ILONANI

A table-top robot for remote interaction

Department of Automation and
Systems Technology
Helsinki University of Technology
Raúl Rodríguez Pearson
2009

Table of Contents

Index of Illustrations

Index of Tables

Glossary and abbreviations

ACC – also known as HE-ACC, High Efficiency Advance Audio Coding

AVR – microcontroller family developed by ATMEL

HTTP – Hyper-Text Transfer Protocol

GSM – Global System for Mobile communications

MMS – Multimedia Messaging Service

MP3 – MPEG-1 audio layer 3

NSV – Nullsoft Streaming Video

PBX – Private Branch eXchange

PDA – Personal Digital Assistant

PSTN – Public Switched Telephone Network

PWM – Pulse-Width Modulated

SCP – Secure Copy (can also refer to Secure Copy Protocol or Secure Copy Program), used to transfer files between a local and a remote host or between two remote hosts using SSH

SHOUTcast – server software for streaming media developed by Nulsoft

SMS – Short Messaging Service

SSH – Secure Shell, a network protocol that allows data to be exchanged using a secure channel between to networked devices.

Theora – lossy video compression codec developed by the Xiph.Org Foundation as part of their Ogg project

USB – Universal Serial Bus

Vorbis – lossy audio compression codec receiving its name from the Vorbis project, headed by the Xiph.Org Foundation

VoIP – Voice over Internet Protocol

WLAN – Wireless Local Access Network

Resumen en español

Introducción

Este documento presenta los resultados de la investigación realizada con el objetivo de desarrollar un robot para la interacción remota. La concepción de este "robot de mesa" incluye el uso de un Internet Tablet de Nokia, el N810, como la cabeza y cara del robot, concentrando las funciones de procesamiento y la interfaz con los usuarios.

Este dispositivo iría montado sobre una base que proporcionaría la capacidad de giro y balanceo, controlado con un sistema formado por un microcontrolador y un grupo de servomecanismos. El trabajo realizado bajo este proyecto se concentra en el desarrollo de una configuración software que permita la realización de video conferencia entre dos de estos dispositivos.

Los resultados obtenidos tienen dos facetas: por un lado, se configuró un sistema que proporcionaba la funcionalidad de video conferencia (aunque no de la manera anticipada), y por otro, se obtuvo un conocimiento destilado en este documento con la idea de que fuese utilizado en futuros proyectos.

Visión y estado del arte

En el futuro, nuestros hogares oficinas y lugares de ocio estarán inundados de dispositivos inteligentes y sistemas con capacidad de crear interconexiones. Ya es comúnmente aceptado el hecho de que gran mayoría de los objetos que llevamos encima incluyen cierta cantidad de electrónica. A medida que vaya avanzando la tecnología, podemos esperar que estos dispositivos serán más potentes y funcionales.

Se esperará que estos dispositivos sean capaces de interconectarse entre ellos y con redes más grandes, como Internet. Incorporarán en su diseño conceptos propios de los conocidos como "dispositivos ambientales" [3], proporcionando información de manera no intrusiva. Nos acostumbraremos a la presencia de estos objetos y daremos por hecho los servicios que proporcionen. Probablemente, la potencia de computación, almacenamiento de información y la capacidad de razonamiento inteligente estará distribuida entre todos estos "objetos inteligentes".

Toda esta interconexión conlleva un incremento de las posibilidades de comunicación. Comunicación entre objetos, entre objetos y entre objetos y personas. Aparecerán nuevas formas de interacción humano-máquina,

formas más intuitivas y naturales. Se dará por hecho que los objetos pueden interactuar entre si para proporcionar mejores servicios a los humanos. Probablemente, incluso la sociedad será más abierta y comunicativa.

La información será ubiqua. Será una sólo una consecuencia más de la interconectividad: si todos los artefactos y sistemas están conectados, pueden compartir información mucho mejor, lo que hace que esté disponible en muchos más sitios, para más sistemas y personas. Esta información no sólo será información objetiva sobre hechos, como noticias o empresas en tu entorno local, sino que incluirá información del terreno personal. Ya a día de hoy, la gente está compartiendo grandes cantidades de información personal a través de blogs y redes sociales, lo que parece un compromiso que muchas personas están dispuestas a hacer a cambio de la personalización de los servicios que les son prestados. Al menos hasta cierto punto.

Aparecerán nuevas formas de entretenimiento. La tendencia ya ha sido establecida por páginas como Youtube, Flickr y Legal Torrents. Ofrecen contenido creado por los propios usuarios del sistema. Los contenidos del futuro serán distribuidos de esta manera, a diferencia del modo actual en que una gran productora de contenidos realiza todo el esfuerzo creativo para que los usuarios luego consuman los resultados de manera pasiva. El contenido se vuelve interactivo y los usuarios aportan contenido original y remezclan el contenido aportado por otros usuarios.

También podemos esperar que el futuro sea tele-todo. Si las cosas siguen evolucionando como hasta ahora, resolver los problemas de CO2, sobre problación, consumo excesivo de energía y demás, quizá puedan ser resueltos con cosas como la tele-medicina, de manera que los pacientes reciben atención en casa o en un hospital cercano, sin tener que desplazarse grandes distancias, tele-trabajo, de manera que desaparezcan las horas puntas de tráfico en las ciudades, etc.

En el mundo del futuro, dispositivos como ILONANI irán jugando un papel cada vez más importante.

NABAZTAG – OBJETOS INTELIGENTES, DISPOSITIVOS AMBIENTALES Y COMPUTACIÓN UBICUA

Uno de los proyectos que inspiró la visión de este proyecto fin de carrera fue Nabaztag. Nabaztag es un objeto de electrónica de consumo con forma de conejo. Se conecta a Internet a través de una conexión inalámbrica y puede ser configurado y programado para hacer cosas como consultar e informar al usuario sobre previsiones meteorológicas, comprobar el email y leer los mensajes en alto, leer archivos de sindicación RSS, reproducir música en streaming y mucho más.

Por un lado, puede ser considerado como un “dispositivo ambiental” y, por otro, también es un objeto inteligente. Ambos conceptos han sido creados en los últimos años y proporcionan la promesa de un gran número de aplicaciones y mercados potenciales.

Los “dispositivos ambientales” son un nuevo género de productos de electrónica de consumo que hacen uso de la capacidad de proceso “pre-atentiva”. Esta es la capacidad que tiene el cerebro de procesar información sin tener que prestarle atención de manera explícita. Estos dispositivos proporcionan información de una manera no intrusiva. Este concepto surgió en el MediaLab del MIT, que resultó en una spin-off llamada Ambient Devices [7]. Uno de los productos ofrecidos por esta compañía es el Orb, que brilla con diferentes colores en función de información de la bolsa, la congestión del tráfico, previsiones meteorológicas o de cualquier otro canal de información Ambiental (flujos de información proporcionados por Ambient Devices).

Los objetos que disponen de componentes electrónicos que los hacen más “inteligentes”, reciben el nombre de objetos inteligentes. Aprovechando el bajo coste de muchos componentes electrónicos, objetos corrientes pueden ser equipados con conectividad inalámbrica, sensores e identificadores. Esto permite la creación de “objetos interconectados”, la Internet de las cosas. Esto abre el camino para multitud de aplicaciones nuevas – por ejemplo, un detector de humo podría pedir a los electrodomésticos de una casa que se apagasen, evitando quizá que el fuego fuese a peor; una nevera inteligente preparada para leer etiquetas RFID podría alertar al usuario de que hay ciertos alimentos que están a punto de caducar o de enviar una lista de la compra con los objetos agotados al email del usuario.

Esto nos acerca a otra tendencia interesante. Las etiquetas RFID parecen una buena forma de crear esa red de objetos interconectados. Los usuarios pueden comprar paquetes de etiquetas que pueden ser programadas y adjuntadas a los objetos para llevar a cabo ciertas acciones. Por ejemplo, los Ztamp:s de Violet [8] permiten al usuario pegar una etiqueta a un paraguas y obtener una previsión meteorológica en la pantalla del ordenador con solo acercarlo al lector de etiquetas (el Mir:ror [9]).

Telecomunicaciones, redes sociales y entretenimiento 2.0

Hoy en día asumimos como normal que cualquiera puede ser contactado desde cualquier sitio y a cualquier hora, pero no siempre fue así. Antes, tele-comunicarse no era algo tan común – si uno quería comunicarse con alguien, se establecía una cita y se trataban las cosas cara a cara. Actualmente, la tecnología permite la comunicación entre personas separadas por muchos kilómetros. Esto abre también multitud de posibilidades, muchas de las cuales ya son una realidad.

Los orígenes de las telecomunicaciones se remontan a las señales de humo y los tambores pero, a efectos de esta discusión, podríamos trazarlos a la invención de los primeros sistemas de telecomunicación eléctricos. El primero sería el telégrafo, que hacía posible la transferencia de mensajes usando señales eléctricas transmitidas por cable. El primer telégrafo comercial comenzó a operar en 1839 a lo largo de un trazado de 21 kilómetros. Unos años después, en 1876, Alexander Graham Bell inventó el teléfono convencional, y los primeros servicios comerciales de telefonía fueron puestos en marcha entre 1878 y 1879 en Estados Unidos y Gran Bretaña. Sin embargo, no fue hasta 1927 cuando fue posible conectar los continentes a ambos lados del Atlántico a través de un enlace de radio, y en 1956 a través del primer cable transatlántico [11].

A través de esta acelerada evolución, estos primeros sistemas han dejado paso a lo que existe hoy en día, una compleja red de sistemas y servicios de telecomunicación que proporcionan a los usuarios una amplia selección de opciones entre las que elegir. Hoy en día, los usuarios pueden llamar a un amigo usando una línea fija de la Red Telefónica Conmutada, un teléfono móvil usando GSM, comprobar su buzón de entrada de correo electrónico desde cualquier ordenador, iniciar una conversación de mensajería instantánea con alguno de sus conocidos, enviar un SMS o un MMS usando su PDA o una video-conferencia usando su cliente jabber. Las posibilidades son tan numerosas, que son difíciles de cubrir.

El concepto fundamental a tener en cuenta es el de red. Una red, en este contexto, es un sistema de entidades interconectadas. Con la invención del telégrafo, se creó una red de líneas a través de las cuales podían enviarse mensajes. La invención el teléfono proporcionó un empuje adicional al desarrollo de estas redes de telecomunicaciones. A día de hoy, estas redes originales han sido mejoradas con la tecnología actual, proporcionando la base para Internet, la red de redes [12].

La tendencia será la de que más y más servicios estén ejecutándose sobre este soporte de interconexiones. Una muestra de los servicios que existen a día de hoy son:

- Programas de mensajería instantánea que permiten la comunicación a través de chats.

- Sistemas de VoIP (voz sobre IP) que enrutan las llamadas a través de redes de conmutación de paquetes con una calidad de servicio más que aceptable.

- Streaming de video desde productores de contenido o entre usuarios, que se hace más común a medida que las tecnologías VoIP comienzan a soportar la transmisión de video de manera más generalizada.

Esta ubicuidad de las redes de telecomunicaciones ha proporcionado un terreno fértil para el florecimiento de nuevas interacciones sociales. Las personas pueden interactuar de nuevas maneras. Por ejemplo:

- Blogs: es muy sencillo crear un espacio virtual donde expresar opiniones o a través del cual mantener al día al mundo sobre nuestras últimas ocurrencias.

- Myspace ha adquirido gran popularidad entre adolescentes de todo el mundo y ha supuesto una especie de estándar de páginas web de artistas y músicos de todos los niveles.

- Las redes sociales como Facebook son prueba de que la gente está dispuesta a compartir cierta cantidad de información privada a cambio de la posibilidad de reconectar con gente en su vida.

- La mensajería instantánea, la voz sobre IP y el correo electrónico son versiones rápidas y elegantes del telégrafo y el teléfono, acercando a gente de todo el mundo.

- Portales como Youtube o Legal Torrents abren las puertas a nuevas formas de comunicación en las cuales la gente genera contenidos audiovisuales con fines artísticos, de entretenimiento o para comunicar un mensaje.

