MEMORIA DEL PROYECTO

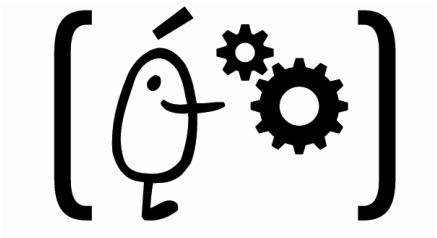
DE

FIN DE CARRERA

Desarrollo y evaluación de un sistema tipo mesa interactiva basado en tecnología de pantalla "Multi-Táctil"

INGENIERÍA INFORMÁTICA

ESCUELA TECNICA SUPERIOR DE INGENIERÍA



Curso 2008-2009

Alumno

Fernando Manuel Sánchez Monserrat

Tutor

Pedro Morillo Tena

Agradecimientos

A mis padres. Gracias a ellos he llegado a ser lo que soy hoy y a tener lo que hoy tengo. Gracias también por vuestro apoyo incondicional, sin vosotros todo esto hubiera prácticamente imposible. Siempre habéis querido lo mejor para mí y me habéis ayudado a que lo consiguiera.

A mis abuelos. Habéis sido una pieza fundamental en mi vida y lo seguiréis siendo siempre. Vosotros estuvisteis ahí cuando mi personalidad y mi cabeza se estaban definiendo. Si soy como soy, en gran parte es gracias a vosotros.

A mis tíos. Siempre habéis sido como mis otros padres, esos a los que se les cuentan las cosas que a los padres no se les pueden contar. ¡Jejeje!. Sé que nunca en la vida, pase lo que pase, me vais a fallar y sé que siempre me habéis ayudado en todo lo que habéis podido, incluso sin yo pedíroslo que es lo que más vale. Muchas gracias.

A mi novia Alicia. Muchísimas gracias por apoyarme siempre en todo lo que me he propuesto, por aguantar mis malos días con una sonrisa intentando que se me pase cuanto antes y por ayudarme y animarme en los momentos que más lo necesitaba. Gracias por quererme tanto pequeñaja.

A mi tutor Don Pedro Morillo Tena. Gracias a ti se me brindó la oportunidad de realizar este proyecto, gracias al cual he podido conocer áreas de la informática que me han servido para cambiar mi pensamiento sobre la misma. Gracias por toda tu ayuda (que sería de mi memoria sin ella, XD) y paciencia.

A mis compañeros del Instituto de Robótica. Todos en mayor o menor medida me habéis ayudado cuando lo he necesitado. Muchas gracias a todos por ello. Pero por encima de todos tengo que destacar a uno. Jesús, sin tu ayuda este proyecto hubiera sido interminable para mí. Muchas gracias por tu ayuda, muchas gracias por estar ahí siempre que me surgía algún problema, muchas gracias por todo.

Y por último pero no por ello menos importante, a LOS DE SIEMPRE. Vosotros sabéis a quienes me refiero. Los días interminables de biblio y los ataques de risa, los nervios de los exámenes, el pitillo de la victoria, las fiestas de después de los exámenes… Vosotros habéis sido con quien me he desahogado contando mis penas, mis rallazos, mis alegrías… Muchas gracias a todos, de corazón, sois muy importantes para mí.

Índice

Índice de figuras

Índice de Tablas

0. Resumen

El propósito de este trabajo consiste en el análisis, diseño, construcción y evaluación de un sistema tipo mesa interactiva basado en tecnología de pantalla “Multi-Táctil”. Concretamente, este trabajo se centra tanto en el diseño y construcción del subsistema hardware del sistema (la propia mesa en sí), como en el análisis, diseño e implementación del subsistema software del mismo. Además de esto, se crearan algunas aplicaciones de prueba donde se comprobará si el sistema desarrollado realmente funciona. Estas aplicaciones comprobarán el funcionamiento tanto en Single-Touch como en Multi-Táctil.

Esta tecnología está siendo utilizada cada vez más por industrias debido a que se están produciendo grandes avances en ella, lo que le permite ser un perfecto complemento a los demás periféricos (ratón, teclado, botones, etc..) en los casos donde estos no cumplen con las expectativas que en un principio se espera de ellos. Todo esto hace que cada vez sea más aconsejable su implantación en sistemas de uso cotidiano.

Como se ha comentado en el primer párrafo la realización de este proyecto se dividirá durante todo el tiempo que dure el desarrollo del mismo en dos subsistemas:

- el **subsistema hardware**, que se dividirá en 3 subsistemas: sistema de captura, sistema de visualización y esqueleto físico del sistema. En esta parte se efectuará un estudio sobre las diferentes tecnologías existentes para este tipo de sistemas, explorando las capacidades y planteando las ventajas/desventajas de cada una de ellas. Finalmente y después de la evaluación de todas ellas se escoge la tecnología que consideramos más adecuada para cumplir los requisitos exigidos por los futuros clientes. A partir de ese momento comenzará la tarea de seleccionar los componentes más adecuados para su correcto funcionamiento teniendo en cuenta los requisitos exigidos por los clientes.

- el **subsistema software**, donde se realizará el estudio y evaluación de las diferentes posibilidades existentes a la hora de la codificación de la librería donde se realizará la captura y el manejo de los puntos de contacto realizados en el subsistema anterior. También se decidirá que tipo de aplicaciones de prueba se realizarán y, como en el caso anterior, la realización de las mismas se someterán a estudio antes de realizar su implementación. Estas aplicaciones deben estar relacionadas con el trabajo futuro para el que va a ser utilizado el sistema.

1. Introducción

Desde los comienzos de la informática, esta ha estado en constante evolución intentando siempre encontrar mejoras que pudieran ser utilizadas tanto en ámbitos particulares, como en los empresariales. En este caso se va a tratar el entorno empresarial.

Estos avances tienen como finalidad facilitar el uso de las aplicaciones, las cuales mediante la tecnología existente no poseen una utilización intuitiva y eficiente. Esto desemboca en una pérdida del interés en la utilización de las aplicaciones y por consiguiente, una pérdida de rendimiento del empleado provocando un daño en la economía de la empresa.

Primeramente y antes de comenzar con la introducción a nuestro sistema en concreto se debe realizar una pequeña introducción al significado de la tecnología Multi-Touch en sí. Ya que hoy en día y después de 25 años de desarrollo sigue existiendo muy poca información o experiencia y por lo tanto una gran confusión sobre la misma.

Multi-Touch es el nombre con que se conoce a una técnica de interacción hombre-máquina y al hardware que la implementa. El "Multi-Touch" (del inglés *múltiple tacto*) consiste en una pantalla táctil o touchpad que reconoce simultáneamente múltiples puntos de contacto, así como el software asociado a esta que permite interpretar dichas interacciones simultáneas. Esta interacción se puede realizar mediante distintos diseños como se observa en la tesis de L. Y. L. Muller [17].

Después de la definición realizada en el párrafo anterior, a continuación se van a comentar tres puntos para finalizar la introducción al mundo de la tecnología Multi-Touch.

1. Primeramente cabe recordar que tuvieron que pasar 30 años desde que el ratón fue inventado por Engelbart y English en 1965 hasta que se hizo omnipresente en la publicación de Windows 95. Este fue publicado comercialmente en 1982 en las estaciones de trabajo de Xerox Star y PerQ. Pero, estadísticamente esto no es relevante, como hemos dicho antes, tuvieron que pasar 30 años para que alcanzara su punto álgido. Así que, tomando esto como referencia, las tecnologías Multi-Touch tienen todavía 5 años por delante antes de quedar en el olvido.

2. El truco para saber si la tecnología MultiTouch "es lo mejor para esto, o lo peor" es preguntarse, que, para que, cuando, para quien, donde y lo más importante, ¿por qué? Si lo que se busca es reemplazar el ratón, no tiene sentido seguir. El ratón es bueno para muchas cosas, pero no lo es para todo. Es en esas cosas (donde el ratón no están bueno o está en desuso) donde debemos encontrar técnicas que lo complementen o lo sustituyan.

3. Para realizar una mejora significativa de un producto/tecnología, probablemente se produzca un coste bastante grande, tanto económico, como de tiempo y esfuerzo. Se debe conocer cuál es el estado de la tecnología y si está abierta a mejoras por la que merezca la pena

realizar ese gasto. En este punto se ve que las entradas de la tecnología MultiTouch son todavía bastante primitivas. Por lo tanto es factible seguir mirando este tema.

El proyecto que se va a elaborar intenta, mediante la tecnología Multi-Touch simplificar y mejorar el uso de aplicaciones en las que los periféricos existentes en la actualidad no cumplen con las expectativas que de ellos se espera o simplemente no es posible su utilización mediante los periféricos actuales. Un ejemplo podría ser una aplicación donde el trabajo cooperativo de dos o más empleados reduzca el tiempo de finalización considerablemente. Cabe destacar que la finalidad de este periférico no es sustituir a los periféricos ya existentes sino que intenta ser un complemento de los mismos como se comenta en el punto 2 anterior.

Para poder lograr esto, obviamente, es necesario proveer tanto un sistema físico que se encargue de la captura de los puntos y la visualización de los mismos en la pantalla, como otro lógico, el cual se encarga de realizar el tratamiento correspondiente a los puntos captados por el subsistema hardware para su utilización en la aplicación que en ese momento se esté manejando [10].

Esto se realizará mediante la tecnología de infrarrojos. Más adelante, se comentará cuál es su principio básico de funcionamiento, cuales son los componentes necesarios para su desarrollo y lo más importante, porque esta y no otra es la tecnología escogida para realizar el proyecto.

La tecnología Mutli-Touch es una tecnología en desarrollo y para la que se están realizando avances continuamente, lo que influye directamente en el aumento de sus capacidades. Todo esto hace que la tecnología este siendo adoptada cada vez por más industrias que observan que su implantación puede reportarles grandes beneficios a corto plazo. Estos beneficios puede que no sean directamente económicos, pero pueden ser un incremento de la productividad lo que finalmente producirá un beneficio económico para la empresa.

Para comprobar el correcto funcionamiento del sistema se han llevado a cabo una serie de pruebas funcionales consistentes en, primero, evaluar la funcionalidad de cada subsistema por separado (sistema de captura, de visualización, físico y de software) y segundo, poner en marcha el sistema completo realizando ejercicios similares a los que deberá realizar cuando sea implantado en las empresas interesadas.

Por último, se han calculado los costes totales del desarrollo del sistema y se ha hecho un supuesto de venta a una empresa. Se pone de manifiesto que poco más de 11000 euros que cuesta el sistema, son una inversión que se pueden permitir las empresas interesadas en el desarrollo del mismo.

1.1 Motivaciones

La tecnología Multi-Touch está cada día más involucrada en la sociedad actual y aunque es una tecnología “nueva” promete revolucionar la vida de las personas en un futuro no muy lejano. Cada vez están más a la orden del día los móviles con pantalla táctil, esto parecía impensable que pudiera ocurrir hace tan sólo un par de años donde la mayoría de la sociedad ni siquiera conocía de la existencia de esta tecnología fuera de la ficción. Por otro lado, cada vez son más las empresas que se atreven a comercializar ordenadores con pantalla táctil aunque estos todavía tienen precios prohibitivos en la mayoría de los casos. Estos cambios en la sociedad hacen que se deba estar preparado para dicho cambio, pudiendo así explotar al máximo las capacidades que tendrá la tecnología “Multi-Touch” en los próximos años.

Por otro lado, estamos en una sociedad en la que los sistemas de información están presentes en cualquier campo que se pueda pensar. Por tanto, es importante conocer el concepto de sistemas de la información, así como las diferentes formas de implementarlos que se pueden encontrar hoy en día.

Actualmente estos sistemas están evolucionando hacia una arquitectura en que se proporciona un servicio que se está alimentando de datos proporcionados por terceras aplicaciones que se ejecutan local o remotamente. La creación de una aplicación que utilice este tipo de arquitectura permitirá que sea utilizada por infinitas aplicaciones utilizando esta como servidor de puntos y eventos.

Si se unen estos conceptos con una premisa fundamental en los objetivos de una empresa, como es la de maximizar los beneficios (en este caso mejorando el rendimiento de los empleados, aumentando la productividad, etc) se obtiene como resultado el ámbito de la aplicación del sistema a desarrollar.

Actualmente y como se ha comentado existen empresas que están comenzando a comercializar productos utilizando la tecnología “Touch” pero tienen dos problemas fundamentales:

La mayoría de estos productos no poseen soporte “Multi-Touch” únicamente “Single-Touch”. Lo que significa que solo puede reconocer y manejar un punto de contacto imposibilitando el trabajo cooperativo.

Los sistemas “Multi-Touch” tienen un precio prohibitivo para las pequeñas y medianas empresas.

La motivación por la cual se ha decidido desarrollar un sistema de este tipo a sido básicamente la de intentar solucionar los puntos anteriores. Conseguir realizar un sistema “Multi-Touch” a un precio asequible para las pequeñas y medianas empresas que puedan estar interesadas en adquirirlo.

De todo esto se deriva el presente proyecto: **un sistema de captura y reconocimiento “Multi-Touch”**.

1.2 Objetivos

El objetivo final de este Proyecto Final de Carrera es el análisis, diseño e implementación de un sistema de un sistema “Multi-Touch”. Concretamente un sistema basado en la detección de puntos de contacto en una superficie mediante una cámara para su posterior procesamiento y utilización en distintas aplicaciones mediante el paso de información utilizando una arquitectura Cliente-Servidor.

Para poder conseguir esta meta, los objetivos se han desglosado en seis subobjetivos:

- Realizar un estudio de las alternativas existentes actualmente en el campo de los sistemas “Touch”, seleccionando justificadamente la más adecuada para las aplicaciones en las que se va a utilizar el sistema.

- Desarrollar una aplicación web para poder realizar la configuración de la cámara que se encargará de la detección de los puntos. Esta configuración variará dependiendo del entorno en el que se esté trabajando y la aplicación con la que se esté haciendo.

- Desarrollar una librería que se encargue de la manipulación de los puntos obtenidos por la cámara para su posterior utilización en otras aplicaciones remotas o locales.

  En este punto se puede destacar dos subapartado:

  - Realizar la codificación de una arquitectura de paso de información mediante una arquitectura Cliente-Servidor para poder transmitir los puntos y eventos manipulados por la librería a aplicaciones remotas.

  - Realizar la codificación de los eventos del ratón y del reconocimiento de gestos.

- Desarrollar aplicaciones de prueba del sistema. Estas aplicaciones servirán para demostrar que el sistema funciona correctamente en los ámbitos para los que ha sido desarrollado.

- Desarrollar un sistema físico del proyecto. Este sistema estará compuesto de tres subsistemas:

    - Sistema de captura: este sistema se encargará de capturar los puntos de contacto realizados en la pantalla

    - Sistema de visualización: este se encargará de mostrar la imagen en la pantalla

    - Sistema físico: este es el soporte físico del sistema.

  Estos tres puntos serán comentados con más detalle durante la documentación del proyecto.

- Realizar un estudio sobre la viabilidad técnica y económica de un sistema que cumpla dichas características sobre un caso típico de uso como son las posibles empresas interesadas en el mismo.

2. Estado del Arte

A través de este apartado, lo que se pretende es realizar una breve introducción del contexto que rodea al Proyecto Fin de Carrera que se va a desarrollar, con el objetivo de, describir y comparar el estado actual de las diferentes tecnologías relacionadas con este proyecto, y describir cualquier otro aspecto importante para el mismo.

La información que aparece en los siguientes puntos la he obtenido a partir de un proceso de documentación y posterior investigación sobre el área específica del proyecto.

En las primeras fases del desarrollo del proyecto se tuvieron que tomar numerosas decisiones en cuanto a la tecnología a utilizar, tanto en el subsistema hardware como en el software. En el campo específico de este proyecto existen muchas opciones, unas más validas que otras, para llegar al objetivo final. Por todo esto, he considerado de gran importancia este punto, ya que me ha dado una visión general de la evolución y el estado actual de diversas tecnologías y además servirá para justificar las decisiones que he ido tomando.

A continuación se va a realizar una explicación sobre las diferencias más importantes que se han encontrado y que pueden provocar confusión en personas no introducidas en el mundo de la tecnología Multi-Touch:

- **Touch-Tablets vs Touch-Screen:** Básicamente, esto son las 2 partes opuestas de la misma cosa. Si tu dedo (o lápiz) toca directamente el objeto con el que se desea interactuar estaremos hablando de una Touch-Screen, si por el contrario se toca una superficie que no está en la pantalla será una Touch-Tablet, como se puede observar en el paper de S.K. Lee and K.C. Smith [2]. Un ejemplo gráfico de esta diferencia se puede observar en la “Figura 1”.

Figura 1. Diferencia entre una Touch-Tablet Vs una Touch-Screen

- **Continuo vs Discreto:** La naturaleza de la interacción con las entradas “MultiTouch” es completamente dependiente de la naturaleza de las acciones discretas vs continuas soportadas. Algunas interfaces de Touch-Screens convencionales estás basadas en ítems discretos tales como los de presión también llamados “light buttons”.

  Un ejemplo de una interfaz Multi-Touch con acciones discretas sería utilizar un teclado “QWERTY”, donde un dedo mantiene la tecla shift y otro aprieta la tecla mayúscula que se desea mostrar. Un ejemplo de 2 dedos generando un movimiento continuo sería cuando los extendemos opuestamente en la diagonal de un rectángulo (“Figura 2”). Entre los 2 tenemos una situación continua/discreta tal como la emulación de un ratón, usando un dedo para indicar la posición continua y los otros (cuando se produzca el contacto) indican el click del ratón (por ejemplo).

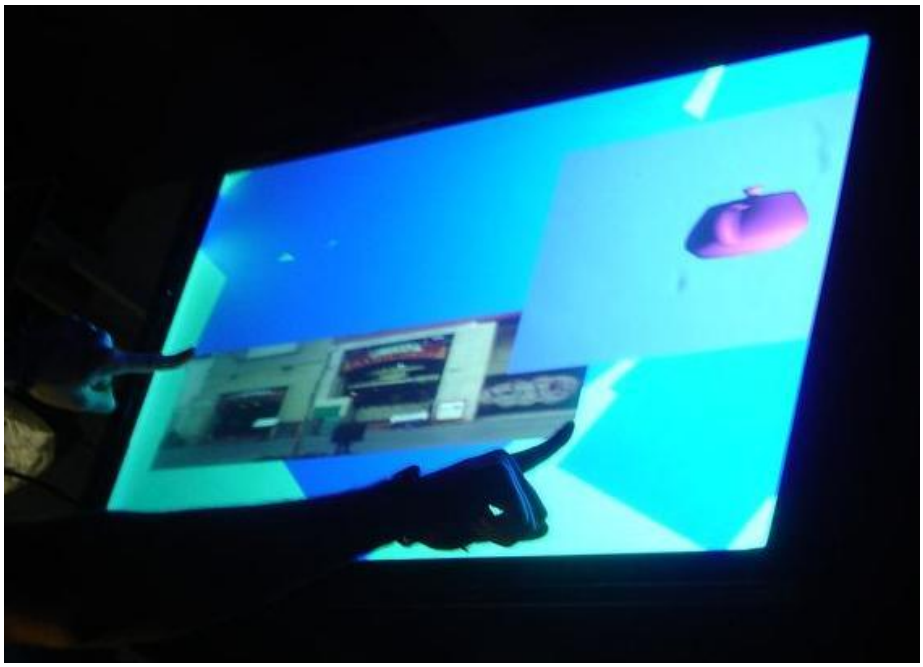


Figura 2. Ejemplo de una interacción continua

- **Grados de libertad:** La calidad de la interacción está ampliamente relacionada con los grados de libertad, concretamente, los grados de libertad continuos soportados por la tecnología. La GUI está en gran parte basada en el movimiento de un cursor 2D (por ejemplo el ratón). Esto significa que posee 2 grados de libertad. En cambio si se detecta la posición de 2 dedos tenemos 4DOF.

  Usadas apropiadamente estas tecnologías ofrecen el potencial para empezar a recoger el tipo de entradas que encontramos en nuestro día a día, y esto se realiza aprovechando las habilidades que se han ido adquiriendo durante la vida. Este punto y el anterior están muy relacionados.

- **El tamaño importa:** El tamaño en gran parte determina que grupos de músculos usamos, cuantos dedos/manos pueden ser activos en la superficie y que tipos de gestos son apropiados para el dispositivo. Por lo tanto queda claro que cuanto mayor sea el tamaño de la pantalla mayor será el número de músculos necesarios para su utilización y a su vez, mayor será el número de dedos/manos activos.

- **Importancia de la orientación vertical vs horizontal:** Las superficies táctiles grandes tienen tradicionalmente un problema ya que solo pueden detectar un punto de contacto. Así que si se apoya la mano en la superficie y también el dedo con el que se quiere señalar, estamos confundiendo a la pantalla. Esto no ocurre en las superficies montadas verticalmente.

  Por lo tanto grandes pizarras electrónicas usan tecnologías de detección de un solo punto sin problemas.

- **Detección de puntos es mucho más que contacto y posición:** Históricamente, las pantallas táctiles solo nos decían que la pantalla había sido tocada y donde. Esto es cierto para dispositivos “Single-Touch” y “Multi-Touch”. Sin embargo existen ciertos aspectos que deben ser aprovechados en algunos sistemas y tienen potencial para mejorar la experiencia del usuario:

  1. **Grado de contacto/presión:** Una pantalla táctil que puede detectar independiente y continuamente los grados de contacto para cada punto tiene mucho más potencial para realizar una gran interacción. Notar que se utiliza grados de contacto en lugar de presión, ya que, lo que entendemos por presión nos podría dar lugar a efectos colaterales (cuanto más fuerte presiones la marca del dedo se extenderá alrededor del punto de contacto, por lo tanto, lo que se detecta actualmente es el área de contacto no la presión realizada).

     Cualquiera de las 2 es mejor que el “binario contacto/no contacto”, pero existen sutiles diferencias entre ellos.

  2. **Ángulo de aproximación:** Bastantes sistemas han demostrado la capacidad de sensorizar el ángulo entre la pantalla y el dedo (por ejemplo McAvinney´s Sensor Frame). Esto proporciona al dedo la capacidad de funcionar más o menos como un joystick virtual en el punto de contacto. Con esto también se puede conseguir especificar un vector que puede ser proyectado en el mundo virtual 3D detrás de la pantalla en el punto de contacto. Algo que podría ser relevante en juegos o aplicaciones 3D.

  3. **Vectores de fuerza:** A diferencia del ratón, una vez en contacto con la pantalla el usuario puede aprovechar la fricción entre el dedo y la pantalla para conseguir varios vectores de fuerza. Por ejemplo, sin mover el dedo, se puede aplicar un vector de fuerza a lo largo de cualquier vector paralelo a la superficie

de la pantalla incluyendo uno rotacional. Estas técnicas fueron descritas en 1978 por Herot, C. & Weinzapfel, G [22] y de nuevo cinco años después por Minsky [23].

Es importante recordar, después de todo lo dicho en los 3 puntos anteriores, que es la capacidad humana, no la tecnología, en la que nos debemos centrar y tener en cuenta en nuestras consideraciones. Mientras se descubran nuevas capacidades accesibles a un coste razonable será un desafío el conseguir realizarlas, es erróneo pensar que cualquier cosa esta dicha sobre "Multi-Touch".

- **El tamaño importa II:** La habilidad de detectar el tamaño del área del punto que está en contacto cola superficie puede llegar a ser tan importante como el tamaño de la superficie. En el ejemplo de Synaptics, vemos que el dispositivo puede detectar la diferencia entre unos puntos pequeños (dedos) y unos grandes (mejilla), por lo tanto, se podrá contestar al teléfono colocándotelo directamente en la mejilla. Esto, aunque a simple vista parece una obviedad, no es siempre así. En la actualidad muchas de las pantallas táctiles no diferencian el contacto entre dos puntos de diferentes tamaños.

- **Single-finger vs Multi-finger:** A partir de que las pantallas táctiles fueron desarrolladas a finales de 1982, la gran mayoría de estas son "single touch", esto quiere decir que solo pueden trabajar/manipular un punto (como se hace con un ratón, joystick, etc…) restringiendo así los gestos que se pueden realizar. Estamos dotados de múltiples miembros por una razón, sacar el máximo provecho de ellos.

- **Multi-point vs Multi-touch:** Es realmente importante pensar si los tipos de gestos y técnicas interactivas utilizadas son características a la tecnología o no. Muchas, sino la mayoría de las llamadas tecnologías "multi-touch", son realmente "multi-point". Para saber diferenciar una de la otra basta con comentar un breve ejemplo: no se dice que se estén utilizando técnicas diferentes solo porque se esté utilizando el laptop del portátil en lugar del ratón.

- **Multi-hand vs Multi-finger:** En la mayoría de estos tipos de sistemas, el control no solo puede provenir de diferentes dispositivos o diferentes dedos. Una gran cantidad de este control depende de la escala del dispositivo de entrada. Un pequeño ejemplo sobre esto podría ser, en referencia a la tradicional GUI, se puede seleccionar un icono con el ratón, hacer click con el botón izquierdo del mismo y arrastrar. O se puede apuntar con el puntero del ratón y hacer click con un pedal. Esto es la misma técnica de arrastre pero se divide en dos partes y dos dispositivos.

- **Multi-person vs Multi-touch:** Si 2 puntos están siendo detectados, existe una gran diferencia si estos son dados de la misma mano de un usuario o de un dedo de la mano izquierda de 2 usuarios diferentes. Con la mayoría de las tecnologías "Multi-Touch" no se requieren 2 cursores. Pero si son 2 personas trabajando en la misma superficie, como se puede observar en la "Figura 3", esto puede ser exactamente lo que queremos. Y en la medida en que las

tecnologías “Multi-Touch” se ocupen de esto, podría ser de gran valor sensorizar de donde viene la persona que ha tocado la superficie. Como puede hacerse en el sistema Diamond Touch de MERL (foto abajo) y en el cual se puede profundizar en [8].

Figura 3. Ejemplo de Multi-Person

- **Puntos vs Gestos:** La mayoría de los primeros trabajos relevantes, como el de Krueger tienen que ver con detectar la posición (y la dinámica) de la mano, así como, la orientación. Eso significa que esto va muchos más lejos que la simple tarea de detectar múltiples puntos.

- **Pen vs Dedo:** Existen grupos de gente que dice que se debe tomar la decisión de utilizar el “pen” o el dedo. Un ejemplo de cada una de las dos opciones se puede observar en la “Figura 4”. Es cierto que en muchos sistemas que utilizan el “pen” no funcionan muy bien con el dedo, pero cada vez más, muchos “sensores touch” funcionan igualmente con un “pen” y el dedo. No debería tener que ser una pregunta el elegir uno u otro, aunque por temas de diseño es posible que convenga escoger uno. Cada uno tiene sus propias ventajas y desventajas [11].

  Solo hay que tener en cuenta que: si el dedo es el mejor dispositivo, ¿Por qué Picasso? No se restringió a pintar con el dedo. Y por otro lado, si quieres medir la temperatura del agua ¿Qué utilizarías el dedo o un lápiz?

Figura 4. Diferencia entre "Pen" vs dedo

- **Manos y Dedos vs Objetos:** El “pen” es solo un objeto que podría ser utilizado para una interacción multipunto. Algunos sistemas multi-point/multi-touch pueden no solo sensorizar varios objetos diferentes sobre ellos, sino que también pueden averiguar que objeto es, donde está y cuál es su orientación. Ver el trabajo de N. Motamedi, donde los objetos pueden ser o no utilizados simultáneamente con los dedos [15].

- **Different *vs* The Same: ¿**Cuando algo es lo mismo, diferente o evidente? Esto depende de si usted es un usuario, programador, científico o abogado. Desde el punto de vista del usuario, se pueden comentar tres puntos que serán conocidos y asumidos por cualquier persona experta en la materia:

    1. ***Device-Independent Graphics:*** Esto indica que la misma técnica implementada con un dispositivo de entrada alternativa sigue siendo la misma técnica. Por ejemplo, se puede trabajar la GUI con una pantalla táctil, ratón, joystick, touchpad, … y se considera acciones como el doble-click, arrastrar, etc como “la misma” técnica.

    2. ***The Interchange of devices is not neutral from the perspective of the user:*** Aunque se pase de utilizar el GUI con el ratón a utilizarse con el tuchpad, el usuario considera que está utilizando la misma técnica. Pero, los distintos dispositivos tienen sus propias cualidades y defectos. Así que, mientras el usuario considera las diferentes técnicas iguales, su rendimiento varia (velocidad, precisión, confort, etc) de dispositivo a dispositivo. También se puede comentar que la experiencia de uso de cada dispositivo cambia, aunque se utilice la misma técnica, dejando como

consecuencia que los usuarios e investigadores no suelen intentar cambiar de un dispositivo a otro para controlar la misma técnica.

A continuación introduciremos los diferentes tipos de hardware utilizados para la realización de este tipo de sistemas. Más tarde comentaremos los diferentes tipos de software utilizados en sistemas similares al nuestro.

Para finalizar comentaremos un último apartado en el que desarrollaremos los “gestos” más comunes utilizados en pantallas táctiles, entendiendo por gestos predefinidos en el sistema movimientos de los dedos sobre la pantalla, cuya ejecución dará como resultado una acción específica.

2.1 Hardware

Denotaremos por hardware todas las partes físicas y tangibles del sistema: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos (sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado).

El conjunto de todo esto nos dará como resultado diferentes tecnologías, las cuales buscan fines semejantes entre ellas. A continuación definiremos las más relevantes explicando cual es su funcionamiento y sus componentes más significativos:

2.1.1 Resistiva

Una pantalla táctil resistiva [12] está formada por varias capas. Las más importantes son dos finas capas de material conductor entre las cuales hay una pequeña separación. Cuando algún objeto toca la superficie de la capa exterior, las dos capas conductoras entran en contacto en un punto concreto. Un ejemplo gráfico de lo anterior se puede observar en la “Figura 5”. De esta forma se produce un cambio en la corriente eléctrica que permite a un controlador calcular la posición del punto en el que se ha tocado la pantalla midiendo la resistencia. Algunas pantallas pueden medir, aparte de las coordenadas del contacto, la presión que se ha ejercido sobre la misma.

Las pantallas táctiles resistivas son por norma general más asequibles pero tienen una pérdida de aproximadamente el 25% del brillo debido a las múltiples capas necesarias. Otro inconveniente que tienen es que pueden ser dañadas por objetos afilados. Por el contrario no se ven

afectadas por elementos externos como polvo o agua, razón por la que son el tipo de pantallas táctiles más usado en la actualidad.

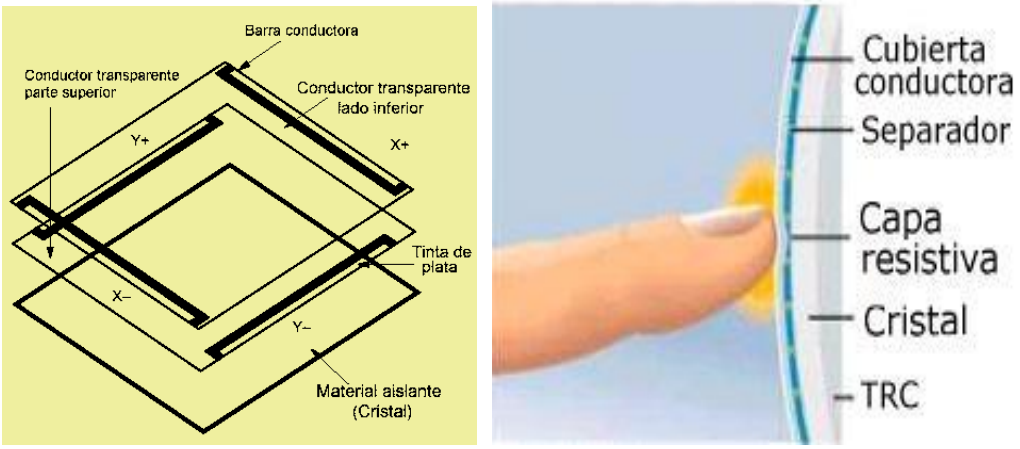


Figura 5. Estructura de capas de una pantalla resistiva

2.1.2 De Onda Acústica Superficial

La tecnología de onda acústica superficial (denotada a menudo por las siglas *SAW*, del inglés *Surface Acoustic Wave*)[13] utiliza ondas de ultrasonidos que se transmiten sobre la pantalla táctil. Cuando la pantalla es tocada, una parte de la onda es absorbida. Este cambio en las ondas de ultrasonidos permite registrar la posición en la que se ha tocado la pantalla y enviarla al controlador para que pueda procesarla (“Figura 6”).

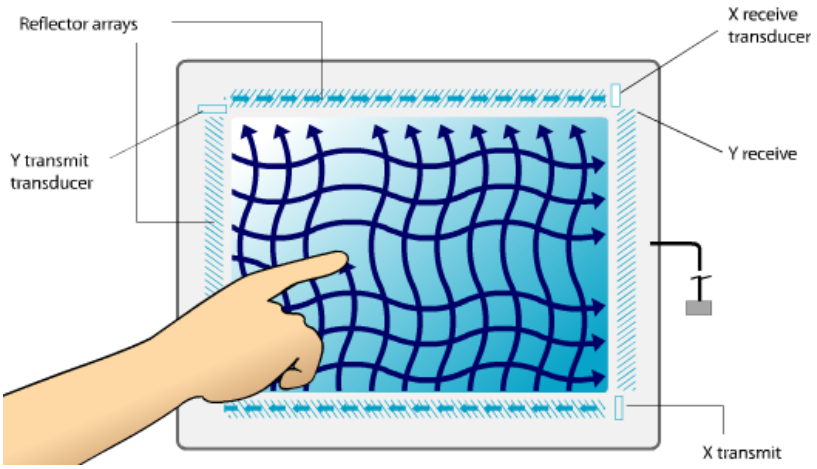


Figura 6. Funcionamiento de una pantalla de onda acústica

El funcionamiento de estas pantallas puede verse afectado por elementos externos. La presencia de contaminantes sobre la superficie también puede interferir con el funcionamiento de la pantalla táctil.

A través de la superficie del cristal se transmiten dos ondas acústicas inaudibles para el hombre. Una de las ondas se transmite horizontalmente y la otra verticalmente. Cada onda se dispersa por la superficie de la pantalla rebotando en unos reflectores acústicos (“Figura 7”).

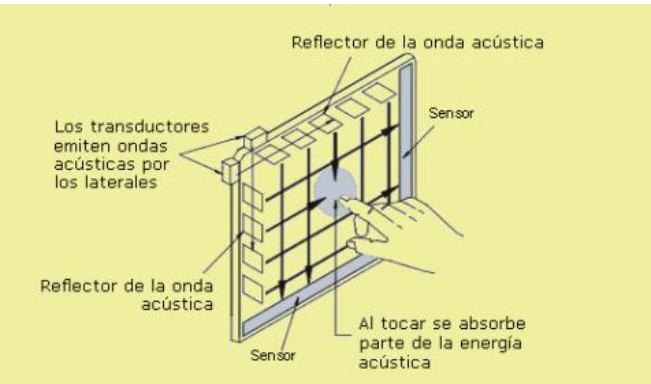


Figura 7. Componentes de una pantalla de onda acústica

2.1.3 Capacitivas

Una pantalla táctil capacitiva [6] (“Figura 9”) está cubierta con un material, habitualmente óxido de indio y estaño que conduce una corriente eléctrica continua a través del sensor. El sensor por tanto muestra un campo de electrones controlado con precisión tanto en el eje vertical como en el horizontal, es decir, adquiere capacitancia. El cuerpo humano también se puede considerar un dispositivo eléctrico en cuyo interior hay electrones, por lo que también dispone de capacitancia.

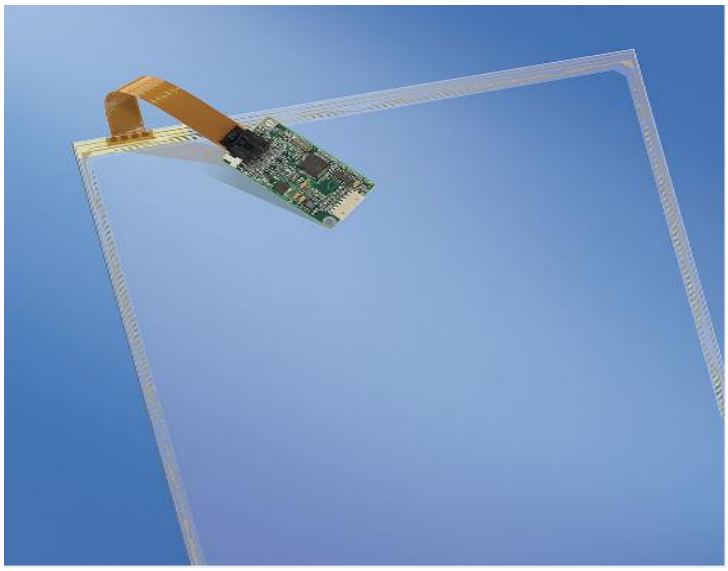


Figura 8. Ejemplo pantalla capacitiva

Cuando el campo de capacitancia normal del sensor (su estado de referencia) es alterado por otro campo de capacitancia, como puede ser el dedo de una persona (“Figura 9”), los circuitos electrónicos situados en cada esquina de la pantalla miden la 'distorsión' resultante en la onda senoidal característica del campo de referencia y envía la información acerca de este evento al controlador para su procesamiento matemático.

