Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

CONTROLADOR DE SUELO INTELIGENTE

Alumno: Araceli Cañadas Ruiz

Tutor: Prof. D. Ángel Luis García Fernández
Cotutor: D. Daniel Zafra Romero
Dpto: Informática

Septiembre, 2017

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

D. Ángel Luis García Fernández y D. Daniel Zafra Romero, tutor y cotutor respectivamente del Proyecto Fin de Carrera titulado: Controlador de suelo inteligente, que presenta Araceli Cañadas Ruiz, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, septiembre de 2017

La alumna: Los tutores:

Araceli Cañadas Ruiz D. Ángel Luis García Fernández D. Daniel Zafra Romero

ÍNDICE

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

1.1. Motivación

Las funciones convencionales del suelo de las habitaciones han empezado a extenderse más allá del mero soporte mecánico (y estético) para abarcar otros usos como la climatización o la reducción de ruido. Sin embargo, teniendo en cuenta el hecho de que durante el día estamos en contacto directo con el suelo, nos podemos preguntar si es posible explotar esta estrecha relación con funciones más avanzadas como detectar presencias y caídas.

Según la OMS, cada año se producen 37,3 millones de caídas cuya gravedad requiere atención médica [1], lo cual nos lleva a la conclusión de que sería de utilidad para una persona que haya sufrido una caída y se encuentre sola, que el propio suelo detectara su caída y fuese capaz de interaccionar con ella para determinar si se encuentra bien y la caída no ha sido grave o, en caso de que la persona perdiese la consciencia, sea incluso capaz de solicitar atención médica. En la Figura 1 se muestra el porcentaje de caídas que se producen en interiores en relación a la posición previa a la caída y en función de la edad.

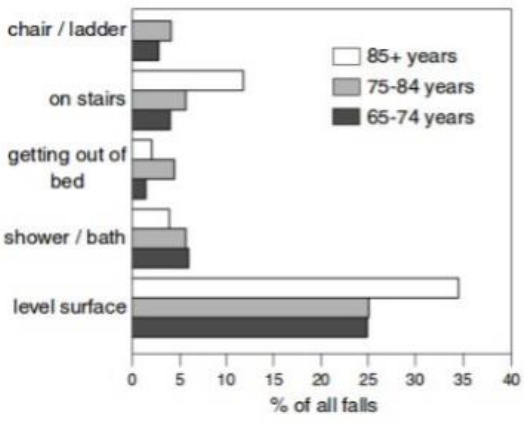


Figura 1. Porcentaje de caídas en interiores respecto a posición en función de la edad [1]

Al abordar este problema no aparece una única solución. De hecho, en un artículo sobre las caídas en personas de edad avanzada de la Revista española de geriatría y gerontología [2] se establecen dos grandes grupos según el emplazamiento de los sensores: sensores ambientales, que monitorizan el entorno del paciente, y sensores portátiles, que supervisan la actividad del paciente, detectando variaciones de movimiento y posición. Todas estas soluciones tienen sus limitaciones.

El presente trabajo tratará de establecer una solución mediante la implementación de un sistema basado en la monitorización del entorno utilizando un suelo inteligente de la marca SensFloor desarrollado por la empresa Future Shape. Para el desarrollo se cuenta con dos instalaciones de SensFloor del CEATIC (Centro de Estudios Avanzados en Tecnologías de la Información y de la Comunicación) de la Universidad de Jaén. Una de ellas es un sistema de pruebas y la otra un sistema instalado en el suelo de un laboratorio inteligente denominado *SmartLab* que se encuentra en la dependencia 109 del edificio C6 de la Universidad de Jaén.

Como ventajas principales frente al resto de soluciones, este tipo de sensores, una vez instalados, proporcionan una monitorización constante, utilizan la corriente eléctrica por lo que no presentan problemas de autonomía, permiten detectar distintos tipos de caídas y no suponen un cambio brusco en el entorno por lo que no suelen provocar el rechazo del usuario. Como desventajas, están su instalación (es necesario colocar los sensores bajo el pavimento), que no detectan falsos positivos (por ej. sentarse en el suelo de forma brusca) y su coste elevado.

La herramienta SensFloor convierte el suelo de una habitación en un gran sensor capaz de detectar la ubicación, el número y el movimiento de las personas que se sitúan sobre él, logrando así permanecer “invisible” para el usuario y sin interferir en el aspecto de su entorno.

El software del que viene acompañado se denomina FSSensFloorGUI. Está constituido por los archivos ‘FSSensFloorGUI.exe’, aplicación de escritorio desarrollada en Matlab y 'FSSensFloorGUIContext.dat', que proporciona datos de contexto. [3] FSSensFloorGUI está programado en Matlab. Matlab es un lenguaje precompilado que requiere un entorno de ejecución (MCR – Matlab Compiler Runtime) para ejecutarse.

Pese a que el software cuenta con muchas utilidades, su falta de transparencia supone una gran limitación para su uso en un laboratorio. Es una aplicación cerrada y no es posible realizar modificaciones en la forma de visualizar los datos emitidos por el suelo (más allá de los que la interfaz tiene implementados) ni utilizarlos para mandarlos a otra aplicación que se desarrolle distinta a la original y con otras utilidades.

Esto lleva a la necesidad de realizar un estudio de SensFloor para conseguir crear una nueva interfaz, dando mayor accesibilidad a los datos proporcionados por el mismo.

1.2. Objetivo

El objetivo principal del trabajo será analizar los datos que emite SensFloor para procesarlos estableciendo cuáles serán de utilidad para establecer la posición de una persona y realizar una interfaz gráfica web donde hacer un seguimiento.

Además, la aplicación podrá detectar la caída de una persona para realizar una acción pertinente en el caso de se produzca este evento.

2. PLANIFICACIÓN

2.1. Actividades

Para conseguir una visión general de las actividades del proyecto, hemos creado el diagrama de la Figura 2 con la aplicación web XMind [4]

Figura 2. Actividades del proyecto

2.2. Estimación de tiempo del proyecto

El desarrollo del trabajo lo realizaremos basándonos en metodologías ágiles, dividiendo las tareas en *sprints* (iteraciones).

En la Tabla 1 se muestra una estimación de las tareas que después se irán desgranando y desplegando en los sprints y el tiempo en horas que se dedicará a cada una de ellas:

ID	Nombre	Horas
INV-SUE	Investigación del funcionamiento del suelo	24
INV-PROT	Investigación sobre los protocolos de comunicación	16
DES-FIL	Estudio y filtrado de datos recogidos por el suelo	72
DES-ENV	Almacén y envío de datos filtrados	24
DES-IMPS	Implementación del servidor	128
DES-IMPC	Implementación del cliente	80
MEM-FIN	Pruebas y finalización de la memoria	56
	Total:	**400**

Tabla 1. Estimación de tareas y tiempo dedicado

En cada una de las actividades del trabajo que se han enumerado se incluye su propia parte relativa a la redacción de la memoria. Aun así, hay que tener en cuenta que la fase de finalización de la misma requerirá un tiempo adicional.

2.3. Diagrama de Gantt

Se ha realizado, con la ayuda de la herramienta *Smartsheet* [5], un diagrama de Gantt para conseguir una representación visual de las tareas a realizar en el trabajo (Figura 3):

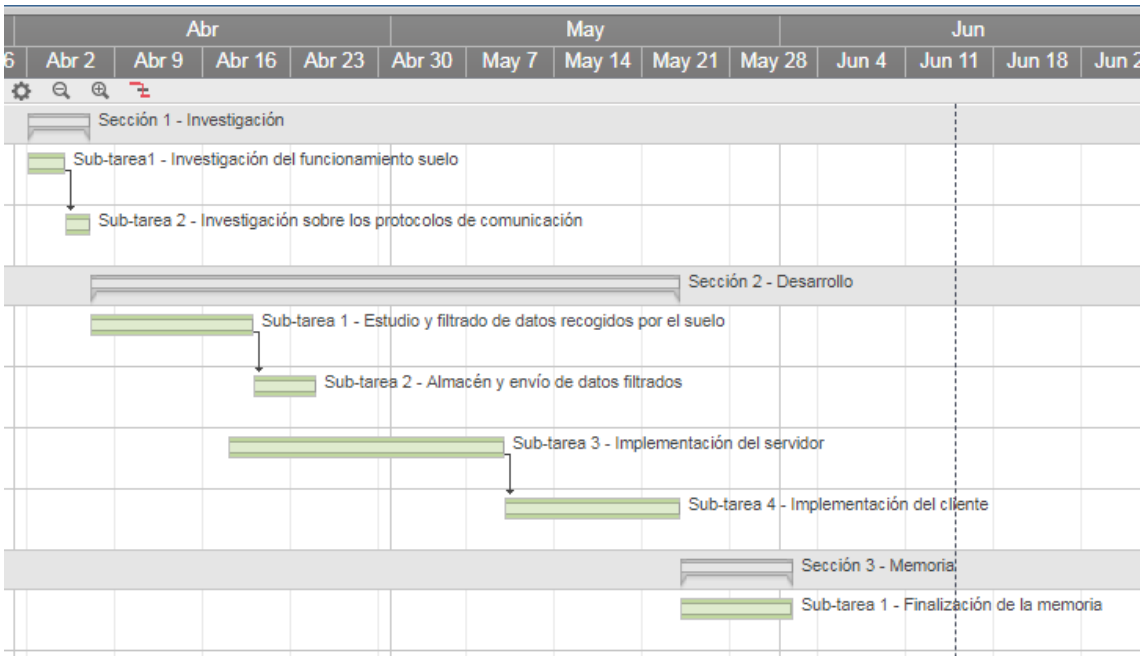


Figura 3. Diagrama de Gantt

Para establecer el calendario estimado se ha supuesto una jornada completa de ocho horas de lunes a viernes sin tener en cuenta fines de semana.

2.4. Presupuesto

2.4.1. Costes de personal

Se enumera a continuación la información del personal que va a participar en el proyecto. Este personal será de dos categorías de acuerdo a la tabla de equivalencias SCP(AEC-ANEIMO) [6] :

- **Analista informático:** será el encargado de realizar un análisis inicial, así como de recopilar información suficiente para establecer el desarrollo del proyecto en lo que respecta a su diseño y obtención de los algoritmos.

- **Programador:** llevará a cabo la implementación así como la instalación de las herramientas necesarias para llevar el proyecto a cabo.

Así pues, las tareas asignadas para estos dos perfiles serán:

- Analista: realización del diseño del proyecto; esto competerá desde la investigación previa tanto del suelo como la elección del protocolo para establecer las tecnologías que se utilizarán en el desarrollo. También tendrá como labor la revisión y finalización de la memoria.
  TAREAS (de acuerdo a la Tabla 1 : INV-SUE , INV-PROT , MEM-FIN.
- Programador: realización del estudio y filtrado de los datos recogidos así como de toda la implementación del sistema. Será tarea suya además la preparación del hardware y software necesario para la realización de todo el proyecto.
  TAREAS (de acuerdo a la Tabla 1): DES-FIL, DES-ENV,DES-IMPS,DES-IMPC.

Personal contratado	Salario Base (€)
Programador	15.442,56
Analista	21.969,50

Tabla 2. Costes de personal, salario base

Teniendo como salario base al establecido en el convenio [6] (recogido en la Tabla 2) y tomándolo como un salario distribuido en 14 pagas, a lo largo de 12 meses con 21 días hábiles de 8 horas, se constatan como costes de personal para este proyecto los siguientes (Tabla 3) :

Personal contratado	Coste total/hora (€)	Horas totales	Coste (€)
Programador	6,57	304	1.995,98
Analista	9,34	96	896,71
		Total costes de personal	**2892,69 €**

Tabla 3. Resumen costes de personal

2.4.2. Costes de equipamiento

Después de calcular los costes de personal, que es la partida de costes más grande, se continúa con el cálculo de costes de equipamiento informático.

El “coste” de un equipo que se declara en el presupuesto no es igual a su precio de compra, sino a la fracción de dicho precio que se amortiza en razón de su uso en el proyecto.

Para el presente proyecto será necesario un equipo informático al que se establece un periodo de amortización de 5 años (Tabla 4).

Equipamiento	Meses de uso	Meses de vida útil	%Uso	Precio (€)	Coste (€)
Equipo informático	2,00	60	100%	2.000,00	66,67

Tabla 4. Costes de equipamiento informático

En esta partida no incluimos el coste de SensFloor ya que será su uso será cedido por el CEATIC sin coste.

2.4.3. Otros costes

En esta partida se incluyen todos los gastos de fungibles, licencias y demás relacionados con el desarrollo del proyecto y que se utilizarán durante el periodo de ejecución del mismo. Sin embargo, se utilizará software libre y *OpenSource* por lo que no se añade gasto en esta partida exceptuando el uso de la herramienta Excel de Microsoft Office que se incluye (al igual que el sistema operativo Windows utilizado) como parte de la partida equipamiento, ya que se considerará incluido en el coste del equipo adquirido.

2.4.4. Resumen presupuesto

Se implanta un beneficio en la partida presupuestal del 20%, así como el porcentaje del impuesto sobre el valor añadido establecido como el 21% del total (Tabla 5).

Gastos de personal	2.892,69	€
Gastos de equipamiento	66,67	€
Beneficio (20%)	591,87	€
I.V.A. (21%)	745,76	€
Coste total	**4.296,99**	**€**

Tabla 5. Resumen presupuesto

3. TECNOLOGÍAS UTILIZADAS

3.1. SensFloor

SensFloor se compone de una superficie con microelectrónica integrada y sensores de proximidad que se puede instalar debajo de prácticamente cualquier tipo de revestimiento de suelo (Figura 4). Cuando una persona camina por el suelo, los sensores integrados en la base se activan y se envía una secuencia de eventos específicos que contienen su ubicación y el tiempo a la unidad de control central. Con la ayuda del reconocimiento de patrones y el cálculo, estas señales se pueden utilizar para identificar diferentes tipos de eventos. Así, es posible detectar la presencia de personas, la dirección de su movimiento, y reconocer a una persona acostada en el suelo después de una caída. [7]

Además, el sistema SensFloor puede utilizarse para contar personas y evaluar el flujo de visitantes a un lugar, demostrando que el control de acceso y la detección de presencia sin cámaras son factibles. [7]

Figura 4 . Ilustración SensFloor [7]

El principio de medición capacitiva permite una ventaja única en comparación con los sensores de presión: como los sensores reaccionan desde una cierta distancia sin contacto directo, no hay restricción en el revestimiento del suelo. La capa base SensFloor tiene un espesor de 3 mm, y es instalable por debajo de

otro pavimento. La única excepción es escoger para éste un material conductor, debido a que en otro caso su blindaje no permitiría realizar una medición de la capacitancia. [7]

La base SensFloor es capaz de detectar personas que caminan o se acuestan sobre ella y diferencia entre estas situaciones. Para este propósito, la base tiene varias áreas sensibles que detectan cambios de la capacitancia por encima de su superficie. Si estos cambios están por encima de un límite dado, los módulos de radio integrados transmiten los valores de las capacidades medidas a uno o varios transceptores SensFloor en su entorno. El transceptor utiliza los datos para identificar diferentes eventos como un paso o una caída en el piso. [7]

3.1.1. Características técnicas del suelo

SensFloor está disponible en distintas medidas y formatos. La instalación a la que se tiene acceso se compone de parches rectangulares, divididos a su vez en triángulos, como el que se muestra en la Figura 5. La Tabla 6 detalla sus características técnicas.

