Sistema de vídeo distribuido para Centros Infantiles

FASE 2 - PROTOTIPO

Índice de contenido:

1. Introducción

Antes de pasar a comentar la información acerca de la realización del prototipo se ha añadido un pequeño resumen con el que poder retomar o recordar la idea que el equipo Prometheus propuso realizar.

Se sugirió diseñar y desarrollar un servicio para los Centros Infantiles con el fin de que los familiares de los niños puedan ver en tiempo real a los mismos, a través de una señal de vídeo. El servicio determinará la cámara más adecuada para visualizarlos, buscando la habitación en la que se encuentra el niño, así como la cámara más adecuada para obtener imagen del mismo, y enviando la emisión a sus tutores legales -o a quien éstos autoricen- gracias a la tecnología *RTI Connext DDS*.

También cabe destacar un pequeño cambio que hemos realizado en cuanto a los dominios utilizados (con el fin de utilizar un menor ancho de banda, evitando la saturación de la red). Se ha pasado de tener dos dominios a tres: dominio de sensores (datos para la elección de la cámara), dominio de conexión de padres-hijos (en el que cada tópico representa a un niño), dominio de cámaras (en el que cada tópico es una cámara). Anteriormente se utilizaban 2 dominios (uno para la comunicación entre los computadores de la red interna, que decidían qué cámara era la más adecuada para un niño y otro en el que existían tantas emisiones de vídeo como niños había introducidos en el sistema). Así, aunque parezca que al aumentar el número de dominios se consume un mayor ancho de banda, se reducen el número de emisiones de vídeo (que son las que más cargan la red).

Para exponer el prototipo de nuestro producto, dividiremos dicha fase en varias tareas:

Adquisición de los materiales necesarios: En primer lugar, requerimos obtener algunos materiales y componentes que usamos en nuestro prototipo:

- Raspberry PI
- Bluetooth HC-06
- Bluetooth v4.0 CSR4.0 USB Dongle
- Hub USB
- Cable alargador USB
- Cámaras:
  - *Eye Toy*
  - *Logitech QuickCam Express*
  - *NGS ZC0305*

Realización de los programas necesarios para conseguir que funcione el sistema:

- Programa para obtener RSSI de los sensores

- Envío de los valores RSSI de sensores a la Raspberry Pi principal de la habitación (utilizando *RTI Connext DDS*)
- Programa para cálculo de la distancia entre las cámaras y el niño y su posición en la habitación
- Programa para determinar qué cámara es la más adecuada para obtener el vídeo de cada niño
- Notificar a los padre qué cámara es la más adecuada (utilizando *RTI Connext DDS*)
- Programa para transmisión de vídeo (utilizando *RTI Connext DDS*)
- Programa para la recepción del vídeo (utilizando *RTI Connext DDS*)

Con estas dos tareas realizadas en paralelo, hemos efectuado un conjunto de numerosas pruebas con las que hemos ido determinando y comprobando el funcionamiento de los diferentes pasos del prototipo, de modo que hemos podido corregir y mejorar nuestros programas para poder implementar el sistema de la forma más óptima posible.
Antes de pasar a explicar las partes del sistema, pasaremos a explicar el sistema en general en cuanto a Dominios de DDS se refiere.