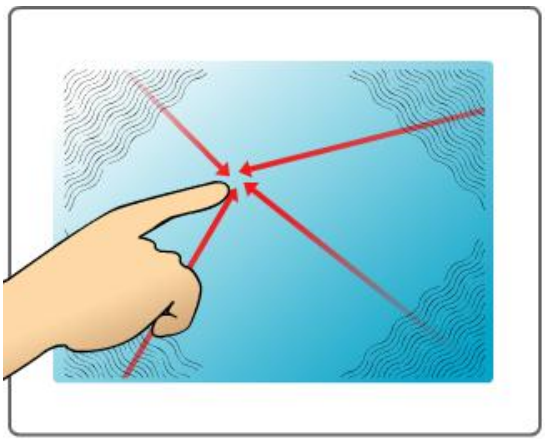


Figura 9. Funcionamiento tecnología capacitiva

Los sensores capacitivos deben ser tocados con un dispositivo conductivo en contacto directo con la mano o con un dedo, al contrario que las pantallas resistivas o de onda superficial en las que se puede utilizar cualquier objeto.

Las pantallas táctiles capacitivas no se ven afectadas por elementos externos y tienen una alta claridad, pero su complejo procesado de la señal hace que su coste sea elevado.

2.1.3.1 Multi-Touch Screen Capacitiva

Por su tecnología, las pantallas capacitivas necesitan ser manejadas **mediante el dedo** o un objeto que disponga de capacitancia, como se observa en la “Figura 10”, no siendo aptas para los típicos stylus. Por otro lado, pueden detectar varias **pulsaciones simultáneas** o gestos, lo que permite diversas formas de actuar con ellas, aumentando su capacidad para ser controladas. Las pulsaciones o gestos no requieren presión, basta con deslizar el dedo para controlar la pantalla del dispositivo.

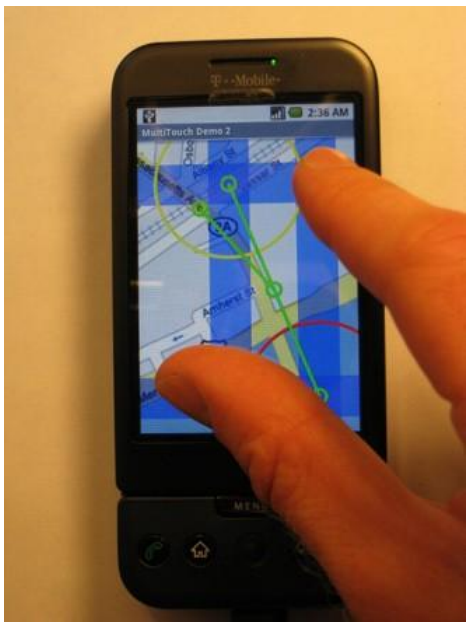


Figura 10. Ejemplo de utilización de una pantalla capacitiva

Al ser utilizadas directamente por el dedo, sin objetos intermedios, y no ser necesario realizar ninguna presión, la **experiencia para el usuario** al manejar este tipo de pantallas es bastante buena. La impresión es de rapidez, de inmediatez, siempre que el sistema operativo y el programa que estemos manejando este bien diseñado, claro está.

También tienen sus limitaciones. Tener que usar los dedos, menor precisión y no detectar la presión limitan las posibilidades del software que pueden ejecutar.

Apple con iPhone (“Figura 11”), los HTC Dream y Magic, modelos de Samsung y LG, Blackberry Storm o la próxima Palm Pre disponen de pantallas capacitivas. Los grandes ausentes eran, hasta ahora, Windows Mobile y Symbian. El MWC 2009 nos ha dejado el Samsung Omnia HD, con sistema Symbian, que está fabricado con pantalla capacitiva y que nos ha dejado muy buenas impresiones.

Figura 11. Iphone, ejemplo de tecnología capacitiva

2.1.3.2 TOUCHPAD

El **touchpad** o **trackpad** es un dispositivo táctil de entrada que permite controlar un cursor o facilitar la navegación a través de un menú o de cualquier interfaz gráfica. Un ejemplo de TouchPad se puede encontrar en la “Figura 13”.

El touchpad está formado por una rejilla de dos capas de tiras de electrodos, una vertical y otra horizontal, separadas por un aislante y conectadas a un sofisticado circuito. El circuito se encarga de medir la capacidad mutua entre cada electrodo vertical y cada electrodo horizontal. Un dedo situado cerca de la intersección de dos electrodos modifica la capacidad mutua entre ellos al modificarse las propiedades dieléctricas de su entorno. Este funcionamiento se puede observar gráficamente en la “Figura 12”.

La posición del dedo se calcula con precisión basándose en las variaciones de la capacidad mutua en varios puntos hasta determinar el centroide de la superficie de contacto. La resolución de este sistema es impresionante, hasta 1/40 mm. Además se puede medir también la presión que se hace con el dedo.

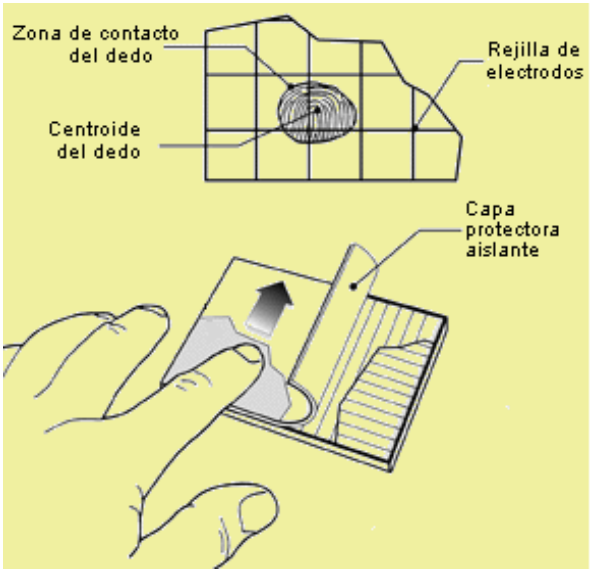


Figura 12. Funcionamiento TouchPad

No se pueden usar lápices u otros materiales no conductores como punteros. Es muy resistente al entorno, soporta perfectamente polvo, humedad, electricidad estática, etc. Además es ligero, fino y puede ser flexible o transparente.

Figura 13. Ejemplo TouchPad

2.1.4 Infrarrojos

El sistema más antiguo y fácil de entender es el sistema de infrarrojos. En los bordes de la pantalla, en la carcasa de la misma, existen unos emisores de infrarrojos. Estos emisores rodean por completo el perfil de la pantalla. Tenemos una matriz de rayos infrarrojos vertical y horizontal. Al pulsar con el dedo o con cualquier objeto, sobre la pantalla interrumpimos un haz infrarrojo vertical y otro horizontal (“Figura 14”). El controlador detecta que rayos han sido interrumpidos, conoce de este modo dónde hemos pulsado y actúa en consecuencia.

Este sistema tiene la ventaja de la simplicidad y de no oscurecer la pantalla, pero tiene algunas desventajas: son voluminosas, muy sensibles a la suciedad y pueden detectar fácilmente falsas pulsaciones.

Este tipo de pantallas son muy resistentes por lo que son utilizadas en muchas de las aplicaciones militares que exigen una pantalla táctil.

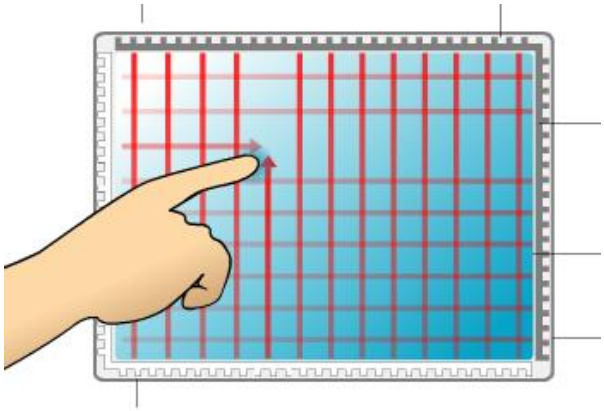


Figura 14. Pantalla infrarrojos

2.1.4.1 Multi-Touch Sensing through Frustrated Total Internal Reflection (FTIR)

En primer lugar comenzaremos dando una breve definición de **Reflexión Interna Total** y a continuación describiremos algunos de los componentes necesarios para construir la multi-touch con este método:

Se denomina **reflexión interna total** al fenómeno que se produce cuando un rayo de luz, atravesando un medio de índice de refracción *n* más grande que el índice de refracción en el que este se encuentra, se refracta de tal modo que no es capaz de atravesar la superficie entre ambos medios reflejándose completamente. Mediante este método no se produce ninguna pérdida de energía mientras el haz atraviesa el medio (“Figura 15”).

Una vez comentado lo anterior ya podemos introducir una breve explicación de cuál es el funcionamiento del sistema mediante la **Reflexión Interna Total [1], [14]**.

Con este método utilizamos un panel de acrílico, el cual iluminamos desde sus bordes con LEDs infrarrojos. Utilizamos acrílico y no cristal, porque su transmitancia frente a la luz infrarroja es ideal, mientras que la del cristal no nos sirve para lograr este efecto.

Lo que estamos haciendo es "inundar" el panel de luz IR, la cual va rebotando en su interior (reflexión interna total). Cuando colocamos los dedos sobre el panel, estamos frustrando la reflexión interna y reflejando la luz hacia abajo, donde tenemos una cámara infrarroja, o una videocámara normal con un filtro IR que detectará esos "blobs" de luz, los cuales procesaremos con el ordenador para obtener su posición y de ese modo emplearlos como controladores.

Sobre el panel acrílico colocamos un material difusor sobre el que proyectar la imagen del ordenador desde abajo.

Al utilizar los dedos desnudos sobre el panel, el efecto de "frustración" funciona muy bien, especialmente si tenemos los dedos húmedos, debido a que la piel de los dedos es semitransparente y se deforma al apretarla contra el panel, sin embargo al colocar la pantalla difusora, eliminamos el efecto y es necesario hacer bastante presión para obtener el resultado deseado. Para ello debemos utilizar una superficie intermedia entre el panel y el difusor (conocida como complaint surface). Se ha investigado bastante sobre este tema, y actualmente lo que mejor resultado da es crear una fina película de silicona transparente (Sort A Clear Silicone Rubber). Esto complica algo el diseño del prototipo, y lo encarece un poco.

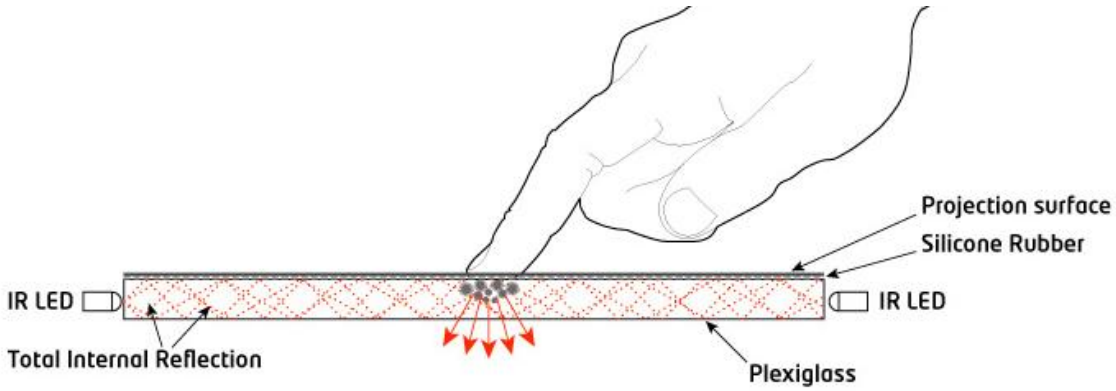


Figura 15. Explicación FTIR

2.1.4.2 Sphere multi-touch computing

Es un nuevo sistema con tecnología "Multi-Touch", se llama Microsoft Sphere [3]("Figura 17") y como su nombre lo indica, se trata de una esfera con sensibilidad "Multi-Touch" y en la que se puede realizar acciones básicas como girar, ampliar, rotar, trasponer, y enviar imágenes, visualizar videos e incluso pintar, jugar y configurar puntos para funciones especificas.

Sphere utiliza un sistema de cámaras infrarrojas para detectar las manos y elementos de contacto, además de un proyector para las salidas “output”, y así crear una especie de computador “Multi-Touch. La cámara y el proyector comparten el mismo eje óptico en virtud de los espejos. La esfera tiene poco mas de 60cm de diámetro lo que se traduje en una esfera de 18″ montada en un pedestal y capacidad de giro de 360 grados en varios sentidos (izquierda, derecha, arriba, abajo, diagonal). En la “Figura 16” se puede observar un diseño del sistema Sphere.

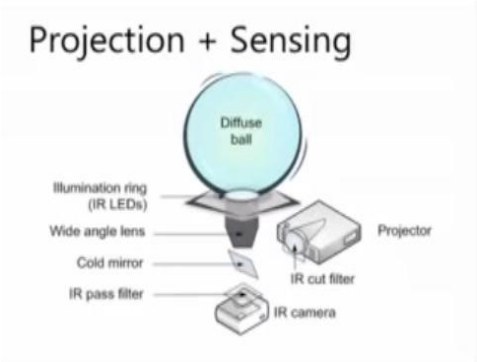


Figura 16. Diseño sistema Sphere.

Por ahora Sphere es solo un proyecto experimental en el cual Microsoft lleva las tecnologías “Multi-Touch” a dispositivos de formas geométricas distinta a las planimétricas pantallas planas, por lo tanto, Sphere no posee por el momento planes comerciales.

Figura 17. Ejemplo Sphere en funcionamiento

2.1.4.3 ARForce - another Infrared AR Concept

ARForce es un sensor creado en la Universidad de Tokyo, con el que podemos saber la posición, magnitud y dirección de la fuerza aplicada por el usuario. También tiene capacidad “Multi-Touch”. Este sistema puede ser observado a continuación, en la “Figura 18”.

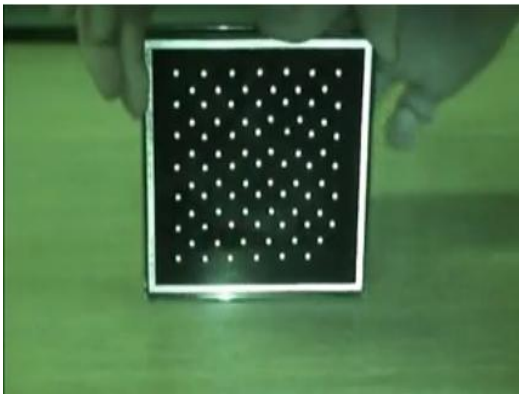


Figura 18. Sistema ARForce

ARForce consiste en un array de puntos visible bajo luz infrarroja y una cámara. Cuando se presiona, los puntos se realinean un poco, activando el cálculo del vector de presión. La diferencia entre los puntos actuales y los iniciales nos sirven para identificar los movimientos realizados. Un ejemplo de utilización de este sistema se puede observar en la “Figura 19”.

Figura 19. Ejemplo uso ARForce

2.1.5 Galga Extensiométrica

Cuando se utilizan galgas extensiométricas la pantalla tiene una estructura elástica de forma que se pueden utilizar galgas extensiométricas para determinar la posición en que ha sido tocada a partir de las deformaciones producidas en la misma. Esta tecnología también puede medir el eje Z o la presión ejercida sobre la pantalla.

Se usan habitualmente en sistemas que se encuentran expuestos al público como máquinas de venta de entradas (“Figura 20”), debido sobre todo a su resistencia al vandalismo.

Figura 20. Ejemplo pantalla Galga Extensiométrica

2.1.6 Imagen Óptica

Es un desarrollo relativamente moderno en la tecnología de pantallas táctiles, dos o más sensores son situados alrededor de la pantalla, habitualmente en las esquinas. Emisores de infrarrojos son situados en el campo de vista de la cámara en los otros lados de la pantalla. Un toque en la pantalla muestra una sombra de forma que cada par de cámaras puede triangularizarla para localizar el punto de contacto. Esta tecnología está ganando popularidad debido a su escalabilidad, versatilidad y asequibilidad, especialmente para pantallas de gran tamaño.

2.1.6.1 Multi-Touch Sensing through Diffused Illumination (DI)

Un método utilizado en este tipo de tecnología es la iluminación difusa [14] (Luz que ha sido difundida por la reflexión o al pasar por un material translúcido).

En este caso no inundamos de luz IR el panel (el cual ahora puede ser de plexiglás, cristal escarchado o lo que queramos), sino que lo iluminamos desde la parte inferior empleando lámparas LED, y procurando que quede totalmente iluminado. Al colocar los dedos sobre el panel, estaremos reflejando la luz que viene desde abajo otra vez hacia allí, donde tenemos nuestra cámara. Un ejemplo gráfico del funcionamiento de esta tecnología se puede observar en la “Figura 21”.

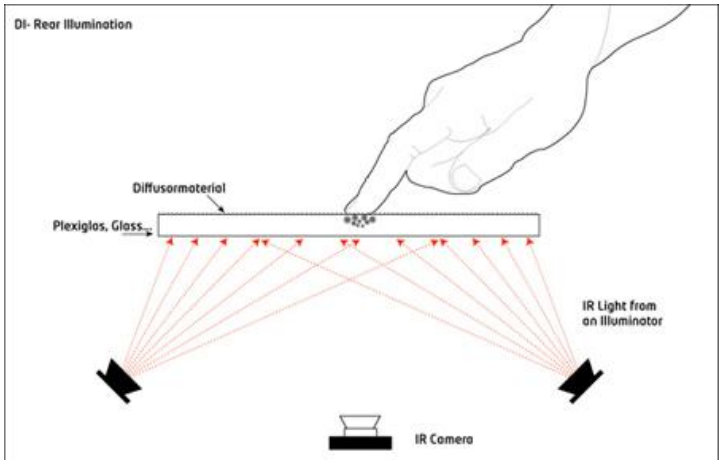


Figura 21. Funcionamiento Iluminación Difusa

2.1.7 Tecnología de Señal Dispersiva

Introducida en el año 2002, este sistema utiliza sensores para detectar la energía mecánica producida en el cristal debido a un toque. Unos algoritmos complejos se encargan de interpretar esta información para obtener el punto exacto del contacto. Esta tecnología es muy resistente al polvo y otros elementos externos, incluidos arañazos. Como no hay necesidad de elementos adicionales en la pantalla también proporciona unos excelentes niveles de claridad. Por otro lado, como el contacto es detectado a través de vibraciones mecánicas, cualquier objeto puede ser utilizado para detectar estos eventos, incluyendo el dedo o uñas.

Un efecto lateral negativo de esta tecnología es que tras el contacto inicial el sistema no es capaz de detectar un dedo u objeto que se encuentre parado tocando la pantalla.

2.1.8 Reconocimiento de Pulso Acústico

Introducida en el año 2006, estos sistemas utilizan cuatro transductores piezoeléctricos situados en cada lado de la pantalla para convertir la energía mecánica del contacto en una señal electrónica (“Figura 22”). Esta señal es posteriormente convertida en una onda de sonido, la cual es comparada con el perfil de sonido preexistente para cada posición en la pantalla. Esto se puede comprobar gráficamente en la “Figura 23”.

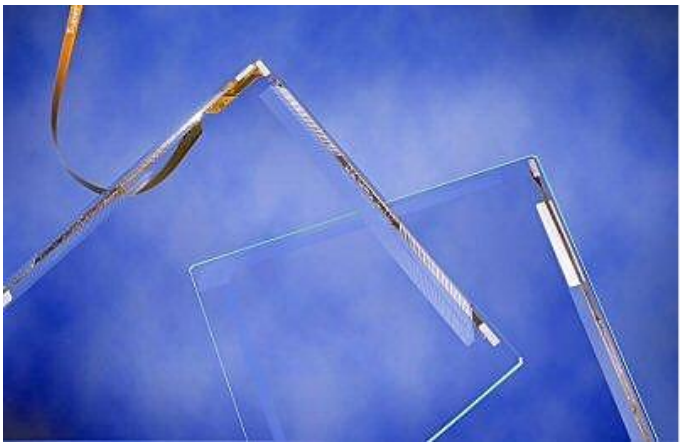


Figura 22. Pantalla de Reconocimiento de Pulso Acústico

Este sistema tiene la ventaja de que no necesita ninguna malla de cables sobre la pantalla y que la pantalla táctil es de hecho de cristal, proporcionando la óptica y la durabilidad del cristal con el que está fabricada.

También presenta las ventajas de funcionar con arañazos y polvo sobre la pantalla, de tener unos altos niveles de precisión y de que no necesita ningún objeto especial para su utilización.

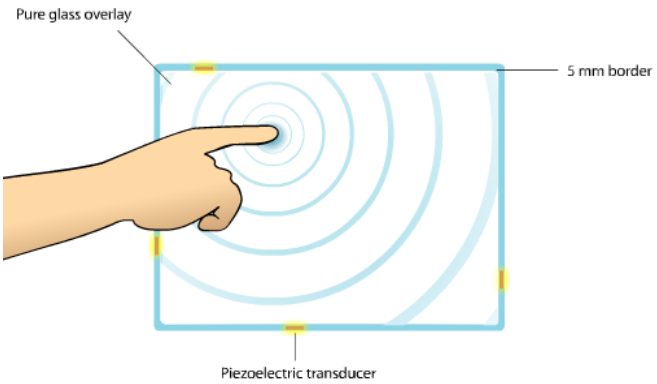


Figura 23. Reconocimiento de contacto de Pulso Acústico

2.1.9 Otras Tecnologías

2.1.9.1 3D Touch Panel

El prototipo de pantalla de la imagen siguiente que, creado por Mitsubishi, sencillamente no necesita contacto. En efecto, interpreta los movimientos del dedo a distancia**.** Esto se puede observar claramente en la “Figura 24”.

Eso sí, tiene que ser una distancia muy corta, de apenas unos centímetros. Una proximidad necesaria para que pueda trabajar todo el conjunto de sensores de esta pantalla.

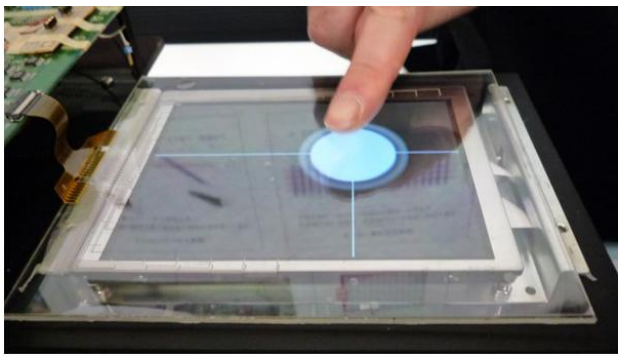


Figura 24. Ejemplo 3D TouchPanel.

No es que la pantalla muestre las imágenes en tres dimensiones, y tampoco incluye una interfaz con iconos a los que podemos darle la vuelta. El motivo de su nombre está en que los sensores detectan el movimiento del dedo en sus tres dimensiones: desplazamiento vertical, desplazamiento horizontal y acercamiento/alejamiento**.**

En la práctica, este sistema de manejo podría funcionar de distintos modos. Por ejemplo, aumentando el tamaño de los iconos cuando pasemos el dedo por encima de ellos (pero sin tocarlos, “Figura 24”)**.** Esto es lo que Mitsubishi llama la función *“mouseover”*. O creando un cursor en forma de punto, cuyo tamaño aumentaría a medida que acercamos el dedo, y menguaría de forma gradual mientras vamos alejando la yema de la pantalla**.**

Además de eso, los sensores también pueden detectar la velocidad del movimiento. Por ejemplo, programando la iluminación para que adquiera un tono rojizo al detectar movimientos rápidos, y que pase a azul con movimientos más pausados.

Un apartado de posibilidades bastante completo. Además, en Mitsubishi afirman que se ha usado como base un tipo de pantalla que ya está disponible en el mercado, y que añadirle toda esta nueva tecnología no supondría un aumento significativo de los costes de producción**.**

Este primer prototipo tiene **5,7** pulgadas de diagonal y resolución VGA (640 x 480 píxeles). Y parece ser el tamaño tope con el que han conseguido buenos resultados. De momento, pues están trabajando en un modelo de 10 pulgadas. Eso sí, **descartan la posibilidad de pantallas grandes**. Nada de una Microsoft Surface que se maneje sin tocar su pantalla, por poner un ejemplo. El invento, de momento, está enfocado a pantallas pequeñas, como las de los teléfonos móviles**.**

El segundo problema está en la **s**ensibilidad. El dispositivo reacciona peor en el llamado estado de proximidad (no tocamos la pantalla) que en el llamado estado de contacto. Al fin y al cabo, no hay que olvidar que la pantalla es táctil y que, por lo tanto, también podemos tocarla para manejar las funciones que queramos. En cualquier caso, desde Mitsubishi afirman que, en la práctica, la diferencia entre ambos estados no sería demasiado sustancial para el usuario**.**

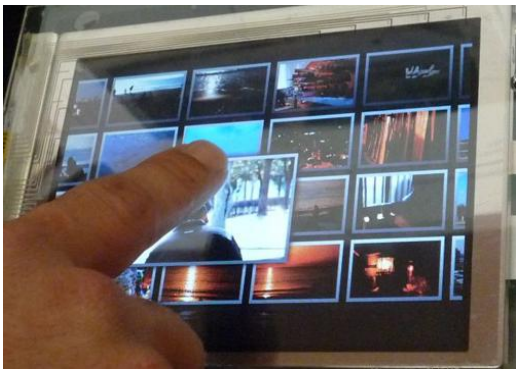


Figura 25. Ejemplo de uso 3D Panel

2.1.9.2 Flexible Multi-Touch

Este sistema consiste en la integración de un componente flexible en el sistema. Dicho componente es la pantalla de la “Multi-Touch" y quedando como resultado una pantalla combinada flexible (“Figura 27”).

A continuación, se describe el método de funcionamiento, el aparato y el sistema de una “Multi-Touch” flexible. Esta invención consiste en dos láminas fabricadas en substrato flexible, una donde se proyectará la imagen desde arriba y otra sensitiva donde se colocarán los sensores de presión (como se puede observar en la “Figura 26”), configurados para detectar varios puntos de contacto en localizaciones distintas de la capa y para generar distintas señales que representan la localización de cada pulsación. Por lo tanto, el método de funcionamiento del Touch Panel Flexible estará compuesto por: manejo de uno o más de los sensores de presión comentados anteriormente y detección de más de un contacto simultáneo.

Esta tecnología también está dirigida a pantallas flexibles “Multi-Touch”, esta comprenderá:

- Un display (pantalla) como interfaz de usuario.
- Un panel “Multi-Touch” con una parte flexible para combinar con el display anterior pudiendo así detectar simultáneos dedos en la pantalla.

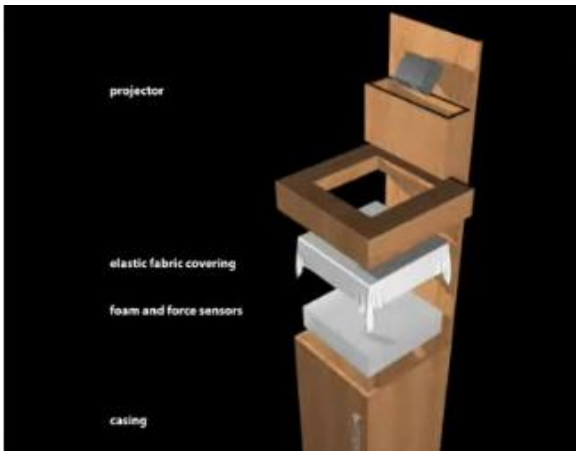


Figura 26. Vista general del sistema

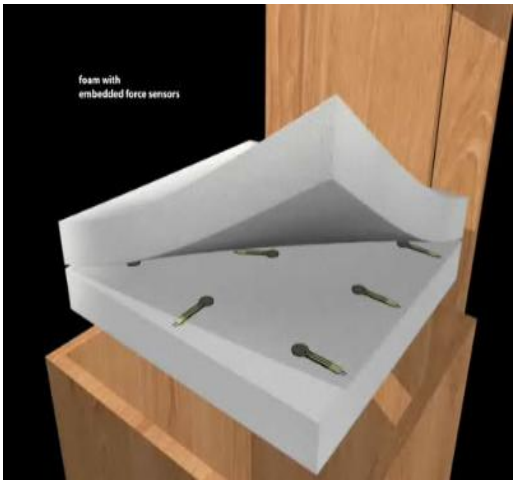


Figura 27. Vista de la "pantalla" de contacto

2.1.10 Tabla Comparación Hardware

A continuación realizaremos una pequeña comparación de las tecnologías más relevantes de las que hemos comentado anteriormente, destacando tanto los puntos positivos como negativos de cada una de ellas.

	Resistiva	Onda Acústica	Capacitiva	Infrarrojos
Pantalla < 10"	Muy bueno	No posible	Neutro	No muy aceptable
Pantalla > 10"	No posible	Neutro	No muy aceptable	Muy bueno
Eje Z	No posible	Muy bueno	No posible	No posible
Fiabilidad	Neutro	Neutro	Neutro	Muy bueno
Precisión	Muy bueno	Muy bueno	No muy aceptable	Muy bueno
Sensibilidad	Neutro	Neutro	Bueno	Muy bueno
Potencia Necesaria	Neutro	Neutro	Neutro	Neutro
Durabilidad	Bueno	Muy bueno	Bueno	Muy bueno
MultiTouch	No posible	No posible	Muy bueno	Muy bueno
Coste	Neutro	Neutro	Bueno	Neutro
Principales ventajas	No se ven afectadas por elementos externos.	Capaz de detectar el eje Z. Presión aproximada ejercida.	Imagen más clara. Puede medir la presión.	MultiTouch asequible. Simplicidad No oscurece la pantalla
Principales desventajas	Pérdida de luminosidad. No MultiTouch	No MultiTouch	No lápices u materiales no conductores como punteros.	Voluminosas Muy sensibles a la suciedad

Como se puede observar en la tabla anterior. De todas las alternativas posibles que se comentan en el apartado “2.1 Hardware” del Estado del Arte se realizó un filtrado dejando las tecnologías más utilizadas y que, en un principio, mejor se ajustaban al sistema a desarrollar. Estas son:

Resistiva: De esta tecnología se puede destacar que posee una gran durabilidad, precisión y un coste aceptable. Por otro lado, posee inconvenientes que hacen que no sea apropiada. Estos son: no se pueden construir pantallas mayores de 10 pulgadas y no soporta más de un contacto simultaneo. Es decir, no puede ser “Multi-Touch”.

Onda acústica: Esta ofrece una gran precisión y posibilidad de fabricación de pantallas mayores de 10 pulgadas. Además, su coste se encuentra dentro de los límites exigidos por los requisitos de los usuarios. Su principal desventaja es que no distingue más de un contacto simultáneo. Esto se deberá tener muy en cuenta a la hora de realizar la elección final.

Capacitiva: Esta tecnología, a pesar de permitir la captura de un contacto simultaneo, poseer un coste muy bueno y una durabilidad y fiabilidad bastante buenas, posee una gran desventaja, no es recomendable su desarrollo en pantallas mayores de 10 pulgadas ya que su coste se dispara. Esto hace que se replantee su utilización en el desarrollo del sistema.

Infrarrojos: La última tecnología que se va a destacar, a primera vista, cumple todas las condiciones exigidas por los clientes. Es posible la fabricación de pantallas grandes, su coste esta dentro de los límites, puede ser “Multi-Touch”, posee una gran durabilidad y precisión con lo que se tendrá en consideración a la hora de realizar la elección.

2.2 Software:

Para la realización del software (**soporte lógico** de un computador digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica) de la aplicación examinamos las posibilidades que más se ajustaban a nuestras necesidades las cuales detallamos a continuación:

2.2.1 TouchLib

Es una librería utilizada para la realización de aplicaciones “Multi-Touch” utilizando las técnicas FTIR e Iluminación Difusa. También ofrece algo de soporte en la identificación y tracking de puntos de luz infrarroja, mandando a las aplicaciones que lo soliciten esos eventos.

Touchlib maneja las imágenes adquiridas de la webcam u otro hardware de captura de video utilizando algunas librerías conocidas. Esta librería también maneja el procesamiento de estas imágenes, mediante “Blob Detection” y “Blob Tracking”. Y podemos realizar el envío de eventos MultiTouch, como un “finger down”, “finger moved”, “finger released” entre otros a nuestro programa en c++ mediante una interfaz muy sencilla.

Esta incluye una configuración de app y algunas demos para la iniciación del usuario, una de ellas se puede observar en la “Figura 28”, y reconocerá la mayoría de webcams y dispositivos de captura de video. Si se desea utilizar la Touchlib deberemos estar preparados para crear nuestras propias apps. Esto se puede realizar de diferentes maneras. Podemos crear aplicaciones en C++ y aprovecharnos de la potencia de la librería Touchlib.

Actualmente solo funciona bajo Windows aunque se está trabajando en su migración a otras plataformas.

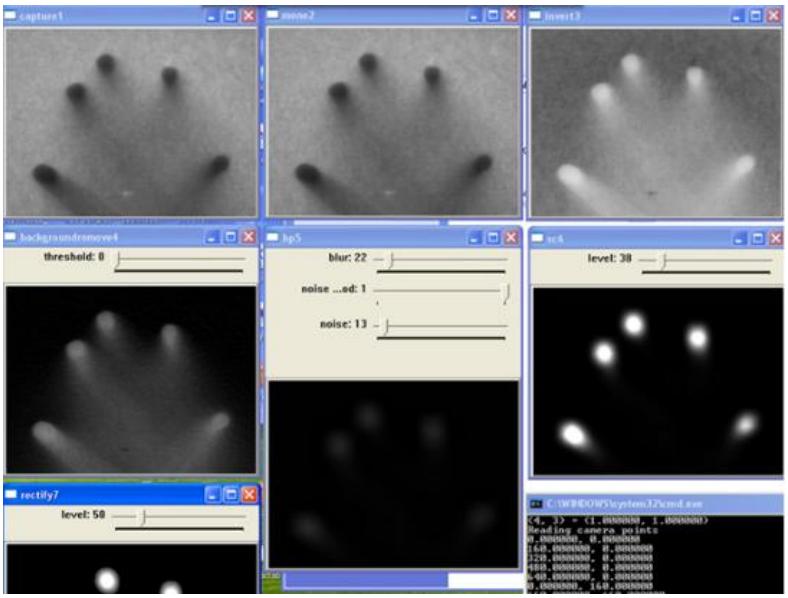


Figura 28. Ejemplo de una aplicación de la Touchlib

2.2.2 Johnny Lee

Johnny Chung Lee es un estudiante graduado de la Universidad de Carnegie Mellon y miembro del Human-Computer Interaction Institute. Este hombre, usando el Wiimote (mando principal de la consola Wii de Nintendo), un sensor IR y cinta reflectante nos muestra una interfaz Multi-Touch sin el Touch (“Figura 29”). La cinta reflectante en sus dedos le permite seguir el movimiento de estos para controlar esta interfaz.

Figura 29. Sistema detección de puntos Jhnnt Lee

El funcionamiento para “trackear” los dedos es el siguiente. Usando un vector de leds y algo de la cinta reflectante, usamos la cámara infrarroja del mando de Wii para trackear los objetos, como los dedos, en el espacio 2D. Esto te permite interactuar con tu computadora simplemente moviendo los dedos en el aire (“Figura 30”). Esta tecnología es muy similar a la visto en la película “Minority Report”. El Wiimote puede “trackear” hasta 4 dedos simultáneamente.

El software multipunto está realizado en C#. Para la modificación del mismo debemos tener instalado en nuestro PC una copia del Microsoft Visual C# Express (para compilar el código) y del DirectX SDK(para poder realizar cambios en el código). Por el contrario si lo que deseamos es únicamente ejecutar la aplicación, solo necesitamos bajarnos el .exe y ejecutarlo en el directorio principal.

Para más información sobre este y otros proyecto relacionados por Lee visitar la página que se nombra a continuación: http://johnnylee.net

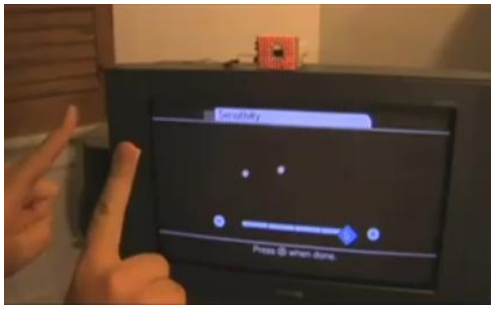


Figura 30. Ejemplo de reconocimiento de dos puntos.

2.2.3 SDK

Son un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, ordenadores, videoconsolas, sistemas operativos, etc.