- Cualquier persona puede escribir y publicar un libro usando servicios como Lulu, que pueden publicar contenidos bajo pedido. La tecnología que usan hace posible que sea viable la publicación incluso de obras que vayan a tener poca tirada.

- Los mundos virtuales online proporcionan la posibilidad de crear un avatar, una personalidad virtual, y de interactuar con objetos y otras personalidades virtuales, controladas a su vez por otras personas.

Estos ejemplos son sólo la punta del iceberg. Existen muchas otras maneras de comunicar e interactuar a través de la red, y muchas más verán la luz en el futuro cercano. Si enlazamos esta idea con la de los objetos ambientales e inteligentes, es claro que artilugios como Ilonani no sólo tendrán un lugar en el mundo de hoy y del mañana, sino que jugarán un papel importante en nuestros hogares, lugares de trabajo y de ocio.

Adicionalmente, la tendencia de la industria del software de producir aplicaciones accesibles a través del navegador de Internet, las aplicaciones web, hace que sea más razonable, si cabe, pensar que en el futuro todos los dispositivos estarán conectados a “la nube”, proporcionando posibilidades ilimitadas de interacción y comunicación.

Tele-asistencia

No es algo nuevo el hecho de que la población en los países desarrollados está envejeciendo. Se espera que esta tendencia aparezca también en los países en vías de desarrollo. Las sociedades envejecidas supondrán una carga muy grande a los sistemas de seguridad social, y se tenderá a buscar soluciones que provean de estos servicios de una manera más eficiente y barata [14].

Si fijamos nuestra atención en las nuevas tecnologías como las mencionadas en el apartado anterior, es evidente que muchas de estas soluciones incluirán elementos del área de las telecomunicaciones. En este sentido, uno de los usos previstos para Ilonani y para los dispositivos de su clase es el de sistemas de monitorización de uso amigable. La gente mayor necesita dispositivos que sean fáciles de utilizar, con grandes botones e interfaces intuitivas. Los cuidadores podrían usar dispositivos como Ilonani para facilitar el cuidado de las personas mayores a su cargo.

Estos dispositivos también pueden tener lugar en el área de la tele-medicina. La evolución de objetos que permitan interactuar e intercambiar información de manera que soporten servicios de tele-medicina y tele-asistencia será una tendencia a la que convendrá prestar atención.

Ilonani, el concepto

Concepto y objetivos

El objetivo inicial del proyecto se centró en la construcción de un robot de mesa que proporcionase una interfaz amigable para personas mayores, para ayudar en las labores de tele-asistencia. La lista de objetivos perseguidos ha sido:

- Investigar la posibilidad de usar el N810 como plataforma para implementar el “cerebro” y la “cara” de Ilonani. Investigar la viabilidad de realizar video-conferencia (teniendo en cuenta distintos escenarios de aplicación) y si proporciona potencial para realizar una aplicación con una interfaz suficientemente amigable (teniendo en cuenta el requerimiento de que debería ser usable por personas mayores).

- Si el dispositivo de Nokia resultaba ser una buena elección, investigar qué posibles soluciones existirían; qué posibles implementaciones existían para el sistema, y cuál sería la mejor opción.

- Diseñar y construir un prototipo físico del robot, incluyendo una base de carga, un chasis que alojase los componentes electrónicos y un mecanismo de giro y balanceo.

La visión original también incluía la funcionalidad de seguimiento del interlocutor con la cámara. Después de un poco de investigación, se estimó que resultaba demasiado complicado y fuera, por tanto, del alcance de un proyecto fin de carrera como este.

Además, el proyecto original fue dividido en dos. El proyecto descrito en este documento se hará cargo de los dos primeros objetivos. La información relativa al diseño y construcción de un prototipo físico se encontrará en la documentación del proyecto final de carrera de Gonzalo Zubieta.

Otro de los objetivos que ha sido tenido en cuenta es el de tener en cuenta, en la fase de diseño, la extensibilidad del sistema. Desde el principio ha resultado evidente que multitud de ideas (de aplicación y desarrollo de Ilonani) tendrían que ser dejadas de lado, puesto que no sería posible implementar todas bajo este proyecto. Para conseguir que el resultado de esta investigación y diseño sea extensible, se ha realizado un gran esfuerzo para que de verdad pueda servir como base para el trabajo de futuros proyectos.

Diagrama de bloques

El sistema estará constituido por los bloques mostrados en la figura 7 (página 34). como ya se ha mencionado, el Nokia N810 será el “cerebro” y la “cara” del robot. Estará conectado por USB con un micro-controlador AVR que será usado para generar las señales PWM para el control de los servos del sistema de giro y balanceo.

Adicionalmente, se usará una tarjeta impresa donde se incluirá toda la electrónica de adaptación de potencia para generar los niveles de alimentación necesarios para los demás elementos del sistema. Esta tarjeta también dispondrá de una conexión a batería (para obtener potencia o para recargarla, dependiendo de si el robot está conectado o no a una toma de corriente eléctrica). También se decidió alimentar el tablet de Nokia usando su propio conector de alimentación, para simplificar el sistema y disminuir la probabilidad de que produzcan daños en el aparato.

El Nokia N810 es un internet tablet, un dispositivo desarrollado por Nokia bajo la gama N de productos. No se trata de un teléfono móvil, ya que no dispone de ninguna forma para conectarse a la red celular típica de los teléfonos móviles. Sin embargo, dispone de conectividad Wi-Fi y Bluetooth que le puede proporcionar acceso a Internet (en el caso del Bluetooth, cuando es utilizado en conjunción con un teléfono móvil con acceso a Internet). La tabla 1, en la página 35, proporciona un resumen de las principales características del dispositivo.

Esta línea de productos de Nokia ejecuta un versión modificada del sistema Debian GNU/Linux llamada Maemo. Es un sistema operativo hecho a medida para los internet tablets de Nokia.

La tarjeta del micro-controlador AVR fué desarrollada por Antti Karjalainen del Departamento de Automática y Sistemas de la Universidad de Tecnología de Helsinki. Dispone de un AT90USB162 de ATMEL. La tarjeta ha sido usada satisfactoriamente para el control de servos desde sistemas operativos Linux. Además, usar esta placa proporcionaba el potencial de extender el sistema con el uso de redes Zigbee desarrolladas bajo otro proyecto dentro también del laboratorio de dicho Departamento.

Se prevée que la tarjeta de alimentación incluirá componentes electrónicos para la monitorización de la carga de la batería (y para actuar acorde al estado de la misma), un LED de indicador de batería baja, los drivers para los servos, adaptación potencia para alimentar la tarjeta del AVR y una serie de conectores que permitan la interconexión de todos los elementos.

La batería será elegida tal que soporte un amperio nominal y 2,6 amperios de pico (1,3 amperios por servo, cuando estos se encuentran bloqueados). La especificación inicial requería que el sistema pudiese funcionar durante 6 horas sin recarga.

Los servos elegidos son los Futaba S3003, puesto que proporcionan una solución asequible y fiable. El control de los mismos se llevará a través de una señal PWM. En la figura 8 y la tabla 2 (página 37) puede observarse más información sobre dichos servos.

También se estudió la inclusión de una serie de botones físicos. Se estimó razonable la inclusión como mínimo de un botón de encendido/apagado y unos botones para controlar el movimiento de giro y balanceo. No obstante, también se estudiaron las ventajas de incluir unos botones por software, aprovechando que el Nokia 810 proporciona pantalla táctil.

La aplicación software

Introducción

La visión para el desarrollo de la aplicación de software incluía la siguiente funcionalidad:

- Video-conferencia bidireccional fiable
- Interfaz de usuario sencilla:
    - Para controlar y configurar la video-conferencia

- Para controlar el movimiento del mecanismo de giro y balanceo.

Al comienzo del proyecto, no estaba comprobado que el Nokia N810 fuese capaz de soportar video-conferencia bidireccional de manera robusta. Aparentemente, no existía ninguna solución de video-conferencia que funcionase de fábrica, y diversas opciones ya habían sido investigadas dentro del Grupo de Investigación de Tecnología de Automática [21]. Como no parecía razonable suponer que el dispositivo no fuese capaz de realizar video-conferencia, se invirtió un gran esfuerzo en averiguar cómo conseguirlo.

Esto resultó ser más complicado de lo que cabía esperar. El resultado de esta investigación sí incluye una configuración básica que permite la video-conferencia, pero no de la forma prevista.

Explorando la plataforma – maemo y gstreamer

La plataforma Maemo es el nombre con el que se conoce a la pila de software desarrollada para los internet tablets de Nokia. Incluye un sistema operativo y un kit de desarrollo de software.

Maemo está basado en la distribución Debian GNU/Linux e incorpora la mayor parte de la interfaz gráfica, frameworks y librerías del proyecto GNOME. Usa Matchbox como su gestor de ventanas y Hildon como su framework de interfaz gráfica y de aplicaciones.

Uno de los frameworks que resulta más relevante para este proyecto es GStreamer, un framework multimedia basado en una arquitectura de tuberías, programado en C y que usa el sistema de tipos GObject. La arquitectura de este sistema proporciona la posibilidad de crear un amplio rango de aplicaciones multimedia, desde sencillos reproductores de audio hasta complicados editores de video y aplicaciones de streaming multimedia. La figura 10 de la página 41 muestra un diagrama de bloques de este framework.

En el apartado 3.2. se elabora una exposición del estudio que se realizó de este framework. Se llegó a implementar un sistema que permitía el streaming de video en una dirección, pero se abandonó esta línea de investigación por la previsión de que no habría suficiente tiempo para construir una solución basada en ella.

En el apartado 3.3. se presentan las líneas de investigación más importantes de entre las que se siguieron. En el apartado 3.4. se discute la opción elegida finalmente. Como se comentó anteriormente, esta solución final no había sido anticipada e implementar el sistema de video-conferencia resultó ser mucho más sencillo de lo que se pensaba. Como se describe en dicho apartado, el sistema estaba basado en un par de cuentas GTalk y la apropiada configuración del cliente de comunicaciones del Nokia N810.

Conclusión

Resultados

Los resultados fundamentales de este proyecto son:

- un sistema de video-conferencia funciona,
- el conocimiento adquirido y condensado en este documento y
- código fuente que podría ser usado como base para la construcción de una aplicación que siguiese la visión esbozada al principio de este apartado.

El sistema de video-conferencia no se parece al que se tenía en mente al inicio del proyecto, pero al menos proporciona la funcionalidad con una calidad bastante razonable. También requiere muy poco desarrollo y funciona prácticamente de fábrica. Los invconvenientes son que se necesita un agente externo proveedor del servicio (es decir, Google) y que el cliente de comunicaciones disponible no proporciona toda la funcionalidad requerida.

Del estudio desarrollado se desprende que podrían existir dos tipos de soluciones: soluciones sin servidor y soluciones del tipo servidor/cliente. Las ventajas y desventajas de cada opción son expuestas en el apartado 3.3. pero, como conclusión, se considera la opción de servidor/cliente como más apropiada. Una opción que no ha sido documentada por haber sido investigada sólo superficialmente es la del sistema de comunicaciones GIMnet [46]. Se estima que esta opción merecería la atención de futuros trabajos que pretendan continuar el iniciado bajo este proyecto.

En el Appendix B. se incluye el código fuente desarrollado para implementar la primera opción explorada de video-conferencia sin servidor central. Como ya se ha comentado, este camino fue abandonado porque se estimó que faltaría tiempo para llevarla a cabo con rigor.

A la vista de la solución implementada finalmente, podría parecer que el trabajo realizado bajo este proyecto habría sido poco productivo. Para ser justos, hay que tener en cuenta que muchos descubrimientos tuvieron que tener lugar hasta llegar al punto en el que se pudo implementar dicha solución. Gran cantidad de esfuerzo y energía tuvo que invertirse en identificar problemas de conectividad derivados de la infraestructura de la que se disponía en el laboratorio, incluyendo problemas derivados del NAT y los corta-fuegos.

Trabajos futuros

Puede que sea necesario revisar la visión detrás de Ilonani. Sin embargo, se estima que sería interesante continuar el trabajo realizado en este proyecto. Muchas falsas pistas y caminos sin salida han sido descubiertos, y tenerlos en cuenta en el futuro sería adecuado.