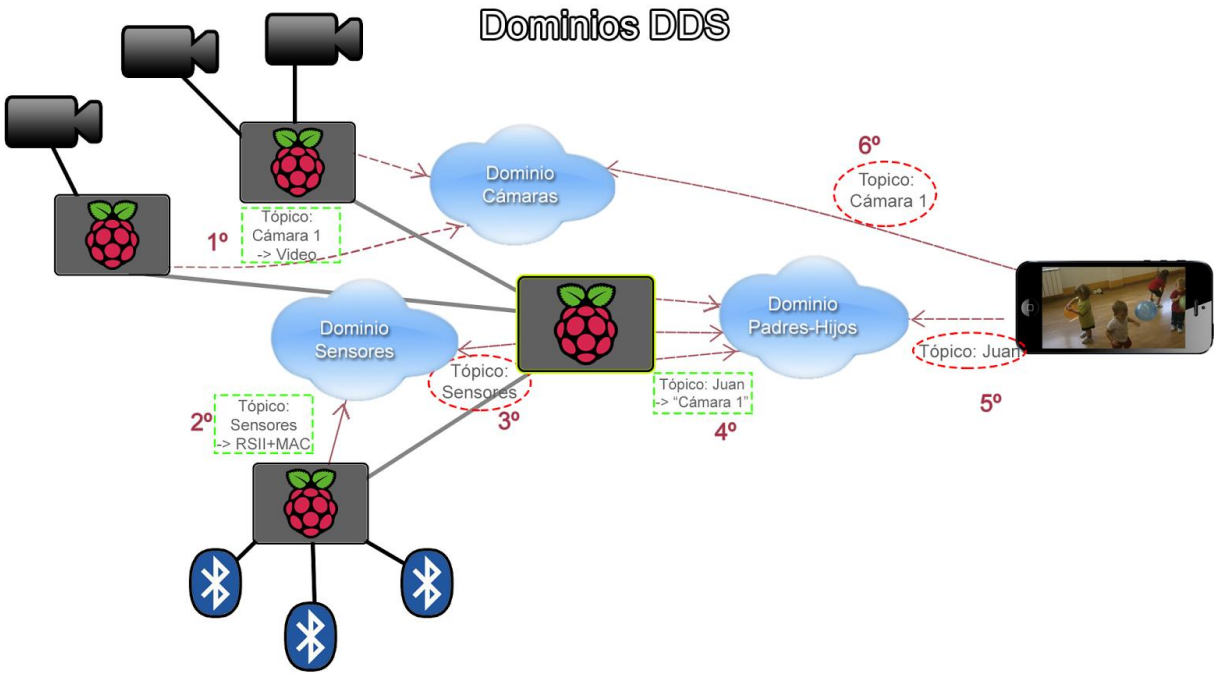

Figura 0. Dominios DDS del sistema

En esta figura representamos el conjunto de todas las comunicaciones establecidas entre cada una de las partes de nuestro servicio. Las líneas continuas indican las uniones físicas, mientras que las líneas discontinuas esquematizan las uniones de los suscriptores y publicadores a los dominios. Por otra parte las elipses discontinuas rojas hacen referencia a los elementos

correspondientes a la suscripción, mientras que los rectángulos discontinuos verdes se muestran los elementos pertenecientes a la publicación.

El proceso de selección y obtención de la imagen del niño queda de la siguiente forma:
Las *Raspberry Pi*, toman de los sensores la intensidad de señal recibida por las pulseras y las envían (mediante *RTI Connext DDS* en el dominio “Sensores”) a una *Raspberry Pi* principal. Allí, se procesan los datos y se determina qué cámara es la más adecuada para ver al niño. A continuación, se le notificará al padre (indicándole en el dominio “Padres-Hijos” a qué tópico de cámara debe suscribirse) y este deberá tomar la imagen de las *Raspberry Pi* que tienen conectadas las cámaras (se vuelve a utilizar el protocolo *RTI Connext*, siendo cada cámara un tópico en el dominio “Cámaras”).

En los siguientes apartados se han analizado cada uno de los programas que hemos diseñado para la realización del sistema, junto con la explicación de las diferentes pruebas que hemos hecho para poder demostrar la eficacia de nuestro código. También se han añadido algunos comentarios sobre los diferentes problemas que nos han aparecido así como la solución proporcionada.

2. Selección de cámara y posicionamiento

En este apartado vamos a describir los diferentes procedimientos para determinar la cámara más adecuada para proporcionar el vídeo del niño. Para ello, primero hemos llevado a cabo un estudio de la potencia recibida en los sensores *Bluetooth,* con el objetivo de analizar de forma empírica cómo se comportan, y obtener así con mayor precisión una fórmula que permita convertir la potencia de la señal recibida del *Locaviewer*[1] a la distancia que hay entre el sensor y el niño. Una vez realizado dicho estudio, pasamos a obtener los datos de *RSSI*[2] e introducirlos en un programa implementado en *Octave* con el que obtenemos la posición del niño dentro de la habitación. Finalmente, mediante un algoritmo, deducimos cuál es la cámara más adecuada para proporcionar imagen del niño en dicha posición calculada.