Es algo tan sencillo como una interfaz de programación de aplicaciones o API (del inglés application programing interface) creada para permitir el uso de cierto lenguaje de programación, o puede, también, incluir hardware sofisticado para comunicarse con un determinado sistema embebido. Las herramientas más comunes incluyen soporte para la detección de errores de programación como un entorno de desarrollo integrado o IDE (del inglés Integrated Development Environment) y otras utilidades.

2.2.3.1 Optitrack SDK

El Optitrack SDK nos permite crear fácilmente nuestras propias aplicaciones de tracking. Podemos observar múltiples cámaras simultáneamente, “trakear” múltiples objetos e integrar los datos recogidos en nuestras propias aplicaciones. La API está disponible vía la interfaz COM. Eso la provee de compatibilidad con C/C++, Delphi, Visual Basic, VBScript y algún otro lenguaje.

Existen algunas aplicaciones de prueba del Optitrack SDK disponibles en la web de NaturalPoint (http://www.naturalpoint.com/optitrack/support/sample-applications.html) donde podemos encontrar el código fuente completo con las que podremos basarnos para realizar nuestras propias aplicaciones o simplemente utilizar las que ya existen. A continuación, en la “Figura 31”, se muestra una interfaz correspondiente a una aplicación de prueba de una cámara Optitrack.

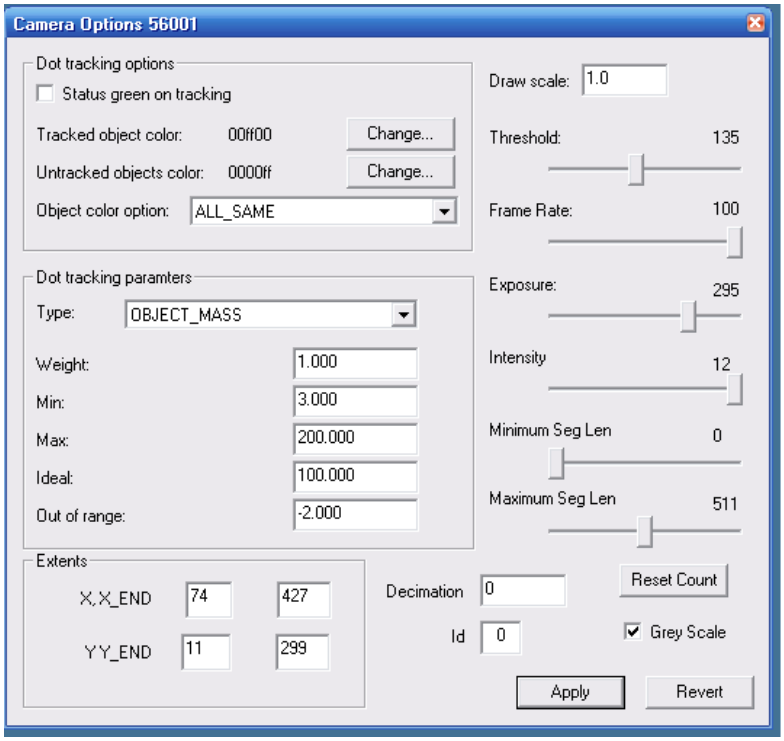


Figura 31. Interfaz de configuración del SDK de Optitrack

2.2.4 OpenCV

OpenCv es una librería de código abierto de Visión por Computador. Es una colección de funciones de C y C++ que implementan algunos algoritmos de Procesamiento de Imágenes y Visión por Computador.

En el caso de los sistemas MultiTouch, utilizamos las funciones específicas de OpenCV de “Calibración de Cámaras” (../OpenCV/docs/ref/opencvref_cv .htm) para pasar los puntos de la imagen recogida por la cámara, la cual estará en coordenadas de cámara, a coordenadas de la pantalla.

Por último, cabe destacar que en su última versión OpenCV tiene también soporte para detección y “tracking” de puntos. Este también puede ser un buen punto de partida para la implementación de un sistema MultiTouch.

Figura 32. Ejemplo de marca utilizada para reconocimiento de puntos en la OpenCV.

2.3 Gestos MultiTouch

Como hemos comentado en la introducción de este apartado, existe la posibilidad de realizar un análisis en tiempo de ejecución del sistema para reconocer diferentes gestos que se pueden necesitar en la ejecución del sistema [4] [5] [7] [8]. El reconocimiento de cualquiera de estos gestos implica una respuesta desde el sistema (la respuesta dependerá del gesto reconocido). Algunos de estos gestos se detallan a continuación:

Con un dedo.

- **Circunferencia:** Para reconocer este gesto se realiza una circunferencia en cualquier posición de la pantalla. Esto se hará con un solo punto por lo que deberemos comprobar SIEMPRE que haya un punto sobre la misma si se está realizando el gesto o no. Comprobaremos en cada iteración si el punto está a una distancia “x” (nuestro centro) más o menos, si completamos la circunferencia y se han cumplido las premisas se reconoce como gesto realizando la tarea asignada a dicho gesto.

- **Mantener pulsado:** Para este gesto utilizamos un “tiempo de vivo”, el cual, si el punto es encontrado durante ese “tiempo de vivo” determinado, el sistema lo reconocerá como un gesto, mostrando o realizando la tarea asignada a ese gesto, por ejemplo, mostrar el menú del click derecho del ratón.

Con dos dedos.

- **Deslizar 2 dedos**: Puedes hacer “scroll”, es decir desplazar una página arriba y abajo, con sólo deslizar dos dedos en la dirección deseada. Este gesto se realizará cuando se detecten 2 puntos a una distancia x y estos se desplacen en el mismo sentido.

- **Alejar/Acercar 2 dedos:** Se puede acercar o alejar una imagen o documento acercando o alejando respectivamente los dedos (que deberán estar a una distancia mínima/máxima seleccionada) con dicha imagen o documento seleccionado.

- **Con 1 dedo pulsado, colocar otro a una distancia x:** Este gesto será reconocido cuando, teniendo pulsado un dedo en la pantalla, coloquemos otro a una distancia x del mismo. Esto se reconocerá como el click del ratón.

Con tres dedos.

- **Mover 3 dedos en cualquier dirección una distancia determinada:** Para este gesto utilizaremos las posiciones de los 3 puntos (3 dedos). Compararemos en cada frame el punto actual con el punto anterior durante una distancia determinada. Una vez rebasada esa distancia y verificado que los puntos que hemos seguido son los mismos desde el principio del movimiento al final, el sistema lo reconocerá como un gesto.

- **Mantener 3 dedos un tiempo determinado:** Para este gesto utilizamos un “tiempo de vivo”, el cual, si los puntos son encontrados durante ese “tiempo de vivo” determinado, el sistema lo reconocerá como un gesto, mostrando o realizando la tarea asignada a ese gesto.

Con cuatro dedos.

- **Mantener 4 dedos un tiempo determinado:** Para este gesto utilizamos un “tiempo de vivo”, el cual, si los puntos son encontrados durante ese “tiempo de vivo” determinado, el sistema lo reconocerá como un gesto, mostrando o realizando la tarea asignada a ese gesto.

- **Acercar 4 dedos:** Este gesto se aceptará cuando los 4 puntos (que estén a una distancia máxima marcada por nosotros) se acerquen los unos a los otros hasta una distancia (también nosotros la señalaremos) mínima. En ese momento será reconocido como gesto.

Capítulo 3 Especificación

Este apartado tiene por objetivo identificar las necesidades expuestas por los clientes del sistema y que deberá soportar el sistema a desarrollar. Para ello se utilizará un sistema de entrevistas, en el que se conocerá de primera mano los requisitos de los usuarios identificándolos como, requisitos específicos y los más importantes, a los que daremos la máxima prioridad.

Una vez se tengan estos requisitos se hará una aproximación a las especificaciones de los recursos que serán necesarios para el desarrollo del proyecto.

Posteriormente, ya se podrá hacer una valoración del coste que conllevará el desarrollo del presente proyecto. En esta valoración se marcarán una serie de objetivos intermedios, llamados hitos, que nos permitirán observar si se están cumpliendo los tiempos especificados y por tanto, si el coste final de desarrollo será parecido al que se calculará en este apartado.

3.1- Análisis de requisitos de usuario

Para conocer de primera mano las necesidades del usuario se ha optado por llevar a cabo una serie de entrevistas. Concretamente se han concertado varias visitas con representantes de diferentes empresas y algunos miembros del departamento del servicio técnico en las cuales se ha manifestado que existen 2 subsistemas claramente diferenciados en las cuales se tendrá que asentar nuestro análisis. Estos son, el subsistema de hardware y el de software.

Primeramente comentaremos los requisitos obtenidos para el subsistema de hardware, en los que, destaca un hecho sobre todos lo demás, y es que el sistema sea de bajo coste. Por tanto, se deberá hacer hincapié en este tema, examinando las diferentes alternativas y teniendo este hecho muy en cuenta a la hora de realizar la elección final.

Otro de los aspectos en el que se ha insistido ha sido que el sistema debe ser lo más intuitivo posible, es decir, que se perciba clara e inmediatamente el funcionamiento del sistema, sin necesidad de realizar ningún razonamiento lógico.

El sistema ha implementar debe ser fácil de manejar, esto quiere decir, que no se necesite mucho tiempo de aprendizaje, ni ningún tipo de conocimiento en esta materia para su correcta utilización.

Además debe estar fabricado por componentes de alta disponibilidad, esto significa que en caso de producirse la rotura o avería de algún elemento que compone nuestro sistema el tiempo necesario en arreglar o reemplazar ese elemento sea el mínimo tiempo posible.

Por último y para finalizar con el análisis de este subsistema, en este momento la estética del sistema no es importante, este apartado no tiene ninguna relevancia con el funcionamiento del mismo y por lo tanto no es un hecho relevante para ninguna de las personas a las que se realizó la entrevista. Se destacó que una vez el sistema cumpliera los requisitos anteriores se realizaría un diseño definitivo del mismo en el cual estaría incluida el subsistema físico.

A continuación se continúa con los requisitos derivados del subsistema software del sistema. Este debe funcionar sobre la mayor cantidad de aplicaciones convencionales posible, esto quiere decir que nuestro sistema deberá ser capaz de trabajar como un periférico de entrada/salida.

El sistema deberá poseer una librería de desarrollo lo más sencilla y entendible posible y sobre todo bien documentada para facilitar posibles ampliaciones o modificaciones que se realicen en la librería. Ya que estas podrían ser realizadas por personas que no participaron en el desarrollo del proyecto actual. La librería deberá poseer código de intercambio de información para poder ser utilizada tanto remotamente como localmente.

Uno de los aspectos en los que más énfasis se puso en las entrevistas es que el sistema debía ser “Multi-Táctil” para las aplicaciones ya existentes que posean soporte “Multi-Touch”. Y además se concretó en que debía ser fácilmente adaptable a nuevas aplicaciones con soporte “Multi-Táctil” creadas a posteriori de la entrega del proyecto. También debe ser compatible para aplicaciones existentes o creadas posteriormente donde el soporte en ellas sea “Single-Táctil”.

Se acordó que el sistema utilizado para la calibración y el manejo de la pantalla fuera lo más sencilla posible, ya que la mayoría de los futuros usuarios de este no están familiarizados con sistemas de este tipo, con lo que, cuanto más complicada sea su utilización mayor será el coste de integración y peor la aceptación.

Uno de los aspectos que también se trató con bastante ahínco fue que la mesa deberá ser compatible con el resto de periféricos que posea el PC donde sea instalado. Esto se debe a que no se pretende que el sistema sustituya a los demás periféricos que se tienen actualmente, lo que se pretende es que complemente a estos utilizando cada uno de estos en el ámbito donde sea más apropiado su uso.

El último hecho a destacar en el apartado de software fue que el proyecto final deberá estar orientado a funcionar con aplicaciones multimedia ya sean aplicaciones con soporte “Multi-Touch” o “Single-Touch”, ya que en aplicaciones de este tipo van a ser en gran parte sobre las que trabaje el sistema.

Por último, debido a los pocos conocimientos técnicos sobre la materia que poseen generalmente en las empresas entrevistadas, todos querrían contratar un mantenimiento del sistema que permitiese su rápida reparación en caso de avería, o consultas en caso de dudas puntuales.

3.2- Especificaciones del sistema

En este apartado se mostrarán los elementos necesarios para poder desarrollar un sistema que cumpla con los requisitos expuestos por las empresas en las entrevistas llevadas a cabo.

Como se comentó anteriormente, el objetivo final es la fabricación de una mesa Multi-Táctil con tres subsistemas claramente diferenciados.

El primero se centra en la construcción de la plataforma hardware, en la cual destacamos tres puntos:

- **Estructura Física:** Se propone la fabricación de la estructura del sistema mediante perfiles estructurales de aluminio BOSH-REXROTH de 30x30 mm de diámetro que. La pantalla estará formada por dos capas:

  Una inferior de **retro-proyección**, estas combinan la funcionalidad y la estética del diseño para ofrecer una pantalla resistente, fácil de mantener y de larga duración. Este tipo de pantallas permiten la proyección con luz exterior de día.

  Una superior de **metacrilato**, que se encargará de proteger a la capa inferior además de realizar la función (FTIR) que nos permitirá recuperar los puntos cuando la pantalla sea tocada por alguien.

  Cabe destacar que alrededor del metacrilato se colocaran unas guías (una en cada borde) donde se propone colocar las tiras de cinco leds infrarrojos (LEDs IR SFH485 5mm) alimentados mediante una fuente de alimentación de la marca “Hercules” de 11Voltios y 5 Amperios.

  Esta pantalla tendrá un tamaño de 36 pulgadas el cual se consideró suficiente en las reuniones realizadas. Un boceto de lo que en un principio se comentó en las reuniones se puede observar en la “Figura 33”.

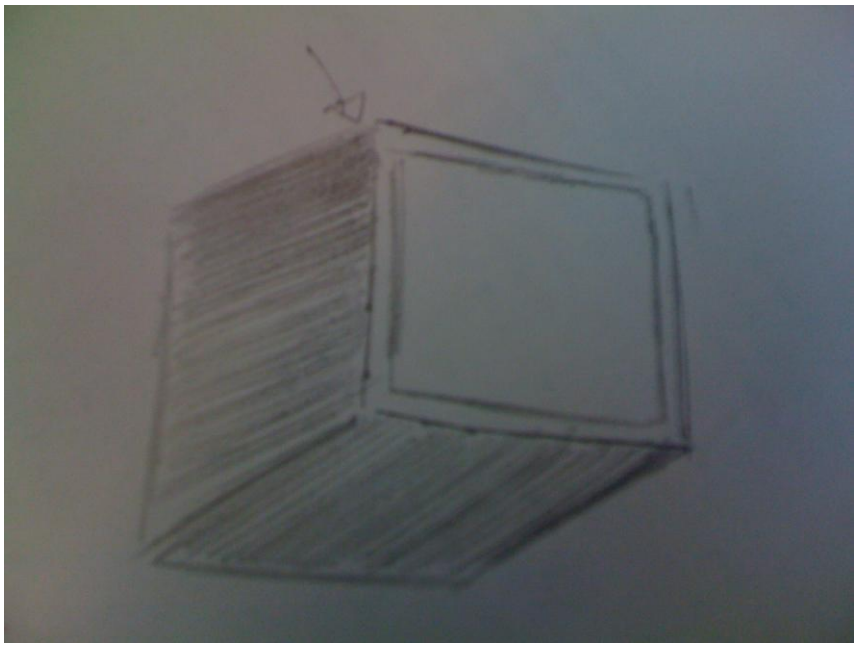


Figura 33. Primer boceto de la mesa (frontal)

- **Proyección:** Para realizar la proyección de la imagen en la pantalla se presenta la posibilidad de utilizar un proyector de uso habitual de la marca EIKI LC-XIP 2000.

- **Captura:** Después de realizar comparaciones entre diferentes alternativas. Para la captura se propone utilizar una cámara infrarroja de la marca Optitrack FLEX C-120 que es la que mejor se ajustará a los requisitos exigidos.

Tanto la proyección como la captura, se realizará a través del reflejo en un espejo para conseguir que tanto la imagen del proyector como la cámara infrarroja abarque toda la pantalla sin necesidad de alejarlos en demasía. Un sencillo boceto se muestra en la “Figura 34”.

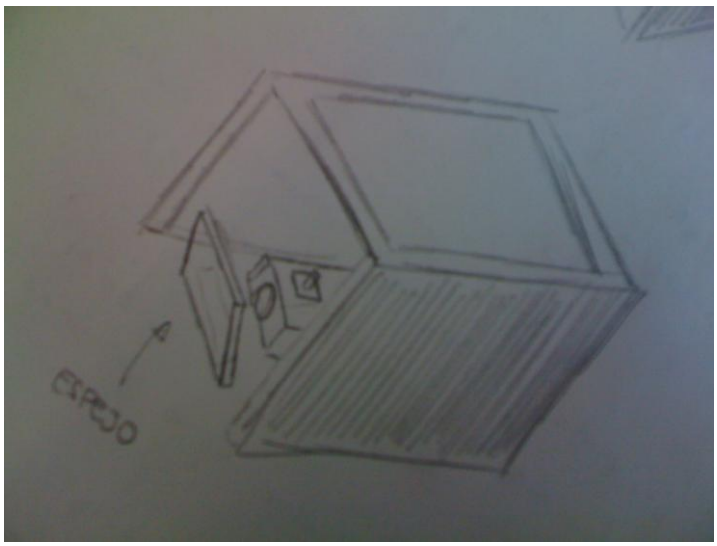


Figura 34. Primer boceto de la mesa (detrás)

Para finalizar, se destaca que todo lo comentado en el punto anterior cumple con las condiciones indicadas por los clientes, ya que son componentes fáciles de conseguir y de reemplazar. Además, su precio, contrastado con otros tipos de componentes encontrados es el más adecuado.

Para satisfacer los requisitos especificados por los clientes (funcionar sobre la mayoría de aplicaciones posibles, compatible con la mayor cantidad de periféricos, etc) en los otros dos puntos a desarrollar que forman parte del subsistema software. Se propone que sean realizadas bajo la plataforma del Sistema Operativo Microsoft Windows XP. Con esto, conseguimos que funcione sobre la mayoría de aplicaciones posibles.

Se propone realizar una librería en C++ utilizando el Microsoft Visual Studio 2005. Ella estará compuesta de una serie de funciones que se encargarán de realizar y controlar las acciones capturadas por el hardware de la mesa. Sirviendo además como interfaz al usuario.

Para la realización de uno de los aspectos en los que más énfasis se puso que fue que existiera soporte “Multi-Táctil”. Se propone la programación de nuevas funciones en la librería que se encarguen de capturar la cantidad de puntos que se desee tener como punteros. El número quedará definido por el usuario, ya que, en un principio, no existe limitación de hardware ni software en este aspecto.

Estos puntos “capturados” luego se tratarán para que cada uno de ellos pueda ser un puntero (ya en coordenadas de la pantalla). Para ello se deberá realizar un nuevo manejador ya que Windows sólo posee manejador para un puntero.

Al mismo tiempo como se acordó en la reuniones previas, la librería realizada se debe tratar que sea lo más clara posible y se escribirá una breve documentación comentado cuál es su funcionamiento.

Se plantea la posibilidad de que el sistema posea un método de calibración y manejo parecido a los dispositivos actuales (móviles con pantalla táctil, PDA´s, tablet PC´s, etc) donde se utilizan 4 o 6 puntos de calibración en la pantalla, a partir de los cuales se puede conseguir una matriz que convierta los puntos recogidos por la cámara a puntos en coordenadas de la pantalla.

Nuestro sistema como se indicó anteriormente deberá estar orientado a funcionar con aplicaciones multimedia, esto es, aplicaciones que usan simultáneamente diferentes formas de contenido como texto, sonido, imágenes, animaciones y video para informar o entretener.

Hasta ahora, los sistemas disponibles en las empresas entrevistadas funcionan con periféricos de uso común (ratón, teclado, etc), y para mantener la compatibilidad con ellos se ha desarrollado un driver de ratón que realiza esa función.

Este driver nos permitirá también realizar acciones sobre estas aplicaciones de manera mucho más intuitiva que mediante los periféricos convencionales.

Como por ejemplo “la metáfora de la pinza” en aplicaciones de manejo de fotografías. Esta metáfora consiste en trabajar con las fotos como si las tuviéramos en la mano, rotándolas con dos dedos, acercándolas separando esos dos dedos o moviéndolas “cogiéndolas con un dedo” y trasladándolas al lugar escogido entre otras cosas.

Además mediante el manejador anteriormente comentado es posible interactuar con más de un puntero en la pantalla y dará soporte a nuevas aplicaciones ya sea por medio de la librería o por medio de sockets.

Por último y para finalizar con este punto, se asignará a gente que colaborará en la creación del sistema para realizar las tareas de mantenimiento y/o reparación de posibles averías que se produzcan en el mismo.

3.3- Planificación y estimación de costes

Como en cualquier otra actividad de la vida cotidiana, previamente a realizar alguna tarea, se deben hacer estimaciones, lo más reales y aproximadas posible, sobre los costes que va a acarrear llevar a cabo dicha tarea.

Por ejemplo, cuando se le pide a un pintor que pinte una casa, lo normal es que la pinte previa aprobación de un presupuesto por parte del dueño de la casa. El pintor, para realizar dicho presupuesto, ha tenido que hacer una estimación del tiempo que va a estar pintando la casa en función del número de peones que lleve consigo, ha estimado el coste de los materiales que va a necesitar, ha calculado el salario que va a tener que pagar a los peones y a él mismo e incluso ha tenido en cuenta que tiene que pagar unos impuestos por ser autónomo. La suma de todos estos factores son los que componen el presupuesto, que como su propio nombre indica, es un supuesto que se realiza previamente a la faena y cuya función es valorar económica y temporalmente el coste de realizarla. Si este pintor comete cualquier tipo de error al hacer el presupuesto y el cliente lo acepta, estará obligado a pintar la casa incluso aunque dicha faena le repercuta pérdidas económicas. Por tanto, queda clara la importancia de ser lo más objetivos y detallistas posibles a la hora de presupuestar y planificar cualquier tarea.

Este proyecto informático se encuentra exactamente en la misma situación que el presupuesto del pintor. Requiere una planificación, una implantación de objetivos, intermedios a la finalización del proyecto, que sirvan de referencia, y una valoración económica teniendo en cuenta los recursos personales, hardware y software que se van a utilizar.

Como en todo proyecto es muy difícil hacer una aproximación exacta del coste antes de ponerse “manos a la obra”, e incluso una vez inmersos, pueden ocurrir imprevistos como la

indisposición de un empleado. Es por ello que una de las primeras fases de la planificación consista en la división del proyecto en pequeñas tareas. Tras la finalización de cada una de estas tareas, se comprobará si se están cumpliendo las estimaciones iniciales, haciendo las modificaciones pertinentes en caso de que no sea así. Con esto se quiere decir que la planificación de un proyecto no es estática e inmutable, sino que se va modificando conforme se cumplen o no las previsiones iniciales.

3.3.1- Recursos hardware y software necesarios

Los recursos utilizados pueden separarse principalmente en cuatro bloques: esqueleto físico, sistema de captura, sistema de visualización y aplicaciones software para el manejo. Dentro de cada uno de estos bloques se hará la separación entre hardware y software.

Cabe destacar que todo el software aquí nombrado funciona bajo el sistema operativo Microsoft Windows XP (primer recurso necesario).

3.3.1.1 Recursos del sistema de captura

Los recursos utilizados para la implementación del sistema de captura de imágenes son los siguientes:

Espejo:

Superficie lisa y brillante hecha de una placa de vidrio recubierta en su parte posterior de mercurio, acero u otro metal, que refleja los objetos. Un ejemplo visual del concepto comentado se muestra en la “Figura 35”.

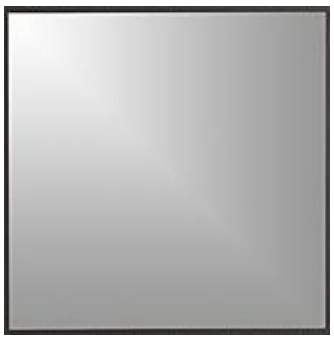


Figura 35. Espejo utilizado en el sistema.

Cámara:

Se pueden utilizar diferentes tipos de cámaras dependiendo del sistema de detección de puntos que se utilice en el sistema. Los tipos que hemos considerado más relevantes son:

- **Cámara Web:**

  Una **cámara web** o **web cam (“Figura 36”)** es una pequeña cámara digital conectada (normalmente mediante USB) a una computadora, la cual puede capturar imágenes y transmitirlas en directo, ya sea a una página web o a otra u otras computadoras de forma privada.

  Todas las cámaras digitales disponen de un sensor, normalmente CMOS. Estos sensores son sensibles a toda la luz visible y también a la luz infrarroja que es invisible. Para evitar que la parte infrarroja de la luz sature todos los colores generando una imagen irreal, el sensor lleva un filtro infrarrojo que deja pasar solamente la luz visible para el ojo humano.

  La modificación que se realizará se trata básicamente de eliminar ese filtro IR y sustituirlo por uno que elimine toda la luz visible y deje pasar solamente el infrarrojo, para ello usaremos se puede usar un negativo fotográfico velado.

Figura 36. Cámara web

- **Cámara de Infrarrojos:**

  Es un aparato que percibe la radiación infrarroja emitida de los cuerpos detectados y que la transforma en imágenes luminosas para ser visualizada por el ojo humano.

  En este apartado se hace una revisión de los principales sistemas de captura de tipo óptico, existentes actualmente en el mercado. Como esta lista es muy extensa, se ha reducido a los principales, mostrando la descripción de un sistema de cada unos de los siguientes tipos:

- Sistemas infrarrojos de alta precisión.
- Sistemas infrarrojos de bajo coste.

Sistemas Infrarrojos de alta precisión: ARTTrack & DTrack

El sistema ARTtrack & DTrack, es un sistema óptico de captura, que utiliza luz infrarroja (IR) para medir la posición de hasta 20 *targets* (más de 80 markers), en un volumen de captura previamente definido. Este sistema es capaz de ofrecer los resultados con un bajo retraso, entre 20 y 40 msec., lo cual permite realizar capturas en tiempo real.

La parte de este sistema que captura físicamente la información son las "**Cámaras infrarrojas ARTtrack1**" mostradas en la "Figura 37". Cada cámara lleva incorporados un grupo de LEDs, que se encargan de iluminar el área de captura con flashes de luz infrarroja. Por medio de la luz reflejada, las cámaras son capaces de reconocer markers retro-reflectantes y calcular su posición en 2-D con alta precisión.

Esta característica hace especialmente interesantes estas cámaras, puesto que se realiza un procesado interno en las mismas para interpretar la imagen y obtener los marcadores, no limitándose (como otras cámaras infrarrojas del mercado) a enviar la imagen al ordenador para dejarle a este todo el procesamiento.

Visto el sistema de captura ARTTrack & DTrack, se puede observar que sus características cubren prácticamente cualquier requisito normal de captura para un usuario: alta precisión, capacidad de capturar estructuras con 6DOF, fácil de calibrar, etc…. Pero aunque este sistema, y otros parecidos como por ejemplo los de la compañía Vicon, ofrezcan tan buenas características, su elevado precio los convierte en prohibitivos a la hora de integrarlos en un sistema de coste normal.

Figura 37. Cámaras infrarrojas ARTrack

Sistemas Infrarrojos de bajo coste: Natural Point.

Como solución a este problema, han surgido otros sistemas de captura ópticos en el mercado, que ofreciendo menores prestaciones, tienen un precio asequible para un usuario común. Este hecho permite acercar este tipo de tecnología al usuario común, sin ser necesarias instalaciones especializadas que ocupan mucho espacio y tienen un coste de decenas de miles de euros.

En este apartado destacan los dispositivos creados por la compañía Natural Point [24]: TrackIR, SmartNav y OptiTrack ("Figura 38"). La tecnología de los 3 se basa en el mismo principio que los sistemas anteriores, emitiendo luz infrarroja que es reflejada por unos marcadores colocados sobre el usuario. Pero en este caso ya no es posible capturar tantas estructuras con 6 DOF, ni con tanta precisión.

OptiTrack

Este es el dispositivo para usuarios "expertos" de la compañía Natural Point. El funcionamiento nuevamente es similar a los sistemas anteriores, aunque este dispositivo es muy parecido a las cámaras ARTTrack. Sus características técnicas principales son:

- Frecuencia de captura configurable entre 4 y 120 frames por segundo.
- Lentes intercambiables entre 35º y 60º de campo de visión horizontal.
- Sincronización entre múltiples cámaras.
- Resolución de imagen 355 x 288 píxeles.
- Potencia de los LEDs configurable.
- Conexión al PC por USB (hasta 4 cámaras por concentrador).

De las características, se observa que manteniendo un precio asequible (alrededor de 300,00€) este dispositivo es muy potente, y compatible con cualquier PC actual (por su interfaz USB).

La principal desventaja de este tipo de cámaras, con respecto a los sistemas profesionales (de mayor coste) se encuentra en el procesamiento. En este caso el interfaz software básico, solo captura 6DOF para la cabeza y el resto de puntos para cada frame por separado. Es decir, en su opción básica no ofrece un interfaz que permita seguir puntos, capturar varios bodies o calibrar un sistema con varias cámaras.

Recientemente Natural Point ha puesto a la venta un software que cumple algunas de estas características, con un precio moderado (alrededor de 2000,00€). Este software necesita al menos 3 cámaras para calcular posiciones 3D de los marcadores (frente a las 2 necesarias en el caso de ARTTrack), y es capaz de seguir nubes de puntos e identificar un pequeño número de bodies.

Figura 38. Cámara infrarroja Optitrack C-FLEX120

Pantalla acrílica:

Para la pantalla existen diferentes materiales utilizados para la construcción de mesas “Multi-Touch”, entre los más destacados podemos encontrar:

- **Cristal:** En física del estado sólido y química, un cristal es un sólido homogéneo que presenta una estructura interna ordenada de sus partículas reticulares, sean átomos, iones o moléculas. Material mostrado en la “Figura 39”.

  En un cristal, los átomos e iones se encuentran organizados de forma simétrica en celdas elementales, que se repiten indefinidamente formando una estructura cristalina. Un cristal suele tener la misma forma de la estructura cristalina que la conforma.

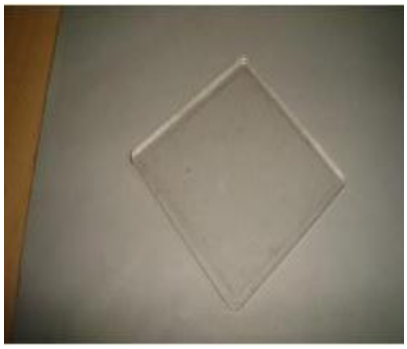


Figura 39. Vidrio común

- **Acrílico:** Dentro de los plásticos de ingeniería podemos encontrarlo como Polimetilmetacrilato, también conocido por sus siglas PMMA. El acrílico se obtiene de la polimerización del metacrilato de metilo y la presentación más frecuente que se encuentra en la industria del plástico es en gránulos ('pellets' en inglés) o en láminas. Los gránulos son para el proceso de inyección o extrusión y las láminas para termoformado o para mecanizado.

- **Policarbonato:** El policarbonato (“Figura 40”) es un grupo de termoplásticos fácil de trabajar, moldear y termoformar, y son utilizados ampliamente en la manufactura moderna. El nombre "policarbonato" se basa en que se trata de polímeros que presentan grupos funcionales unidos por grupos carbonato en una larga cadena molecular.

Figura 40. Acrílico.

Como conclusión se puede decir que el acrílico compite en cuanto a aplicaciones con otros plásticos como el policarbonato (PC), pero el acrílico se destaca frente a otros plásticos transparentes en cuanto a resistencia a la intemperie, transparencia y resistencia al rayado.

LEDs:

Es un dispositivo semiconductor (diodo) que emite luz dispersa de espectro reducido cuando se polariza de forma directa la unión PN del mismo y circula por él una corriente eléctrica. Este fenómeno es una forma de electroluminiscencia. Una imagen de ejemplo de un LED se puede observar en la “Figura 41”.

Figura 41. LED de infrarrojos

El color (longitud de onda), depende del material semiconductor empleado en la construcción del diodo y puede variar desde el ultravioleta, pasando por el visible, hasta el infrarrojo.

Los diodos emisores de luz que emiten luz ultravioleta también reciben el nombre de UV LED (***U****ltra****V****'*iolet **L**ight-**E**mitting **D**iode) y los que emiten luz infrarroja suelen recibir la denominación de IRED (***I****nfra-****R****ed* ***E****mitting* ***D****iode*). Estos diodos pueden combinarse formando tiras de LEDs, esto en ocasiones puede ser útil para propagar una señal a través de cualquier objeto. Estas tiras se pueden observar en la “Figura 42”.

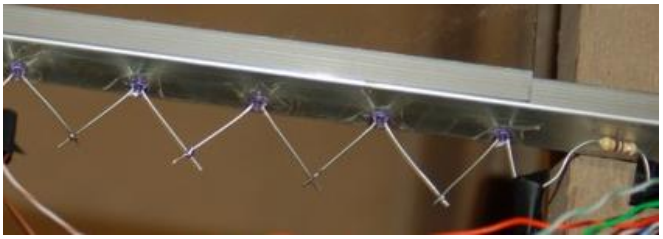


Figura 42. Tira de LEDs del marco de la pantalla

Cableado:

El cableado que será utilizado en el desarrollo del sistema será de cobre ya que su único es conducir electricidad. Esto es así debido a la excelente conductividad de este material ya su no menos importante propiedad, es aislante.

Resistencias:

Se denomina **resistencia eléctrica** la dificultad u oposición que presenta el componente al paso de una corriente eléctrica para circular a través de él. Ejemplos de diferentes resistencias se pueden ver en la “Figura 43”. En el Sistema Internacional de Unidades, su valor se expresa en ohmios, que se designa con la letra griega omega mayúscula, Ω.

Aunque pequeños, estos componentes de carbón y otros elementos resistivos son exactamente igual de importantes que cualquiera de los demás.

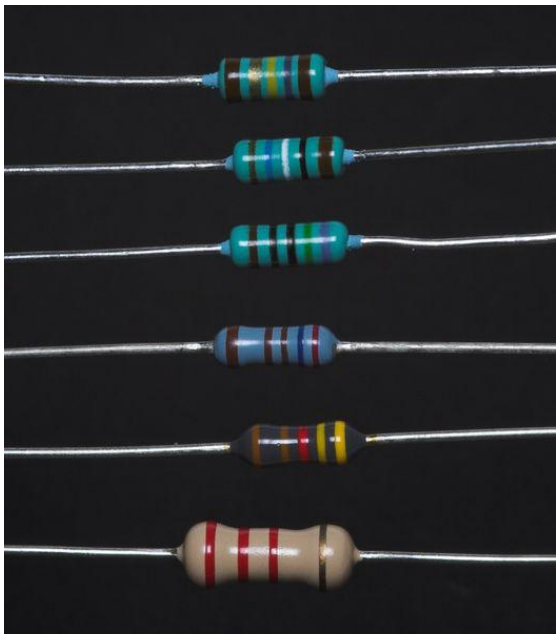


Figura 43. Resistencias de diferente resistividad

Fuente Alimentación:

Se denomina fuente de alimentación (“Figura 44”) al aparato, o parte del mismo, que hace de generador de tensión, es decir, que suministra la tensión e intensidad de corriente adecuadas para el funcionamiento de la instalación o del aparato en cuestión. Es un circuito que convierte la tensión alterna de la red industrial (220V) en una tensión prácticamente continua (voltaje necesario para el funcionamiento del sistema).

Figura 44. Ejemplo de Fuente de Alimentación

3.3.1.2 Recursos del sistema de visualización

Proyector:

Un **proyector de vídeo** o **cañón proyector (“Figura 45”)** es un aparato que recibe una señal de vídeo y proyecta la imagen correspondiente en una pantalla de proyección usando un sistema de lentes, permitiendo así visualizar imágenes fijas o en movimiento.

Todos los proyectores de vídeo utilizan una luz muy brillante para proyectar la imagen, y los más modernos pueden corregir curvas, borrones y otras inconsistencias a través de los ajustes manuales. Los proyectores de vídeo son mayoritariamente usados en salas de presentaciones o conferencias, en aulas docentes, aunque también se pueden encontrar aplicaciones para cine en casa. La señal de vídeo de entrada puede provenir de diferentes fuentes, como un sintonizador de televisión (terrestre o vía satélite), un ordenador personal, etc.

En la actualidad existen diferentes tipos de proyección asociadas a los sistemas multi-touch, los cuales comentamos a continuación:

Proyección: La imagen se proyecta desde delante de la pantalla.

Retro-Proyección: La imagen se proyecta desde detrás de la pantalla, utilizando una superficie de retro-proyección.

Figura 45. Proyector EIKI

Espejo: Ver punto anterior.

Pantalla de retro:

La pantalla de retro proyección se compone de dos elementos primarios. En primer lugar, la superficie interior de Fresnal, que dispersa y aumenta el brillo de la imagen a través de la pantalla. En segundo lugar, hay una superficie exterior lenticular, lo que contribuye a la integridad de la imagen, así como mejora el ángulo de visión. Esta capa exterior da a la pantalla un tacto rugoso.

