Top = 45
  Width = 305
  Height = 84
  Caption = 'Streaming de Video RTP/RTSP'
  TabOrder = 1
end
object CheckBoxRTP: TCheckBox
  Left = 420
  Top = 80
  Width = 265
  Height = 17
  Caption = 'Habilitar Broadcast de Streaming para o Servidor'
  TabOrder = 0
  OnClick = CheckBoxRTPClick
end
object GroupBoxBW: TGroupBox
  Left = 16
  Top = 45
  Width = 369
  Height = 220
  Caption = 'Largura de Banda do Streaming (Servidor Interno)'
  TabOrder = 5
end
object CheckBox50k: TCheckBox
  Left = 32
  Top = 128
  Width = 321
  Height = 17
  Caption = ' 50Kbps (Usar em redes EGPRS, WCDMA e Wi-Fi)'
  TabOrder = 2
  OnClick = CheckBox50kClick
end
object CheckBox70k: TCheckBox
  Left = 32
  Top = 176
  Width = 321
  Height = 17
  Caption = ' 70Kbps (Usar em  redes EGPRS, WCDMA e Wi-Fi)'
  TabOrder = 3
  OnClick = CheckBox70kClick
end
object CheckBox110k: TCheckBox
  Left = 32
  Top = 224
  Width = 321
  Height = 17
  Caption = '110Kbps (Usar em  redes WCDMA e Wi-Fi)'
  TabOrder = 4
  OnClick = CheckBox110kClick
end
object CheckBox35k: TCheckBox
  Left = 32
  Top = 80
  Width = 321
  Height = 17
  Caption = '35Kbps (Usar em Redes HSCSD e GPRS)'
  TabOrder = 6
  OnClick = CheckBox35kClick
end
```

```
    object RadioGroup1: TRadioGroup
      Left = 408
      Top = 145
      Width = 305
      Height = 121
      Caption = 'Servidor de Streaming'
      TabOrder = 7
    end
    object RadioButtonServInt: TRadioButton
      Left = 420
      Top = 176
      Width = 193
      Height = 17
      Caption = 'Servidor Interno (Host 127.0.0.1)'
      TabOrder = 8
    end
    object RadioButtonServExt: TRadioButton
      Left = 420
      Top = 224
      Width = 241
      Height = 17
      Caption = 'Servidor Externo (Internet 200.196.119.105)'
      TabOrder = 9
    end
  end
end
object StatusBar1: TStatusBar
  Left = 0
  Top = 501
  Width = 763
  Height = 19
  Panels = <
    item
      Text = ' Numero da Pagina: 0'
      Width = 110
    end
    item
      Text = ' STATUS do Detector: DESATIVADO!!!'
      Width = 195
    end
    item
      Text = ' SEM MOVIMENTO!!!'
      Width = 200
    end>
  ExplicitTop = 497
end
object StatusBar2: TStatusBar
  Left = 0
  Top = 482
  Width = 763
  Height = 19
  Panels = <
    item
      Text = ' STATUS do Streaming de V'#237'deo: DESATIVADO!!!'
      Width = 400
    end
    item
      Text = ' Largura de BANDA do Streaming = (n'#227'o selecionada!!!)'
      Width = 250
    end>
  ExplicitTop = 478
```

```
end
object ProgressBarStr: TProgressBar
  Left = 28
  Top = 448
  Width = 320
  Height = 16
  TabOrder = 3
end
object MainMenu1: TMainMenu
  Left = 592
  Top = 440
  object Menu1: TMenuItem
    Caption = 'Menu'
    object Sair1: TMenuItem
      Caption = 'Sair'
      OnClick = Sair1Click
    end
  end
  object ConfiguraesdeVdeo1: TMenuItem
    Caption = 'Configura'#231#245'es de V'#237'deo'
    object PropriedadesdaWebCam1: TMenuItem
      Caption = 'Propriedades da WebCam'
      OnClick = PropriedadesdaWebCam1Click
    end
    object MododeVdeo1: TMenuItem
      Caption = 'Modo de V'#237'deo'
      OnClick = MododeVdeo1Click
    end
  end
  object Ajuda1: TMenuItem
    Caption = 'Ajuda'
    object Sobre1: TMenuItem
      Caption = 'Sobre'
    end
  end
end
object VLDSCapture1: TVLDSCapture
  VideoPreview.Enabled = False
  AudioPreview.Enabled = False
  FrameRate.VariableRate = False
  VideoFormat = vfRGB24
  TVTuner.Format = tvfCurrent
  TVTuner.InputType = tviCurrent
  ClosedCaptions.Enabled = False
  ClosedCaptions.Apply = False
  VideoCaptureDevice.AlternativeDevices = <>
  VideoCaptureDevice.Data = (
    ''
    'None')
  AudioCaptureDevice.AlternativeDevices = <>
  Adjustment.Brightness.Mode = amUseCurrent
  Adjustment.Brightness.Value = 0
  Adjustment.Contrast.Mode = amUseCurrent
  Adjustment.Contrast.Value = 0
  Adjustment.Hue.Mode = amUseCurrent
  Adjustment.Hue.Value = 0
  Adjustment.Saturation.Mode = amUseCurrent
  Adjustment.Saturation.Value = 0
  Adjustment.Sharpness.Mode = amUseCurrent
  Adjustment.Sharpness.Value = 0
  Adjustment.Gamma.Mode = amUseCurrent
```