Una de las cosas que convendría hacer es estudiar la idoneidad de usar el Nokia N810 como plataforma. Aunque se consiguió elaborar una solución de video-conferencia, puede que sea buena idea investigar otras opciones, sobre todo teniendo en cuenta que la evolución en el mundo del hardware nunca descansa.

Si se decidiese mantener esta plataforma, habría que elegir qué arquitectura de comunicaciones sería la más adecuada. Este documento presenta los resultados de una investigación inicial, pero se estima que habría que investigar un poco más a fondo las opciones que parezcan más interesantes antes de elegir una como la definitiva.

Abstract

This document presents the results of the research done on the topic of developing a table-top robot for remote interaction. This table-top robot was conceived to include a Nokia Internet Tablet N810 as its “head and face”, concentrating the functions of main processor and interface with users. This N810 would sit on a base providing controlled “turn-and-tilt” movement through some microcontroller and servomechanisms. The work done under this project concentrates on the development of a solution that would enable video-conferencing between two Nokia Internet Tablets N810.

The document starts with a description of the context in which this robot is conceived: an interconnected world of smart and ambient devices where people expect to be able to tele-communicate anywhere, any time.

The work done under this project has focused on researching the Maemo Platform (used by the aforementioned Internet Tablets), it's frameworks and programming. This was done with the intention of evaluating the feasibility of building a video-conferencing system using this hardware.

Additionally, different possible video-conferencing architectures and servers have been researched briefly, in order to provide a stepping stone for future projects.

The results of the project are double: on one hand, a system was put in place to support video-conferencing (though not in the form anticipated); on the other, the biggest deliverable coming from the effort invested in this project corresponds to the knowledge that could be used in future projects and that is summarized in this document.

Chapter 1. Introduction

1.1. The vision

In the future, our homes, work and leisure places will be inundated by smart devices and systems embedded with connectivity capabilities [1]. It's already accepted as common the fact that most of the devices we carry or interact with are built with at least some kind of basic electronics including, in most cases, some type of programmable micro-controller. In the future, as these technologies get better and better, we can expect more powerful and functional devices.

Illustration 1: The internet of things [2]

These devices will be expected to interconnect between each other and to bigger telecommunication networks (like the Internet). They will make use of the concepts of ambient devices [3], providing information and interaction capabilities in a non-intrusive manner. We will become oblivious to the fact that they exist and will take their services for granted. Our mobile phone (that will have become our mobile computer by that time) will provide ubiquitous video-conferencing capabilities and will be able to guide us using GPS technologies. It will carry (or have access to) our whole music collection and will be able to talk to our car's computer and stream a play-list of songs custom made to match our current state of mind. Most probably, computing power, storage capacity and intelligent reasoning will be distributed among this assortment of smart objects. Their interconnection will be seamless and natural to us [1].

All this interconnection brings the possibilities for communication. Communication, as we say, between objects, between persons and between objects and persons. New unimaginable ways of Human Machine Interaction will be devised, tending to more natural and intuitive methods. It will be assumed that objects interact together and exchange information: information that empowers objects to provide better services to humans. Most probably, society will also be more open and communicative.

The availability of information will be ubiquitous. This is just a consequence of interconnectedness: if all things and systems are connected, they can share information better, which makes it available for more systems and persons and in more places. This information will not only be objective information about facts, news or business available in your local area, it will surely permeate to the realm of personal information. Already, people are sharing huge amounts of personal information through blogs and social networks, and this seems a trade-off for personalization (of services) that people are prepared to pay, at least until some extent.

New forms or entertainment will appear. The trend has already been set with sites like Youtube, Flickr and Legal Torrents, that offer user-created content to other users. Content will be created in a distributed manner, instead of the old way were a big content provider produces films and programmes that are then consumed by passive users. Content becomes user-originated and interactive, where users remix other users' contributions, respond or interact in some other way with already created content.

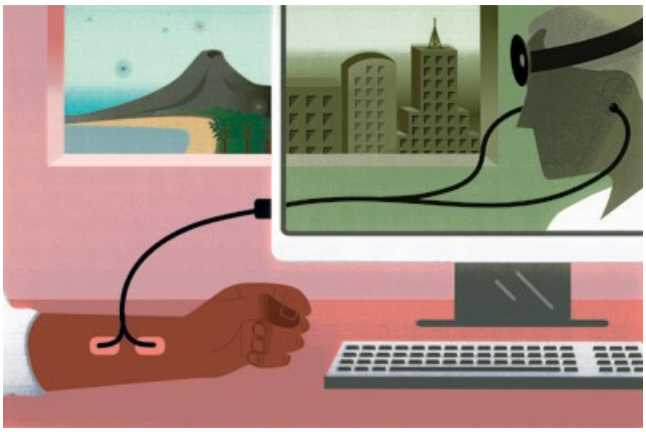

Illustration 2: "Telemedicine comes home" [4]

We can also expect the future to become the world of tele-everything. If things continue to develop in the same way, solving problems of CO^2 emissions, over-population, over-consumption of energy, etc. might be solved by providing services at home: tele-medicine will distribute good quality medical services to home, so patients don't have to transit to the

doctors office; tele-work will prevent everyday morning and afternoon rush hours, etc.

In the world of the future, devices like ILONANI will be playing a bigger and bigger role.

1.2. State of the art

NABAZTAG – SMART OBJECTS, AMBIENT DEVICES AND UBIQUITOUS COMPUTING

One of the projects that inspired the vision for this thesis was Nabaztag. Nabaztag is a rabbit-like consumer electronics device manufactured by Violet [5]. It can connect to the Internet through a wireless connection and be programmed and configured to do things like check and inform about weather forecasts, check email and read messages out loud, read RSS feeds, play streaming music and much more.

Illustration 3: Nabaztag bunnies [6]

Nabaztag is one of a new kind of consumer electronics. On one hand, it can be considered as an ambient device and, on the other, it's also a smart object. Both concepts are quite recent, and they promise huge potential for new applications and markets.

Ambient devices are a new genre of consumer electronics that make use of pre-attentive processing. These is the capacity of the brain to process information without having to pay explicit attention. These devices deliver information in a non-intrusive way. The idea behind these devices was conceived in MIT's MediaLab and resulted in a spin-off called Ambient Devices [7]. One of the products offered by this company is the Orb, which glows with different colours to display real time stock market trends, traffic congestion, weather forecasts or any other Ambient information channel (information streams provided by Ambient Devices).

Objects that have some electronics that make them smarter are called smart objects. Taking advantage of cheap electronics, objects can be

embedded with wireless connection capabilities, sensors and unique identifiers. This makes it possible to create “networked objects”, the internet of things. This creates a huge potential for new applications – for example, a smoke detector could ask all home appliances to turn off, maybe preventing the fire from getting worse; an intelligent fridge enabled to read RFID tags could alert the user about products that are about to go out of date or email a shopping list of items that have run out.

This brings us to another interesting trend. RFID tags seem to be a very nice way in which to create this network of things. Users can buy packs of tags that can be programmed and attached to objects to perform certain functions. For example, Violet's Ztamp:s [8] allow users to stick a tag on an umbrella and get a weather forecast on the computer when placing the tag in the range of the tag reader (Mir:ror [9]).

Telecommunication trends, social web and entertainment 2.0

We now take for granted that anyone can be contacted from anywhere and at any time, but it wasn't always like this. Before, *tele*-communicating was not such a common thing – if you wanted to communicate with someone, you normally set up a meeting and dealt with them face to face. Nowadays, technology is starting to enable the communication between persons that might be at many kilometres from each other. This opens up many applications, many of which already exist.

Illustration 4: 1896 telephone [10]

Even though the history of telecommunications could be considered to start with smoke signals and drums, for our purposes, we could trace it back to the invention of the first electrical telecommunication systems. The first one would be the telegraph, which made it possible to send messages using electrical signals between locations connected by an electrical cable. The first commercial telegraph started operation in 1839 over a twenty-one kilometres stretch. Some years later, in 1876,

Alexander Graham Bell invented the conventional telephone, and the first commercial telephone services were set up in 1878-79 both in the USA and the UK. Nevertheless, it wasn't until 1927 that telephone communications between both sides of the Atlantic were possible through a radio link, and until 1956 through the first transatlantic cable [11].

Through high-paced evolution, these primal systems have developed up to what we have today, a complex web of telecommunication systems and services that give users a vast array of possibilities to choose from. Nowadays, users can call a friend using a fixed phone on the PSTN or a mobile phone using GSM, check their email inbox from any computer, start Instant Messaging conversations with their contacts, send and SMS or MMS using their hand-held PDA or start up a voice/video conference using their jabber client. The selection is so huge that it's difficult to cover. Nevertheless, an effort to provide an overview of the current state will be made in the following lines.

Illustration 5: Social networking sites

The main thing that we should take into account in order to understand the current state of telecommunications is the idea of networks. A network, in this context, is a system of interconnected entities. With the invention of the telegraph, a network of lines through which to send messages was created. The invention of the telephone provided further push for the development of these telecommunication networks. Nowadays, these original networks, improved with new technology, provide the basis on with Internet, the network of networks, is built [12].

The trend will probably be to have more and more telecommunication services running on top of this network. To this regard, we can observe today applications like the following:

- Instant messaging programs that enable text chatting in almost real time.

- VoIP services that route voice calls through the packet-switching network with quite reasonable Quality of Service.

- Video streaming from content providers or from user to user, as new VoIP technologies enable the transmission of video as well as audio content.

These ubiquitous presence of telecommunication networks has provided fertile ground for the emergence of new social interactions. People are now able to communicate an interact in new ways. For example:

- Blogs: it's very easy to create a web-space in which to express your opinions or update the world on your latest musings.

- Myspace has proved to be a very popular way of communicating among teenagers and has surprised everybody for being adopted by musicians and artists of all levels.

- Social networks in the form of Facebook prove that people are willing to share their information and connect with new people.

- Instant messages, Voice over IP and email are quicker, smarter versions of the telegraph and the telephone, empowering many individuals to communicate with people around the globe.

- Sites like Youtube.com or LegalTorrents.com open the doors of new forms of communication in which users generate audiovisual content with artistic, entertaining or communicating purposes, changing the old

- Anyone can write and publish a book using sites like Lulu.com, that offer the service of on-demand publishing, making it possible for anyone to create and publish a book, even when the expected number of books to be sold is low.

Illustration 6: Second life logo [13]

- Online virtual worlds provide users with the possibility of creating a virtual personality, logging into a virtual world and interacting with virtual objects and personalities that are in turn controlled by other human beings.

These are just the point of the iceberg. Many more ways of communicating and interacting exist on the web and many more will see the light in future years. If we couple this with the idea of smart objects and ambient devices, it's clear that conceptions like Ilonani not only have a place in today's and tomorrow's world, we can expect to have them all around, in our homes, our working places and during our leisure time.

Added to this, the current trend in the software development industry of making applications available through a browser, web applications, will make it even more reasonable to think that all devices will be connected to the cloud, providing limitless possibilities of interacting and communicating.

TELE-ASSISTANCE

It's not new to us the fact that the population in developed countries is becoming older. This trend is expected to appear in developing countries as well. Aged societies will be a great load to social security systems and the tendency will be to search for solutions that enable social services in a cheaper and more efficient way [14].

If we turn our attention to new technologies like the ones mentioned in the previous section, it's clear that some of these solutions will involve telecommunication systems and networks. In this regard, one of the visions for Ilonani and similar devices is the use as user-friendly monitoring systems. Aged people need user-friendly devices with big buttons and intuitive interfaces. Carers could use devices like Ilonani to keep an eye on elderly people.

The use of these kind of devices will also provide useful for the area of tele-medicine. Having objects that allow us to interact and exchange information in ways that support services like tele-medicine and tele-assistance will surely be a trend to watch out in the near future.

Chapter 2. Ilonani, the concept

2.1. The concept and goals

Having the previous section in mind, the goal for the project started by designing and constructing a table-top robot providing a friendly user interface for elderly people to aid in the supply of tele-assistance services. The list of goals pursued has been:

- Research the possibility of using Nokia's N810 internet tablet as the platform for the robots “brain” and “face”. Finding out if video-conferencing is feasible (taking into account different application scenarios) and if it provides potential for a friendly enough user interface (given our requirement that it should be usable for elderly people).