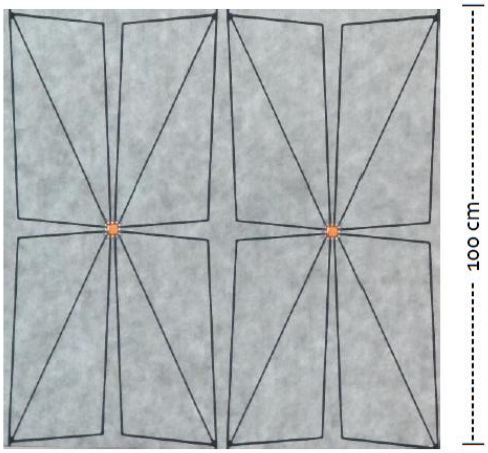


Figura 5. Esquema placa suelo SensFloor [5]

Anchura / longitud de la base	100 cm / 50 cm
Grosor de la capa / peso	2,5 + - 0,6 mm / 720 g + - 60 g por m2

Cuadrícula	100 cm x 50 cm, 2 módulos de radio / 16 áreas de sensores / m2
Corriente por módulo / m2	Máx. 25 mA / 50 mA
Tensión de alimentación	+ 5V - + 12V (se requiere toma de tierra)
Frecuencia de radio	868,0 MHz
Código	Protocolo patentado entre los componentes SensFloor®
Rango / potencia de transmisión	20 m / +10 dBm máx.
Medición / frecuencia	Detección de proximidad capacitiva / 10 Hz
Instalación	Debajo del suelo no conductor
Temperatura de funcionamiento	-10 ° C a + 40 ° C

Tabla 6. Características técnicas de los parches SensFloor instalados [7]

3.1.2. Características técnicas del receptor

Para la recogida de datos se requiere un transceptor SE3-P (que deberá conectarse a un puerto USB de un ordenador), que convierte todos los mensajes de radio SensFloor en una secuencia de datos en serie. Utiliza por defecto el número de puerto COM más alto disponible, aunque puede elegirse uno manualmente. [3]

Simultáneamente, el SE3-P puede utilizarse para enviar datos de configuración del PC a los productos SensFloor. Es posible transmitir los datos del sensor inalámbrico de cualquier producto SensFloor a un ordenador para su procesamiento posterior. [3]

La Figura 6 y la Tabla 7 muestran el SE3-P y sus características técnicas.

Figura 6 . Transceptor SensFloor SE3-P [7]

Fuente de alimentación	PC-USB socket, longitud del cable de 180 cm
Corriente	Max. 20 mA
Frecuencia de radio	868,0 MHz
Código	Protocolo propio entre componentes SensFloor
Rango / potencia de transmisión	20 m campo libre / +10 dBm máx
Antena	Antena PCB integrada
Interfaz de serie	8N1, 115 kBaud
Formato del mensaje	FD (hex) +16 Bytes
Capacidad de procesamiento	Hasta 128 m² de baja resolución SensFloor y 16 alfombrillas SensFloor
Temperatura de funcionamiento	-10 ° C a + 40 ° C

Tabla 7. Características técnicas del transceptor SensFloor SE3-P [7]

3.1.3. SensFloor instalado

Para la realización del trabajo se cuenta con dos instalaciones de SensFloor, un sistema de pruebas que se compone únicamente de dos módulos de 8 triángulos cada uno y un segundo que se compone de un total de 38 módulos (un total de

aproximadamente 20 metros cuadrados de superficie) instalados en el laboratorio inteligente *SmartLab* del CEATIC (Centro de Estudios Avanzados en Tecnologías de la Información y de la Comunicación) en la dependencia 109 del edificio C6 de la Universidad de Jaén. Se incluye un plano del laboratorio (Figura 7) al que se ha superpuesto la distribución de los módulos SensFloor.

Figura 7. Plano del laboratorio inteligente del CEATIC [8]

El laboratorio está dividido en tres partes: salón (que incluye una pequeña entrada), dormitorio (donde también hay un inodoro y un lavabo a modo de cuarto de baño) y cocina. Está equipado con más de 130 sensores de más de 30 tipos diferentes con el fin de monitorizar el comportamiento y los hábitos de las personas en un ambiente controlado y supervisado [8].

En la Figura 8 se muestra un esquema de la distribución de los módulos SensFloor en el laboratorio SmartLab.

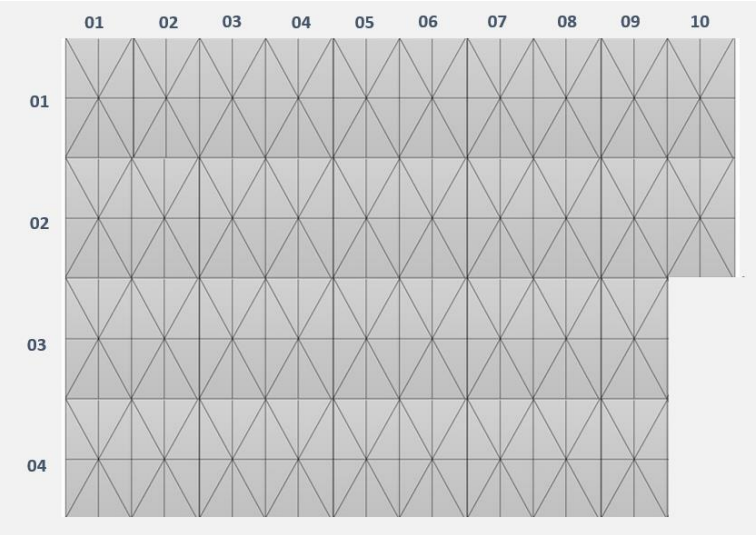


Figura 8. Esquema SensFloor en el laboratorio CEATIC

3.2. Arquitectura cliente/servidor

Se ha decidido desarrollar el sistema con una arquitectura cliente/servidor. El uso de esta arquitectura en el trabajo ofrece una serie de ventajas: [9]

- Administración centrada en el servidor. Se puede desligar la administración del cliente, el cual queda apenas sin trascendencia para la administración.
- Centralización de los recursos. Todos los clientes toman los recursos del servidor lo que evita la redundancia o inconsistencia de datos.
- Aumenta la seguridad. Se centraliza la autentificación en el servidor, por lo que las posibilidades de acceso indebido se reducen.
- Escalabilidad de la instalación. La administración de los clientes es externa a la configuración del servidor.

Inconvenientes de la arquitectura cliente/servidor: [9]

- Coste elevado. Tanto la instalación como el mantenimiento son más elevados debido al perfil muy técnico del lado servidor.

- Dependencia del servidor. Toda la red está construida al rededor del servidor y si éste deja de funcionar o lo hace con un rendimiento inadecuado, afectará a toda la infraestructura.

Afortunadamente, este último inconveniente está superado, al menos en parte, gracias a sistemas como los servidores redundantes, la tolerancia a fallos y los sistemas de almacenamiento en modo RAID.

3.3. Protocolo MQTT vs. REST

Para determinar qué protocolo de comunicación se utilizará entre el servidor y los clientes se ha realizado un estudio de los dos candidatos; REST y MQTT, para valorar cuál ofrece mayores ventajas teniendo en cuenta el tipo de aplicación que se va a desarrollar.

3.3.1. Protocolo MQTT

MQTT (Message Queue Telemetry Transport) es un protocolo muy simple y extremadamente eficiente para publicar / suscribir, usado para la comunicación machine-to-machine (M2M) en “el Internet de las cosas”. Este protocolo está orientado a la comunicación de sensores, ya que permite a los dispositivos abrir una conexión, mantenerla abierta utilizando muy poca energía y recibir eventos o comandos con tan sólo 2 bytes de sobrecarga. Esto lo hace muy útil si utilizamos dispositivos empotrados con pocos recursos hardware. Ahora está en progreso para convertirse en un estándar OASIS M2M. IBM ha abierto todo su código fuente MQTT a través de Eclipse.org, incluyendo el nuevo HTML5 MQTT sobre WebSocket JavaScript, que permite utilizar MQTT en cualquier HTML.

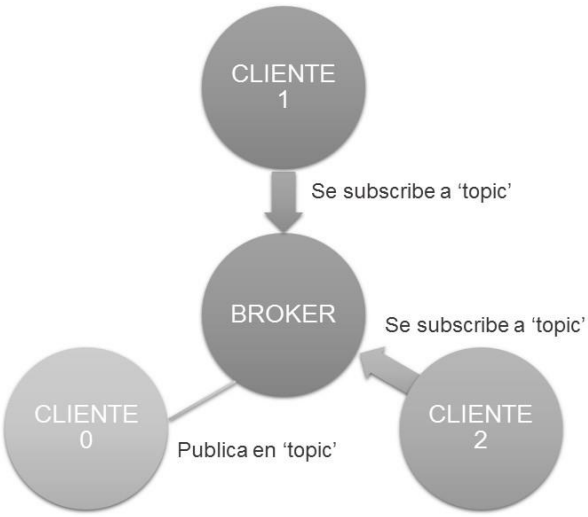


Figura 9. Topología de la arquitectura MQTT

La arquitectura de MQTT sigue una topología de estrella (Figura 9), con un nodo central como intermediador de mensajes (bróker) que hace de servidor con una capacidad de hasta 10000 clientes. Para mantener activo el canal, los clientes mandan periódicamente un paquete (PINGREQ) y esperan la respuesta del bróker (PINGRESP). Esto hace que sea una buena opción para la comunicación remota pero que no sea una gran opción para la comunicación de red local entre dispositivos. El bróker se puede instalar en cualquier servidor público. [10]

MQTT se basa en la arquitectura publicación / suscripción, de manera que un cliente crea un tema (*topic*) donde publicará los mensajes que desee emitir y al cual cada nodo puede suscribirse para recibirlos. Los *topics* pueden estar distribuidos jerárquicamente, de forma que se puedan seleccionar exactamente las informaciones que se desean.

Presenta una serie de ventajas: [10]

- Está especialmente adaptado para utilizar un ancho de banda mínimo
- Es ideal para utilizar en redes inalámbricas
- Consume muy poca energía
- Es muy rápido y posibilita un tiempo de respuesta inferior al resto de protocolos web actuales

- Permite una gran fiabilidad si es necesario
- Requiere pocos recursos hardware.

3.2.2. Protocolo REST

REST es toda interfaz entre sistemas que usa HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON.

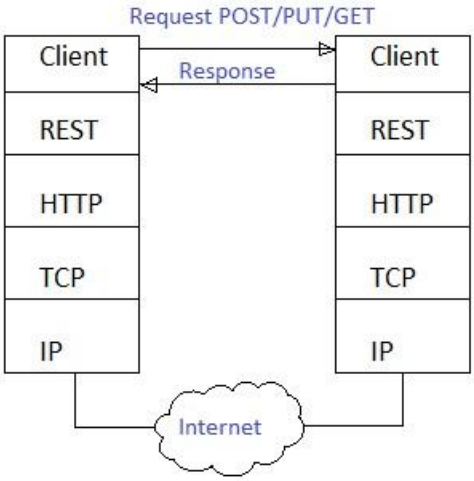


Figura 10. Protocolo REST [9]

REST funciona como una pila de protocolos y está construida sobre capas HTTP / TCP (Figura 10). El protocolo REST utiliza arquitectura basada en bus, donde no se necesita el componente intermediario y los dispositivos finales pueden comunicarse directamente: cada petición HTTP contiene toda la información necesaria para ejecutarla.

Las acciones más importantes sobre los recursos son POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar) que estarán identificados con una URI. [11]

Los objetos en REST siempre se manipulan a partir de la URI como elemento identificador de cada recurso en el sistema REST. Esto facilita la existencia de una interfaz uniforme que sistematiza el procesamiento de la información.

Las principales ventajas que ofrece REST para el desarrollo son: [11]

- Separación entre el cliente (interfaz de usuario), el servidor y el almacenamiento de datos
- Mejora la portabilidad de la interfaz a otro tipo de plataformas
- Aumenta la escalabilidad de los proyectos
- Permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente
- Fácil migración a otros servidores para realizar todo tipo de cambios en la base de datos
- Independencia del tipo de plataformas, lenguajes o entornos de desarrollo. Sólo es necesario mantener el lenguaje de intercambio en las respuestas a las peticiones (XML, JSON, …).

3.2.3. Comparativa MQTT Vs. REST. Justificación de uso

En la Tabla 8 se recoge una comparativa como ayuda para la visualización de las características de ambos protocolos recopiladas.

Característica	MQTT	REST
Forma	Cola de mensajes	Representational State Transfer
Tipos de mensajes utilizados	Conectar, desconectar, publicar, subscribir,…	GET, PUT POST y DELETE
Arquitectura	Publicar/Subscribir	Solicitar respuesta
Necesidad de intermediario centralizado	Indispensable, los dispositivos se comunican a través de un bróker	No es necesario, los dispositivos se comunican directamente
Protocolo de transporte	TCP / IP	TCP / IP
Protocolo de seguridad	TLS	HTTPS

Tolerancia a fallos	Bróker en SPoF	Servidor en SPoF
Alcance	Cloud to cloud	Cloud to cloud

Tabla 8 . Tabla comparativa MQTT y REST

Se decide utilizar la arquitectura MQTT. La principal ventaja que supone el uso de la arquitectura MQTT es la política de publicación / suscripción a *topics*. Esto hará que se reduzca el tráfico de datos, ya que la aplicación cliente no tendrá que estar preguntando a cada mínimo intervalo de tiempo si se ha producido una nueva entrada de datos al servidor, sino que éste publicará un nuevo dato en un *topic* y el cliente estará suscrito al mismo para recibirlo en el caso de que se produzca. Mientras tanto, estará a la espera sin provocar tráfico de datos alguno.

3.4. Base de datos

Es imprescindible para el sistema la utilización de una base de datos. En ella se almacenarán los datos proporcionados por el suelo previamente preprocesados para excluir aquellos que sean irrelevantes para el propósito. Así, se estructurará la información obtenida y será almacenada de forma ordenada para poder hacer uso de ella desde el cliente lo más cómodamente posible.

Se propone MySQL como sistema de gestión de bases de datos para implementar el sistema.

3.4.1. Justificación de uso

El plantear el uso de MySQL se justifica alegando su rendimiento, confiabilidad y facilidad de uso. La flexibilidad de plataforma es también una característica destacable de MySQL, ya que soporta distintas versiones de Linux, UNIX y Windows, entre otros.

Por otro lado, MySQL ofrece uno de los motores de bases de datos transaccionales más potentes. Las características incluyen un soporte completo

de ACID (atómica, consistente, aislada, duradera) [12] bloqueo a nivel de filas, posibilidad de transacciones distribuidas y soporte de transacciones con múltiples versiones, donde los lectores no bloquean a los escritores y viceversa También se asegura la integridad completa de los datos mediante integridad referencial y niveles de aislamiento de transacciones especializados.

Otras características como las tablas en memoria, índices B-tree y hash, y tablas comprimidas hasta un 80% hacen de MySQL una buena opción para aplicaciones web y de *business intelligence* [13] , por lo cual actualmente MySQL es de las opciones más utilizadas en entornos Web.

MySQL ofrece características de seguridad que aseguran una protección absoluta de los datos; potentes mecanismos para asegurar que sólo los usuarios autorizados tengan acceso al servidor. [13]

Uno de los motivos principales por los que se integra MySQL en esta propuesta es porque hace posible utilizar drivers (ODBC, JDCBC, entre otros) que permiten que distintos tipos de aplicaciones puedan usarlo como gestor de bases de datos sin importar si se desarrolla en PHP, Perl, Java, Visual Basic, o .NET.