```
  Adjustment.Gamma.Value = 0
  Adjustment.ColorEnable.Mode = amUseCurrent
  Adjustment.ColorEnable.Value = False
  Adjustment.WhiteBalance.Mode = amUseCurrent
  Adjustment.WhiteBalance.Value = 0
  Adjustment.BacklightCompensation.Mode = amUseCurrent
  Adjustment.BacklightCompensation.Value = False
  Adjustment.Gain.Mode = amUseCurrent
  Adjustment.Gain.Value = 0
  Adjustment.Enabled = False
  Graph.AdditionalFilters = <>
  Graph.Register = False
  OutputPin.SinkPins = (
    Form1.VLMotionDetect1.InputPin
    Form1.VLDSImageDisplay1.InputPin)
  Left = 624
  Top = 440
end
object VLDSVideoLogger1: TVLDSVideoLogger
  InputPin.SourcePin = Form1.VLMotionDetect1.OutputPin
  Graph.AdditionalFilters = <
    item
    end>
  Graph.Register = False
  VideoCompression.Enabled = False
  VideoCompression.Compressions = <
    item
      Quality = 75000
      KeyFrameRate = 9999
      WindowSize = 10
      PFramesPerKeyFrame = 0
      Data = (
        '@device:cm:{33D9A760-90C8-11D0-BD43-00A0C911CE86}\div3'
        'DivX ;-) MPEG-4 Low-Motion')
    end>
  AudioCompression.Enabled = False
  AudioCompression.Compressions = <
    item
      Channels = 1
      BitsPerSample = 16
      SamplesPerSec = 8000
      ChannelMask = [spFrontLeft, spFrontRight]
      Data = (

          '@device:dmo:{1F1F4E1A-2252-4063-84BB-EEE75F8856D5}{33D9A761-90C8' +
          '-11D0-BD43-00A0C911CE86}'
        'WMA Voice Encoder DMO')
    end>
  Left = 656
  Top = 440
end
object VLMotionDetect1: TVLMotionDetect
  InputPin.SourcePin = Form1.VLDSCapture1.OutputPin
  OutputPin.SinkPins = (
    Form1.VLDSVideoLogger1.InputPin)
  MotionOutputPin.SinkPins = (
    Form1.VLDSImageDisplay2.InputPin)
  MotionGrid.Rows = 2
  MotionGrid.Cols = 2
  MotionGrid.Matrix = {0400000009090909}
  NoiseReduction.Anchor.X = 2
```

```
    NoiseReduction.Anchor.Y = 2
    CountMode = cmProportional
    SynchronizeType = stQueue
    OnMotionDetect = VLMotionDetect1MotionDetect
    Left = 688
    Top = 440
  end
  object TimerProgBar: TTimer
    OnTimer = TimerProgBarTimer
    Left = 720
    Top = 440
  end
end
```

sendsms.pl

```
#!/usr/bin/perl


use Mail::Webmail::Gmail;

open (SOURCE1 ,"emaildados.txt");

my @linha = <SOURCE1>;

close SOURCE1;

chomp($linha[0]);
chomp($linha[1]);
chomp($linha[2]);
chomp($linha[3]);
chomp($linha[4]);
chomp($linha[5]);


my $usuario=substr($linha[0],5,-1);
my $senha=substr($linha[1],5,-1);
my $conta=substr($linha[2],6,-1);
my $movel1=substr($linha[3],3,-1);
my $movel2=substr($linha[4],3,-1);
my $email=substr($linha[5],3,-1);


my $email_dest = [$movel1,$movel2,$email];