- If the Nokia is sufficient, research what possible solutions exist, what different paths can be followed to create a working system and implement the best solution.

- Design and build a physical prototype of the robot, including a charging base, a case hosting all the necessary electronics and a turn-and-tilt mechanism.

The original vision also included the functionality of tracking the speaker with the camera but after some research, this was deemed too difficult. Additionally, the original project was split in two, so this project will actually take care of the first two goals, concerning all the issues around the video-conferencing solution and the N810. More information about the design and building of a physical prototype will be found in Gonzalo Zubieta's Final Project, which does not exist at the time of these writting.

Extensibility and basis for future work

Another objective that has been kept in mind is trying to develop a design that will be extensible. Since the beginning it has been obvious that a lot of great ideas (of development and application for Ilonani) would have to be set aside since it would not be possible to implement all of them in the time given. Pursuing this goal means that a lot of extra research has been done in order to set the basis for future projects wanting to build on this one. Hopefully, it also means that the information written in this document will be easy to follow and put into practice.

2.2. Blocks description

The system will consist of the blocks depicted in Illustration 7. As already stated, Nokia's N810 will be the main “brain” and “face” of the robot. It will have USB connection with an AVR microcontroller card used to generate the low level PWM signals to control the turn-and-tilt servos. Additionally, we will use a power card to concentrate the electronics needed to adapt and provide the supply lines to power the AVR microcontroller card and servos. The power card will also provide an interface with the battery (to recharge it or to draw power from it, depending if the robot is connected or not to a power line through the power connector). We also decided to power the Nokia tablet using it's own power connector, to make the system simpler and prevent any damage from occurring.

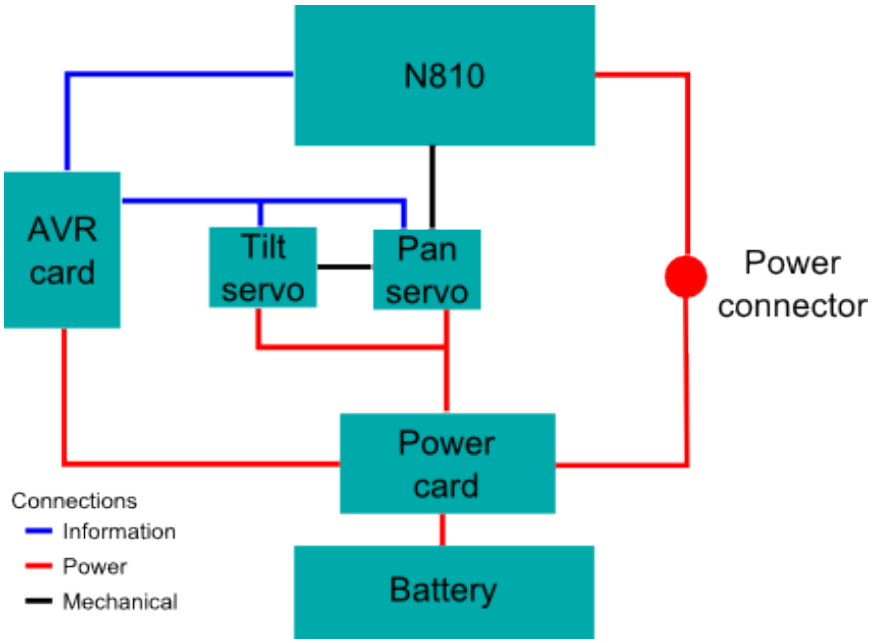

Illustration 7: Ilonani block diagram

Nokia's internet tablet, the N810

Nokia's N810 is an internet tablet, a kind of internet appliance (a device aimed mainly at accessing internet services) developed by Nokia under the N series. These internet tablets should not be considered as mobile phones, since they have no way of connecting to any cellular network typical for mobile phones. Nevertheless, they provide Wi-Fi and Bluetooth connectivity that enable access to the internet (with the latter, when used in conjunction with a web-enabled mobile phone) [15].

Historically, it's the third version of Nokia's product, preceded by the 770 and the N800. The first version, the 770, was presented in May of 2005, and received many critics, mainly due to its slowness and it's short

battery-life. This model evolved into the N800, introduced in January of 2007, which addressed many of the 770's shortcomings. That same year, in October, the N810 was released. A WiMAX enabled edition was announced on April of 2008, but it's production was cancelled in January of 2009, probably due to the low availability of WiMAX providers [16].

A list of the N810's technical specifications can be found in Table 1.

Size
- Volume: 128 cc
- Weight: 226 g
- Length: 72 mm
- Width: 128 mm
- Thickness: 14 mm

Display
- High-resolution 4.13” WVGA display (800 x 480 pixels) with up to 65,000 colours

Processor
- TI OMAP 2420, 400Mhz

Memory
- DDR RAM 128MB
- Flash 256MB

Storage
- Up to 2GB internal memory
- Support for compatible miniSD and microSD memory cards (with extender). Supports cards up to 8GB. (SD cards over 2GB must be SDHC compatible.)

Operating Times
- Battery: Nokia Battery BP-4L
- Continuous usage (display on, wireless LAN active): up to 4 hours
- Music playback: up to 10 hours
- Always online time: up to 5 days
- Standby time: up to 14 days

Other characteristics
- Smooth slide with integrated QWERTY keyboard
- Built-in GPS receiver
- High quality stereo speakers and sensitive microphone
- High-resolution wide-screen display
- Integrated desk stand
- Integrated VGA web camera
- HW key to lock touch screen and keys
- Ambient light sensor

Connectivity
- WLAN standard: IEEE 802.11b/g
- Bluetooth specification v.2.0 . +EDR (profiles supported: HID, FTP, DUN, GAP, SPP, HSP, SAP and OPP)
- USB high speed for PC connectivity
- 3.5 mm stereo headphone plug (Nokia AV Connector)

Table 1: N810's technical specifications [17]

The tablets run a modified version of Debian GNU/Linux called Maemo. It's a custom made operating system for Nokia Internet Tablets. Similarly to other tablet operating systems, it features a home screen with a menu

bar, an application-launching area and a customizable “desktop” where different applets, ranging from RSS readers to clocks, can be run.

It's heavily based on the GNOME project, from which it draws many libraries, GUI components and application frameworks (like GStreamer, used for multimedia handling). It uses Matchbox as window manager and the GTK-based Hildon application framework for GUI elements [20].

Maemo offers a very wide range of software applications – from telecommunication applications (for Internet, instant messaging and VoIP), office and media applications to games and a huge range of applications common in Linux desktop PCs.

As shown in Illustration 7, the N810 is intended to be connected with an AVR microcontroller card (developed in the Automation Technology Laboratory) using a USB interface. The goal is to implement some code to control the movement of the turn-and-tilt mechanism from the tablet. This code should be developed so that it will be embeddable in applications developed for the N810. For example, as part of the user interface, some virtual buttons could be presented on the screen to allow the user to pan or tilt the robot's head (which would be the Nokia itself). It could also allow the implementation of a tele-controlled system, where a remote user controls the movement of the system by sending commands to the software running on the tablet.

AVR MICROCONTROLLER BOARD

The AVR microcontroller board has been developed by Antti Karjalainen at the Department of Automation and Systems Technology of TKK. It features an AT90USB162 microcontroller from ATMEL.

Relevant to this discussion, it has already been successfully used within the Laboratory to control sets of servos from Linux machines. The only work left would be to adapt the systems and Linux drivers to the Maemo environment.

Additionally, using this board provides the potential to extend the system through the use of the Zigbee network developed under a different project within the Laboratory. More information in Gonzalo Zubieta's Final Project.

POWER SUPPLY BOARD

An idea considered has been to have all the power adaptation and electronics concentrated in one board. In an initial vision, this would include:

- Electronics that measure the charge of the battery: this gives the information to operate the low battery indicator and to operate the

recharging of the battery (that should stop when a certain voltage level is reached, that is, the battery is properly charged).

- Low battery LED indicator
- Drivers for servos
- Power adaptation to provide supply for AVR microcontroller board.
- Connector where a 220V AC to 12V DC transformer could be plugged in, to provide connection to the mains. Connectors for battery, AVR microcontroller board and servos.

Not all features might be implemented. For the final version and a more in depth description, go to Gonzalo Zubieta's Final Project.

Battery

From a preliminary study, the peak current demanded by each servo (when blocked) is around 1.3 Amperes. On the other hand, nominal consumption of the whole system was found to be around 1 Ampere. The initial specification required batteries to support 6 hours of normal operation before needing a recharge. Details will follow in Gonzalo Zubieta's Final Project.

Servos

The servos used to build the turn-and-tilt mechanism are the classical Futaba S3003. They provide an affordable and reliable solution. Using a PWM signal, they can be position controlled.

Illustration 8: S3003 servo

Dimension	40.4 mm x 19.8 mm x 36.0 mm
Weight	37.2 g
Operating Speed	0.23 sec/60° (4.8V) 0.19 sec/60° (6.0V)
Output torque	3.2 kg-cm (4.8V) 4.1 kg-cm (6.0V)

Table 2: Futaba S3003 specifications

Buttons

The idea of adding physical buttons was also pondered. On/off and movement buttons seem to be a minimum. They would add usability, but they would also make the system more complex.

Hardware buttons:

- Advantages: they are fixed and don't move while pressing.

- Disadvantages: more complicated to build, difficult to add multi-functionality, not so flexible

Software buttons:

- Advantages: they are more flexible, the information is easily captured, processed and sent to AVR microcontroller as commands. All of the “thinking” happens inside the tablet.
- Disadvantages: they move while being pressed (the tablet moves, so it's not so comfortable).

Another solution would be to implement the buttons via software. Each solution has it's own advantages and disadvantages.

Chapter 3. The software application

3.1. Introduction

The vision for the software application to be developed in this project included the following functionality:

- Reliable two-way videoconferencing
- Simple user interface:
  - To control and configure the videoconferencing
  - To control the movement of the projected “turn-and-tilt” mechanism

At the start of the project, it wasn't clear that videoconferencing with the tablet was possible. Apparently, no out of the box solution existed for this and several approaches had been already investigated within the Automation Technology Research Group [21]. Believing that the N810 should be capable of performing video-conference, a big effort was devoted to finding out how to.

This proved to be more difficult than expected. The result of the work *does* include a basic set up providing video-conferencing functionality but, as it will be explained in this chapter, not in the form initially envisioned.

A great deal of the effort invested in this project did not produce specific deliverables but was needed in order to acquire the knowledge that made the following steps possible. In order to save future projects from having to invest this effort again, this chapter attempts to summarize this knowledge.

The chapter starts with a brief overview of the platform, Maemo and the GStreamer framework, and a discussion of different approaches that could be followed to implement a video-conferencing system. A description of the system that was put into place is presented, followed by some user interface considerations made at the end of the chapter.

3.2. Exploring the platform

Maemo Platform is the name given to the the software stack developed for Nokia Internet Tablets. It includes the Maemo operating system and the Maemo Software Development Kit.

Maemo is the operating system for the Nokia Internet Tablet line of handheld computers. Similar to many handheld operating systems, it features a "Home" screen—the central point from which all applications and settings are accessed. The Home Screen is divided into areas for launching applications, a menu bar, and a large customisable area that can display information such as an RSS reader, an Internet radio player or a Google search box.

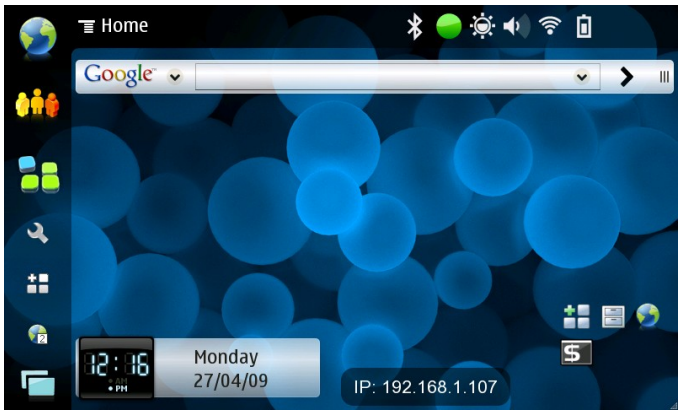

Illustration 9: Example home view on N810

Maemo is based on Debian GNU/Linux and draws much of its GUI, frameworks, and libraries from the GNOME project. It uses the Matchbox window manager, and the GTK-based Hildon as its GUI and application framework.