Por último, es destacable que al ser propiedad de Oracle, MySQL tiene un modelo de coste y soporte que ofrece una combinación única entre la libertad del *open source* y la confianza de un software con soporte, lo cual es una de las razones por las cuales es uno de los gestores preferidos en el mundo. [13]

3.5. JSON

JSON es un formato ligero para intercambio de datos que surge como alternativa a XML en AJAX. Se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia, cuando la fuente de datos es explícitamente de confianza y donde no es importante el no disponer de procesamiento XSLT para manipular los datos en el cliente. Este formato de datos es más liviano que XML y es en apariencia más sencillo, ya que utiliza el código JavaScript como modelo de datos. [14]

JSON se basa en dos estructuras:

- Una colección de pares nombre / valor. En los diferentes lenguajes de programación se implementa como un objeto, registro, estructura, diccionario, tabla hash, lista con clave, o una matriz asociativa.
- Una lista ordenada de valores. En la mayoría de los idiomas, esto se implementa como una matriz, vector, lista, o secuencia. [15]

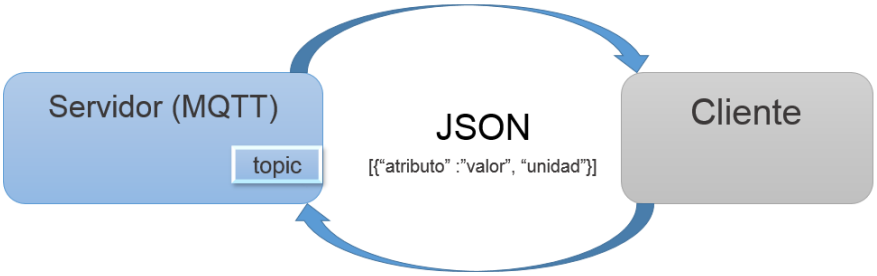


Figura 11. Uso JSON

En nuestro caso, se utilizará como formato para enviar los datos (Figura 11) tratados del *dataset* original que se recibe en el servidor al bróker por su agilidad y porque su estructura ayudará a organizar de modo jerárquico los datos en función de su procedencia. Estas categorías ayudarán en el tratamiento posterior de los datos como se explica más adelante.

4. DESARROLLO

4.1. Especificación de requerimientos

Al ser la primera fase del proceso de ingeniería, se van a determinar cuáles son las claves donde el sistema debe dar las mejores soluciones. Es muy importante definir bien este punto, ya que será el pilar del software desarrollado. Para ello, se procede a realizar una descripción completa del comportamiento del sistema, dividiendo los requisitos en dos grupos:

4.1.1. Requerimientos funcionales

- El sistema debe ser capaz de recoger los paquetes de datos que envía SensFloor y analizarlos, determinando cuáles son significativos y cuáles prescindibles para nuestro propósito
- El sistema debe ser capaz de almacenar estos datos en una base de datos similar a la utilizada en la plataforma ya desarrollada [16]
- El sistema debe ser capaz de enviar los datos a un cliente web para poder visualizarlos
- El cliente debe ser capaz de determinar cuándo hay una presencia sobre el suelo y visualizarla en una interfaz web amigable
- El cliente debe ser capaz de detectar una caída y visualizarla en la interfaz
- El cliente debe ser capaz de ir mostrando en tiempo real la ruta seguida por una persona
- La interfaz debe proporcionar la opción de mostrar distintas configuraciones del suelo. Al menos dos: un sistema de prueba y el suelo instalado en el laboratorio

4.1.2. Requerimientos no funcionales

- El sistema debe enviar los datos con la agilidad suficiente para que el cliente los reciba en tiempo real

- Los datos deben ser almacenados de manera que sea fácil su integración en posteriores implementaciones
- La interfaz debe ser sencilla y usable

4.2. Metodología utilizada

Para la realización del trabajo se seguirá una metodología ágil basada en un desarrollo iterativo e incremental. En este tipo de desarrollo el proyecto se planifica en diversos bloques temporales [17] (en el presente trabajo entre una y dos semanas) llamados iteraciones o *sprints*.

Se entienden iteraciones como pequeños sub-proyectos, de manera que en cada una de ellas se utiliza el mismo proceso de trabajo consiguiendo proporcionar en cada iteración un resultado completo sobre el sistema final. Así el cliente puede percibir los avances del proyecto de forma incremental.

En cada iteración, por tanto, se consigue una evolución del trabajo a partir de los resultados completados en las iteraciones anteriores, ampliándolo con nuevos objetivos/requisitos si fuese necesario o mejorando los que ya fueron acabados [17].

4.3. Análisis y diseño inicial

A continuación se especifica cuál será el esquema de funcionamiento del sistema para satisfacer los requisitos analizados.

Gráficamente, se muestra el sistema diseñado para el trabajo en Figura 12:

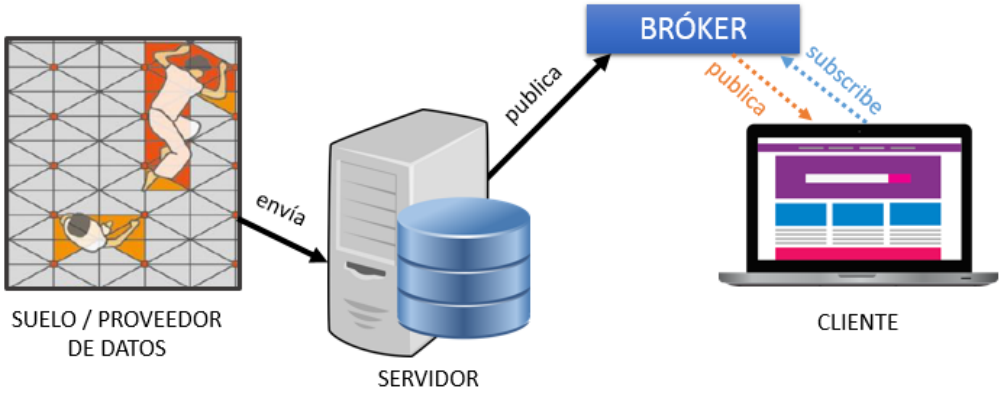


Figura 12 . Esquema de la apliación

En el esquema del sistema el suelo provee de datos al servidor. Éste realiza el filtrado de los datos y los almacena en una base de datos (dejando un registro de todos los datos recogidos que permitirán un uso posterior) y los publica en el bróker. El cliente se suscribe al bróker para recoger la información que el usuario desee mostrar para visualizarla en una aplicación web.

4.4. Primera iteración

En esta primera iteración se realiza un estudio para identificar los datos que proporciona SensFloor a través del receptor. La finalidad de este estudio es descubrir qué datos nos proporcionan suficiente información para establecer que una persona está sobre el suelo y cuándo no hay nadie pisándolo. Una vez establecido este filtrado, se determinará qué estructura tendrá la base de datos MySQL para almacenarlos.

La duración de esta iteración será de dos semanas. Como resultado de esta iteración se conseguirá un estudio de los dataset proporcionados por SensFloor y el diseño e implementación de la base de datos que los almacenará.

4.4.1. Filtrado de datos

Antes de comenzar con el estudio del formato del *dataset* que se recoge de SensFloor, es conveniente conocer la estructura del mismo para entender mejor a qué nos referimos con cada uno de los parámetros.

El conjunto del suelo que cubrirá el espacio de una habitación se divide en zonas rectangulares que, a su vez, se dividen en un total de ocho triángulos rectángulos como se muestra en la Figura 13:

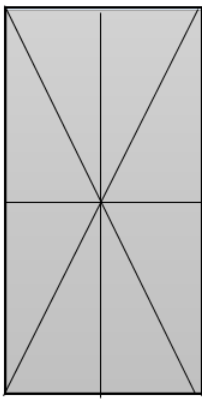


Figura 13. Disposición de las ocho zonas para unas mismas coordenadas

Como se ha explicado en la en apartado 3.1. de esta memoria, entre las características de SensFloor está que los sensores no miden por diferencia de presión sino de capacitancia. Antes de continuar es importante entender qué es la capacitancia y qué es lo que realmente se mide con el sensor. La capacitancia de un dispositivo es la medida de su capacidad de almacenar carga y energía potencial eléctrica [18]. Es un valor constante dependiente de la geometría y del material, por lo que un sensor capacitivo se basa en el hecho de que cuando se acerca un objeto se produce el fenómeno de polarización en su superficie. Es decir, cuando una persona se sitúa sobre el suelo aumenta la capacitancia de la placa del sensor y cuando se aleja se reduce. Por tanto, en realidad se miden diferencias de capacitancia, aunque en adelante nos refiramos como *capacitancia* a este valor.

Otro concepto que es conveniente recordar ya que lo utilizaremos en adelante es el de *transceptor*. Un transceptor es un dispositivo que cuenta con un transmisor y un receptor que comparten parte de la circuitería o se encuentran dentro de la misma caja [19]. Durante el estudio debemos tener en cuenta que SensFloor emite el mensaje que recibe el transceptor. El transceptor pasa estos mensajes al PC pero además puede, a su vez, producir mensajes. Atendiendo a este hecho los mensajes se clasifican en dos categorías: **mensajes de los transceptores** y **mensajes de los módulos del suelo**. Así pues, también se debe buscar una forma de identificar qué parte del mensaje indica su emisor para filtrarlo.

Por último, no sólo es diferenciable el emisor del mensaje que se recibe, sino también tipo de mensaje. Existen dos tipos de mensajes: **mensajes del sensor** *(sensor messages)* **con diferencias de capacitancias** y **mensajes de eventos** (*event messages)*. Estos mensajes de eventos los proporciona el suelo con un sistema propio de detección de patrones. Sin embargo, el fabricante no ofrece documentación acerca del mismo, así que no se han considerado válidos y se filtrarán obviándolos en el trabajo.

4.4.1.1. Formato del mensaje

Cada mensaje de SensFloor está codificado en 16 bits. Un mensaje tipo tiene la siguiente apariencia en formato hexadecimal (Tabla 9) :

		DIRECCIÓN						SIGNIFICADO		BLOQUE DE DATOS							
Posición		**1º**	**2º**	**3º**	**4º**	**5º**	**6º**	**7º**	**8º**	**9º**	**10º**	**11º**	**12º**	**13º**	**14º**	**15º**	**16º**
FORMATO HEXADECIMAL		**01**	**E4**	**01**	**02**	**FE**	**00**	**11**	**01**	**98**	**82**	**7F**	**7F**	**7F**	**7F**	**92**	**A3**
FORMATO DECIMAL		1	228	1	2	254	0	17	1	152	130	127	127	127	127	146	163

Tabla 9. Mensaje 'tipo' de SensFloor

Los primeros 4 bytes codifican la dirección (*address*) del "módulo de radio" del cual procede el mensaje.

El séptimo byte codifica el sentido (*meaning)* del mensaje y los últimos ocho bytes contienen el bloque de datos de información útil (*data block)*.

Los restantes bytes son para propósitos especiales solamente y no serán requeridos para procesar los datos del sensor.

A continuación, se irán desgranando uno a uno cada uno de estos bloques de datos para determinar cuáles de ellos son interesantes para el desarrollo del trabajo.

1. *Dirección (address)*

	DIRECCIÓN						SIGNIFICADO		BLOQUE DE DATOS							
Posición	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º	13º	14º	15º	16º
HEX	**01**	**E4**	**01**	**02**	FE	00	11	01	98	82	7F	7F	7F	7F	92	A3
DEC	**1**	**228**	**1**	**2**	254	0	17	1	152	130	127	127	127	127	146	163

Tabla 10. Bloque de bytes para determinar la dirección

La dirección (*address*) de un mensaje SensFloor consiste en un espacio de cuatro bytes: los dos primeros son etiquetados como ID (rID) (01 y e4 en el ejemplo de la Tabla 10. Este valor es el mismo para todos los dispositivos SensFloor en una instalación. Sólo los dispositivos con el mismo rID pueden comunicarse entre sí.

Los dos bytes siguientes del campo de dirección (01 y 02 en la Tabla 10) enumeran los dispositivos SensFloor. Para los módulos de radio en el SensFloor, los valores son usualmente las coordenadas (fila y columna) de los módulos dentro de la instalación.

Los valores 01 02, por tanto, determinan que se trata de la primera fila y la segunda columna de la capa inferior (en azul en la Figura 14)

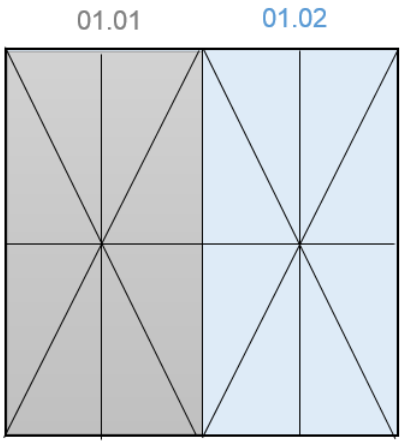


Figura 14. Determinación espacio del suelo que se identifican con los bytes de dirección

Como los módulos están dispuestos en una rejilla regular con distancia dada (usualmente 50 cm), la ubicación del módulo en la sala se puede calcular a partir de estas coordenadas.

Las direcciones de los transceptores SensFloor se enumeran usualmente usando los valores 80 00, 80 01, etc., de manera que los mensajes de los transceptores se pueden distinguir fácilmente de los mensajes de los módulos del suelo comprobando estos bytes.

2. *Sentido (Meaning)*

El séptimo byte (11), destacado en el ejemplo de la Tabla 11, codifica el sentido del mensaje y necesita ser interpretado poco a poco.

	DIRECCIÓN						SIGNIFICADO		BLOQUE DE DATOS							
Posición	1º	2º	3º	4º	5º	6º	**7º**	8º	9º	10º	11º	12º	13º	14º	15º	16º
HEX	01	E4	01	02	FE	00	**11**	01	98	82	7F	7F	7F	7F	92	A3
DEC	1	228	1	2	254	0	**17**	1	152	130	127	127	127	127	146	163

Tabla 11. Bloque de bytes para determinar el sentido

Comenzando con el bit menos significativo a la derecha, los bits tienen el siguiente significado en conjunto:

Posición bit	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
11 =	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**

- **bit 0:** si el mensaje proviene de un módulo de sensor, la activación de este bit indica un cambio significativo de los valores de capacitancia y el bloque de datos del paquete contiene estos valores. Si el mensaje proviene de un transceptor, es un reconocimiento de un mensaje de evento.
- **bit 1:** si este bit está a 1, indica que el mensaje es un mensaje maestro (*master message)* enviado regularmente por un transceptor SensFloor para sincronizar los módulos de sensores en una instalación.
- **bit 2:** un valor 1 en este bit indica que el mensaje es una solicitud de estado (*status request)* enviada usualmente por un transceptor SensFloor que solicita activamente a un dispositivo SensFloor su estado.
- **bit 3:** si este bit está activado, este mensaje es una respuesta a una solicitud de estado (*status request)*.
- **bit 4:**
    - Si el mensaje proviene de un módulo sensor, este bit con valor 1 indica que los valores de capacitancia en el bloque de datos se interpretan de tal manera que el valor 80 (128 en sistema decimal) corresponde a la capacitancia base del campo sensor. De esta manera se pueden expresar además de las **capacitancias positivas (valores 80 a FF) las capacitancias "negativas" (de 00 a 7F)** correspondientes a la eliminación de un objeto estático de un sensor. Si el bit no está activado, **la capacitancia de base es 00 y sólo se pueden expresar las capacitancias positivas (valores de 00 a FF).**
    - Si el mensaje se transmite desde un transceptor, este bit indica que el mensaje es un mensaje de evento.