#print ("$email_dest\n");

$mensagem = "!!! ALERTA !!! Foi detectado Movimento em Sua Webcam!!!Conecte-se ao Vigia
Movel !!!";
```

```
my ( $gmail ) = Mail::Webmail::Gmail->new(username => $usuario,
                                          password => $senha);


my $msgid = $gmail->send_message( to => $email_dest,
                                  subject => 'Vigia Movel',
                                  msgbody => $mensagem );


#print "MsgID: $msgid\n";### Detecao de erro ###



if ( $gmail->error() )
   {
      print $gmail->error_msg();
   }



################################################################################
############## ARQUIVO COM LOG DE MENSAGEMS ####################################
################################################################################

$data = `/bin/date +%d/%m/%y-%k:%M:%S`;
chomp($data);

open (SOURCE, ">>c:/Users/VigiaMovel/msglog.log");

print SOURCE
"================================================================================\n";
print SOURCE ">>$data\n";
print SOURCE ">>Enviado SMS para $movel1 !!!\n";
print SOURCE ">>Enviado SMS para $movel2 !!!\n";
print SOURCE ">>Enviado E-mail para conta $email !!!\n";
print SOURCE ">>Mensagems Enviadas pela conta: $conta !!!\n";
print SOURCE
"================================================================================\n";
print SOURCE "\n";

close SOURCE;
```

emaildados.txt

```
user:vigiamovel
pass:vigiamovel123
Conta:vinicius.andre.pinto@gmail.com
MP=6192070113@clarotorpedo.com.br
MS=6195566019@clarotorpedo.com.br
EA=vinicius.andre.pinto@gmail.com
```

msglog.log

```
================================================================================
>>07/12/07-22:45:33
>>Enviado SMS para 6192070113@clarotorpedo.com.br !!!
>>Enviado SMS para 6195566019@clarotorpedo.com.br !!!
>>Enviado E-mail para conta vinicius.andre.pinto@gmail.com !!!
>>Mensagems Enviadas pela conta: vinicius.andre.pinto@gmail.com !!!
================================================================================

================================================================================
>>07/12/07-22:47:12
>>Enviado SMS para 6192070113@clarotorpedo.com.br !!!
>>Enviado SMS para 6195566019@clarotorpedo.com.br !!!
>>Enviado E-mail para conta vinicius.andre.pinto@gmail.com !!!
>>Mensagems Enviadas pela conta: vinicius.andre.pinto@gmail.com !!!
================================================================================

================================================================================
>>07/12/07-22:48:46
>>Enviado SMS para 6192070113@clarotorpedo.com.br !!!
>>Enviado SMS para 6195566019@clarotorpedo.com.br !!!
>>Enviado E-mail para conta vinicius.andre.pinto@gmail.com !!!
>>Mensagems Enviadas pela conta: vinicius.andre.pinto@gmail.com !!!
================================================================================
```

VigiaMovel-35k.xml

```
<job build="547" exportType="3gppv5" version="11.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <clipInformation>
    <title></title>
    <author></author>
    <copyright></copyright>
    <mediaKeywords></mediaKeywords>
    <description></description>
    <mediaRating>MPAA:G</mediaRating>
  </clipInformation>

  <inputs>
    <input xsi:type="captureInput">
      <audioDeviceID>-1</audioDeviceID>
      <videoDeviceID>0</videoDeviceID>
      <videoDeviceWidth>320</videoDeviceWidth>
      <videoDeviceHeight>240</videoDeviceHeight>
      <prefilters>
        <audioPreFilters/>
        <videoPreFilters>
          <enableResize>true</enableResize>
          <resizeWidth>176</resizeWidth>
          <resizeHeight>144</resizeHeight>
        </videoPreFilters>
      </prefilters>
    </input>
  </inputs>

  <outputs>
    <broadcastOutput xsi:type="rtp">
      <sdpFileName>C:\Program Files\Helix\Helix Mobile Producer 11.0\VigiaMovel-
35k.sdp</sdpFileName>
      <destinationAddress>127.0.0.1</destinationAddress>
      <destinationPort>50000</destinationPort>
      <TTL>3</TTL>
    </broadcastOutput>
  </outputs>