The Maemo SDK provides a sandboxed maemo development environment on a GNU/Linux desktop system. It is largely built on a tool called Scratchbox and, in most ways, it behaves like the operating system on the device, but with added development tools. This means that the development process is very similar to a normal desktop GNU/Linux, and the kinks of embedded development, such as cross-compiling, are handled transparently by Scratchbox. For more information on application development for Maemo, check Maemo Diablo Reference Manual [23].

Introduction to the GStreamer framework

GStreamer is a pipeline based multimedia framework programmed in the C language and using the GObject type system. The pipeline architecture enables the building a varied range of applications, from simple audio players to complicated video editors or streaming media broadcasters. GStreamer is part of the GNOME desktop environment and is included in Nokia's Maemo Platform [24].

The framework is based on plugins that provide various functionality. GStreamer provides an API for multimedia applications, a plugin

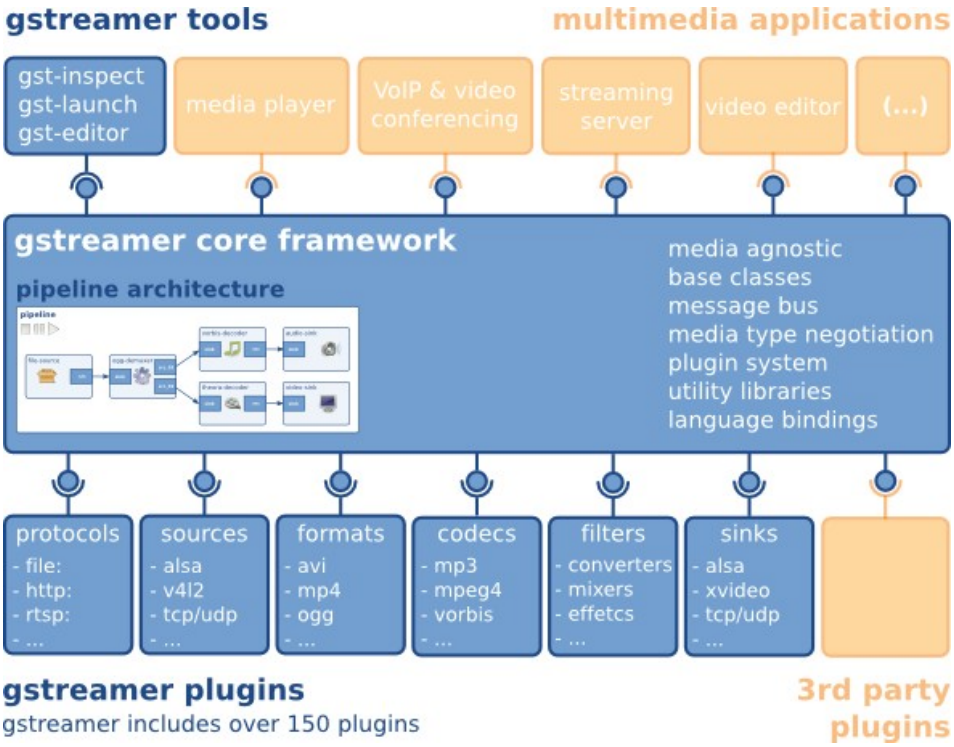

Illustration 10: GStreamer overview [26]

architecture, a pipeline architecture, a mechanism for media type handling/negotiation, over 150 plugins and a set of tools [25].

According to the official documentation, plug-ins can be classified into protocols handling, sources (for audio and video), formats (parsers, formaters, muxers, demuxers...), codecs, filters and sinks (for audio and video). Illustration 10 presents a graphical overview of the framework.

The three basic objects in GStreamer are elements, pads and bins. Elements are the most important class of objects – they provide some basic functionality (which could be reading of data from a file, decoding of this data, outputting this data to your sound card...) and are normally chained together (forming a pipeline) to provide some global functionality (like playing an audio file).

Pads provide elements' input and output. Pads have *capabilities* that define what kind of data can flow through them. Pads can be considered as plugs with different shapes (depending on their capabilities) that can be used to connect elements together. Two types of pads exist: *source* pads are used to output data from an element and *sink* pads are used to input data into an element.

Lastly, *bins* are containers for a collection of elements. They are used to simplify and provide structure. Since they are subclasses of elements themselves, they can also be controlled like elements [27].

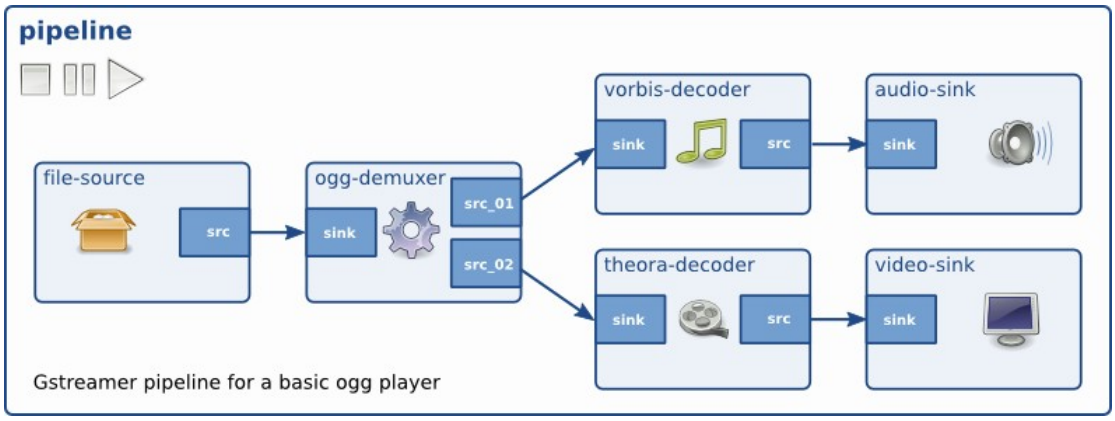

Illustration 11: GStreamer pipeline for a simple Ogg player [22]

Example application for camera operation

The Maemo Diablo Reference Manual [23] provides an example C program that captures video from the camera and outputs it to the screen. This simple program is still far from the concept of Ilonani's software application, but it will provide some basic understanding of how to work with GStreamer. The code provided in Appendix A. is an adapted version of the example provided with Maemo documentation [23].

The functions implemented in the program are:

- `gboolean key_press_cb(GtkWidget * widget, GdkEventKey * event, HildonWindow * window)`
- `static void bus_callback(GstBus *bus, GstMessage *message, AppData *appdata)`
- `static gboolean expose_cb(GtkWidget * widget, GdkEventExpose * event, gpointer data)`
- `static gboolean initialize_pipeline(AppData *appdata, int *argc, char ***argv)`
- `static void destroy_pipeline(GtkWidget *widget, AppData *appdata)`
- `void gui_initialize(HildonProgram **program, HildonWindow **window, int *argc, char ***argv, gchar *example_name):`
- `void gui_run(HildonProgram *program, HildonWindow *window)`
- `int main(int argc, char **argv)`

The initial version of the example used a stream format that produced unsatisfactory results. After a bit of tweaking, the end final format found to produce best results was:

```
caps = gst_caps_new_simple(
     "video/x-raw-rgb",
     "width", G_TYPE_INT, 352,
     "height", G_TYPE_INT, 288,
     "bpp", G_TYPE_INT, 24,
```

```
    "depth", G_TYPE_INT, 24,
    "framerate", GST_TYPE_FRACTION, 15, 1,
    NULL);
```

After setting up the SDK and compiling the executable, we need to transfer it to the tablet[1]. For this, we need to set up USB networking between the tablet and our Linux host. The steps that need to be followed are:

- Make sure that `rootsh`, `xterm` and `openssh` are installed on the tablet (check them out from the application catalogue or the maemo.org website).
- Open the terminal window on the tablet (Applications > Utilities > X Terminal) and execute the command `sudo gainroot`.
- On the tablet, edit `/etc/network/interfaces` so that it includes the following lines:

```
auto usb0
iface usb0 inet static
    address 192.168.2.15
    netmask 255.255.255.0
    gateway 192.168.2.14
```

- On the Linux host, make sure that the kernel has `usbnet` enabled and edit `/etc/network/interfaces` so that it includes the following lines:

```
mapping hotplug
    script grep
    map usb0
auto usb0
iface usb0 inet static
address 192.168.2.14
netmask 255.255.255.0
network 192.168.2.0
broadcast 192.168.2.255
up iptables -t nat -A POSTROUTING -o eth0 -s
192.168.2.15 -j MASQUERADE
up echo 1 > /proc/sys/net/ipv4/ip_forward
down iptables -t nat -D POSTROUTING -o eth0 -s
192.168.2.15 -j MASQUERADE
down echo 0 > /proc/sys/net/ipv4/ip_forward
```

- On the tablet's X Terminal, execute following (note that `insmod` might have to be executed twice):

```
# insmod /mnt/initfs/lib/modules/2.6.21-
omap1/g_ether.ko
# ifup usb0
```

- Connect the tablet to the host machine using the USB cable and test the connection using ping, e.g., from the host machine:

```
# ping 192.168.2.15
```

You will know be in a position to use `ssh` to run a remote terminal session from the Linux host or to transfer files to and from the tablet using `scp`. Once the executable is copied to the home directory in the tablet

1 For details on how to do this, check the Maemo Diablo Reference Manual [23]

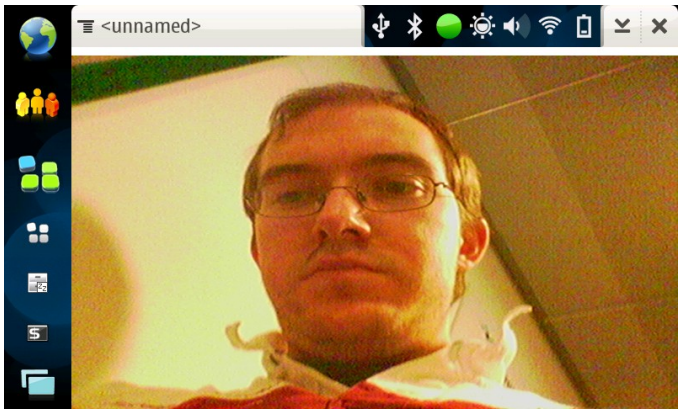

Illustration 12: Example application for camera operation

(`/home/user`), the program can be run. The result of running the program can be seen in the following screen-shot:

Using GStreamer tools to find the right pipeline

GStreamer provides a command line interface that allows for quick testing of pipeline combinations. To access this interface, `gst-launch` has to be used. The package `gstreamer-tools` should be installed on the tablet.

Capturing video from the camera and displaying it on the screen can be done easily using the following command[2]:

```
gst-launch v4l2src ! video/x-raw-yuv, width=640,
height=480, framerate=15/1 ! xvimagesink
```

This produces the same result as the C program presented on the previous section. This could make it seem that using this second approach is much better, since the code needed is much simpler and no compilation hassle has to be dealt with. For simple settings like the previous one, this will be the case, but producing a full-blown video-conferencing application will require turning to C programming.

This might become obvious when we look at the command needed to produce one way audio/video streaming. On the emitter:

```
gst-launch-0.10 gconfv4l2src ! video/x-raw-yuv,
width=640, height=480, framerate=\(fraction\)15/1 !
hantro4200enc stream-type=1 profile-and-level=1001 !
video/x-h263,framerate=\(fraction\)15/1 ! rtph263ppay
mtu=1438 ! application/x-rtp, clock-rate=90000 !
udpsink host=192.168.1.106 port=5434 dsppcmsrc ! queue
! audio/x-raw-int, channels=1, rate=8000 ! mulawenc !
rtppcmupay mtu=1438 ! udpsink host=192.168.1.106
port=5432
```

On the receiver:

2 For a detailed explanation of the `gst-launch` command, check Linux manual pages.

```
gst-launch-0.10 udpsrc port=5434 ! application/x-rtp,
clock-rate=90000 ! rtph263pdepay ! video/x-h263,
framerate=\(fraction\)15/1 ! hantro4100dec ! video/x-
raw-yuv, width=640, height=480, framerate=\
(fraction\)15/1 ! xvimagesink
```

The limits of producing video-conferencing using this command line interface lie on the fact that media negotiation is not possible. These command line tools are useful to determine the best combination of elements and parameters for our pipeline. Once these have been established, the pipeline should be coded into a proper Maemo application.

