Sistemas Informáticos

Curso 2009-2010

SmartRoom

Arquitectura para Instalaciones Artísticas

Elías Boukamza Unsain_
Ignacio Gil Iza_
Eduardo Gil Ruiz_

Dirigido por:
Juan Pavón Mestras
Dep. Ingeniería de Software e Inteligencia Artificial

Concesión de Derechos

Los abajo firmantes autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria como la documentación y/o el prototipo desarrollado.

Elías Boukamza Unsain	Ignacio Gil Iza	Eduardo Gil Ruiz

En Madrid a de de 2010.

Agradecimientos

Queremos agradecer a Juan Pavón, nuestro director de proyecto, su ayuda y disposición este año para la realización de este proyecto.

También queremos agradecer al colectivo de artistas MASDELOMISMO [1] por su inestimable ayuda y compartir su tiempo para darnos ideas en las que poder trabajar.

Por último queremos dar gracias a Gabriel Aranda. Cada integrante del grupo quiere agradecer a su familia y amigos por aguantar todo este año.

Resumen

SmartRoom es una aplicación que permite relacionar la percepción de estímulos procedentes de sensores con la reproducción de contenido multimedia utilizando reglas.

SmartRoom nace con el propósito de acercar el uso de sensores al mundo del arte digital. Permite construir instalaciones artísticas con nuevos tipos de experiencias en la interacción del visitante con la obra de arte. Para ello se podrán programar, de manera simple e intuitiva, una serie de reglas que ante determinados estímulos ejecuten tareas audiovisuales.

El uso de reglas hace que la instalación tenga una cierta inteligencia y una respuesta frente a estímulos externos. Además, la plataforma SmartRoom permite tener todos los sistemas conectados en red lo que ofrece la posibilidad de tener todos los elementos distribuídos pero siendo manejados de manera centralizada.

Palabras clave: Sensores, Reglas, Instalación Artística, SmartRoom, Drools, Corba, JMF, Processing, Arte.

Abstract

SmartRoom is an application that allows the connection of messages from sensors to the playback of multimedia content using rules.

SmartRoom was born with the purpose of approaching the use of sensors in Digital Art. This proyect let you build Artistic installations with new types of experiences between the interaction of the visitor and the work of art. It´s possible to program, in an easy and intuitive way, a set of rules that triggers media events when receives an especific message.

Using rules provides some intelligence to the installation and makes it able to respond to external messages. In addition, each peace of the application will be connected to a network, having all the elements being distributed but centrally managed.

Key Words: Sensors, rules, artistic installation, SmartRoom, Drools, Corba, JMF, Processing, Art.

Índice General

CAPÍTULO 1

1.Introducción

El concepto de instalación artística ha ido evolucionando a lo largo del tiempo. Empezó a utilizarse el término en el mundo del arte a finales de los años sesenta.

Como se explica en el libro "Instalaciones " de Josu Larrañaga [2], el término 'instalación' en el arte ha sido utilizado con numerosas interpretaciones y muchas de ellas contradictorias. Desde afirmar que era otro tipo de arte plástica (como la pintura o escultura), a compararlo con el teatro o situarlo entre la arquitectura, el propio teatro , la escultura y la pintura.

En el mismo libro se comenta también que "(la instalación) *surge para nombrar una determinada manera de proponer arte, y no de definirlo.[...] No es más que una de las posibilidades expresivas nacidas de las extraordinarias transformaciones del arte de las últimas décadas*".

Por tanto, no se debe buscar una definición cerrada y correcta. Sería más adecuado utilizar una serie de consejos y conceptos como guía.

En este proyecto, una instalación artística se entiende como una disposición de elementos en el espacio, creando con varios objetos una obra de arte. Con entornos distintos se crea una experiencia diferente a si la instalación no estuviera allí. Se relaciona con el lugar en el que se encuentran, modificándolo mediante el comportamiento de la instalación y la experiencia que provoca cambia con él.

En la actualidad, muchos artistas se van planteando la utilización de medios digitales como herramientas útiles de expresión. Lo que antes se hacía mediante elementos analógicos (como antiguas grabaciones de vídeo por ejemplo), hoy en día se pueden realizar, con menor esfuerzo, tanto personal como económico.

Los elementos básicos de toda instalación artística son sensoriales. El proyecto, al que nos referiremos con el nombre de *SmartRoom*, influye sobre la vista y el odio. Se centra en facilitar la definición de los elementos que constituyen la instalación y sus relaciones, dejando a un lado el mensaje artístico y proporcionando un plataforma para que el artista la utilice.

1.1.Motivación

a) Artística

Desde el punto de vista artístico, el motivo del proyecto SmartRoom es la inclusión de herramientas que faciliten la definición de una instalación artística.

Dejar que los artistas creen sin preocuparse por los asuntos técnicos. Ofrecer una interfaz gráfica de fácil manejo para crear sus propias reglas, siendo éstas las que decidirán lo que debe pasar dentro de su obra. Se pretende investigar hasta dónde puede llegar un artista con nuestra herramienta, utilizándola y creando un mensaje.

Otro de los motivos por los que este proyecto es atractivo es la participación desde su inicio de los usuarios finales del sistema. Se ha contado con la colaboración activa del colectivo de artistas MASDELOMISMO.

Han sido muy importantes las distintas reuniones con ellos, estudios de requisitos, validaciones por su parte e investigaciones de posibles nuevas tecnologías por la nuestra. El conocer las necesidades del usuario de primera mano ha ayudado al diseño de los objetivos y al desarrollo de los mismos.

b)Técnica

Desde el punto de vista técnico, se pretenderá reutilizar componentes ya existentes e integrarlos en una plataforma que facilite la configuración de un sistema distribuido de dispositivos sensores y actuadores, así como el procesamiento inteligente de las señales. Incorporar distintos componentes y paquetes software para dar un entorno de fácil instalación y uso.

El uso de técnicas de inteligencia artificial viene dado por la necesidad de razonar sobre las señales que llegan. A partir de éstas, elegir qué actuador ejecutará qué acciones. Debido a las necesidades del problema, se ha optado por el uso de sistemas de razonamiento basados en reglas.

La inteligencia artificial tendrá una relación estrecha con los procesos distribuidos. La separación de ambos viene dada por la funcionalidad. Uno razonará sobre la información que le llega mientras que otro se encargará de que esa información llegue a su destino, sin interpretarla.

Se utilizarán estos procesos distribuidos (CORBA) para comunicar los distintos módulos software con tecnologías multimedia como video-cámaras, micrófonos, pantallas, detectores de movimientos,...

Además de utilizar lenguajes de programación como JAVA, DROOLS o XML, se integrará PROCESSING en varios apartados del mismo.

1.2. *Objetivos del trabajo*

El objetivo principal del trabajo es proporcionar una herramienta para ayuda al desarrollo y configuración de instalaciones artísticas con dispositivos sensores y actuadores. Debido a la naturaleza de este tipo de instalaciones se ha considerado la posibilidad de tener un sistema distribuido que facilite su realización. Se pretende poder integrar diversos tipos de sensores y varios actuadores bajo una gestión centralizada.

Esta gestión incluirá la creación de reglas que relacionen unos con otros. El diseño de este tipo de reglas es natural (del tipo "si...entonces...") lo que hace que para personas no familiarizadas con entornos de programación no sea demasiado complicado plasmar sus ideas con cierta facilidad. En este punto deberá tenerse en cuenta la realización de una interfaz gráfica para el usuario que resulte lo más sencilla posible.

El hecho de integrar diversos elementos como son sensores, actuadores con tecnologías muy diferentes requerirá tener presente la gran cantidad de errores posibles. Además se deberá prestar especial atención en este aspecto a todo el sistema de comunicación. Será necesaria la realización de una amplia batería de pruebas frente a los escenarios posibles.

El funcionamiento final de la instalación será del tipo acción-reacción por lo que deberá funcionar de una manera fluida sin que aparezcan tiempos de espera o retardos. Esto podría afectar en la percepción de la obra del visitante obteniendo un resultado insatisfactorio.

Es necesario conseguir un cierto grado de ampliación con la inclusión en un futuro de nuevos tipos de sensores y que ello no suponga un gran impacto en la implementación de la aplicación.

Para todo esto, se definen los siguientes objetivos concretos:

- Definición y desarrollo de un sistema de comunicación que permita que la gestión de los distintos elementos sea centralizada en uno sólo, pero a su vez, cada elemento de esta red tenga autonomía propia.
- Definición y desarrollo de un sistema de comunicación que estandarice el envío de los estímulos detectados por los sensores.
- Definición y desarrollo de un sistema de control y decisión para controlar las acciones llevadas a cabo en el sistema sobre el entorno, haciendo uso de

máquinas de estados y sistemas de razonamiento basados en reglas. Además debe permitir la creación y gestión de reglas.

- Definición y desarrollo de una arquitectura lo suficientemente flexible que permita la inclusión de nuevos tipos de sensores.
- Definición y desarrollo de una arquitectura lo suficientemente flexible que permita la inclusión de nuevos dispositivos sobre los que ejecutar acciones (medios).

1.3. Plan de Trabajo

Para abordar los objetivos propuestos en el apartado anterior se ha seguido el siguiente plan de trabajo.

- Investigación sobre instalaciones artísticas realizadas y de los elementos empleados en ellas.
- Investigación sobre tipos de sensores disponibles en el mercado y su posible gestión e integración mediante software.
- Investigación de sistemas de comunicación que permitan conectar unos elementos del sistema con otros.
- Búsqueda de librerías que proporcionen métodos para reproducir el contenido multimedia.
- Búsqueda e investigación de sistemas gestores de reglas.
- Implementación de un pequeño prototipo del sistema que sirva para familiarizarse con el uso de todas las librerías y elementos reutilizados.
- Realización de la especificación y la arquitectura del sistema siguiendo los métodos de Ingeniería de Software que permitan hacer un buen diseño de la aplicación.
- Implementación siguiendo la estructura diseñada en el punto anterior.
- Realización de pruebas ante posibles escenarios y comprobación de errores.

1.4.Estructura de la Memoria

Esta memoria está dividida en los siguientes capítulos:

- **Capítulo 1** Esta misma introducción.
- **Capítulo 2** Se detalla el por que de las elecciones hechas y las tecnologías actuales en cada campo.
- **Capítulo 3** Se detalla la arquitectura general del sistema así como su la arquitectura específica de cada sistema.
- **Capítulo 4** Se muestra como se organiza y ejecuta la comunicación entre los distintos sistemas.
- **Capítulo 5** Se muestran ejemplos de instalaciones artísticas concretas y una guía para llevarlas a cabo.
- **Capítulo 6** Se realizan conclusiones sobre el trabajo realizado y un esquema de mejoras posibles e implementaciones futuras.

CAPÍTULO 2

2. Estado del Arte

En este capítulo se realiza una pequeña revisión a la mayoría de ámbitos en los que se mueve la aplicación. Estos ámbitos van desde las instalaciones artísticas más innovadoras actualmente hasta las técnicas de captura de movimientos mediante video-cámaras.

El grado de desarrollo de estas tecnologías, su acceso con licencias OPEN SOURCE o sus usos estandarizados han sido parte de los motivos por los que se ha decidido utilizar ciertas tecnologías en vez de otras.

En este proyecto, se separará el Estado del Arte en dos partes:

- Revisión de las tecnologías y software existente para controlar sensores o medios y procesar la información de éstos.
- Herramientas disponibles actualmente para ayudar a los artistas a desarrollar sus instalaciones.

Los principales campos considerados en el trabajo son los siguientes: Sistemas de Comunicación, Motores de reglas y Sistemas de detección. Al final de este capítulo se incorporará un apartado de conclusiones en el que se justificará la elección de cada una de las tecnologías.

Por otro lado, se verán las herramientas utilizadas en instalaciones por parte de los artistas y hacia donde pueden evolucionar.

2.1.Sistemas de Comunicación

Para alcanzar el objetivo de comunicar varios sistemas existen en el mercado muchas soluciones. Se ha realizado una amplia investigación para elegir una tecnología que cumpliera con todos los requisitos manteniendo una estructura genérica y extensible.

Debido al propósito del proyecto de relacionar percepción y acción, uno de requisitos que deben satisfacerse en este apartado es que los intercambios de mensajes entre los sistemas se produzca sin retardos y sea tolerante a fallos.

Por un lado se han investigado librerías basadas en el protocolo OSC, por otro middelware de objetos distribuidos, CORBA e ICE.

a)Open Sound Control

En una primera investigación, se estudió Open Sound Control (o OSC) [3]. Es un protocolo de comunicación entre ordenadores, sintetizadores de sonido y dispositivos multimedia, pensado para compartir información musical en tiempo real sobre una red.

Las ventajas más visibles incluyen la interoperabilidad, precisión, flexibilidad y facilidad para la organización y documentación.

A pesar de ser un protocolo de los considerados 'sencillos', tiene un gran poder, dando toda la funcionalidad que se necesita para el control de sistemas en tiempo real que utilicen audio u otros tipos de media. Todo esto mientras mantienen la flexibilidad y facilidad de implementación.

En un principio fue ideado para hacer frente a MIDI, superándolo en características y capacidades. Las cualidades que distinguen a este protocolo son:

- Ampliable, dinámico. Esquema de nombres simbólicos tipo URL
- Datos numéricos simbólicos y de alta resolución
- Lenguaje de coincidencia de patrones, para especificar múltiples receptores de un único mensaje.
- Marcas de tiempo de alta resolución.
- Mensajes “empaquetados” para aquellos eventos que deben ocurrir simultáneamente
- Sistema de interrogación para encontrar dinámicamente las capacidades de un servidor OSC y obtener documentación

Las áreas más importantes en las que se está aplicando éste protocolo son:

- Sensores basados en gestos para instrumentos musicales.
- Mapear datos no musicales a sonido.
- Interfaces Web
- Realidad Virtual
- Aplicaciones para el IPhone

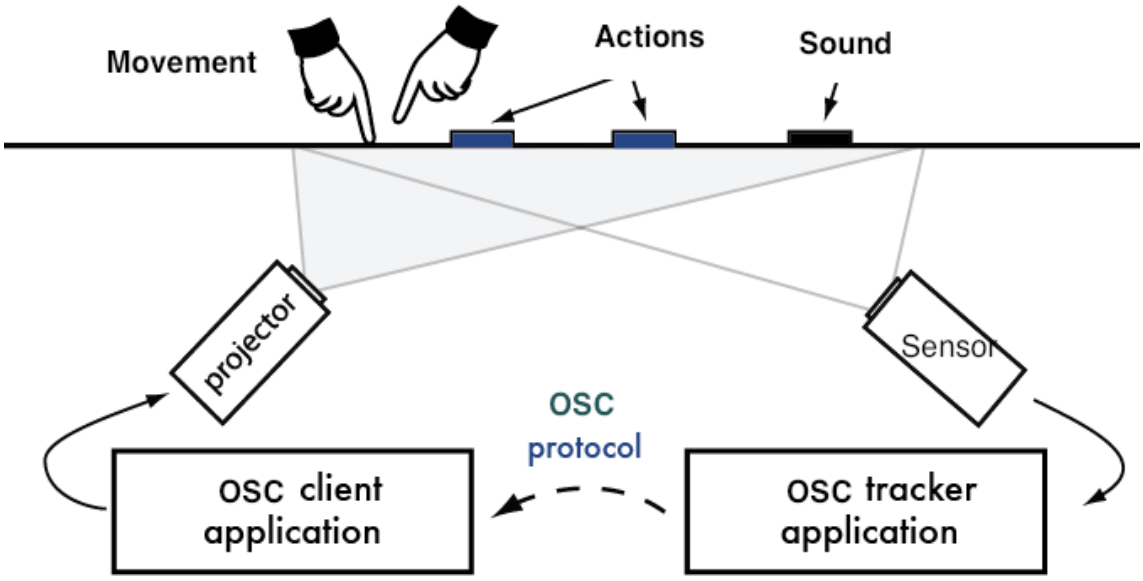


Ilustración 1: Protocolo OCS

b)CORBA - Common Object Request Broker Architecture

Introducción a CORBA

CORBA es un middleware que permite crear aplicaciones distribuidas. CORBA nació gracias al trabajo del OMG (Object Management Group)[10]. El OMG es un consorcio internacional que reúne a las empresas tecnológicas más importantes y es el encargado de especificar los estándares de tecnologías orientadas a objetos.

El propósito de esta nueva tecnología era dotar a los desarrolladores de una nueva manera con la que poder comunicar sus aplicaciones con otras que no estuvieran implementadas en el mismo lenguaje, promoviendo así la interoperabilidad entre aplicaciones cliente-servidor [5].

Para explicar como funciona CORBA hay que centrarse en tres conceptos fundamentales. El lenguaje para describir la interfaz de los objetos o IDL, el bus de comunicaciones ORB (Object Request Broker)[8] y el protocolo de comunicación IIOP(Internet Inter-Orb Protocol).

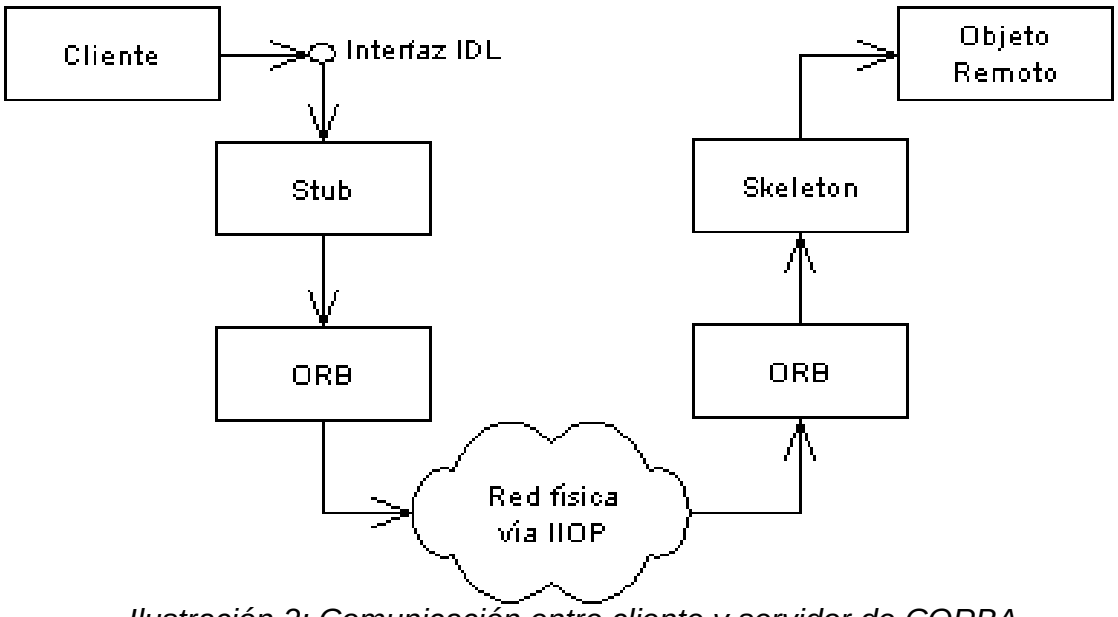


Ilustración 2: Comunicación entre cliente y servidor de CORBA

Si se observa la figura anterior se podrá tener una idea sencilla de como se realiza la comunicación entre aplicaciones cliente y servidor. El Cliente, que sólo conoce del objeto remoto su interfaz (IDL), realiza una llamada a un método del objeto distribuido. El cliente interactúa con un proxy, llamado STUB y en la parte del servidor este interactúa con un proxy de servidor llamado SKELETON. La capa del ORB sirve de intermediaria entre los proxys, la red y el sistema operativo. Las comunicaciones se realizan utilizando el protocolo IIOP (basado en TCP). El Servicio de Nombres proporciona la manera con la que el cliente puede localizar la referencia a un objeto remoto sin necesidad de conocer el lugar en el que está ni la forma de encontrarlo.

El lenguaje IDL

En 1991 especificaron el Interface Definition Language, IDL cuyo propósito es el de definir la interfaz de aquellos objetos que se ponen a disposición de forma remota. Gracias a él una aplicación cliente sabe cual es el interfaz del objeto distribuido y puede llamar a sus métodos remotamente (lo que se conoce como RPC o Remote Procedure Call), siendo trasparente el lenguaje en el que está programado dicho objeto [6].

El IDL permite especificar cuales son los objetos que van a estar disponibles remotamente, así como los atributos y métodos de dichos objetos y las excepciones que estos pueden lanzar. También permite definir tipos de datos estructurados. Además para cada lenguaje con el que se puede utilizar CORBA existe una herramienta que a partir de este IDL genera todas aquellas clases necesarias para la comunicación.

El ORB

El Object Request Broker es el encargado de trasportar las llamadas y de activar los objetos remotos. Guarda las referencias a los objetos distribuidos, IOR (Interoperable Object Reference), para que después puedan ser localizados por aquellos objetos que necesitan utilizarlos. El ORB permite a su vez utilizar el Servicio de Nombres con el que asociar cada referencia a una cadena de caracteres, de esta forma cualquier objeto con acceso al ORB podrá obtener una referencia a un objeto distribuido sin preocuparse de nada más que cual es su nombre [8].

IIOP

Este protocolo de comunicación es utilizado por el ORB para la comunicación. Es una implementación basada en TCP de las definiciones abstractas del protocolo GIOP. El IIOP se encarga de trasmitir los datos del ORB por la red física. También permite comunicar distintos ORB entre sí, lo que posibilita comunicar aplicaciones por internet utilizando CORBA.

c) ICE – Internet Communications Engine

Introducción

ICE (Internet Communications Engine) al igual que CORBA es un middleware orientado a objetos que facilita la construcción de aplicaciones distribuidas simplificando la forma de acceder a los objetos compartidos. ICE ha sido desarrollado al margen del OMG por una parte de los colaboradores que trabajaron en CORBA años atrás [14].

Uno de los motivos por los que se trabajó en el desarrollo de ICE fue para mejorar aquellos aspectos negativos que tenía CORBA derivados de la rigidez de las especificaciones del OMG. Esto ha permitido simplificar el diseño para ganar rapidez y hacerlo más sencillo de utilizar y aprender.

ICE se basa en el mismo principio que CORBA, la interoperabilidad, por lo que las similitudes entre estas dos tecnologías son muy grandes. Para definir la interfaz de los objetos se creó un nuevo lenguaje, el SLICE, bastante similar al IDL pero más potente. Sin embargo existen algunas diferencias que merece la pena comentar.

Diferencias con CORBA

ICE simplifica las operaciones de CORBA facilitando aún más el desarrollo de aplicaciones distribuidas.

Los SLICE son parecidos a los IDL pero también pueden ser utilizados para describir los estados de los objetos y hacerlos persistentes en el tiempo.
ICE da servicio a lenguajes más actuales como los orientados a dispositivos móviles.

ICE es más reciente y permite desarrollar con facilidad aplicaciones distribuidas para móviles, al no guiarse por un comité y sólo por su comunidad de desarrolladores, las mejoras se realizan y actualizan más dinámicamente y están enfocadas a la mejora del rendimiento.

ICE facilita la comunicación segura, entre aplicaciones en redes no seguras, permitiendo el cifrado de los datos [15].

Arquitectura

En la arquitectura de ICE se aprecian grandes similitudes con CORBA, así como la simplificación en el diseño y en la comunicación.

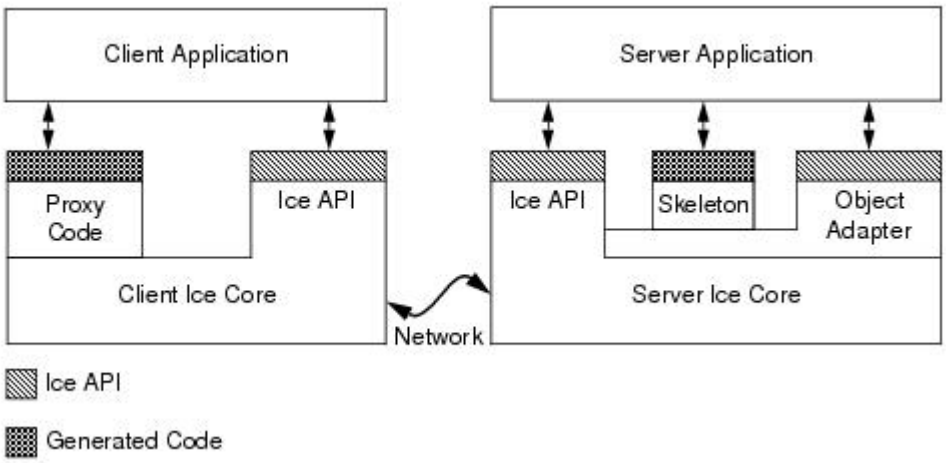


Ilustración 3: Arquitectura de la comunicación en ICE

Los términos de Cliente y Servidor no designan entidades en particular sino que denotan roles en zonas de la aplicación durante el transcurso de la petición.

Los clientes son entidades activas cuya función es hacer peticiones a los s*ervers*. Por otro lado los s*ervers* son entidades pasivas. Proveen servicios en respuesta a las peticiones de los clientes. Muchas veces un *server* puede actuar como tal frente a un cliente pero como cliente frente a otro servidor. Son muy flexibles en su funcionamiento.

Un Objeto de ICE es una entidad conceptual que reside en una dirección local o remota y puede responder a una petición de un cliente. Puede ser instanciado por un único servidor, o por muchos. Aunque sea instanciado por muchos, sigue siendo un único objeto. Cada objeto de ICE tiene una o más interfaces así como un valor de vuelta para todos los métodos.

2.2.Motores de Reglas

Uno de los principales objetivos a cumplir en este proyecto es la posibilidad de configurar la reacción del sistema ante determinados estímulos percibidos. Para poder resolver este objetivo de una manera fácil para el usuario se pretende que pueda definir una serie de reglas del tipo "si...entonces...". Como todo esto debe realizarse de manera dinámica y en tiempo de ejecución se ha investigado el campo de los motores de reglas.

El objetivo de estos motores, es proporcionar una infraestructura desacoplada del código fuente, de manera que el sistema pide servicios al motor para evaluar una regla.

En estos motores existe un Working Memory donde residen las reglas que serán invocadas por medio de un protocolo de aplicaciones.

De esta manera, se puede construir un sistema para que el usuario pueda definir sus reglas y cambiarlas.

Esto permite que los desarrolladores no sean los responsables de cambiar reglas ya hechas ni de crear algunas nuevas.

La tecnología que respalda a algunos de estos motores, se fundamenta en un algoritmo utilizado por los sistemas expertos, RETE [16] , que optimiza el proceso de coincidencia de las reglas.

El funcionamiento de estos motores de reglas se puede esquematizar de la siguiente manera:

- Creación y mantenimiento de un banco de reglas.
- Introducción de una serie de hechos en Working memory.
- Ejecución del motor de inferencia que en base a esos hechos ejecuta las acciones concluidas por las reglas.