Un ejemplo de este tipo de pantallas y de su funcionamiento se pueden observa a continuación, en la “Figura 46”.

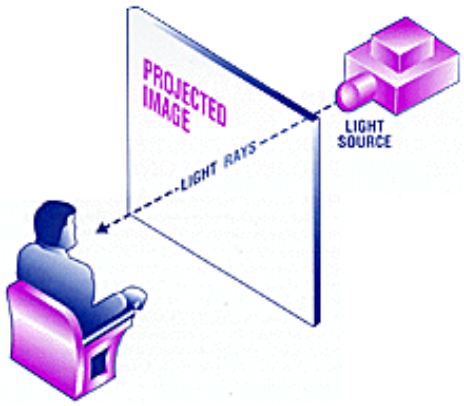


Figura 46. Ejemplo de retro proyección

3.3.1.3 Recursos de aplicaciones software

Los **Recursos Hardware** utilizados tanto para desarrollar tanto la librería de manejo de puntos detectados, como el manejador de los punteros es un PC de sobremesa con las características que se comentan a continuación:

Ordenador:	
Tipo de ordenador	Equipo multiprocesador ACPI
Sistema operativo	Microsoft Windows XP Professional
Placa base:	
Tipo de procesador	DualCore Intel Core 2 Duo, 2666 MHz (10 x 267)
Chipset de la Placa Base	Intel Lakeport-G i945G
Memoria del Sistema	1024 MB (DDR2-667 DDR2 SDRAM)
Tipo de BIOS	Award (03/26/07)
Monitor:	
Tarjeta gráfica	NVIDIA GeForce 8500 GT (512 MB)
Monitor Proview	769(N) [17" CRT] (11403961)
Almacenamiento:	
Controlador IDE	Intel(R) 82801G (ICH7 Family) Ultra ATA Storage Controllers - 27DF
Controlador IDE	Intel(R) 82801GB/GR/GH (ICH7 Family) Serial ATA Storage Controller - 27C0
Disco duro	ST3250820AS (250 GB, 7200 RPM, SATA-II)

Tabla 1: Recursos Hardware usados para la librería

Cabe mencionar que dado que la vida útil de un PC de desarrollo suele ser de 3 años, el coste del PC se ha divido para amortizarlo en esos 3 años, de los cuales se usará algo más de medio para el desarrollo de este proyecto.

Los **Recursos Software** utilizados para la realización de la aplicación son:

- Sistema Operativo Windows XP Professional sobre el cual se realizará la misma.
- Microsoft Visual Studio 2005, es un entorno de desarrollo integrado para sistemas operativos Windows. Mediante este se podrá realizar el desarrollo de las aplicaciones (librería y driver) necesarias para el funcionamiento del sistema.

3.3.1.4 Recursos del esqueleto físico

Aluminio Barras:

Se adquieren barras de aluminio para realizar el esqueleto de la mesa. Se decide la utilización de este tipo de material debido a su poco peso y su bajo coste. Estas barras como se comenta en el punto “4.2 Especificaciones del sistema” serán de la marca BOSH-REXROTH y poseerán un diámetro de 30x30 mm.

Aluminio Marco:

Para el marco también se optará por la utilización del aluminio por las mimas razones que en punto anterior. La única diferencia es que en este caso no serán barras sino planchas en forma de L.

Cobertura:

Para alejar de las manos de los futuros usuarios y evitar que entre luz que podría engañar a nuestro sistema se opta por recubrir la mesa con planchas de madera. La madera es lo suficientemente resistente para evitar que golpes fortuitos dañen los componentes que se encontrarán en el interior y además permite que todos los componentes del sistema estén recogidos.

3.3.1.5 Otros recursos

Además de los tres grandes bloques ya nombrados, también se han utilizado otros recursos, todos ellos software. Son los siguientes:

Generación de diagramas y planificación:

- Visual Paradigm for UML 6.1 Standard Edition
- Microsoft Project para la creación de los diagramas de Gantt.

Testeo del sistema:

- Aplicación de captura y manejo de puntos
- Aplicación OSG “Multi-Touch” y aplicación de prueba “Single-Touch”.

Documentación:

- Microsoft Office 2007

3.3.2- Estimación económica del hardware

Para instalar el sistema y comprobar su funcionamiento no es necesario comprar todo el sistema que se tendría que instalar en la empresa. Es por ello que se haga una separación entre el coste de la realización de este proyecto y el coste de su implantación en una empresa.

Por otro lado, como ya se ha comentado anteriormente, el coste invertido en este proyecto no es "real" ya que se han comprado materiales pensando en futuras aplicaciones de éstos y no únicamente basándonos en las necesidades, con el fin de minimizar el coste.

Se propondrá también otro presupuesto con el precio del material que habría sido suficiente para llevar cabo esta realización. En este punto se tendrá en cuenta que el sistema se vende de forma permanente y que, por tanto, no se le aplicará ningún tipo de amortización al material.

3.3.2.1- Coste real hardware mesa Multi-Touch

La siguiente tabla muestra una estimación del coste real del sistema:

Descripción	Precio Ud. (€)	Cantidad	Precio (€)
Proyector EIKI LC-XIP 2000	1500,00	1	1500,00
Cámara Optitrack FLEX C-120	300,00	1	300,00
Espejo	45,00	1	45,00
Pantalla de retro proyección Traulux	60,00	1	60,00
Pantalla de metacrilato	30,00	1	30,00
LEDs IR SFH485 5mm	1,43	100	143,00
Resistencias	0,10	20	2,00
Fuente Alimentación	20,00	1	20,00
Perfiles estructurales aluminio BOSH-REXROTH	27,00	1	27,00
Plancha contrachapado	8,00	1	8,00
Equipo multiprocesador ACPI	937,00 € con IVA	1	937,00
TOTAL			**3072,00**

Tabla 2: Coste real Hardware

Cabe mencionar que dado que la vida útil de un PC de desarrollo suele ser de 3 años, el coste del PC se ha divido para amortizarlo en esos 3 años (312.3 €/año), de los cuales se usarán unos 10 meses para el desarrollo de este proyecto (260,00€). Todos los componentes necesarios para la realización del sistema se adquieren para uso exclusivo del mismo.

3.3.2.2- Coste ideal hardware mesa Multi-Touch

Este punto se incluye para mencionar que se podría haber escogido un proyector de menor coste que el que se utilizó para la realización del proyecto. Se utilizó el EIKI ya que era el único del que se disponía en ese momento y la relación calidad-precio es muy buena. El proyector incluido en este punto es bastante más barato pero la calidad también es mucho menor.

La siguiente tabla muestra una estimación del coste ideal del sistema (proyector de menor coste):

Descripción	Precio Ud. (€)	Cantidad	Precio (€)
AIPTEK POCKET V1	350,00	1	350,00
Cámara Optitrack FLEX C-120	300,00	1	300,00
Espejo	45,00	1	45,00
Pantalla de retro proyección Traulux	60,00	1	60,00
Pantalla de metacrilato	30,00	1	30,00
LEDs IR SFH485 5mm	1,43	100	143,00
Resistencias	0,10	20	2,00
Fuente Alimentación	20,00	1	20,00
Perfiles estructurales aluminio BOSH-REXROTH	27,00	1	27,00
Plancha contrachapado	8,00	1	8,00
Equipo multiprocesador ACPI	937,00 € con IVA	1	937,00
TOTAL			**1922,00**

Tabla 3: Coste "ideal" Hardware

3.3.3- Estimación económica del software

El coste de cada uno de los programas utilizados en el desarrollo del proyecto es el siguiente:

Descripción	Precio (€)
Microsoft Windows XP Service Pack 3	0,00
Microsoft Visual Studio 2005 Standard Edition	259,00
Visual Paradigm for UML 6.1 Standard Edition	0,00
Microsoft Office 2007	0,00
Optitrack SDK	0,00

Tabla 4: Estimación coste software

La explicación de algún precio que puede parecer incorrecto en un principio es la siguiente:

- *Microsoft Windows XP Service Pack 3:* incluido en el PC que se utilizará primeramente para desarrollar el proyecto y más tarde se utilizará para contener el código desarrollado.

- Optitrack SDK: Es una librería gratuita de manejo de las cámaras infrarrojas de Natural Point.

3.3.4 Coste general del sistema

Generalmente el coste temporal asociado a los proyectos se realiza mediante una única estimación basada en COCOMO II. En este caso y al tratarse de un proyecto en que el subsistema Hardware es igual de importante que el Software, la estimación se realizará utilizando una combinación de dos sistemas: Analogía de proyectos y Evaluación de expertos para la parte de Hardware, y COCOMO II para la de Software.

3.3.4.1- Coste basado en "Analogía de proyectos" y "evaluación de expertos"

Este tipo de sistemas se utiliza en la estimación del tiempo de desarrollo de un proyecto Hardware

Por un lado, se efectuará la estimación mediante la **Evaluación de expertos**. Esta se basa en la comunicación con empresas especializadas en el sistema a realizar. A estas empresas se les comenta el tipo de sistema a desarrollar y después de un estudio concluirán estimando un tiempo de evaluación de los diferentes métodos existentes para su realización y un tiempo estimado de montaje.

Por otro lado, se realizará la estimación mediante el sistema de **Analogía de proyectos**. Este consiste en evaluar el tiempo de desarrollo necesario para proyectos similares al que se desea realizar. Para esta parte también sería conveniente contar con los expertos comentados en el punto anterior.

La evaluación del sistema a elaborar se realizó a través de varias reuniones con expertos en este tipo de sistemas. Estas reuniones se realizaron de forma individual para que no influyera la opinión de alguno de ellos en los demás. Los expertos con los que se consultaron las citas fueron cuatro gerentes de las empresas GOTESA (suministrador de material electrónico de ARTEC), ITRONIX

(empresa importadora de nuevas tecnologías, especializadas en electrónica de consumo e informática), VIATEK (empresa dedicada a la importación de componentes electrónicos con experiencia de varios años en el sector), MISANA (empresa dedicada a la investigación, diseño, fabricación, y comercialización, así como servicio post venta y de mantenimiento de equipos de electrónica industrial a medida).

La media del tiempo estimado para las dos estimaciones realizadas fue de 3 meses en total. Este tiempo se consideró suficiente por los desarrolladores del sistema por lo que se aceptó.

3.3.4.2- Coste basado en COCOMO 2

COCOMO II [19] es una revisión de COCOMO 81 [18] que refleja los cambios que ha habido en las prácticas de desarrollo profesional de software desde 1990. Estos cambios incluyen la migración de los desarrolladores de ”mainframes” a equipos de escritorio, el incremento de la reutilización de software existente, el desarrollo basado de componentes software ”off–the–shelf” y el mayor uso de tiempo para gestionar y diseñar el software que el usado para su creación.

COCOMO II es un modelo de tres niveles que permite hacer estimaciones cada vez más detalladas:

- **Nivel inicial de prototipado:** indicado para proyectos con prototipado o que hacen uso intensivo de la reutilización y tienen en cuenta el uso de herramientas CASE. Se basa en la estimación de la productividad del desarrollador en puntos objeto/mes.
- **Nivel inicial de diseño:** indicado para fases iniciales de desarrollo cuando los requerimientos han sido establecidos.
- **Nivel post-arquitectura:** el modelo incluye el desarrollo actual y el mantenimiento de un producto software, apropiado cuando un ciclo de vida del software ha sido desarrollado. En nuestro caso, en función de las características de nuestro proyecto, utilizamos el nivel inicial de diseño.

A continuación se va a realizar una estimación del coste de personal necesario para el desarrollo del proyecto

La ecuación de esfuerzo para este nivel es

$$E = A \times KLDC^{B} \times M$$

Donde:

- **E** es una estimación de los recursos humanos disponibles para la realización del proyecto.
- **A** es una constante cuyo valor según la USC (University of Southern California) es de 2,94.
- **B** es un exponente que se calcula teniendo en cuenta los siguientes cinco factores:
  - **PREC:** predecesores
  - **FLEX:** flexibilidad en el desarrollo
  - **RESL:** la resolución de la arquitectura / riesgo
  - **TEAM:** la cohesión del equipo
  - **PMAT:** la madurez del proyecto

  Estos cinco factores toman valores entre 0 y 5, siendo 0 un valor muy bueno y 5 muy malo. B es la centésima parte de la suma de estos factores.

  Un ejemplo de estos factores de escala propuestos por la "University of Southern California" (USC) se puede observar en la Tabla 5. La tabla 6 contiene los factores que se consideraron oportunos para la realización del proyecto actual:

	Muy Bajo	Bajo	Nominal	Alto	Muy Alto
PREC	6.20	4.96	3.72	2.48	1.24
FLEX	5.07	4.05	3.04	2.03	1.01
RESL	7.07	5.65	4.24	2.83	1.41
TEAM	5.48	4.38	3.29	2.19	1.1
PMAT	7.80	6.24	4.68	3.12	1.56

Tabla 5: Factores de escala USC-COCOMO II

El cálculo es como sigue:

Factor	Valor
PREC	3
FLEX	2
RESL	3
TEAM	0
PMAT	5
TOTAL	**13**

Tabla 6: Factores escogidos para el cálculo

$$B = PREC + FLEX + RESL + TEAM + PMAT \quad (1.1)$$

- **M** son 17 multiplicadores de esfuerzo que siguen una escala desde 0 a 6. Estos multiplicadores están detallados en la tabla. Para el caso del diseño temprano, que es el que más se ajustaba a nuestro problema, ya que quisimos hacer la estimación en la fase inicial del proyecto y el COCOMO II sugiere que muchos de estos multiplicadores no pueden estimarse, por lo que los reduce a 7, que fueron los que utilizamos, los cuales comentamos a continuación:
    - **RCPX:** Confiabilidad y complejidad del producto
    - **RUSE**: Nivel de reutilizabilidad del desarrollo
    - **PDIF**: Dificultad de uso de la plataforma
    - **PERS**: Capacidad del personal de desarrollo
    - **PREX**: Experiencia del personal de desarrollo
    - **FCIL**: Facilidades de desarrollo
    - **SCED**: Exigencias sobre el calendario.

  Donde

$$M = PERS * RCPX * RUSE * PDIF * PREX * FCIL * SCED \quad (1.2)$$

En la tabla siguiente se pueden observar los factores de los multiplicadores de esfuerzo proporcionados por la USC- COCOMO II.

	Extr. Baja	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extr. Alto
RCPX	0,73	0,81	0,98	1	1,3	1,74	2,34
RUSE			0,95	1	1,07	1,15	1,24
PDIF			0,87	1	1,29	1,81	2,61
PERS	2,12	1,62	1,26	1	0,83	0,63	0,5
PREX	1,59	1,33	1,12	1	0,87	0,71	0,62
FCIL	1,43	1,3	1,1	1	0,87	0,73	0,62
SCED		1,43	1,14	1	1	1	

Tabla 7: Multiplicador de Esfuerzo USC-COCOMO II

A estos multiplicadores se les asigna un valor en función de la siguiente tabla:

Descripción	Valor
Actúa negativamente sobre el proyecto	> 1
No interfiere con el proyecto	= 1
Actúa favorablemente sobre el proyecto	< 1

Tabla 8: Información de asignación de valores a los multiplicadores.

Los valores seleccionados para cada multiplicador para el sistema tipo interactivo basado en tecnología “Multi-Touch” son los siguientes:

Multiplicador	Valor
RCPX	1
RUSE	1,1
PDIF	1
PERS	0.83
PREX	1,1
FCIL	1
SCED	0,83
M	0,8335

Tabla 9: Valores asignados a los multiplicadores.

La ecuación de cálculo de la duración será:

$$D = \left(3.67 * E^{(0.28)+0.2*(B-1.01))}\right) * SCED\%/100 \quad (1.3)$$

Donde:

E será el esfuerzo de desarrollo sin el multiplicador SCED.

SCED % será el porcentaje de reducción o incremento en el calendario nominal del proyecto.

Esfuerzo:

Para calcular el esfuerzo en primer lugar hay que estimar el valor para cada uno de los multiplicadores de esfuerzo. Para ello utilizaremos la Tabla 7, que incluye los valores que utiliza la USC, en su programa de estimación USC-COCOMOII, para representar la Tabla 9. Aplicando la ecuación 1.2:

M = 1 *1.1 *1 *0.83 *1.1 *1 *1.1
M = 0.8335

Aplicando la ecuación del cálculo de la ecuación 1.1, utilizando los valores de la Tabla 6:

B = 3 + 2 + 3 + 0 + 3 = 10 / 100
B= 1,00

Y por último aplicando la ecuación del esfuerzo anteriormente comentada:

E = 2.94 * 3.000 * 0.8335
E = 7.3514

Duración:

Aplicando la ecuación del cálculo de la duración de la fórmula 1.3:

$D = (3.67 * 7.3514^{(1.00–1.01)}) * 1.00$
tomando SCED % como 100
D = 6.4 meses

3.3.4.3 Conclusión Final

Para concluir con la estimación del coste de desarrollo del sistema se realizara la suma de los subsistemas que lo componen (SW y HW) ya que estas no se pueden realizar en paralelo. Primero se deberá realizar el subsistema Hardware y una vez esta esté concluida se comenzará con el Software.

El resultado final es:

Tiempo Desarrollo = Tiempo SW + Tiempo HW = 6,4 + 3 = 9,4 meses

Contando que el coste total asociado a cada empleado es de 2.500,00 (1750,00€ de salario más el 30% de costes sociales) y que es un único trabajador, se quedaría en un coste de 23.500,00€.

3.3.5- Costes totales

En esta sección se calcularán los costes del sistema “Multi-Touch”. Por un lado se hará el coste real que ha tenido desarrollar el sistema. Por otro lado se repetirán los cálculos pero suponiendo que se compran los materiales “ideales”.

3.3.5.1- Coste real de desarrollo

La siguiente tabla muestra los factores que se tiene en cuenta para calcular el coste total del desarrollo con los materiales que se han utilizado:

Descripción	Subtotal (€)
Recursos hardware	3072,00
Recursos software	259,00
Recursos de personal	23500,00
TOTAL	**26831,00**

Tabla 10: Coste real desarrollo

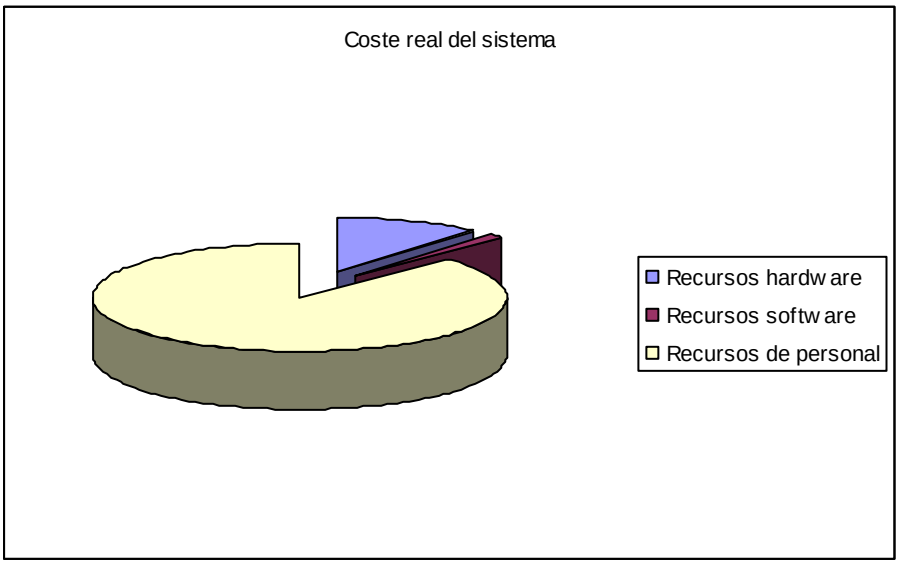


Figura 47. Gráfica del Coste real del desarrollo

Mediante la gráfica anterior ("Figura 47") se puede observar a simple vista que el mayor coste viene de los recursos de personal. Estos serán aproximadamente un 88% de los costes del desarrollo. Los recursos hardware implicarán más o menos un 10% del mismo y los software sobre un 2%.

3.3.5.2- Coste "ideal" de desarrollo

La siguiente tabla muestra el coste que habría tenido el desarrollo del SMG-RFID en caso de haber utilizado los elementos nombrados en el apartado "3.3.2.1- Coste ideal hardware mesa Multi-Touch":

Descripción	Subtotal (€)
Recursos hardware	1922,00
Recursos software	259,00
Recursos de personal	23500,00
TOTAL	**25681,00**

Tabla 11: Coste ideal desarrollo

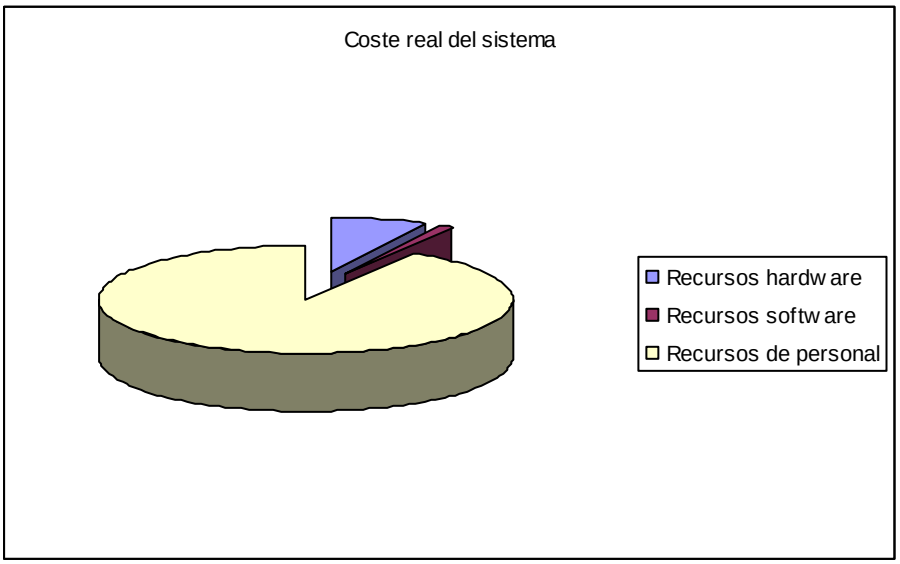


Figura 48. Coste ideal del desarrollo

Examinando la gráfica anterior ("Figura 48") se llega a la misma conclusión que el punto de "3.3.5.1 Coste real del desarrollo". La única diferencia es que en este caso. Los recursos de personal serán un 91%, lo recursos hardware serán un 7% del total y los software un 2%.

3.3.5- Identificación de hitos

Los hitos son objetivos intermedios en el proceso de desarrollo del proyecto, y constituyen los pasos previos que conducen a la meta final. Podemos identificar los siguientes hitos en el presente proyecto:

- *Hito H1*: Finalización de la fase de "Búsqueda de información".
- *Hito H2:* Finalización de la fase de "Análisis de la información".
- *Hito H3:* Finalización de la fase de "Análisis del sistema".
- *Hito H4:* Finalización de la fase de "Diseño".
- *Hito H5:* Finalización de la fase de "Implementación".
- *Hito H6:* Finalización de la fase de "Pruebas del sistema".
- *Hito H7:* Finalización de la fase de "Documentación".

3.3.6- Descomposición del trabajo

Se ha descompuesto cada uno de estos hitos en pequeñas tareas para un cálculo temporal [20] más realista. Ésta sección se basa en la filosofía “divide y vencerás”. La siguiente tabla ilustra la descomposición realiza en el sistema.

Para ver una separación de estas subtareas gráficamente se utilizará un diagrama de Gantt (Tabla 13). El proyecto aun habiendo sido realizado por una persona, se paralelizará en la medida de lo posible. Esto es, utilizando las mañanas para realizar el análisis y el diseño del subsistema hardware, mientras que las tardes se realizará el análisis y diseño del subsistema software. La implementación, en cambio, sí que debe realizarse de forma secuencial ya que se necesita el hardware para poder realizar el software.

Si se hubiese dispuesto de más recursos se habría podido adoptar una mejor técnica de optimización con el objetivo de realizar de forma paralela aquellas actividades que en este caso no ha sido posible.

Se puede concluir que cualquier retraso en cualquiera de las tareas implicaría un retraso en toda la realización del sistema, que en el caso de tratarse de un proyecto empresarial, a buen seguro estaría penado económicamente. En este caso se corre el riesgo de no entrar en el plazo de la convocatoria elegida.

ID	TAREAS	DIAS
1	**T1: BÚSQUEDA DE INFORMACIÓN**	**10**
2	1.1- Existencia de sistemas con funcionalidades Multi-Touch	**2**
3	1.2- Información sobre las diferentes tecnologías Multi-Touch	**3**
4	1.3- Información sobre los componentes utilizados en las diferentes tecnologías	**2**
5	1.4- Información sobre intercambio de información	**1**
6	1.5- Información sobre el software disponible en el manejo de puntos	**1**
7	1.6 Aspectos legales sobre las tecnologías Multi-Touch	**1**
8	**T2: ANÁLISIS DE LA INFORMACIÓN ENCONTRADA**	**6**
9	2.1- Análisis y comparación de las diferentes tecnologías existentes	**2**
10	2.2- Análisis y comparación de los componentes utilizados en cada tecnología	**2**
11	2.3- Análisis y comparación de las diferentes arquitecturas de intercambio de información	**1**
12	2.4 Análisis y comparación del software encontrado para el manejo de puntos	**1**
13	**T3: ANÁLISIS DEL SISTEMA**	**10**
14	**3.1 Software**	**10**
15	3.1.1- Recopilación y estudio de requisitos	**4**
16	3.1.2- Análisis de casos de uso y definición de roles	**4**
17	3.1.3- Prueba de entornos de programación donde desarrollar el sistema	**1**

18	3.1.4- Diagrama de secuencia general del sistema	**1**
19	**3.2 Hardware**	**10**
20	3.2.1- Recopilación y estudio de requisitos	**3**
21	3.2.2- Estudio de alternativas hardware (visualización, captura, esqueleto físico)	**7**
22	**T4: DISEÑO**	**12**
23	**4.1 Software**	**12**
24	4.1.1 Diseño de la arquitectura general del sistema	**6**
25	4.1.2 Propuestas de las posible librerías a utilizar (incluida la propia)	**3**
26	4.1.3 Propuesta de la comunicación Cliente-Servidor	**1**
27	4.1.3 Propuesta de la interfaz gráfica a desarrollar	**2**
28	**4.2 Hardware**	**12**
29	4.2.1 Propuesta de la tecnología a utilizar	**6**
30	4.2.2 Propuesta de los componentes a utilizar en captura y visualización (tecnología propuesta)	**3**
31	4.2.3 Propuesta del diseño del esqueleto físico del sistema	**3**
32	**T5: IMPLEMENTACIÓN**	**107**
33	**5.1 Software**	**83**
34	5.1.1- Instalación y configuración del software de programación	**1**
35	5.1.2- Implementación de la interfaz de la aplicación	**6**
36	5.1.3- Implementación de la aplicación de captura y manejo de puntos	**58**
37	5.1.4- Implementación de la arquitectura Cliente-Servidor	**5**
38	5.1.5- Implementación de la aplicación de prueba del manejo de puntos	**7**
39	5.1.6- Implementación de la aplicación de prueba con OpenGL	**6**
40	**5.2 Hardware**	**24**
41	5.2.1- Fabricación de la pantalla	**9**
42	5.2.1- Fabricación de la cobertura de la mesa y el marco de la pantalla	**9**
43	5.2.3- Fabricación del esqueleto físico del sistema	**6**
44	**T6: PUEBAS DEL SISTEMA**	**20**
45	6.1- Montaje físico del sistema	**2**
46	6.2- Pruebas de funcionamiento	**8**
47	6.3- Modificación de la implementación	**6**
48	6.4- Pruebas de funcionamiento	**4**
49	**T7: DOCUMENTACIÓN**	**25**
	TOTAL	**190**

Tabla 12: Descomposición en tareas

Id	Nombre de tarea	Duración
1	**T1: BUSQUEDA DE INFORMACIÓN**	10 días
2	1.1- Existencia de sistemas con funcionalidades Multi-Touch	2 días
3	1.2- Información sobre tecnologías Multi-Touch	3 días
4	1.3- Información sobre los componentes de la tecnología	2 días
5	1.4- Información sobre las posibilidades en la arquitectura C/S	1 día
6	1.5- Información sobre software disponible para la captura y manejo de puntos	2 días
7	**T2: ANÁLISIS DE LA INFORMACIÓN ENCONTRADA**	6 días
8	2.1- Análisis y comparación de las diferentes tecnologías existentes	2 días
9	2.2- Análisis y comparación de los componentes utilizados en cada tecnología	2 días
10	2.3- Analisis y comparación de la diferentes arquitecuras C/S	1 día
11	2.4- Análisis y comparación del sw encontrado para la detección y el manejo de	1 día
12	**T3: ANÁLISIS DEL SISTEMA**	10 días
13	3.1- Hardware	10 días
14	3.1.1- Recopilación y estudio de requisitos	3 días
15	3.1.2- Estudio de alternativas hardware(captura,visualización, esqueleto)	7 días
16	3.1- Software	10 días
17	3.1.1- Recopilación y estudio de requisitos	4 días
18	3.1.2- Análisis de casos de uso y definición de roles	4 días
19	3.1.3- Prueba de entornos de programación donde desarrollar el sistema	1 día
20	3.1.4- Diagramas de secuencia del sistema	1 día
21	**T4: DISEÑO**	12 días
22	4.1- Hardware	12 días
23	4.1.1- Propuesta de la tecnología a utilizar	6 días
24	4.1.2- Propuesta de los componentes a utilizar en captura y visualización	3 días
25	4.1.3- Propuesta del diseño del esqueleto físico	3 días
26	4.2- Sofware	12 días
27	4.2.1-Diseño de la arquitectura general del sistema	6 días
28	4.2.2- Propuestas de las posibles librerías a utilizar (incluida propia)	3 días
29	4.2.3- Propuesta comunicación Cliente-Servidor	1 día
30	4.2.4- Propuesta interfaz gráfica a desarrollar	2 días
31	**T5: IMPLEMENTACIÓN**	107 días
32	5.1- Hardware	24 días
33	5.1.1- Fabricación de la pantalla	9 días
34	5.1.2- Fabricación de la cobertura y el marco	9 días
35	5.1.3- Fabricación del esqueleto físico del sistema	6 días
36	5.2- Software	83 días
37	5.2.1- Instalación y configuración del sw de programación	1 día
38	5.2.2- Implementación de la interfaz de la aplicación	6 días
39	5.2.3- Implementación de la aplicación de captura y manejo de puntos	58 días
40	5.2.4- Implementación de la arquitectura Cliente-Servidor	5 días
41	5.2.5- Implementación de la aplicación de pruebadel manejo de puntos	7 días
42	5.2.6- Implementación de la aplicación de prueba con OpenGL	6 días
43	**T6: PRUEBAS DEL SISTEMA**	20 días
44	6.1- Montaje físico del sistema	2 días
45	6.2- Pruebas de funcionamiento	8 días
46	6.3- Modificación de la implementación	6 días
47	6.4- Pruebas de funcionamiento	4 días
48	**T7: DOCUMENTACIÓN**	25 días

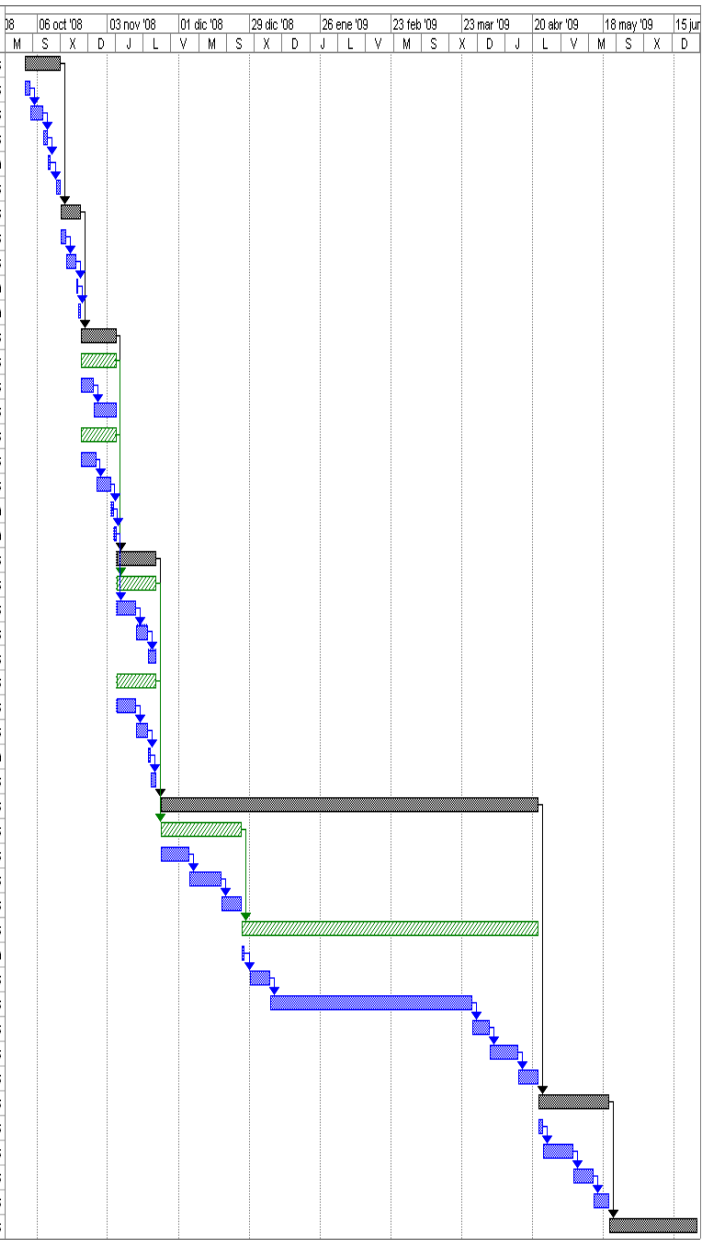


Tabla 13: Diagrama de Gantt estimado

4.- Desarrollo del proyecto

En este capítulo especificaremos cómo hemos diseñado y realizado el proyecto, teniendo en cuenta los requisitos acordados previamente con el cliente. Estos requisitos fueron detalladamente expuestos en el capitulo anterior. Todo proyecto de ingeniería del software tiene tres etapas importantes, y que es conveniente abordarlas de una manera adecuada, sin precipitarse en la codificación que finalmente realizaremos, efectuando los cambios necesarios para ir ajustando el proceso. Estos pasos son los siguientes:

- ***Análisis***: una vez analizados y acordados los requisitos del sistema, se hace una primera aproximación a la visión general del sistema. Esta se basará en el conocimiento previo conocido por otros grupos de investigación (referencias) y con las que veremos más claramente la complejidad real del sistema.

- ***Diseño***: a partir de los primeros resultados obtenidos en el apartado anterior, procederemos a la división en clases del subsistema software del sistema. Esto es así dado que vamos a realizar un proyecto orientado a objetos. Estas clases serán las que definan más concretamente el comportamiento del sistema, y dejarán claro como se realizará la tarea a la hora de programar. En este punto también se realizaran los primeros esbozos del subsistema físico del mismo, donde se especificará la forma, el tamaño y demás características del sistema.

- ***Implementación***: partiendo de las dos fases anteriores, ya se debe tener una idea muy clara de cómo afrontar la programación y la posterior puesta en marcha del sistema. De la misma forma, en este punto se debe tener un pensamiento muy claro y conciso de cuál será el resultado final del subsistema físico del sistema. Se aprovechará este apartado para hacer hincapié en las dificultades que han surgido durante la fase de programación/construcción.

Es importante destacar que cada pequeño fallo o despiste que haya en cualquiera de las secciones será arrastrado inevitablemente a las siguientes fases del desarrollo.

Es por esto que esta fase del proyecto debe realizarse muy cuidadosamente y con el suficiente tiempo como para poder tener la certeza de no haber cometido fallos que vayan a desencadenar un fallo en cascada que, o bien complicará en gran medida la implementación del proyecto, o bien conducirá a un error que será encontrado en la puesta en marcha del sistema, con la consecuente pérdida de tiempo en su análisis, localización y reparación.

Por otro lado, un mal diseño del sistema puede no conducir a ningún error catastrófico, pero sí a una gran penalización en las prestaciones que se esperan del producto obtenido.

4.1- Análisis

En este punto se dividirá el sistema en dos subsistemas claramente diferenciadas:

- La primera parte a comentar será el subsistema software. El objetivo de esta actividad, común tanto para análisis estructurado como para análisis orientado a objetos, es facilitar el análisis del sistema llevando a cabo la descomposición del mismo en subsistemas. Con el tiempo ha surgido un estándar industrial, conocido como Unified Modeling Language o UML, desarrollado por Booch [21], Se decidió utilizar este estándar para modelar esta parte del sistema. La decisión de emplear UML vino avalada porque reúne una colección de las mejores prácticas en la ingeniería, las cuales han sido usadas con éxito para modelar sistemas grandes y complejos. En el apartado de diseño veremos el UML desarrollado para este sistema.