2.1. Estudio de potencia de los sensores *Bluetooth*

En primer lugar nos centraremos en la parte de asignación de la cámara a cada uno de los niños. Antes de ello, hemos analizado cómo varía la potencia del bluetooth en función de la distancia y del ángulo respecto al emisor.

Se ha utilizado el *Bluetooth v4.0 CSR4.0 USB Dongle* para la toma de medidas reales de la potencia recibida (mediante el parámetro RSSI enviado por el módulo) en función del ángulo (variando entre 0º y 180º) y de la distancia (variando entre 0.5 m y 5 m), obteniendo así la gráfica que se muestra en la que cada línea representa cada uno de los diferentes ángulos:

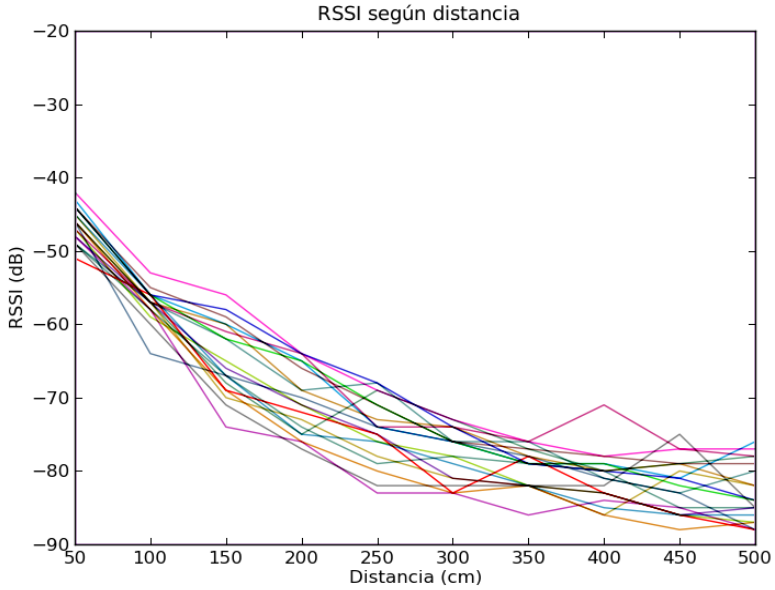

Figura 1. Potencia en función de la distancia para todos los ángulos (de 0º a 180º).

[1] Pulsera que lleva el niño con módulo *Bluetooth*.
[2] *Received Signal Strength Indication,* valor de potencia recibida devuelto por el sensor *Bluetooth*.

Se ha construido un mapa de radiación con las medidas experimentales de los sensores *Bluetooth*, para ver visualmente la potencia recibida por el módulo *Bluetooth* (pulsera):