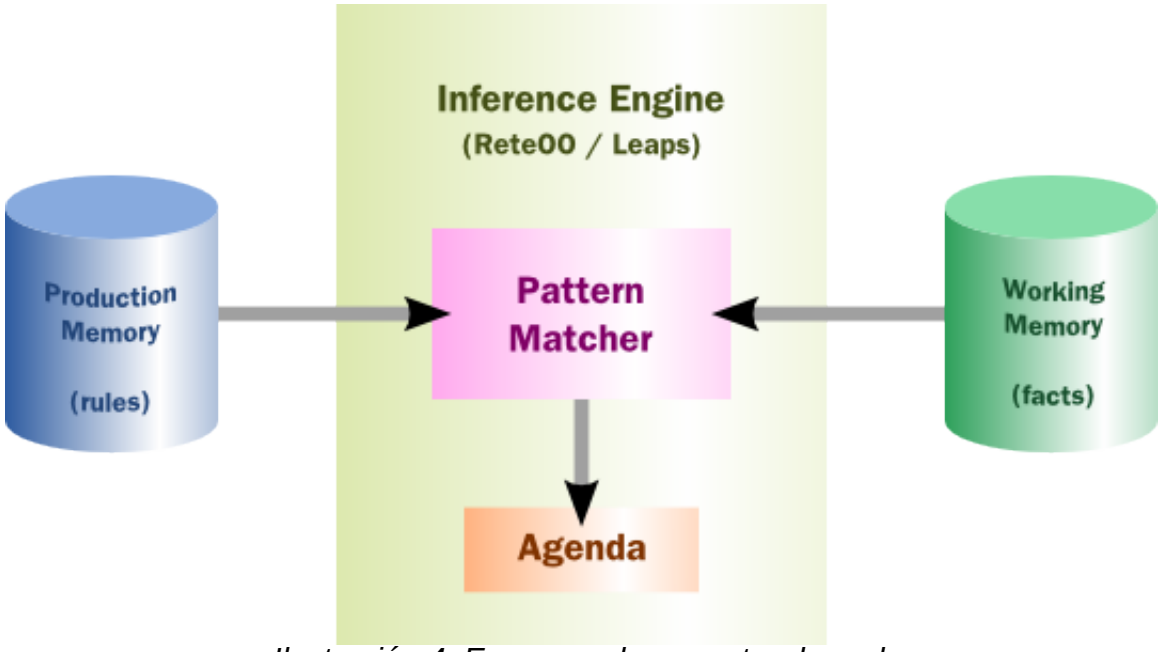


Ilustración 4: Esquema de un motor de reglas

Actualmente existen varias implementaciones en Java, pero en su mayoría son comerciales. A continuación comentaremos algunas de ellas.

a) Jess

Es un sistema basado en reglas de producción y un entorno de scripting implementado en Java por Ernest Friedman-Colina.

Jess es pequeño, ligero y uno de los motores de reglas mas rápidos disponibles. Su lenguaje de scripting permite el acceso a todas las API de Java.
Para su funcionamiento utiliza una versión avanzada del algoritmo RETE para procesar reglas. RETE mejora la eficacia.

Permite encadenamiento hacia delante y hacia atrás. Además puede manipular y razonar directamente sobre objetos Java, crear objetos, llamar métodos sin tener que compilar previamente el código.

Aunque Jess no es gratuito, puede ser usado para fines académicos pudiéndose adquirir una licencia en otro caso. [17]

b) Drools

Es un sistema de administración de reglas de negocio (BRMS) con razonamiento en cadena hacia delante y con un motor de reglas basado también en el algoritmo Rete pero en esta ocasión adaptándolo a objetos. Permite expresar de una forma más natural las reglas interactuando directamente con los objetos. Proporciona una separación de la lógica (reglas) y los datos (hechos).

También provee soporte para la programación declarativa.
Además cuenta con la implementación completa de la JSR-94 Rule Engine API.
Las reglas pueden ser escritas en Java entre otros lenguajes de programación. Además éstas pueden definirse y almacenarse en formato XML.

Drools está disponible como open source.[18]

2.3.Sistemas de Detección

Este punto cobra una especial relevancia puesto que la percepción de sucesos externos es uno de los pilares de los que se compone el proyecto. Debido a lo extenso del tema se ha tenido que realizar una gran labor de síntesis y documentación.

En una primera fase ha tenido que definirse claramente el tipo de datos que serían apropiados detectar en función del objetivo principal del proyecto, la interacción del sistema con el visitante. De esta manera pueden descartarse un gran número de sensores que no resultan adecuados.

Una vez tenido esto en cuenta se pasará a detallar las distintas posibilidades de sensores encontrados:

a) Phidgets

Son un set de sensores del tipo '*plug & play*', construyendo bloques a un precio razonable, mediante conexión USB, siendo controlados desde el ordenador. Con la compra de los sensores también se tiene acceso a las API´s para gestionar los sensores. Se encuentran disponibles para lenguajes de programación como C/C++, C#, Cocoa, JAVA, MatLab, ...

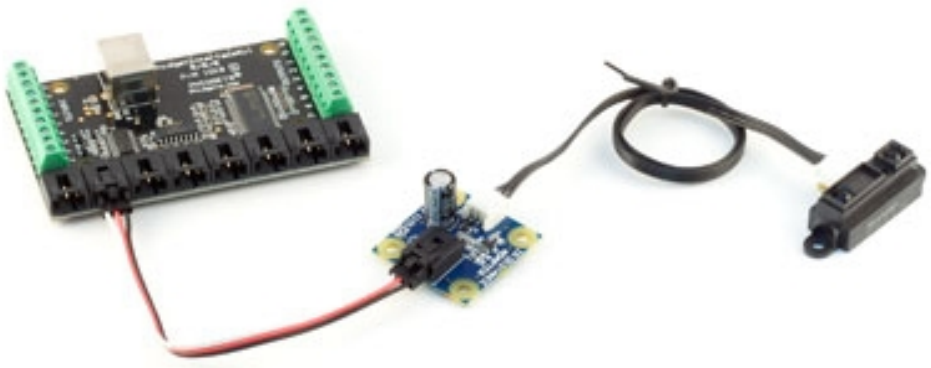


Ilustración 5: Aspecto de los Phidgets

Los Phidgets no son de un solo tipo, sino que engloban gran variedad de ellos. Detectan desde distancias, fuerzas ejercidas sobre superficies, superficies táctiles, detectores de movimientos, voltajes, y muchos más. También incluyen varios tipos de motores y accesorios.

Aún así, los Phidgets no se pueden conectar directamente al ordenador, necesitan de un interfaz mecánico que es quien de verdad se conecta al ordenador. Las librerías que dan ayudan a controlar los distintos eventos que detectan los sensores.

El principal problema de este tipo de sensores es el alcance. Son sensores que captan toda acción que ocurre en un espacio, pero éste debe ser muy reducido.

Pongamos como muestra los sensores de distancia.

El sensor **3522 - Sharp Distance Sensor 2Y0A02** [20] es el de mayor precio y alcance. Con esto, únicamente aseguran que mide distancias desde 20 cm hasta 150 cm. Para una aplicación que necesite medidas mayores de distancias, el resultado queda muy lejos de la mayoría de necesidades de instalaciones artísticas.

Aún con todo, estos sensores son buenos debido a la sencillez de uso de sus librerías. Se puede apreciar en la siguiente figura cómo son los métodos de uso para un acelerómetro desde la API.

Method Summary

void	**addAccelerationChangeListener**(AccelerationChangeListener l) Adds an acceleration change listener.
double	**getAcceleration**(int index) Returns the acceleration of a particular axis.
double	**getAccelerationChangeTrigger**(int index) Returns the change trigger for an Axis.
double	**getAccelerationMax**(int index) Returns the maximum acceleration value that this axis will report.
double	**getAccelerationMin**(int index) Returns the minimum acceleration value that this axis will report.
int	**getAxisCount**() Returns the number of accelerometer axes.
void	**removeAccelerationChangeListener**(AccelerationChangeListener l)
void	**setAccelerationChangeTrigger**(int index, double newVal) Sets the change trigger for an Axis.

Ilustración 6: Javadoc de API Phidgets

Una instalación que se plantee crear con estos sensores debe de tener especial cuidado con el precio. Al no tener los sensores mucho alcance, habría que llenar la instalación de ellos, aumentando el precio considerablemente. [19]

b)Cámaras

Las cámaras no pueden considerarse como sensores en sí mismas excepto si se les aplica el software adecuado.

En este campo existen gran cantidad de librerías de las que puede hacerse uso fácilmente y que dotan a estos dispositivos de múltiples usos. Éstos comprenden desde la detección de movimiento, de caras, de cuerpo e incluso de patrones de realidad aumentada.

Dentro de todas estas librerías se destacarán las siguientes:

OpenCV

Es una biblioteca libre y multiplataforma de visión artificial desarrollada en su origen por Intel. Ha sido utilizada en numerosas aplicaciones con el fin de detectar movimiento o para el reconocimiento de objetos. [22]

Aunque se encuentra implementada en C y C++ es posible encontrar wrappers para poder usarla en Java además de otros lenguajes.

Ofrece un gran número de funciones (más de 500) que abarcan una gran gama de áreas en el proceso de visión. Sin embargo para los objetivos de este proyecto serán especialmente útiles las capacidades de detección de caras.

Dentro de las posibles implementaciones de esta librería se encuentra una que permite que pueda ser utilizada dentro del entorno de Processing. Ésta será la que más tarde se utilizará como sensor.Error: Reference source not found

Tiene una licencia BSD que permite que sea usada libremente para propósitos comerciales y de investigación.

ARToolKit

Es una librería que facilita el uso de patrones para la construcción de aplicaciones de Realidad Aumentada. Para ello una cámara detecta un patrón consistente en un rectángulo con un código en su interior. El sistema ha sido previamente entrenado para el reconocimiento de dicho patrón. [25]

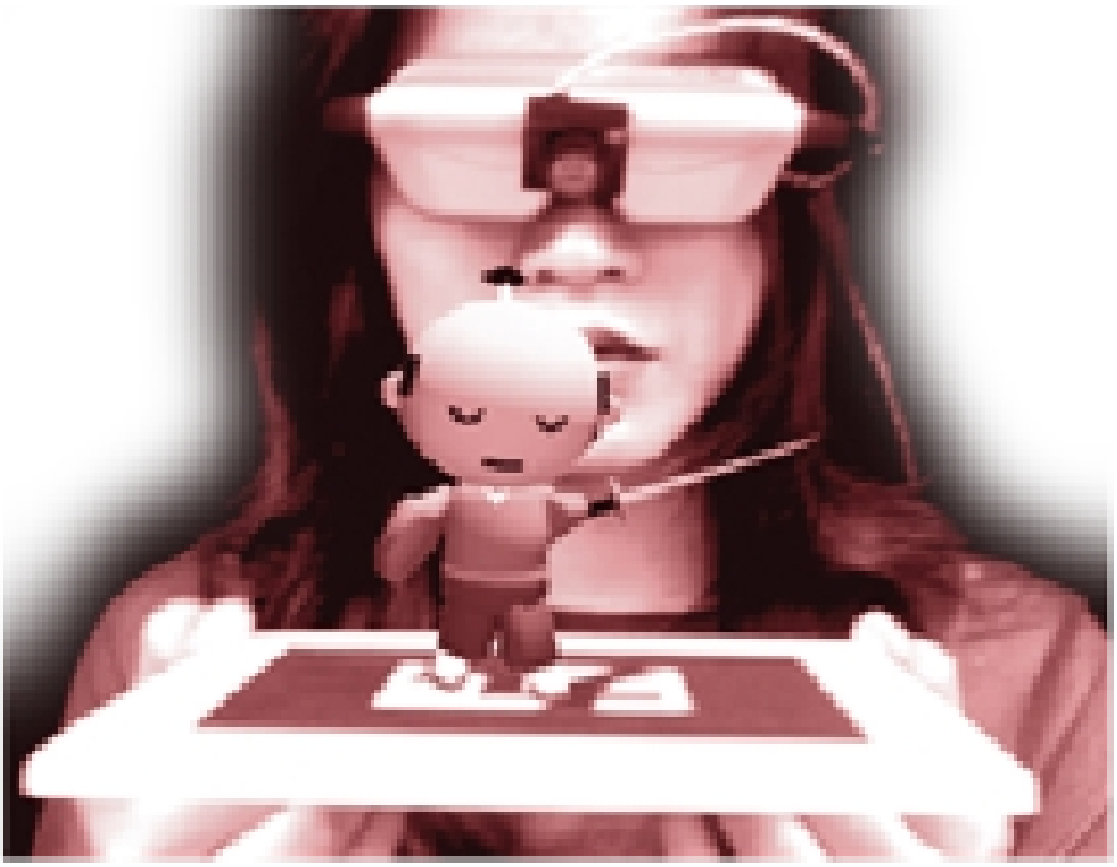


Ilustración 7: Aspecto de ArToolKit

Actualmente se mantiene como un proyecto de código abierto alojado en SourceForge con licencias comerciales disponibles en ARToolworks. [26]

c)Micrófonos

De la misma manera que las cámaras, los micrófonos por sí mismos tampoco constituyen un sensor, al menos no en la manera que se pretenden utilizar. Sin embargo, al igual que ocurre con las cámaras, es posible encontrar librerías que permiten hacer de estos dispositivos sensores más interesantes.

Debido a la gran extensión de librerías disponibles se ha centrado la documentación sobre una en concreto, Minim. [27]

Minim es una librería que al mismo tiempo hace uso de otras librerías como JavaSound API y Tritonius. Proporciona la posibilidad, entre otras, de detectar sonidos según una serie de cualidades de éstos.

Aunque es una librería mucho más potente, ha sido interesante en la investigación por su faceta de dotar cualidades de sensor a un micrófono.

Cabe destacar que es una librería que se integra dentro de un scketch de Processing, herramienta que se detallará más adelante.

Minim es una librería que dispone de una licencia GNU Lesser General Public License (LGPL)

2.4.Arte

El Arte que interviene en ésta aplicación se puede dividir en dos corrientes:

- Programas de Artistas Digitales.
- Las Instalaciones Artísticas Digitales.

La explicación de esta división se hará dentro de cada uno.

a)Programas de Artistas Digitales

La definición de Artista Digital ha variado mucho en pocos años. Antes era cualquiera que utilizase un programa digital para crear algo extraordinario. Hace 20 años, por ejemplo, nadie conocía Photoshop y los que lo utilizaban sí que podían ser considerados Artistas Digitales.

Hoy en día, se ha globalizado y democratizado el uso de herramientas digitales. Todo el mundo puede tomar una fotografía digital, volcarla en el ordenador, modificarla con algún software de edición de imágenes, crear música desde su propia casa, subirla a internet, ... En definitiva, ya no se considera arte en sí.

Es dominio de los artistas y del arte el buscar nuevas formas de expresión. Ya no se quiere crear algo en un programa de 3D, porque aunque el artista elija algún elemento, la última palabra siempre la tendrá el programador que creó el software.

Lógicamente, un artista con pocos conocimientos de informática no puede programar en C o en JAVA, pero sí que se podría adaptar algo para ellos.

No es la primera vez que esto ocurre, ya que con los ActionScripts de Flash sucedió algo parecido.

Existe una nueva corriente de jóvenes artistas que crearon lo que actualmente se conoce como PROCESSING. [31]

Processing es un lenguaje de programación, basado en JAVA y orientado por y para artistas. Consta de una IDE muy sencilla, con reconocimiento de sintaxis por colores y una amplia comunidad. Al ser Open Source todas las librerías existentes (que son muchas), están al alcance de cualquiera.

Como dice Paola Antonelli en su libro "Design and the elastic mind" [30], "*(Processing) es suficientemente sencillo para empezar para alguien que nunca ha programado y sofisticado para diseñadores de alto nivel , para arquitectos, para visualizaciones y proyectos de animación. Processing ha tenido un poderoso impacto como una herramienta poderosa e inspirativa".*

Paola Antonelli es Conservadora del museo MOMA de Nueva York y una de las 100 personas más influyentes en el mundo del arte según la revista Art Review.

Processing no es el primer lenguaje de programación en donde se involucran artistas sino que es el primero pensado exclusivamente para ellos.

Antes muchos artistas utilizaban ActionScript de Flash como por ejemplo los creativos de muchas agencias de publicidad. Anterior a estos dos, también existió MSW LOGO, nacido a finales de los sesenta y que con el tiempo diseñadores fueron creando obras con él.

b)Instalaciones Artísticas Digitales

En las instalaciones artísticas, el uso de elementos digitales lleva utilizándose desde la invención del ordenador. El vídeo y el audio en una instalación se presuponen pero la forma de disparar ambas muchas veces es casi analógica, teniendo que estar una persona pendiente de la obra para activar, cuando sea necesario, unos eventos u otros.

Es cierto que también existirán siempre instalaciones artísticas que no necesitarán de tecnología (como por ejemplo la de Marti Gauixé en el ExperimentaDesign del 2008 en Amsterdam). [Mul9]

Cuando se habla de automatizar, no se dice para eliminar el componente humano, sino a que el artista cree ciertos comportamientos en la aplicación y estos se repitan según sus órdenes.

Siempre se pueden utilizar los medios más actuales, como esta pantalla de Philips, llamada Daylight [Mul7]. Consiste en un cristal transparente que además es una pantalla de gran calidad. Existen otros ejemplos de tecnologías y de instalaciones artísticas como "Sustainable Dance Floor" de Daan Roosegaarde [Mul8] que mediante el contacto del visitante con el suelo genera energía para iluminarse a si misma.

No hay que pensar que las instalaciones artísticas con tecnología se encuentran en museos o exposiciones únicamente. Dando un paseo por Madrid, en la Plaza de las letras, se puede encontrar una exposición de Pablo Valbuena llamada "Entramado" [Mul10]. Se trata de un proyector que al llegar la noche dibuja desde una de las esquinas todo el entramado de baldosas que tiene la plaza.

Ilustración 8: Instalación de Pablo Valbuena

2.5. Conclusiones

Después de toda la investigación anterior, estas son las conclusiones para cada uno de los ámbitos del proyecto.

a) Elección del sistema de comunicación

En cuanto a la comunicación entre componentes, SmartRoom empezó utilizando la librería de processing [4] oscP5 (basada en el protocolo OSC) para la comunicación entre componentes. Sin embargo esto limitaba y dificultaba la comunicación ya que sólo permite enviar por red tipos de datos primitivos (ver OSC) además de ocasionar retardos. Una vez descartada esta posibilidad se investigó sobre tecnologías para crear aplicaciones cliente-servidor, en especial ICE y CORBA.

ICE es una tecnología novedosa que ofrece ventajas sobre CORBA pero que no son necesarias para la realización de este proyecto es por ello que se eligió CORBA, al primar que está totalmente integrado con Java. Además abunda los recursos sobre documentación y permite llevar a cabo todos los requisitos de la aplicación, aunque también hubiese sido acertado elegir ICE.

Otras alternativas para la comunicación podían haber sido .NET Remoting, RMI y los Servicios Web (utilizando SOAP). Pero existían limitaciones en cada una de ellas. La limitación de Remoting es que el sistema operativo debe ser de la familia Windows NT. RMI es multiplataforma sin embargo el lenguaje solamente puede ser Java. Elegir SOAP podría haber desviado el proyecto de su objetivo al tener que dedicar gran parte del tiempo a comprender y aprender a utilizar esta tecnología. Además con CORBA, cabe la posibilidad en un futuro de utilizar otros lenguajes para programar otro tipo de sensores más específicos.

b) Elección del Motor de Reglas

En la elección del sistema de gestión de reglas se ha elegido Drools. Existe mucha documentación, dispone de licencia open source, permite interactuar directamente sobre objetos en Java y pueden almacenarse y gestionarse fácilmente las reglas en formato XML.[18]

c) Elección de los Sistemas de Detección

Una vez llevada a cabo una amplia investigación se decidió hacer uso de las librerías OpenCV ya que su uso está muy extendido en el ámbito de la detección con cámaras. Además permite la detección de caras de una manera muy simple.

Están compiladas para windows, linux y mac.Error: Reference source not found

La librería Minim, aunque es mucho más extensa de lo necesario, permite la detección de sonidos de una manera muy fácil también. [27]

Ambas hacen uso, aunque no necesariamente, de Processing para funcionar así que encaja con los propósitos de integrar este tipo de lenguaje al proyecto.

Se ha descartado el uso de Phidgets por ahora porque la funcionalidad de detección de movimientos se puede hacer con una cámara en vez de con sensores. Para las necesidades de las aplicaciones desarrolladas de momento no es necesario. No obstante, su integración en SmartRoom sería sencilla gracias a las APIs definidas para Java.

CAPÍTULO 3

3.Arquitectura de SmartRoom

3.1.Visión general Del Sistema

La aplicación *SmartRoom* se puede dividir en tres módulos: *SmartRoomSensor* (sensor), *SmartRoomPlayer* (medio) y *SmartRoomCenter* (central).

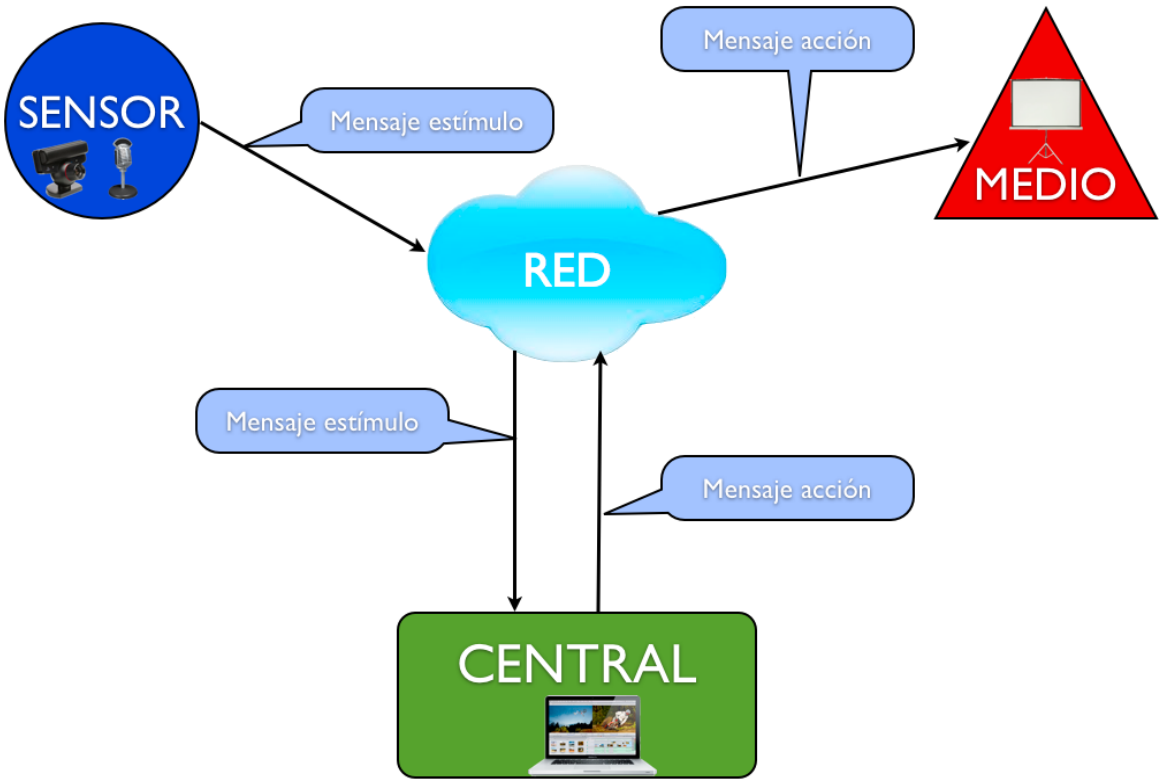


Ilustración 9: Diagrama General de SmartRoom

Cada instalación únicamente tendrá un ***SmartRoom Central***. A su vez, esta instalación tendrá un número no definido de ***SmartRoom Sensor*** y ***SmartRoomPlayer*** (Medio).

Como se ha comentado, son tres tipos de módulos, dos de ellos, Sensor y Medio, conectados mediante la red a la Central. Los tres, por separado, están basados en el patrón de Modelo Vista Controlador. Eso es internamente y será visto más adelante, pero al relacionarse entre sí siguen otra arquitectura externamente.

La arquitectura de los tres módulos en conjunto es un tipo de arquitectura Centralizada en red, ya que son distintos componentes, conectados en red y todos comunicándose con la Central. Tanto Sensores como Medios se comunican únicamente con la Central y si ésta se cayese de la red, el resto se desconectaría de inmediato.

Se barajaron otro tipo de arquitecturas descentralizadas en las que cada Sensor se comunicara con el Medio que quisiera, pero al introducir el razonamiento a través de la reglas se tuvo que utilizar una Arquitectura Centralizada para manejar cada componente.

Cabe destacar el uso de "interfaces tipo" [IDL1] para comunicarse entre sensor -> central y central -> medio. Esto ayuda a extender los tipos de sensores y medios que se quiera introducir, si cumplen con nuestras especificaciones serán introducidos en la nube de *SmartRoom* como otro sensor o medio más.

Tanto los medios como los sensores se pueden iniciar sin existir ninguna Central en la red. No se comunicarán con nadie pero esto puede ser útil para motivos de configuración antes de iniciar el sistema.

A continuación se explicarán más en detalle cuál es la estructura y funcionamiento de cada uno de esos módulos. Para ello, en un principio se realizará un análisis de su funcionamiento autónomo y en el capítulo siguiente se comentará el funcionamiento global de todo el sistema. Para SmartRoom es vital la comunicación entre los distintos sistemas por medio de CORBA por lo que tendrá un capítulo a parte.

a)Diagrama de Componentes

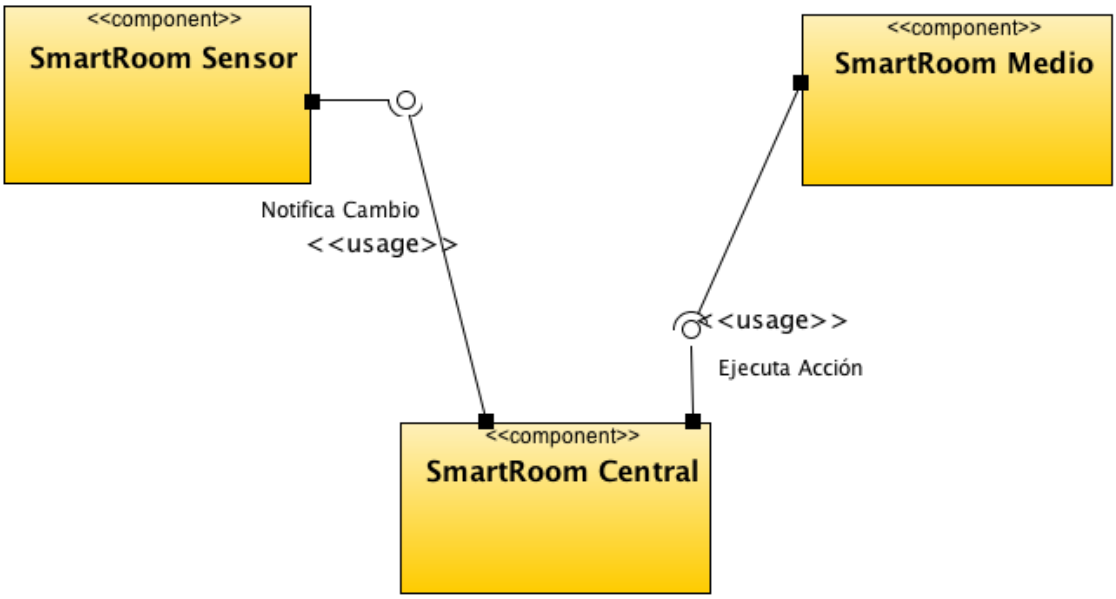


Ilustración 10: Diagrama de Componentes General

Este diagrama de componentes permite observar el funcionamiento del sistema SmartRoom. Se puede observar como hemos eliminado la capa de Red que une todo para clarificar el diagrama.

SmartRoom Central ofrece a Sensor una interfaz Notifica Cambio. Cuando esto se produce SmartRoom Central utilizará su interfaz Ejecuta Acción con el componente SmartRoom Medio.

Con este diagrama se ilustra la organización de SmartRoom, permite observar cómo es la comunicación entre los distintos módulos.

b) Ejemplo de Despliegue

Aquí se muestra un ejemplo de despliegue del sistema de forma distribuida haciendo uso de CORBA para la invocación remota de métodos. Como se puede observar, resulta más conveniente que las aplicaciones estén distribuidas en nodos diferentes. Esto es así porque en principio, cada una puede encontrarse en salas diferentes o en ordenadores diferentes para hacer un mejor uso de los recursos tales como poder de procesamiento o dispositivos de captura y de reproducción.