- La segunda parte será el subsistema hardware. En este se realizará un estudio en el que se tendrá en cuenta la información recabada en la parte hardware del “Estado del arte” y los requisitos exigidos por los clientes en las entrevistas. Como resultado se obtendrán las tecnologías que más se aproximen a cumplir esos requisitos. Además de todo esto se complementará el análisis con las ventajas e inconvenientes (en referencia al sistema a desarrollar) de cada una de ellas.

4.1.1- Casos de uso

Para esto vamos a utilizar un método de la ingeniería del software llamado “Casos de Uso del Sistema”, donde se puede definir de manera muy abstracta las funcionalidades que va a tener cada participante en el sistema. De éste método se van a utilizar los siguientes componentes:

- **Actores**: entidad que utiliza alguno de los casos de uso del sistema (representado con el dibujo de una persona).

- **Casos de uso**: denota un requisito funcional del sistema (representado con los círculos).

- **Relaciones**: especifica que actuar puede llevar a cabo cada caso de uso (representado con líneas).

- **Generalizaciones**: se refiere a la generalización de los distintos actores que puede llevarse a cabo (representado por flechas).

El proyecto a realizar no es una típica aplicación llena de funcionalidades representadas en un interfaz, pudiéndose detallar en gran medida en este apartado. El único lugar donde se pueden encontrar algunos es en la librería realizada en el proyecto. Esta librería posee una interfaz que se utilizará como API de la mesa “Multi-Touch”. Esta tendrá los diferentes ajustes que a un usuario puede interesar modificar.

A continuación, se dará una breve descripción de los Casos de Uso que se han encontrado, los cuales están representados en el diagrama que se puede encontrar en la “Figura 49”:

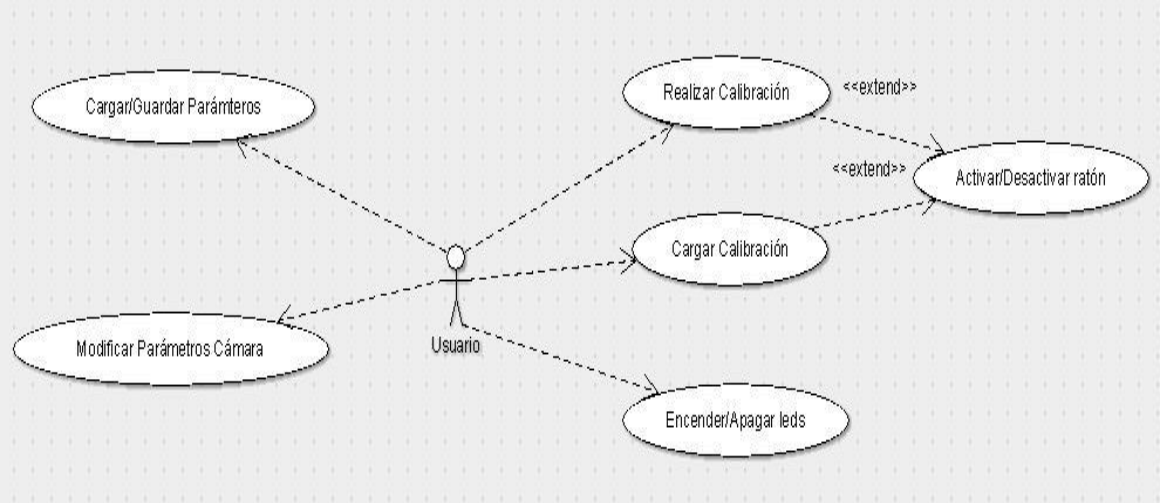


Figura 49. Diagrama de casos de uso

Una vez realizado el diagrama general de Casos de uso se va a proceder a realizar una breve explicación de cada uno de ellos:

Caso de uso: Realizar Calibración

Actores: Usuario

Resumen: El usuario calibra el sistema utilizando el método proporcionado por la aplicación. Este procedimiento es, como hemos comentado anteriormente, presionando con el dedo sobre las cruces que aparecerán en la pantalla.

Curso normal de los eventos

Actores	Sistema
1. Este caso de uso comienza cuando el usuario decide calibrar la pantalla.	
2. El usuario toca la pantalla en el centro de la cruz.	
	3. El sistema captura ese punto como un punto de calibración.

Figura 50. Caso de uso “Realizar Calibración”

Se repetirán los pasos 2 y 3 hasta que se toquen todos los puntos necesarios para la calibración del sistema implementado. Una vez esto ocurra, el sistema utilizará estos para pasar los puntos captados por la cámara a coordenadas de la pantalla.

Caso de uso: Cargar Calibración

Actores: Usuario

Resumen: El usuario ya ha realizado la calibración de la pantalla en una ejecución anterior de la aplicación y desea cargar esos valores para la ejecución actual.

Curso normal de los eventos

Actores	Sistema
1. El caso de uso comienza cuando el usuario hace click sobre el botón “Cargar Calibración” de la interfaz.	
	2. El sistema carga la calibración guardada anteriormente.
3. El usuario comienza el uso del sistema ya calibrado	

Figura 51. Caso de uso "Cargar Calibración"

Cursos alternos

- **Paso 2:** Si no existe ninguna calibración guardada anteriormente, el sistema nos alertará de que no ha sido posible realizar la carga.

Caso de uso: Activar/Desactivar ratón

Actores: Usuario

Resumen: El usuario (habiendo realizado previamente la calibración) activa o desactiva el control del manejador del ratón de Windows.

Curso normal de los eventos

Actores	Sistema
1. El caso de uso comienza cuando el usuario hace “click” en el Text Box “Activar/Desactivar mouse”	
	2. El sistema comenzará o dejará de recoger los eventos del ratón para utilizarlos en la aplicación.

Figura 52. Caso de uso "Act/Desact. Ratón"

Cursos alternos

- **Paso 2:** Si aún no se ha producido la calibración del sistema. Saltará un aviso en el que se dirá que antes de activar el ratón se debe realizar la calibración.

Casos de uso relacionados

- **Extiende** “Realizar Calibración”.
- **Extiende** “Cargar Calibración”.

Caso de uso: Modificar Parámetros Cámara

Actores: Usuario

Resumen: El usuario modifica los parámetros de la cámara (umbral, intensidad de los LEDs, FR o exposición) dependiendo de las condiciones en las que vaya a trabajar con el sistema. Existen unos valores para funcionamiento estándar asignados por defecto.

Curso normal de los eventos

Actores	Sistema
1. Este caso de uso comienza cuando el usuario modifica alguno de los valores de la cámara mediante los sliders de la interfaz.	
	2. El sistema modificará internamente estos valores en la cámara.

Figura 53. Caso de uso "Modificar Parámetros cámara"

Caso de uso: Cargar/Guardar Parámetros Cámara

Actores: Usuario

Resumen: El usuario carga/guarda los parámetros de la cámara (umbral, intensidad de los leds, FR y exposición) desde/a un fichero de texto.

Curso normal de los eventos

Actores	Sistema
1. El caso de uso comienza cuando el usuario hace “click” en “Cargar Configuración” o “Guardar Configuración”.	
	2. El sistema cargará/guardará la configuración que exista en ese momento en el/la fichero/aplicación.
3. El usuario proseguirá con la ejecución normal de la aplicación.	

Figura 54. Cargar/Guardar parámetros cámara

Cursos alternos

Paso 2: Si al realizar la carga de la configuración esta no existe. El sistema informará mediante un mensaje de alerta que antes de realizar una carga de la configuración se debe haber guardado esta.

4.1.2 Diagrama de clases

El objetivo principal de este modelo es la representación de los aspectos estáticos del sistema, utilizando diversos mecanismos de abstracción. El diagrama de clases recoge las clases de objetos y sus asociaciones. En este diagrama se representa la estructura y el comportamiento de cada uno de los objetos del sistema y sus relaciones con los demás objetos, pero no muestra información temporal.

A continuación se detalla el diagrama de clases del sistema (“Figura 55”):

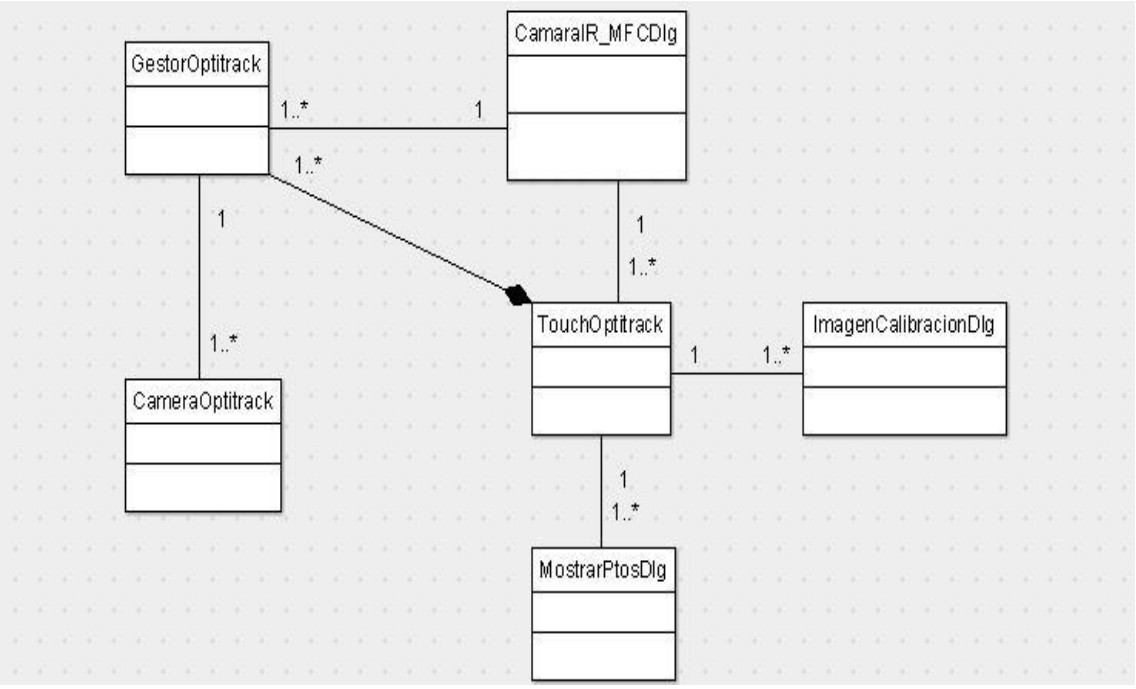


Figura 55. Diagrama de clases general

Después de observar el diagrama de clases del sistema, se realiza una breve explicación de las clases más relevantes.

CamaraIR_MFCDlg: Es la clase donde se codifica la API con la que el usuario interacciona con el sistema ajustando los parámetro como crea conveniente, calibrando la pantalla, etc…

Esta clase únicamente se relaciona con TouchOptitrack esto es así para independizar lo máximo posible el subsistema visual del subsistema lógico del sistema.

Así mismo, esta también será la encargada de codificar el servidor TCP que poseerá la aplicación para comunicarse con las demás aplicaciones cliente vía sockets..

TouchOptitrack: Esta clase será la única a la que nuestras interfaces tendrán acceso. Esta se encargará de manejar la lógica de nuestra aplicación. Utilizará la lista de puntos obtenidos mediante la clase “CameraOptitrack” para satisfacer las demandas del dialogo que esté usando esta clase en este momento. Lo que conseguimos separando los diálogos de la lógica es poder utilizar esta lógica con otras aplicaciones sin tener que realizar ninguna modificación en el código.

MostrarPtosDlg: Es una clase que se utiliza para comprobar que la calibración se ha realizado correctamente. En ella, una vez realizada la calibración, cuando se produzca un contacto en la pantalla, aparecerá bajo la superficie de contacto una esfera que representará esa superficie. Ésta aumentará o disminuirá dependiendo del tamaño del área de contacto que capte la cámara.

ImagenCalibracionDlg: Este clase es el dialogo de calibración del sistema. Al realizar la calibración se mostrará este dialogo con una cruz en la esquina izquierda de la pantalla. Cuando se produzca un contacto en la pantalla (ya sea cerca de la cruz o lejos) la cruz cambiará de lugar y así sucesivamente hasta completar la calibración.

Una vez se realice esto se utilizarán estos puntos de calibración y los puntos de las cruces para conseguir la matriz que convertirá los puntos a coordenadas de la pantalla.

GestorOptitrack: Esta clase se utilizará para modificar/consultar cualquier parámetro de la cámara infrarroja. Cuando se produzcan cambios/consultas sobre las características de la cámara en la interfaz, esta llamará a “TouchOptitrack” la cual se encargará de acceder a “GestorOptitrack” para realizar estas modificaciones/consultas en la cámara.

CameraOptitrack: Es la clase donde se realiza la búsqueda, tratamiento y seguimiento de los puntos recogidos por la cámara infrarroja. En esta clase se obtendrán los puntos capturados por la cámara, filtrándolos (ignorando aquellos que no cumplan los requisitos), ordenandolos (los que se hayan aceptado) y calculándolos en función de coordenadas de la pantalla (previamente se debe haber calibrado la pantalla). Quedando finalmente un vector de puntos los cuales ya se pueden utilizar en las aplicaciones que se desee.

Una vez finalizado el análisis del subsistema software del sistema comenzaremos a analizar el subsistema hardware. Esta se desarrolla en paralelo a la anterior y se realizará una verificación de las diferentes alternativas existentes.

Esto se realizará examinando la documentación desarrollada en el Estado del Arte, donde se recabó información sobre las diferentes tecnologías que se pueden encontrar en este momento y analizando los requisitos expuestos en el punto anterior por los clientes, estudiando minuciosamente cada una de ellos observando cuál de ellas podría ser la más recomendable.

Después de esto se concretan cuales podrían ser las que más se ajusten a lo dicho anteriormente y son las siguientes:

Resistiva

El funcionamiento de esta tecnología se basa en los cambios de corriente eléctrica entre las diferentes capas que lo componen. Mediante la captura de estos es posible calcular la posición del punto de contacto.

Como principales ventajas tenemos:

- Esta tecnología es una de las mejor consideradas cuando la pantalla a realizar es menor de 10 pulgadas.
- La precisión con la que se pueden obtener los puntos es muy buena, esto se debe a que es muy exacto el lugar de la pantalla donde se produce el cambio en la corriente eléctrica.
- Este tipo de pantallas poseen una durabilidad realmente excelente, esto en gran medida es gracias a que no se ven afectadas por elementos externos.
- Por último y para finalizar con las ventajas comprobamos que el coste final necesario para la construcción de la pantalla “Multi-Touch” con esta tecnología se encuentra dentro del rango de costes que se indicó en los requisitos de los usuarios.

Como principales desventajas observamos:

- No es posible (o inviable) la fabricación de pantallas táctiles resistivas cuyo tamaño de pantalla sea mayor de 10 pulgadas.
- No es posible la codificación de soporte “Multi-Touch” para este tipo de pantallas.

Aunque existe alguna desventaja más para este tipo, observamos que con las dos comentadas en el punto anterior es suficiente para descartarla. Ya que incumple dos de los requisitos indispensables exigidos por los clientes en las entrevistas realizadas.

Onda acústica

Su funcionamiento se basa en el cambio en las ondas de ultrasonidos transmitidas a través de la pantalla registrando la posición en la que se ha producido el contacto.

Sus principales ventajas son:

- La durabilidad mínima de la pantalla es muy alta, lo que supone un ahorro a los clientes en mantenimiento o reparaciones de la misma.
- Es posible el cálculo de la presión ejercida en la pantalla.
- La precisión en la detección de los puntos de contacto es también muy buena. Esto se debe al funcionamiento en el que fundamenta esta tecnología, ya que la posición obtenida mediante el cambio producido en las ondas es casi exactamente donde se produjo el contacto.

Las desventajas más destacables son:

- Es posible (aunque no muy recomendable) la fabricación de pantallas mayores de 10 pulgadas, condición indispensable requerida por los clientes.
- Esta tecnología no permite la codificación de soporte “Multi-Touch”.

Exactamente igual que lo comentado en el punto anterior. Esta tecnología no puede satisfacer dos de los requisitos indispensable solicitados por el cliente. Por este motivo queda descartada.

Capacitiva

El funcionamiento de esta tecnología está basado en la alteración de las capacitancias. Unos circuitos electrónicos se encargan de medir la ‘distorsión’ provocada al producirse un contacto en la pantalla

Las ventajas más relevantes son:

- La durabilidad es del sistema es considerable, debido a que no le afectan los elementos externos como ocurre con algunas de las tecnologías comentadas.
- Cumple unos de los requisitos indispensables requeridos por los usuarios. Permite la utilización de más de un puntero, es decir, soporta el uso de la tecnología “Multi-Touch”.

- Su coste de fabricación se encuentra dentro de los límites comentados por los clientes en las reuniones previas.

- Puede medir la presión ejercida por el usuario en el punto de contacto. Permitiendo diferenciar un simple toque de una pulsación más fuerte.

Sus principales desventajas son:

- La principal desventaja es que no es muy aceptable la construcción de pantallas de este tipo mayores de 10 pulgadas. Ni por coste ni por prestaciones. Ya que la precisión de este tipo no es muy buena y empeora cuanto mayor es el tamaño de la pantalla.

De nuevo se vuelve a incumplir una condición necesaria e indispensable para la satisfacción final de los futuros clientes, y es que la pantalla sea mayor de 10 pulgadas. Por lo tanto queda desechada también esta tecnología.

Infrarrojos

Su funcionamiento se basa en rodear mediante leds infrarrojos la pantalla produciendo un fenómeno llamado FTIR (Frustrated Total Internal Reflection). Esto provocará que se inunde de luz el metacrilato. A pulsar con el dedo o cualquier otro objeto interrumpimos esa luz produciéndose un reflejo, el cual será capturado detectado por la cámara infrarroja.

Las ventajas de esta tecnología son:

- Es factible la fabricación de pantallas mayores de 10 pulgadas utilizando esta tecnología sin aumentar en demasía el coste final ni empeorar las prestaciones ofrecidas en la teoría.

- Es posible la realización de un soporte “Multi-Touch” asequible debido a la facilidad de captura y manejo de diferentes puntos que posee esta tecnología.

- Su coste de fabricación se encuentra dentro de los límites de presupuesto dados por los clientes.

- Su precisión y sensibilidad en la detección de puntos son muy buenas. Aunque esto depende en la calidad de los componentes utilizados. A mayor calidad mejor será la detección y por consiguiente las precisión y sensibilidad anteriormente comentadas.

- Su durabilidad es muy buena, debido a que el sistema en conjunto está aislado de los agentes externos. El único componente que está en contacto con los usuarios es la plancha de metacrilato, la cual tiene una resistencia muy buena.

Los inconvenientes son:

- El principal inconveniente que se ha encontrado es que son pantallas más voluminosas que las tecnologías anteriormente comentadas. No es muy relevante ya que no se comentó nada sobre el volumen que deberá tener el producto final.

- No es muy recomendable la utilización de esta tecnología para pantallas menores de 10 pulgadas. En este caso, tampoco tiene ninguna relevancia ya que el requisito era totalmente el contrario. Que la pantalla fuera bastante mayor a 10 pulgadas.

Después de realizar este pequeño estudio sobre las tecnologías más destacadas. Es esta tecnología la que se observa que es la más indicada de todas las comparadas para la realización del proyecto. Cumple a la perfección con todos los requisitos exigidos por los clientes en las reuniones realizas.

Además es la única que se ha encontrado (después del estudio de todas ellas) que cumple todos, ya que las demás cumplían algunos de ellos, pero siempre había algún requisito indispensable el cual no se cumplía y por lo tanto, había que desechar esa opción.

4.2- Diseño

4.2.1 Introducción

Tras realizar el análisis detallado del sistema, llega el momento de ahondar en el nivel de abstracción, definiendo de manera más concreta los casos de uso más relevantes, mediante los llamados “Diagramas de Secuencia”. Estos diagramas son parte de un estándar para la definición de proyectos de ingeniería del software llamado “Unified Modeling Language” o UML.

Por otro lado, también se utilizarán los llamados “Diagramas Entidad-Relación” (DER). Estos diagramas presentan de forma esquemática y muy clarificadora las relaciones que hay entre las distintas entidades que forman un sistema de información.

Por tanto, el objetivo de esta fase del proyecto es definir cualquier detalle que haya quedado sin demasiada especificación en las fases anteriores del desarrollo.

4.2.1.1- Diagramas de secuencia

Tal y como ya se introdujo en el apartado de diseño, en esta sección se intentará profundizar un poco más la manera en que se implementará cada caso de uso.

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario.

Los conceptos más importantes relacionados con los diagramas de secuencia son:

- **Línea de vida de un objeto** (*lifeline*): La línea de vida de un objeto representa la vida del objeto durante la interacción. En un diagrama de secuencia, un objeto se representa como una línea vertical punteada con un rectángulo de encabezado y con rectángulos a través de la línea principal, que denotan la ejecución de métodos (activación). El rectángulo de encabezado contiene el nombre del objeto y el de su clase.

- **Activación**: Muestra el período de tiempo en el cual el objeto se encuentra desarrollando alguna operación, bien sea por sí mismo o por medio de delegación a alguno de sus atributos. Se denota como un rectángulo delgado sobre la línea de vida del objeto.

- **Mensaje**: El envío de mensajes entre objetos se denota mediante una línea sólida dirigida, desde el objeto que emite el mensaje hacia el objeto que lo ejecuta.

Adicionalmente se suelen poner recuadros que contienen comentarios explicativos de algún suceso que no quede del todo clarificado.

4.2.1.2- Cargar Calibración

Si el usuario desea cargar una calibración hecha anteriormente (para no tener que calibrar cada vez que se arranque la aplicación), utilizará esta opción. Si no existe una calibración anterior el sistema alertará al usuario y no cargará nada.

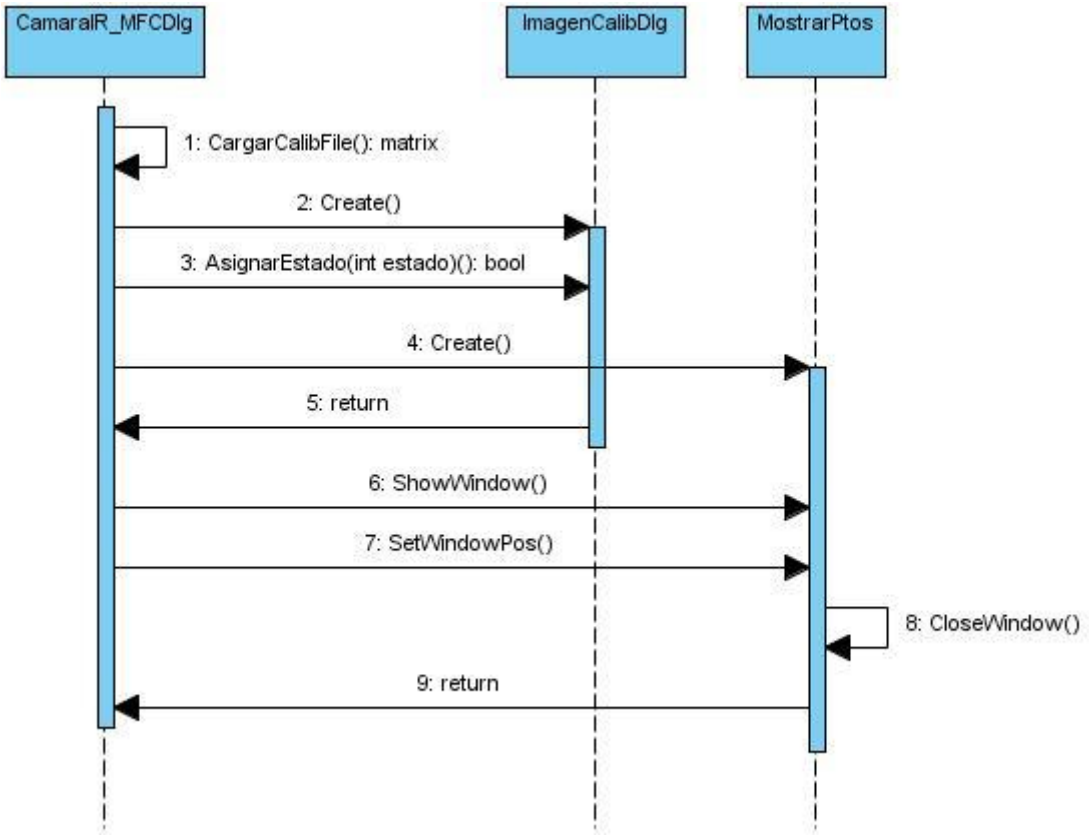


Figura 56. Diagrama de secuencia "Cargar Calibración"

Una vez cargada la calibración se comenzarán a capturar los puntos capturados por la cámara y se comenzarán a mostrar en el dialogo MostrarPtos. Si este se cierra, los puntos seguirán

capturándose pudiendo utilizarse como punteros diferentes si se activa la función de ratón. El dialogo de calibración no se mostrará ya que la calibración se realiza desde fichero.

4.2.1.3- Realizar Calibración

Si el usuario desea mejorar la calibración cargada o simplemente es la primera calibración que se realiza del sistema utilizará esta opción.

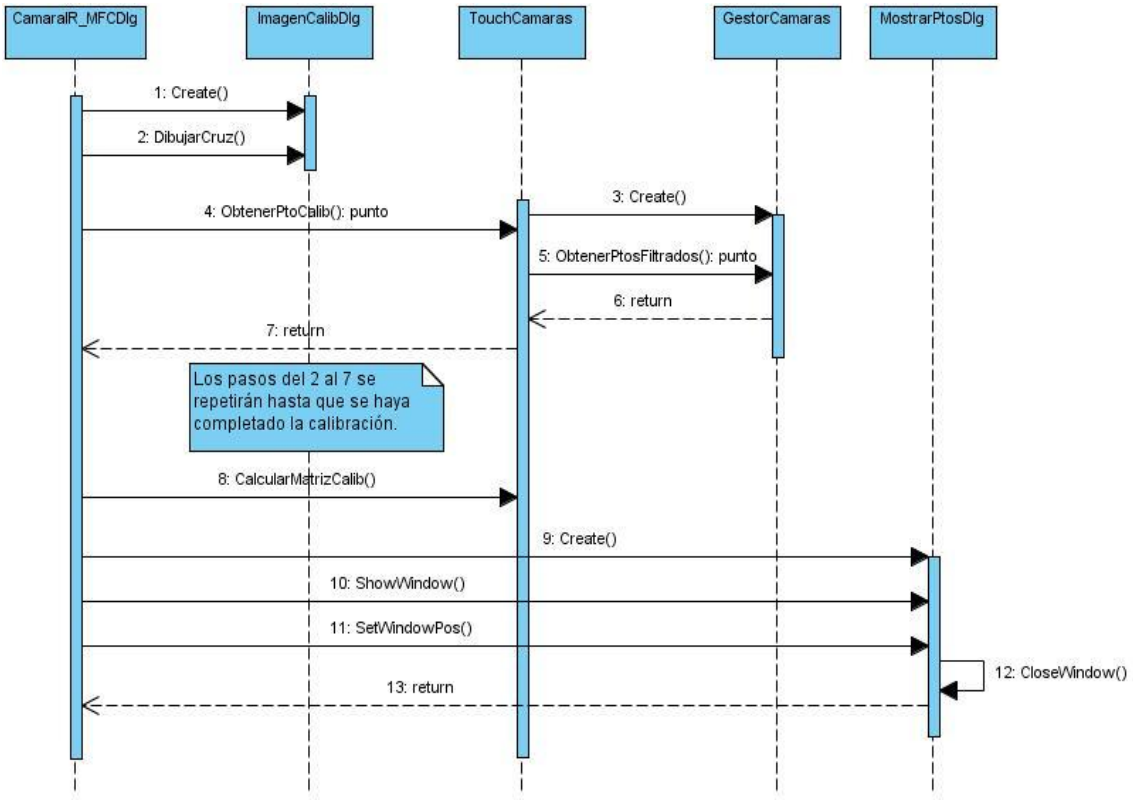


Figura 57. Diagrama de secuencia "Realizar Calibración"

Una vez se ha producido la calibración comenzará a ejecutarse el bucle de dibujado, donde se dibujarán en la pantalla los puntos de contactos existentes utilizando para ello la matriz de calibración calculada.

Una vez cerrada la ventana de “MostrarPtos” podremos activar los eventos del ratón de Windows pudiendo utilizar el dedo como un puntero.

4.2.1.4- Modificar Parámetros cámara

En cualquier momento de la ejecución de la aplicación se podrá modificar cualquier parámetro de la cámara para ajustar la imagen observada por esta, así como los LEDs de infrarrojos que tiene integrados.

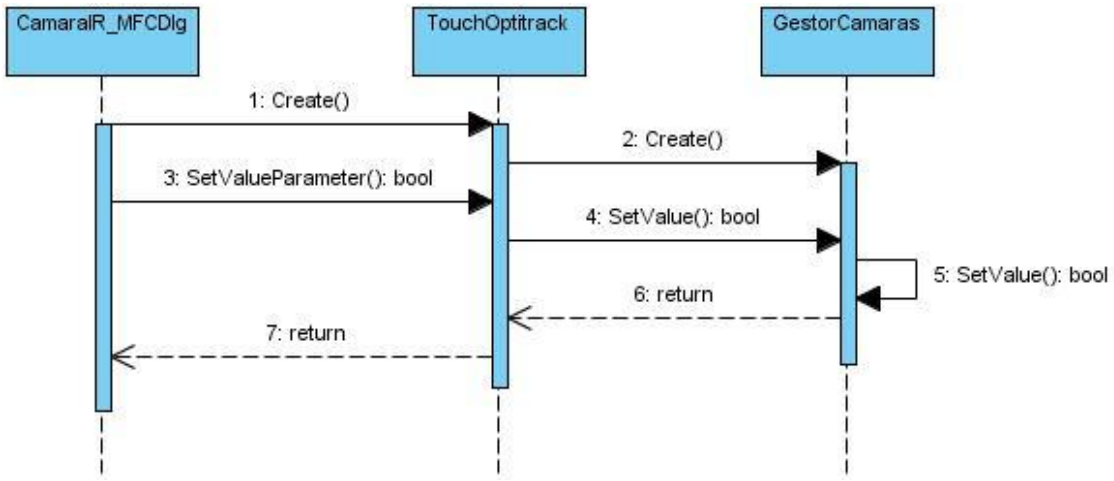


Figura 58. . Diagrama de secuencia "Modificar parámetros cámara"

La modificación de estos valores significará el cambio automático de los valores. Esto es, sin necesidad de apretar ningún botón de confirmación. Se recomienda realizar la modificación de estos valores en la primera de la ejecución de la aplicación, ya que estos deberán variar dependiendo del entorno donde se esté utilizando el sistema.

4.2.1.5- Cargar/Guardar configuración

En cualquier momento de la ejecución podremos cargar desde un fichero los parámetros de la cámara. Si no existen datos guardados de los parámetros el sistema alertará al usuario y no realizará ninguna carga.

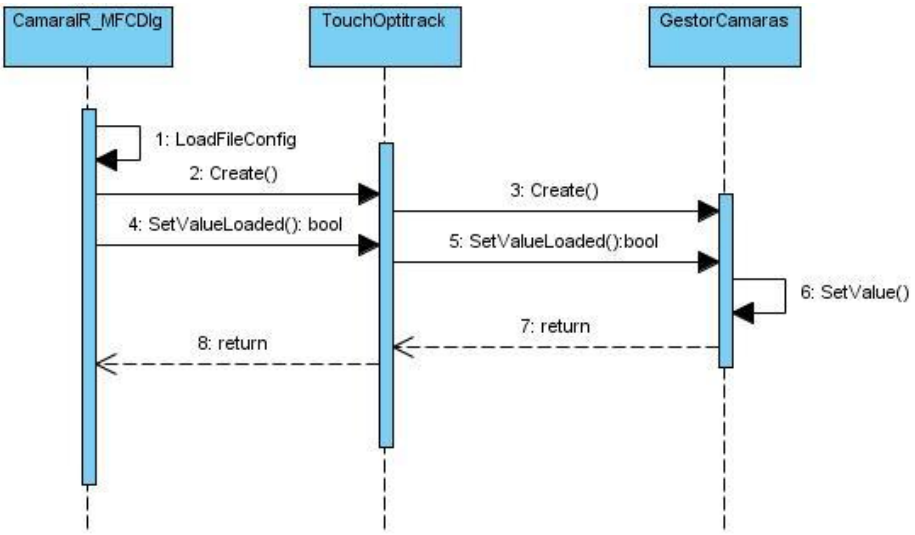


Figura 59. Diagrama de secuencia "Cargar Configuración"

Después de la carga estos parámetros serán colocados en las variables de la cámara correspondientes y comenzarán a considerarse como los actuales.

Por otra parte, si lo que se quiere realizar es el almacenamiento de los valores actuales. Estos se recogerán de la cámara y se guardarán en un fichero de texto creado expresamente para ello.

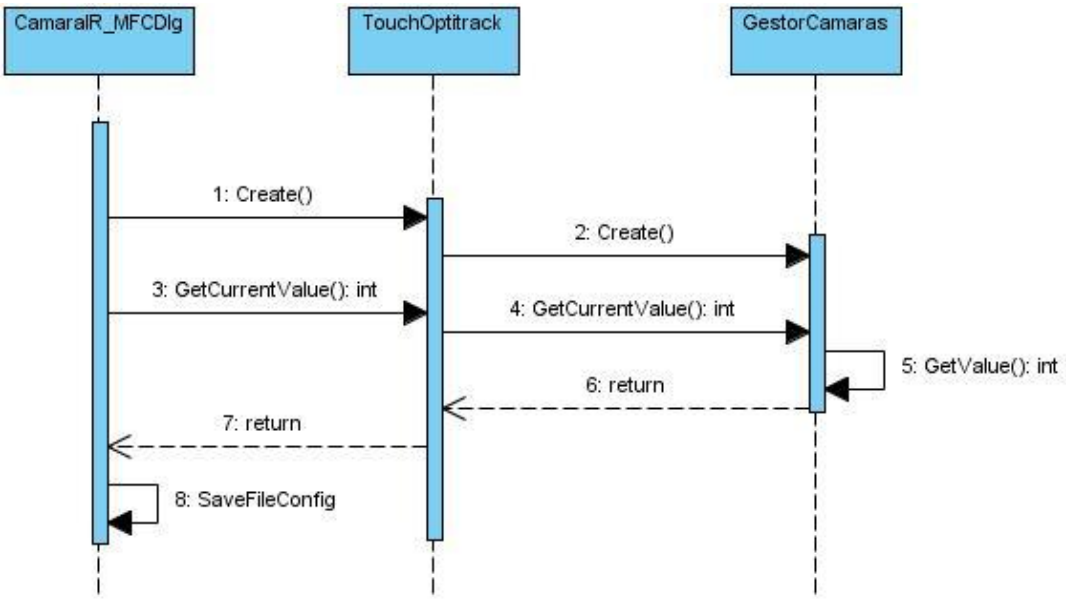


Figura 60. Diagrama de secuencia "Guardar Calibración"

Después del guardado los valores de los parámetros actuales seguirán siendo los mismos que antes del almacenamiento. Este caso de uso no supondría a simple vista ningún cambio en la ejecución de la aplicación.

A continuación y como se ha venido realizando a lo largo de la documentación de este proyecto comentaremos la parte de diseño hardware del mismo.

4.2.3 Hardware

Como se especificó en el punto anterior, la tecnología que se ajustaba más a las especificaciones exigidas por los clientes era la de infrarrojos. Por ello se propuso la realización del hardware del sistema mediante el uso de esa tecnología. Seguidamente se efectuará una minuciosa explicación del diseño ofrecido para su fabricación.

Primeramente se realizará la construcción del esqueleto de la mesa, cuyo diseño se puede observar en la “Figura 61”, utilizando para ello las barras de aluminio comentadas en el punto anterior. Para ello se necesitarán:

- 4 barras de 48.3 cm que serán las patas de la mesa.(1).
- 4 barras de 40 cm que utilizaremos como extensiones de las patas originales para aumentar la altura a la que estará la mesa en los casos en los que se considere necesario.(2).
- 2 barras de 46.3 cm que se encargarán de dar estabilidad a la mesa. Estas barras se colocarán debajo de las patas (o las extensiones) de la misma, uniendo con cada una las 2 más cercanas (como observamos en la “Figura 61”).(3)
- 2 barras de 77.3 cm y 2 de 53 cm con las que formaremos el marco de la mesa donde posteriormente se colocarán las pantallas del sistema (la de contacto y la de reto-proyección).(4 y 5)
- La altura total propuesta para el diseño es de 83 cm, quedando 13.3 cm de las patas de extensión sin sacar. Se dejan esos 13 cm para darle sujeción a las extensiones. (6).

El diseño final de este esqueleto se realizó con el Google Sketch up para facilitar la compresión a la hora de su fabricación y una imagen de este se muestra a continuación. En esta imagen se especifican las medidas de las barras utilizadas y su identificación para solventar las posibles dudas que puedan surgir.

Por otra parte, cabe destacar, que se utilizó dicha herramienta para el diseño ya que es gratuita y se trata de una buena herramienta de prototipaje.