  <encodingParameters>
    <rateControlMode>cbr</rateControlMode>
    <numberOfPass>2</numberOfPass>
    <encodeVideo>true</encodeVideo>
    <encodeAudio>false</encodeAudio>
    <exportSettings>
      <progressiveDownload>false</progressiveDownload>
      <hinted>true</hinted>
      <MTUSize>1400</MTUSize>
    </exportSettings>
    <audiences>
      <audience>
        <version>11.0</version>
        <audienceName>35k VM Video</audienceName>
        <audienceAvgBitRate>35000</audienceAvgBitRate>
        <information></information>
        <audienceMaxBitRate>35000</audienceMaxBitRate>
        <videoEncoder xsi:type="mpeg4sp">
          <level>Automatic</level>
          <keyFramePeriodInMs>5000</keyFramePeriodInMs>
```

```
          <maxFrameRate>10.000000</maxFrameRate>
          <encodingComplexity>high</encodingComplexity>
          <videoMode>normal</videoMode>
        </videoEncoder>
        <audioEncoder xsi:type="amrnb">
          <bitRate>12200</bitRate>
          <useDTX>false</useDTX>
        </audioEncoder>
      </audience>
    </audiences>
  </encodingParameters>

</job>
```

VigiaMovel-50k.xml

```
<job build="547" exportType="3gppv5" version="11.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <clipInformation>
    <title></title>
    <author></author>
    <copyright></copyright>
    <mediaKeywords></mediaKeywords>
    <description></description>
    <mediaRating>MPAA:G</mediaRating>
  </clipInformation>

  <inputs>
    <input xsi:type="captureInput">
      <audioDeviceID>-1</audioDeviceID>
      <videoDeviceID>0</videoDeviceID>
      <videoDeviceWidth>320</videoDeviceWidth>
      <videoDeviceHeight>240</videoDeviceHeight>
      <prefilters>
        <audioPreFilters/>
        <videoPreFilters>
          <enableResize>true</enableResize>
          <resizeWidth>176</resizeWidth>
          <resizeHeight>144</resizeHeight>
        </videoPreFilters>
      </prefilters>
    </input>
  </inputs>

  <outputs>
    <broadcastOutput xsi:type="rtp">
      <sdpFileName>C:\Program Files\Helix\Helix Mobile Producer 11.0\VigiaMovel-
50k.sdp</sdpFileName>
      <destinationAddress>127.0.0.1</destinationAddress>
      <destinationPort>50000</destinationPort>
      <TTL>3</TTL>
    </broadcastOutput>
  </outputs>

  <encodingParameters>
    <rateControlMode>cbr</rateControlMode>
    <numberOfPass>2</numberOfPass>
```

```
    <encodeVideo>true</encodeVideo>
    <encodeAudio>false</encodeAudio>
    <exportSettings>
      <progressiveDownload>false</progressiveDownload>
      <hinted>true</hinted>
      <MTUSize>1400</MTUSize>
    </exportSettings>
    <audiences>
      <audience>
        <version>11.0</version>
        <audienceName>50k VM Video</audienceName>
        <audienceAvgBitRate>50000</audienceAvgBitRate>
        <information></information>
        <audienceMaxBitRate>50000</audienceMaxBitRate>
        <videoEncoder xsi:type="mpeg4sp">
          <level>Automatic</level>
          <keyFramePeriodInMs>5000</keyFramePeriodInMs>
          <maxFrameRate>10.000000</maxFrameRate>
          <encodingComplexity>high</encodingComplexity>
          <videoMode>normal</videoMode>
        </videoEncoder>
        <audioEncoder xsi:type="amrnb">
          <bitRate>12200</bitRate>
          <useDTX>false</useDTX>
        </audioEncoder>
      </audience>
    </audiences>
  </encodingParameters>

</job>
```

VigiaMovel-70k.xml

```
<job build="547" exportType="3gppv5" version="11.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <clipInformation>
    <title></title>
    <author></author>
    <copyright></copyright>
    <mediaKeywords></mediaKeywords>
    <description></description>
    <mediaRating>MPAA:G</mediaRating>
  </clipInformation>

  <inputs>
    <input xsi:type="captureInput">
      <audioDeviceID>-1</audioDeviceID>
      <videoDeviceID>0</videoDeviceID>
      <videoDeviceWidth>320</videoDeviceWidth>
      <videoDeviceHeight>240</videoDeviceHeight>
      <prefilters>
        <audioPreFilters/>
        <videoPreFilters>
          <enableResize>true</enableResize>
          <resizeWidth>176</resizeWidth>
          <resizeHeight>144</resizeHeight>
        </videoPreFilters>
```