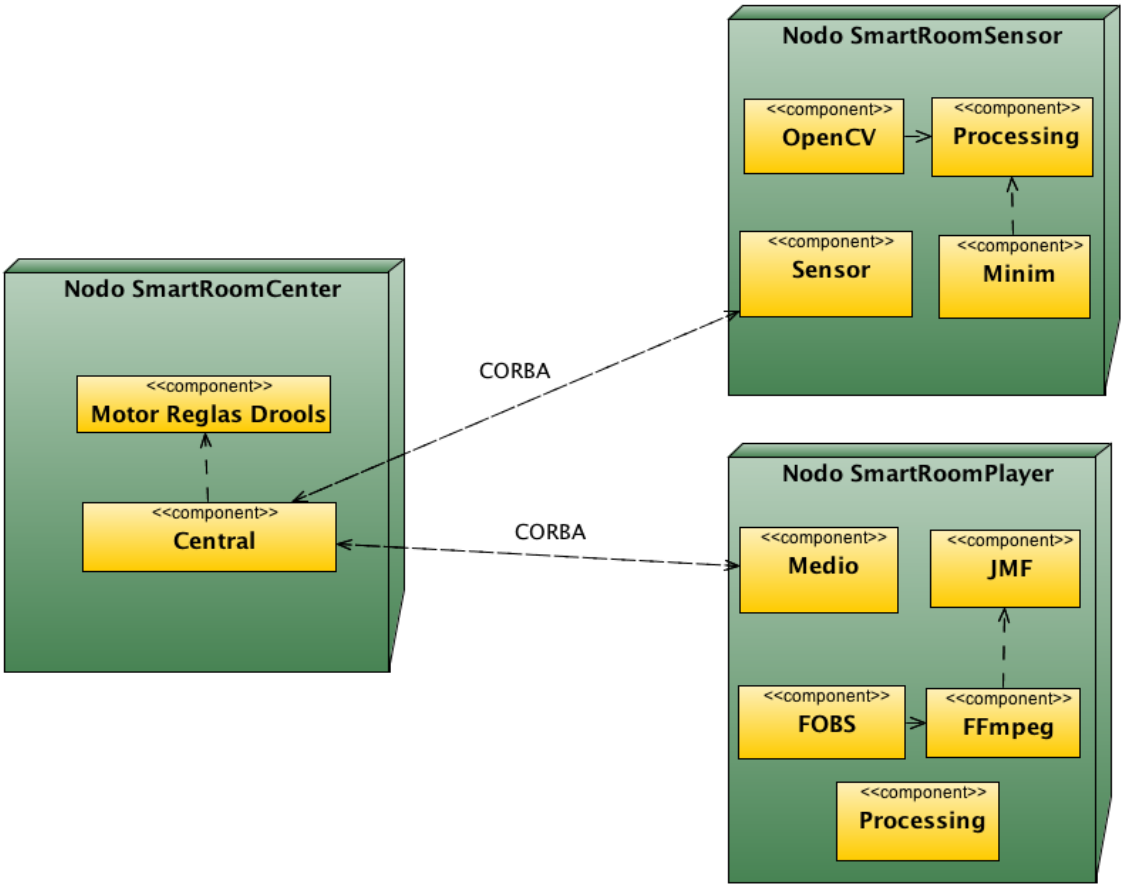


Ilustración 11: Diagrama de Despliegue de Componentes

c) Modelo Vista Controlador

Para la implementación del proyecto, se basa el patrón Modelo Vista Controlador (MVC). Permite separar la lógica de control, la lógica de negocio y la lógica de presentación. [32]

Esta separación de los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos consigue que las modificaciones al componente de la vista puedan ser hechas con un mínimo impacto en el componente del modelo de datos. Esto es muy útil ya que los modelos suelen ser relativamente estables (dependiendo de la estabilidad en el dominio del problema que está siendo modelado). Además, utilizándolo se consigue normalización y estandarización del desarrollo de Software.

En el patrón de diseño MVC, el flujo de la aplicación está dirigido por un Controlador central. El Controlador delega solicitudes -- en este caso, el modelo en sí – a un manejador apropiado que sepa hacer la tarea encomendada por el modelo. Los manejadores están unidos al Modelo, y cada manejador actúa como un adaptador entre la solicitud y el Modelo. El Modelo representa, o encapsula, un estado o lógica de la aplicación. Finalmente, el control normalmente es devuelto a través del Controlador hacia la Vista apropiada.

En términos generales, construir una aplicación usando una arquitectura MVC implica definir tres clases de módulos.

- **Modelo**: La representación específica del dominio de la información sobre la cual funciona la aplicación. La lógica de dominio añade significado a los datos.
- **Vista**: Presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario.
- **Controlador**: Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

El MVC desacopla vista y modelo estableciendo un protocolo de notificación entre ellos. Una vista debe asegurar que su apariencia refleja el estado del modelo. Cuando el modelo cambia, lo notifica a la vista que depende de él. Como respuesta, cada vista tiene la oportunidad de actualizarse. Este enfoque permite utilizar múltiples vistas de un modelo que proveen diferentes presentaciones. Se puede también crear nuevas vistas para un modelo sin reescribirlo.

Es común pensar que una aplicación tiene tres capas principales: presentación, dominio y acceso a datos. En MVC, la capa de presentación está

partida en controlador y vista. La principal separación es entre presentación y dominio; la separación entre V/C es menos clara.

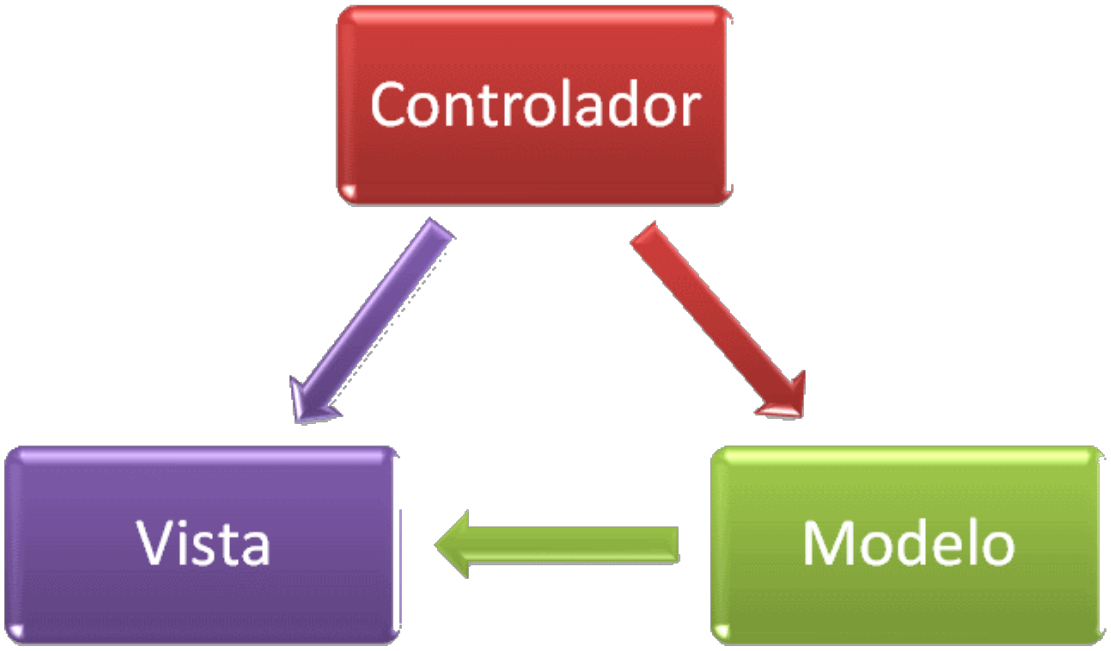


Ilustración 12: Esquema Modelo Vista Controlador

3.2.Elementos del Sistema

En este punto, se separan los distintos sistemas de la aplicación.

a)Arquitectura de SmartRoomSensor

Casos de Uso

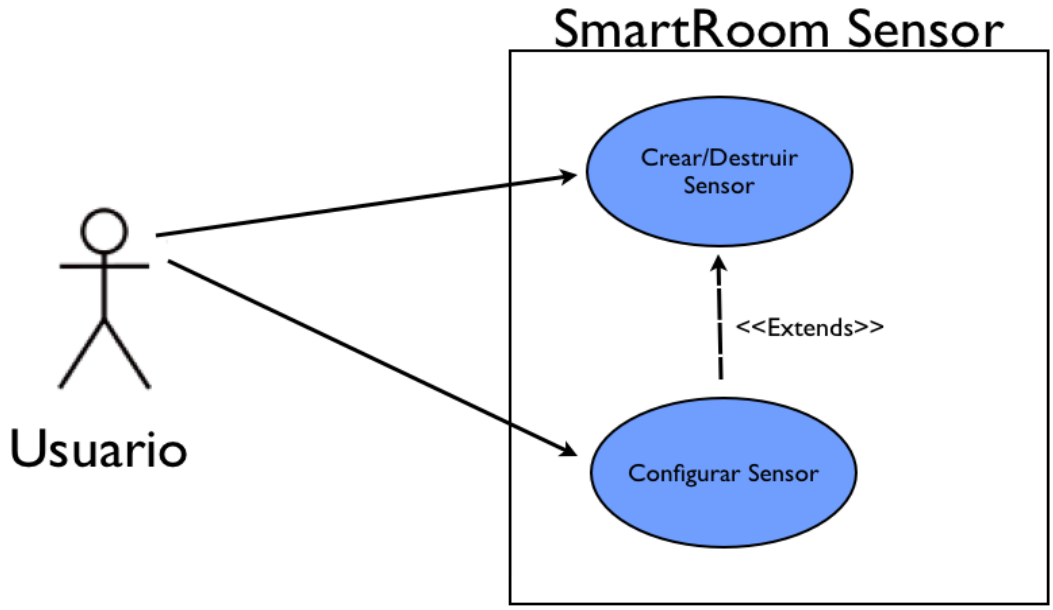


Ilustración 13: Casos de Uso de Sensor

El SmartRoom Sensor proporciona al usuario la posibilidad de configurar y activar sensores, entre aquellos disponibles. La aplicación mostrará al usuario por pantalla lo que el sensor está captando en tiempo real. SmartRoom Sensor es la aplicación encargada de informar a la Central de cuales son los estímulos que se pueden detectar y notificar a la central cuando éstos se produzcan.

Diagrama de Clases

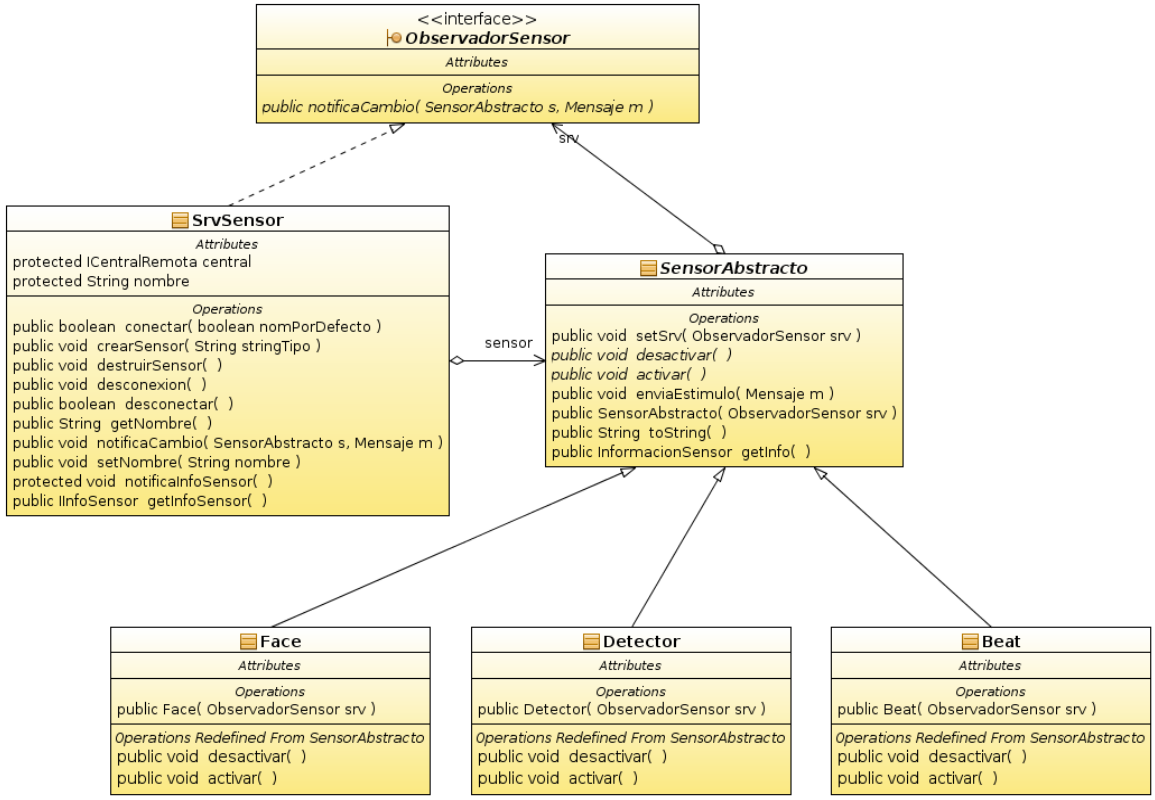


Ilustración 14: Diagrama de clases del Sensor

El anterior diagrama de clases muestra la estructura que permite crear sensores.

La clase SrvSensor ejerce como controlador de los sensores. Permite crear y destruir los sensores. Los sensores se activan al ser creados y se desactivan al ser destruidos.

El Servidor de Sensores implementa el interfaz ObservadorSensor a través del cual un sensor puede informarle de los cambios que percibe. Más adelante se profundizará en la manera en que esos cambios (también llamados estímulos) son transmitidos a la Central.

El hecho de que un sensor solo conozca el interfaz ObservadorSensor permite que pueda ser reutilizado en otras aplicaciones en el que el comportamiento del Observador sea distinto.

Arquitectura de los Sensores

Los sensores son muy heterogéneos aun utilizando los mismos recursos (por ejemplo una cámara). Es por ello que a pesar de dichas diferencias el sistema permite abstraer la manera de funcionar de los sensores, permitiendo en todo caso implementar un nuevo tipo de sensor e incluirlo en la aplicación de manera sencilla.

A continuación se detallará el funcionamiento de los sensores que se han implementado.

Detector de Movimiento

El comportamiento del sensor viene determinado por los parámetros de configuración. Éstos comprenden desde la sensibilidad que se desea aplicar, hasta los fps (frames por segundo) de la cámara y la frecuencia de envío de mensajes de estímulo.

El Detector utiliza una de las cámaras disponibles para detectar el movimiento. Para ello el applet divide la imagen capturada en una rejilla. Así es capaz de detectar el movimiento en cada una de las casillas (cuadrantes) de manera independiente.

La idea para detectar el movimiento es comparar la diferencia de color de cada píxel de la imagen recién capturada con el píxel de la anterior. De este modo si en uno de los cuadrantes la suma de los cambios de color es superior a la sensibilidad (configurada como parámetro) se supondrá que existe movimiento.

Este método funciona de manera correcta en ambientes en los que la iluminación no varía bruscamente. Al procesar cada frame se notifica por medio de un Mensaje al Observador del sensor la ausencia o presencia de movimiento en cada uno de los cuadrantes.

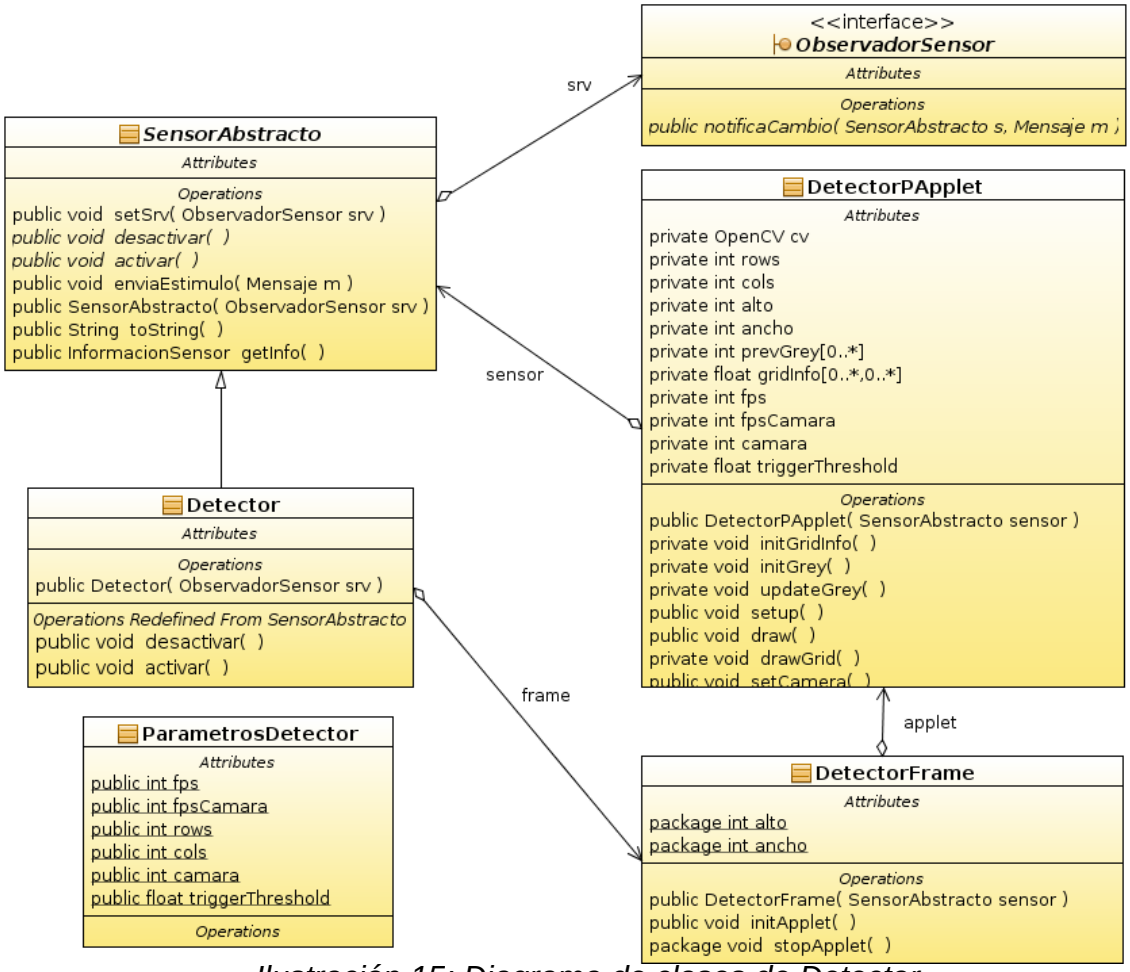


Ilustración 15: Diagrama de clases de Detector

Detector de Caras

Este sensor utiliza la librería OpenCV [22] para procesar la imagen y detectar el número y la posición de las caras. Esta librería simplifica en gran medida la manera de detectar las caras.

La forma de interacción es similar a la del detector de movimiento. Con una frecuencia determinada (configurable como parámetro) se notifica con un Mensaje al ObservadorSensor del número de caras detectadas.

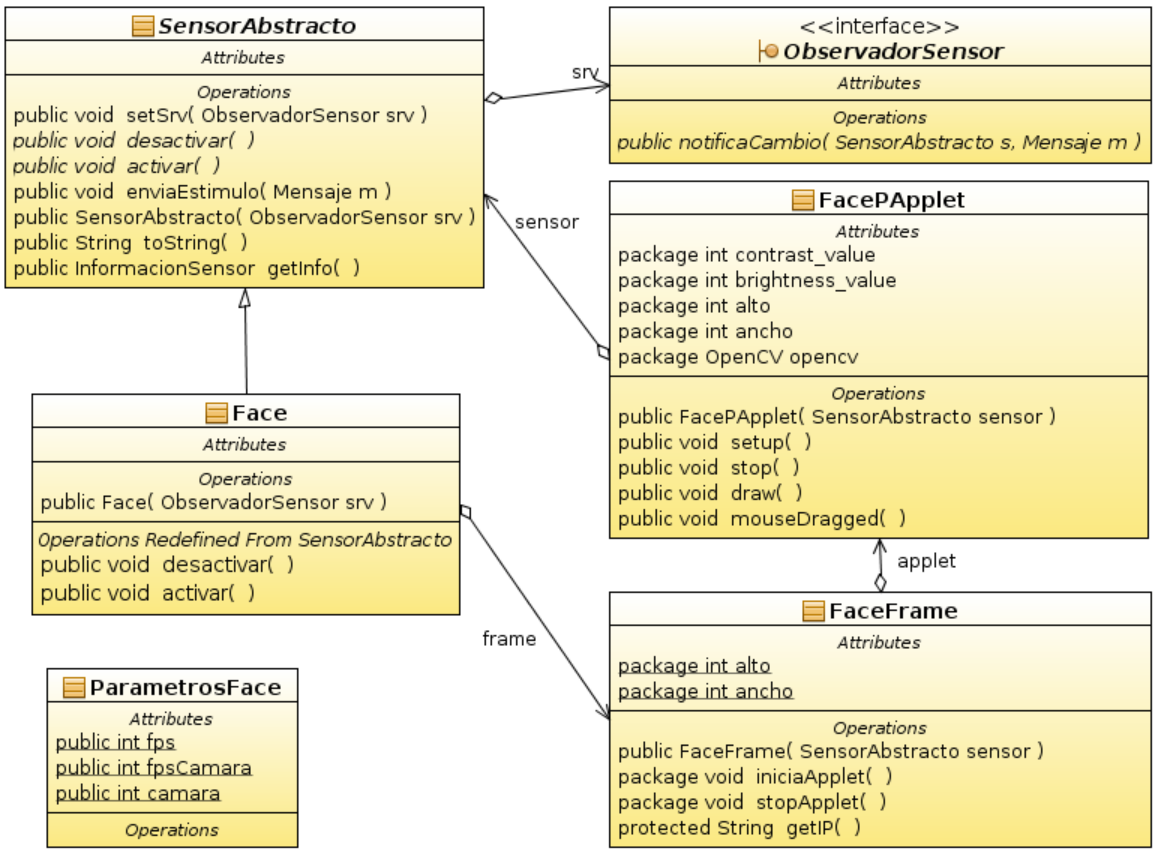


Ilustración 16: Diagrama de clases de Face

Detector de Sonido

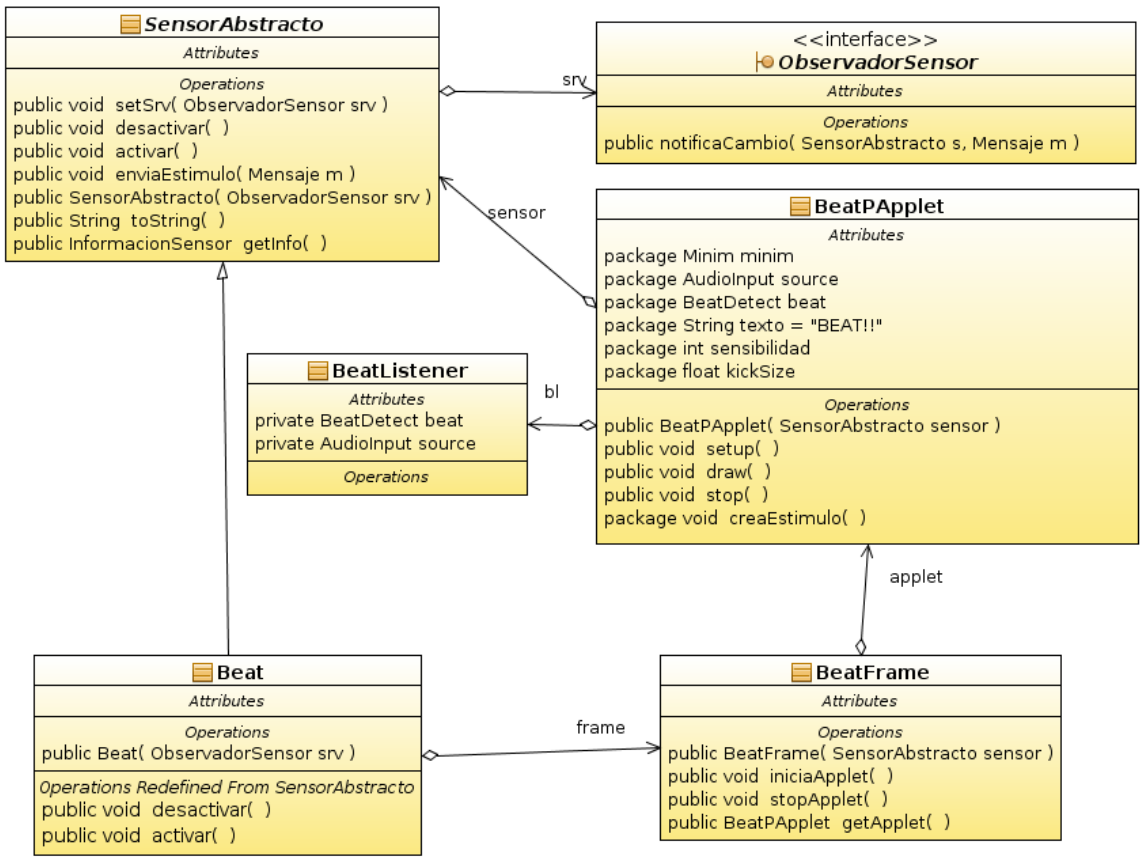


Ilustración 17: Diagrama de clases de Beat

El detector de sonido utiliza las librerías de sonido Minim [27]. Su comportamiento es sencillo. Cada vez que se detecta un sonido por el dispositivo de entrada (normalmente un micrófono) se notifica al ObservadorSensor del cambio por medio de un Mensaje.

Desarrollo de Nuevos Sensores

Independientemente del tipo de sensor y las librerías que se quieran utilizar para implementarlo la manera de integrar estos sensores en el sistema es siempre la misma. Todos ellos heredan de SensorAbstracto e implementan los métodos activar y desactivar. Utilizando el método envíaEstímulo(Mensaje m) se notifican los cambios a su Observador.

b)Arquitectura de SmartRoomPlayer

SmartRoomPlayer también llamado "medio" en la aplicación. Su función principal es la de conectar dispositivos de salida de vídeo y audio al sistema. Además debe ser capaz de reproducir el contenido multimedia en dichos dispositivos.

En esencia se compone de una biblioteca de archivos y de una serie de salidas de audio y vídeo conectadas. Cada salida de vídeo tiene asignada una ventana de reproducción que será capaz de mostrar el contenido de un applet. Se ha decidido aplicar este diseño con el fin de poder integrar applets de Processing además de contenido audiovisual habitual.

Se ofrece al usuario una interfaz gráfica sencilla con 3 botones para añadir o eliminar archivos a la biblioteca y otro para probar archivos en los dispositivos conectados. Cabe destacar que dichos archivos deben estar disponibles en el ordenador en el que se encuentre corriendo esta aplicación.

Reproductores

Para poder reproducir el contenido de la biblioteca se ha decidido utilizar la librería Java Media Framework (JMF) [Lib.2]. Aunque es una librería muy amplia, para los propósitos de esta aplicación resulta un tanto compleja y falta de plugins que permitan la reproducción de varios formatos de audio y de vídeo. Para conseguir resolver todas estas carencias y simplificar el uso de JMF se ha incluido también otra librería: FOBS [Lib.3]

Se han implementado acciones básicas de vídeo y audio como son reproducción (play), parada (stop), reproducción en bucle, aplicar restricciones de tiempo, aumento y disminución del volumen. El uso de estas librerías tan amplias permiten que en un futuro puedan ampliarse las posibilidades de uso del contenido.

Casos de Uso

Como ya se comentó antes, los usos de esta aplicación se reducen a labores de gestión de la biblioteca, añadiendo y eliminando archivos, probando que estos funcionan y que la aplicación SmartRoom Player ha detectado todos los Displays que tiene conectados al ordenador.