- **bit 5:** si está activado indica que el mensaje se utiliza para cambiar la configuración de los parámetros del dispositivo SensFloor. Esta es una función de nivel superior y no será necesaria para un tratamiento sencillo de los datos.
- **bit 6:** activando este bit en un mensaje a un módulo sensor, su medición de capacitancia se recalibra.
- **bit 7:** si este bit está activado, el mensaje proviene de un transceptor y no de un módulo de sensor.

3. *Mensajes del sensor (sensor messages)*

Este bloque corresponde a los últimos 8 bytes del paquete (Tabla 12)

	DIRECCIÓN						SIGNIFICADO		BLOQUE DE DATOS							
HEX	**01**	**E4**	**01**	**02**	**FE**	**00**	**11**	**01**	**98**	**82**	**7F**	**7F**	**7F**	**7F**	**92**	**A3**
DEC	1	228	1	2	254	0	17	1	**152**	**130**	**127**	**127**	**127**	**127**	**146**	**163**

Tabla 12.Bloque de bytes para determinar la diferencia de capacitancias

Como el séptimo byte del mensaje de ejemplo (Tabla 12) es 11 (bits 0 y 4 activados) el paquete es un mensaje de sensor, y los 8 bytes en el bloque de datos son valores de capacitancia con valor de base igual a 80.

Los ocho campos de sensor conectados a cada módulo sensor se enumeran en el sentido de las agujas del reloj. Por lo tanto, los sensores 1, 7 y 8 en el ejemplo tienen una capacidad que es significativamente **superior a la capacitancia base** como muestran los valores 98, 92 y A3 indicados en el ejemplo (Tablas 12 y 13).

HEX	**98**	**82**	**7F**	**7F**	**7F**	**7F**	**92**	**A3**
DEC	152	130	127	127	127	127	146	163

Tabla 13. Bytes del bloque de datos de mensaje de sensor de la Tabla 12

Por lo tanto, el mensaje puede interpretarse como una activación de estos campos sensores en el suelo causados, por ejemplo, por una persona que ha caminado hasta ese lugar en la habitación.

El problema siguiente es determinar a qué triángulo corresponde cada uno de los bytes del mensaje. Para conseguir determinarlo se prueba a recolectar los datos de cada byte del bloque de datos del mensaje mientras alguien pisa durante 3 minutos en uno de los triángulos. Después se desplaza hacia el que sitúa a su derecha, permaneciendo ahí 3 minutos más y a continuación moviéndose al siguiente. Este proceso se repite para los 8 triángulos.

Se representan a continuación los datos obtenidos en un gráfico (Figura 15) para poder visualizar el resultado:

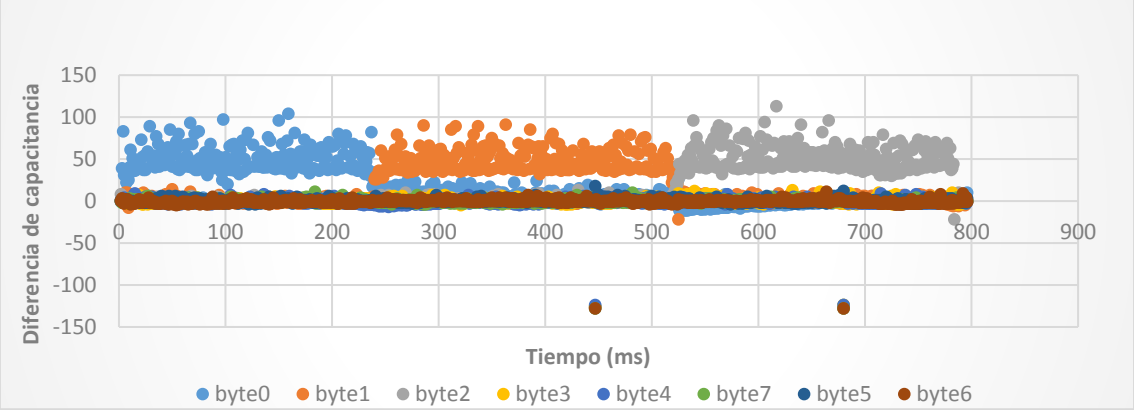


Figura 15. Diferencia de capacitancia frente a tiempo

Como resultado de este estudio se determina, con precisión, la distribución de las áreas sensibles en cada módulo (Figura 16).

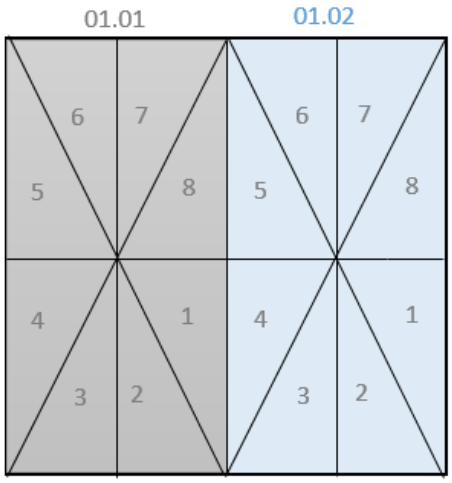


Figura 16. Numeración de las 8 zonas de cada una de las cuadrículas del suelo

4.4.1.2. Establecimiento de patrones

Para establecer los patrones de medición, se realizan una serie de pruebas y se recopilan los datos para analizarlos. Será necesario determinar de forma unívoca cuándo hay una presencia sobre el suelo (y cuando no hay presencia) y en qué región se sitúa ésta.

En los siguientes apartados se determinan los valores que servirán para determinar las reglas que se implementan en el proyecto.

1. *Capacitancia límite*

   Se denomina *capacitancia límite* al valor a partir del cual se determina si existe presencia sobre el suelo. Un valor inferior a ese indicaría que no hay nadie ‘pisando’.

   Para establecer este valor se realiza una prueba con el suelo y se almacenan los valores de capacitancia para realizar un análisis de los mismos.

La prueba consiste en no pisar el suelo durante 20 minutos para establecer qué valores máximos y mínimos se registran durante este tiempo.

Los datos obtenidos se engloban en el la Figura 17 y la Tabla 14.

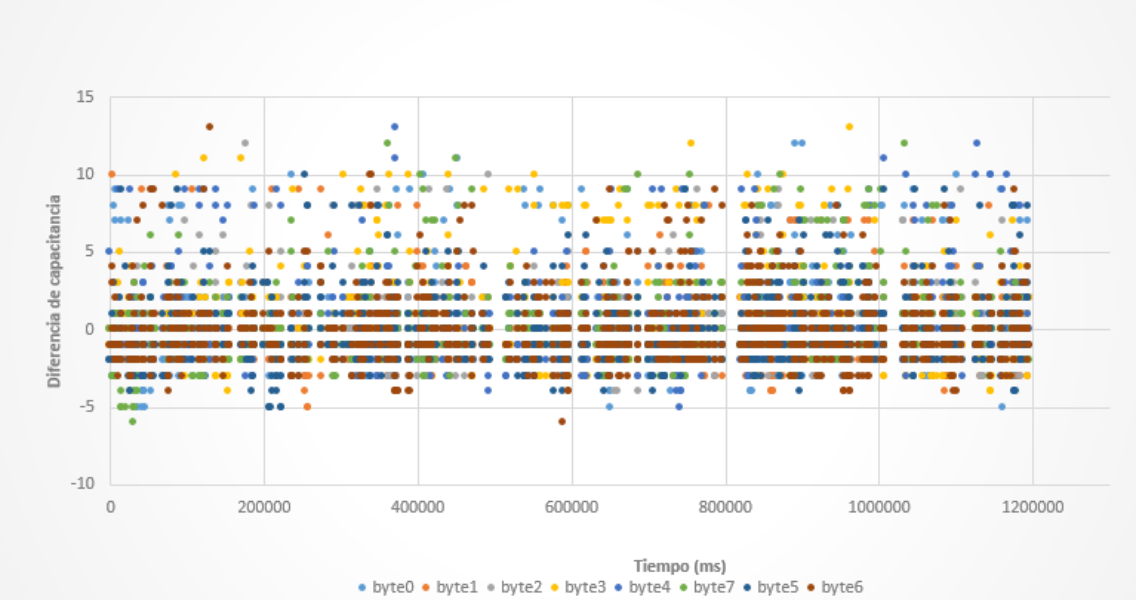


Figura 17. Gráfico de diferencia de capacitancias cuando no hay nadie pisando

Capacitancia máxima	13
Capacitancia mínima	-6
Tiempo transcurrido (ms)	1194895
Tiempo transcurrido (min)	19,91

Tabla 14. Resultados prueba 'nadie pisando'

La capacitancia máxima registrada tiene un valor ’13’ por lo que se establece como *capacitancia límite* un valor de ‘13’.

Por otro lado, puesto que la capacitancia mínima registrada es de ‘-6’ se utilizará también ‘-6’ como *capacitancia límite negativa*. Sin embargo, en la implementación para simplificar se utiliza el valor -13. De manera que un valor comprendido entre -13 y 13 se considerará como ‘nadie pisando’.

2. *Capacitancias cuando se detecta presencia*

Como se ha explicado en el apartado 4.4.1.1 (en el apartado “Mensajes del sensor”), se realiza otra prueba pisando cada 3 minutos en uno de los triángulos (Figura 15) hasta pasar por tres triángulos consecutivos.
De esta prueba se ha obtenido la información recopilada en la Tabla 15:

MINUTO		***CAPACITANCIA***							
1	**MAX**	**104**	26	8	7	9	6	5	11
	MIN	**-3**	-8	-2	-4	-5	-4	-5	-3
	MEDIANA	**42**	0	0	-1	-1	0	-1	0
2	**MAX**	35	**91**	13	7	7	18	7	10
	MIN	-4	**-3**	-3	-5	-7	-3	-4	-4
	MEDIANA	5	**40**	-1	0	-1	0	0	0
3	**MAX**	26	40	**113**	13	9	12	11	5
	MIN	1	-2	**2**	-1	-1	-1	-2	2
	MEDIANA	-1	0	**44**	-1	-1	-1	-1	0

Tabla 15. Resultados prueba 'pisando'

A la vista de estos resultados, se establece que el valor medio de capacitancia cuando hay alguien pisando se encuentra entre 40 y 45, aunque puede tomar valores mucho mayores, llegando hasta cerca de 120.

3. *Frecuencia de datos*

En la realización de las pruebas anteriores se ha detectado que existe una mayor frecuencia de datos cuando los sensores del suelo detectan presencia. Se documenta este hecho analizando el tiempo que pasa entre la llegada de dos datos consecutivos en las pruebas anteriores (Figura 18).

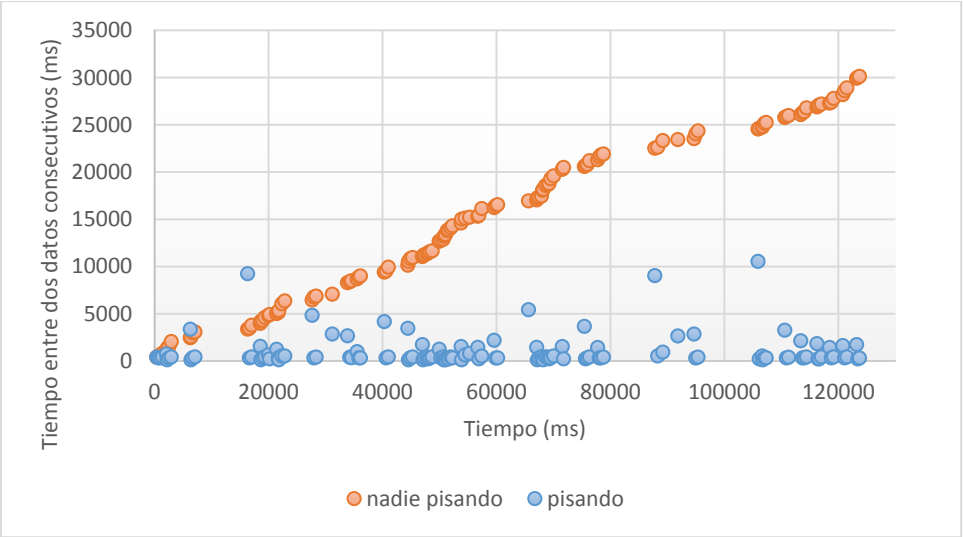


Figura 18. Frecuencia envío de datos

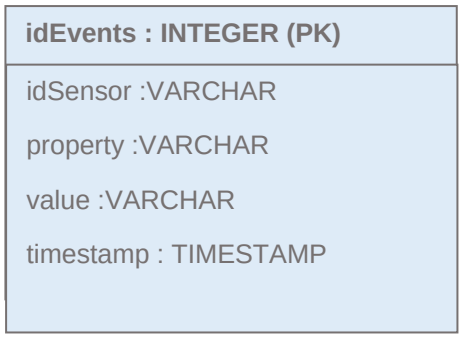
Como se puede observar en la Figura 18, cuando el suelo no detecta presencia envía los datos cada vez más distanciados en el tiempo. Sin embargo, cuando hay presencia, el tiempo de envío entre dos datos consecutivos suele ser estable y no mayor a cinco segundos.

4.4.2. Descripción del almacén de datos

Se almacenarán los datos con el fin de tener persistencia de los mismos en caso de necesitarlos para realizar análisis posteriores. Puesto que el laboratorio del CEATIC cuenta con muchos más sensores que almacenan los datos recogidos en una base de datos común, se intentará que nuestro diseño se adecúe al que ya utilizan para poder unificar todos los dispositivos y sensores en un futuro.

4.4.2.1. Modelo Entidad/Relación

En la Figura 19 se muestra el modelo entidad/relación de la base de datos relacional:



Se realiza el diseño de la base de datos de forma que se adapte a la que se utiliza actualmente en el laboratorio del CEATIC para facilitar la posterior integración de éstos en futuros trabajos.

Tabla	Característica
Suelo	Tabla que almacena los datos recogidos por el suelo

Tabla 16. Tabla base de datos

Así, la base de datos se conformará de sólo una única tabla (“Suelo”, como podemos visualizar en la Tabla 17) cuyos atributos son:

- idSensor: identificará al sensor que ha emitido el dato que se almacena en la tupla.
- property: indicará la propiedad medida (en el presente trabajo siempre tendrá el valor ‘capacitancia’).
- value: almacenará el valor tomado por el sensor.
- timestamp : marca temporal que incida cuándo se recogió el dato.

4.4.2.2. Diccionario de datos

Se incluye en la Tabla 17 un listado detallado y organizado de todos los campos que constituyen la tabla de la base de datos.

El campo “idSensor” se crea como identificador único de cada uno de los triángulos que componen la totalidad del sistema. De manera que resulta de la

unión de los tres identificadores: habitación en la que se encuentra (en el presente proyecto sólo existen dos posibilidades: laboratorio y pruebas), módulo en el que está instalado (el cual lo identificamos por coordenadas) y número de triángulo en el módulo (de uno a ocho).