Conclusion

The work of this project didn't continue on this path due to lack of time – the end was coming closer and, even though the viability of following this path is clear, it would have required more time and knowledge to create an application using this approach.

After reaching this point, this project focused on finding some other approach that would provide video-conferencing even if it wasn't through a custom made application that complied with all of the details of the initial vision. To this respect, the rest of the chapter will be devoted to describing briefly some of the options that were explored and, in more detail, the option that was implemented at the end.

Should a future project continue the work of this one, the recommended line of action would be to start at this point: after understanding the GStreamer framework and having found a working pipeline, code the application that would support server-less point-to-point video-conferencing and then extend the system by supporting a server of some kind.

3.3. Different approaches for video-conferencing

The system envisioned defined a software application that would perform audio & video communication between two N810 (or between one N810 and any other device conceivable) in a simple, reliable and straightforward manner. The user interface provided on the N810 would be intuitive, user-friendly and clean.

This requirements definition leaves a lot of room for creativity regarding the system used to perform this communication. No appropriate off-the-self solutions were known at the start of this project, so researched focused on finding frameworks and platforms that would be convenient for developing a custom-made solution. The most noteworthy options explored will be discussed in the following text.

Server-less point to point video-conferencing

The first option explored was the one that seemed most basic, having a point to point communication established between tablets. One application that was doing almost this was Peekaboo [28], an open source application developed as a school project. Still, this application was developed for the N800 and included just basic one way streaming without NAT traversal [29]. At this point, research focused on establishing the feasibility of developing a new video-conferencing application from scratch.

Advantages	Disadvantages
Relatively easy to build No need to have a serve running on some external system	Not robust communications (would not work properly through NAT) Not completely automatized (users would have to manually exchange IP addresses)

Table 3: Evaluation of server-less point to point solution

The video-conferencing that was intended to be achieved consisted in using the GStreamer framework to stream audio and video from one tablet to a certain IP and port (which could be another tablet, a PC or any other capable device). This first solution wouldn't include NAT traversal either and would require users to manually exchange IP addresses and to introduce them in the application through some kind of user interface.

Using Icecast server

Icecast is a quite popular open source streaming server maintained by the Xiph.org Foundation (dedicated to producing free multimedia formats and tools, like the Ogg family of formats). This server is capable of streaming Vorbis and Theora content over HTTP and MP3, ACC and NSV over the SHOUTcast protocol [30].

Advantages	Disadvantages
Relatively easy to build Communication and streaming would be more robust Standard streaming format Many clients can plug-in to the stream at the same time Streaming based on proven technologies	Need for an external host running the server

Table 4: Evaluation of icecast solution

The setting that would need to be put in place would include the installation and configuration of an icecast server on some PC host and

the development of a client for the Maemo operating system. This client would only need to be able to capture both audio from the microphone and video from the camera (in a timely manner) and stream it to the icecast server. The stream coming from the other tablet could be visualised using any media player supporting these streaming formats [31].

USING JABBER/XMPP

This solution would be very similar to the one explained in the section Implemented solution. In this case, we would set up our own XMPP/Jabber server and would not have to rely on any external service like GTalk.

A popular Jabber/XMPP server is ejabberd, a very powerful, highly compliant, open source server written mostly in the Erlang language (designed to support distributed, fault-tolerant, soft real-time, non-stop applications). It runs on Windows and many Unix-like systems [32].

On the client side, we could still use Nokia's built-in client or develop our own. To develop our own client, we count with GStreamer, Telepathy and Farsight frameworks. Additionally, existing open source clients could be reused in the building of a custom-made application.

Additionally, we could use the Asterisk server in conjunction with the Jabber/XMPP server to greatly increase the functionality of the system. Asterisk is an open source software implementation of a telephone PBX (Private Branch eXchange) system. A PBX is a private telephone exchange or telephone switch, which has traditionally been a system of electronic components to connect telephone calls [34].

Advantages	Disadvantages
Communication and streaming would be more robust Standard streaming format Many clients can plug-in to the stream at the same time Streaming based on proven technologies Streaming would be pluggable with existing IM and VoIP services. Great potential for extensibility	Need for an external host running the server Need to learn the workings of GStreamer, Telapathy and Farsight frameworks.

Table 5: Evaluation of Jabber/XMPP solution

The name for Asterisk comes from the name of the symbol * and was used to reflect the fact that it was meant to be “the Swiss army knife for telecommunications”, like the wild-card in Unix systems. Owing to its name, Asterisk offers a very wide range of possibilities, from connecting

VoIP calls with PSTN lines (normal telephone lines) to handling Jabber communications.

3.4. Implemented solution

The final solution to implement the video-conferencing between tablets was obtained by using the built-in VoIP client and two GTalk accounts.

Nokia N810 tablets come with a built-in VoIP client. These clients support SIP and Jabber accounts, as well as Google Talk ones. Using the approach discussed in this section, in generic terms, we would just need:

- Choosing a certain protocol (SIP, Google Talk or Jabber).
- A server, which could be run by ourselves or by an external service provider [35].
- Setting up two accounts (both on the server an on each of the tablets' clients).

In 1998, Jeremie Miller started the Jabber project, aimed at setting up the technology to enable Instant Messaging and Presence using open-source software and open standards. In 2000, a first version of the first Jabber server, jabberd, was released.

This early Jabber protocol evolved to XMPP, which is (as Jabber was) an open, XML-based protocol aimed originally at instant messaging and presence information. This means that many projects developing both servers and clients have arisen [36].

Nowadays, XMPP has been extended (which shouldn't be a surprise, since the X is for eXtensible) to support voice/video or file transfer. Google, in collaboration with the XMPP Standards Foundation, developed Jinggle, to allow peer-to-peer signalling for these kind of multimedia interactions [37].

Following a different path we have SIP, which stands for Session Initiation Protocol. This is a TCP/IP based signalling protocol used for setting up and tearing down multimedia communication sessions. It fits into the application layer of the OSI reference model [38] (though it is sometimes placed under the session layer). SIP is quite general purpose and independent of the transport protocol used to send and receive the actual data (which could be TCP, UDP or SCTP). It's based on the same technologies that power the Internet, so it's certainly a good option to take into account.

In this implemented solution, we used a set up including Google accounts. Nevertheless, as the reader might have already noticed, it would have been quite straightforward to set up a private Jabber server running on the WLAN which would have prevented us from having to count on an external service provider. The resulting video-conferencing experience should remain the same to the user, whichever of this two

solutions is employed. Using an Asterisk [43] server with SIP, Jabber or even GTalk accounts could have worked also.

Many other possibilities exist and were explored in the research done for this project. The reason why the GTalk solution was chosen has to do with the resources available: after many unfruitful attempts, not many spare time was left and configuring this setting proved to be the quickest solution. A screenshot of the result can be seen on Illustration 8.

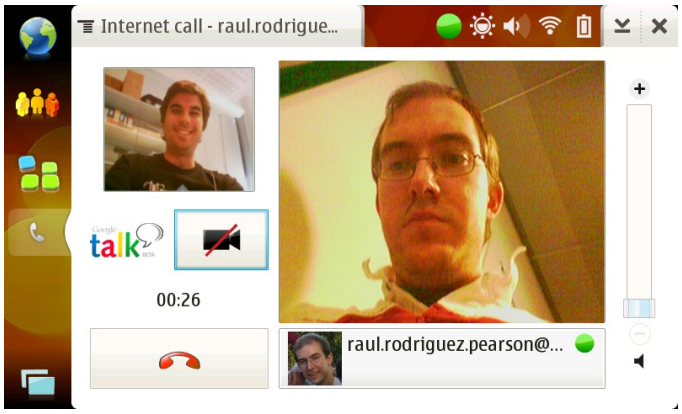

Illustration 13: Screenshot of implemented solution

3.5. Wrapping it up with some user interface

The initial vision for Ilonani was to have a clean and simple application including just the most important functionality. The intention was to have a big video display with simple and visible user interface, including big buttons. The concept looked something like .

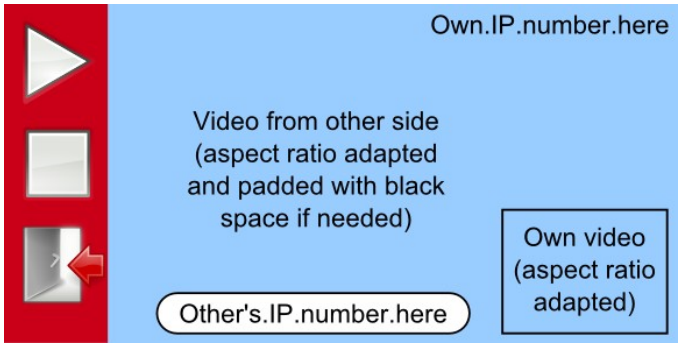

Illustration 14: Concept user interface

The concept presented in Illustration 9 should also include some space for the buttons that would be used to move the servos, if this solution was to be adopted (check the Buttons section on page 37 for more information). Additionally, some configuration window should be accessible through the drop-down menu usually available by clicking on the Application title area (check Illustration 17 on page 58 for the naming of the different areas).

Chapter 4. Conclusion

4.1. Results

The main results of the project are three:

- a working videoconferencing solution,
- the knowledge acquired and summarized in this document and
- source code that could used to build a custom-made application that would follow the vision outlined at the beginning of the project.

The videoconferencing system built does not look like the system conceived at the beginning, but it does provide videoconferencing with a very reasonable quality. It also requires very little development and works almost out of the box. The main drawbacks are that an external service provider needs to be used (i.e. Google) and that the built-in communications client that we use doesn't provide the kind of user interface that we were looking for.

The knowledge acquired pertains mainly to the exploration of videoconferencing possibilities using the N810 and different technologies. From the studied undertaken, it seems that two types of architecture could be put into place: server-less and server/client configurations. Advantages and disadvantages for both have been put forward in section 3.3. but, as a conclusion, we believe that a server/client architecture makes for a better solution. To this respect, a solution that was only briefly investigated (and hence, not included in this study) was the use of the GIMnet communications infrastructure [46]. We believe that this solution deserves some more investigation.

Finally, some source code is also included (Appendix B.). This code represents the initial research done following the line of server-less videoconferencing. It seemed at the moment to be a good idea, in order to get acquainted with the Maemo Platform. The idea was to quickly build on that code to create the application using some of the server architectures outlined later. As already mentioned, any projects wanting to continue the work of this project could start by building on top of this code. The reason no more effort was devoted to this part was that time was running out and it was deemed convenient to move onto different grounds in order to be able to provide a more comprehensive product at the end of the project.

Hindsight might make it look as though the progress done in this project was little and without any clear direction. This has some truth to it, but it

should be taken into account that many discoveries had to be made. On the first hand, it wasn't clear that videoconferencing was possible with the N810 (though it seem that it should); secondly, after searching the web, no reports were found of people trying to do this – all of the links consulted pointed to the use of basic GStreamer lines that would enable the construction of such a system. Only after thorough research, the different possibilities started to be uncovered. Additionally, not until the last month of the project, the possibility of performing videoconferencing with the suggested implemented solution was discovered. This was in part due to the belief bias [45], but also because of technical difficulties involving firewalls and routers at the place were the research was developed. Even the GTalk + built-in communications client solution was deemed not useful at some point in the research since it seems (but was not confirmed) that videoconferencing using this configuration is only supported between tablets and not between a table and a normal computer.

4.2. Future work

The vision of Ilonani, a tablet-top robot for remote interaction, might want to be revised. Nevertheless, if it's considered to be current when new work is to be undertaken, it might be useful to follow the work done on this project, since many false trails have been uncovered and much work could be reused.