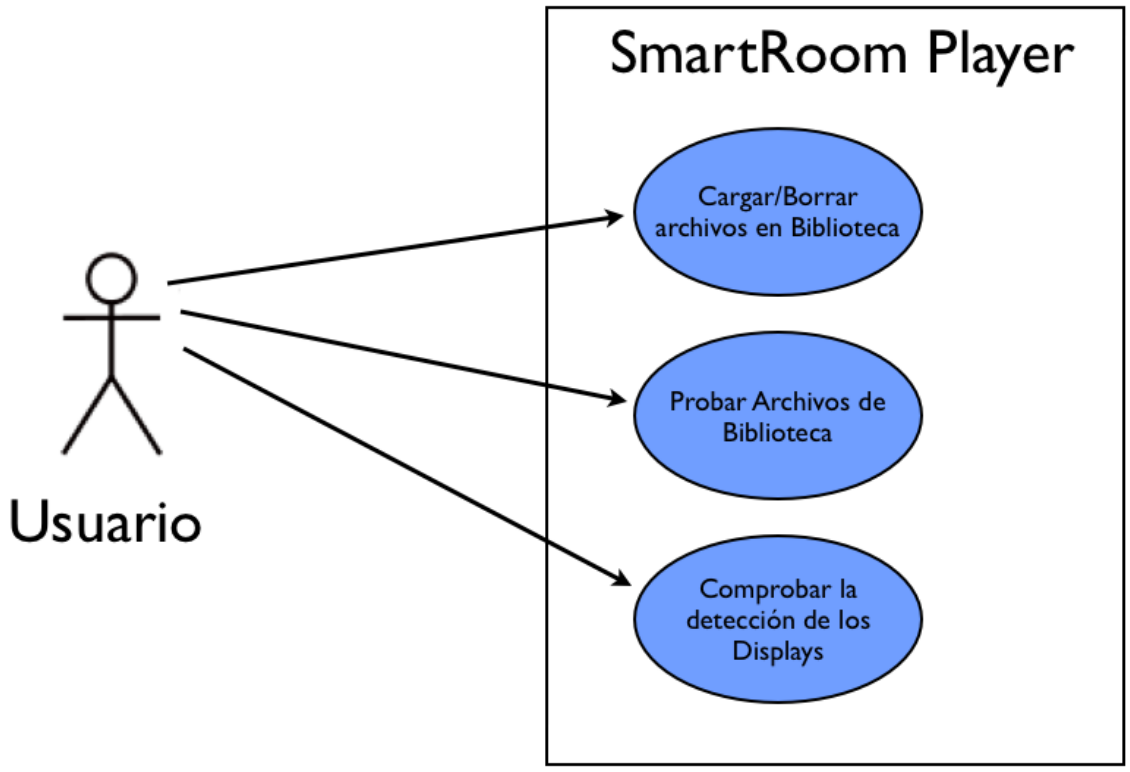


Ilustración 18: Casos de Uso de SmartRoom Player

Caso de Uso: Cargar Archivo de Biblioteca

Descripción:

El usuario puede cargar un archivo directamente desde el ordenador donde se encuentra SmartRoomPlayer a la biblioteca de la aplicación.

Escenario:

1. El usuario selecciona el archivo que quiere incluir en la biblioteca.
2. Se verifica que el archivo ha sido introducido en la biblioteca

Caso de Uso: Probar Archivo de Biblioteca

Descripción:

El usuario puede elegir qué archivo probar para ver si funciona en el reproductor.

Escenario:

1. El usuario selecciona el Display en el que quiere realizar la prueba.
2. El usuario selecciona el archivo que quiere reproducir en la biblioteca.
3. El archivo se reproduce con normalidad en el Display elegido.

Error: Si la aplicación no es capaz de reproducir el archivo elegido debido a que no soporta el formato se informará al usuario. Esto no eliminará el archivo de la biblioteca.

Diagrama de Clases

El siguiente diagrama muestra las clases involucradas en el SmartRoom Player. Se puede observar que todas las clases están alrededor del controlador ControlJMF. Éste tiene acceso a las ventanas de reproducción que se ejecutan en el medio. Cada una incluye un Applet (ya sea reproductor para media o de Processing) y un dispositivo gráfico sobre el que se va a ejecutar.

Por cada dispositivo disponible existirá una ventana de Reproducción, siempre bajo la supervisión del controlador.

Este esquema sigue el Modelo Vista Controlador, siendo el controlador la clase ControlJMF, la Vista la VentanaReproducción y el Modelo el Applet.

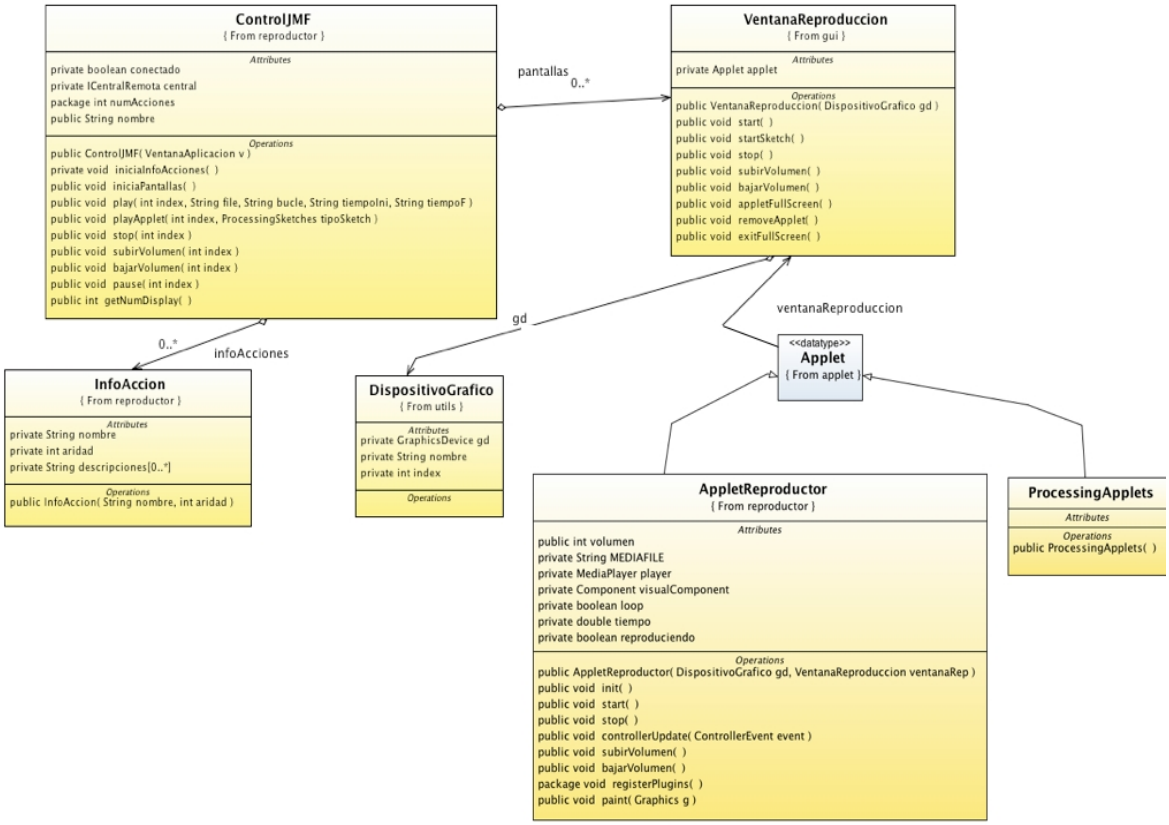


Ilustración 19: Diagrama de Clases de SmartRoom Player

Diagramas de Secuencia

Diagrama de Secuencia del controlador del medio con el método **play**()

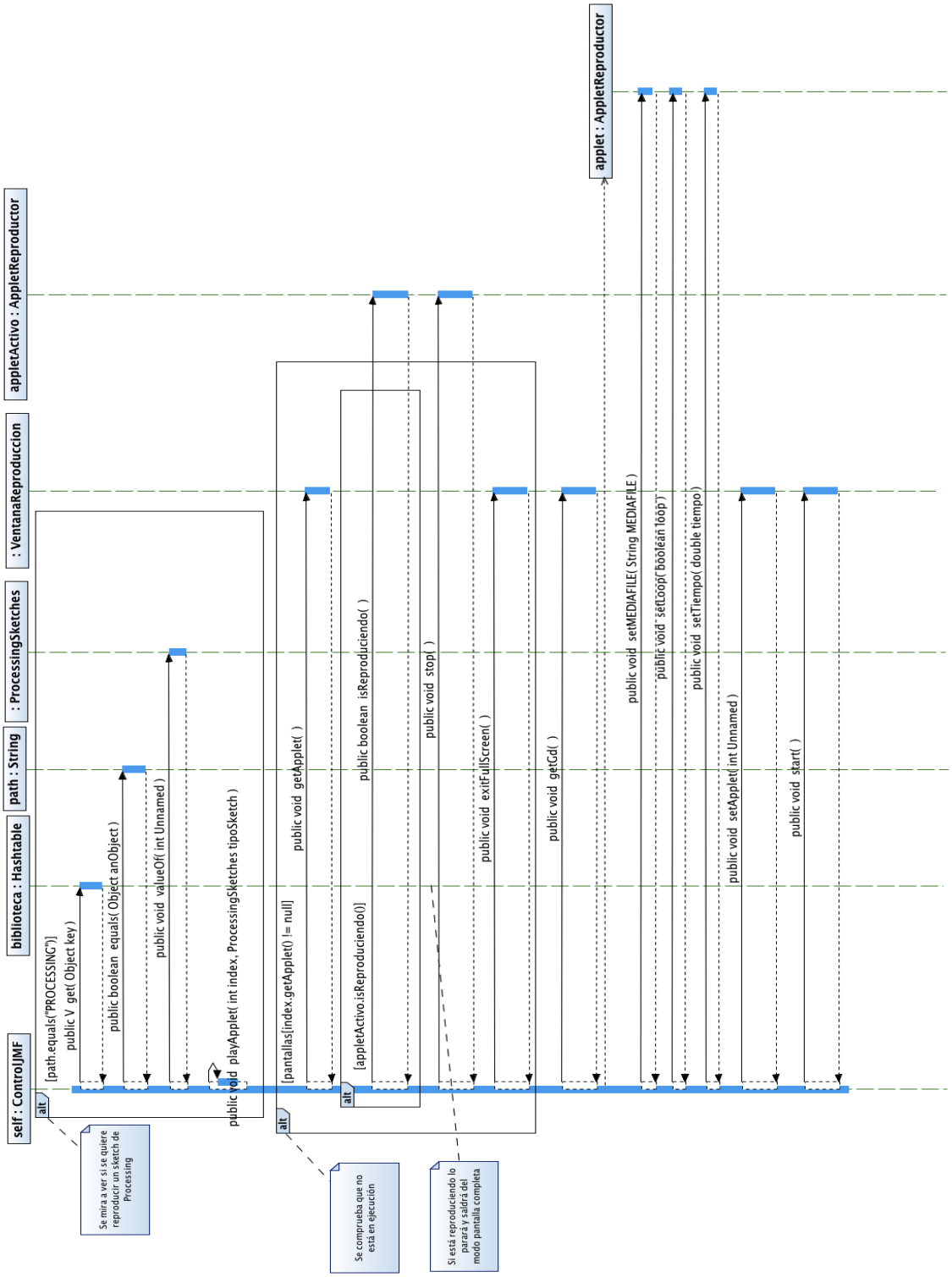


Ilustración 20: Diagrama de Sencuencia del Controlador

En un primer momento se comprobará si lo que se desea reproducir es un Applet de Processing. En tal caso se hará una llamada al método oportuno para reproducirlo.

En caso contrario se hará un chequeo de si la VentanaReproduccion se encuentra en uso. En caso de no estarlo se eliminará el AppletReproductor que estaba en uso, creándose otro nuevo con los atributos de reproducción indicados. Si la VentanaReproduccion se encontraba en uso se saldrá de la función sin afectar al estado de ésta.

c) Arquitectura de SmartRoomCenter

SmartRoomCenter se presenta como la aplicación principal de gestión de todo el sistema. Ofrece una interfaz al usuario que le permitirá gestionar diversas actividades que comprenden desde labores de conexión hasta la creación y mantenimiento de la base de reglas.

Dada la extensión de este módulo se dividirá su explicación mediante diagramas que faciliten la comprensión de su organización.

Casos de Uso

Se explican únicamente los relacionados con el sistema de reglas. Más adelante se comentarán los relacionados con la conexión y su interacción con los demás elementos del sistema.

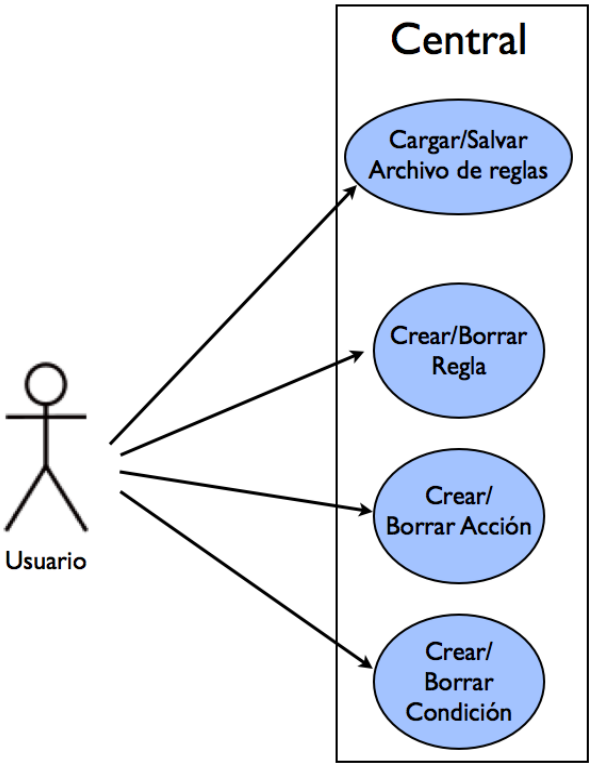


Ilustración 21: Diagrama de Casos de uso de la Central

Caso de Uso: Cargar Archivo de Reglas

Descripción:

El usuario puede cargar la base de reglas, condiciones y acciones previamente guardadas.

Escenario:

1. El usuario selecciona el archivo de reglas.
2. Si el archivo es válido: (ir paso 3) Si no: (ir Error)
3. Se leen reglas y son cargadas en el motor quedando activas.
4. Se muestran en la interfaz las reglas obtenidas.

Error: Si el archivo no es válido se informa al usuario. No se modifica la base de reglas actual.

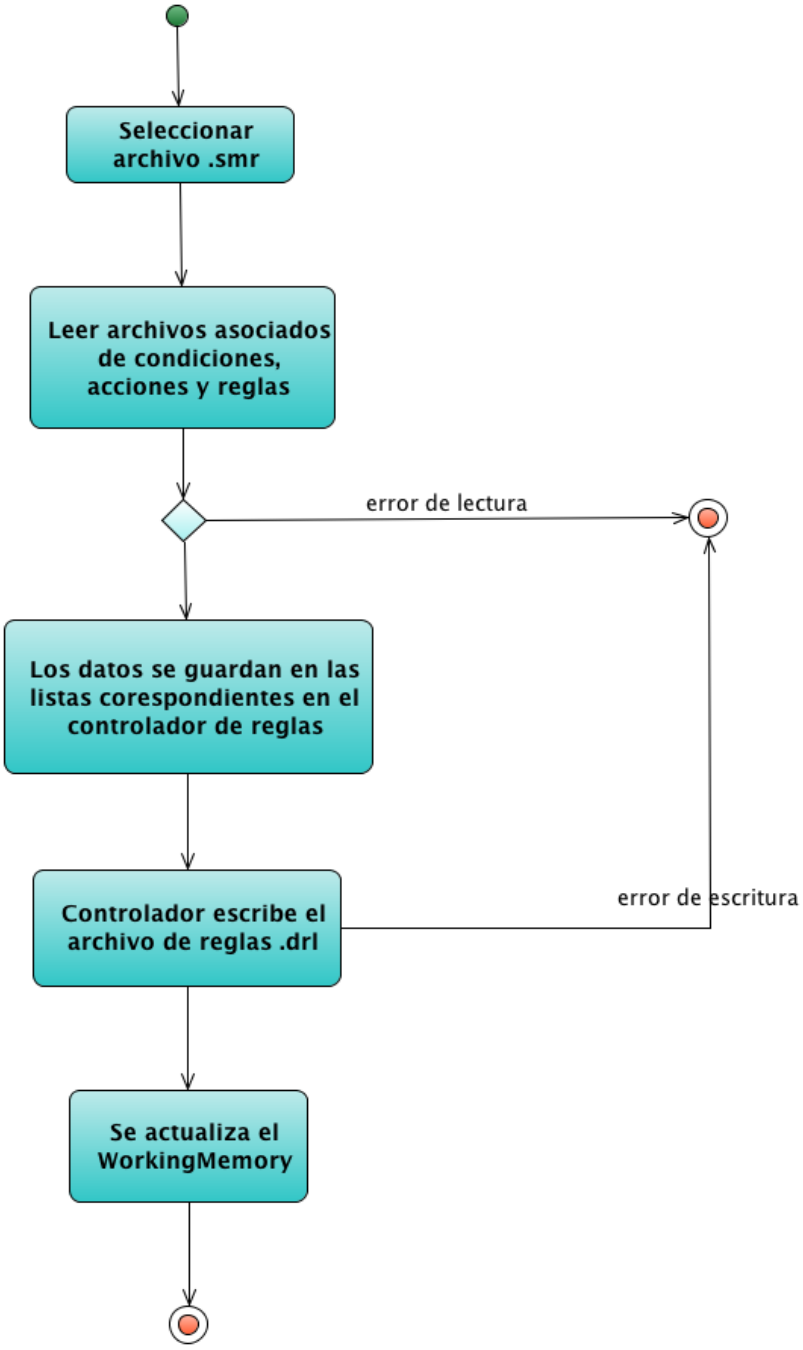


Ilustración 22: Diagrama de Actividad de carga de Archivo de reglas

Caso de Uso: Salvar Archivo de Reglas

Descripción:

El usuario puede salvar las condiciones, acciones y reglas que estén definidas.

Escenario:

1. El usuario selecciona una ruta y un nombre de archivo.
2. Se crea el archivo en la ruta indicada.

Caso de Uso: Crear Condición

Descripción:

Permite al usuario crear una condición para una regla a partir de la información que pueda proporcionar un sensor conectado al sistema.

Escenario:

1. El usuario selecciona un sensor.
2. El sistema muestra al usuario los posibles estímulos que pueden ser detectados por el sensor.
3. El usuario elige uno de los estímulos detectables y define una condición.
4. Se da un nombre a la condición.

Caso de Uso: Crear Acción

Descripción:

Permite al usuario crear una acción para una regla que pueda ser ejecutada en uno de los medios conectados al sistema.

Escenario:

1. El usuario selecciona un medio.
2. El sistema muestra al usuario las posibles acciones que pueden ser ejecutadas por el medio así como sus parámetros.

3. El usuario introduce los valores para dichos parámetros siendo avisado en caso de olvidar alguno.

4. Se da un nombre a la acción.

Caso de Uso: Crear Regla

Descripción:

Permite al usuario crear una regla compuesta por condiciones y acciones creadas con anterioridad.

Escenario:

1. El usuario selecciona una condición de la tabla.

2. El usuario selecciona una o varias acciones de la tabla.

3. Se da un nombre a la regla.

4. Una vez creada, la regla pasa a estar activa.

Diagrama de Clases

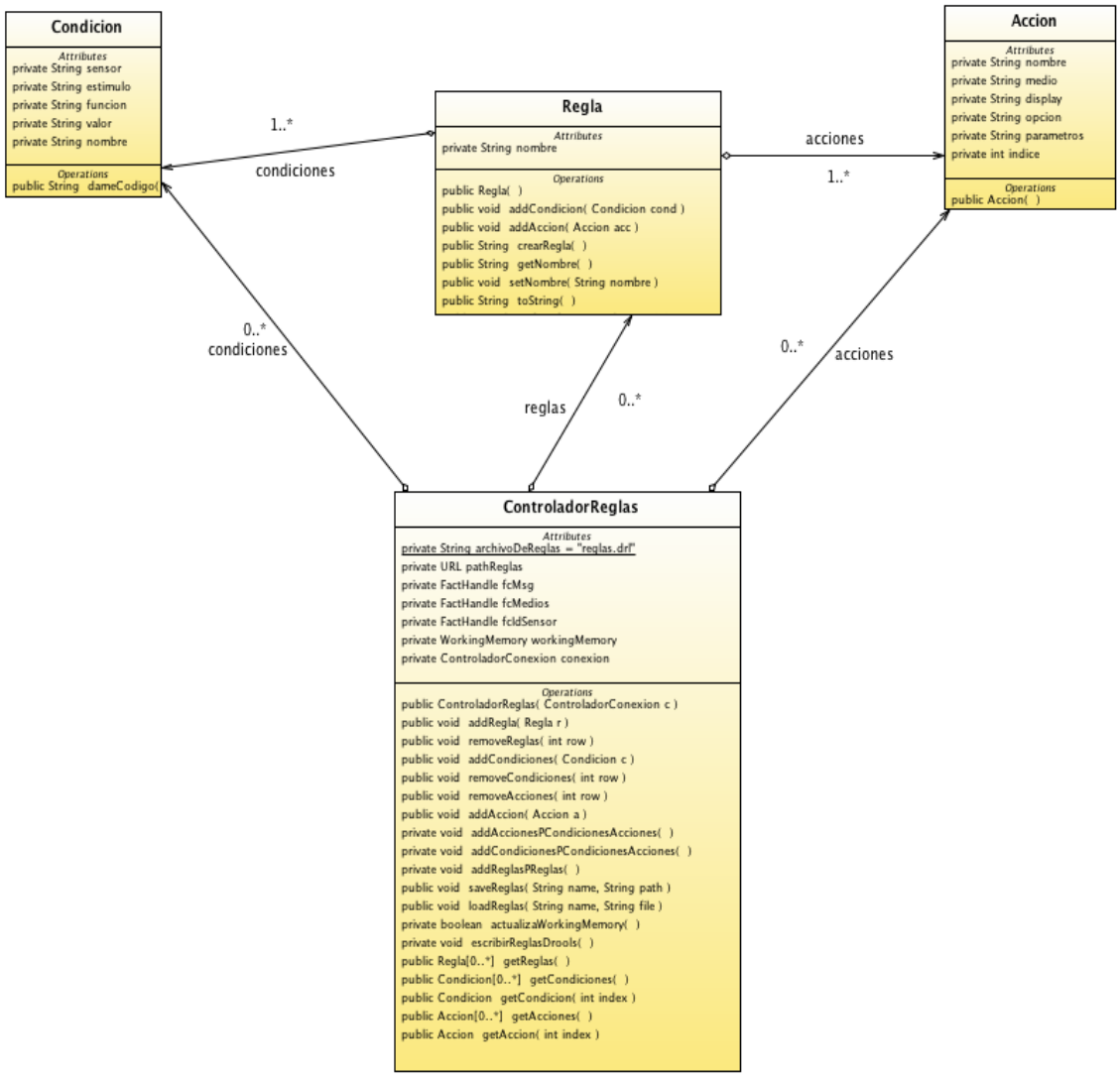


Ilustración 23: Diagrama de Clases de SmartRoom Center

Este diagrama muestra una idea global de cómo se articula el sistema de reglas. Puede observarse que una regla se compone de condiciones que deben cumplirse y de acciones que deben ejecutarse.

Hay que aclarar en este aspecto que, aunque el diseño de clases lo permita, se ha limitado a uno la cantidad de condiciones que componen una regla. Esta limitación está impuesta por la interfaz de usuario, que no permitirá añadir más de una condición a la misma regla.

El motivo de esto es que dos estímulos procedentes de distintos sensores no llegarán en el mismo mensaje por lo que no sería posible que esa regla pudiera activarse. De todos modos el diseño de clases aplicado permite la implementación de esto en un futuro.

La clase Regla es capaz además de auto-generarse como regla en formato *Drools* a partir de sus condiciones y acciones permitiendo así ser escrita e incluida más adelante.

Como ya se ha comentado anteriormente, el motor gestor de reglas usado es *Drools*. La clase que maneja este motor es ControladorReglas. Esta clase mantiene un WorkingMemory con las reglas que se encuentran activas y unos hechos (facts) con los estímulos recibidos. De esta manera *Drools* tiene todo lo necesario para poder activar y disparar las reglas oportunas.

Además también lleva a cabo la carga y guardado de reglas. Tanto las reglas como las condiciones y acciones son guardadas en formato XML. Para ello se ha hecho uso de una librería llamada XStream. Ésta simplifica notablemente las tareas de *serializar* objetos a XML. [Lib.1]

El formato para crear reglas en *Drools* es muy simple y sigue un patrón muy parecido al que se sigue en la definición de éstas en SmartRoomCenter.

```
<rule name="Detecta_3_caras_y_Play_video">
<parameter identifier="idSensor" >
<class>java.lang.String</class>
</parameter>
<parameter identifier="msg" >
<class>remote.IMensaje</class>
</parameter>
<parameter identifier="tablaMedios" >
<class>java.util.Hashtable</class>
</parameter>
<java:condition>
idSensor.equals("Sensor Caras")
</java:condition>
<java:condition>
msg.descripcion().equals("Caras Detectadas")
</java:condition>
<java:condition>
Integer.parseInt(msg.valor())==3
</java:condition>
<java:consequence>
if(((remote.IMedio)tablaMedios.get("Proyector1")) != null){try{((remote.IMedio)
tablaMedios.get("Proyector1")).play(0,"Eclipse.mov","si","","5");}catch (java.lang.Exception
ex){}}
</java:consequence>
</rule>
```

Ilustración 24: Ejemplo de regla definida en Drools

Puede apreciarse el paralelismo que existe entre las reglas que se definen desde la aplicación y el formato que finalmente adquieren dentro del archivo Drools. Las reglas se componen de un nombre, una serie de parámetros, condiciones que deben cumplirse y acciones (consecuencias en Drools).

En este caso la regla se llama "Detecta_3_caras_y_Play_video". La condición que debe cumplirse es que el "Sensor Caras" detecte "Caras Detectadas" y ese valor sea "== 3". Cuando un mensaje llega a la central es enviado al componente Drools y este lo procesa. Dentro del mensaje se encuentra el identificador del sensor que lo envía y el estímulo y valor detectado. Cuando estos valores satisfacen las condiciones de la regla se disparan las acciones pertinentes. En este caso sólo hay una: reproducción de "Eclipse.mov" en "Proyector1". Además puede observarse que esta función admite tres parámetros más.

Las reglas pueden estar compuestas de más de una acción. En el archivo "reglas.drl" se guardan las reglas que se encuentran en el WorkingMemory de Drools .

Al salvar las reglas del sistema también se guardan las condiciones y acciones que se encuentren disponibles. Se generarán tres archivos además de uno con extensión ".smr". Dentro de cada uno de estos archivos se guardarán las reglas, condiciones y acciones respectivamente. Todo ello se realiza mediante el uso de la librería XStream [Lib.1].

CAPÍTULO 4

4.Comunicación Remota

4.1.Visión general

Uno de principales requisitos de la aplicación era que fuera distribuida, lo que posibilitaría que tanto la Central como los Medios y los Sensores pudieran ser ejecutados en diferentes máquinas. Para ello la tecnología escogida fue CORBA. Gracias a esta tecnología se ha desarrollado una aplicación distribuida, multi-plataforma y capaz de funcionar en cualquier tipo de red, haciendo más sencillo la comunicación entre objetos alojados en distintas máquinas.

El requisito más importante fue fijar de forma clara el interfaz de comunicación entre los distintos objetos distribuidos.

El *IDL*(Inteface Description Language) [IDL1] permite describir dicho interfaz de comunicación y además generar automáticamente las clases necesarias para poder llevar a cabo dicha comunicación.

En este caso se han necesitado tres Interfaces principales para describir las operaciones disponibles remotamente de *Central*, el *Medio* y el *Sensor*. Tres clases auxiliares: *InfoSensor* e *InfoMedio (*para determinar la información a cerca de los sensores y los medios) y *Mensaje* para encapsular el estímulo que desde el sensor puede ser notificado a la Central.