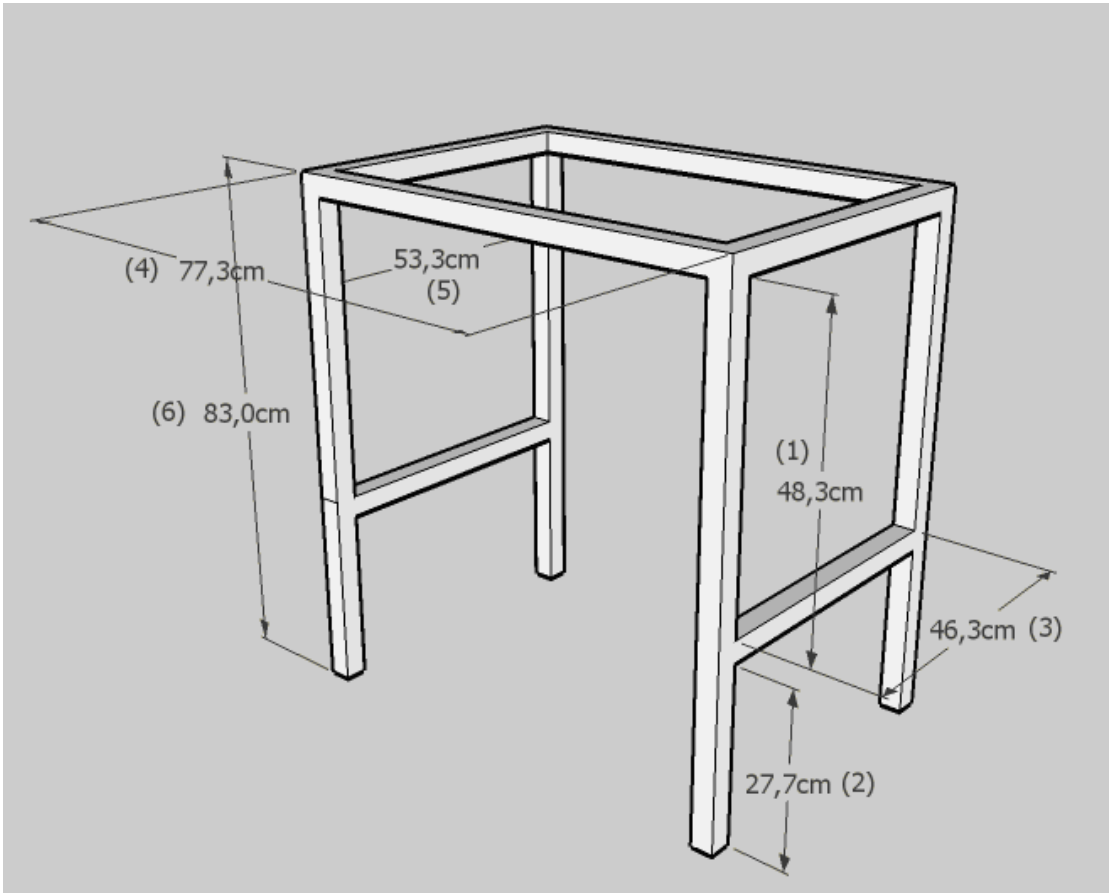


Figura 61. Diseño esqueleto físico del sistema

Por otro lado y en paralelo se puede desarrollar el diseño de la pantalla del sistema. Esta estará compuesta por:

- 1 Plancha de plexiglás común transparente de 79 cm x 48.5 cm x 0.8cm y una pantalla de retro-proyección de la marca Traulux con las mismas medidas. Se recomienda escoger un grosor de 8mm para asegurarse de que las pantallas son rígidas y no se doblan al presionarla. Estas 2 pantallas se superpondrán la una a la otra quedando la de Plexiglás encima de la de retro-proyección.

- 100 LEDs de infrarrojos SFH485 de 5mm con una amplitud de onda de 880nm y un ángulo de visión de 20º. Longitudes de onda superiores son más difíciles de filtrar para nuestro propósito, y el ángulo de emisión de 20º es suficiente para el grosor del plexiglás adquirido.

- 20 resistencias de 10 ohm, 1W y cable de cobre necesario para montar el circuito.

- 1 Fuente de alimentación Siemens de 13 Voltios/7 Amperios que se encargará de la alimentación de los LEDs.

- 4 guías en forma de “U” de la marca BOSH-REXROTH. En estas guías se realizarán orificios cada 2.54 cm (una pulgada). Estos orificios deben realizarse lo más al centro posible de las guías. Asegurando así que el haz de luz incidirá completamente en el metacrilato.

Una vez realizados los orificios, se colocarán los leds de infrarrojos anteriormente comentados. Los LEDs operan a 100 mA con una caída de voltaje de 1,5V. Dado que utilizaré una alimentación de 13V, los colocaré en series en paralelo de 5 LEDs cada una, utilizando una resistencia de 10 ohm, 1W en cada serie. Al finalizar esto, se unirán todos los polos positivos de todas las tiras y todos los negativos para facilitar su conexión a la fuente de alimentación (“Figura 64”). Un par de imágenes de ejemplo del diseño de estas tiras se pueden ver en la “Figura 62” y “Figura 63”.

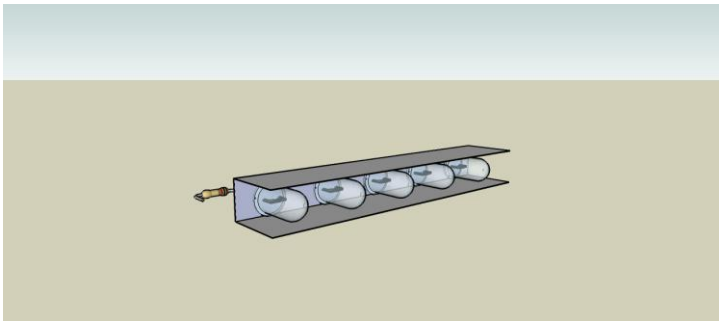


Figura 62. Diseño tira de LEDs frontal

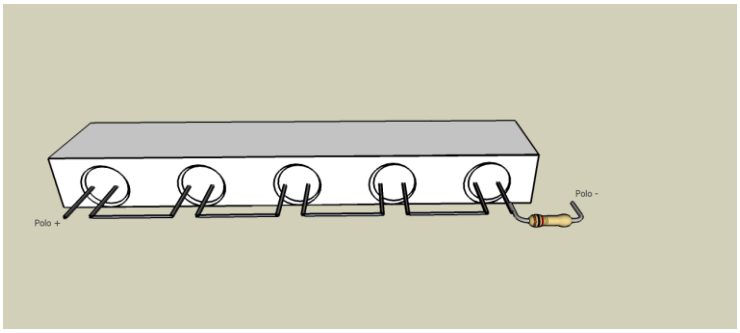


Figura 63. Diseño tira de LEDs trasero

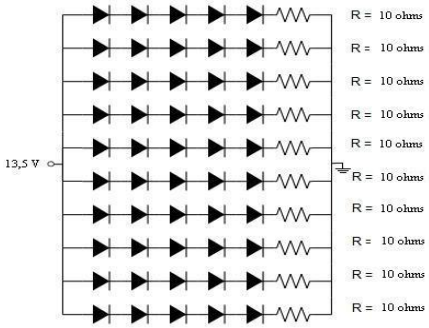


Figura 64. Ejemplo alimentación de las tiras de LEDs

Una vez concluido esto se unirán las 4 guías formando un marco (“Figura 65”) quedando encajada la pantalla construida anteriormente. Con todo esto conseguimos una pantalla de dos capas consistente y con los LEDs colocados en su sitio y preparados para su conexión a la corriente.

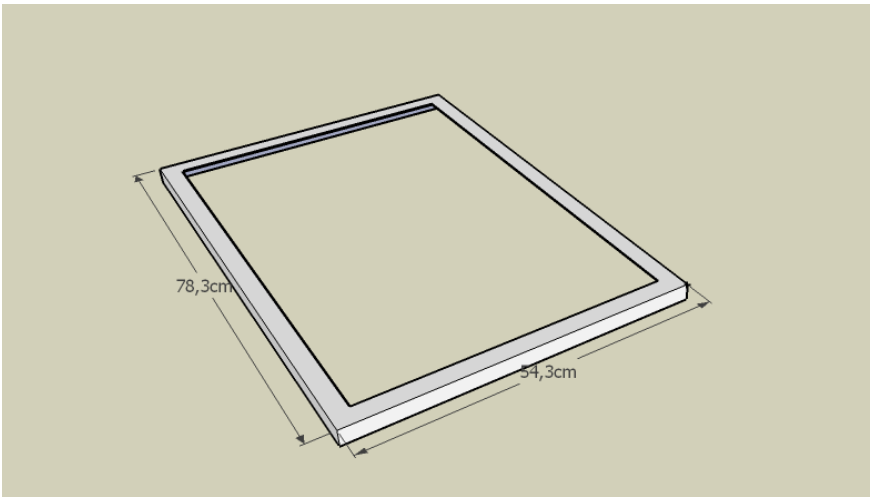


Figura 65. Diseño marco del sistema

Al mismo tiempo se realizará la cobertura del sistema. Esta se utilizará para ocultar y proteger los componentes del sistema. Estará formada por tres planchas de contrachapado de 1cm de grosor unidas entre sí formando una “U” La “Figura 66” representa un diseño de la misma.

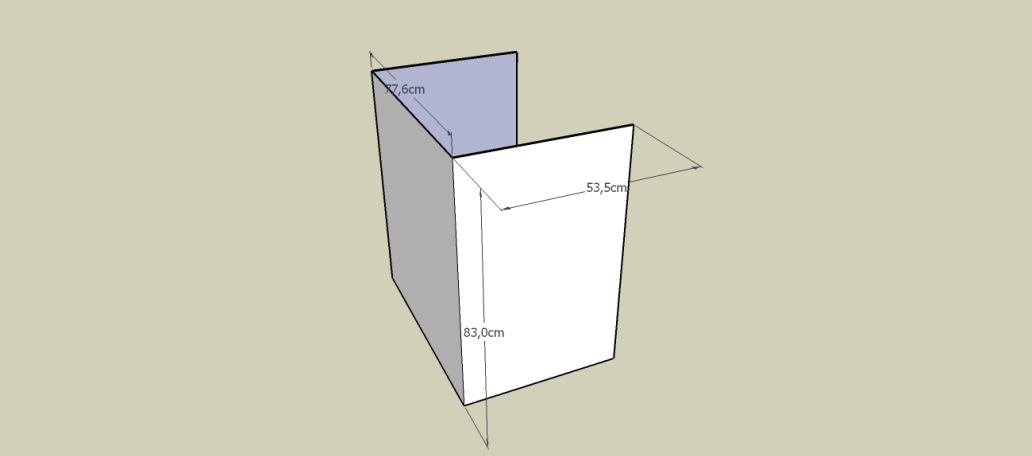


Figura 66. Diseño cobertura

Por último y una vez realizado todo lo anterior se procederá a realizar el diseño de los subsistemas de captura y visualización del sistema. Para la cual se necesitará:

- 1 Proyector de video. En este caso se utilizará un EIKI LC-XIP 2000, con el que se proyectará la imagen en la pantalla realizada anteriormente. Se debe conseguir calibrar el proyector para que la imagen ocupe por completo la pantalla.

- 1 cámara infrarroja. Se utilizará la Optitrack FLEX-120. Con ella se deberá comprender como mínimo, lo mismo que abarque la imagen proyectada. Si abarca más se omitirán los puntos que se encuentren fuera del rango.

Con esto se finalizará el diseño del subsistema hardware del sistema que quedará de la siguiente manera, ver Figuras “67” y “68”. Notar que se ha añadido un espejo ya que si no sería inviable proyectar la imagen en toda la pantalla (ya que el proyector debería estar mucho más lejos de lo posible en nuestro diseño).

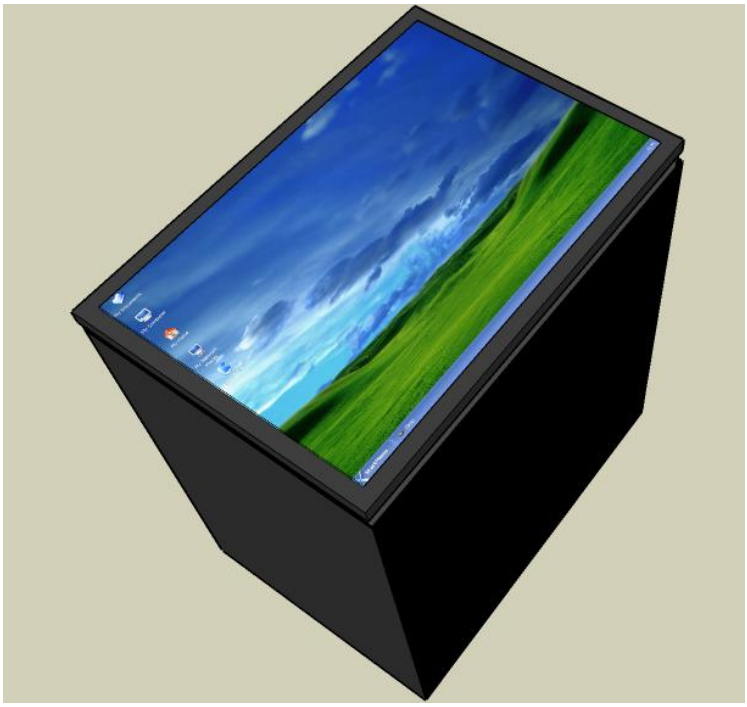


Figura 67. Diseño final del sistema (frontal)

Figura 68. Diseño final del sistema (trasero)

4.3- Implementación

4.3.1 Introducción

Hasta ahora se han definido los requisitos que deben cumplir cada uno de los 2 subsistemas que se ha estado tratando durante toda la documentación: subsistema de software y subsistema de hardware. Conociendo los requisitos que marcan lo que se debe hacer, se realizo un análisis para alcanzar una primera aproximación lógica de cada sistema. Posteriormente se realizo la fase de diseño en la que se establece una primera solución, la cual se describe solo de forma global. Por tanto en este momento se conoce todo lo que debe hacerse, y los esquemas generales para hacerlo, con lo cual solo queda implementar la solución siguiendo la estructura creada en los apartados anteriores. La implementación es la parte de más bajo nivel del proyecto, donde se crean soluciones a problemas concretos planteados en el diseño.

Estos 2 subsistemas los dividiremos en 5 apartados, explicando las partes relevantes de la implementación de cada uno de ellos:

- **Soporte físico del sistema:** en este punto se construirá “el esqueleto” del sistema. Este se encargará de sostener la pantalla con las 2 capas y los LEDs y de mantener todos los demás componentes del sistema fuera de la vista de los usuarios.

- **Sistema de captura:** la implementación de este subsistema consiste en la creación de la pantalla donde se producirán el contacto y la colocación de la cámara infrarroja. Esta cámara será la encargada de recoger los puntos donde se haya producido el contacto. La aplicación para realizar la captura es suministrado por el fabricante del sistema de captura.

- **Sistema de visualización:** en este subsistema debe implementarse tanto la pantalla de retro donde el usuario ve la imagen, como el proyector con el cual se proyectará la imagen sobre la pantalla.

- **Sistema de implementación de la librería:** la implementación de este subsistema consiste en la creación de una serie de clases que contengan todas las estructuras necesarias para el manejo de los puntos capturados por el subsistema hardware.

- **Sistema de implementación del driver del ratón:** en este punto se realizará la codificación necesaria para utilizar los puntos recogidos por la cámara anteriormente como puntero de ratón.

Toda la implementación se realizara utilizando el lenguaje de programación C++, ya que es el utilizado hasta ahora en el desarrollo de la aplicación utilizada en la captura de las imágenes.

Cabe pensar que habiendo realizado un buen análisis y diseño de todo el sistema, esta fase del proyecto no debería de presentar demasiadas complicaciones.

4.3.1.1- Implementación hardware

En este punto se comentarán los detalles de fabricación del subsistema hardware del sistema realizado. Se mostrarán imágenes del resultado final con las diferentes pruebas y pasos que se han ido realizando.

Esta se va a dividir en tres subsistemas: Sistema físico, sistema de captura y sistema de visualización.

4.3.1.2- Sistema físico (esqueleto)

En este punto no se encontró ningún problema de implementación para ninguna de los tres objetos que lo componen.

Desde un principio se pensó que la mejor manera de realizar el esqueleto físico era mediante perfiles estructurales de aluminio BOSH-REXROTH de 30x30 mm de diámetro idea que se confirmó al comenzar a realización del mismo. A continuación se muestran imágenes del esqueleto finalizado.

Por otro lado se pensó que la tapa que cubre el sistema se realizará mediante planchas de contrachapado de 1cm de grosor que ya se comentó en puntos anteriores de la documentación. A esta se le añadieron unas bisagras en las dos esquinas para facilitar la manipulación de los componentes internos del sistema si fuera necesario.

Por último, el marco que recubre la pantalla y oculta los LEDs y cables a la vista del usuario se fabricó con cuatro barras en “L” de aluminio uniendo cada una de ellas con un pequeño punto de soldadura.

Tanto la tapa como el marco se pintó de color negro para mejorar un poco la estética del sistema. Aunque en un principio no era un requisito de los futuros usuarios se pensó que era una buena idea como quedó demostrado tras la finalización.

Imágenes de estos dos puntos así como del sistema finalizado se pueden observar a continuación en la “Figura 69:

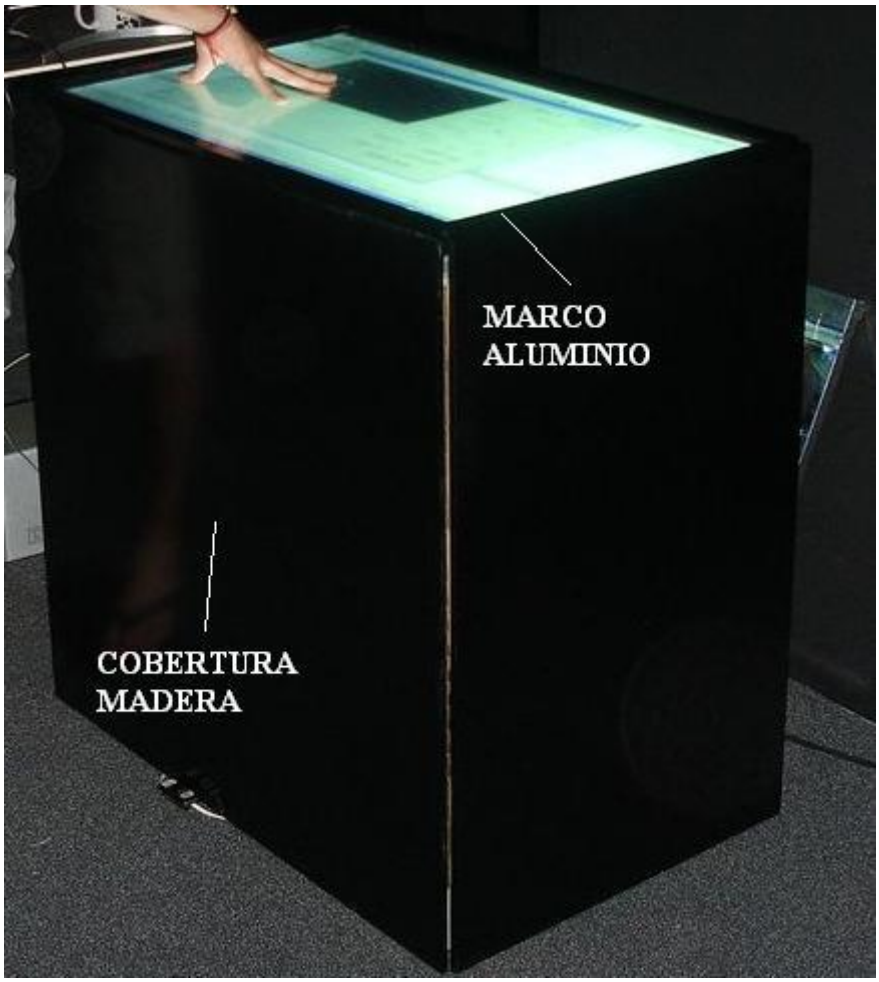


Figura 69. Sistema físico finalizado

4.3.1.3- Sistema de captura

El mayor problema que surgió al comenzar la implementación del sistema de captura del proyecto fue que tipo de plexiglás se utilizaría como pantalla de contacto del mismo. En un principio se optó por utilizar un plexiglás semitransparente, de esta manera se estaba intentando disminuir en la medida de lo posible el coste del sistema. Pero después de la realización de algunas pruebas a menor escala que el proyecto final se desechó este plexiglás ya que no poseía la propiedad en la que se basa el sistema a desarrollar, la FTIR. El siguiente que se probó fue un metacrilato totalmente transparente el cual desde un principio dio excelentes resultados. Por lo tanto se decidió a utilizar este en la fabricación de este subsistema.

Otro problema que surgió en el momento de implementar este punto fue que tipo de LEDs infrarrojos utilizar. Esto se resolvió realizando búsquedas en Internet donde se consultaron diferentes tipos de LED y se adquirieron los que se observó cumplían de sobra los requisitos

necesarios. Los LEDs adquiridos se excitan con una corriente relativamente pequeña y su ángulo de apertura en lo suficientemente pequeño para que el haz de luz infrarroja se pueda enfocar hacia dentro del plexiglás.

La primera prueba que se realizó fue la del montaje de una tira de 5 LEDs + 1 Resistencia con lo que se comprobó cual sería el voltaje necesario para alimentar los 100 LEDs que compondrían el sistema. Esto se realizó primeramente utilizando una Fuente de Alimentación con Voltímetro regulable. Con esta observamos que se necesitaba 1,3V para que estos se comenzaran a excitar. También cabe destacar que cada tira de 5 LEDs disipaba unos 0,2 Amperios a 9Voltios. Por todo lo anteriormente comentado la Fuente de Alimentación definitiva que se debe adquirir es una fuente de 11V/4W.

Por otra parte, se observó un pequeño problema al intentar que la cámara abarcara toda la pantalla de contacto. En un primer momento se intentó colocarla mirando directamente a la pantalla (con la idea de reducir en la medida de lo posible los costes), idea que se desechó rápidamente al comprobar que no se conseguía abarcar toda la superficie ya que no se encontraba a la distancia necesaria de la pantalla. La solución que se planteó fue la idea que se indicó en el diseño del sistema. Esta es, colocando la cámara mirando a un espejo con lo que se consigue reducir la distancia necesaria para abarcar toda la superficie.

Por último, el sistema de cámaras infrarrojas no supuso ningún problema ya que se tenía totalmente definido cuáles eran las que mejor se comportarían en un sistema como el que se estaba desarrollando. La implantación en el mismo corroboró la teoría.

Seguidamente, se observa una imagen del sistema comentado resaltando cada uno de los componentes utilizados para su desarrollo (“Figura 70”):

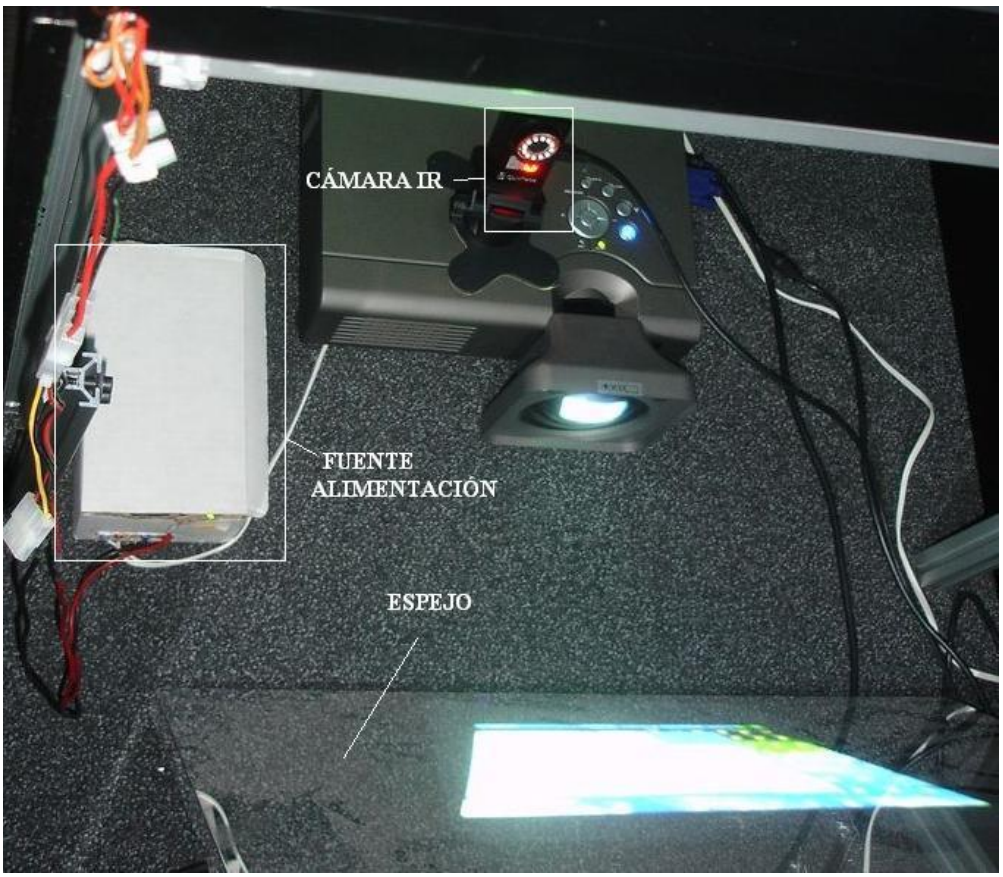


Figura 70. Componentes del sistema de captura

4.3.1.4 Sistema de visualización

En este punto del sistema la única duda que podría surgir era que tipo de proyector se debería utilizar. En las pruebas se utilizó un proyector común de la marca EIKI y funcionó perfectamente con lo que el posible problema se desvaneció. Por otro lado la pantalla de retro utilizada es una pantalla de uso común en todas las retro-proyecciones y ya se conocía de antemano que no iba a suponer ningún problema.

A continuación se muestran imágenes del interior de la mesa y de la pantalla donde se encuentran tanto el sistema de visualización como el de captura, en este caso se resaltan los componentes del sistema de visualización (“Figura 71”):

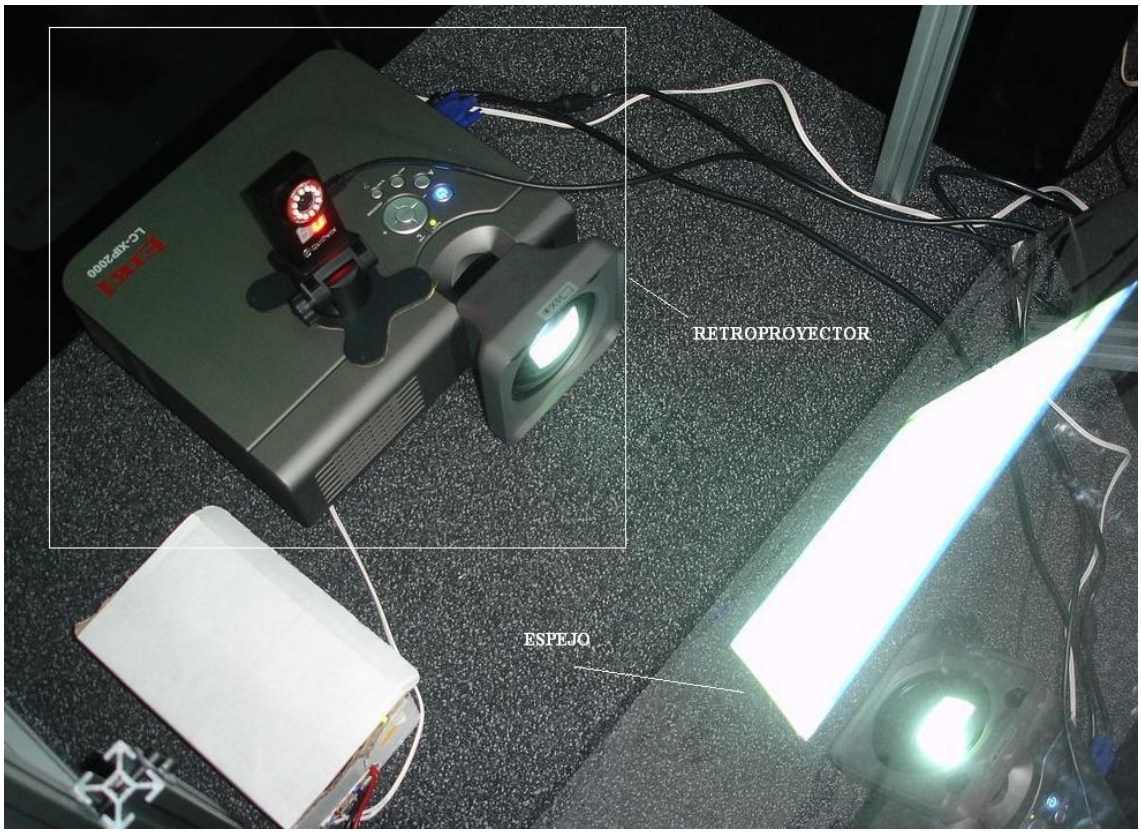


Figura 71. Componentes del sistema de visualización

4.3.2.1 Implementación software

En esta sección se explicará de forma amplia los detalles de implementación software que se han tenido en cuenta para la realización del proyecto. Se mostrarán los fragmentos de código más relevantes o más peliagudos, así como capturas de algunas de las pantallas que se han ido generando.

Esta implementación se va a dividir en dos subsistemas claramente diferenciables, las cuales se comentan a continuación.

4.3.2.2 Implementación de la librería

Esta es la primera de los dos puntos que se desean comentar. En un principio se pensó en implementar la librería utilizando Windows Forms. Esto se descartó al surgir el primer problema en el comienzo de desarrollo de la librería. Este fue que mediante Windows Forms no se consiguió mostrar por pantalla los puntos recogidos y manipulados por la aplicación condición indispensable para la realización del proyecto.

Esto hizo que se comenzará la codificación utilizando MFC (librería que envuelve parte de las clases de la API. Mediante algunas de estas clases es posible manejar algunos de los objetos gestionados por Windows) con la cual no se tuvo ningún problema de relevancia.

Para el desarrollo de la librería se utilizaron tres clases de codificación. Estas son: TouchOptitrack, GestorOptitrack y CameraOptitrack. Donde algunos de los métodos más relevantes son:

TouchOptitrack:

MultMatriz: la función principal de este método es la modificación de los puntos capturados por las cámaras (los cuales están en coordenadas de la cámara) a puntos en coordenadas de la pantalla.

Esto se realiza utilizando una matriz obtenida mediante una función de la OpenCV llamada “cvWarpPerspectiveQMatrix” donde se le pasan los cuatro puntos de calibración de la pantalla. Utilizando esta matriz y los puntos a calibrar se obtiene como resultado los puntos en coordenadas de la pantalla.

Esta función además también se encargará de modificar estos puntos dependiendo de la orientación de la imagen y rellenar la estructura que se manda a los clientes vía sockets.

ComprobarGestos: este se encarga de comprobar si se ha producido o se esta produciendo algún gesto en cada iteración de la aplicación. Esto se realiza llevando un seguimiento de los puntos de contacto que existen en cada momento y comprobando si estos cumplen las condiciones necesarias para la realización del gesto. Los gestos implementados en nuestro sistema son:

- Mover 3 dedos en cualquier dirección: Rota la pantalla hacia la dirección de los dedos.
- Circunferencia con un dedo: Muestra un menú.
- Punto detectado-punto NO detectado: Se trata del click del ratón.

ComprbarPuntoEncontrado: este método se utiliza para detectar si un punto que en el frame anterior existía sigue existiendo. Si es así, el nuevo punto detectado por la cámara será tratado como hemos comentado en "MultMatriz", si por el contrario ese punto ya no existe el sistema se debe asegurar de que se elimine y no se trate como si existiera ya que esto daría lugar a fallos en la aplicación.

GestorOptitrack:

Simplemente se encarga de controlar las características de la cámara de captura de puntos. No tiene ningún método o suficientemente relevante para añadirse a la memoria.

CameraOptitrack:

PuntosDetectadosFiltrados: Este método utiliza el Ccom de la cámara para obtener el número de puntos que se han detectado en el frame. Estos puntos son "filtrados" mediante el método "limpiarListaPuntos" que se comenta en el punto siguiente. El resultado de este método es una lista de puntos con los que se va a trabajar.

limpiarListaPuntos: es el método encargado de eliminar puntos detectados que podrían suponer un problema a la hora de la ejecución de la aplicación. De esta manera, se eliminarán los puntos que se encuentren a una distancia muy pequeña de otro punto (posible reflejo) y los puntos cuyo área sea menor al área mínima.

CacularMenorDistancia: en este método se comparará la lista de puntos del frame anterior con la lista de puntos del actual. Esto se realiza para determinar que puntos del frame anterior se corresponden con que puntos del frame actual, cuáles de ellos son nuevos y cuales han desaparecido.

OrdenarDistancia: método donde para cada punto encontrado en el frame actual se ordena la lista de puntos del frame anterior de menor a mayor.

ObtenerMasCercano: utilizando el método “OrdenarDistancia” este método se encarga para cada punto del frame anterior de encontrar el punto más cercano de los actuales. Cuando un punto del frame anterior obtiene su “punto actual” este se elimina de las listas de los demás puntos para que no existan posibles errores en la asignación. Al finalizar esta función cada punto anterior tendrá su homónimo y estos pasarán a ser los puntos anteriores del próximo frame.

El código de todos los métodos comentados en los puntos anteriores se puede encontrar en el apéndice A de la memoria. Por razones obvias no se ha incluido en este punto.

Para finalizar, colocamos un par de imágenes de la interfaz de la librería mientras está siendo utilizada (“Figura 72” y “Figura 73”):

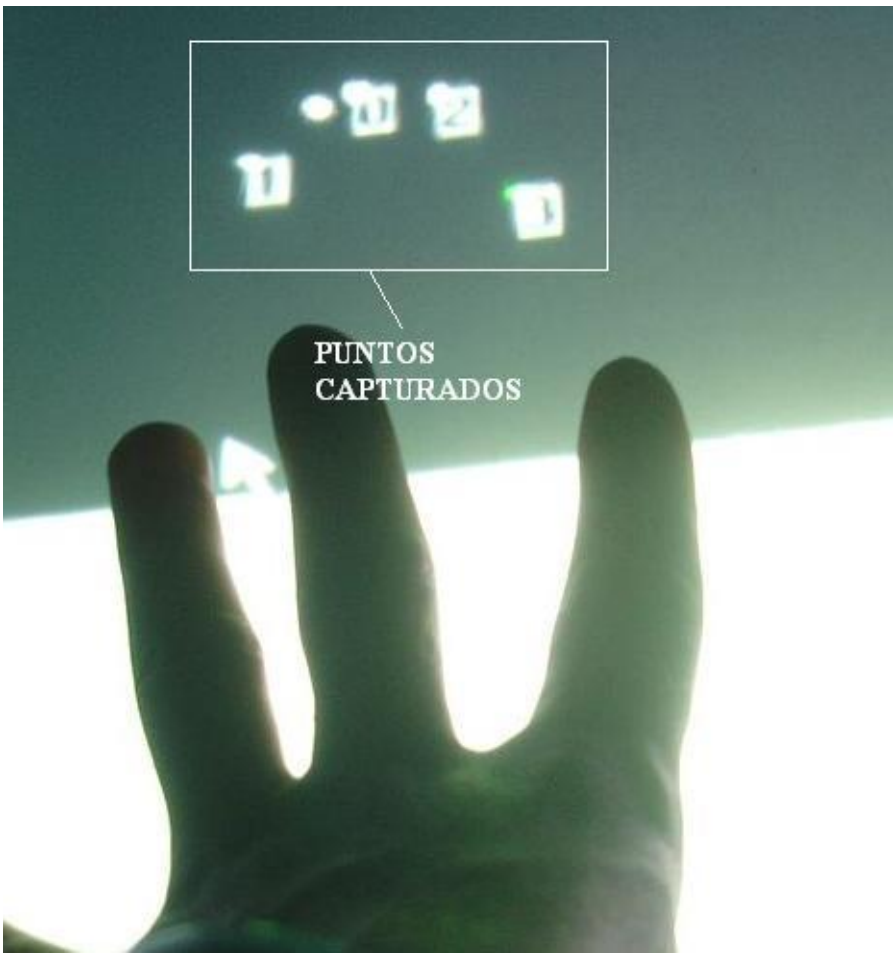


Figura 72. Puntos capturados por la librería

Figura 73. Ejemplo utilización de la librería

4.3.2.3 Implementación de la arquitectura Cliente-Servidor

En la otro punto nos encontramos con la implementación de la arquitectura del paso de información del sistema. De la misma manera que en el punto anterior en un primer momento se aceptó como posible solución integrar cualquier tipo de código nuevo dentro de nuestra librería (para futuras aplicaciones). Esto se descartó rápidamente ya que era modificar gran parte del código desarrollado y se pensó que una mejor manera de afrontar esto era mediante la utilización de sockets TCP.