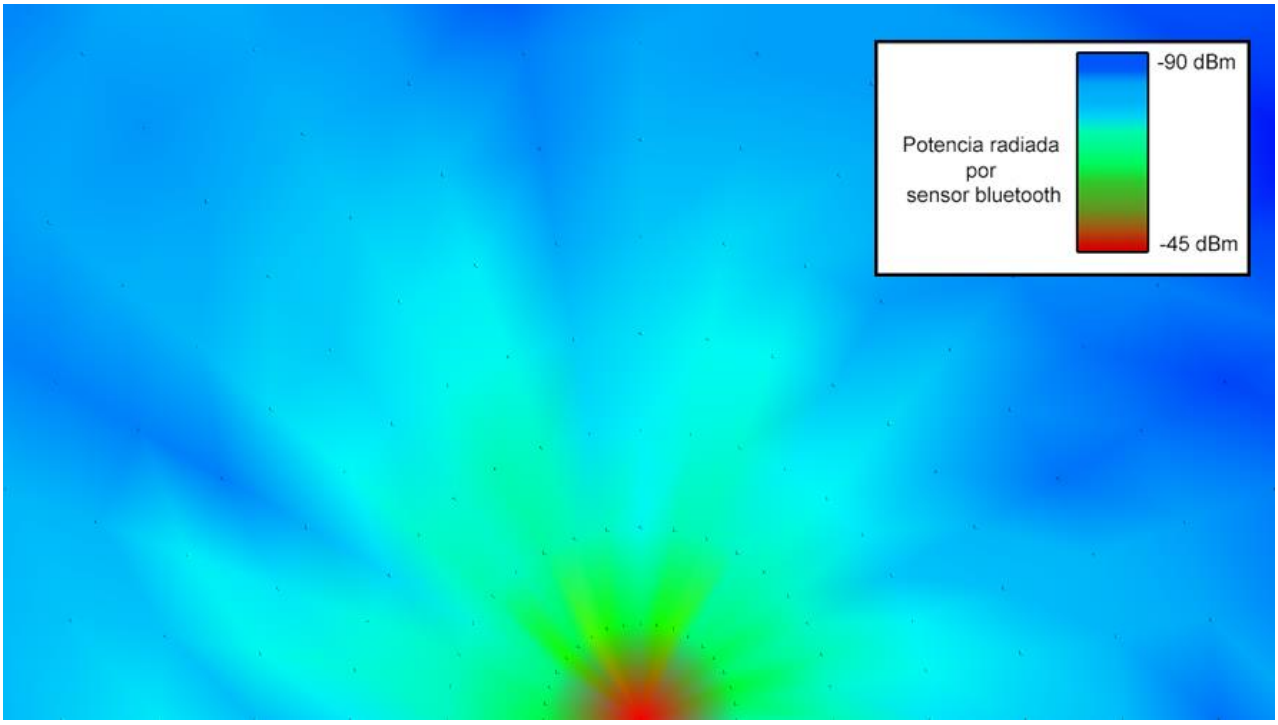

Figura 2. Mapa de radiación del sensor Bluetooth

En este mapa de radiación se representa la variación de la potencia emitida por el sensor *Bluetooth* en función de las coordenadas “x” e “y” de la habitación que se corresponde con una variación del módulo de la distancia y del ángulo con respecto al centro del sensor bluetooth.

Fijándonos en la figura 1, hemos observado que, para los resultados obtenidos al medir la potencia con ángulos de 30º y 120º, se obtienen los valores límite entre los que varía la potencia de la señal recibida, es decir, que para el rango de ángulos entre 0 y 180º, las variaciones de las potencias recibidas se encuentran en dicho rango.