La Central es la encargada de crear y manejar el bus ORB y registrar a los medios y sensores que se conocen. Tanto medios como sensores tienen dos formas de conectarse. En la primera es la central la que se encarga de asignarle un identificador, la segunda permite al extremo proponer un nombre. Si no existe otro sensor o medio con ese mismo nombre se seguirá el proceso de registro. Una vez verificado que el identificador es único se procede a registrar el nuevo medio/sensor en el *Servicio de Nombres*. De esta forma se podrá localizar cada objeto por la cadena que lo identifica. Si no se produce ningún tipo de error la conexión quedará establecida, permitiendo definir condiciones y acciones según el tipo.

Para desconectar tanto medios como sensores, llamarán al método *desregistrar()* de la central que les avisará de que han sido desconectados y borrará sus referencias. Si la central se desconecta se informará a todos los medios y sensores registrados de que ya no están conectados con la Central.

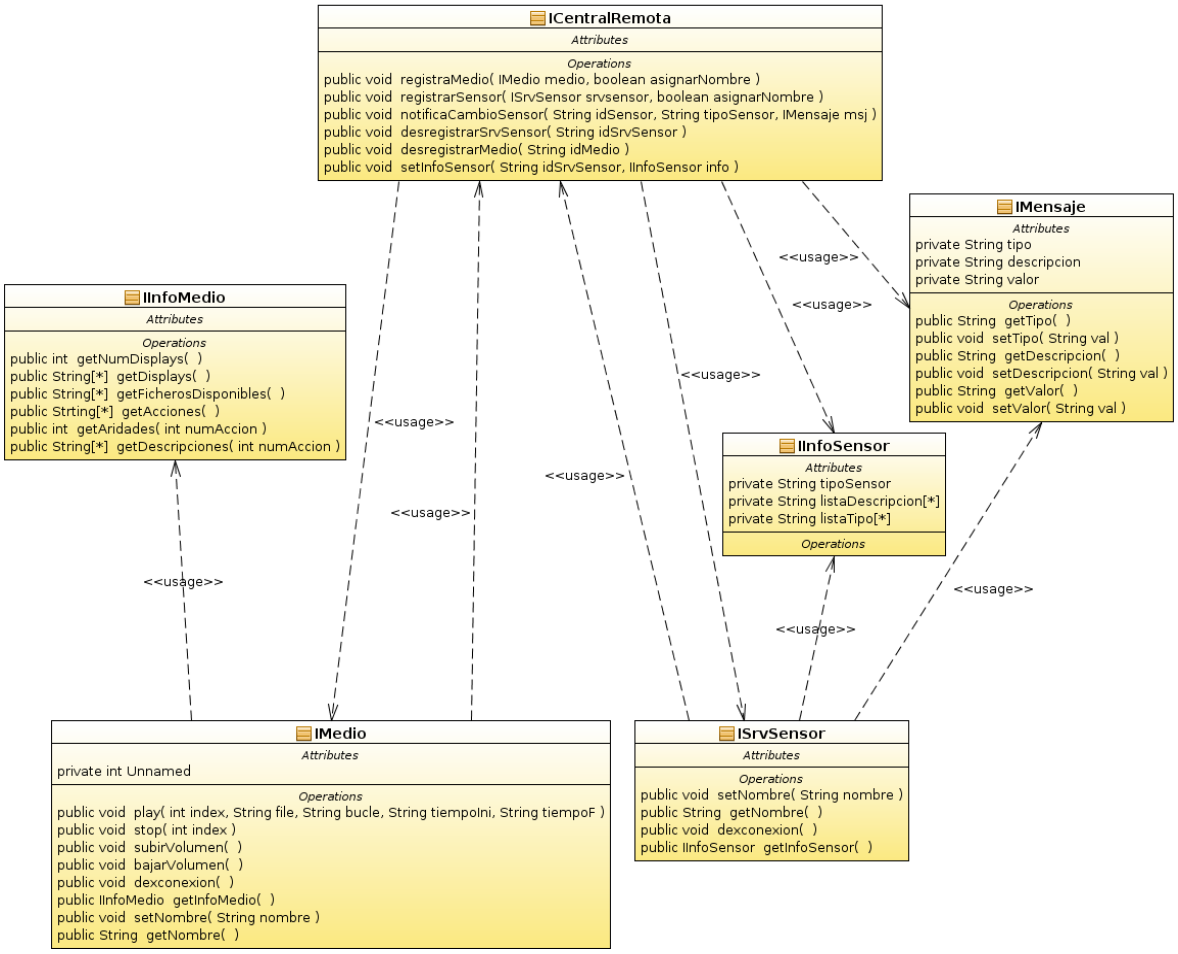


Ilustración 25: Diagrama de clases de los objetos distribuidos

a) Uso del Servicio de Nombres

CORBA permite registrar referencias a objetos en el *servicio de nombres*. Esto facilita a la central recuperar las referencias a los Medios y Sensores conectados y además que estos al conectarse recuperen una referencia al objeto Central.

El nombre con el que se registran los medios y sensores debe ser único para evitar errores. Cuando un Medio o Sensor solicita la desconexión, la Central lo elimina del servicio de nombres.

La central se registra a su vez en el servicio de nombres para que pueda ser recuperada por los sensores y medios cuando estos soliciten la conexión.

b) Ejemplo de Actividad

Un ejemplo de actividad entre las distintas aplicaciones del sistema sería el siguiente:

El Sensor capta un estímulo y genera un Mensaje. Utilizando la clase *ImensajeHelper* se genera una referencia de tipo *IMensaje*. Se llama al método de central *notificaCambio()* indicándole el identificador del sensor que captó el estímulo, el tipo de sensor y el Mensaje que contiene la información del estímulo.

El método notificaCambio() en Central llamará al método que consulta la base de reglas y si alguna es lanzada, ésta llamará al método de la regla en el medio especificado.

4.2.SmartRoom Central

La Central posee dos estados Conectado y Desconectado. Cuando la central se conecta se inicializa el bus ORB y el Servicio de Nombres. Cuando la central se encuentra Conectada permite que se conecten a ella varios sensores y medios. Cuando un medio o sensor se conecta, lo registra en el Servicio de Nombres y guarda su referencia y la Información (InfoMedio o InfoSensor) en una Tabla. Los sensores una vez conectados envían la información que recogen a la Central.

La Central a su vez puede llamar a los métodos de los Medios conectados para ejecutar las acciones.

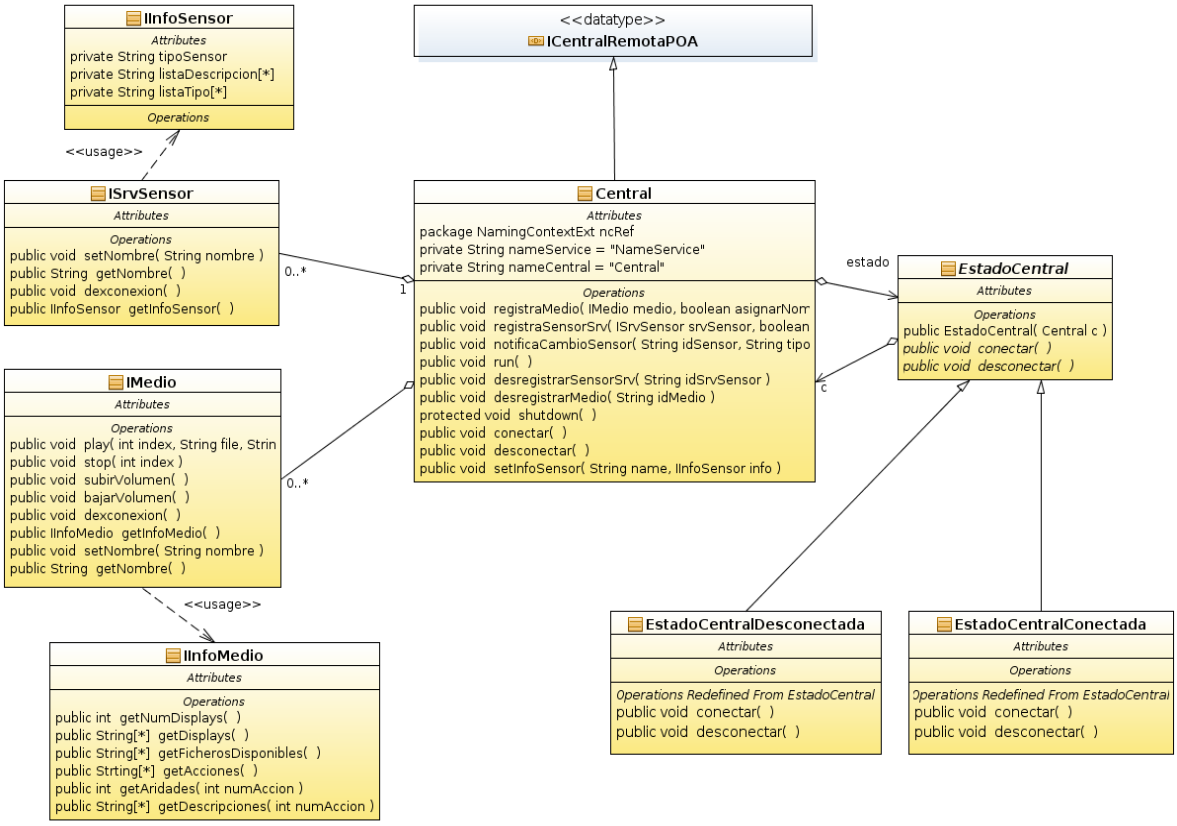


Ilustración 26: Diagrama de clases de Central

4.3.SmartRoom Sensor

a) Diseño

Para el sensor se diseñó una máquina de estados. De tal manera que el comportamiento del sensor varía en función del estado en el que se encuentre.

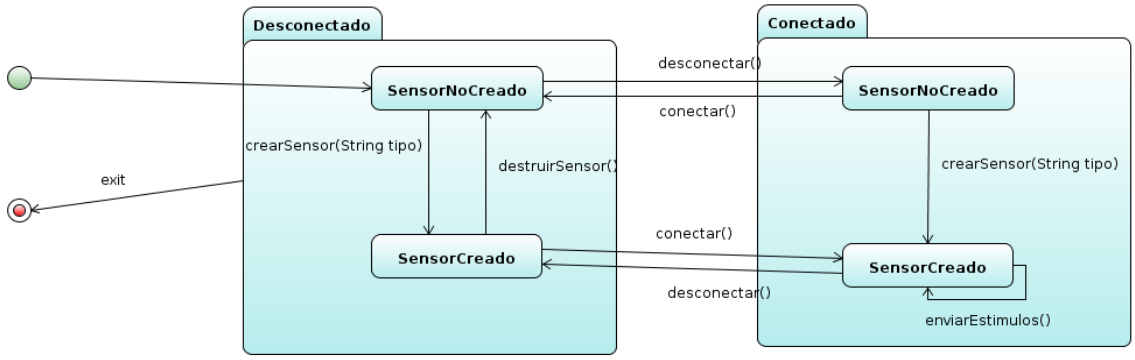


Ilustración 27: Diagrama de estados de un sensor

Existen dos super-estados principales que representan la conexión con la *Central*.

Desconectado

Cuando la aplicación está desconectada se puede crear o destruir sensores. Los sensores pueden ser configurados antes de ser creados; éstos funcionan y generan estímulos pero no son enviados remotamente.

Conectado

En este estado la aplicación se comporta según tenga un sensor creado o no.

Si el sensor está creado y se establece la conexión con la central, se le informa del tipo sensor (lo que puede detectar) que ha sido creado. Si la conexión tiene éxito el Sensor empieza a enviar notificaciones de sus cambios a la central.

Si el sensor no está creado, la central solo recibe la conexión del sensor pero no sabe cuales es su información a cerca de los estímulos posibles. La central puede considerarlo como inactivo.

Si se recibe la señal de desconexión desde la central, se pasará al estado correspondiente de *Desconectado* según esté el sensor creado o no. Si el

programa se cierra y el sensor está creado se informa a la central de que el sensor ya no existe y se le pide la desconexión.

b) Implementación

El patrón State ha servido como referencia para implementar los posibles estados del Sensor. Las ventajas en el uso de este patrón son flexibilidad para modificar la máquina de estados y las transiciones entre los mismos. Además facilita implementar el comportamiento que ha de tener el sensor en un estado dado y el control de errores.

Aquellas operaciones que no son posibles en un estado determinado lanzarán una excepción que el sistema capturará. Además el estado pasará al siguiente después de realizar la operación.

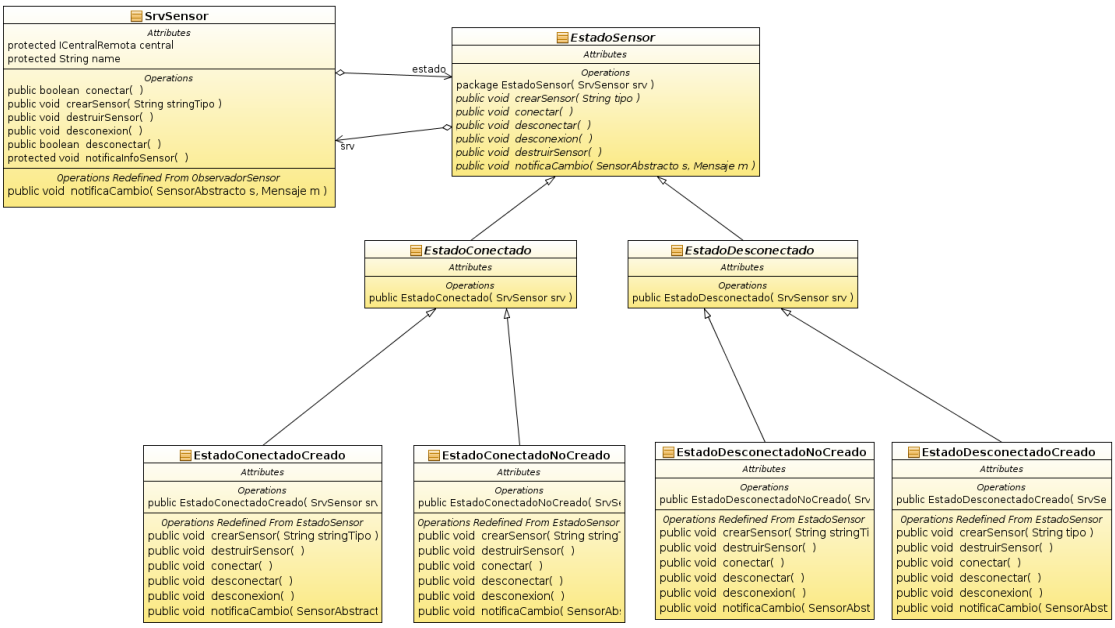


Ilustración 28: Estados del sensor basados en el patrón State

c) Comunicación con la central

El sensor a través del Servicio de Nombres localiza la referencia a la Central, llama al método registrarSensor() y guarda la referencia a dicha Central para futuras peticiones. Para notificar un cambio utiliza esa referencia para llamar al método notificaCambioSensor() al que le pasa como parámetro su nombre en el Servicio de Nombre, su tipo y el Mensaje con la información del cambio.

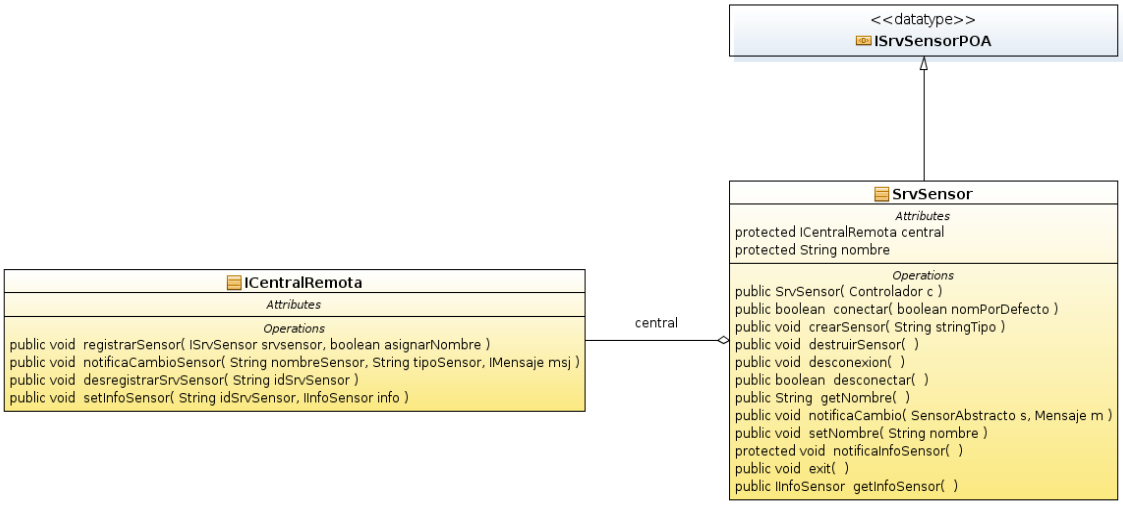


Ilustración 29: Comunicación entre Sensor y CentralRemota

4.4.SmartRoom Medio

a) Comunicación con la Central

Al igual que el Sensor, el Medio localiza la referencia a la central a través del Servicio de Nombres y se registra. Una vez registrado el Medio actúa como servidor atendiendo a las peticiones de la Central. Ésta ejecutará sus métodos para llevara acabo las acciones definidas en sus reglas.

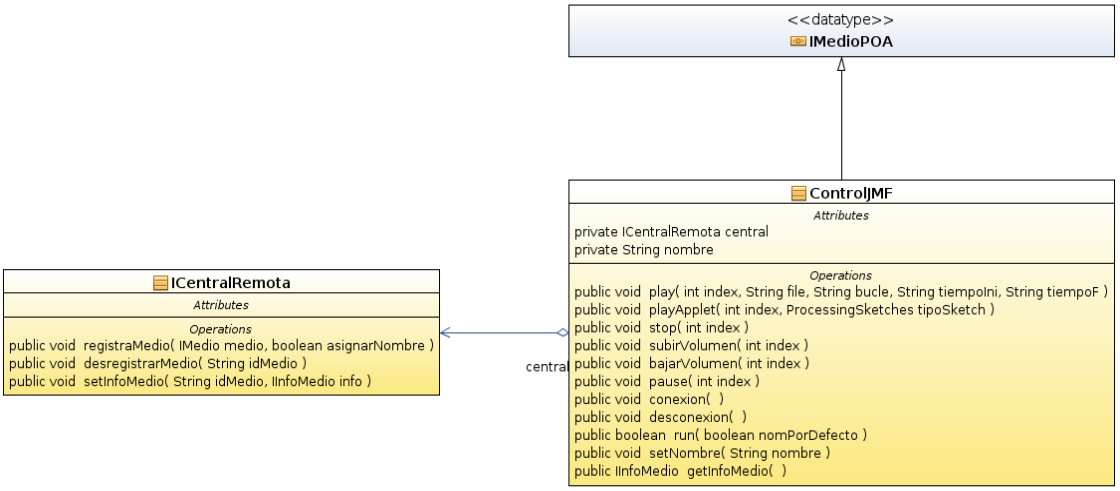


Ilustración 30: Comunicación entre Medio y CentralRemota

4.5.Actividad del Sistema

a)Diagramas de secuencia

A continuación se detallan algunas de las actividades más importantes del sistema por medio de diagramas de secuencia.

Conexión de un Sensor/Medio

Este es un diagrama simplificado que muestra el proceso de registro de un sensor pero es válido también para los medios. Cabe destacar que la central es la encargada de registrarlo en el bus ORB a través del método *bind()* y añadir su referencia en el Servicio de Nombres.

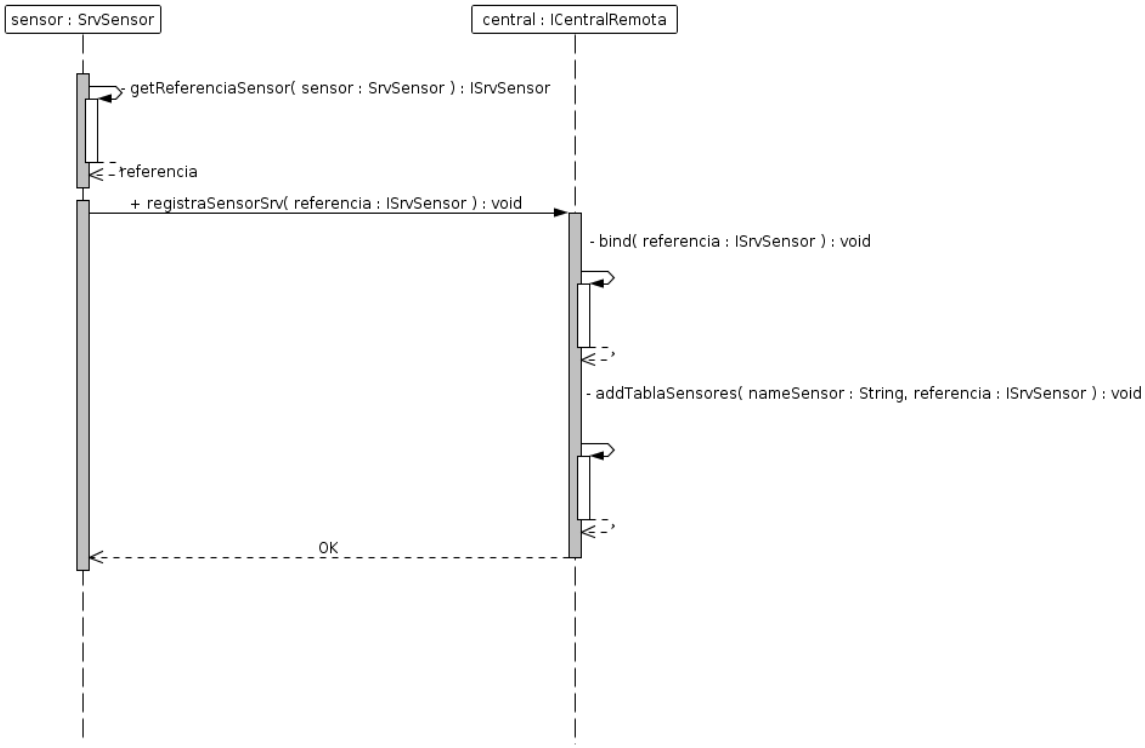


Ilustración 31: Secuencia de registro de un Sensor

Desconexión de un Sensor/Medio

Cuando un sensor o medio se desconecta, llama al método de la central desregistrarSensor() y desregistrarMedio() respectivamente. Esta se encarga del proceso de borrado del sensor del registro. Al igual que al desconectar, es la central la que se encarga de borrar su referencia en el Servicio de Nombres Se ha omitido el control y tratamiento de los errores.

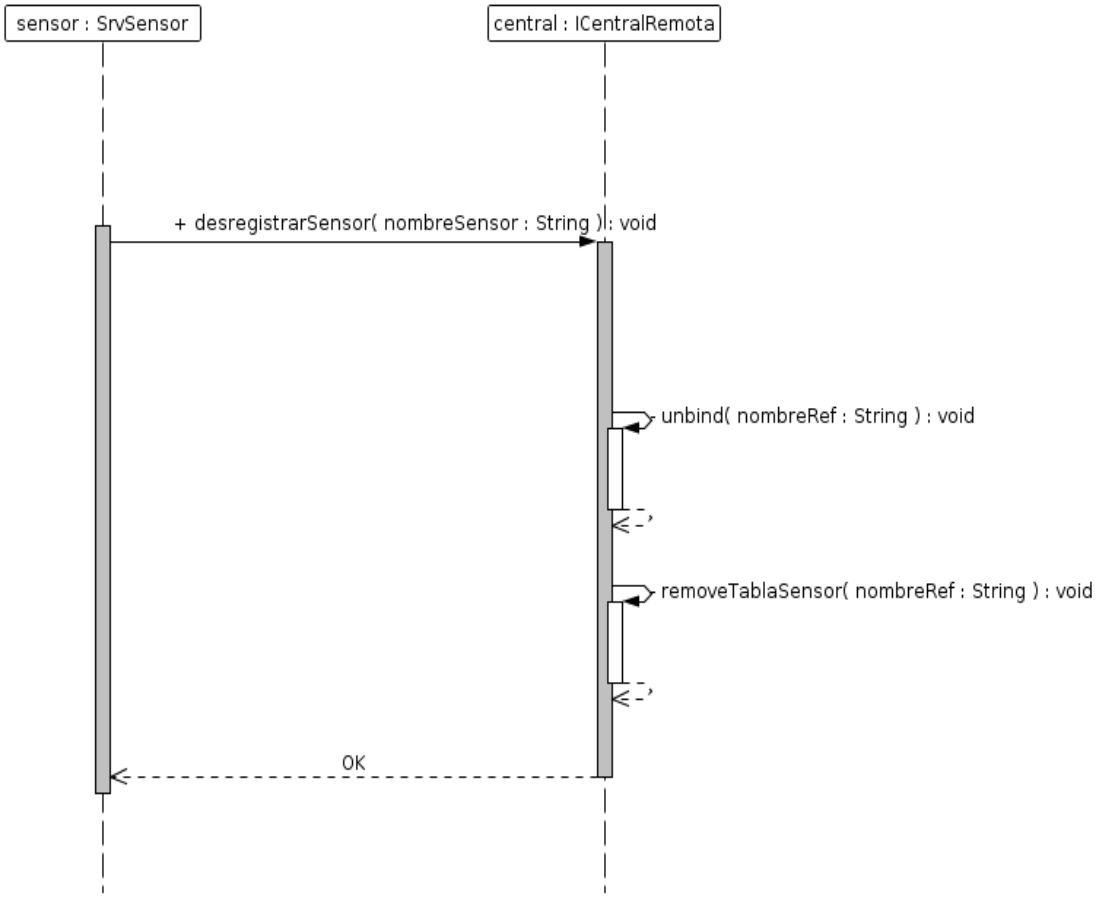


Ilustración 32: Secuencia de desconexión de un Sensor

Desconexión de la Central

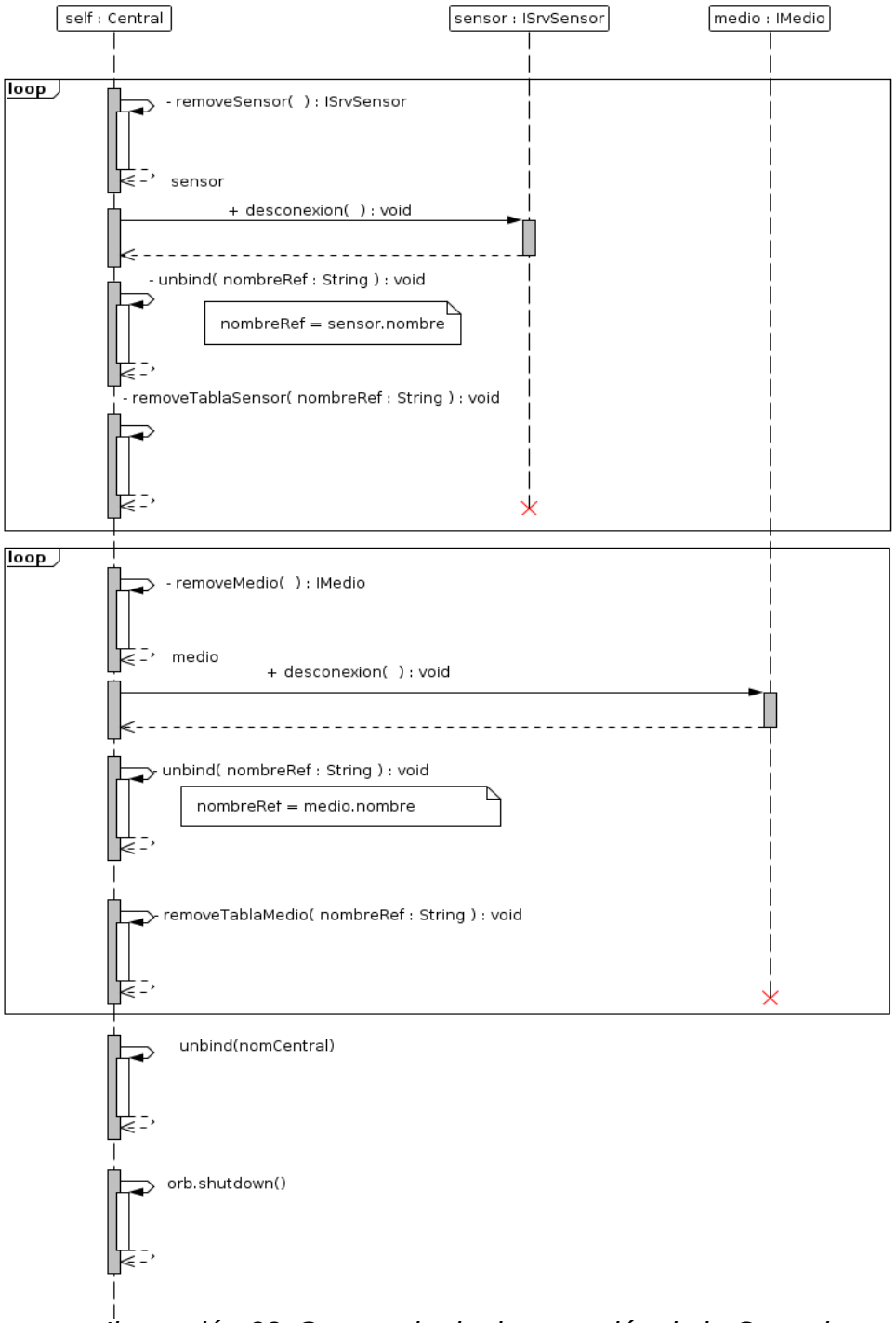


Ilustración 33: Secuencia de desconexión de la Central

El diagrama anterior muestra de manera simplificada el proceso de desconexión de la central.