On the other hand, some amount of work would have to be done on studying the convenience of using a different platform, different from the N810. This was considered at some point in this project, when it wasn't clear that videoconferencing was going to be achieved, but discarded due to lack of time to restart research on a different platform. As demonstrated with this work, it *is* possible to videoconference using this tablet. Nevertheless, Nokia's Internet Tablet does pose some other limitations that might want to be considered.

If a decision to keep using the N810 was made, the next thing to do would be to establish the kind of communication architecture that is best suited for the system. An initial research has been presented here, but a deeper investigation should be performed in order to evaluate each of the approaches, taking also into account the conditions particular to the situation (e.g., previous knowledge of the implementer, availability of time, etc.).

Finally, the code and experience summarized in this document could be reused in the coding of a final application, in order to save time and effort.

References

[1] Horizon Report 2009, http://www.nmc.org/pdf/2009-Horizon-Report-K12.pdf (checked April 30, 2009)

[2] TIME magazine Best inventions of 2008, http://www.time.com/time/specials/packages/article/0,28804,1852747_1854195_1854158,00.html (checked April 30, 2009).

[3] Ambient devices philosophy, http://www.ambientdevices.com/cat/philosophy.html (checked April 30, 2009)

[4] "Telemedicine comes home", The Economist Technology Quarterly June 5, 2008, http://www.economist.com/displaystory.cfm?story_id=11482580&CFID=52750040&CFTOKEN=80324836 (checked April 30, 2009)

[5] http://www.violet.net (checked April 30, 2009)

[6] http://www.nabaztag.com (checked April 30, 2009)

[7] http://ambientdevices.com (checked April 30, 2009)

[8] http://www.violet.net/_ztamps-rfid-tag-that-give-powers-to-your-objects.html (checked April 30, 2009)

[9] http://www.violet.net/_mirror-give-powers-to-your-objects.html (checked April 30, 2009)

[10] http://en.wikipedia.org/wiki/File:1896_telephone.jpg (checked April 30, 2009)

[11] http://en.wikipedia.org/wiki/History_of_telecommunication (checked April 30, 2009)

[12] http://en.wikipedia.org/wiki/Internet (checked April 30, 2009)

[13] http://secondlife.com/ (checked April 30, 2009)

[14] "The shape of things to come", Population Action International, http://www.populationaction.org/Publications/Reports/The_Shape_of_Things_to_Come/Summary.shtml (checked April 30, 2009)

[15] http://en.wikipedia.org/wiki/Internet_Tablet (checked April 30, 2009)

[16] http://en.wikipedia.org/wiki/Nokia_N810 (checked April 30, 2009)

[17] http://europe.nokia.com/find-products/devices/nokia-n810/specifications (checked April 30, 2009)

[18] http://europe.nokia.com/find-products/nseries (checked April 30, 2009)

[19] http://Maemo.org/intro/white_paper/ (checked April 30, 2009)

[20] http://en.wikipedia.org/wiki/Maemo_(operating_system) (checked April 30, 2009)

[21] 2008-A33-Omaishoitajan kuvapuhelinjärjestelmä, Simo Heimonen (2008)

[22] http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/section-intro-basics-bins.html (checked April 30, 2009)

[23] http://maemo.org/maemo_release_documentation/maemo4.1.x/ (checked April 30, 2009)

[24] http://en.wikipedia.org/wiki/Gstreamer (checked April 30, 2009)

[25] http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/ (checked April 30, 2009)

[26] http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/chapter-gstreamer.html (checked April 30, 2009)

[27] GStreamer Application Development Manual: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/index.html (checked April 30, 2009)

[28] http://peekaboo.garage.maemo.org (checked April 30, 2009)

[29] http://en.wikipedia.org/wiki/NAT_traversal (checked April 30, 2009)

[30] http://en.wikipedia.org/wiki/Icecast (checked April 30, 2009)

[31] http://maemo.org/downloads/OS2008/ (checked April 30, 2009)

[32] http://en.wikipedia.org/wiki/Ejabberd (checked April 30, 2009)

[33] http://en.wikipedia.org/wiki/Erlang_programming_language (checked April 30, 2009)

[34] http://wiki.jabber.org/index.php/XMPP_Case_Studies (checked April 30, 2009)

[35] http://xmpp.org/services/ (checked April 30, 2009)

[36] http://xmpp.org/tech/ (checked April 30, 2009)

[37] http://code.google.com/apis/talk/open_communications.html (checked April 30, 2009)

[38] http://en.wikipedia.org/wiki/Osi_reference_model (checked April 30, 2009)

[39] http://www.jabber.org/index.php/about/ (checked April 30, 2009)

[40] http://en.wikipedia.org/wiki/Session_Initiation_Protocol (checked April 30, 2009)

[41] Peter Saint-Andre [47] interviewed by Randal Schwartz and Leo Laporte: http://twit.tv/floss49/ (checked April 30, 2009)

[42] http://en.wikipedia.org/wiki/Instant_messaging (checked April 30, 2009)

[43] http://www.asterisk.org/ (checked April 30, 2009)

[44] N800 and N810 user guide: https://fjallfoss.fcc.gov/prod/oet/forms/blobs/retrieve.cgi?attachment_id=846889&native_or_pdf=pdf (checked April 30, 2009)

[45] http://en.wikipedia.org/wiki/Belief_bias

[46] GIM Integrator documentation, section 1.2.2. GIMnet: http://gim.tkk.fi/RP9/Documents?action=download&upname=Integrator-final.pdf (checked April 30, 2009)

[47] http://xmpp.org/xsf/people/stpeter.shtml (checked April 30, 2009)

[48] https://www.google.com/accounts/NewAccount (checked April 30, 2009)

Appendix A. User manual

The following appendix is heavily based on the N810 User's Guide [44].

A.1. Getting to know the system

After power-up and Nokia's splash-screen, the user will be presented with Maemo's home view. The touch-screen is divided into the following areas (Illustration 15):

1. Task navigator
2. Application title area
3. Status indicator area
4. Minimize and close buttons
5. Application area
6. Toolbar
7. Application switcher

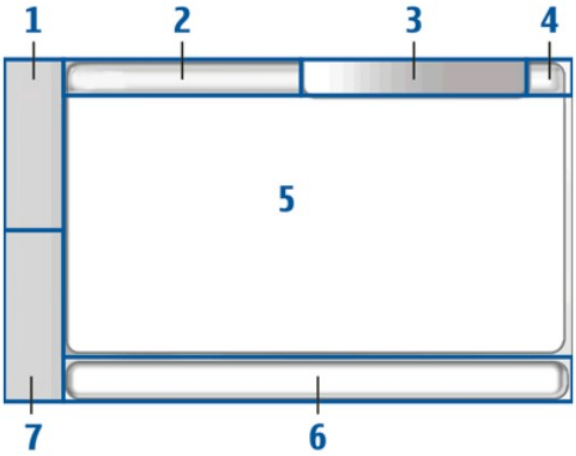

Illustration 15: Maemo touchscreen areas [44]

The user should familiarize himself with the platform by browsing through the Internet Tablet 2008 edition User Guide [44] and by playing around with the different buttons and menus.

A.2. Setting up the system

In order to get this solution running we need the following:

- Two Nokia Internet Tablets running the appropriate version of Maemo. In this project, two N810 running Maemo Diablo release number 5.2008.43-7 were used. An appropriate version of Maemo should include an up to date version of Nokia's built-in VoIP client.
- A working WLAN and a firewall/routing configuration that allows VoIP communications. In particular, it should allow authentication and use of GTalk accounts.

Setting up a connection

The first thing to do is to connect both tablets to the wireless LAN. Once the tablet has booted, click on the Application menu icon from the Task navigator.

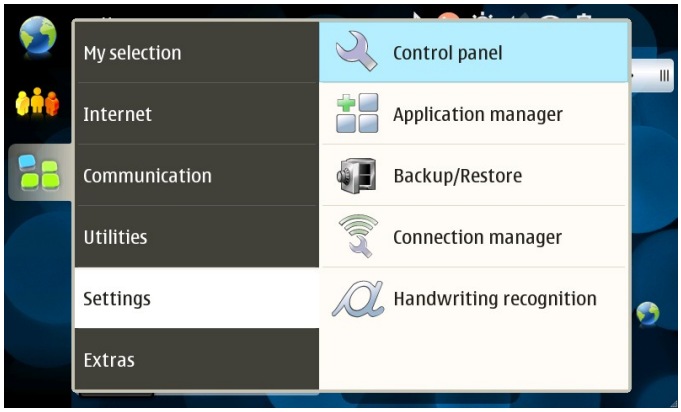

Illustration 16: Settings menu

In the menu that pops up (Illustration 16), select Settings > Control Panel and then, inside the Control Panel, select Connectivity. Connections > New, and follow the Connection setup wizard (Illustration 17).

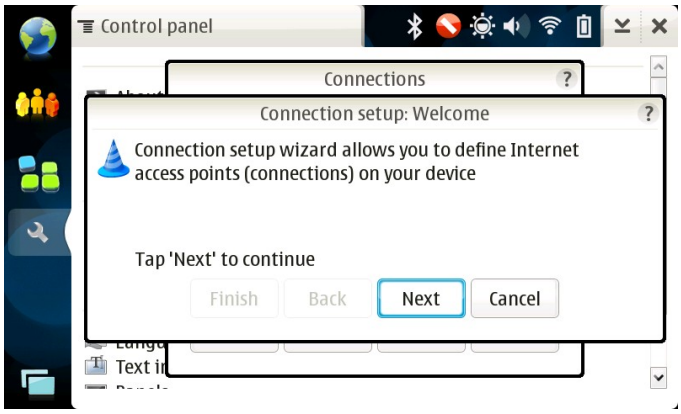

Illustration 17: Connection setup wizard

Setting up an account

Once both Nokia's have a working WLAN/Internet connection, you need to configure an account. Go back to the Control Panel and select Accounts > New and follow the Account setup wizard (.

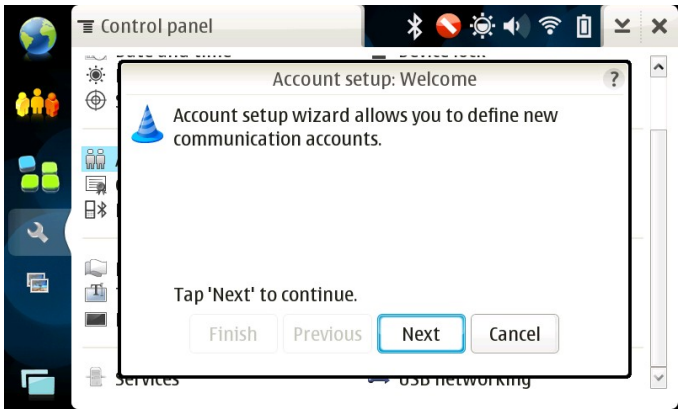

Illustration 18: Account setup wizard

For our proposed setup you will need two GTalk accounts. You can create a Google Talk account from the Account setup wizard or by accessing the Google account web page [48] on a web browser. You will also need to add each contact to each others' contact list (in order to do this, we recommend that you use GMail or GTalk themselves).

A.3. Using the system

To try out the system, make sure both tablets are connected and have managed to authenticate correctly with the GTalk server. Use one of the tablets to initiate the call by clicking on the Contacts icon in the Task navigator. Select New Internet call in the menu that pops up, select the right contact and press the phone button.