Campo	Tipo de dato	PK	FK	No nulo	Por defecto	Descripción
idEvents	INTEGER	x		x	(Auto – generado)	Identificador numérico
idSensor	VARCHAR			x		Dirección del sensor (Room-Device-Triangle)
property	VARCHAR			X	“Capacitancia”	Propiedad que está monitorizando el sensor, (Temperatura, Luz, Apertura, Vibración, Humedad....)
value	VARCHAR			x		Valor de la propiedad, en nuestro caso, la variación de la capacitancia.
timestamp	TIMESTAMP			x		Timestamp de cuando se produce el evento.

Tabla 17. Diccionario de datos

4.5. Segunda iteración

En esta segunda iteración, tras hacer el análisis del mensaje que se recibe de SensFloor y teniendo claros los objetivos, se comienza la implementación del servidor.

En el servidor se realizará el tratamiento del *dataset*, se extraerá la información que interesa y se generará un nuevo mensaje basado en JSON que será el que

después se envíe al bróker. Por último, se realizará la conexión y se almacenará la información en la base de datos creada anteriormente.

La duración de esta iteración será de una semana. Como resultados de la misma se conseguirá iniciar la implementación del servidor de manera que consiga recibir, tratar y almacenar el dataset en la base de datos creada en la primera iteración. También se implementará la conexión y el publicador siguiendo el protocolo MQTT.

4.5.1. Diagrama UML

Se realiza la implementación del servidor utilizando el lenguaje Java ya que es un lenguaje orientado a objetos (lo que nos ayudará a simplificar y organizar las estructuras que necesitamos para la implementación), flexible, multiplataforma (lo cual facilitará una posible integración posterior en otro entorno), abierto y, además, existen muchas bibliotecas disponibles de programadores independientes que serán de ayuda durante el desarrollo. De entre estas bibliotecas se han utilizado las bibliotecas 'RXTS Java Communication' y 'Paho Java Client' (cliente MQTT). Se incluye en la Figura 19 el diseño inicial en un diagrama UML. Como se puede observar, el diseño se compone de tres clases. La clase principal (*main)* se encarga del enlace con SensFloor, realizar la conexión con el bróker y convocar a la clase *PublishMQTT* , la cual publicará el mensaje en el bróker. La clase *FloorEvent* es la encargada de ordenar y clasificar los atributos del dataset recibido.

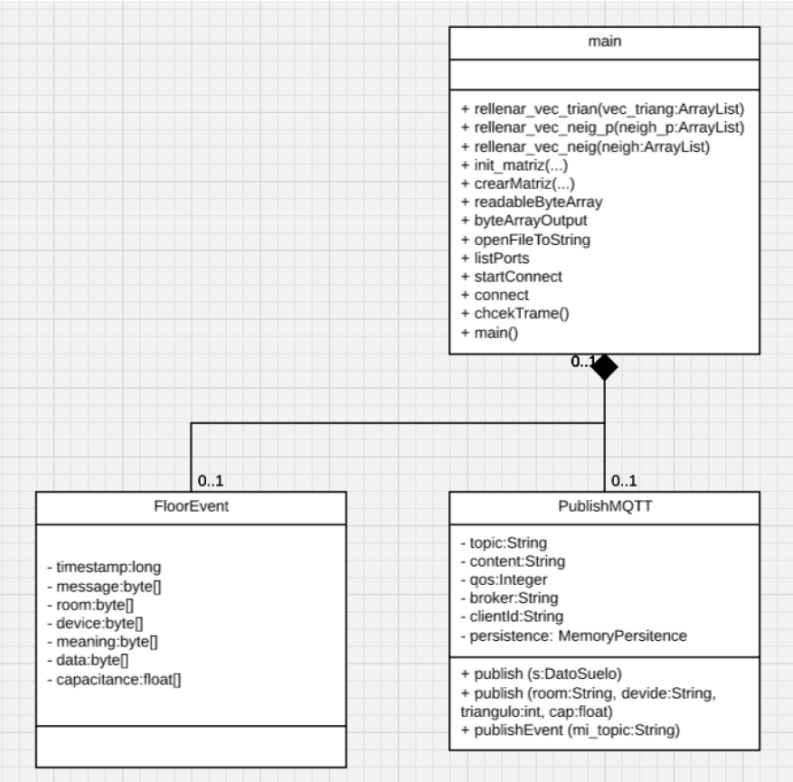


Figura 19. Diagrama inicial UML

4.6. Tercera iteración

En esta iteración se instalará y configurará el bróker necesario para el paso de mensajes entre el servidor y el cliente web. A continuación, se establecerá una estrategia en el sistema de *topics* para clasificar el paso de los mensajes.

La duración de esta iteración será de una semana. Como resultado se obtendrá un bróker listo para recibir el paso de mensajes entre el servidor y el cliente.

4.6.1. Instalación del bróker

Como bróker para el trabajo se opta por instalar Eclipse Mosquitto (versión 1.4.2), porque es un intermediario de mensajes de código abierto que

implementa las versiones del protocolo MQTT 3.1 y 3.1.1. [20]. Así, facilita un método para realizar paso de mensajes aplicando el modelo de publicación / suscripción. Esto lo hace adecuado para el tráfico de mensajes en el "Internet de las cosas" por lo que es muy utilizado en domótica con sensores de baja potencia o dispositivos móviles como teléfonos, ordenadores integrados o microcontroladores como Arduino [20] y será adecuado para transmitir la información proporcionada por SensFloor hacia un cliente externo ya que es muy sencillo conectarlo a una aplicación (tanto móvil como web) desde cualquier lenguaje de programación.

Se instala Mosquitto sobre una máquina virtual con sistema operativo Debian Jessie y se inicializa el servicio. Su descarga e instalación en Linux es muy sencilla, ya que el propio gestor de paquetes de Debian lo proporciona [21]. El puerto predeterminado para MQTT es el 1883 pero lo modificamos a 8060 (Tabla 8), ya que el puerto por defecto no está encriptado [22] y no es seguro.

IP	150.214.174.25
Puerto	8060

Tabla 18. Datos conexión bróker MQTT

4.6.2. Establecimiento de *topics*

Como se ha adelantado en el apartado 3.2.3., en la arquitectura MQTT es muy importante el concepto de *topic,* ya que a través de estos *topics* se articula la comunicación. Tanto emisores como receptores deben estar suscritos a un *topic* común para poder emprender la comunicación**.** Este concepto es prácticamente el mismo que se emplea en colas, donde existen un publicador (que publica o emite información) y unos suscritores (que reciben dicha información) siempre que ambas partes estén suscritas al mismo *topic* [23]. Así, no hay necesidad de configurar un *topic* puesto que publicar en él es suficiente.

Los *topics* funcionan como una jerarquía, utilizando una barra (*/*) como separador. Esto permite compactar temas comunes de forma similar a los

sistemas de archivos [22]. Esto se utilizará en este trabajo para establecer la jerarquía /habitación/sección/triángulo que se visualiza en la Figura 20:

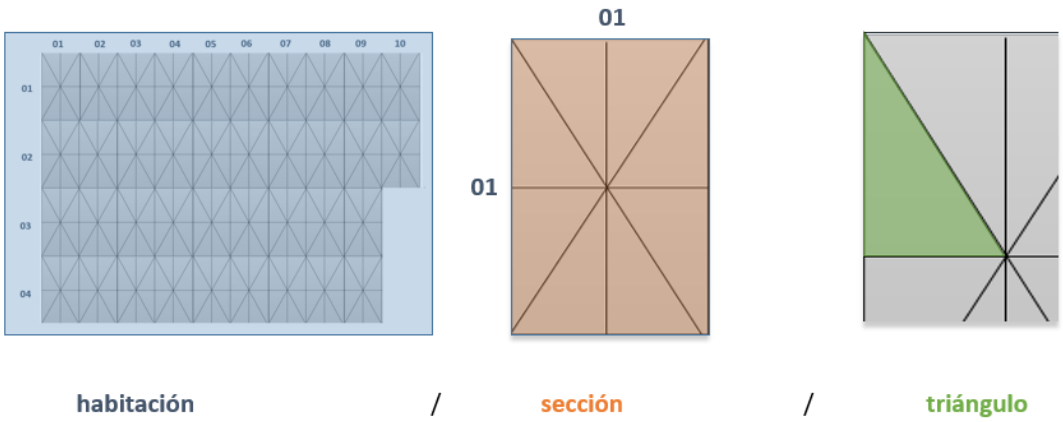


Figura 20. Jerarquía topics MQTT

Con ello, además de ordenar la información, se consigue que el cliente pueda suscribirse solamente a la habitación que desee y a la sección o secciones que requiera, o incluso simplemente a una porción de zona explícita utilizando los triángulos. Otra opción a destacar para los suscriptores es el uso de comodines. Con MQTT existen dos comodines disponibles, '+' y '#':

- '+' se puede utilizar como comodín para un solo nivel de jerarquía. Por ejemplo, suscribirse a "/hab1/+/triangulo2" se suscribirá a todas las secciones pertenecientes a la habitación 1 que cuenten con un triángulo 2.

- '#' se puede utilizar como comodín para suscribirse a todos los niveles restantes de jerarquía. Esto significa que debe ser el carácter final en una suscripción [22] . Por ejemplo, suscribirse a "/hab1/#" se suscribirá a todos los triángulos de todas las secciones pertenecientes a la habitación 1.

4.7. Cuarta iteración

En esta iteración, ya configurado el bróker y listo el servidor, se establecerá la conexión entre ambos. Se iniciará la implementación del cliente que, en este punto del desarrollo, simplemente consistirá en conseguir la conexión con el bróker y comprobar que es correcta.

La duración será de una semana. Como resultados obtendremos ya un servidor publicando mensajes en un topic en el bróker y, suscrito a éste, un cliente básico que es capaz de recibirlos.

4.7.1. Conexión con el cliente

La conexión entre el bróker y el servidor es sencilla y se reduce a unas líneas de código Java. Sin embargo, para conseguir la conexión con el cliente es necesario instalar *websockets*. WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP [24]. Se utilizará para poder establecer la comunicación entre el cliente web (navegador) y el bróker (Mosquitto), creando la conexión cliente-servidor.

Las últimas versiones de Mosquitto (desde 1.4) incluyen la interfaz websocket; sin embargo, si realizamos la descarga del software desde su sitio web (http://mosquitto.org/files/source/mosquitto-1.4.2.tar.gz) ésta no se incluye. Así que es necesario descargar el código fuente y construir un paquete propio [25]. Por último, se configura Mosquitto para usar *websockets* modificando el fichero *config.mk.* En la Tabla 19 se incluyen tanto el puerto como la IP utilizados para la conexión.

IP	150.214.174.25
Puerto	8069

Tabla 19. Datos conexión websocket

4.8. Quinta iteración

En esta iteración comenzaremos el desarrollo del cliente web, que implementaremos en JavaScript. Estableceremos su estructura, el prototipado de la misma e iniciaremos su implementación generando los elementos básicos.

La duración de esta iteración será de una semana. Como resultado se conseguirá una primera implementación del cliente web que será capaz de visualizar dos modelos diferentes (el que simula el suelo del laboratorio y el que simula el suelo de prácticas), detectando ya una presencia utilizando los datos recogidos del tópic.

4.8.1. Cliente web: estructura

El primer paso será establecer el contenido que deberá incluir el cliente:

- Modelo del suelo que simule el suelo instalado en el laboratorio de CEATIC en la Universidad de Jaén
- Modelo que simule el suelo que utilizamos para realizar pruebas
- Menú para navegación en el sitio

El cliente es bastante simple en cuanto a arquitectura. Puesto que las jerarquías son una manera simple para organizar la información, será útil para este sistema que sólo cuenta con tres niveles. Utilizaremos, por tanto, un patrón jerárquico vertical como se puede visualizar en la Figura 21.

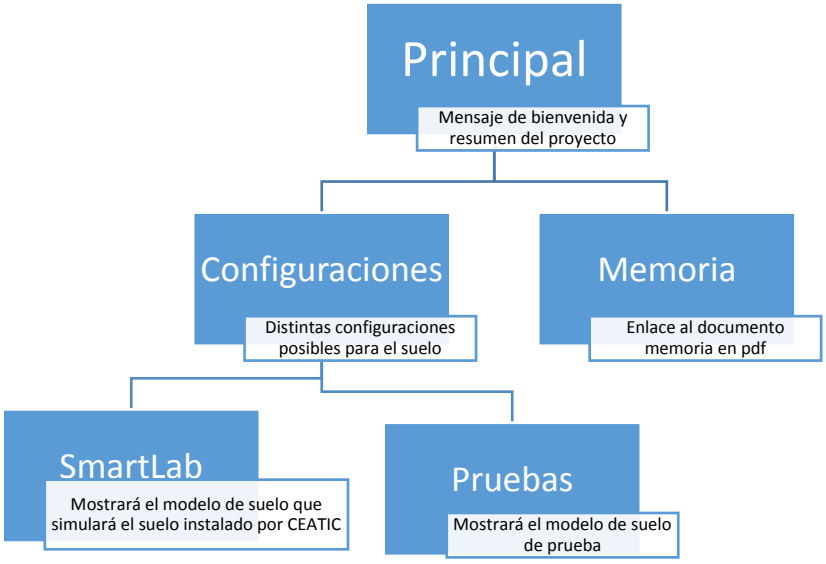


Figura 21. Arquitectura cliente web

Se hace uso de prototipado para reproducir los aspectos básicos de la interfaz. Se compone de tres páginas (Figura 22):

- Inicio: página donde se incluirá el nombre del proyecto y una pequeña presentación del mismo. También añadiremos el menú principal de navegación por el sitio
- Suelo SmartLab: incluirá el modelo del suelo que simule el suelo instalado en el laboratorio del CEATIC de la Universidad de Jaén
- Suelo pruebas: modelo del suelo que simule el suelo de pruebas que sólo se compone de dos módulos

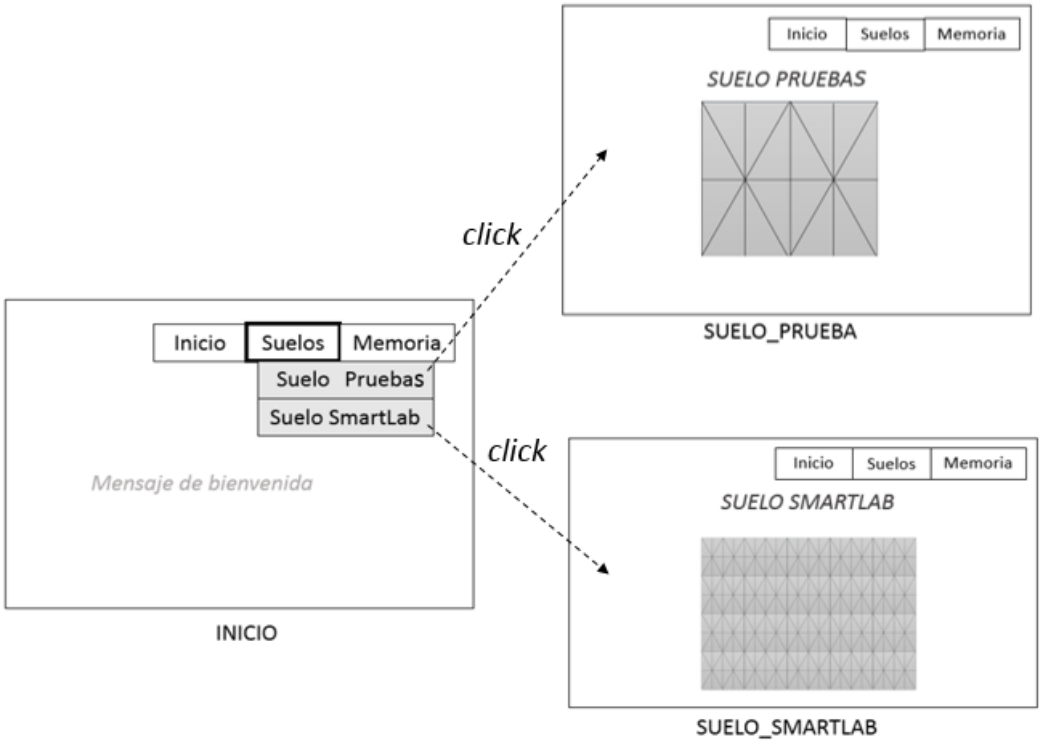


Figura 22. Prototipado cliente web

4.8.2. Cliente web: detección de presencia

Puesto que la capacitancia *límite* que se ha establecido era de valor '13' el sistema detectará que existe una presencia siempre y cuando registre un cambio en el valor de la capacitancia mayor a 13. La implementación de esta regla es un simple condicional, por lo que no lo incluimos como algoritmo en este documento.