Esta arquitectura utiliza la librería como Servidor de puntos y eventos. Esto es, cuando la pantalla se ha calibrado y ya se poseen eventos y puntos en coordenadas de la misma se arranca el servidor. Este servidor está esperando continuamente a posibles peticiones de datos por parte de los clientes. Cuando un cliente se conecta al servidor, este le pasa los puntos y los eventos recogidos en cada frame. Por su parte el cliente utilizará los datos recogidos en su aplicación.

A continuación se muestra parte del código destacado de la arquitectura utilizada, donde existen 2 subsistemas diferenciados:

Por un lado se tendrá el **Servidor:**

Primeramente se establecerá la conexión con el socket cliente (si este existe) mediante el método “aceptarCliente” (“Figura 74”). Este método se encarga de realizarla, si existe algún cliente que desee realizar la conexión, y si no se ha llegado al límite de clientes permitido.

```
int CCamaraIR_MFCDlg::aceptarCliente()
{
	SOCKET AcceptSocket;
	fd_set lectura;
	struct timeval tiempo;
	int val, i, libre, max_port = 0;
	n_clientes = 3;

	FD_ZERO(&lectura);

	FD_SET(SockServer,&lectura);

	tiempo.tv_sec=0;
	tiempo.tv_usec=1000; /* 1 milisegundo */
	//Cuando se salga del select() es porque:
	//1) se ha intentado conectar un nuevo cliente
	//2) uno de los clientes ya conectados nos ha enviado un mensaje
	//3) uno de los clientes ya conectados ha cerrado la conexión.
	val=select(SockServer+1,&lectura,NULL,NULL,&tiempo);
	libre = -1;
	int clientes = 0;

	if ((val)>0)
	{
		AcceptSocket = accept( SockServer, NULL, NULL );

		for (i=n_clientes-1; i>=0; i--) {
			if (SockClient[i] <= 0) {
				libre = i;
			} else {
				clientes ++;
			}
		}

		if (libre == -1) {
			closesocket(AcceptSocket);
		} else {
			SockClient[libre] = AcceptSocket;
		}
	}

	return 0;
}
```

Figura 74. Código "aceptar cliente" del servidor

Se comprueba si el cliente ha solicitado datos al servidor mediante el método “hayDatos” que se comenta a continuación, en la “Figura 76”, y si existe dicha petición se mandan los datos al cliente.

```
//Establecemos la conexion
aceptarCliente();

for (int i=0; i<n_clientes; i++) {
	if (SockClient[i] > 0) {

		// COMPROBAR SI SE HA REALIZADO PETICION DE DATOS
		int datos = hayDatos(SockClient[i]);
		if (datos > 0) {


			// ENVIAR DATOS
			if (send(SockClient[i],(char*)TouchCamaras->ds,sizeof(TouchCamaras->ds), 0) < 0) {
				closesocket(SockClient[i]);
				SockClient[i] = 0;
			} else {
				send(SockClient[i],(char *)TouchCamaras->ds,sizeof(TouchCamaras->ds), 0);
			}
		}
```

Figura 75. Implementación general del servidor

Como se comentó en el párrafo anterior, mediante esta función el servidor determinará si el cliente le está demandando datos. Esto ocurre de la siguiente forma. El cliente manda un carácter cualquiera cuando desea datos del servidor. Este carácter es recibido por el servidor que inmediatamente saltará del “select” con un valor de “val” mayor que 0, lo que producirá que se devuelva este valor al código anterior y se manden los datos requeridos. El código asociado se puede observar en la “Figura 75”.

```
int CCamaraIR_MFCDlg::hayDatos(SOCKET socke)
{

	fd_set lectura;
	struct timeval tiempo;
	int val, max_port = 0;
	char *cad;

	FD_ZERO(&lectura);
	FD_SET(socke,&lectura);


	tiempo.tv_sec=0;
	tiempo.tv_usec=100; /* 1 milisegundo */
	val = select(socke+1,&lectura,NULL,NULL,&tiempo);
	if (val > 0) {
		cad = (char *)malloc(val * sizeof(char));
		recv(socke, cad, val, 0);
		free(cad);
	}

	return val;

}
```

Figura 76. Función "hayDatos" del servidor

Por otro lado tendremos el **Cliente:**

Donde se inicializará el socket (omitido el código ya que se determinó que era demasiado obvio).

```
void clienteSockets()
{

	//inicializarSocket();
	//Conectar la servidor.
	TouchCamaras->nEventos = 0;

	send(ConnectSocket, b, (int)strlen(b), 0);
	res_socket = recv(ConnectSocket, (char*)&ds, sizeof(ds), 0);

	if ( (res_socket > 0) ) {
		//this->lConexion->Text = "Recibiendo...";

		contador++;
		for(int i  =0; i < 10; i++)
		{
			if(ds[i].ClickUp)
			{
				TouchCamaras->EventoBoton[i].tipo = 2;
				TouchCamaras->EventoBoton[i].x = (int)ds[i].x;
				TouchCamaras->EventoBoton[i].y = (int)ds[i].y;
				TouchCamaras->nEventos++;
			}
			if(ds[i].ClickDown)
			{
				TouchCamaras->EventoBoton[i].tipo = 1;
				TouchCamaras->EventoBoton[i].x = (int)ds[i].x;
				TouchCamaras->EventoBoton[i].y = (int)ds[i].y;
				TouchCamaras->nEventos++;
				ds[i].ClickMove = false;
			}
			if(ds[i].ClickMove)
			{
				TouchCamaras->EventoBoton[i].tipo = 3;
				TouchCamaras->EventoBoton[i].x = (int)ds[i].x;
				TouchCamaras->EventoBoton[i].y = (int)ds[i].y;
				TouchCamaras->nEventos++;
			}

		}



		//this->fMostrarPuntos->RecibirPuntos((char*)dibujar);
	}
	else
	{
		WSACleanup();
		puts("no hay recepcion");
	}
	//send(ConnectSocket, b, (int)strlen(b), 0);
}
```

Figura 77. Código del cliente

A partir de este momento la conexión Cliente-Servidor se habrá producido y cuando el cliente desee le demandará datos al servidor como se muestra en el código anterior.

5.- Pruebas y resultados

En este capítulo se mostrarán los resultados obtenidos para discutir si el sistema se ajusta a los requisitos establecidos en las primeras etapas del proyecto.

En primer lugar se analizarán las pruebas llevadas a cabo para garantizar que sistema es viable funcionalmente, es decir, que la tecnología seleccionada y el sistema implementado es capaz de capturar y manejar los puntos correctamente. Así mismo, también se deberá comprobar que esos puntos son enviados a los clientes que lo soliciten.

Por último se discutirá si el precio final del sistema estará o no al alcance de las empresas interesadas. Para ello se calculará el coste final de desarrollo del sistema y se hará un supuesto de venta con el que calcular una aproximación al valor de venta.

5.1- Descripción de los experimentos

Las pruebas realizadas se pueden agrupar en tres de los cuatro grupos que se han ido comentando en anteriores puntos de la documentación. El subsistema “esqueleto físico del sistema” no se ha tenido en cuenta en este apartado ya que no es objeto de ninguna prueba.

- Pruebas del sistema de captura.
- Pruebas del sistema de visualización.
- Pruebas de la aplicación.

Cabe destacar que la realización de las pruebas del “sistema de captura” y del “sistema de visualización” no se pueden desarrollar de la misma manera que las pruebas de un sistema de software (como es el caso de la aplicación) por ellos se contó con la ayuda de 15 alumnos de Ingeniería Informática de la universidad de Valencia con edades comprendidas entre los 18 y los 26 años. Estos alumnos fueron trasladados al Instituto de Robótica (donde se encuentra instalado el sistema) en 3 grupos de 5 alumnos. Con esto lo que se quería conseguir era obtener información lo más imparcial posible acerca de las pruebas que se iban a realizar.

A continuación se describe las pruebas realizadas para cada uno de los sistemas comentados:

5.1.1 Descripción de las pruebas del sistema de captura

Para el sistema de captura se quiso comprobar cual era la disminución del rendimiento de nuestro sistema al aumentar el número de puntos capturados y manejados por este. Esto se hizo calculando la latencia en 3 casos:

- Desde que se produce el contacto del dedo en la pantalla, hasta que la cámara capta los puntos.
- Desde que la aplicación recupera los puntos capturados por la cámara hasta que estos son mostrados (ya en coordenadas de la pantalla) en la pantalla.
- Desde que la aplicación recupera los puntos capturados por la cámara hasta que estos son enviados mediante sockets a la aplicación cliente.

Esta prueba consiste en aumentar secuencialmente los puntos de contacto realizados hasta unos 25 contactos simultáneos. Con esto lo que se deseaba comprobar era si realmente el sistema podría soportar el manejo de una cantidad de puntos considerable. Para su realización fue necesaria únicamente la colaboración de 3 de los alumnos comentados anteriormente.

5.1.2 Descripción de las pruebas del sistema de visualización

En el sistema de visualización se quiso comprobar que tipo de luminosidad era la más adecuada para utilizar en el sistema desarrollado. Para esto se utilizaron 3 tipos diferentes de luminosidad: baja, media, alta. Las condiciones en la cuales se encontraba el sistema a la hora de la realización de la prueba eran las mismas o muy parecidas a las que se podría encontrar en su futuro entorno normal de trabajo.

La primera prueba consistió en organizar 3 grupos de 5 alumnos. Cada unos de estos observaron el sistema en los 3 niveles de luminosidad comentados en el párrafo anterior y apuntaron de forma anónima en una hoja si veían correctamente o no la imagen en cada uno de los niveles.

Por otro lado, la segunda prueba radicó en escoger cual sería el tamaño más adecuado para la pantalla del sistema. Como en la prueba anterior, los alumnos también se organizaron en 3 grupos de 5 personas y se les dio a elegir entre 3 tamaños de pantalla diferente: pequeña (unas 26”), mediana (unas 30“) o grande (unas 36”). Los alumnos, al igual que en el ejercicio anterior, anónimamente escogieron el tamaño de pantalla que les resultó más apropiado y cómodo a la hora de realizar el trabajo.

Cabe destacar que siempre se va a trabajar con este sistema en lugares cerrados por lo tanto no tendría sentido que las pruebas a realizar se desarrollaran en un entorno diferente.

5.1.3 Descripción de las pruebas de la aplicación

Hasta este momento se han desarrollado todos los subsistemas necesarios para realizar la captura de puntos, manejo y visualización de los mismos: implementación del sistema físico, del sistema de captura y del sistema de visualización. Pero para demostrar el correcto funcionamiento de todo este trabajo, y las diferentes posibilidades de la tecnología “Single-Touch” y “Multi-Touch” se van a realizar diferentes tipos de pruebas con la librería:

5.1.3.1 Pruebas unitarias

Estas pruebas consisten en verificar el correcto funcionamiento de cada función de la librería por separado. Las pruebas que se van a realizar y las funciones sobre las que se van a realizar se comentan a continuación:

- **Captura y seguimiento de puntos:** Examinar si realmente los puntos capturados corresponden a los captados por la cámara y si se realiza el seguimiento de los mismos correctamente (en cada frame los puntos no cambian de id, se respetan el tiempo de vivo, etc).

- **Paso a coordenadas de pantalla:** Verificar si realmente la conversión de coordenadas de la cámara en coordenadas de la pantalla se realiza correctamente. En este punto queda implícita la comprobación de los puntos de calibración. Esto es así ya que son los que se utilizan para el paso de puntos a coordenadas de la pantalla.

- **Arquitectura Cliente-Servidor:** Comprobación en el paso de los puntos y los eventos desde el servidor (librería) a una aplicación de prueba (cliente).

- **Utilización de un punto como manejador del ratón de Windows:** Comprobación de que realmente podemos utilizar el primer punto de contacto pantalla como ratón de Windows.

- **Reconocimiento de gestos:** Comprobación de las funciones del sistema de reconocimiento de gestos, mediante la cual, el usuario puede ejecutar distintas acciones realizando los gestos codificados.

5.1.3.2 Pruebas de conjunto

Las pruebas de conjunto por el contrario se encargan de comprobar el buen funcionamiento de la librería en sí. El objetivo principal de estas pruebas es comprobar si realmente el sistema es “Single-Touch”, y lo más importante, si realmente es “Multi-Touch”. Esto se realizará mediante dos aplicaciones de prueba:

- **Dialogo Muestra de Puntos:** Comprobar que es posible la muestra de los puntos en la pantalla mediante MFC. Esta fue la razón por la que se desestimó la utilización de Windows Forms (opción en un principio escogida).

- **Aplicación Phun v 5.28:** Aplicación libre cuya estructura permite ser un perfecto banco de pruebas “Single-Touch”.

- **Aplicación OpenGL:** Aplicación de manejo de imágenes, videos, imágenes 3D mediante la cual se podrá comprobar si el sistema realmente es Multi-Táctil. En esta aplicación se puede realizar zoom con dos dedos, rotar las imágenes, etc…

Con estas aplicaciones, se pretende demostrar la funcionalidad del trabajo realizado en este proyecto, y el amplio campo de investigación que se abre en torno al tema de la tecnología Multi-Touch, la cual tiene un fuerte aliciente actualmente.

5.2 Resultados y Discusión

En este apartado se comentaran los resultados de las pruebas realizadas para los diferentes subsistemas del sistema final. Este punto es muy importante en la realización de un proyecto ya que todo desarrollo de todo proyecto tiene como finalidad que sea productivo, es decir, que arroje un resultado favorable entre los precios y los costes y esto únicamente es posible si los resultados de las pruebas salen adecuados a lo esperado.

Esta discusión por lo tanto, debe tener en cuenta cuál es la finalidad del sistema para poder lograr dilucidar cuales de los resultados obtenidos serán positivos y cuales de ellos serán negativos para el sistema (si es que los hay).

A continuación se comentarán cada uno de los resultados obtenidos en las pruebas realizadas. Estos estarán divididos en los apartados comentados en el punto anterior.

5.2.1 Resultados del sistema de captura

El análisis de las pruebas del sistema de captura se dividió en las tres partes que se describieron en el apartado “5.1.1 Descripción de las pruebas del sistema de captura”:

- **Latencia contacto-captura:** Esta latencia se observó que venía asociada al número de frames por segundo captados por la cámara infrarroja. Esto quiere decir que, la latencia mínima/máxima será de 1000/nº de frames. En nuestro caso el número de frames de la cámara será de 45, por lo tanto, la latencia será igual a 0,022 segundos. Podría ocurrir que llegara un momento en que la cámara no consiguiera captar todos los puntos en un solo frame, por lo que, únicamente en ese hipotético caso la latencia en este punto sería mayor a la comentada.

  Cabe destacar que en la pruebas realizadas no existió ningún problema por lo tanto la latencia fue de 0,022 segundos (hasta 25 ptos).

- **Latencia captura-muestra:** En este punto se realizaron las pruebas especificadas en el apartado “5.1.1”. Estas pruebas consistieron, como se comentó anteriormente, en ir calculando los tiempos desde que se recoge el primer punto y se muestra, hasta que existen se han recogido 25 puntos y están mostrándose en la pantalla. Estas pruebas dieron como resultado que no existía diferencia alguna entre 1 y 25 puntos, la latencia era exactamente la misma para todos los casos. Esta latencia era de 0,015 segundos.

- **Latencia captura-envío:** Por último se realizaron las mediciones de tiempo para el envío mediante sockets a una aplicación cliente. En este caso, la latencia era mayor

que para el punto anterior pero siguió siendo la misma aunque se aumentara el número de puntos. En este caso la latencia fue de 0,032 segundos.

Conclusión

Observando los resultados obtenidos se puede concluir que, en ningún caso, para el número máximo de puntos (25) para el que se han realizado las pruebas, existe ningún tipo de impedimento que no permita su utilización en el sistema, ya que, la latencia es la misma que para 1 solo punto. No obstante, el número de puntos que se ha considerado suficiente por las personas implicadas en el proyecto es de 10. Así mismo, si en cualquier momento se deseara aumentar este número (cosa poco probable), la modificación a realizar sería sumamente sencilla.

Por otro lado, también se ha concluido, que la latencia es la misma en ese rango de puntos ya que las operaciones realizadas no suponen un gran esfuerzo para la CPU. Esto quiere decir, que no existe una diferencia apreciable entre hacer los cálculos para uno y para 25 puntos. Si se aumentara el número de puntos a una cantidad muy grande (200 puntos) se apreciarían los aumentos en las latencias.

Para concluir, en las gráficas siguientes se puede observar visualmente todo lo comentado en este punto para su mejor comprensión.

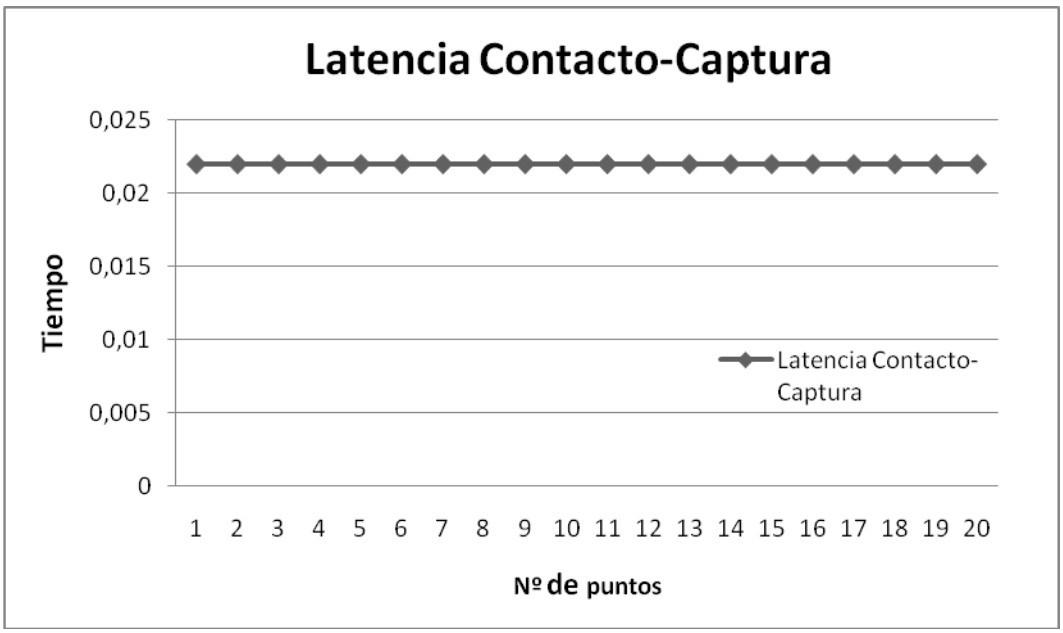


Figura 78. Gráfica de la latencia Contacto-Captura

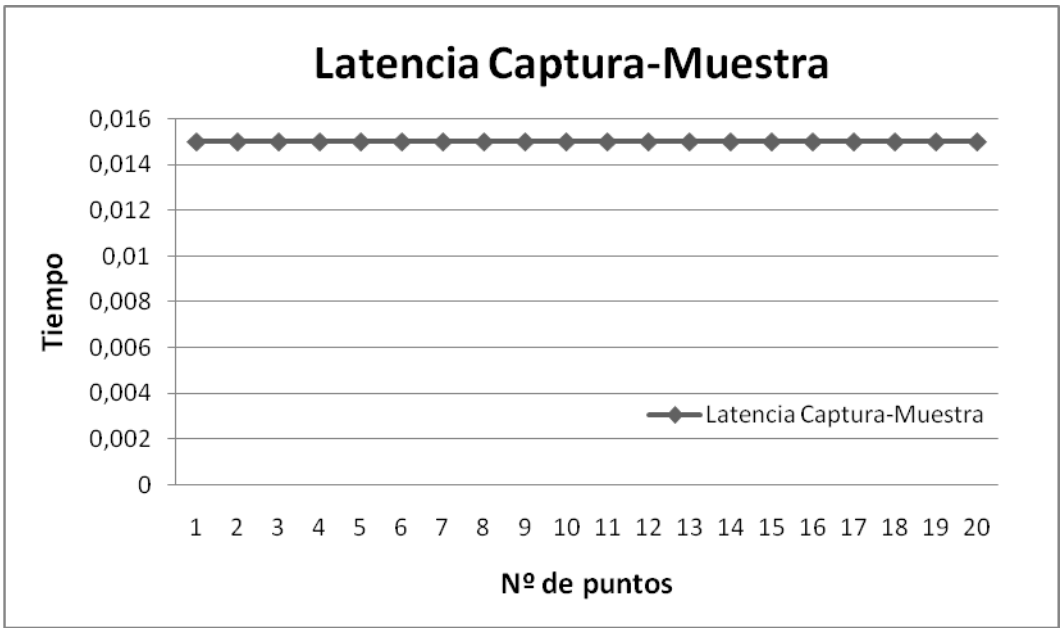


Figura 79. Gráfica de la latencia Captura-Muestra

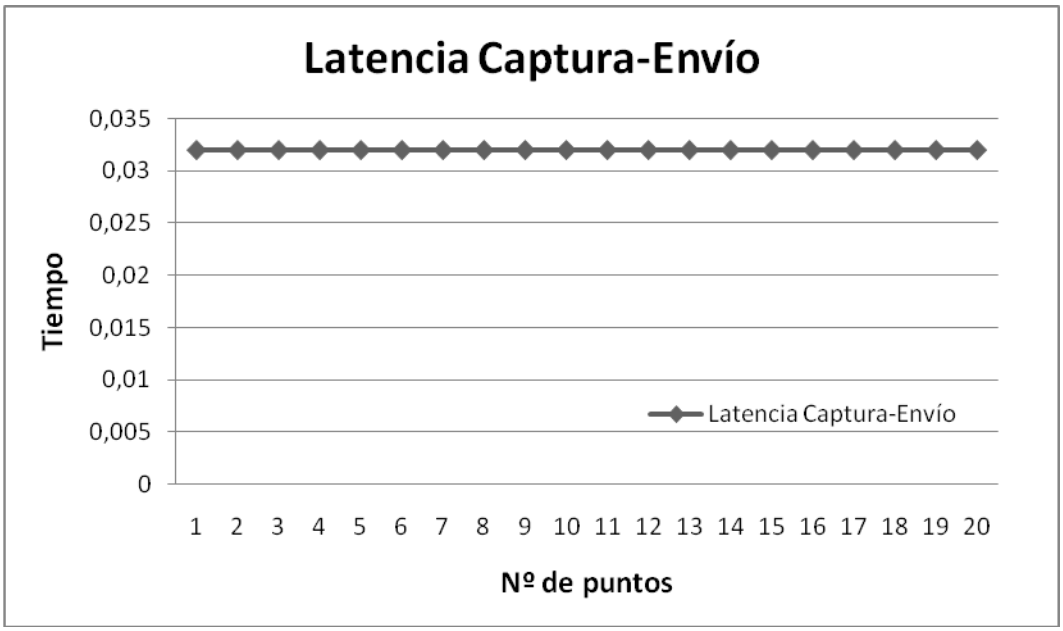


Figura 80. Gráfica de la latencia Captura-Envío.

La “Figura 78” corresponde a la gráfica de la latencia que se produce entre que se realiza el contacto hasta que la cámara recoge los puntos. El eje X corresponde al número de puntos que se capturarán como máximo para su procesamiento, y el eje Y al tiempo transcurrido (latencia). En ella se puede observar que el tiempo es el mismo sea cual sea el número de puntos.

En la “Figura 79” se observa la latencia entre la captura de los puntos hasta que estos se muestran por pantalla. La descripción de los ejes es la misma que en la gráfica anterior, el eje Y corresponde al tiempo transcurrido entre la recepción de los puntos que provienen de la cámara y su muestra por pantalla, y el eje X es el número de puntos capturados en cada caso. Como se puede observar, el tiempo es el mismo independientemente de los puntos que se hayan capturado.

A simple vista se puede observar que el tiempo transcurrido entre dos frames (0,022 s) es superior al tiempo necesario para realizar la lógica de los puntos capturados en el frame (0,015 s). Esto indica la posibilidad de procesar todos los frames recibidos de la cámara. Si, por el contrario, el tiempo de procesamiento de cada frame fuera inferior al tiempo de procesamiento de los puntos capturados, esto indicaría que, en algunos casos, llegaría uno nuevo antes de que se finalizara el procesamiento de los puntos del frame anterior, por lo tanto, se deberían desechar algunos frames de la cámara.

Por último, en la “Figura 80” se puede contemplar la gráfica obtenida de la latencia entre la captura de los puntos y el envío de estos a la aplicación cliente. Los ejes son los mismos que en las gráficas comentadas anteriormente. Como resultado observamos que el tiempo es mayor que en la gráfica anterior, pero, es el mismo para todas las pruebas de puntos que se realizaron.

5.2.2 Resultados del sistema de visualización

Cabe destacar antes de comenzar con la discusión de las conclusiones obtenidas que tanto las pruebas para la luminosidad como las de elección del tamaño de la pantalla se hicieron en 3 días diferentes. Esto podría haber afectado si el lugar de la prueba hubiera sido al aire libre, pero como se trataba del “Visionarium” que se encuentra dentro del Instituto de Robótica no afectó en absoluto a los resultados.

Los resultados obtenidos para las diferentes luminosidades que se ponían a prueba fueron los siguientes:

- El 40% de los alumnos, 6 de los 15, comentaron que se veía correctamente y se podía trabajar perfectamente con **luminosidad baja**. En ningún momento de la prueba ningún alumno se quejó de molestias en los ojos ni nada por el estilo aunque en algunos casos los alumnos tenían dificultad para diferenciar bien algunas palabras u objetos

- El 80% de los mismos, 12 de los 15, explicaron que con **luminosidad media** también se podía realizar las tareas a las que eran sometidos sin ningún esfuerzo visual. A estos niveles de luminosidad tampoco hubo ningún sujeto al que le molestaran los ojos en el transcurso de la prueba.

- Mientras que solo el 13% de los alumnos, 2 de 15, comentaron que con **luminosidad alta** las tareas se podían realizar correctamente. La mayoría de ellos decía que le molestaba en los ojos al cabo de estar un par de minutos ”trabajando” la intensidad de la luz a esos niveles tan elevados.

Por otro lado, para la segunda prueba se produjeron los siguientes resultados:

- Únicamente 1 de los 15 alumnos a los que se le realizó la prueba, el 6%, comentó que el tamaño de 26 pulgadas era suficiente para la realización de las pruebas utilizando la pantalla.

- El 53% de los alumnos, 8 de los 15, concluyó que con una pantalla de unas 30 pulgadas los trabajos que les fueron requeridos se podían completar perfectamente y sin ningún esfuerzo. El 43% restante comentó que el trabajo se podía realizar pero existían momentos en que la pantalla se quedaba pequeña, lo que producía que se complicara la realización del ejercicio.

- Por último el 100% de los alumnos, 15 de 15, observó a la hora de realizar las pruebas que mediante este tamaño de pantalla la realización era más sencilla que con las anteriores. En ningún momento la pantalla se quedó pequeña para ninguno de los usuarios y esta puede ser una de las razones por la que los ejercicios parecían más sencillos de realizar que en los casos anteriores.

Conclusión

Como conclusión y observando los resultados obtenidos se decide que la luminosidad más adecuada para su utilización en el sistema es la **luminosidad media.** Utilizando este nivel la mayoría de los usuarios se sintieron completamente a gusto realizando las pruebas sin ningún dolor ni molestia en los ojos.

También es importante destacar que los usuarios realizaron las pruebas en un menor tiempo que con las otras luminosidades utilizadas. Esto puede tomarse como otro punto más a favor de la luminosidad escogida.

Examinando los resultados obtenidos queda claro que el tamaño de pantalla a escoger debe ser el de 36 pulgadas. Esto queda totalmente claro al ser el 100% de los alumnos preguntados el que escoge esta solución.

Como ocurre en la prueba anterior, los alumnos finalizan antes los ejercicios en la pantalla escogida. Esto puede deberse a que se sienten más cómodos con mayor espacio para realizar el trabajo.

5.2.3 Resultados de las pruebas de la aplicación

Para la realización de estas pruebas se necesitó la colaboración de los 15 alumnos que realizaron la prueba de visualización. Pero a diferencia de la anterior, esta se realizó en 6 tandas donde cada grupo realizó la prueba dos veces. Esto se debió a que en la primera prueba se observó que era necesaria una pequeña toma de contacto con el sistema antes de comenzar la prueba en sí para que los resultados tuvieran validez realmente.

Hay que destacar que los alumnos solo realizaron las pruebas de conjuntos, las pruebas unitarias fueron realizadas por personas asociadas al proyecto.

Como en el punto anterior de **“Resultados de las pruebas de aplicación”** este también se dividirá en dos subsistemas:

5.2.3.1 Resultados pruebas de unitarias

Los resultados de estas pruebas se pueden separar en los siguientes puntos:

- **Captura y manejo de puntos:** Las pruebas realizadas sacaron a la luz pequeños fallos que existían en la librería. Cuando dos puntos estaban muy cerca, en ocasiones se producía un cambio en el “id” del punto. Esto se soluciono colocando una “distancia mínima” entre dos puntos. Otro fallo encontrado fue que en ocasiones la cámara captaba pequeños reflejos que reconocía como puntos lo que hacía que engañara a la librería. La solución a esto fue colocar en el código un “tamaño mínimo” para que un punto fuera “real”.

- **Paso a coordenadas de pantalla:** Mediante las pruebas realizadas se comprobó que el paso a coordenadas de la pantalla no era totalmente correcto. Existía un pequeño desplazamiento que aumentaba cuando los puntos de contacto se alejaban del (0,0). Esto se descubrió que era porque la función de OpenCv utilizada para pasar a coordenadas de la pantalla utilizaba también la coordenada Z (profundidad) que en nuestro caso no se tenía en cuenta. La solución fue dividir las coordenadas X e Y resultantes entre la Z resultante.

- **Arquitectura Cliente-Servidor:** En las pruebas realizadas en este punto no se detectó ningún fallo. El único fallo posible era que se perdieran paquetes si se utilizaban sockets UDP pero esto se solvento al implementar la arquitectura mediante sockets TCP.

- **Punto como manejador de ratón de Windows:** Al realizar las pruebas se observó que en ocasiones se perdían algunos eventos enviados al manejador del ratón. Esto se solvento revisando el código y verificando en cada caso que se enviaba el evento correspondiente.

- **Reconocimiento de gestos:** En este punto surgieron problemas típicos de la codificación. Estos problemas consistían en que era muy fácil “engañar” al reconocedor de los gestos, es decir, cualquier movimiento “extraño” era reconocido como un gesto. La única solución posible a esto fue recodificar los mismos.

5.2.3.2 Resultados pruebas de conjunto

Estos resultados fueron evaluados por personas implicadas en el proyecto pero las pruebas fueron realizadas por los alumnos mencionados anteriormente. La división será la misma que en el punto de descripción:

- **Dialogo de muestra de puntos:** En estas pruebas se observó que realmente no existía ningún problema. Las pruebas fueron completamente satisfactorias.

- **Aplicación Phun v. 5.28:** Con esta aplicación se comprobó el funcionamiento de la librería como un sistema “Single-Touch”. Mediante las pruebas realizadas a los alumnos se comprobó que la respuesta de la librería fue excelente. Además los usuarios concluyeron en que la interactividad con el sistema era muy buena. También cabe destacar que una única calibración de los parámetros de la cámara fue suficiente para que todos los alumnos realizaran la prueba satisfactoriamente (no fue necesario modificar la configuración para cada alumno).

- **Aplicación OpenGL:** Como ya es sabido, mediante esta aplicación se comprobó que el sistema desarrollado era realmente “Multi-Touch”. Al realizar las pruebas de utilización de la aplicación se comprobó que al utilizar dos puntos para rotar o ampliar una imagen se perdían algunos eventos por lo que se producían errores en la aplicación. Esto se solucionó mediante la revisión del código. Así mismo, otro fallo que se encontró fue a la hora de ampliar o rotar la imagen, la ampliación o rotación no se realizaba correctamente. La solución se obtuvo revisando las operaciones realizadas en el código, ya que al realizar la rotación el offset de la misma debía ser cero en el ángulo que formaban los dos dedos, no en el ángulo que formaban las esquinas inferiores de la foto.

La conclusión general de estas pruebas fue que los usuarios del sistema necesitan un periodo de adaptación al sistema. Este tiempo dependerá de lo familiarizados que esté el usuario en cuestión con sistemas de este tipo.

5.3 Evaluación presupuestaria

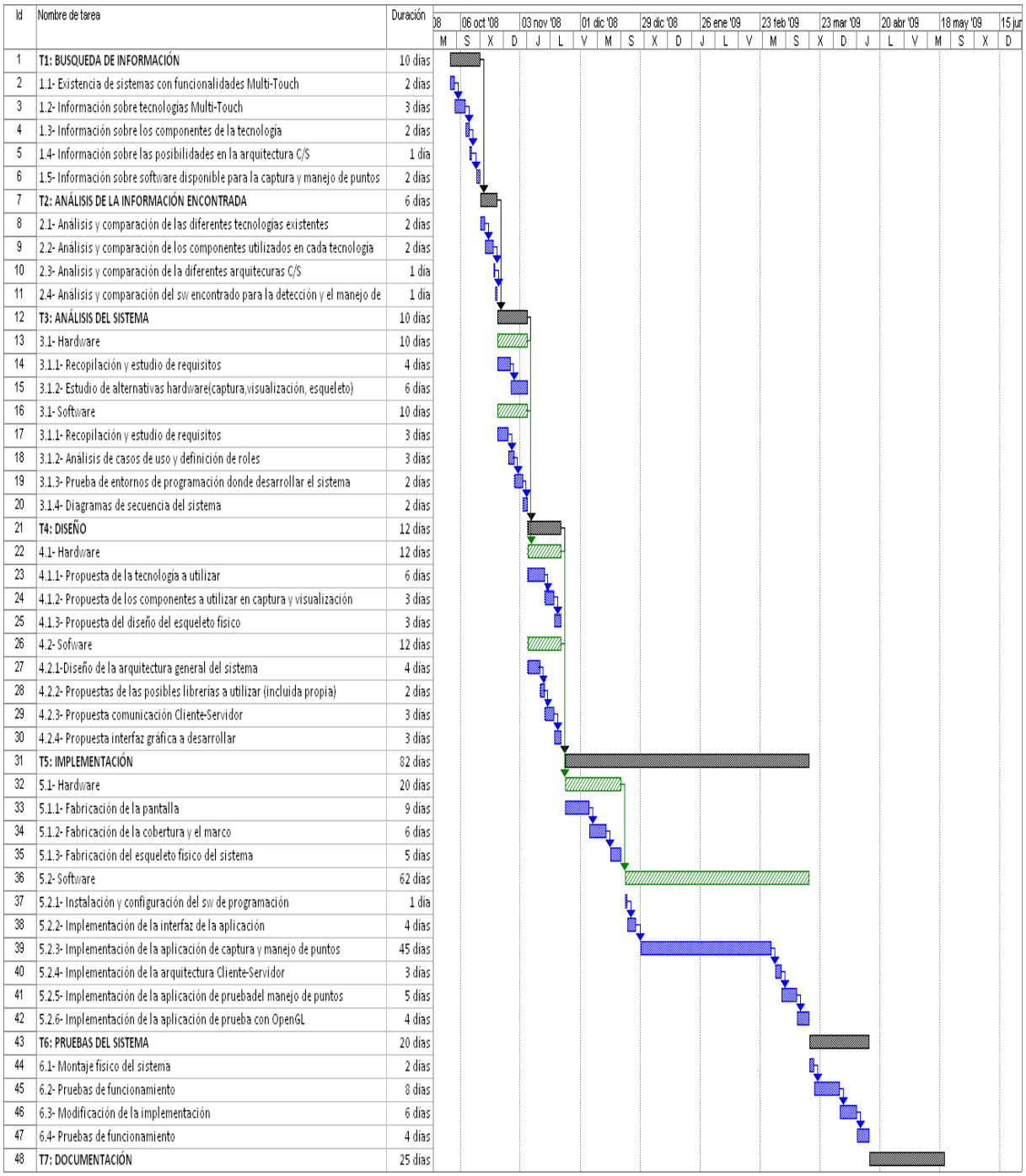
Tras dar por finalizado el desarrollo y las pruebas del sistema, se pueden hacer los cálculos reales de lo que ha costado, y por tanto, el precio al que se venderá el sistema a las empresas interesadas.

5.3.1- Coste final de desarrollo

El proyecto ha sido realizado en 8 meses trabajando una media de 8 horas diarias 5 días a la semana, una sola persona. Esto hace un total de 800 horas. La justificación de la diferencia entre el tiempo real y el calculado mediante el método COCOMO2, en la Sección 4.3 de la documentación se basa en cuatro aspectos:

- Se ha conseguido reutilizar algo de código. En los cálculos iniciales no se planteó esta posibilidad ya que no se conocía la existencia del mismo.
- La empresa NaturalPoint facilitó una pequeña aplicación de ejemplo que ha sido de utilidad en el subsistema de captura de puntos de nuestra aplicación. Gracias a este ejemplo se ha podido realizar un desarrollo incremental en lugar de comenzar de cero.
- Programar con MFC de Windows no es tan complicado como parece en un principio y después de un tiempo trabajando con él todo se hace más sencillo. Por lo tanto la experiencia indicada en un principio en el cálculo aumentó rápidamente.
- El modelo COCOMO2 es un modelo muy pesimista. Debido a sus cálculos internos, siempre se obtiene una aproximación al coste temporal del proyecto mayor de lo que es en realidad.