Appendix B. Camera operation example program

The following listing provides the code used to operate the camera. It captures video from the camera and outputs it to the screen.

```
#include <stdlib.h>

#include <string.h>

#include <gtk/gtk.h>

#include <gst/gst.h>

#include <gst/interfaces/xoverlay.h>

#include <hildon/hildon-banner.h>

#include <hildon/hildon-program.h>

#include <gdk/gdkkeysyms.h>


/* Define sources and sinks according to

 * running environment

 * NOTE: If you want to run the application

 * in ARM scratchbox, you have to change these*/

#ifdef __arm__

/* The device by default supports only

 * vl4l2src for camera and xvimagesink

 * for screen */

#define VIDEO_SRC "v4l2src"

#define VIDEO_SINK "xvimagesink"

#else

/* These are for the X86 SDK. Xephyr doesn't
```

```
 * support XVideo extension, so the application
 * must use ximagesink. The video source depends
 * on driver of your Video4Linux device so this
 * may have to be changed */
#define VIDEO_SRC "v4lsrc"
#define VIDEO_SINK "ximagesink"
#endif

/* Define structure for variables that
 * are needed thruout the application */
typedef struct {
  HildonProgram *program;
  HildonWindow *window;
  GstElement *pipeline;
  GtkWidget *screen;
  guint buffer_cb_id;
} AppData;

/* Callback for hardware keys */
gboolean key_press_cb(GtkWidget * widget, GdkEventKey * event,
                      HildonWindow * window)
{
  switch (event->keyval) {
  case HILDON_HARDKEY_UP:
    hildon_banner_show_information(GTK_WIDGET(window), NULL,
"Navigation Key Up");
    return TRUE;

  case HILDON_HARDKEY_DOWN:
```

```
    hildon_banner_show_information(GTK_WIDGET(window), NULL,
"Navigation Key Down");

    return TRUE;



  case HILDON_HARDKEY_LEFT:

    hildon_banner_show_information(GTK_WIDGET(window), NULL,
"Navigation Key Left");

    return TRUE;



  case HILDON_HARDKEY_RIGHT:

    hildon_banner_show_information(GTK_WIDGET(window), NULL,
"Navigation Key Right");

    return TRUE;



  case HILDON_HARDKEY_SELECT:

    hildon_banner_show_information(GTK_WIDGET(window), NULL,
"Navigation Key select");

    return TRUE;



  case HILDON_HARDKEY_FULLSCREEN:

    gtk_window_fullscreen(HILDON_WINDOW(window));

    hildon_banner_show_information(GTK_WIDGET(window), NULL, "Full
screen");

    return TRUE;



  case HILDON_HARDKEY_INCREASE:

    hildon_banner_show_information(GTK_WIDGET(window), NULL,
"Increase (zoom in)");

    return TRUE;



  case HILDON_HARDKEY_DECREASE:
```

```
    hildon_banner_show_information(GTK_WIDGET(window), NULL,
"Decrease (zoom out)");

    return TRUE;


  case HILDON_HARDKEY_ESC:

    hildon_banner_show_information(GTK_WIDGET(window), NULL,
"Cancel/Close");

    return TRUE;

  }


  return FALSE;

}


/* Callback that gets called whenever pipeline's message bus has

 * a message */

static void bus_callback(GstBus *bus, GstMessage *message, AppData
*appdata) {

  gchar *message_str;

  const gchar *message_name;

  GError *error;


  /* Report errors to the console */

  if(GST_MESSAGE_TYPE(message) == GST_MESSAGE_ERROR) {

    gst_message_parse_error(message, &error, &message_str);

    g_error("GST error: %s\n", message_str);

    g_free(error);

    g_free(message_str);

  }


  /* Report warnings to the console */
```

```
  if(GST_MESSAGE_TYPE(message) == GST_MESSAGE_WARNING) {
    gst_message_parse_warning(message, &error, &message_str);
    g_warning("GST warning: %s\n", message_str);
    g_free(error);
    g_free(message_str);
  }
  /* See if the message type is GST_MESSAGE_APPLICATION which means
   * that the message is sent by the client code (this program) and
   * not by gstreamer. */
  if(GST_MESSAGE_TYPE(message) == GST_MESSAGE_APPLICATION) {
    /* Get name of the message's structure */
    message_name =
gst_structure_get_name(gst_message_get_structure(message));

    /* The hildon banner must be shown in here, because the bus
callback is
     * called in the main thread and calling GUI-functions in
gstreamer threads
     * usually leads to problems with X-server */
  }
}

/* Callback to be called when the screen-widget is exposed */
static gboolean expose_cb(GtkWidget * widget, GdkEventExpose * event,
gpointer data) {

  /* Tell the xvimagesink/ximagesink the x-window-id of the screen
   * widget in which the video is shown. After this the video
   * is shown in the correct widget */
  /*  gst_x_overlay_set_xwindow_id(GST_X_OVERLAY(data),
```

```
      GDK_WINDOW_XWINDOW(widget->window));*/
  return FALSE;
}

/* Initialize the the Gstreamer pipeline. Below is a diagram
 * of the pipeline that will be created:
 *
 *                              |Screen|  |Screen|
 *                           ->|queue |->|sink  |-> Display
 * |Camera|  |CSP   |  |Tee|/
 * |src   |->|Filter|->|   |\  |Image|   |Image |  |Image|
 *                           ->|queue|-> |filter|->|sink |-> JPEG
file
 */
static gboolean initialize_pipeline(AppData *appdata,
              int *argc, char ***argv) {
  GstElement *pipeline, *camera_src, *screen_sink, *image_sink;
  GstElement *screen_queue, *image_queue;
  GstElement *csp_filter, *image_filter, *tee;
  GstCaps *caps;
  GstBus *bus;



  /* Initialize Gstreamer */
  gst_init(argc, argv);

  /* Create pipeline and attach a callback to it's
   * message bus */
  pipeline = gst_pipeline_new("test-camera");
```

```
  bus = gst_pipeline_get_bus(GST_PIPELINE(pipeline));
  gst_bus_add_watch(bus, (GstBusFunc)bus_callback, appdata);
  gst_object_unref(GST_OBJECT(bus));

  /* Save pipeline to the AppData structure */
  appdata->pipeline = pipeline;

  /* Create elements */
  /* Camera video stream comes from a Video4Linux driver */
  camera_src = gst_element_factory_make(VIDEO_SRC, "camera_src");
  /* Colorspace filter is needed to make sure that sinks understands
   * the stream coming from the camera */
  csp_filter = gst_element_factory_make("ffmpegcolorspace",
"csp_filter");
  /* Tee that copies the stream to multiple outputs */
  tee = gst_element_factory_make("tee", "tee");
  /* Queue creates new thread for the stream */
  screen_queue = gst_element_factory_make("queue", "screen_queue");
  /* Sink that shows the image on screen. Xephyr doesn't support
XVideo
   * extension, so it needs to use ximagesink, but the device uses
   * xvimagesink */
  screen_sink = gst_element_factory_make(VIDEO_SINK, "screen_sink");
  /* Creates separate thread for the stream from which the image
   * is captured */
  image_queue = gst_element_factory_make("queue", "image_queue");
  /* Filter to convert stream to use format that the gdkpixbuf
library
   * can use */
```

```
  image_filter = gst_element_factory_make("ffmpegcolorspace",
"image_filter");

  /* A dummy sink for the image stream. Goes to bitheaven */

  image_sink = gst_element_factory_make("fakesink", "image_sink");


  /* Check that elements are correctly initialized */

  if(!(pipeline && camera_src && screen_sink && csp_filter &&
screen_queue

       && image_queue && image_filter && image_sink)) {

    g_critical("Couldn't create pipeline elements");

    return FALSE;

  }


  /* Set image sink to emit handoff-signal before throwing away

   * it's buffer */

  g_object_set(G_OBJECT(image_sink),

         "signal-handoffs", TRUE, NULL);


  /* Add elements to the pipeline. This has to be done prior to

   * linking them */

  gst_bin_add_many(GST_BIN(pipeline), camera_src, csp_filter,

        tee, screen_queue, screen_sink, image_queue,

        image_filter, image_sink, NULL);


  /* Specify what kind of video is wanted from the camera */

  caps = gst_caps_new_simple("video/x-raw-yuv",

            "width", G_TYPE_INT, 352,

            "height", G_TYPE_INT, 288,

            NULL);
```

```
  /* Link the camera source and colorspace filter using capabilities
   * specified */
  if(!gst_element_link_filtered(camera_src, csp_filter, caps)) {
    return FALSE;
  }
  gst_caps_unref(caps);

  /* Connect Colorspace Filter -> Tee -> Screen Queue -> Screen Sink
   * This finalizes the initialization of the screen-part of the
pipeline */
  if(!gst_element_link_many(csp_filter, tee, screen_queue,
screen_sink, NULL)) {
    return FALSE;
  }

  /* gdkpixbuf requires 8 bits per sample which is 24 bits per
   * pixel */
  caps = gst_caps_new_simple("video/x-raw-rgb",
            "width", G_TYPE_INT, 352,
            "height", G_TYPE_INT, 288,
            "bpp", G_TYPE_INT, 24,
            "depth", G_TYPE_INT, 24,
            "framerate", GST_TYPE_FRACTION, 15, 1,
            NULL);

  /* Link the image-branch of the pipeline. The pipeline is
   * ready after this */
  if(!gst_element_link_many(tee, image_queue, image_filter, NULL))
return FALSE;
```

```
  if(!gst_element_link_filtered(image_filter, image_sink, caps))
return FALSE;


  gst_caps_unref(caps);


  /* As soon as screen is exposed, window ID will be advised to the
sink */
  g_signal_connect(appdata->screen, "expose-event",
G_CALLBACK(expose_cb),
        screen_sink);


  gst_element_set_state(pipeline, GST_STATE_PLAYING);


  return TRUE;
}


/* Destroy the pipeline on exit */
static void destroy_pipeline(GtkWidget *widget, AppData *appdata) {
  /* Free the pipeline. This automatically also unrefs all elements
   * added to the pipeline */
  gst_element_set_state(appdata->pipeline, GST_STATE_NULL);
  gst_object_unref(GST_OBJECT(appdata->pipeline));
}


/* Initialize the gui by creating a HildonProgram
 * and HildonWindow */
void gui_initialize(HildonProgram **program, HildonWindow **window,
         int *argc, char ***argv, gchar *example_name) {
  g_thread_init(NULL);
```

```
  /* Initialize GTK+ */
  gtk_init(argc, argv);

  /* Create HildonProgram and set application name */
  *program = HILDON_PROGRAM(hildon_program_get_instance());
  g_set_application_name(example_name);

  /* Create the toplevel HildonWindow */
  *window = HILDON_WINDOW(hildon_window_new());

  /* Connect destroying of the main window to gtk_main_quit */
  g_signal_connect(G_OBJECT(*window), "delete_event",
        G_CALLBACK(gtk_main_quit), NULL);
}

void gui_run(HildonProgram *program,
        HildonWindow *window)
{
  /* Show the window and widgets it contains
   * and go to the main loop. */
  gtk_widget_show_all(GTK_WIDGET(window));
  gtk_window_fullscreen(GTK_WIDGET(window));
  gtk_main();
}

int main(int argc, char **argv) {
  AppData appdata;
  GtkWidget *hbox, *vbox;
```

```
/* Initialize and create the GUI */
gui_initialize(&appdata.program, &appdata.window,
    &argc, &argv, "Camera example");

vbox = gtk_vbox_new(FALSE, 0);
hbox = gtk_hbox_new(FALSE, 0);
gtk_box_pack_start(GTK_BOX(hbox), vbox, FALSE, FALSE, 0);
appdata.screen = gtk_drawing_area_new();
gtk_widget_set_size_request(appdata.screen, 500, 380);
gtk_box_pack_start(GTK_BOX(vbox), appdata.screen, FALSE, FALSE, 0);

/* Add hardware button listener to application */
g_signal_connect(G_OBJECT(appdata.window),
      "key_press_event", G_CALLBACK(key_press_cb), appdata.window);

/* Connect signal to X in the upper corner */
g_signal_connect(G_OBJECT(appdata.window), "delete_event",
      G_CALLBACK(gtk_main_quit), NULL);

/* Initialize the GTK pipeline */
if(!initialize_pipeline(&appdata, &argc, &argv)) {
  hildon_banner_show_information(GTK_WIDGET(appdata.window),
          "gtk-dialog-error",
          "Failed to initialize pipeline");
}
g_signal_connect(G_OBJECT(appdata.window), "destroy",
      G_CALLBACK(destroy_pipeline), &appdata);
```

```
  /* Begin the main application */
  gui_run(appdata.program, appdata.window);
  /* Free the gstreamer resources. Elements added
   * to the pipeline will be freed automatically */
  return 0;
}
```