Para cada uno de los Sensores que se han conectado al sistema la central llama a su método desconexión() para notificarles que han sido desconectados y a continuación elimina sus referencias del bus. Lo mismo pasa con los Medios. Una vez eliminadas todas las referencias, elimina su propia referencia del bus y procede a su apagado llamando al método shutdown().

En el caso de que un medio o sensor se haya desconectado de manera brusca, la central tratará la excepción al invocar su método desconexión() y procederá de forma normal el proceso de borrado.

CAPÍTULO 5

5.Creación y Configuración de instalaciones artísticas

Uno de los principales retos de la aplicación es lograr abstraer al usuario, en este caso un artista, de los conocimientos más propios de la programación. Se pretende dar facilidad de uso y manejo pero teniendo a la vez un sistema versátil. Para lograr este objetivo se ha tratado de presentar una interfaz clara y sencilla, dividida en tres partes: sensores, medios y una central.

A continuación se explicará el funcionamiento de cada una de estas aplicaciones, configuración y uso.

5.1.Escenarios concretos

Las distintas reuniones con el colectivo MASDELOMISMO [1] permitieron diseñar varios ejemplos de instalación artística.

Al final se acordó planificar, e implementar para posibles concursos, dos tipos de instalaciones.

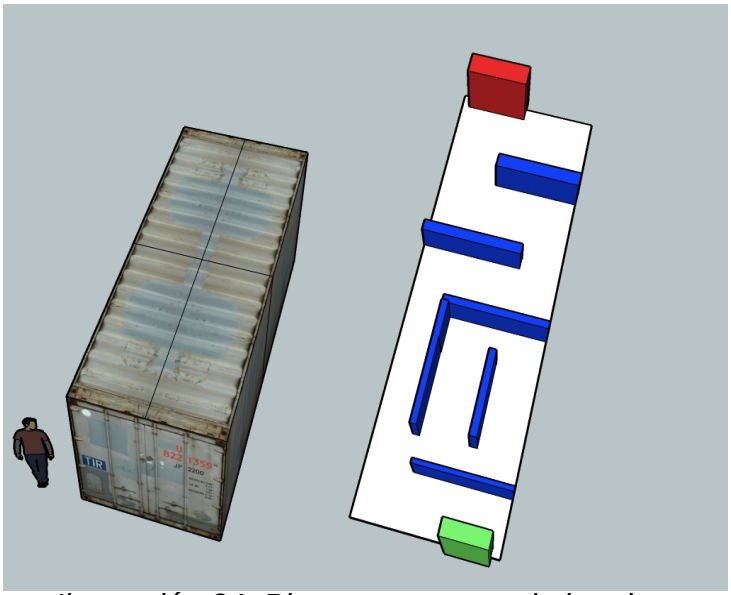


Ilustración 34: Plano y esquema de la primera instalación

Los artistas de MASDELOMISMO comentaron que ya habían utilizado algunos contenedores para realizar otro tipo de instalaciones hace tiempo. Investigando sobre los posibles usos de un espacio cerrado con la aplicación, se llegó a la conclusión de usar uno de los contenedores como una especie de laberinto, donde se entrase por uno de los lados y la salida estuviese en el lado contrario.

El visitante trata de llegar a la salida. Durante el trayecto se ejecutan reglas disparadas por los sensores y acciones en los distintos medios. Todos ellos distribuidos por el contenedor.

El aspecto final, con los sensores y medios situados sería el siguiente:

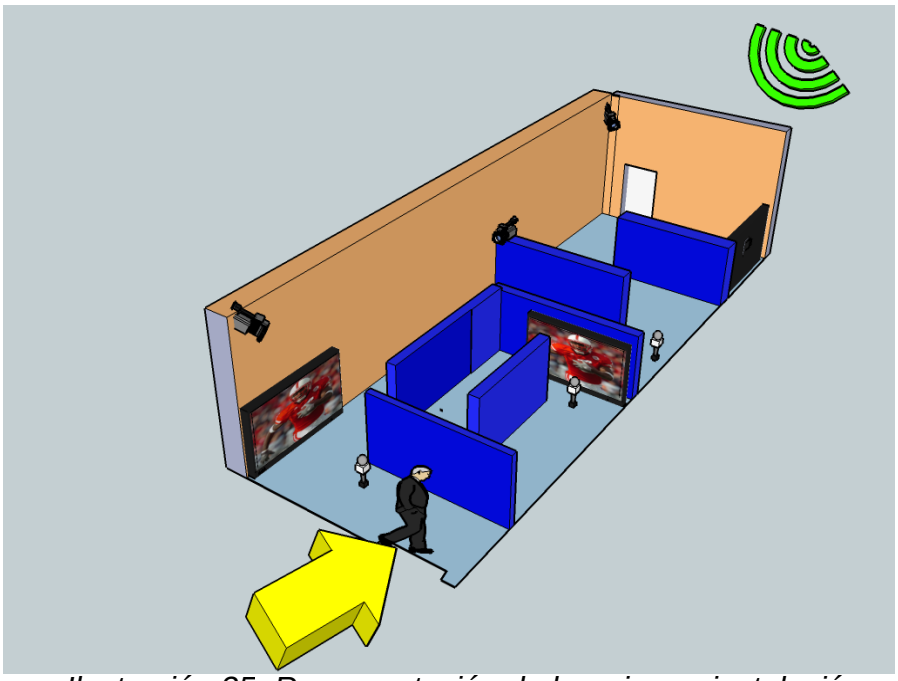


Ilustración 35: Representación de la primera instalación

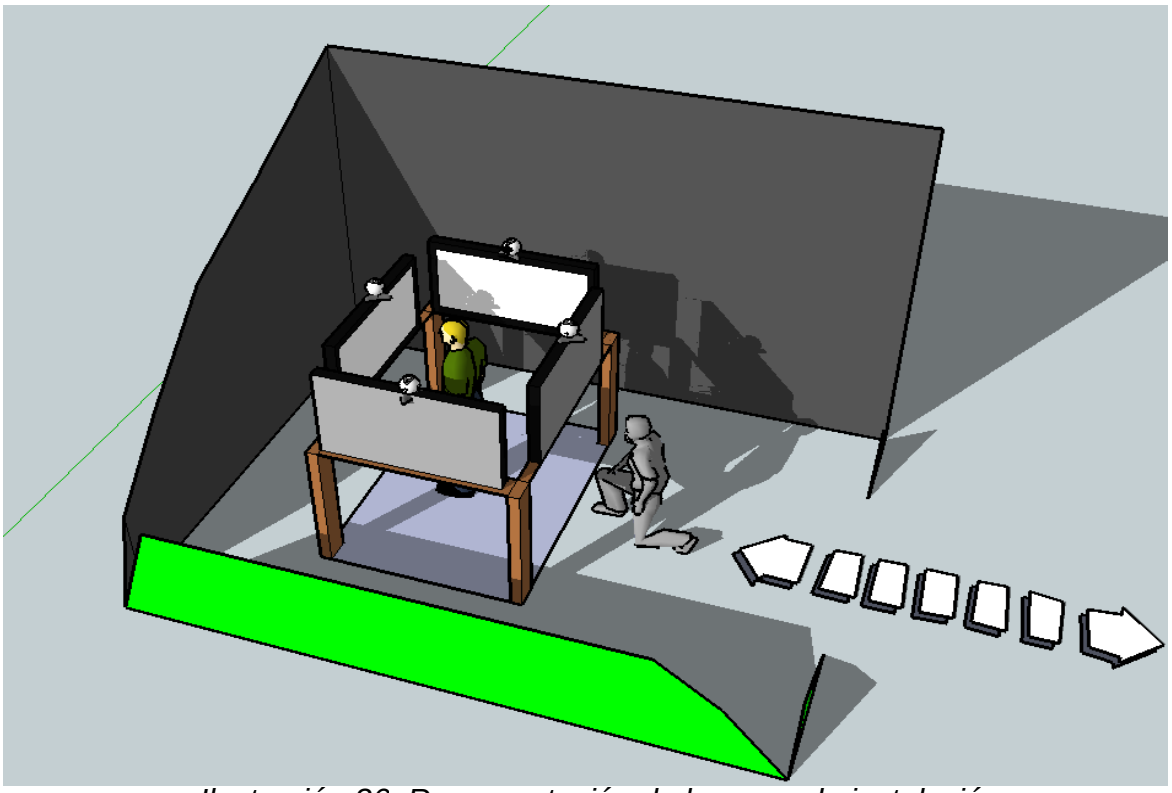


Ilustración 36: Representación de la segunda instalación

Y el centro de control

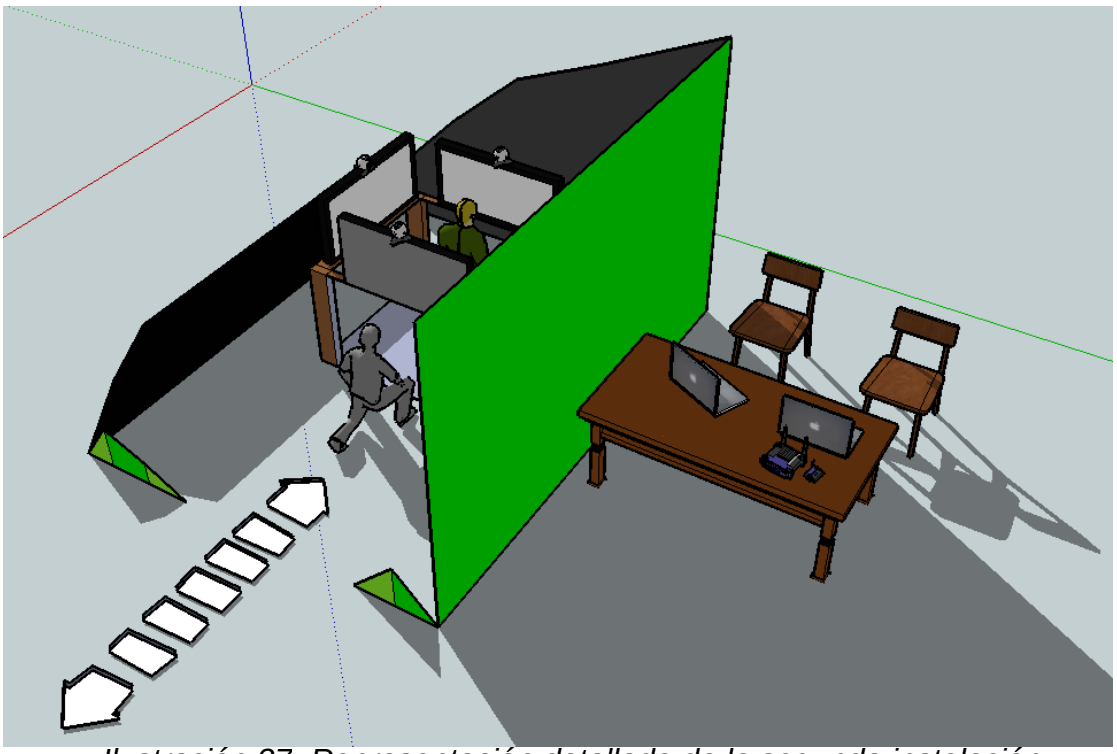


Ilustración 37: Representación detallada de la segunda instalación

Ésta instalación es la que se presentará conjuntamente con el colectivo MASDELOMISMO al concurso "Encuentros Digitales".

Es un concurso del "Centro de Arte Complutense", un museo recién inaugurado y dirigido al arte contemporáneo. Este concurso se fallará en septiembre de este año (2010).

5.2.SmartRoomSensor

Esta aplicación en su versión actual permite crear tres tipos de sensores: detector de movimiento, detector de caras y detector de sonidos. Además para cada uno de ellos podrán aplicarse configuraciones más específicas.

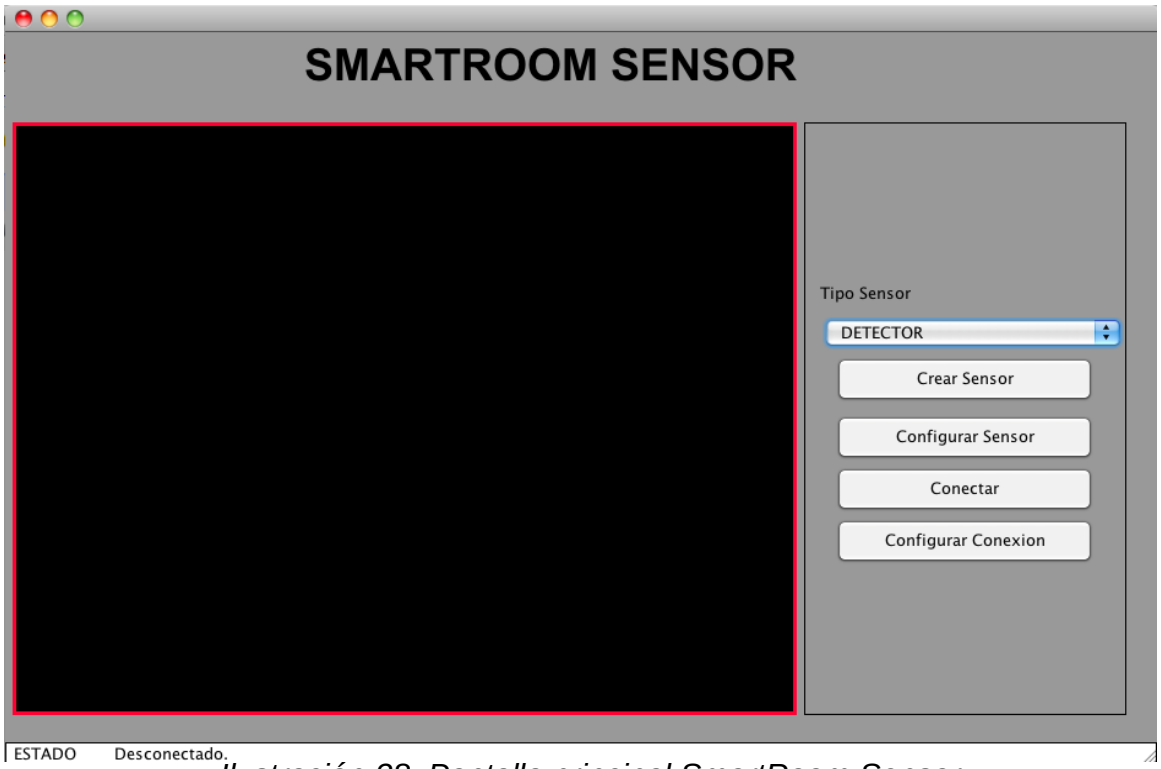


Ilustración 38: Pantalla principal SmartRoom Sensor

La figura anterior muestra la interfaz principal de SmartRoomSensor. En ella el artista podrá configurar y crear un sensor eligiéndolo de entre la lista de sensores disponibles.

La aplicación es capaz de funcionar de manera autónoma pero para poder montar una instalación será necesario que se encuentre conectada con la aplicación central. Para ello pueden cambiarse los parámetros de conexión si se desea. Dentro de estos parámetros se encuentra el nombre que tendrá el sensor dentro de la instalación. Será interesante dar un nombre apropiado que ayude a

identificarlo claramente para facilitar las labores de creación de reglas desde la central. Si no se especifica ninguno será la central la que se lo asigne de manera automática.

Además será posible cambiar la dirección IP y puerto de la central a la que se desea conectar.

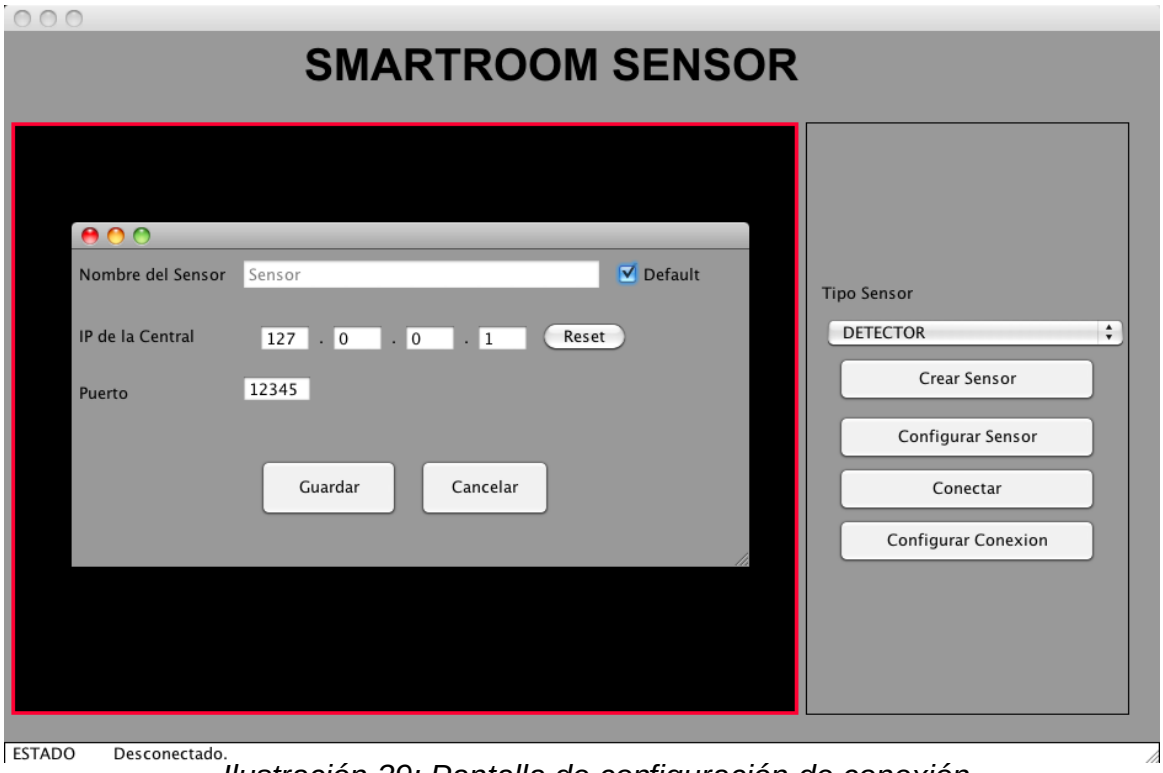


Ilustración 39: Pantalla de configuración de conexión

Esta aplicación permite únicamente la creación de un solo sensor. En caso de querer utilizar varios en un mismo equipo será necesaria la ejecución simultánea de otra aplicación SmartRoomSensor. De todos modos, puede no ser aconsejable ejecutar demasiados sensores a la vez dentro de un mismo equipo puesto que algunos de estos requieren una capacidad de procesamiento algo elevada.

A continuación se pasará a describir cada uno de los sensores disponibles.

a)Sensor Detector de Movimiento (Detector)

Este sensor permite al artista la detección de movimiento a través de una cámara. Para ello se divide el campo de visión de ésta en una cuadrícula. De esta manera el sensor notifica un cambio cada vez que detecta movimiento dentro del cuadrante.

El artista será capaz de crear condiciones para cada uno de estos cuadrantes pudiéndose crear reglas más complejas. La interfaz del sensor, cuando se encuentre en funcionamiento, mostrará dicha cuadrícula aplicando una serie de colores en función de si se detecta movimiento o no.

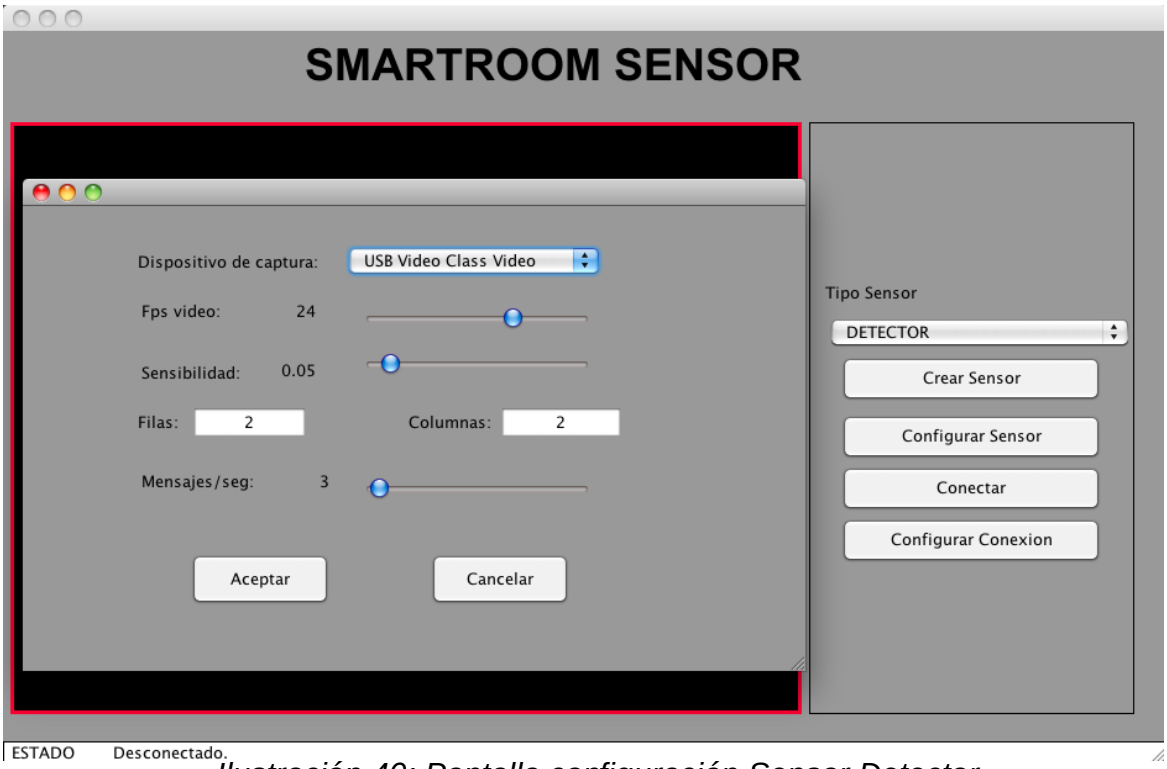


Ilustración 40: Pantalla configuración Sensor Detector

La pantalla de configuración de este sensor ofrece diversas posibilidades. Es posible elegir el dispositivo de captura y la tasa de frames que usará el sensor dentro de los que estén disponibles en ese momento. Además pueden

configurarse las dimensiones de la cuadrícula, la sensibilidad del sensor y la cantidad de notificaciones que se desean enviar por segundo.

Para poder acceder a la configuración del sensor es necesario que no esté actualmente uno en funcionamiento. En ese caso deberá destruirse previamente el existente.

b)Sensor Detector de Caras (Face)

Este sensor permite detectar caras dentro del ángulo de visión de la cámara. Una vez detectadas se notifica del número de ellas pudiendo así crear nuevas condiciones para las reglas. Cabe destacar que además de detectar caras de caras reales, es capaz de detectar caras impresas o incluso dibujos.

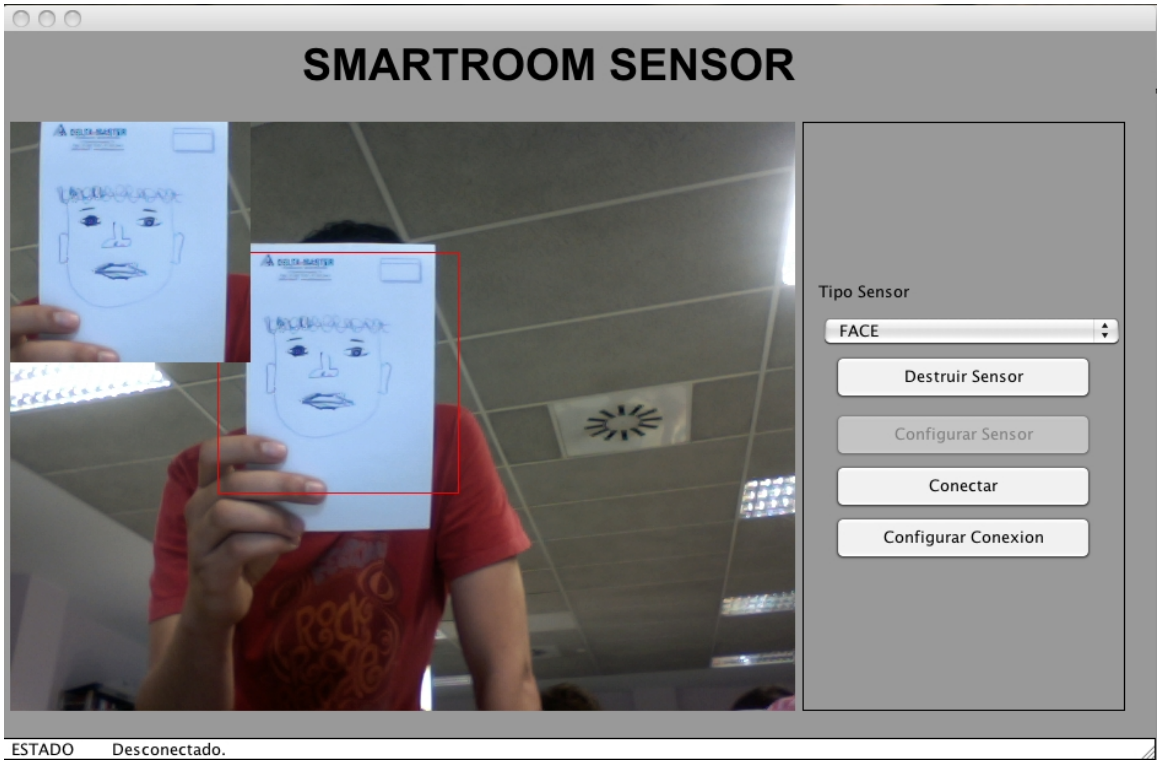


Ilustración 41: Pantalla de detección Sensor Face

Las opciones de configuración de este sensor son muy similares a las del detector. Es posible elegir el dispositivo de captura, los *frames* por segundo y la cantidad de notificaciones que se envían por segundo.

c)Sensor Detector de Sonido (Beat)

Este sensor permite detectar pulsos (beats), aumentos bruscos en la amplitud del sonido. De esta manera, aun habiendo un ruido contínuo de fondo, sólo se detectarán “beats” de sonido cuando haya cambios fuertes de más amplitud que el resto. Además el sensor indicará de manera gráfica que ha detectado un beat mediante una animación en la pantalla.