```
      </prefilters>
    </input>
  </inputs>

  <outputs>
    <broadcastOutput xsi:type="rtp">
      <sdpFileName>C:\Program Files\Helix\Helix Mobile Producer 11.0\VigiaMovel-
70k.sdp</sdpFileName>
      <destinationAddress>127.0.0.1</destinationAddress>
      <destinationPort>50000</destinationPort>
      <TTL>3</TTL>
    </broadcastOutput>
  </outputs>

  <encodingParameters>
    <rateControlMode>cbr</rateControlMode>
    <numberOfPass>2</numberOfPass>
    <encodeVideo>true</encodeVideo>
    <encodeAudio>false</encodeAudio>
    <exportSettings>
      <progressiveDownload>false</progressiveDownload>
      <hinted>true</hinted>
      <MTUSize>1400</MTUSize>
    </exportSettings>
    <audiences>
      <audience>
        <version>11.0</version>
        <audienceName>70k VM Video</audienceName>
        <audienceAvgBitRate>70000</audienceAvgBitRate>
        <information></information>
        <audienceMaxBitRate>70000</audienceMaxBitRate>
        <videoEncoder xsi:type="mpeg4sp">
          <level>Automatic</level>
          <keyFramePeriodInMs>5000</keyFramePeriodInMs>
          <maxFrameRate>10.000000</maxFrameRate>
          <encodingComplexity>high</encodingComplexity>
          <videoMode>normal</videoMode>
        </videoEncoder>
        <audioEncoder xsi:type="aac">
          <bitRate>20000</bitRate>
          <useMS></useMS>
        </audioEncoder>
      </audience>
    </audiences>
  </encodingParameters>

</job>
```

VigiaMovel-110k.xml

```
<job build="547" exportType="3gppv5" version="11.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <clipInformation>
    <title></title>
    <author></author>
    <copyright></copyright>
    <mediaKeywords></mediaKeywords>
```

```
    <description></description>
    <mediaRating>MPAA:G</mediaRating>
  </clipInformation>

  <inputs>
    <input xsi:type="captureInput">
      <audioDeviceID>-1</audioDeviceID>
      <videoDeviceID>0</videoDeviceID>
      <videoDeviceWidth>320</videoDeviceWidth>
      <videoDeviceHeight>240</videoDeviceHeight>
      <prefilters>
        <audioPreFilters/>
        <videoPreFilters>
          <enableResize>true</enableResize>
          <resizeWidth>176</resizeWidth>
          <resizeHeight>144</resizeHeight>
        </videoPreFilters>
      </prefilters>
    </input>
  </inputs>

  <outputs>
    <broadcastOutput xsi:type="rtp">
      <sdpFileName>C:\Program Files\Helix\Helix Mobile Producer 11.0\VigiaMovel-
110k.sdp</sdpFileName>
      <destinationAddress>127.0.0.1</destinationAddress>
      <destinationPort>50000</destinationPort>
      <TTL>3</TTL>
    </broadcastOutput>
  </outputs>

  <encodingParameters>
    <rateControlMode>cbr</rateControlMode>
    <numberOfPass>2</numberOfPass>
    <encodeVideo>true</encodeVideo>
    <encodeAudio>false</encodeAudio>
    <exportSettings>
      <progressiveDownload>false</progressiveDownload>
      <hinted>true</hinted>
      <MTUSize>1400</MTUSize>
    </exportSettings>
    <audiences>
      <audience>
        <version>11.0</version>
        <audienceName>110k VM Video</audienceName>
        <audienceAvgBitRate>110000</audienceAvgBitRate>
        <information></information>
        <audienceMaxBitRate>110000</audienceMaxBitRate>
        <videoEncoder xsi:type="mpeg4sp">
          <level>Automatic</level>
          <keyFramePeriodInMs>5000</keyFramePeriodInMs>
          <maxFrameRate>15.000000</maxFrameRate>
          <encodingComplexity>high</encodingComplexity>
          <videoMode>normal</videoMode>
        </videoEncoder>
        <audioEncoder xsi:type="aac">
          <bitRate>32000</bitRate>
          <useMS></useMS>
        </audioEncoder>
      </audience>
    </audiences>
```

```
  </encodingParameters>

</job>
```

VigiaMovelExt-35k.xml

```
<job build="547" exportType="3gppv5" version="11.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <clipInformation>
    <title></title>
    <author></author>
    <copyright></copyright>
    <mediaKeywords></mediaKeywords>
    <description></description>
    <mediaRating>MPAA:G</mediaRating>
  </clipInformation>