Gráficamente, la visualización en la interfaz de una presencia repercutirá en el color del triángulo que ha registrado la variación. El sistema modificará su color gris a verde para indicar que hay actividad en esa zona. En adelante se referirá en el texto a ese triángulo como 'activo' (Figura 23).

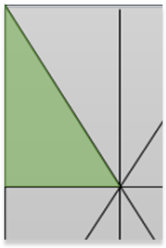


Figura 23. Visualización de un triángulo activo

Por otro lado, se contabilizarán los triángulos que se encuentren activos mediante una estructura de datos, una matriz, con tantos elementos como triángulos compongan la visualización. La matriz (‘*matriz_actividad’)* se inicializará estableciendo todos sus elementos con valor ‘0’. Actualizaremos los valores de esta matriz de manera que cada triángulo activo será un ‘1’ en ella.

4.9. Sexta iteración

El objetivo de esta iteración es conseguir dibujar en la visualización la ruta que sigue una persona que interactúa con el suelo. Lo realizaremos mostrando una línea que unirá los triángulos que se han ido activando consecutivamente durante los últimos instantes.

La duración de esta iteración será de una semana. El resultado será la ampliación del cliente obtenido en la iteración anterior incluyendo la visualización de la ruta.

4.9.1. Establecimiento de rutas

El establecimiento de rutas consistirá en establecer los puntos del suelo por los que ha pasado una persona y unirlos con una polilínea.

Como aproximación para el sistema, ya que no se puede establecer exactamente en qué punto del suelo se posiciona el pie sino que se devuelve la zona triangular que ocupa, se establecerá ese punto como el centroide del triángulo que 'active' al posicionarse. De esa manera, la ruta resultante será una línea que unirá los centroides de los triángulos por los que haya pasado.

La implementación consistirá en una cola donde se irán añadiendo los puntos en orden de llegada. Se ha decidido ubicar la implementación de la cola en el cliente, ya que es necesario calcular la posición del centroide de cada triángulo y ésta será relativa al tamaño del *canvas* en el que se incluye. Así, se almacenará directamente en la estructura de datos la posición 'x' e 'y' de cada punto en lugar de almacenar sólo el triángulo, y se recalcularán cada vez que se pinte la ruta.

Para calcular el centroide del triángulo se utiliza la siguiente fórmula (Figura 24).

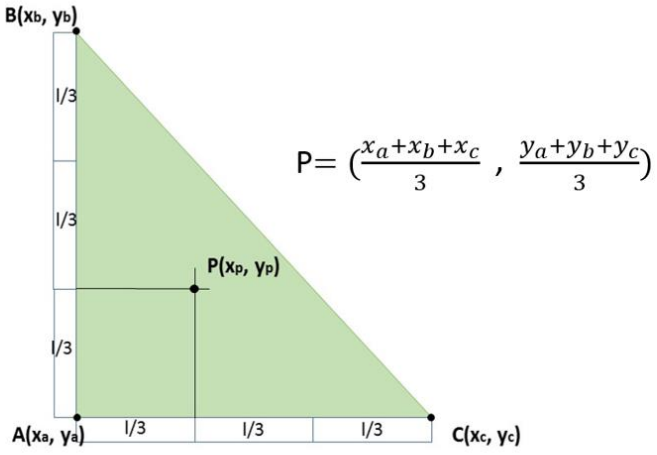


Figura 24. Cálculo del centroide de cada triángulo para dibujar las rutas

La cola tendrá un tamaño máximo, por lo que se irán descartando los puntos más antiguos. De esta manera la imagen sólo mostrará los últimos pasos realizados sobre el suelo (Figura 25). Esto no es un inconveniente, ya que un número muy grande de líneas haría ilegible la imagen. El objetivo es mostrar simplemente los últimos pasos registrados.

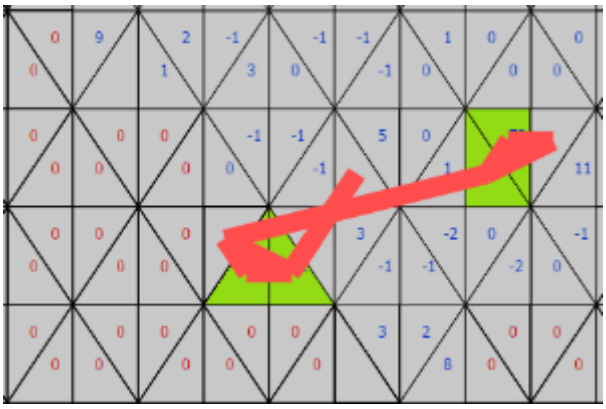


Figura 25. Ejemplo establecimiento de ruta

4.9. Séptima iteración

En esta iteración se ampliará el sistema incluyendo un procedimiento para la detección de caídas. Se debe establecer un patrón para detectar una caída mediante un procedimiento experimental y, conocido el patrón, se deberá implementar un algoritmo para su detección así como establecer cómo reaccionará la aplicación cliente cuando se produzca una caída.

La duración de esta iteración será de una semana. El resultado será la ampliación del cliente obtenido en la iteración anterior incluyendo la detección de caídas.

4.9.1. Pruebas de detección de caídas

El primer paso para implementar la detección de caídas es constatar qué supondrá para el sistema una caída. Los datos con los que se cuenta para detectar si una persona se ha caído no son más que el área que ésta ocupa. Esto es, el número de triángulos que se detectan como ‘activos’ cuando se produce.

Así pues, se han realizado una serie de pruebas para constatar cuántos triángulos existen etiquetados como activos cuando alguien se cae. Se intenta

reproducir un número significativo de posturas que una persona puede experimentar tras una caída (Figura 26).

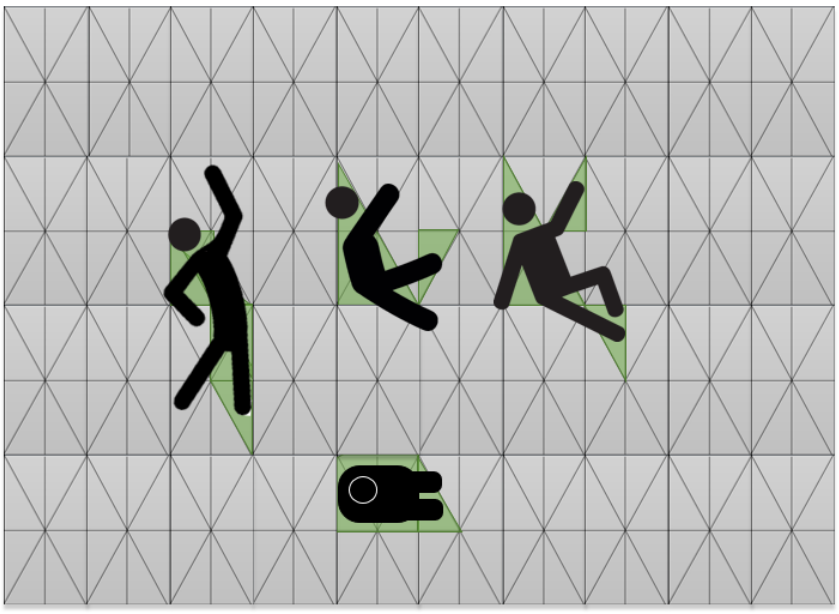


Figura 26. Algunas posturas tras una caída

Se constata que en numerosas ocasiones, cuando simplemente queda una mano apoyada sobre el suelo por ejemplo, el triángulo sobre el que se apoya no sufre un cambio en la capacitancia tan significativo como para que lo trate como 'activo'. Por lo que, como se visualiza en la Figura 26, el número triángulos 'activos' no es exactamente el número de triángulos sobre los que la persona se posa: suele ser un número un poco menor.

Como conclusión de estas pruebas se establece que en una caída, como regla general, se activan al menos 5 triángulos que colindan de alguna manera.

4.9.2. Algoritmo para detectar caídas

Tras establecer la regla que detectará si se ha producido una caída, se construye el algoritmo que ayude a detectar, tras activar un triángulo, qué triángulos son vecinos y por tanto se deben consultar para ver si están activos o no.

El primer paso será establecer la colindancia de módulo. Para ello se establece una matriz que almacene, para cada módulo, qué módulos son sus colindantes al norte, sur, este, oeste, noroeste, noreste, sudeste y suroeste. Se puede visualizar un ejemplo en la Figura 27 donde, de verde, se ha señalado el módulo que contiene al triángulo activo - en este ejemplo el módulo (02, 02) - y cuáles son sus ocho módulos colindantes, en amarillo.

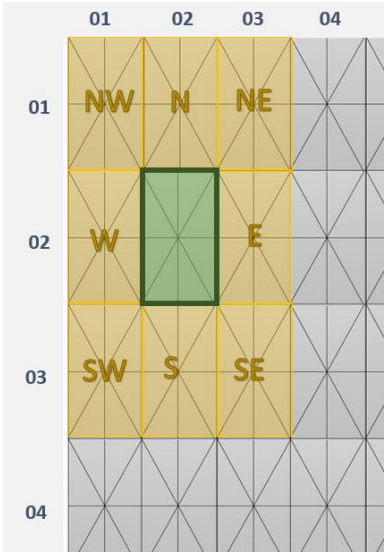


Figura 27. Módulos colindantes

Así pues, esta matriz tendrá tantas filas como módulos existan y 8 columnas, una para cada posición posible (Figura 28).

MÓDULO	W	NW	N	NE	E	SE	S	SW
				...				
02-02	02-01	01-01	01-02	01-03	02-03	03-01	03-02	03-03
				...				

Figura 28. Matriz de módulos colindantes

El siguiente paso será almacenar en una matriz los triángulos vecinos para cada uno de los triángulos. Se pueden establecer dos tipos de vecindad: vecindad de lado y vecindad de vértice. De manera que un 'vecino de lado' será aquél que contenga un lado coincidente con un lado del triángulo activo (dos de sus vértices coincidirán) y un 'vecino de vértice' será aquel donde solo uno de sus vértices

coincida con uno de los vértices del triángulo activo. En la Figura 29, el triángulo en verde (numerado con un 5) es el triángulo activo, mientras que los triángulos coloreados en naranja son sus ‘vecinos de lado’ y, en azul, los ‘vecinos de vértice’.

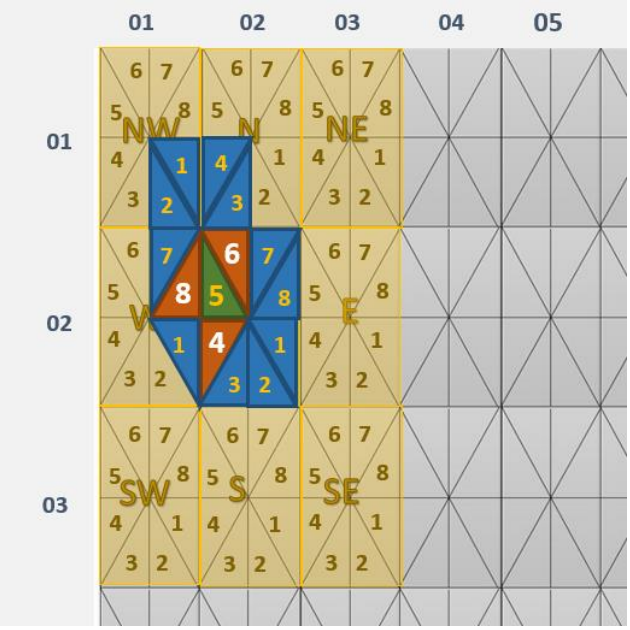


Figura 29. Vecindad de triángulos.

Sin embargo, no bastará con almacenar el número que corresponde a cada triángulo vecino, sino que también se deberá almacenar si corresponde al mismo módulo o a un módulo colindante y, en caso de que sea colindante, cuál de todos los módulos colindantes.

Así pues, la matriz almacenará para cada módulo 8 vectores (uno por triángulo) donde cada vector almacenará sus 14 parejas de datos de vecinos en total (3 ‘vecinos de lado’ y 11 ‘vecinos de vértice’). Cada pareja de datos contendrá el número que corresponda al triángulo y la posición relativa del módulo en el que se encuentra (W, NW, N, NE, E, SE, S ,SW o M si es el mismo módulo). En la Tabla 21 se muestra, como ejemplo, el vector resultante del triángulo 5.

...		
{triangulo: 5	{ vecino :	{ posición = 'M' , triangulo = 6} };
	{ vecino :	{ posición = 'M' , triangulo = 4} };
	{ vecino :	{ posición = 'W' , triangulo = 8} };
	{ vecino :	{ posición = 'W' , triangulo = 1} };
	{ vecino :	{ posición = 'W' , triangulo = 7} };
	{ vecino :	{ posición = 'NW' , triangulo = 2} };
	{ vecino :	{ posición = 'NW' , triangulo = 1} };
	{ vecino :	{ posición = 'N' , triangulo = 4} };
	{ vecino :	{ posición = 'N' , triangulo = 3} };
	{ vecino :	{ posición = 'M' , triangulo = 7} };
	{ vecino :	{ posición = 'M' , triangulo = 8} };
	{ vecino :	{ posición = 'M' , triangulo = 1} };
}	{ vecino :	{ posición = 'M' , triangulo = 2} };
	{ vecino :	{ posición = 'M' , triangulo = 3} };
….		

Tabla 20. Ejemplo vector matriz de vecindad

El algoritmo consistirá en, detectada actividad en un triángulo, sumar el número de triángulos vecinos también activos. Incluimos el pseudocódigo para una mayor comprensión :

```
matriz_vecindad [ ] [ ] ;//matriz de triángulos vecinos Pair <posición, triangulo>
matriz_modulos_colindantes [ ][ ] ; //matriz de módulos colindantes
matriz_actividad [ ][ ]; //matriz que recoge qué triángulos están activos
inicializar_matrices(); // inicialización de las tres matrices anteriores

limite_caida; // número mínimo de vecinos activos para considerar una caída

triangulo_activo = tr_a
modulo_activo = m_a

function  void calcular_vecindad (tr_a , m_a ) {
        num_vecinos = 0;
        for (i in matriz_vecindad [ tr_a ] ) {
                pos_vecino = matriz_vecindad[ tr_a] [ i ] . posicion ;
                tri_vecino = matriz_vecindad[ tr_a] [ i ] . triangulo ;
                modulo_vecino = matriz_modulos_colindantes [m_a]. get( pos_vecino );

                if (  matriz_actividad [modulo_vecino] [triangulo_vecino] == activo) {
                        num_vecinos ++;
                }
        }
        if (num_vecinos > limite_caida) {
                publicar_caida ( ) ;
        }
}
```

4.9.3. Implementación

Se decide implementar todo el proceso de detección de caídas en el servidor. Utilizaremos el bróker con un nuevo *topic* al que denominaremos '/caida'. Cuando se detecte una caída, el servidor publicará un evento en este topic y el cliente, que estará suscrito a él, recibirá el evento y emitirá una alerta directamente, sin tener que tratar el resto de datos.