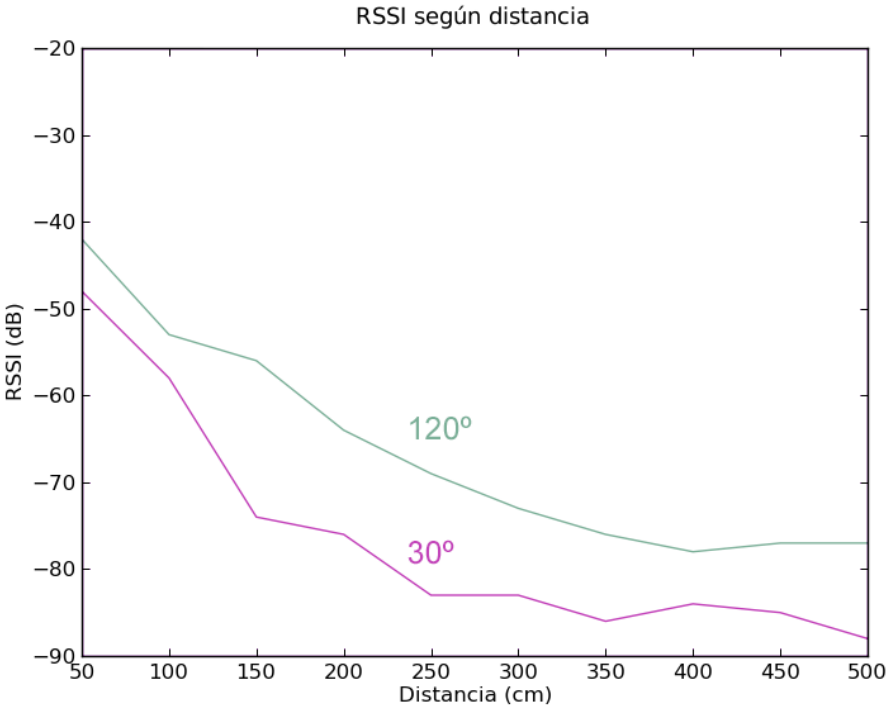

Figura 3. Ángulos límite entre los que varía la potencia recibida (120º y 30º)

Una vez obtenidos los límites, hemos observado que la potencia recibida para un ángulo fijo de 80º presenta un valor medio entre la representación de las líneas de 30º y 120º. Como consecuencia, hemos decidido emplear los resultados obtenidos para dicho ángulo como referencia para obtener la distancia (se estima que si elegimos los límites, el error será menor que si cogemos otro ángulo como referencia).

Usando *Matlab* y los datos de potencia para el ángulo de 80º, hemos obtenido una ecuación polinómica que relaciona la potencia con la distancia:

$$r = -0.00680102923817849 \cdot \mathrm{P}^3 - .04905123190747 \cdot \mathrm{P}^2 - 59.2087843354658 \cdot \mathrm{P} - 1106.35595941215 \quad (1)$$

Donde 'P' es el valor de RSSI recibido y 'r' la distancia al *Bluetooth*.

A continuación se presenta una gráfica con la potencia recibida en función de la distancia para un ángulo de 80º, la aproximación polinómica obtenida y la potencia teórica:

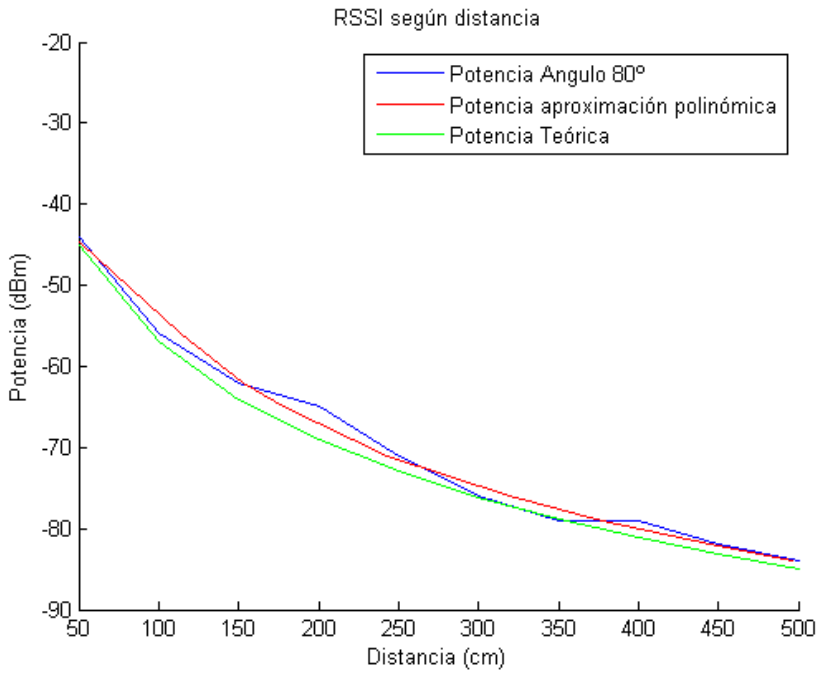

Figura 4.Potencia para ángulo de 80º en función de la distancia, polinómica y teórica

Observamos en la figura 4 que la aproximación polinómica obtenida es bastante cercana a la gráfica teórica que relaciona la potencia en función de la distancia. Por lo que hemos considerado la ecuación polinómica (ecuación 1) como correcta para calcular la distancia del sensor *Bluetooth* a la pulsera *Locaviewer*.