Para ver la separación en subtareas de estos 8 meses, de forma gráfica, se utilizará un diagrama de Gantt. Como podrá observarse la gran diferencia de tiempo está en el punto de implementación, donde se ha pasado de 107 a 82 días (un mes menos de lo previsto en nuestra estimación anterior). De estos 25 días de desfase, 21 (mas de un mes) corresponden a la estimación software y solo 4 (una semana) a la hardware.



Id	Nombre de tarea	Duración
1	**T1: BUSQUEDA DE INFORMACIÓN**	10 días
2	1.1- Existencia de sistemas con funcionalidades Multi-Touch	2 días
3	1.2- Información sobre tecnologías Multi-Touch	3 días
4	1.3- Información sobre los componentes de la tecnología	2 días
5	1.4- Información sobre las posibilidades en la arquitectura C/S	1 día
6	1.5- Información sobre software disponible para la captura y manejo de puntos	2 días
7	**T2: ANÁLISIS DE LA INFORMACIÓN ENCONTRADA**	6 días
8	2.1- Análisis y comparación de las diferentes tecnologías existentes	2 días
9	2.2- Análisis y comparación de los componentes utilizados en cada tecnología	2 días
10	2.3- Analisis y comparación de la diferentes arquitecuras C/S	1 día
11	2.4- Análisis y comparación del sw encontrado para la detección y el manejo de	1 día
12	**T3: ANÁLISIS DEL SISTEMA**	10 días
13	3.1- Hardware	10 días
14	3.1.1- Recopilación y estudio de requisitos	4 días
15	3.1.2- Estudio de alternativas hardware(captura,visualización, esqueleto)	6 días
16	3.1- Software	10 días
17	3.1.1- Recopilación y estudio de requisitos	3 días
18	3.1.2- Análisis de casos de uso y definición de roles	3 días
19	3.1.3- Prueba de entornos de programación donde desarrollar el sistema	2 días
20	3.1.4- Diagramas de secuencia del sistema	2 días
21	**T4: DISEÑO**	12 días
22	4.1- Hardware	12 días
23	4.1.1- Propuesta de la tecnología a utilizar	6 días
24	4.1.2- Propuesta de los componentes a utilizar en captura y visualización	3 días
25	4.1.3- Propuesta del diseño del esqueleto físico	3 días
26	4.2- Sofware	12 días
27	4.2.1-Diseño de la arquitectura general del sistema	4 días
28	4.2.2- Propuestas de las posibles librerías a utilizar (incluida propia)	2 días
29	4.2.3- Propuesta comunicación Cliente-Servidor	3 días
30	4.2.4- Propuesta interfaz gráfica a desarrollar	3 días
31	**T5: IMPLEMENTACIÓN**	82 días
32	5.1- Hardware	20 días
33	5.1.1- Fabricación de la pantalla	9 días
34	5.1.2- Fabricación de la cobertura y el marco	6 días
35	5.1.3- Fabricación del esqueleto físico del sistema	5 días
36	5.2- Software	62 días
37	5.2.1- Instalación y configuración del sw de programación	1 día
38	5.2.2- Implementación de la interfaz de la aplicación	4 días
39	5.2.3- Implementación de la aplicación de captura y manejo de puntos	45 días
40	5.2.4- Implementación de la arquitectura Cliente-Servidor	3 días
41	5.2.5- Implementación de la aplicación de pruebadel manejo de puntos	5 días
42	5.2.6- Implementación de la aplicación de prueba con OpenGL	4 días
43	**T6: PRUEBAS DEL SISTEMA**	20 días
44	6.1- Montaje físico del sistema	2 días
45	6.2- Pruebas de funcionamiento	8 días
46	6.3- Modificación de la implementación	6 días
47	6.4- Pruebas de funcionamiento	4 días
48	**T7: DOCUMENTACIÓN**	25 días

Tabla 14: Diagrama de Gantt real

Con estos 8 meses y un coste mensual de 2.500,00€ por trabajador (cabe destacar que en el coste mensual del trabajador se incluyen los costes sociales, donde el 70% es el sueldo y el 30% los costes), el coste del personal pasaría de los 23750,00€ calculados mediante la estimación anterior a 20000,00€. El coste de hardware y software se mantendría tal y como se calculó inicialmente.

La siguiente tabla muestra el desglose del coste real de desarrollo:

Descripción	Subtotal (€)
Recursos hardware	3072,00
Recursos software	259,00
Recursos de personal (1750,00€ de salario + 30% de gastos sociales)	20000,00
TOTAL	23331,00

Tabla 15: Coste real del desarrollo

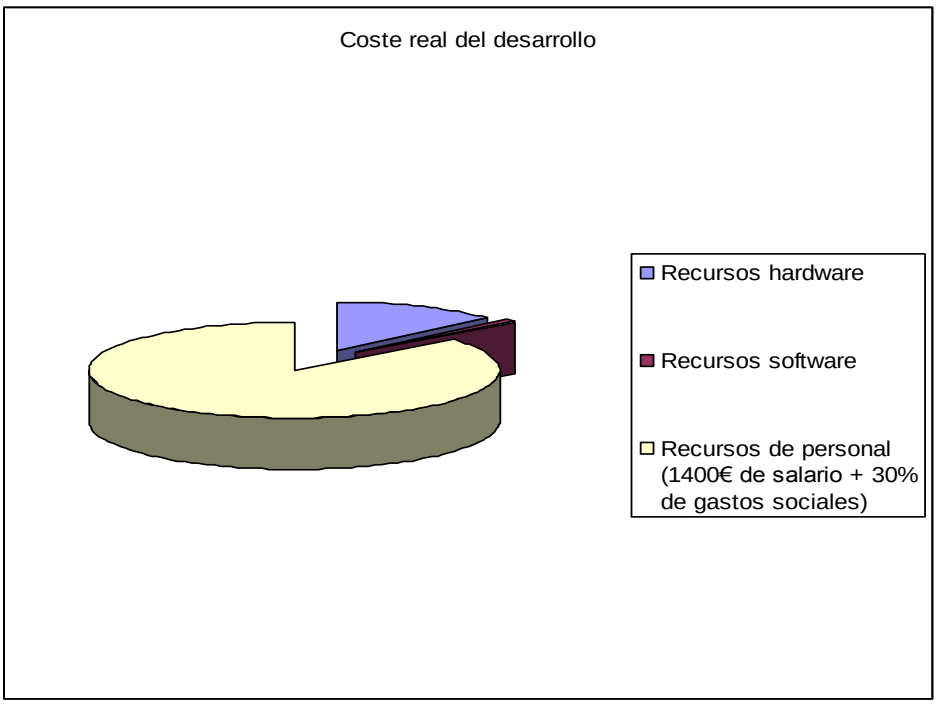


Figura 81. Gráfica del coste real del desarrollo

6.- Conclusiones y trabajo futuro

Analizados los resultados del proyecto se puede concluir que a la finalización del mismo se ha obtenido:

- Evaluación técnica de las principales tecnologías de soporte para el desarrollo del Sistema Tipo Mesa Interactiva. Esta evaluación incluye tanto la tecnología en si, como los componentes necesarios para su implantación.

- Montaje real de un prototipo de mesa a partir de la tecnología escogida en el punto anterior. Este prototipo supone el subsistema físico del sistema, la cual está formada por los tres subsistemas comentados durante toda la documentación: el sistema de captura, el sistema de visualización y el esqueleto físico del mismo.

- Desarrollo de la librería mediante la cual se realiza la gestión de los puntos capturados por la cámara infrarroja que se encuentra en el sistema de captura. Esta librería además se utiliza como servidor de puntos y eventos. Estos, por lo tanto, pueden ser utilizados en aplicaciones tanto remotas como locales. Además de todo esto se puede controlar opcionalmente el puntero de Windows y reconocer algunos de los gestos definidos en el punto “2.3 Reconocimiento de Gestos” de la documentación.

  Por último cabe destacar que la librería posee un interfaz mediante el cual es posible modificar los parámetros más relevantes de la cámara infrarroja así como habilitar opciones de la misma librería (calibrar pantalla, habilitar mouse, etc). Mediante el interfaz es posible realizar un seguimiento en tiempo real de los puntos capturados por la cámara.

- Desarrollo de una aplicación en OpenGL mediante la cual se prueba que la librería realizada es Multi-Touch. Esta aplicación actúa como cliente de la librería comentada en el punto anterior a la que se le pasan los puntos y los eventos para cada frame de la cámara.

- El coste final del sistema es de poco más de 11000,00€ (muy bajo para el precio que suelen costar los sistemas que utilicen esta tecnología). El sistema desarrollado puede ser utilizado por cualquier empresa como un dispositivo mediante el cual se permite el trabajo cooperativo de dos trabajadores, facilidad en el uso de algunas aplicaciones, etc. Esto repercute directamente en un aumento de la velocidad en la que se realiza el trabajo en esas empresas. Por último se ha realizado un estudio de viabilidad económica donde se demuestra que la implementación de este sistema provocaría un incremento en los ingresos en la misma. Esto se debe al aumento de la producción comentado anteriormente.

Una vez comentado todo la anterior y llegados a este punto, se proponen las posibles mejoras al sistema que se explican a continuación:

- Por un lado se tienen las posibles mejoras del hardware. Estas mejoras son el resultado de cambiar los componentes escogidos en el desarrollo del sistema por otros que cumplan mejor el papel que deben realizar. Se debe tener en cuenta que este cambio en los componentes en la mayoría de los casos supone un aumento significativo del coste del proyecto. Por ello se debe tener muy claro cuál va a ser el aumento de prestaciones del sistema con el nuevo componente y si merece la pena el cambio.

  En concreto, una mejora bastante importante sería el cambio de la pantalla de plexiglás por otra de otro material mediante el cual el deslizamiento del dedo fuera mejor que en el sistema actual.

  Otra posible mejora puede ser la modificación del diseño del esqueleto físico por otro más trabajado. En este caso, al tratarse de un prototipo ese aspecto no se ha cuidado en demasía.

- Por otro lado tenemos las mejoras del software. Estas mejoras consisten en realizar cambios en el software que aumente el rendimiento de la aplicación. Este aumento puede conseguirse utilizando otro algoritmo de ordenación más eficiente, mejorando el algoritmo de reconocimiento de puntos, etc.

  También es posible mejorar la interfaz web de la librería dando más opciones al usuario, como por ejemplo dar la opción de escoger el número de puntos que desea detectar en ese momento, lo que indirectamente podría llegar a aumentar el rendimiento.

- Por último se pueden desarrollar nuevas aplicaciones cliente para el sistema desarrollado o incluso mejorar las realizadas en este proyecto.

Para finalizar, como puede comprobarse con el presente texto, se ha documentado cada una de las partes del proyecto, confiando que sirva para futuros progresos en el campo de la tecnología “Multi-Touch”.

7. Referencias

[1] Han, J. Y. "Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection". In Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology. 2005.

[2] *SK Lee, W. Buxton and K. C. Smith. "*A multi-touch three dimensional touch-sensitive tablet*". In Proceedings of CHI '85, pages 21–25, New York, NY, USA, 1985. ACM Press.*

[3] H. Benko , A. D. Wilson and R. Balakrishna**. "**Sphere: Multi-Touch Interactions on a Spherical Display". Microsoft Research and One Microsoft Way, Redmond, WA, USA.

[4] M. Thörnlund. "Gesture analyzing for Multi-Touch Screen Interfaces". Lulea University of Tecnology. 2007.

[5] A. D Wilson. "TouchLight: An imaging Touch Screen and Display for Gesture-Based Interaction". International Conference on Multimodal Interfaces. State College, Pennsylvania, USA. October 13-15, 2004.

[6] J. Y. Han. "Design a capacitive touchscreen system". Embedded Design India. 19 Jan 2005.

[7] V. I. Pavlovic, R. Sharma and T. S. Huang. "Visual Interpretation of Hand Gestures for Human-Computer Interaction: Areview". Pattern Analysis and Machine Intelligence, IEE Transactions on. 1997.

[8] D. Rubine. "Specifying Gestures by example". International Conference on Computer Graphics and Interactive Techniques, Proceedings of the 18th annual conference on Computer Graphics and interactive techniques. 1991.

[9] P. Dietz and D. Leigh. "DiamondTouch: A Multi-User Touch Technology". Symposium on User Interface Software and Technology, Proceedings of the 14th annual ACM symposium on User interface software and technology, Orlando, Florida. October 2003.

[10] W. Westerman. "Hand Tracking, Finger Identification and Chordic Manipulation on a Multi-Touch Surface". PhD thesis, University of Delaware. 1999.

[11] Ren, X. and Moriya, S." Improving Selection Performance on Pen-Based Systems: A Study of Pen-Input Interaction for Selection Tasks". ACM Transactions on Computer Human Interaction 7(3). pp.384-416 .

[12] R. Downs. Using resistive touch screens for human/machine interface. Technical report, Texas Instruments Incorporated, Texas Instruments, Post O_ce Box 655303, Dallas, Texas 75265, 2005.

[13] K. Hinckley and M. Sinclair .”Touch-Sensing Input Devices”. Microsoft Research, One Microsoft Way, Redmond. 1999.

[14] J. Schöning, P. Brandl, F. Daiber, F. Echtler, O. Hilliges, J. Hook, M. Löchtefeld, N. Motamedi, L. Muller, P. Olivier, T. Roth, U. von Zadow. “Multi-Touch Surfaces: A Technical Guide”. Technical Report TUM-I0833. 2008.

[15] N. Motamedi. “Hd touch: multi-touch and object sensing on a high denition lcd tv”. In CHI '08: CHI'08 extended abstracts on Human factors in computing systems, pages 3069{3074, New York, NY, USA. ACM. 2008.

[16] S. Hodges, S. Izadi, A. Butler, A. Rrustemi, and B. Buxton. “Thinsight: versatile multi-touch sensing for thin form-factor displays”. In UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology, pages 259-268, New York, NY, USA. ACM. 2007

[17] L. Y. L. Muller. “Multi-touch displays: design, applications and performance evaluation”. Master's thesis, University of Amsterdam, Amsterdam, the Netherlands, June 2008.

[18] Boehm, “B.W. Software Engineering Economics”. Prentice Hall, 1981.

[19] Center of Software Engineering, University of Southern California. COCOMO II Model Definition Manual.

[20] R. S. Pressman. “Ingeniería del Software: Un enfoque práctico”. Quinta edición. Mc Graw Hill, 2002.

[21] I. Jacobson, G. Booch, J. Rumbaugh. “The Unified Software Development Process (Addison-Wesley Object Technology Series)”.

[22] Herot, C. & Weinzapfel, G. One point Touch Input of Vector Information from Computer Displays, Computer Graphics, 12(3), 210-216. (1978).

[23] Minsky, M. Manipulating Simulated Objects with Real-World Gestures Using a Force and Position Sensitive Screen, Computer Graphics, 18(3), 195-203.]. 1984.

APÉNDICE A

Primeramente, encontramos los métodos de TouchOptitrack comentados en la implementación del sistema:

```
void CTouchOptitrack::MultMatriz(CvMat* out, PTO_DETEC &list_ptos, int indice, bool up, bool down, bool mover )
{
	double salida[3][3];
	double pto[3];

	static int lastX;
	static int lastY;

	if((list_ptos->encontrado)&&(list_ptos->vivo > 0))
	{
		pto[0] = list_ptos->px;
		pto[1] = list_ptos->py;
		pto[2] = 1;
	}
	else
	{
		pto[0] = 0;
		pto[1] = 0;
		pto[2] = 1;
	}

	for(int i=0; i<3; i++){
		dibujar[indice].result[i] = 0;
		for(int k=0; k<3; k++){
			salida[i][k] = cvGetReal2D( out, i, k);
			dibujar[indice].result[i] = dibujar[indice].result[i] + salida[i][k] * pto[k];
		}
	}
	dibujar[indice].result[0] /= dibujar[indice].result[2];
	dibujar[indice].result[1] /= dibujar[indice].result[2];

	// DEPENDIENDO DE LA ORIENTACIÓN CORREGIR LOS PUNTOS
	if (rotacion == 1)
	{
		dibujar[indice].result[0]=GetSystemMetrics(SM_CXSCREEN)-dibujar[indice].result[0];
		dibujar[indice].result[1]=GetSystemMetrics(SM_CYSCREEN)- dibujar[indice].result[1];
	}

	if (rotacion == 3)
	{
		dibujar[indice].result[0]=GetSystemMetrics(SM_CYSCREEN)- dibujar[indice].result[1];
		dibujar[indice].result[1] = dibujar[indice].result[0];
	}
```

```
		if (rotacion == 4)
		{
			dibujar[indice].result[0] = dibujar[indice].result[1];
			dibujar[indice].result[1]=GetSystemMetrics(SM_CXSCREEN)- dibujar[indice].result[0];
		}

		//COMPROBAMOS QUE EL PUNTO CAPTURADO ESTE DENTRO DEL TAMAÑO DE LA PANTALLA SI
		NO LO ESTA LO PONEMOS COMO NO ENCONTRADO
		if((dibujar[indice].result[0] > GetSystemMetrics(SM_CXSCREEN))||
		  (dibujar[indice].result[1] > GetSystemMetrics(SM_CXSCREEN))||
		  (dibujar[indice].result[0] < 0)||
		  (dibujar[indice].result[1] < 0))
			list_ptos->encontrado = false;

		if (up) {
			SetCursorPos(lastX,lastY);
			mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);
		}

		if((list_ptos->encontrado)&&(list_ptos->vivo > 0))
		{
			if((indice == 0)&&(mouse))
			{
				bool ok;
				lastX = dibujar[indice].result[0];
				lastY = dibujar[indice].result[1];
				if (mover) {
					ok = SetCursorPos(dibujar[indice].result[0],
					dibujar[indice].result[1]);
					if (down)
						mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
				}
			}
		}
		dibujar[indice].area = list_ptos->area;
}


CvMat* CTouchOptitrack::MatricesCalibracion()
{
		//Instanciamos las matrices que vamos a utilizar
		CvMat* mat = cvCreateMat( 4, 2, CV_32FC1 );
		CvMat* mat2 = cvCreateMat( 4, 2, CV_32FC1 );

			...

		// Colocamos en varables auxiliares las coordenadas de las  cruces  y  las  de  los  puntos  de
		calibración
		for(int i = 0; i < 4; i++)
		{
			aux1[i].x=ptos_calib[i]->px;
			aux1[i].y=ptos_calib[i]->py;
			aux2[i].x=cruces[i][0];
			aux2[i].y=cruces[i][1];
		}
```

```
	// Con este método conseguimos una matriz (salida) con la cual //podremos convertir los puntos en coordenadas de la cámara en //puntos en coordenadas de la pantalla
	salida = cvWarpPerspectiveQMatrix( (CvPoint2D32f*)aux1, (CvPoint2D32f*)aux2, salida);

	return salida;
}

void CTouchOptitrack::ComprobarGestos(bool &press, bool &boton)
{
		int cnt = 0;
		bool gestoArribaOk[3] = {false, false, false};
		bool gestoAbajoOk[3] = {false, false, false};
		bool gestoDerechaOk[3] = {false, false, false};
		bool gestoIzdaOk[3] = {false, false, false};

		for(int i = 0; i<10; i++)
		{
			if(GestorCamaras->ListaCamaras[0]->ptos_ant[i]->encontrado)
			{
				cnt++;
			}
		}

		if((cnt == 3)&&(cntOld == 0))
		{
			int i = 0;
			for(int j = 0; j < 10; j++)
			{
				if(GestorCamaras->ListaCamaras[0]->ptos_ant[j]->encontrado)
				{
					ptos_gestos[i]->id = j;
					ptos_gestos[i]->px = GestorCamaras->ListaCamaras[0]
					->ptos_ant[j]->px;
					ptos_gestos[i]->py = GestorCamaras->ListaCamaras[0]
					->ptos_ant[j]->py;
					i++;
				}
			}

			if((!gestoAbajo)&&(!gestoArriba)&&(!gestoDerecha)&&(!gestoIzda))
				cntOld = cnt;
		}
		else if ((cnt == 3) && (cntOld != 0))
		{
			for( int i = 0; i < 10; i++)
			{
				for( int h = 0; h < 3; h++)
				{
```

```
                        If ((GestorCamaras->ListaCamaras[0]->ptos_ant[i]
                              ->encontrado) &&
                              (ptos_gestos[h]->id == i) &&
                              ((ptos_gestos[h]->px-GestorCamaras->ListaCamaras[0]
                              ->ptos_ant[i]->px) < 25) &&
                              ((ptos_gestos[h]->py-GestorCamaras->ListaCamaras[0]
                               ->ptos_ant[i]->py) >= 40))
                              {
                                    gestoArribaOk[h] = true;
                                    gestoArriba = true;
                              }
                         else if((GestorCamaras->ListaCamaras[0]->ptos_ant[i]
                               ->encontrado)&&
                               (ptos_gestos[h]->id == i) &&
                                 ((ptos_gestos[h]->px – GestorCamaras
                                  ->ListaCamaras[0]->ptos_ant[i]->px) < 25) &&
                                 ((ptos_gestos[h]->py – GestorCamaras
                                 ->ListaCamaras[0]->ptos_ant[i]->py) <= -40))
                                 {
                                    gestoAbajoOk[h] = true;
                                    gestoAbajo = true;
                              }
                         else if((GestorCamaras->ListaCamaras[0]->ptos_ant[i]
                                  ->encontrado)&&
                                 (ptos_gestos[h]->id == i) &&
                                 ((ptos_gestos[h]->px – GestorCamaras
                                  ->ListaCamaras[0]->ptos_ant[i]->px) >= 40) &&
                                 ((ptos_gestos[h]->py – GestorCamaras
                                 ->ListaCamaras[0]->ptos_ant[i]->py) < 25))
                                 {
                                    gestoIzdaOk[h] = true;
                                    gestoIzda = true;
                              }
                         else if((GestorCamaras->ListaCamaras[0]->ptos_ant[i]
                                  ->encontrado) &&
                                (ptos_gestos[h]->id == i) &&
                                 ((ptos_gestos[h]->px – GestorCamaras
                                  ->ListaCamaras[0]->ptos_ant[i]->px) <=-40) &&
                                 ((ptos_gestos[h]->py – GestorCamaras
                                  ->ListaCamaras[0]->ptos_ant[i]->py) < 25))
                              {
                                    gestoDerechaOk[h] = true;
                                    gestoDerecha = true;
                         }
                   }
              }
}
```

```
else if((cnt == 1)&&(cntOld == 0))
{
        frames = 0;

        for (int i = 0; i < 10; i++)
        {
                if(GestorCamaras->ListaCamaras[0]->ptos_ant[i]->encontrado)
                {
                        IniCircx = GestorCamaras->ListaCamaras[0]->ptos_ant[i]
                        ->px;
                        IniCircy = GestorCamaras->ListaCamaras[0]->ptos_ant[i]
                        ->py;
                        Centrox = GestorCamaras->ListaCamaras[0]->ptos_ant[i]
                        ->px;
                        Centroy = GestorCamaras->ListaCamaras[0]->ptos_ant[i]
                        ->py + 40;
                        primerPunto = true;

                        Centro[frames] = sqrt(pow((GestorCamaras
                        ->ListaCamaras[0]->ptos_ant[i]->px-Centrox),2)+
                        pow((GestorCamaras->ListaCamaras[0]->ptos_ant[i]->py -
                        Centroy),2));
                        frames++;
                }
        }
        cntOld = cnt;
}
else if((cnt == 1)&&(cntOld != 0))
{
        for (int i = 0; i < 10; i++)
        {
                if(GestorCamaras->ListaCamaras[0]->ptos_ant[i]->encontrado)
                {
                    if((GestorCamaras->ListaCamaras[0]->ptos_ant[i]
                        ->px>=IniCircx+5)&&
                        (GestorCamaras->ListaCamaras[0]->ptos_ant[i]->py>=
                        IniCircy+5))
                      {
                        primerPunto = false;
                        Centro[frames] = sqrt(pow((GestorCamaras
                        ->ListaCamaras[0]->ptos_ant[i]->px -Centrox),2) +
                        pow((GestorCamaras->ListaCamaras[0]->ptos_ant[i]->py -
                        Centroy),2));
                        frames++;
                      }

                }
        }
}
```

```
else if((cnt != 3)&&(cnt != 1))
 {
        cntOld = 0;
 }

 for(int i = 0; i <= 2; i++)
 {
        if(!gestoAbajoOk[i])
            gestoAbajo = false;

        if(!gestoArribaOk[i])
            gestoArriba = false;

        if(!gestoDerechaOk[i])
            gestoDerecha = false;

        if(!gestoIzdaOk[i])
            gestoIzda = false;
 }

 if(gestoAbajo)
 {
        system("rundll32.exe nvcpl.dll,dtcfg rotate 1 0");
        cntOld = 0;
        gestoAbajo = false;
        Sleep(500);
        rotacion = 2;
 }
 if(gestoArriba)
 {
        system("rundll32.exe nvcpl.dll,dtcfg rotate 1 180");
        cntOld=0;
        gestoArriba = false;
        Sleep(500);
        rotacion = 1;
 }
 if(gestoDerecha)
 {
        system("rundll32.exe nvcpl.dll,dtcfg rotate 1 90");
        cntOld=0;
        gestoDerecha = false;
        Sleep(500);
        rotacion = 4;
 }
```

```
if(gestoIzda)
{
	system("rundll32.exe nvcpl.dll,dtcfg rotate 1 270");
	cntOld=0;

	gestoIzda = false;
	Sleep(500);
	rotacion = 3;
}


for(int j = 0; j < 10; j++)
{
	int pos = 0;

	if((GestorCamaras->ListaCamaras[0]->ptos_ant[j]->encontrado)&& (!primerPunto))
	{
		for (int i = 0; i < frames-1; i++)
		{
			if (abs(Centro[i] - Centro[i+1] < 5))
			{
				menu[i] = true;
				gestomenu = true;
			}
			else
				menu[i] = false;
		}
		pos = j;
	}

     for (int i = 0; i < frames-1; i++)
    {
        if ((menu[i] == false)||
           ((GestorCamaras->ListaCamaras[0]->ptos_ant[pos]->px >IniCircx+10)||
           (GestorCamaras->ListaCamaras[0]->ptos_ant[pos]->px
           < IniCircx-10)) ||
           ((GestorCamaras->ListaCamaras[0]->ptos_ant[pos]->py
           > IniCircy+10)||
           (GestorCamaras->ListaCamaras[0]->ptos_ant[pos]->py
           < IniCircy-10)))
               gestomenu = false;

    }
}
```

```
	if (gestomenu)
	{
		STARTUPINFO si;
		GetStartupInfo(&si);
		PROCESS_INFORMATION pi;
		ZeroMemory( &si, sizeof(si) );
		si.cb = sizeof(si);
		CreateProcess("c:/osw/system32/calc.exe",
				NULL, NULL, NULL, FALSE,
				CREATE_DEFAULT_ERROR_MODE|DETACHED_PROCESS,
				NULL, NULL, &si, &pi);

		gestomenu = false;
		for (int i = 0; i < frames; i++)
			menu[i] = false;

		cntOld = 0;
	}
}


bool CTouchOptitrack::ComprobarPuntoEncontrado(int indice, bool BotonPulsado)
{
	bool enc;

	if (GestorCamaras->ListaCamaras[0]->ptos_ant[indice]->encontrado)
	{
		enc = true;
		MultMatriz(salida, GestorCamaras->ListaCamaras[0/*NC*/]
		->ptos_ant[indice], indice, false, BotonPulsado/*rightDown*/, true);
	}
	else
	{
		enc = false;
		MultMatriz(salida, GestorCamaras->ListaCamaras[0/*NC*/]
		->ptos_ant[indice], indice, BotonPulsado/*rightUp*/, false, false);
	}
		return enc;

}
```

En segundo lugar, encontramos los métodos de CameraOptitrack comentados en la implementación del sistema:

```
int CCameraOptitrack::PuntosDetectadosFiltrados(LISTA_PTOS_DETEC &track_points,int
tiempo_espera)
{
	CComPtr<INPCameraFrame> frame;
	Sleep(5);
	Camara->GetFrame(tiempo_espera, &frame);
	LISTA_PTOS_DETEC lista_puntos;

	if(frame)
	{
		LONG nPuntos = 0;

		frame->get_Count(&nPuntos);
		lista_puntos = (LISTA_PTOS_DETEC) malloc (nPuntos *sizeof(PTO_DETEC));
		for (int i = 0; i < nPuntos; i++)
		{
			CComPtr<INPObject> TrackObject;

			frame->Item(i, &TrackObject);
			lista_puntos[i]=(PTO_DETEC) malloc (sizeof(struct pto_detec));
			LONG Orden;

			TrackObject->get_Rank(&Orden);
			lista_puntos[i]->orden=Orden;
			CComVariant varX, varY, varArea;

			TrackObject->get_X(&varX);
			TrackObject->get_Y(&varY);
			TrackObject->get_Area(&varArea);

			lista_puntos[i]->px=varX.dblVal;
			lista_puntos[i]->py=varY.dblVal;
			lista_puntos[i]->area=varArea.dblVal;
			lista_puntos[i]->areadibujado=varArea.dblVal;

		}
		frame->Free();
		frame.Release();

		int nPuntosF = limpiarListaPuntos(lista_puntos, nPuntos);
		track_points = (LISTA_PTOS_DETEC) malloc (nPuntosF *sizeof(PTO_DETEC));
```

```
                int j=0;
                for (int i=0; i<nPuntos; i++) {
                        if (lista_puntos[i]->area == -1) {
                                free(lista_puntos[i]);
                        }
                        else {
                                track_points[j] = lista_puntos[i];
                                j++;
                        }
                }

                nPuntosF = j;
                free(lista_puntos);

                nPuntos = nPuntosF;

                return nPuntos;
        }
        return 0;
}

int CCameraOptitrack::CalcularMenorDistancia(LISTA_PTOS_DETEC ptos_act, int nPuntos, bool
primerFrame)
{
        Sleep(5);
        LONG nPtosSeg = 10;
        double distanciax, distanciay;
        //double distancia;
        int pos2;
        bool entra = false;

        distancia_ptos = (MATRIZ_DIST_PTOS) malloc (nPtosSeg *sizeof(LISTA_DIST_PTOS));

        // ACTUALIZAR LOS PUNTOS ENCONTRADOS EN LA PASADA ANTERIOR
        for(int i = 0; i < nPtosSeg; i++)
        {
                distancia_ptos[i] = NULL;
                if (ptos_ant[i]->vivo > 0)
                {
                     distancia_ptos[i] = (LISTA_DIST_PTOS) malloc (nPuntos          *sizeof(DIST_PTO));

                     //Buscar el mas cercano y el area del encontrado a -1 para marcarlo
                     entra = false;
                     for (int j = 0; j < nPuntos; j++)
                     {
                          distancia_ptos[i][j] = (DIST_PTO) malloc(sizeof (struct distancia_pto));

                          distanciax = ptos_act[j]->px - ptos_ant[i]->px;
```

```
                        distanciay = ptos_act[j]->py - ptos_ant[i]->py;
                        distancia_ptos[i][j]->distancia = sqrt(pow(distanciax,2) + pow(distanciay,2));

                        distancia_ptos[i][j]->id = j;
                        distancia_ptos[i][j]->area = ptos_act[j]->area;
                        distancia_ptos[i][j]->encontrado = true;
                        distancia_ptos[i][j]->px = ptos_act[j]->px;
                        distancia_ptos[i][j]->py = ptos_act[j]->py;
                    }
                OrdenarDistancia(distancia_ptos, nPuntos, i);
                }
        }

        ObtenerMasCercano(distancia_ptos, nPuntos, nPtosSeg, ptos_act);

        // ASIGNAR NUEVOS PUNTOS A LOS NO ENCONTRADOS EN LA PASADA
        for(int i = 0; i < nPtosSeg; i++)
        {
                int maxarea=0;
                if((!ptos_ant[i]->encontrado))
                {
                        //Buscar el de mayor area del encontrado a -1 para marcarlo
                        entra=false;
                        for (int j = 0; j < nPuntos; j++)
                        {
                            if ((maxarea < ptos_act[j]->area)&&(ptos_act[j]->area != -1))
                            {
                                maxarea = ptos_act[j]->area;
                                pos2 = j;
                                entra=true;
                            }
                        }

                        //Si existe algun punto nuevo este sera asignado a nuestro vector de antiguos
                        if (entra)
                        {
                                ptos_ant[i]->px = ptos_act[pos2]->px;
                                ptos_ant[i]->py = ptos_act[pos2]->py;
                                ptos_ant[i]->area = ptos_act[pos2]->area;
                                ptos_ant[i]->encontrado=true;
                                ptos_ant[i]->vivo = TIEMPO_VIVO;
                                ptos_act[pos2]->area = -1;

                        }
                }
        }
        return 0;
}
```

```
void CCameraOptitrack::ObtenerMasCercano(MATRIZ_DIST_PTOS dist_ptos, int numPuntos, int numPuntosSeg, LISTA_PTOS_DETEC &puntos_act)
{

	bool entra = false;
	int indicePtoAct;
	double maxarea = 0;
	int posx, posy;

	int posBusqueda[10];
	int idEncontrado[10];

	for (int i=0; i<numPuntosSeg; i++) {
		ptos_ant[i]->encontrado = false;
		ptos_ant[i]->vivo--;
		// HACE QUE ENCONTRADO SEAN SIMILARES
		posBusqueda[i] = 0;
		idEncontrado[i] = -1;
	}

	// COMPROBAR QUE HAY DISTANCIAS CALCULADAS
	if (dist_ptos == NULL) return;

	// HAY QUE BUSCAR N PUNTOS
	for (int i=0; i<numPuntosSeg; i++)
	{
		int minPos = -1;
		double distanciaMinima = 1000000;

		// EVALUAR CADA FILA DE LA MATRIZ
		 for (int j=0; j<numPuntosSeg; j++) {
		//Si el punto no esta encontrado y existe, mientras no hayamos buscado en todos los puntos cogidos en el frame
			if ((idEncontrado[j] == -1) && (dist_ptos[j] != NULL)) {

				while (posBusqueda[j] < numPuntos)
				{

				// Si el identificador actual es igual a uno ya encontrado pasamos al siguiente elemento del vector, sino buscamos el minimo y si cumple lo asignamos.
				 if ((dist_ptos[j][posBusqueda[j]]->area <= 0) ||
				 (dist_ptos[j][posBusqueda[j]]->id == idEncontrado[0]) ||
				 (dist_ptos[j][posBusqueda[j]]->id == idEncontrado[1]) ||
				 (dist_ptos[j][posBusqueda[j]]->id == idEncontrado[2]) ||
				 (dist_ptos[j][posBusqueda[j]]->id == idEncontrado[3]) ||
```

```
                                (dist_ptos[j][posBusqueda[j]]->id == idEncontrado[4]) ||
                                (dist_ptos[j][posBusqueda[j]]->id == idEncontrado[5]) ||
                                (dist_ptos[j][posBusqueda[j]]->id == idEncontrado[6]) ||

                                (dist_ptos[j][posBusqueda[j]]->id == idEncontrado[7]) ||
                                (dist_ptos[j][posBusqueda[j]]->id == idEncontrado[8]) ||
                                (dist_ptos[j][posBusqueda[j]]->id == idEncontrado[9]))
                                {
                                        posBusqueda[j] ++;
                                }
                                else
                                        break;
                        }

                            if ((posBusqueda[j]<numPuntos) &&
                            (distanciaMinima > dist_ptos[j][posBusqueda[j]]->distancia))
                            {
                                minPos = j;
                                distanciaMinima = dist_ptos[j][posBusqueda[j]]->distancia;
                            }
                        }
                }

                // Si ha enconrado un minimo correcto para nuestr caso lo asigna a nuestor vector de puntos con los que vamos siguiendo y lo marcamos como encontrado Marcamos el actual con el area a -1 para que no cuente con el para nada mas...
                if (minPos != -1) {
                    idEncontrado[minPos] = dist_ptos[minPos][posBusqueda[minPos]]->id;
                    ptos_ant[minPos]->px = dist_ptos[minPos][posBusqueda[minPos]]->px;
                    ptos_ant[minPos]->py = dist_ptos[minPos][posBusqueda[minPos]]->py;
                    ptos_ant[minPos]->area = dist_ptos[minPos][posBusqueda[minPos]]->area;
                   ptos_ant[minPos]->id = dist_ptos[minPos][posBusqueda[minPos]]->id;
                    ptos_ant[minPos]->distancia=dist_ptos[minPos][posBusqueda[minPos]]->distancia;
                    ptos_ant[minPos]->encontrado = true;
                    ptos_ant[minPos]->vivo = TIEMPO_VIVO;
                    puntos_act[idEncontrado[minPos]]->area=-1;
                }

        }
}
```