Esto conlleva una serie de ventajas frente a implementarlo en el cliente:

- Mayor escalabilidad: cualquier otra aplicación podría suscribirse directamente al *topic* '/caida' y recibiría información cuando se detecte una caída sin necesidad de tratar todos los datos que emite el suelo.
- Lenguaje: Java es un lenguaje de programación orientado a objetos y es mucho más fuertemente tipado que Javascript, por lo que con Java podremos tener más control del tipo de dato que manejamos. Esto nos hará la implementación más fácil y fiable.

Por último, para producir el mensaje de audio que se emitirá cuando se produzca una caída creamos un audio con FromTextToSpech [26] con el siguiente mensaje: "Se ha producido una caída". Su emisión será realizada por el cliente web. Así cuando el cliente, que estará suscrito a un nuevo topic denominado '/caida', reciba a través de él un evento, reproducirá el audio.

Como se puede apreciar en la Figura 30, se han realizado cambios en esta iteración añadiendo nuevas clases al diagrama UML inicial. Cabe destacar la clase *Neighbour* que será el tipo de dato que utilizamos para almacenar en la matriz de vecindad. De manera que almacenará el triángulo que corresponda como vecino en la posición relativa del módulo en el que se encuentra (W, NW, N, NE, E, SE, S ,SW o M si es el mismo módulo).

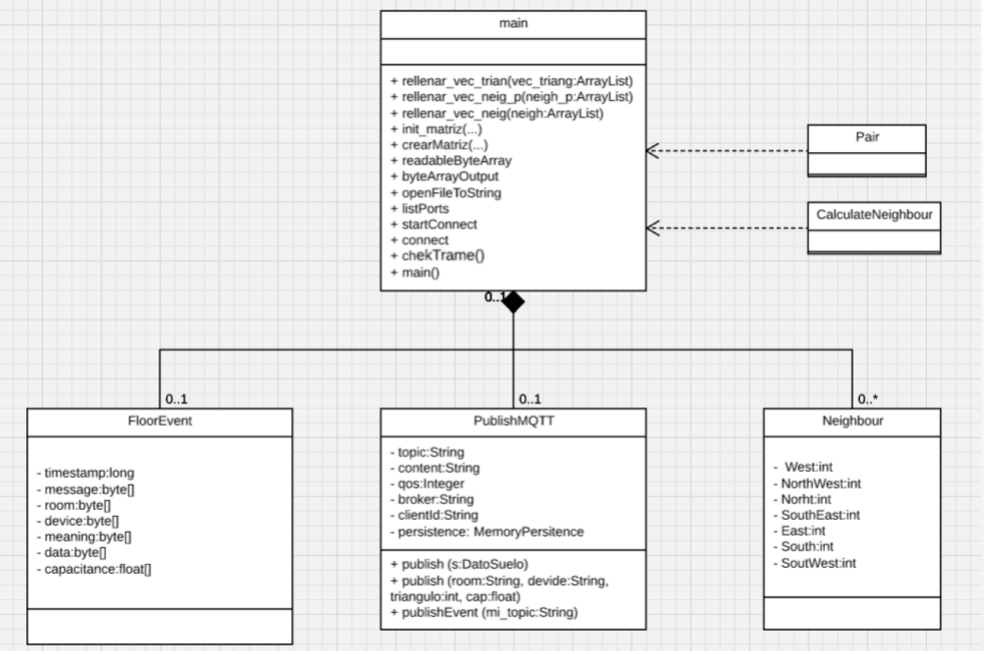


Figura 30. Diagrama UML final

4.10. Octava iteración

En esta última iteración se finalizará la implementación del cliente revisándolo y buscando posibles pequeñas mejoras. Se terminará la presente documentación analizando el proceso realizado para establecer las conclusiones finales así como los conocimientos que se han desarrollado y adquirido y las posibles mejoras que se podrían desarrollar como ampliación del trabajo.

La duración de esta iteración será de dos semanas. El resultado será el sistema finalizado.

4.10.1. Finalización y mejora del cliente

Se mejora la apariencia del cliente web adaptando una plantilla HTML/CSS recogida de TrazosWeb [27] para conseguir una mejor experiencia del usuario final de la aplicación. Se incluyen los resultados del diseño en las Figuras 31 a 34.

Figura 31. Aspecto final cliente. Página de inicio (I)

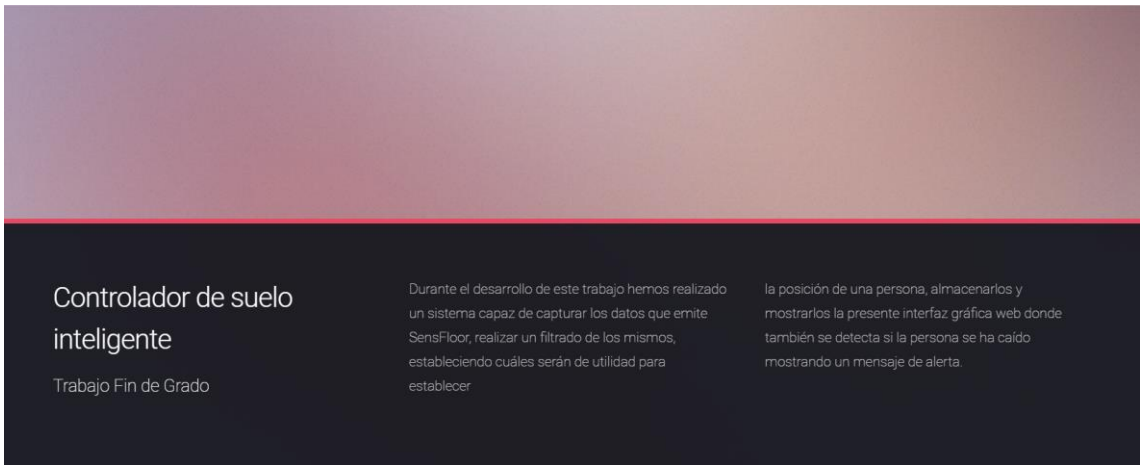


Figura 32. Aspecto final cliente. Página de inicio (II)

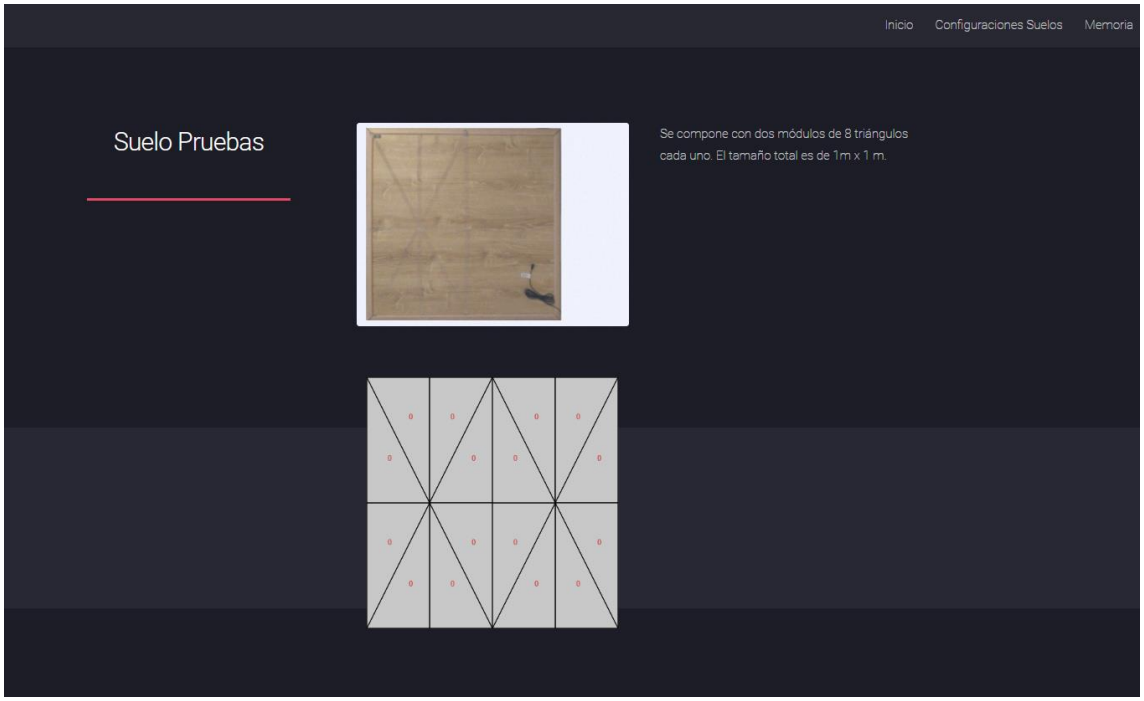


Figura 33 . Aspecto final cliente. Página de suelo de pruebas

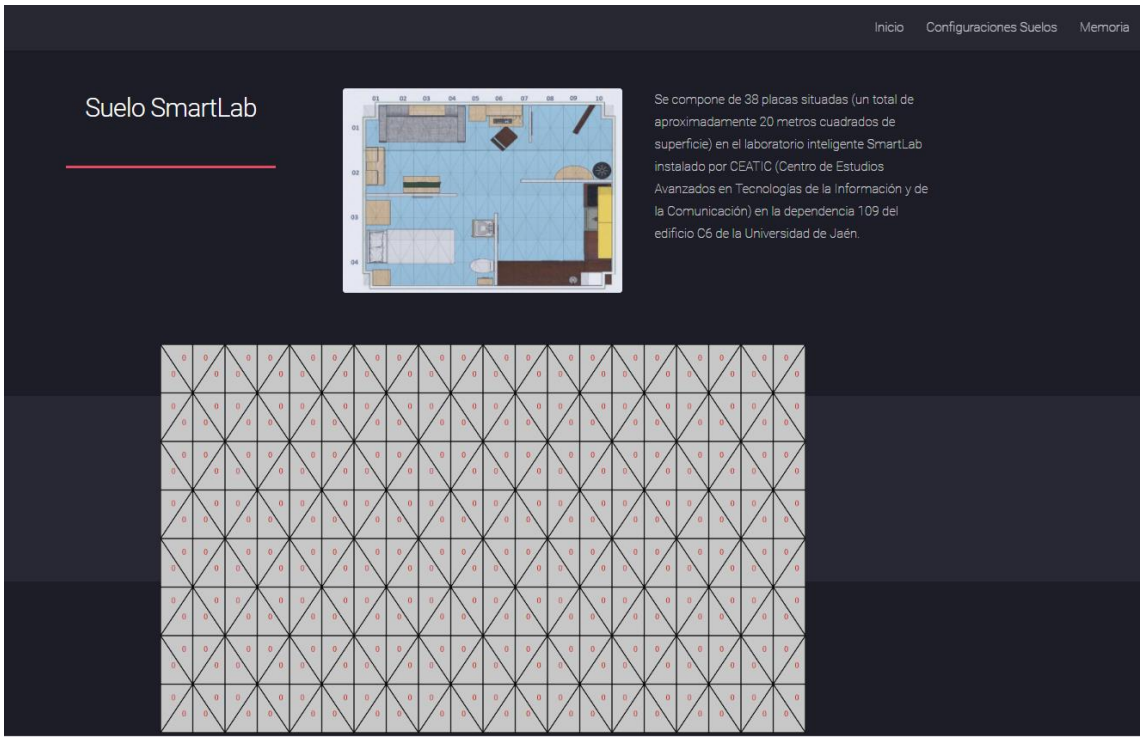


Figura 34. Aspecto final cliente. Página de suelo SmartLab

5. CONCLUSIONES

Durante el desarrollo de este trabajo se ha realizado un sistema capaz de capturar los datos que emite SensFloor, realizar un filtrado de los mismos estableciendo cuáles serán de utilidad para establecer la posición de una persona, almacenarlos y mostrarlos en una interfaz gráfica web. También se detecta si la persona se ha caído, mostrando un mensaje de alerta, así como su recorrido por la habitación.

Así pues, podemos concluir que se han cumplido los objetivos establecidos al comienzo del proyecto.

Los datos son almacenados en una base de datos MySQL con el fin de tener persistencia en caso de necesitarlos para realizar análisis posteriormente.

El proyecto no sólo es funcional, sino que se ha conseguido que sea escalable gracias al formato en el que se publican y se guardan los datos, y a que da la posibilidad de multiplicar la cantidad de habitaciones que puede manejar el sistema.

5.1. Componentes Hardware – Software

El sistema está compuesto de los elementos hardware:

- SensFloor: Dispositivo que obtiene los datos que serán mostrados posteriormente.
- Transceptor SE3-P (que deberá conectarse a un puerto USB) : recoge los mensajes y los convierte en una secuencia de datos en serie.

Como elementos software recopilamos:

- Máquina virtual 1 (servidor) : donde correrá la aplicación desarrollada en Java (SenseFloor.jar)

- Máquina virtual 2 (bróker) : donde está instalado Mosquitto (con websocket)
- Aplicación JavaScript (cliente)

5.2. Conocimientos adquiridos y utilizados

Durante la realización de este proyecto hemos tenido la oportunidad de aplicar múltiples conocimientos adquiridos durante los estudios del Grado en Ingeniería Informática.

Gracias a los conocimientos adquiridos sobre Ingeniería del Software, hemos podido aplicar un modelo de trabajo ágil que nos ha ayudado en todo el proceso de creación del sistema. Nos ha proporcionado las herramientas necesarias para llevar a cabo una planificación de las tareas eficiente, lo que ha supuesto una gran ayuda para conseguir los objetivos propuestos.

Por otro lado, trabajar con varios lenguajes de programación (sobre todo orientados a objetos) durante las diferentes asignaturas han conseguido establecer las bases necesarias no sólo para utilizarlos y ser capaces de establecer cuál de ellos nos será más beneficioso utilizar, sino también para hacernos capaces de aprender cualquier otro lenguaje y adaptarnos a él con relativa facilidad.

Con respecto a los conocimientos adquiridos, hemos tenido la oportunidad de ampliar nuestra formación en:

- Protocolos de comunicación: para este proyecto hemos aprendido dos protocolos de comunicación importantes: REST y MQTT, aunque hemos ahondado más en MQTT. Con MQTT descubrimos un protocolo latente en el “Internet de las cosas”, ligero, que permite una comunicación con el modelo publicar/subscribir reduciendo el tráfico de datos por comunicación. Así como el uso de un bróker (encargado de administrar los mensajes publicados para que alcancen a los suscriptores).

- Lenguajes de programación: en el desarrollo del trabajo, aunque principalmente hemos utilizado *Java (*lenguaje de programación que hemos utilizado ampliamente durante los estudios*)*, la aplicación del cliente la hemos implementado con *JavaScript* (con el cual apenas había trabajado hasta el momento).