2.2. Obtención y envío de los datos RSSI de los sensores

La obtención de los valores de RSSI se ha realizado mediante un *script* que hemos desarrollado en Python. Dicho *script* es llamado desde el programa principal de Java, el cual le pasa como argumentos la MAC del sensor *Bluetooth* del que se quieren leer los valores RSSI que le llegan y el puerto para enviar los datos a la aplicación de Java. De esta forma, con el *script* se abre un *socket* a través del cual envía paquetes que contienen la MAC de la pulsera leída y el RSSI correspondiente a dicha pulsera.

Para obtener el RSSI se envía un paquete "*inquiry*" con dutycycle[3] de 6 (periodos de 1.28 segundos) con el sensor *Bluetooth* que está conectado a la *Raspberry Pi* y espera durante un tiempo respuesta. Cuando recibe respuesta comprueba el evento recibido y si es del tipo *EVT_INQUIRY_RESULT_WITH_RSSI* obtiene la MAC de la pulsera y el RSSI.

[3] Valor óptimo de Duty Cycle para la obtención de respuestas con RSSI

Una vez hecho esto, el valor de RSSI junto con la MAC de la pulsera, son enviados para analizarlos en el programa principal.

En el programa principal realizado mediante Java, tendremos dos procesos corriendo: el "sensor" y el "servidor". El sensor se ejecutará en la *Raspberry Pi* que tiene conectado al menos un sensor *Bluetooth*, mientras que el servidor hará lo propio en otra *Raspberry Pi* que recibirá los datos de los sensores para, más tarde, procesarlos.

- Sensor: recibirá los datos del *script* anteriormente mencionado a través de un *socket*. El *socket* estará asociado a un sensor y, como consecuencia, a la MAC y posición del mismo. Empaquetará los datos en una clase diseñada para las comunicaciones y enviará los datos al sensor mediante *RTI Connext DDS*.
- Servidor: recibirá los datos de las *Raspberry Pi* que tienen conectados los sensores. Después, procesará los datos como veremos en las siguientes secciones.

2.3. Cálculo de la posición del niño mediante triangulación

En este apartado vamos a obtener la posición del niño en la habitación en la que se encuentre conociendo las distancias obtenidas entre el mismo y los diferentes sensores bluetooth.

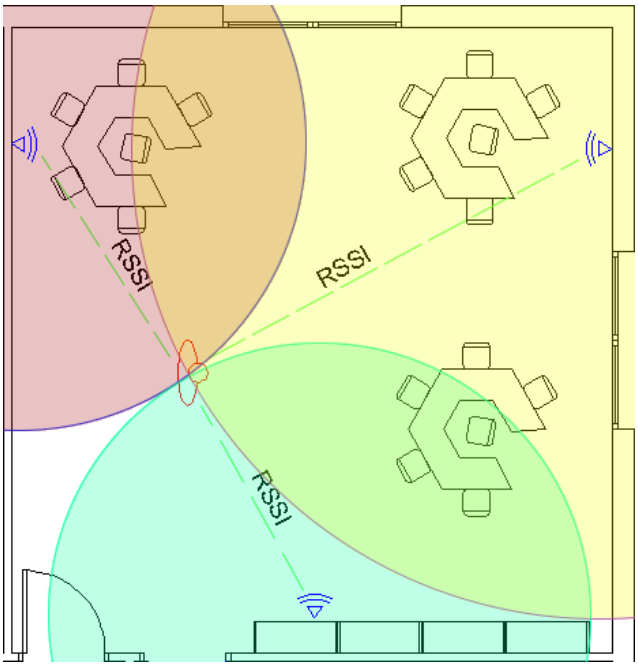

Figura 5. Triangulación

Como vemos en la figura 5, para calcular la posición del individuo dentro de la habitación, es necesario saber la posición de los sensores *Bluetooth* y la distancia de los sensores a las

pulseras *Locaviewer*. De esta forma, una vez conocida la potencia recibida de la señal *Bluetooth* en la pulsera, podemos calcular la distancia que existe entre el niño y el sensor *Bluetooth* con la fórmula obtenida en el estudio de potencia realizado (ecuación 1).