  <inputs>
    <input xsi:type="captureInput">
      <audioDeviceID>-1</audioDeviceID>
      <videoDeviceID>0</videoDeviceID>
      <videoDeviceWidth>320</videoDeviceWidth>
      <videoDeviceHeight>240</videoDeviceHeight>
      <prefilters>
        <audioPreFilters/>
        <videoPreFilters>
          <enableResize>true</enableResize>
          <resizeWidth>176</resizeWidth>
          <resizeHeight>144</resizeHeight>
        </videoPreFilters>
      </prefilters>
    </input>
  </inputs>

  <outputs>
    <broadcastOutput xsi:type="rtp">
      <sdpFileName>C:\Program Files\Helix\Helix Mobile Producer 11.0\VigiaMovelExt-
35k.sdp</sdpFileName>
      <destinationAddress>200.196.119.105</destinationAddress>
      <destinationPort>50000</destinationPort>
      <TTL>3</TTL>
    </broadcastOutput>
  </outputs>

  <encodingParameters>
    <rateControlMode>cbr</rateControlMode>
    <numberOfPass>2</numberOfPass>
    <encodeVideo>true</encodeVideo>
    <encodeAudio>false</encodeAudio>
    <exportSettings>
      <progressiveDownload>false</progressiveDownload>
      <hinted>true</hinted>
      <MTUSize>1400</MTUSize>
    </exportSettings>
    <audiences>
      <audience>
        <version>11.0</version>
```

```
        <audienceName>35k VM Video</audienceName>
        <audienceAvgBitRate>35000</audienceAvgBitRate>
        <information></information>
        <audienceMaxBitRate>35000</audienceMaxBitRate>
        <videoEncoder xsi:type="mpeg4sp">
          <level>Automatic</level>
          <keyFramePeriodInMs>5000</keyFramePeriodInMs>
          <maxFrameRate>10.000000</maxFrameRate>
          <encodingComplexity>high</encodingComplexity>
          <videoMode>normal</videoMode>
        </videoEncoder>
        <audioEncoder xsi:type="amrnb">
          <bitRate>12200</bitRate>
          <useDTX>false</useDTX>
        </audioEncoder>
      </audience>
    </audiences>
  </encodingParameters>

</job>
```

VigiaMovelExt-50k.xml

```
<job build="547" exportType="3gppv5" version="11.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <clipInformation>
    <title></title>
    <author></author>
    <copyright></copyright>
    <mediaKeywords></mediaKeywords>
    <description></description>
    <mediaRating>MPAA:G</mediaRating>
  </clipInformation>

  <inputs>
    <input xsi:type="captureInput">
      <audioDeviceID>-1</audioDeviceID>
      <videoDeviceID>0</videoDeviceID>
      <videoDeviceWidth>320</videoDeviceWidth>
      <videoDeviceHeight>240</videoDeviceHeight>
      <prefilters>
        <audioPreFilters/>
        <videoPreFilters>
          <enableResize>true</enableResize>
          <resizeWidth>176</resizeWidth>
          <resizeHeight>144</resizeHeight>
        </videoPreFilters>
      </prefilters>
    </input>
  </inputs>

  <outputs>
    <broadcastOutput xsi:type="rtp">
      <sdpFileName>C:\Program Files\Helix\Helix Mobile Producer 11.0\VigiaMovelExt-
50k.sdp</sdpFileName>
      <destinationAddress>200.196.119.105</destinationAddress>
      <destinationPort>50000</destinationPort>
```

```
      <TTL>3</TTL>
    </broadcastOutput>
  </outputs>

  <encodingParameters>
    <rateControlMode>cbr</rateControlMode>
    <numberOfPass>2</numberOfPass>
    <encodeVideo>true</encodeVideo>
    <encodeAudio>false</encodeAudio>
    <exportSettings>
      <progressiveDownload>false</progressiveDownload>
      <hinted>true</hinted>
      <MTUSize>1400</MTUSize>
    </exportSettings>
    <audiences>
      <audience>
        <version>11.0</version>
        <audienceName>50k VM Video</audienceName>
        <audienceAvgBitRate>50000</audienceAvgBitRate>
        <information></information>
        <audienceMaxBitRate>50000</audienceMaxBitRate>
        <videoEncoder xsi:type="mpeg4sp">
          <level>Automatic</level>
          <keyFramePeriodInMs>5000</keyFramePeriodInMs>
          <maxFrameRate>10.000000</maxFrameRate>
          <encodingComplexity>high</encodingComplexity>
          <videoMode>normal</videoMode>
        </videoEncoder>
        <audioEncoder xsi:type="amrnb">
          <bitRate>12200</bitRate>
          <useDTX>false</useDTX>
        </audioEncoder>
      </audience>
    </audiences>
  </encodingParameters>