- Tecnologías de almacenamiento y muestreo: este proyecto nos ha brindado la oportunidad de utilizar los conocimientos adquiridos de MySQL y ahondar en sus ventajas, además de utilizar Excel del paquete MSOffice para analizar y visualizar los datos obtenidos

5.3. Posibles mejoras y objetivos futuros

Como posibles mejoras podremos incluir:

- Creación de hilos en el servidor para dividir el proceso que en la actualidad es secuencial: tratar el mensaje recogido – guardar en la base de datos – publicar en el bróker. Estos dos últimos podrían desarrollarse en dos hilos distintos, ya que son independientes entre sí. Esto mejoraría el rendimiento del servidor y los tiempos entre la recepción del mensaje y su publicación serían menores.

- Multiusuario: en la actualidad el módulo de reconocimiento de rutas no reconoce que existan dos usuarios. Esto podría mejorarse incluyendo el factor de ‘proximidad’, de forma que no se estableciese como partes de la misma ruta dos puntos que estuvieran a una distancia mayor a un umbral establecido.

- Emisión de caídas: el sistema sólo cuenta con un mensaje de sonido si se produce una caída. Como mejora, se podría establecer una interacción con la persona que ha sufrido la caída para verificar que está consciente y si necesita ayuda médica.

6. BIBLIOGRAFÍA

[1] J. A. Ward, K. J. Anstey, P. Williams, Stephen R. Lord, Physiological factos associated with falls in older community-dwelling women, Journal of the American Geriatris Society, vol. 42 , 1994.

[2] Guillaume Pérolle, «Detector automático de caídas y monitorización de actividad para personas mayores,» *Revista española de geriatría y gerontología,* vol. 41, 2016.

[3] FSSensFloorGUI: User Manual. Axel Techmer Future-Shape.

[4] XMind, «XMind Website,» 2016. [Última consulta: 20/07/2017]. Available: https://www.xmind.net/.

[5] Smartsheet, «Smartsheet Webapp,» 2016. [Última consulta: 13/06/2017]. Available: https://app.smartsheet.com.

[6] «CCOO Servicios,» [Última consulta: 20/07/2017]. Available: https://www.ccoo-servicios.es/archivos/tic/2013-12-16_Tabla_de_equivalencias_SCP_(AEC-ANEIMO).pdf.

[7] SensFloor, *SENSFLOOR® SYSTEM . CATALOG,* 2017: Future Shape.

[8] UNIVERSIDAD DE JAÉN, «CEATIC - SmartLab,» [Última consulta: 12/08/2017]. Available: https://ceatic.ujaen.es/es/smartlab-0.

[9] P. Ruiz, «Ventajas e inconvenientes de la arquitectura cliente/servidor,» [Última consulta: 20/07/2017]. Available: http://somebooks.es/ventajas-e-inconvenientes-de-la-arquitectura-clienteservidor/.

[10] S. Bonnet, *MQTT: un protocolo específico para el internet de las cosas,* http://www.digitaldimension.solutions/es/blog-es/opinion-de-expertos/2015/02/mqtt-un-protocolo-especifico-para-el-internet-de-las-cosas/, 2015.

[11] BBVAOPEN4U, «API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos,» 2016. [Última consulta: 10/07/2017]. Available: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos.

[12] Á. Gómez Porris, V. López Monte, C. Rodríguez Colliga «Diseño de una base de datos y su motor de explotación para poder inferir usos de aplicaciones informáticas basados en comportamientos históricos (HealthPred 1.0),» 2011.

[13] P.V. Moran Morales, T.E. Delgado Bahamondes, «Implementación de un sistema para automatizar los procesos académicos y administrativos de la escuela de conducción de choferes profesionales de los ríos Manuel Bhruniss Villacres de la ciudad de Babahoyo,» UNIVERSIDAD TECNICA DE BABAHOYO FACULTAD DE ADMINISTRACIÓN FINANZAS E INFORMÁTICA Escuela de Sistema TESIS DE GRADO , 2013-14.

[14] L. Reyero Sainz y J. Peñas Jaramillo, «JLOP (JSON LANGUAGE ORIENTED PROCESSING),» Universidad Complutense de Madrid, 2011/12.

[15] «JSON.ORG,» [Última consulta: 20/07/2017]. Available: http://www.json.org/.

[16] D. Zafra Romero, Sistema para la monitorización de agentes inteligentes a través de la gegstión y acceso a servicios., 2016.

[17] proyectosagiles.org, «Proyectos ágiles,» [Última consulta: 15/08/2017]. Available: https://proyectosagiles.org/desarrollo-iterativo-incremental/.

[18] O. V. García, Física General III.

[19] P. R. Rinaldo, Guía internacional del radiafioacionado., Barcelona, 1995.

[20] Ecplise, «Mosquitto,» [Última consulta: 20/07/2017]. Available: https://mosquitto.org/.

[21] I. M. Ferreras Astorqui, «SENSOR IoT PARA MONITORIZACIÓN DE CONSUMO DE ENERGÍA CONTINUA,» FACULTAD DE INFORMÁTICA. UNIVERSIDAD COMPLUTENSE DE MADRID, 2016.

[22] J.-Y. Tigli, «Middleware for Internet of Things - MQTT,» University of Nice Sophia Antipolis, 2015/16.

[23] R. Vega, Primeros pasos con MQTT, https://ricveal.com/blog/primeros-pasos-mqtt/ , 2016.

[24] A. C. Cid, «MMO de Navegador en Tiempo Real con Node.js y WebSockets,» Facultad de Matemáticas. Universidad de Barcelona, 2014.

[25] GG1, «Six Steps to install mosquitto 1.4.2 with websockets on debian wheezy,» xappsoftware, 18 mayo 2015. [Última consulta: 18/07/2017]. Available: http://www.xappsoftware.com/wordpress/2015/05/18/six-steps-to-install-mosquitto-1-4-2-with-websockets-on-debian-wheezy/.

[26] «From text to speech,» [Última consulta: 9/08/2017]. Available: http://www.fromtexttospeech.com/.

[27] «Trazos Web. 38 plantillas HTML / CSS gratuitas,» [Última consulta: 20/07/2017]. Available: http://www.trazos-web.com/2015/02/11/38-plantillas-html-css-gratuitas/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+TrazosWeb+(Trazos+Web).

7. ANEXOS

A.1. Contenido CD-ROM

En este Anexo se mostrará la estructura del disco que contiene esta memoria de trabajo.

Aplicaciones finales

Para facilitar el futuro uso del software desarrollado, dentro de esta carpeta se encuentran las dos aplicaciones desarrolladas en todo el trabajo.

- Cliente Web: Aplicación HTML
- Servidor : fichero ‘SensFloor’ con extensión .jar

Código fuente

Dentro de esta carpeta se encuentran la aplicación servidora desarrollada en este trabajo. Se abre con el IDE de programación NetBeans 7.5 o superior, siendo necesarias las bibliotecas:

- RXTXcomm.jar
- Org.eclipse.paho.client.mqtt3-1.0.1.jar
- JDK 1.8

Memoria

Documento con extensión .pdf con la documentación de este trabajo.

B.2. Manual de Instalación

En el presente Manual de Instalación se describe cómo desplegar todo el sistema desarrollado. Al tratarse de un sistema con una arquitectura cliente-servidor, se describen los pasos para la instalación de estas dos partes.

B.2.1. Servidor

La aplicación desarrollada en este trabajo ya dispone del siguiente software instalado y configurado en dos servidores cedidos por el CEATIC:

- Base de datos MySQL 5.5
- Eclipse Mosquitto (versión 1.4.2)

Por lo que, para ejecutar el servidor no sería necesario instalarlos ya que se conecta automáticamente a estos. Sin embargo, sí será necesario instalar Java Runtime Environment (JRE) para su ejecución.

También será necesario conectar el transceptor SE3-P al servidor y configurar el puerto COM al que se conecte modificando el fichero 'puerto.txt' (por defecto COM4).

Una vez comprobados estos requisitos bastará con ejecutar el fichero SensFloor.jar.

B.2.2. Cliente

El despliegue de la parte del cliente es simple. Basta con copiar la carpeta del CD donde se quiera instalar y abrir en un navegador el fichero index.html. Es necesario tener activado el soporte de JavaScript en el navegador web.

C.3. Manual de usuario

La página inicial cuenta con un menú (Figura 35, esquina superior derecha) que incluye tres opciones:

- **Inicio** : enlace a la página principal
- **Configuraciones Suelos:** que incluye un submenú donde podremos elegir si visualizar la configuración para el suelo de prueba o para el situado en el laboratorio SmartLab.
- **Memoria:** enlace a la documentación del trabajo.

Figura 35. Inicio

C.3.1. Suelo Pruebas

Si accedemos a ‘Suelo Pruebas’, visualizamos una imagen real del suelo de pruebas y bajo esta, una simulación de éste (Figura 36). En el centro de cada triángulo aparece un número que se corresponderá con la variación de capacitancia que está sufriendo en ese momento el suelo. Si alguien pisa sobre uno de ellos se detectará como ‘activo’ el triángulo con el que se corresponda en el esquema y su color cambiará de gris a verde. Una línea aparecerá siguiendo

la ruta que una los triángulos que se vayan pisando sucesivamente. En el caso de que más de 5 triángulos cercanos entre sí se señalen como ‘activos’ (de verde) la aplicación detectará que se ha producido una caída y se emitirá un mensaje de audio avisando del evento.

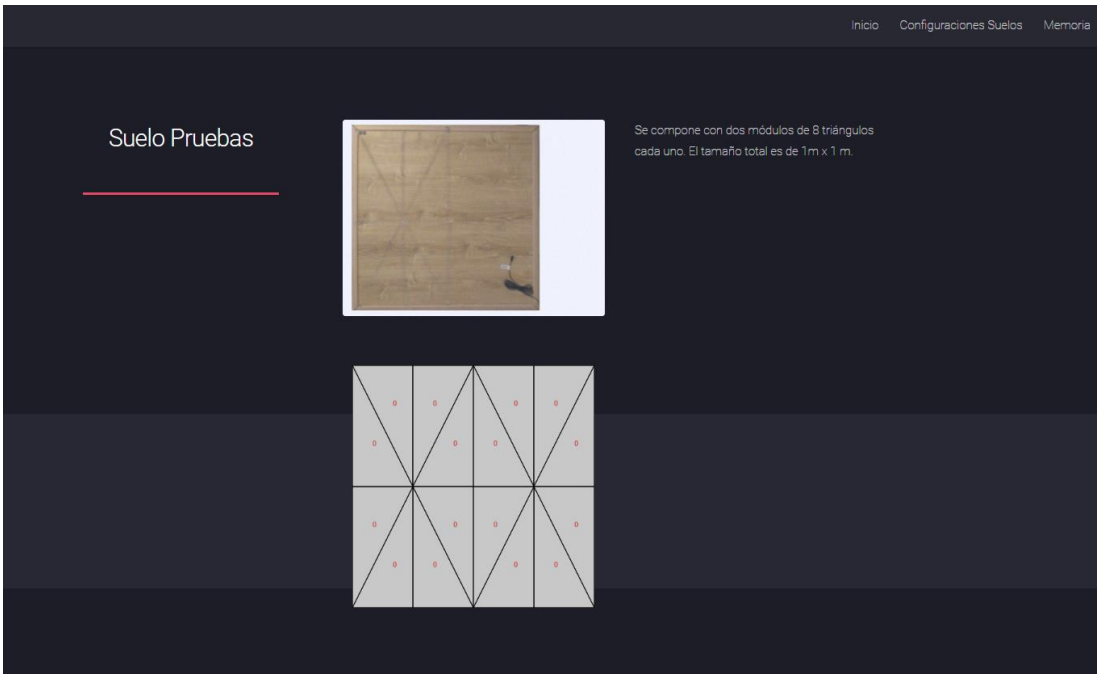


Figura 36. Suelo Pruebas

C.3.2. Suelo SmartLab

Si accedemos a ‘Suelo SmartLab’ visualizamos una imagen del plano del laboratorio inteligente *SmartLab* instalado por el CEATIC (Centro de Estudios Avanzados en Tecnologías de la Información y de la Comunicación) en la dependencia 109 del edificio C6 de la Universidad de Jaén (Figura 37). El plano está dividido de acuerdo a un sistema de coordenadas que se corresponde con los módulos SensFloor instalados. Utilizamos éstas para la simulación del suelo que se sitúa justo debajo de la imagen del plano.

Al igual que para el Suelo de Pruebas, en el centro de cada triángulo aparece un número que se corresponderá con la variación de capacitancia que está sufriendo en ese momento el suelo. Si alguien pisa sobre uno de ellos se detectará como ‘activo’ el triángulo con el que se corresponda en el esquema y

su color cambiará de gris a verde. Una línea aparecerá siguiendo la ruta que una los triángulos que se vayan pisando sucesivamente. En el caso de que más de 5 triángulos cercanos entre sí se señalen como ‘activos’ (de verde), la aplicación detectará que se ha producido una caída y se emitirá un mensaje de audio avisando del evento.

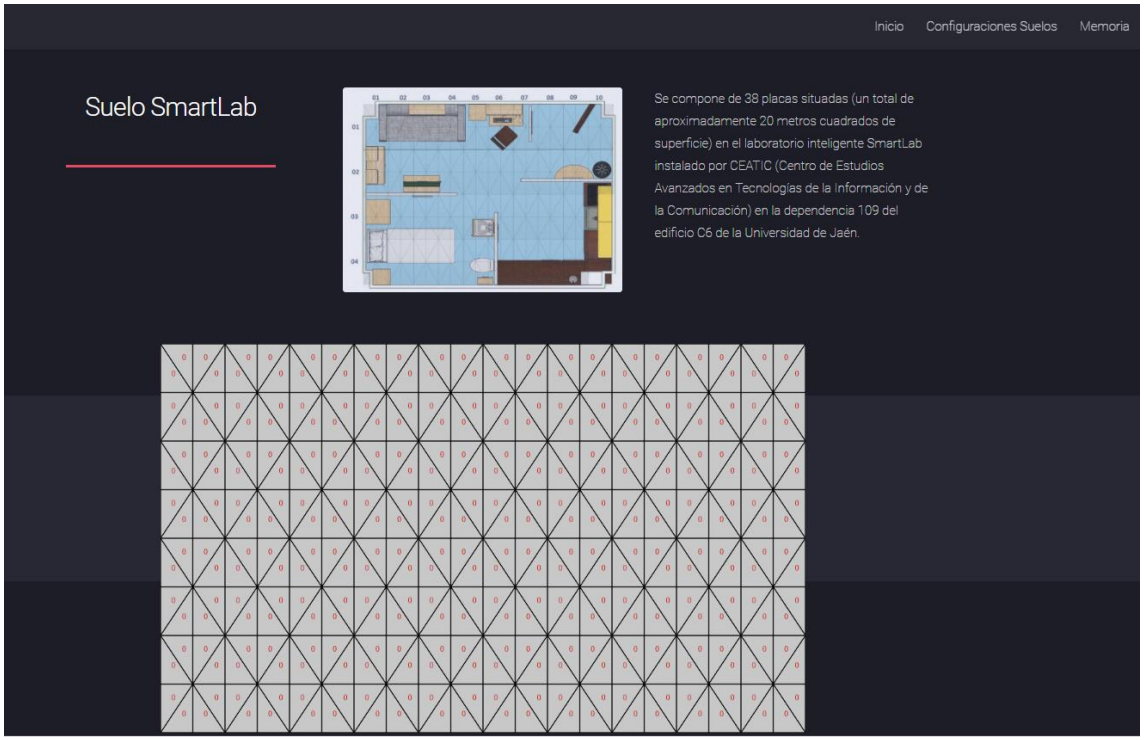


Figura 37. Suelo SmartLab