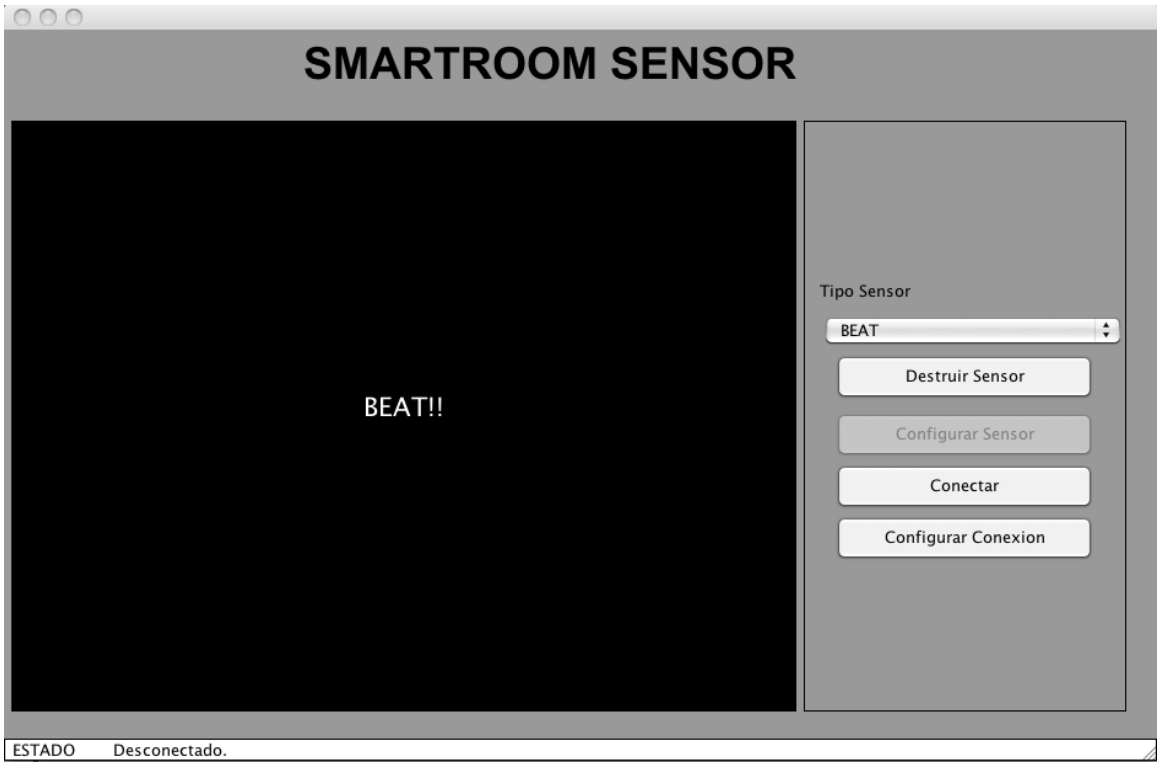


Ilustración 42: Pantalla de detección Sensor Beat

Las opciones de configuración serán la sensibilidad del sensor y la cantidad de notificaciones por segundo que se enviarán en forma de estímulos.

5.3. SmartRoomPlayer

Esta aplicación es la que finalmente ejecutará las acciones de las reglas que serán indicadas por la central. Aunque es posible implementar otros tipos de medios en un futuro, éste ha sido diseñado principalmente para realizar tareas multimedia. Es capaz de reproducir la mayoría de archivos multimedia incluyendo applets de Processing [31].

Para comenzar a utilizar SmartRoomPlayer será necesario conectar todos los dispositivos de salida de vídeo que se deseen utilizar para que la aplicación pueda detectarlos. Una vez hecho esto puede comprobarse que todo esté conectado al sistema mediante la pestaña de pantallas. Por defecto el sistema detectará como pantalla principal la que se tenga conectada al ordenador.

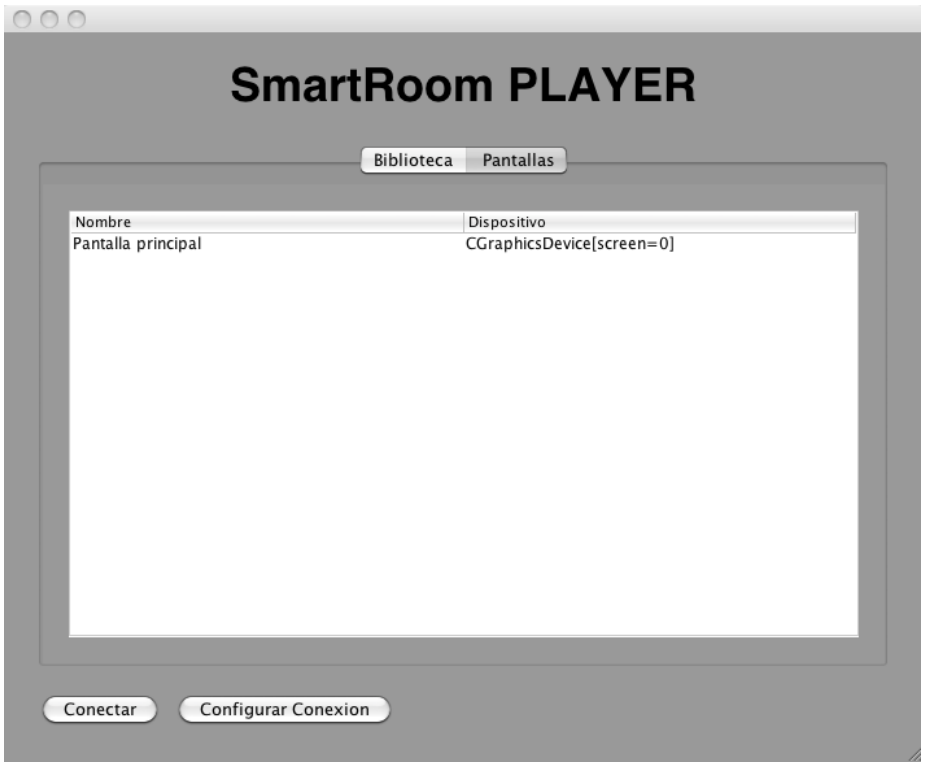


Ilustración 43: Dispositivos conectados al Medio

Una vez conectados todos los dispositivos de salida será necesario poblar la biblioteca con el contenido audiovisual que se desea que esté disponible. Además en la biblioteca estarán visibles los applets de processing que se encuentren instalados en el medio.

La idea general de SmartRoomPlayer es que existe una pantalla de reproducción por cada sistema de salida conectado, pudiéndose ejecutar dentro de cada una el contenido audiovisual que se desee.

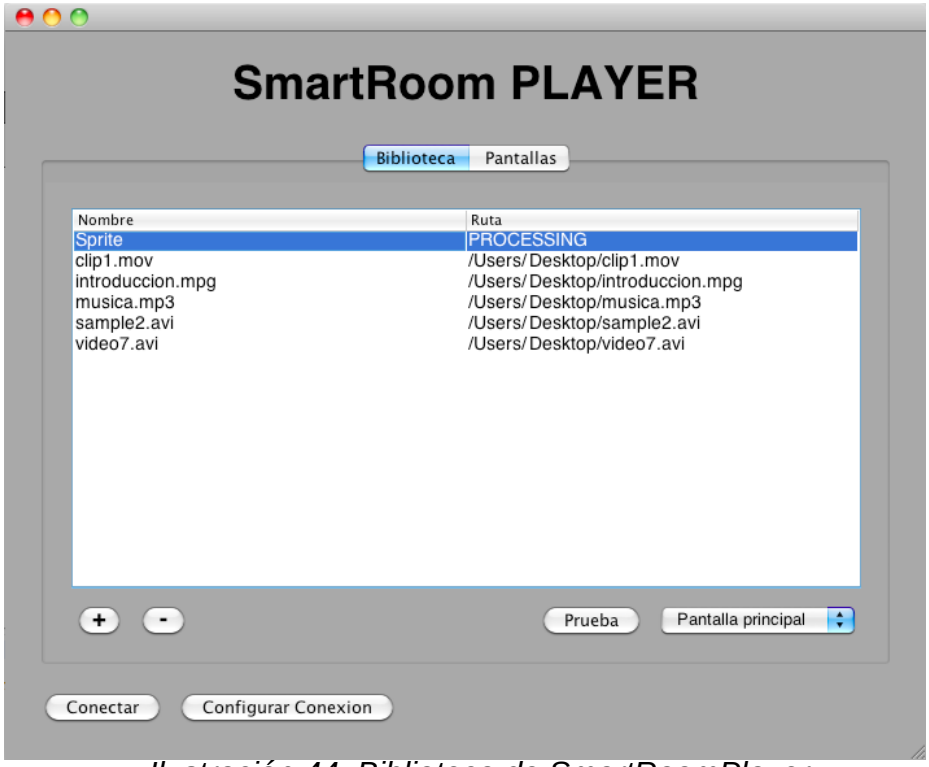


Ilustración 44: Biblioteca de SmartRoomPlayer

Como puede verse en la imagen superior es posible probar los contenidos en los dispositivos que se encuentren conectados. Para ello es necesario seleccionar un archivo de la biblioteca, seleccionar el dispositivo de salida y pulsar el botón de prueba. Es aconsejable realizar algunas de estas pruebas para comprobar que todo se reproducirá de la manera esperada cuando se ejecuten las órdenes de las reglas más adelante.

Al igual que los demás elementos del sistema, SmartRoomPlayer necesita estar conectado a una central para poder ser utilizado. Para ello, al igual que con la aplicación de sensores, es posible configurar ciertos parámetros como la ip y puerto de la central, así también como su propio nombre dentro del sistema.

5.4. SmartRoomCenter

Esta aplicación puede considerarse como la más importante del sistema. Desde ella podrá controlarse el estado de la conexión de los demás dispositivos ya que se conectarán a ésta por red. Además aquí se llevará a cabo la gestión de reglas.

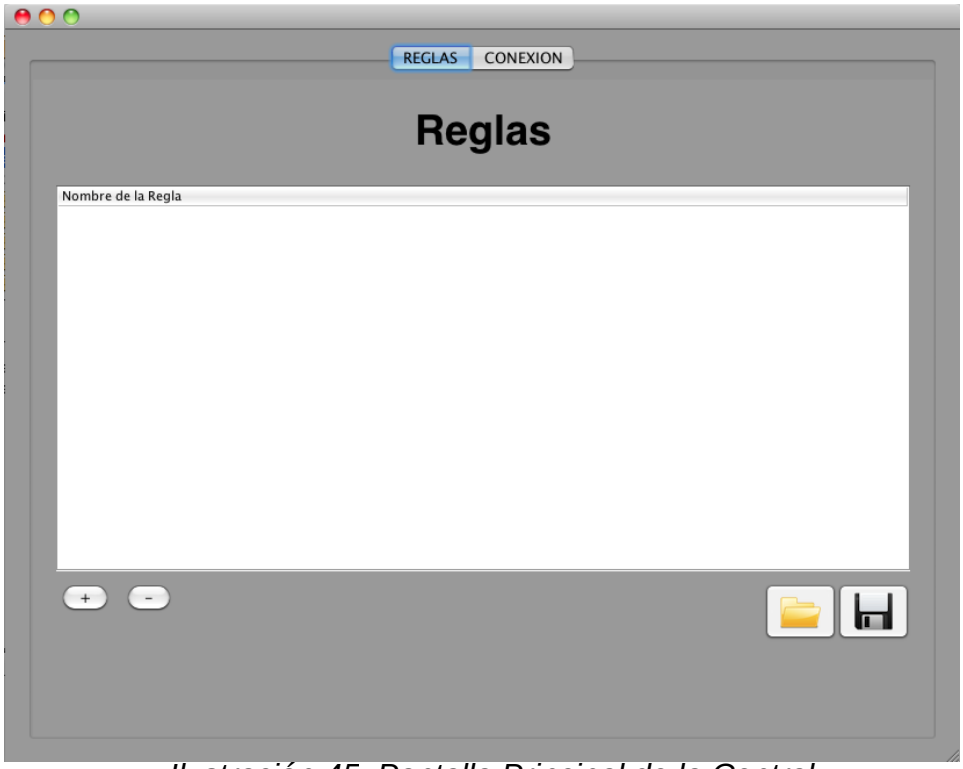


Ilustración 45: Pantalla Principal de la Central

Se disponen de dos pestañas: Reglas y Conexión. Para poder ver el estado de la conexión de los demás dispositivos es necesario pulsar sobre la pestaña correspondiente. Ahí podrá observarse una lista con los dispositivos sensores y actuadores conectados a la central. Al pulsar sobre cada uno de ellos podrá verse información asociada a cada uno. Es importante comprobar tanto los nombres de cada dispositivo conectado como las acciones, archivos disponibles.

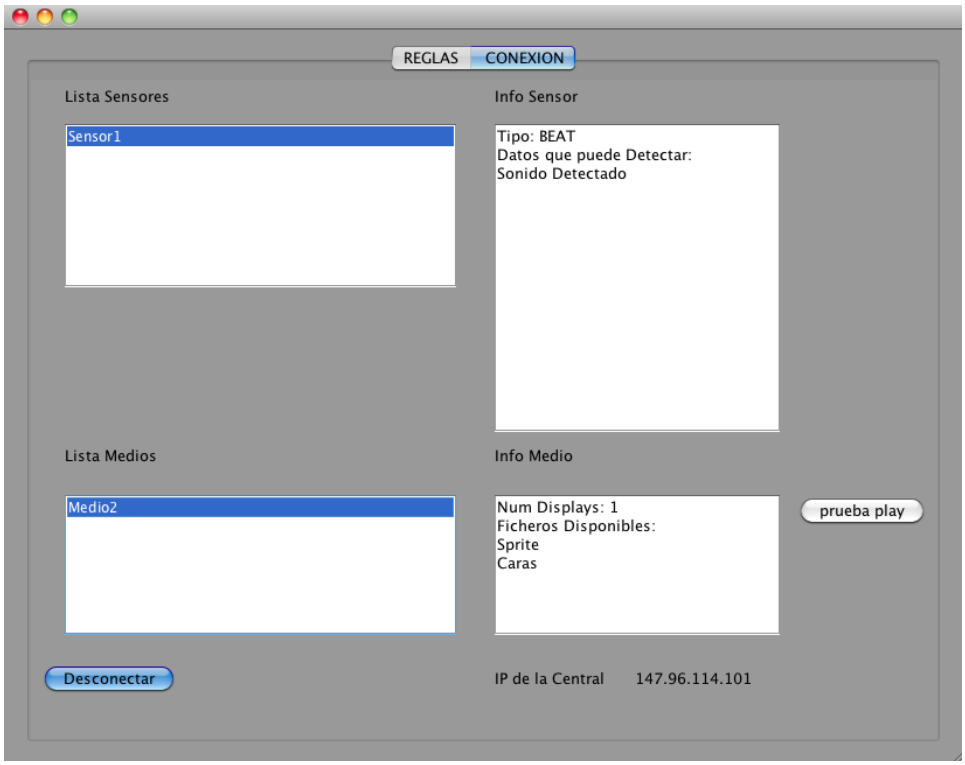


Ilustración 46: Pantalla de conexión de la Central

Para comenzar se deberá pulsar el botón de conectar para que los demás dispositivos puedan registrarse. Puede verse en esta pestaña la ip asociada a la central. Esto es útil porque el resto de componentes del sistema deberán conectarse a esta ip.

En el Panel de Reglas se disponen de cuarto botones. Dos de ellos asociados a la creación o eliminación de reglas y otros dos que permiten salvar o cargar reglas, condiciones y acciones.

El botón para Salvar tiene como dibujo un disco de 3,5. Guardará el estado de la aplicación, tanto las reglas creadas como las acciones y condiciones. Sólo será necesario dar un nombre y una ruta. Se guardarán tres archivos, uno de ellos con extensión '____.smr' y otros tres sin extensión.

El botón de Cargar tiene como dibujo una carpeta amarilla. Elegiremos el archivo '____.smr' guardado previamente. El resto de archivos que se crean al guardar deberán estar en la misma carpeta o no funcionará.

Los botones '+' y '-' situados debajo de la tabla de reglas permiten crear o eliminar reglas respectivamente.

El botón de crear hará aparecer una nueva ventana con dos tablas: Acciones y Condiciones.

Ilustración 47: Pantalla de Condiciones y Acciones

Puede observarse que sigue la misma estructura que la ventana anterior.

Nuevamente se dispondrán de botones para crear o eliminar, en este caso condiciones o acciones.

Para crear una regla habrá que seleccionar una condición y una o varias acciones, dar un nombre a la regla donde pone ' Nombre Regla ' y pulsar el botón de crear. Si alguno de estos pasos no se cumple la regla no se creará y saldrá un mensaje de error explicando cuál fue el error.

Si no se quiere crear ninguna regla, puede volverse a panel de reglas pulsando el botón de CANCELAR.

Si se pulsa sobre el signo ' + ' de añadir en la tabla de condiciones aparecerá la siguiente ventana:

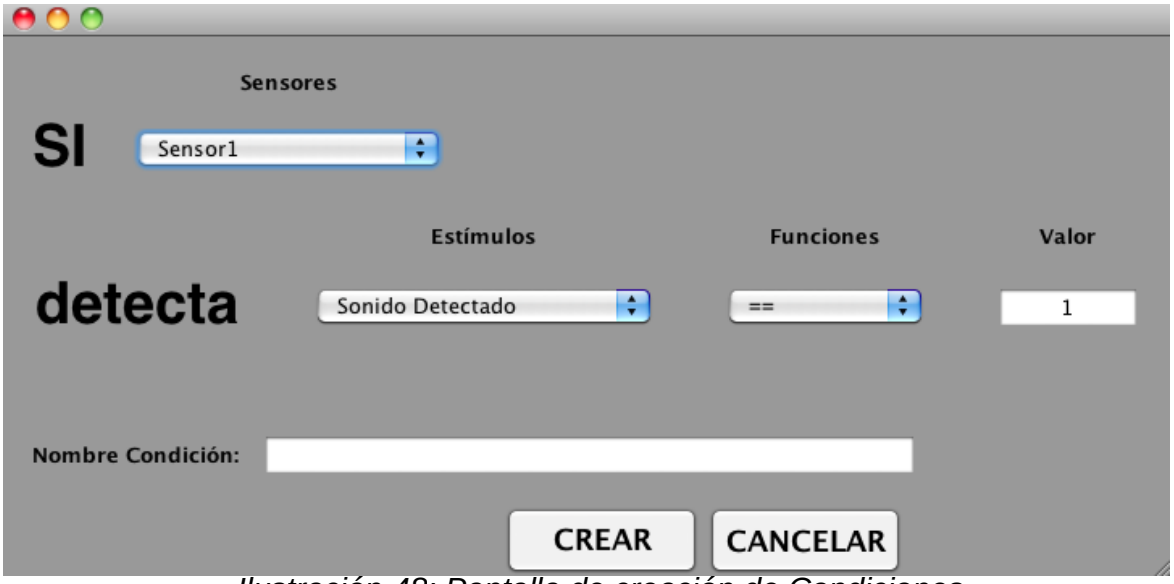


Ilustración 48: Pantalla de creación de Condiciones

Para crear una condición será necesario elegir el sensor, el tipo de estímulo que se quiere detectar (depende del tipo de sensor elegido), se selecciona la función y el valor. Por ejemplo "**Si** Sensor3 **DETECTA** caras detectadas == 3".

Después se introducirá un nombre. Para terminar pulsar CREAR. Si por alguna razón no se quisiera crear una condición, al pulsar el botón de cancelar se volverá a la pantalla anterior.

En la ventana anterior si se pulsa sobre el signo ' + ' de añadir en la tabla de acciones aparecerá la siguiente ventana:

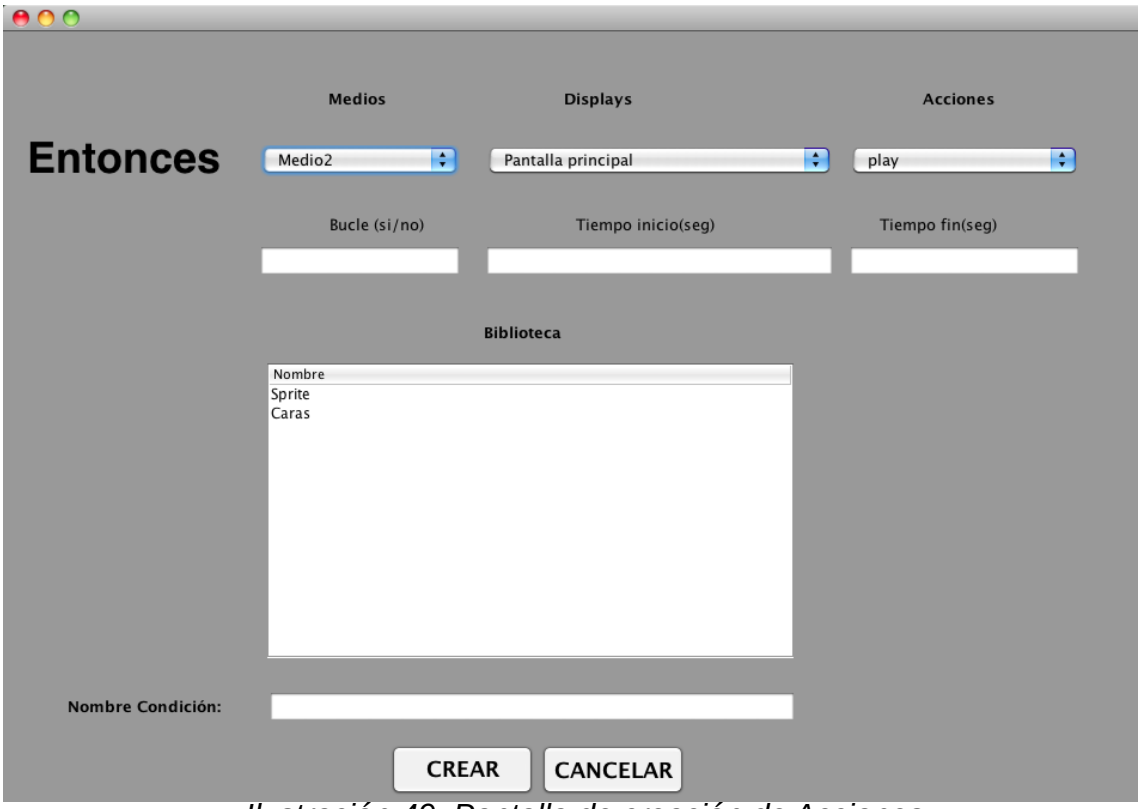


Ilustración 49: Pantalla de creación de Acciones

Para crear una acción lo primero que debe hacerse es elegir el medio donde ejecutar dicha acción y luego el Display si el medio elegido tuviera varios.

Después se selecciona la acción que se quiere ejecutar en dicho display del medio. Dependiendo de la acción elegida, aparecerán unas opciones u otras.

Para terminar, será necesario asignar un nombre a la acción.

Como en el caso anterior, si no se quisiera crear una acción, pulsando cancelar se volverá a la ventana anterior.

CAPÍTULO 6

6.Conclusiones

6.1.Discusiones sobre el trabajo

A continuación se enumerarán las conclusiones obtenidas a lo largo del trabajo desarrollado.

Con respecto a los sensores:

- La gran variedad de sensores hace complicada la elección del tipo adecuado. Esta elección se ha de basar en aquello que se pretende detectar dentro una habitación. Cuanto mejores sean los sensores, mejor será el resultado y menos se tendrán que usar.

- Los mejores sensores son aquellos cuyo uso no está limitado a un ámbito muy especifico, para una instalación artística se deben utilizar sensores polivalentes. Cámaras y micrófonos son una elección acertada para detectar movimiento y ruido, además poder ser empleados para muchas otras cosas. Abstraer el tipo de sensor dentro del proceso de razonamiento basado en reglas permite incluir nuevos tipos de sensores mientras se puedan programar cumpliendo la interfaz de comunicación.

Con respecto a los medios:

- La posibilidad de gestionar todos los dispositivos gráficos (pantallas y proyectores) conectados a un ordenador. Así como la capacidad para reproducir una gran variedad de tipos de archivos multimedia, permite crear instalaciones artísticas muy variadas. Cumpliendo el objetivo de no limitar lo que el usuario final puede llevar a cabo con la aplicación.

- También se ha tenido en cuenta la posibilidad de aumentar las funcionalidades de los medios para que ampliarlas no suponga un gran problema.

Con respecto a las tecnologías de control del sistema:

- Desarrollar una aplicación distribuida permite utilizar más de una máquina para gestionar la instalación artística. Por lo que no existe limitación (a los recursos de una sola máquina) en el número de sensores y medios que es posible manejar. Esto por ejemplo permitiría gestionar una instalación compuesta de distintas habitaciones.

- El uso de CORBA para crear la aplicación distribuida no limita el lenguaje en sean programados medios y sensores. Por tanto, se abre la posibilidad en un futuro a distintas versiones de SmartRoomSensor, para la programación en otro lenguaje distinto a Java de otros tipos de Sensores.

6.2.Alternativas y trabajo futuro

A continuación se enumeran una serie de cuestiones a tener en cuenta como trabajo futuro o como posibles alternativas de implementación:

- **Implementación de nuevos sensores.** Dado el diseño aplicado al sistema es posible añadir nuevos sensores que permitan detectar otros estímulos. Para ello será necesario tener en cuenta los interfaces que se deben implementar.

- **Implementación de nuevos medios.** Agregar nuevos medios que permitan otro tipo de acciones diferentes a las audiovisuales (como encender, apagar o graduar luces, accionar otros mecanismos...). Además pueden añadirse acciones al medio ya implementado.

- **Flags.** Los flags son unos señalizadores, con valores de verdadero o falso. Las identificaremos mediante colores. Se comportan como marcas dinámicas que se podrán añadir a las reglas. De esta forma podemos decir que una regla no se dispare hasta que el flags ROJO y el flag Azul se hallan puesto a verdadero.

- **Transmitir caras detectadas.** Poder transmitir las caras detectadas por los sensores permitiría nuevas posibilidades artísticas como por ejemplo la integración en videos pregrabados.

- **Exposiciones en Instalaciones Artísticas.** Como se ha comentado con anterioridad se pretende que se haga uso de esta herramienta para futuros trabajos artísticos reales.