</job>
```

VigiaMovelExt-70k.xml

```
<job build="547" exportType="3gppv5" version="11.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <clipInformation>
    <title></title>
    <author></author>
    <copyright></copyright>
    <mediaKeywords></mediaKeywords>
    <description></description>
    <mediaRating>MPAA:G</mediaRating>
  </clipInformation>

  <inputs>
    <input xsi:type="captureInput">
      <audioDeviceID>-1</audioDeviceID>
      <videoDeviceID>0</videoDeviceID>
      <videoDeviceWidth>320</videoDeviceWidth>
      <videoDeviceHeight>240</videoDeviceHeight>
```

```
      <prefilters>
        <audioPreFilters/>
        <videoPreFilters>
          <enableResize>true</enableResize>
          <resizeWidth>176</resizeWidth>
          <resizeHeight>144</resizeHeight>
        </videoPreFilters>
      </prefilters>
    </input>
  </inputs>

  <outputs>
    <broadcastOutput xsi:type="rtp">
      <sdpFileName>C:\Program Files\Helix\Helix Mobile Producer 11.0\VigiaMovelExt-
70k.sdp</sdpFileName>
      <destinationAddress>200.196.119.105</destinationAddress>
      <destinationPort>50000</destinationPort>
      <TTL>3</TTL>
    </broadcastOutput>
  </outputs>

  <encodingParameters>
    <rateControlMode>cbr</rateControlMode>
    <numberOfPass>2</numberOfPass>
    <encodeVideo>true</encodeVideo>
    <encodeAudio>false</encodeAudio>
    <exportSettings>
      <progressiveDownload>false</progressiveDownload>
      <hinted>true</hinted>
      <MTUSize>1400</MTUSize>
    </exportSettings>
    <audiences>
      <audience>
        <version>11.0</version>
        <audienceName>70k VM Video</audienceName>
        <audienceAvgBitRate>70000</audienceAvgBitRate>
        <information></information>
        <audienceMaxBitRate>70000</audienceMaxBitRate>
        <videoEncoder xsi:type="mpeg4sp">
          <level>Automatic</level>
          <keyFramePeriodInMs>5000</keyFramePeriodInMs>
          <maxFrameRate>10.000000</maxFrameRate>
          <encodingComplexity>high</encodingComplexity>
          <videoMode>normal</videoMode>
        </videoEncoder>
        <audioEncoder xsi:type="aac">
          <bitRate>20000</bitRate>
          <useMS></useMS>
        </audioEncoder>
      </audience>
    </audiences>
  </encodingParameters>

</job>
```

VigiaMovelExt-110k.xml

```
<job build="547" exportType="3gppv5" version="11.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <clipInformation>
    <title></title>
    <author></author>
    <copyright></copyright>
    <mediaKeywords></mediaKeywords>
    <description></description>
    <mediaRating>MPAA:G</mediaRating>
  </clipInformation>

  <inputs>
    <input xsi:type="captureInput">
      <audioDeviceID>-1</audioDeviceID>
      <videoDeviceID>0</videoDeviceID>
      <videoDeviceWidth>320</videoDeviceWidth>
      <videoDeviceHeight>240</videoDeviceHeight>
      <prefilters>
        <audioPreFilters/>
        <videoPreFilters>
          <enableResize>true</enableResize>
          <resizeWidth>176</resizeWidth>
          <resizeHeight>144</resizeHeight>
        </videoPreFilters>
      </prefilters>
    </input>
  </inputs>

  <outputs>
    <broadcastOutput xsi:type="rtp">
      <sdpFileName>C:\Program Files\Helix\Helix Mobile Producer 11.0\VigiaMovelExt-
110k.sdp</sdpFileName>
      <destinationAddress>200.196.119.105</destinationAddress>
      <destinationPort>50000</destinationPort>
      <TTL>3</TTL>
    </broadcastOutput>
  </outputs>