7. Bibliografía

[1] Colectivo MASDELOMISMO Página Principal: http://www.masdelomismo.net/

[2] Libro: "Instalaciones", Autor: Josu Larrañaga, Editorial: Nerea Año:2001

[3] Página principal de OSC : http://opensoundcontrol.org/

[4] Página de OSC para processing : http://www.sojamo.de/libraries/oscP5/

[5] Introducción a CORBA: http://www.lirmm.fr/~nebut/pub/upload/Enseignement/intro_corba.pdf

[6] Historia CORBA : http://www.omg.org/gettingstarted/history_of_corba.htm

[7] Especificaciones CORBA: http://www.omg.org/technology/documents/spec_catalog.htm

[8] Componentes CORBA: http://www.monografias.com/trabajos16/componentes/componentes.shtml

[9] Página principal de CORBA: http://www.corba.org/

[10] Página del Object Management Group: http://www.omg.org/

[11] Librerías de CORBA adaptadas a JAVA: http://java.sun.com/developer/onlineTraining/corba/corba.html

[12] Imagen cogida de Wikipedia CORBA: http://en.wikipedia.org/wiki/File:Orb.svg

[13] Imagen cogida de la página principal de ICE: http://www.zeroc.com/doc/Ice-3.4.0/manual/Overview.3.2.html

[14] Página principal ICE: http://www.zeroc.com/doc/index.html

[15] ICE vs. CORBA: http://www.zeroc.com/iceVsCorba.html

[16] Definición del Algoritmo RETE: http://es.wikipedia.org/wiki/Algoritmo_Rete

[17] Página principal del Motor de Reglas Jess: http://www.sandia.gov/

[18] Página principal del Motor de Reglas Drools: www.jboss.org/drools

[19] Página principal de los sensores Phidgets:
http://www.phidgets.com/index.php

[20] Ejemplo de Phidget: Sensor Sharp de distancia
http://www.phidgets.com/products.php?category=2&product_id=3522

[21] Imagen cogida de la Página principal de Phidgets :
http://www.phidgets.com/images/3520_0_Functional_Web.jpg

[22] Página Principal OPENCV: http://opencv.willowgarage.com/wiki/

[23] Página de OPENCV de la Wikipedia: http://es.wikipedia.org/wiki/OpenCV

[24] Página de instalación para OPENCV:
http://ubaa.net/shared/processing/opencv/

[25] Página Principal de la librería JAVA ARToolkit:
http://www.hitl.washington.edu/artoolkit/

[26] Ejemplos de uso de ARToolKit : http://www.artoolworks.com

[27] Página Principal de la librería
Minim: http://code.compartmental.net/tools/minim/

[28] Página de desacarga de QuickTime:
http://www.apple.com/quicktime/download/

[29] Página Principal de Adobe Photoshop: https://www.photoshop.com/

[30] “Desing and the elastic mind”, Paola Antonelli, Moma 2008

[31] Página principal del Proyecto Processing: http://processing.org/

[32] Patrón Software de Modelo - Vista- Controlador
para java : http://java.sun.com/blueprints/patterns/MVC.html

[33] Imagen de patrón Modelo - Vista - Controlador : http://boliviaonrails.com/wp-content/uploads/2008/07/mvc.gif

8.ANEXO

8.1.Requisitos del Sistema

Esta aplicación necesita tener instalada la máquina virtual de Java (JRE 1.5) disponible en: http://www.java.com/es/download/.

Tras las pruebas en los distintos sistemas operativos, podemos afirmar que las tres aplicaciones: SmartRoom Sensor, SmartRoom Player y SmartRoom Center son compatibles para Linux Ubuntu 10, Mac OSX, y Windows XP.

Para todos los sistemas operativos es necesario descargarse la librería OpenCV y seguir las instrucciones de la siguiente página [24]. Dependiendo del sistema operativo, habrá que hacer un tipo de instalación u otra por tanto es necesario instalar bien esta librería.

Para Windows además es necesario tener instalado QuickTime [28].

Para que los distintos programas se comuniquen en red, es necesario que las distintas máquinas estén conectadas dentro de una misma red local y con el puerto 12345 abierto en todas las máquinas.

Dado el carácter de la aplicación, es necesario disponer de los periféricos oportunos (cámaras, micrófonos, proyectores) correctamente instalados en aquellas máquinas en las que vayan a ser utilizados por la aplicación.

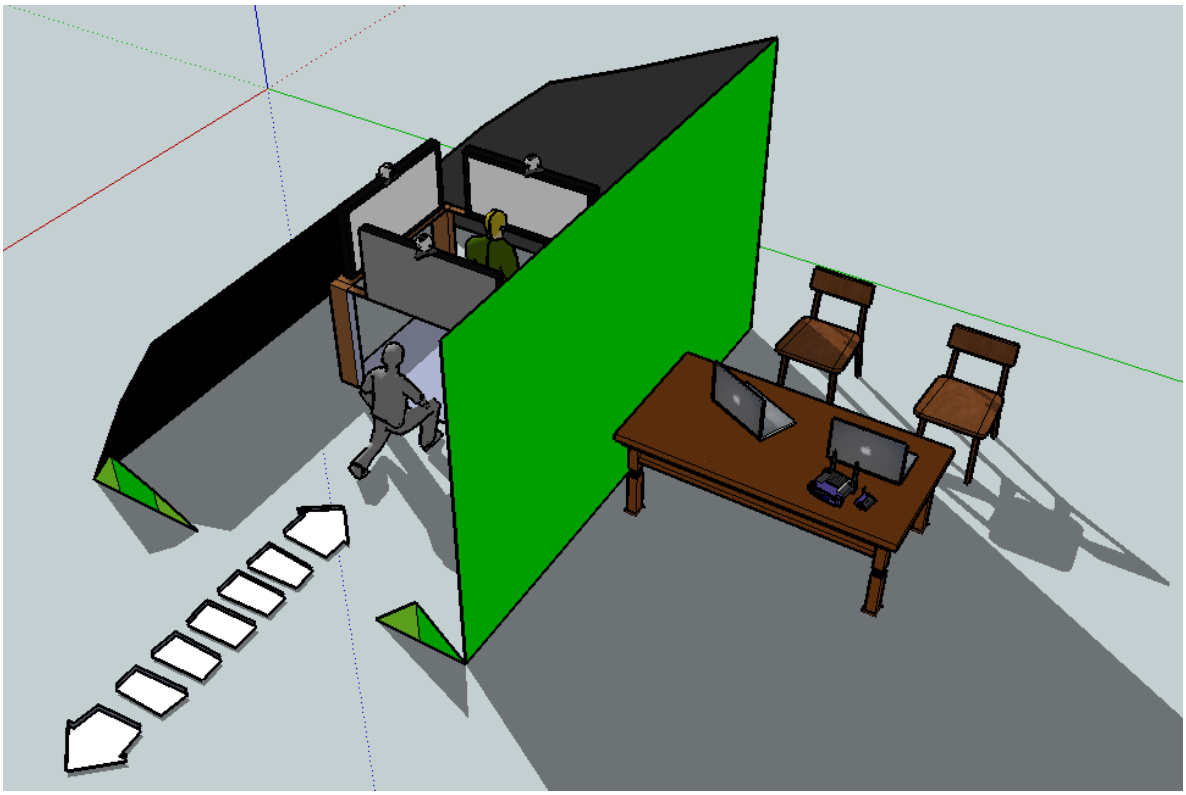
Los requisitos de hardware varían en función de el número de periféricos que se desee conectar. Es aconsejable una tarjeta gráfica con al menos 128MB de memoria de vídeo. También con 512 MB de Memoria RAM.

8.2.Manual de Usuario

Introducción

Se pretende montar la instalación artística de la figura y utilizar *SmartRoom* para su control. En esta instalación se controlan cuatro cámaras (que se usarán como sensores detectores de caras) y cuatro pantallas (medios) en los que reproducir archivos de vídeo.

Para ello se definirán una serie de reglas en *SmartRoom Center* para definir el comportamiento de la instalación.



Illustración 50: Ejemplo de instalación

Requisitos técnicos de la instalación

- 4 pantallas TFT.
- 4 cámaras web.
- 1 router.
- 3 ordenadores portátiles o de sobremesa.
- Cable de red.
- Cables usb, alargadores y hubs en función de las necesidades y el espacio disponible.

Todos los ordenadores estarán conectados al router en una red local por medio de cable. Uno de los ordenadores estará dedicado exclusivamente a la central.

El resto de los ordenadores tendrán conectados las pantallas y las cámaras. Se harán necesarios cables lo suficientemente largos para conectar los periféricos (cámaras y pantallas) a los ordenadores en función de la distancia de estos con la instalación. Siempre teniendo en cuenta que la instalación estará separada del centro de control.

Un ejemplo de disposición con tres ordenadores sería: el primero ejecutando *SmartRoom Central* y cada uno de los otros con dos aplicaciones *SmartRoom Sensor* y dos aplicaciones *SmartRoom Medio* cada uno.

Sin embargo la capacidad para gestionar sensores y medios varía en función de las características de los equipos disponibles. Ya que cuanto mejores sean los ordenadores más dispositivos podrán controlar, siempre buscando el buen funcionamiento de la instalación.

Configuración de la instalación artística

Una vez conectados todos los periféricos a los ordenadores y dispuestos en red. Se configurará la aplicación SmartRoom.

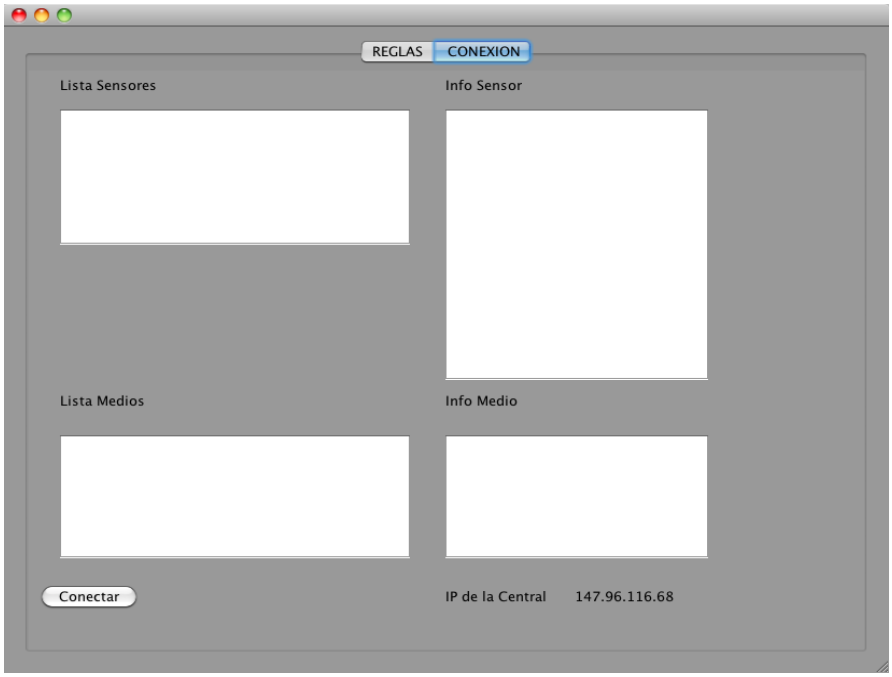
En el ordenador escogido para ejercer de Central se ejecutará el programa **SmartRoom Center.**

Para cada pantalla que maneje un ordenador se ejecutará una aplicación **SmartRoom Player**. Igual que lo anterior para cada ordenador que maneje una cámara se ejecutará una aplicación **SmartRoom Sensor**.

En total para cuatro pantallas y cuatro cámaras habrá cuatro aplicaciones *SmartRoom Sensor* y cuatro *SmartRoom Player*.

Conectar la Central

Una vez ejecutados los distintos programas en los equipos correspondientes el primer paso es **conectar la Central**. Para ello dentro de la aplicación *SmartRoom Center,* ha de seleccionarse la pestaña *Conexión* y pulsar el botón "*Conectar*". A partir de este momento tanto medios y sensores se podrán conectar a ella. La dirección IP de la central se muestra a la derecha del botón de conexión.



Illustración 51: Pestaña de conexión de SmartRoomCenter

Dicha dirección compuesta por cuatro números separados puntos (ej. 192.168.0.10) será aquella a la que tendrán que conectarse las demás aplicaciones(este paso se detalla en el apartado siguiente).

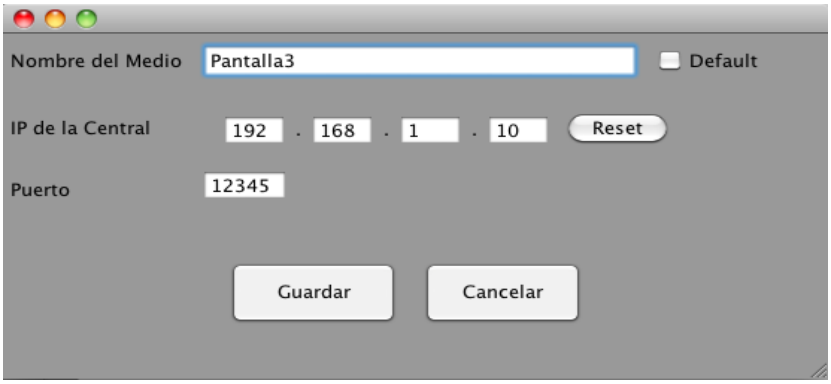
Conectar medios y sensores

Para conectar las aplicaciones *SmartRoom Sensor* y *SmartRoom Player* a la Central, primero hay que **configurar la conexión**. Ambas por defecto están configuradas para conectar con la central en modo local, es decir, suponiendo que la central se está ejecutando en el mismo equipo que ellas.

En el caso de que la central se encuentre en otra de las máquinas de la red, se deberá configurar **la IP en la que se encuentra la Central**. Tanto en

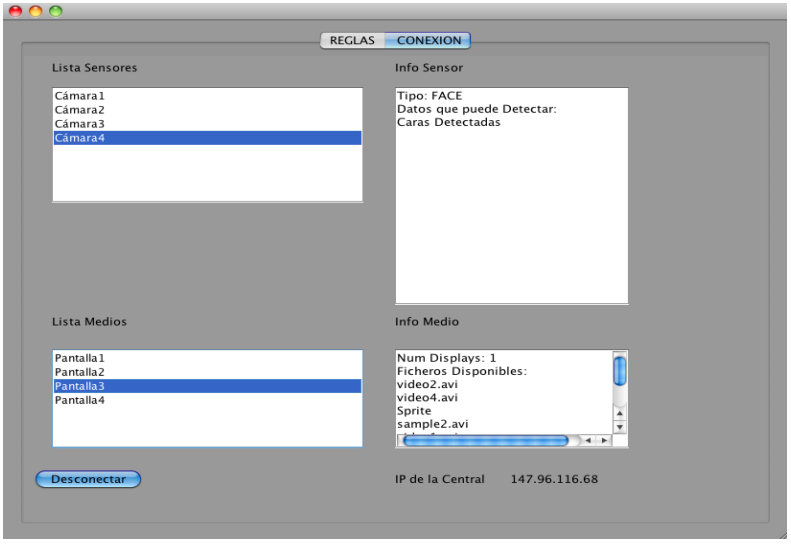
SmartRoom Sensor como en *SmartRoom Medio*, pulsando el botón "*Configurar Conexión*" `se abre una ventana de configuración.

Además de poder elegir la IP en la que se encuentra la Central, también se podrá elegir el nombre del Medio o Sensor, si la casilla "*Default*" se encuentra marcada será la Central la que se encargará de asignar el nombre, se recomienda elegir un nombre representativo.



Illustración 52: Ventana de conexión del medio

Una vez conectado un medio o un sensor aparecerá en la pestaña "Conexión"de *SmartRoom Center* con la información disponible.



Illustración 53: SmartRoom con sensores y medios conectados

Crear los Sensores

En cada aplicación SmartRoom Sensor se crea un sensor de la instalación. Ya sea antes de conectar, como después, se pueden activar el sensor de la aplicación.

Una vez creado el sensor y conectado *SmartRoom Sensor* a la Central, este permanecerá activo trasmitiendo la información de sus capturas a la aplicación *SmartRoom Center*.

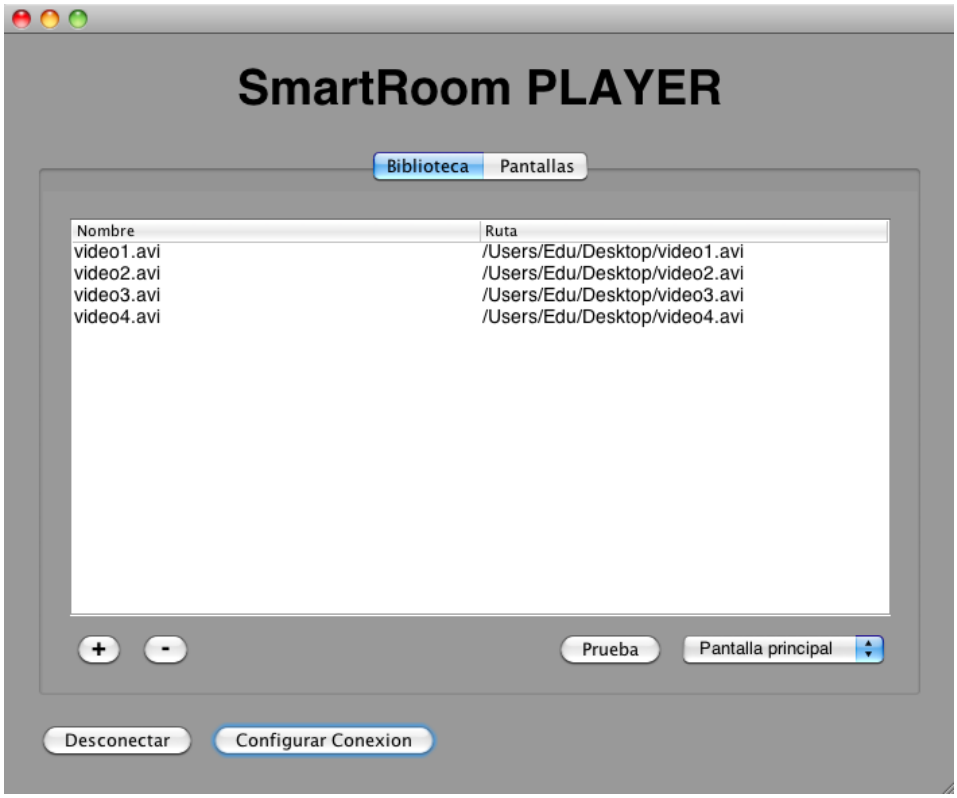
Illustración 54: Botones de configuración del sensor

Añadir archivos a la Biblioteca de SmartRoomPlayer

Para poder reproducir archivos de audio o vídeo estos tienen que formar parte de la biblioteca de *SmartRoom Player*.

Para añadir un archivo a la biblioteca solamente hace falta pulsar el botón “+” y seleccionar la ruta del archivo. Una vez cargado el archivo estará disponible para ser reproducido a orden de la Central.

Crear Reglas. Salvar/Cargar.



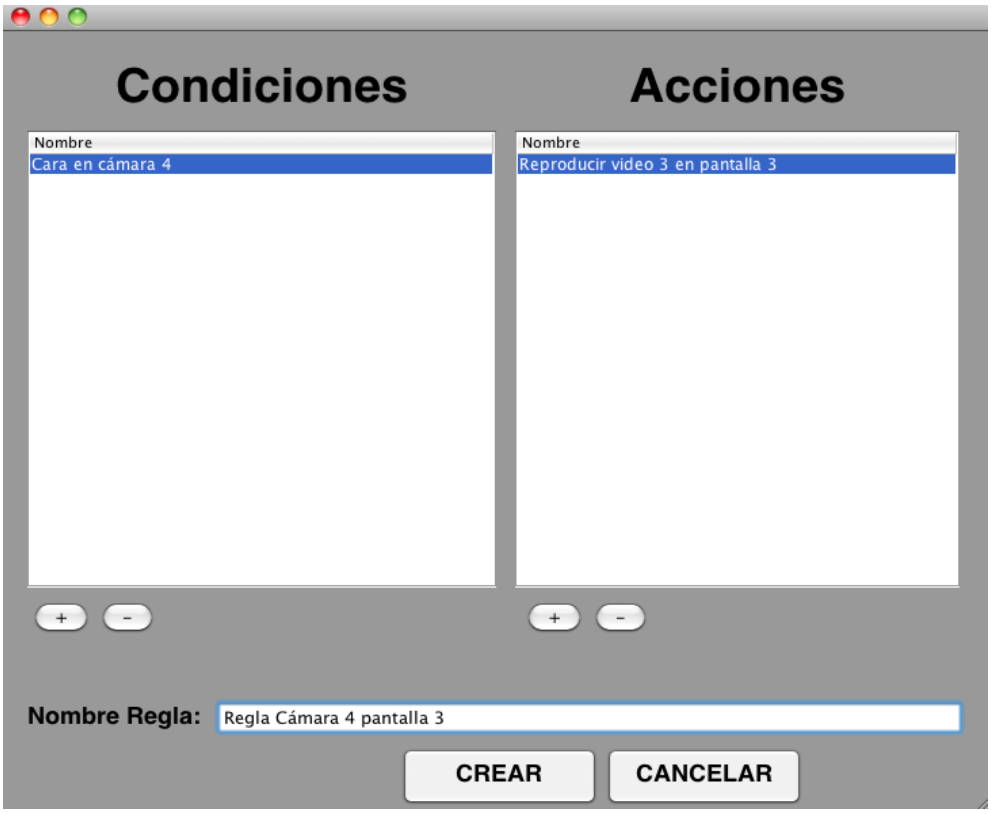
Illustración 55: Biblioteca

Una vez conectados y configurados todos los medios y sensores llega el momento de **crear las reglas de la instalación**. Dentro de la pestaña de reglas se podrán definir las reglas de la instalación. La explicación detallada de la manera de crear reglas se encuentra en el **apartado 5.4** de este documento.

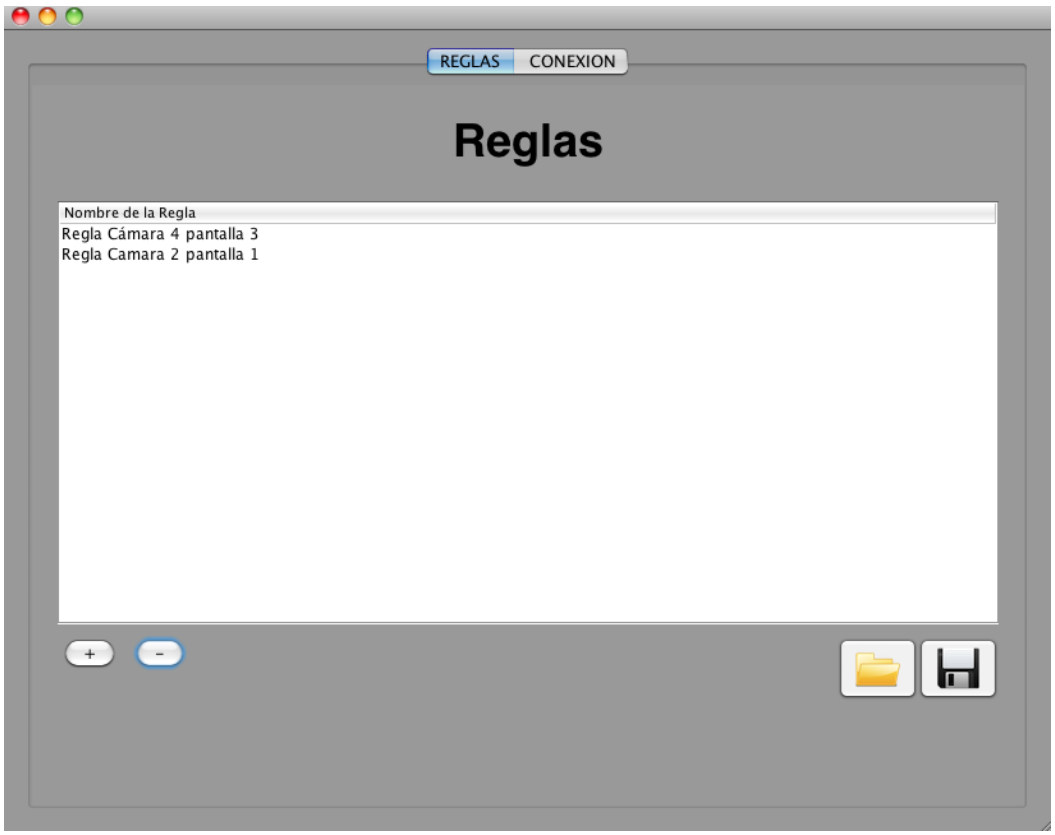
Cabe resaltar que dichas reglas solo pueden ser creadas sobre Medios y Sensores conectados en ese momento. Si se desea **guardar** el archivo de reglas de la instalación se debe pulsar el botón “Guardar” (icono del disquete) y elegir el nombre del archivo y la ruta en la que guardará.

Para **cargar** el archivo de reglas, pulsar el botón “Cargar” (icono de la carpeta) y seleccionar el fichero.

Si las reglas han sido cargadas a partir de un archivo de configuración hay que tener en cuenta que los medios y sensores conectados tienen que respetar la configuración anterior. Es decir, se ha de configurar Medios y Sensores apropiadamente (según las indicaciones anteriores) para que las reglas puedan activarse.



Illustración 56: Panel de creación de Reglas



Illustración 57: Reglas activas

Desconectar Medios y Sensores

Es posible desconectar Medios y Sensores pulsando en el botón "Desconectar" en sus respectivas aplicaciones (*SmartRoom Player* y *SmartRoom Sensor*). En todo momento se puede consultar en la pestaña "*Conexión*" de *SmartRoom Center* cuales son los medios y sensores conectados con la Central.

Desconectar Central

Si se desea desconectar la Central, basta con pulsar el botón "*Desconectar*" que desconectará de la Central a todos los medios y sensores conectados. Si se vuelve a conectar la Central los medios y sensores deberán volver a conectarse para poder ser utilizados.

8.3.Librerías

[Lib.1] XStream: Librería para serializar objetos en formato XML
http://xstream.codehaus.org/

[Lib.2] Java Media Framework (JMF):
http://java.sun.com/javase/technologies/desktop/media/jmf/

[Lib.3] FOBS: http://fobs.sourceforge.net/

8.4.Contenidos Multimedia

Ejemplos de arte en actionScript:

[Mul1] "Flash MX games: ActionScript for artists" por Nik Leve
[Mul2] Obras de Patrick Gunderson http://pat.theorigin.net/tag/adobe-flash/

Ejemplos de Arte en Processing :

[Mul3] Visualizando Imperios, de Pedro M. Cruz
http://mondeguinho.com/master/visual-experiments/visualizing-empires

[Mul4] Bees, de Julia Stanat
http://processing.org/exhibition/works/bees/index_link.html

[Mul5] Proyecto de dos españoles arquitectos : http://www.intheair.es/

[Mul6] Empresa dedicada a animaciones con processing :
http://www.universaleverything.com/

Otros Ejemplos de Instalaciones:

[Mul7] Pantalla Philips Daylight (http://www.youtube.com/watch?v=AIaqFgBppvs).

[Mul8] Instalacion Sustainable danceFloor
http://www.studioroosegaarde.net/work_html.php?id=26&picture_id=1582

[Mul9] Marti Guixé en experimetadesign
http://www.designboom.com/weblog/cat/8/view/3966/marti-guixe-at-experimentadesign-amsterdam-2008.html

[Mul10] Pablo Valbuena: “Entramado”, Plaza de las Letras, Madrid.

8.5.IDL de la aplicación

A continuación se muestra el archivo IDL utilizado para la comunicación.

[IDL1]

```
module remote{
  exception DisplayDesconocidoException{};
  exception FileNotFoundException{};
  exception idNombreNoDisponibleException{};
  typedef sequence<string> listaCadena;
  typedef sequence<long> listaEnteros;

  interface IMensaje{
    attribute string tipo;
    attribute string descripcion;
    attribute string valor;
  };

  interface IInfoSensor{
    attribute string tipoSensor;
    attribute listaCadena listaDescripcion;
    attribute listaCadena listaTipo;
  };

  interface IInfoMedio{
    long getNumDisplay();
    listaCadena getDisplays();
    listaCadena getFicherosDisponibles();
    listaCadena getAcciones();
    long getAridades(in long accion);
    listaCadena getDescripciones(in long accion);
  };
```

```
    interface IMedio{
      void play(in long index, in string file,in string bucle, in string tiempoIni, in string tiempoF) raises
(DisplayDesconocidoException, FileNotFoundException);
      void stop(in long index);
      void subirVolumen(in long index);
      void bajarVolumen(in long index);
      IInfoMedio getInfoMedio();
      void setNombre(in string nombre);
      string getNombre();
      void desconexion();
    };

    interface ISrvSensor{
      void setNombre(in string nombre);
      string getNombre();
      void desconexion();
      IInfoSensor getInfoSensor();
    };

    interface ICentralRemota{
      void registraMedio(in IMedio medio, in boolean asignarNombre) raises
(idNombreNoDisponibleException);
      void registraSensorSrv(in ISrvSensor srvSensor, in boolean asignarNombre) raises
(idNombreNoDisponibleException);
      void notificaCambioSensor(in string idSrvSensor, in string tipoSensor, in IMensaje msj);
      void desregistrarSensorSrv(in string idSrvSensor);
      void desregistrarMedio(in string idMedio);
      void setInfoSensor(in string name, in IInfoSensor info);
    };
  };
```

Índice de ilustraciones