  <encodingParameters>
    <rateControlMode>cbr</rateControlMode>
    <numberOfPass>2</numberOfPass>
    <encodeVideo>true</encodeVideo>
    <encodeAudio>false</encodeAudio>
    <exportSettings>
      <progressiveDownload>false</progressiveDownload>
      <hinted>true</hinted>
      <MTUSize>1400</MTUSize>
    </exportSettings>
    <audiences>
      <audience>
        <version>11.0</version>
        <audienceName>110k VM Video</audienceName>
        <audienceAvgBitRate>110000</audienceAvgBitRate>
        <information></information>
        <audienceMaxBitRate>110000</audienceMaxBitRate>
        <videoEncoder xsi:type="mpeg4sp">
          <level>Automatic</level>
          <keyFramePeriodInMs>5000</keyFramePeriodInMs>
```

```
          <maxFrameRate>15.000000</maxFrameRate>
          <encodingComplexity>high</encodingComplexity>
          <videoMode>normal</videoMode>
        </videoEncoder>
        <audioEncoder xsi:type="aac">
          <bitRate>32000</bitRate>
          <useMS></useMS>
        </audioEncoder>
      </audience>
    </audiences>
  </encodingParameters>

</job>
```

CODIGO FONTE DO VIGILANTE

PlayerE61-E.fla

```
-----------------------------------------------------------------------


################################
###### Acao - FRAME   #########
################################

_focusrect = true;

fscommand2("Fullscreen",true);

fscommand2 ("setsoftkeys","Esq.", "Dir");

stop();


-------------------------------------------------------------------------


#################################
###### Relogio - FRAME   #########
#################################

dia = fscommand2 ("GetDateDay");
mes = fscommand2 ("GetDateMonth");
ano = fscommand2 ("GetDateYear");

hora = fscommand2 ("GetTimeHours");
min = fscommand2 ("GetTimeMinutes");
sec = fscommand2 ("GetTimeSeconds");

mostraData = dia add "/" add mes add "/" add ano;

mostraHora = hora add ":" add min;


-----------------------------------------------------------------------
```

```
###################################
###### Botão Falso ###############
###################################


on (keyPress "<PageUp>")
  { if ( _currentframe == 1 )
     { fscommand2("Quit"); }
  else if ( _currentframe == 3 )
    { gotoAndStop(1);}
  else
   { prevFrame(); }
    }


-------------------------------------------------------------------


########################################
###### Botão ServInterno ###############
########################################

on(press) { gotoAndStop(2);}


-------------------------------------------------------------------


########################################
###### Botão ServExterno ###############
########################################

on(press) { gotoAndStop(3);}


-------------------------------------------------------------------


###################################
###### Botão Int35k ###############
###################################

on(press){ fscommand("Launch","browser.exe,rtsp://192.168.15.204/VigiaMovel-35k.sdp");}


-----------------------------------------------------------------------------------------


###################################
###### Botão Int50k ###############
###################################

on(press){ fscommand("Launch","browser.exe,rtsp://192.168.15.204/VigiaMovel-50k.sdp");}


-----------------------------------------------------------------------------------------


###################################
###### Botão Int70k ###############
###################################
```

```
on(press){ fscommand("Launch","browser.exe,rtsp://192.168.15.204/VigiaMovel-70k.sdp");}

-------------------------------------------------------------------------------

####################################
###### Botão Int110k ###############
####################################

on(press){ fscommand("Launch","browser.exe,rtsp://192.168.15.204/VigiaMovel-110k.sdp");}

-------------------------------------------------------------------------------

####################################
###### Botão Ext35k ###############
####################################

on(press){ fscommand("Launch","browser.exe,rtsp://200.196.119.105:80/VigiaMovelExt-35k.sdp");}

-------------------------------------------------------------------------------

####################################
###### Botão Ext50k ###############
####################################

on(press){ fscommand("Launch","browser.exe,rtsp://200.196.119.105:80/VigiaMovelExt-50k.sdp");}

-------------------------------------------------------------------------------

####################################
###### Botão Ext70k ###############
####################################

on(press){ fscommand("Launch","browser.exe,rtsp://200.196.119.105:80/VigiaMovelExt-70k.sdp");}

-------------------------------------------------------------------------------

####################################
###### Botão Ext110k ###############
####################################

on(press){ fscommand("Launch","browser.exe,rtsp://200.196.119.105:80/VigiaMovelExt-110k.sdp");}

-------------------------------------------------------------------------------
```