Con la distancia entre la pulsera y cada uno de los sensores, procedemos a calcular la posición del niño en el plano de la habitación mediante triangulación.

Para ello, dado un número N de sensores *Bluetooth* de los que conocemos la distancia con la pulsera, comprobamos primero si los valores de las distancias son correctos, es decir, que las distancias obtenidas proporcionan un valor de la posición del niño en el interior de la habitación. Una vez realizada esta comprobación, pasamos a aplicar la fórmula mediante la cual obtenemos la posición de la pulsera en el plano de la habitación:

$$m = (H^T \cdot H)^{-1} \cdot H^T \cdot C$$

(2)

Donde “m” es un vector de dos elementos que indica la posición del niño:

$$m = \begin{bmatrix} x \\ y \end{bmatrix}$$

“H” representa una matriz de dos columnas (una para la coordenada “x” y otra para la coordenada “y”) en las que compara la posición de cada uno de los *Bluetooth* con el resto:

$$H = \begin{bmatrix} h_{x(2,1)} & h_{y(2,1)} \\ \cdots & \cdots \\ h_{x(N,1)} & h_{y(N,1)} \\ \cdots & \cdots \\ h_{x(N,N-1)} & h_{y(N,N-1)} \end{bmatrix}$$

Donde $h_{x(a,b)}$ es dos veces la diferencia de las coordenadas X de la posición entre el *Bluetooth* ‘a’ y ‘b’ y $h_{y(a,b)}$es dos veces la diferencia de las coordenadas Y de la posición entre el *Bluetooth* ‘a’ y ‘b’.

Y C es:

$$C = \begin{bmatrix} C_{2,1} \\ \cdots \\ C_{N,1} \\ \cdots \\ C_{N,N-1} \end{bmatrix}$$

Donde $C_{i,j}$ se obtienen según la siguiente expresión:

$$C_{i,j} = \tilde{r}_i^2 - \tilde{r}_j^2 + x_j^2 - x_i^2 + y_j^2 - y_i^2$$

Donde 'x' e 'y' indican la posición del *Bluetooth* y r̂ es la distancia obtenida entre el sensor *Bluetooth* y la pulsera. De esta forma, tenemos localizada la posición del niño en el plano de la habitación.

2.4. Asignación del tópico niño a un tópico cámara

Una vez conocemos la posición del niño y la de las cámaras, estamos preparados para detectar qué cámara es la más adecuada para proporcionar el vídeo del individuo que estamos analizando.

Para ello, en primer lugar hemos realizado simulaciones prácticas para medir los ángulos de visión de cada una de las cámaras en las que hemos obtenido que la peor cámara de las que usamos tenía un ángulo de visión de 42º, por lo que hemos usado dicho valor como el ángulo de visión de las cámaras qué vamos a usar.

En la siguiente imagen, se puede ver la ejecución de un programa Java que se ha desarrollado como simulador, donde podemos ver en tiempo real la posición de la pulsera (punto rojo), de las cámaras (puntos azules), de los sensores Bluetooth (puntos verdes), la zona de visión de las cámaras, la potencia (en un círculo) leída, así como la cámara elegida para grabar al niño:

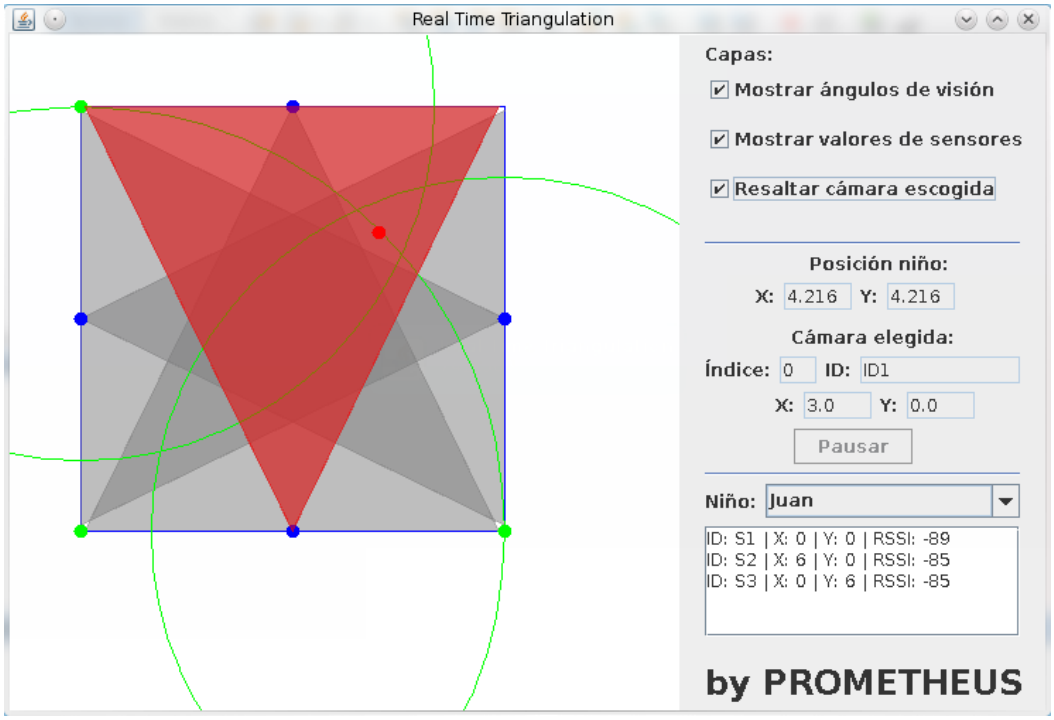

Figura 6. Programa para comprobar el correcto funcionamiento de la selección de cámara

Para detectar qué cámara ofrece mejor imagen del niño, comprobamos que la posición obtenida del mismo está dentro de los límites de la habitación y en caso afirmativo, pasamos a determinar en qué cámaras se obtiene imagen del individuo. Para ello, colocando la habitación con una esquina en el eje de coordenadas y dos de las paredes en los ejes “x” e “y” tal y como se representa en la siguiente figura, detectamos qué cámaras están localizadas en las paredes verticales y cuales en las horizontales.

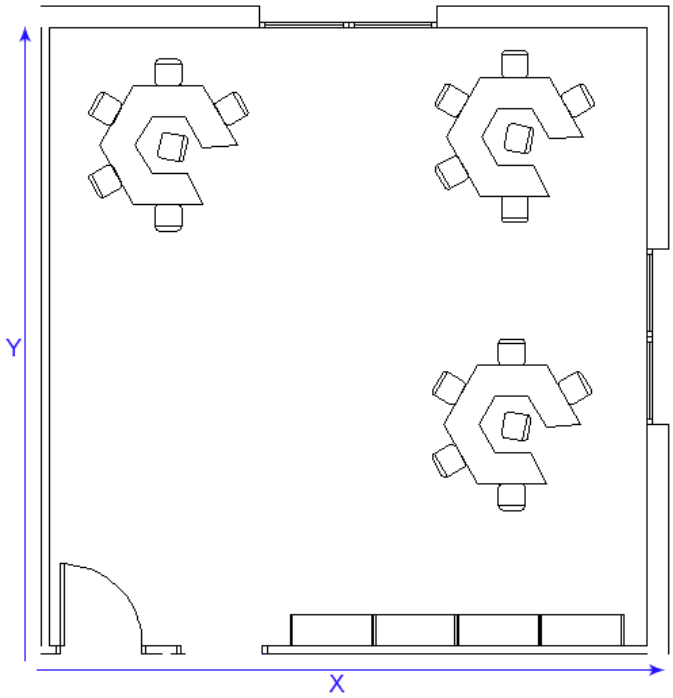

Figura 7.Sistema de coordenadas de la habitación.

Una vez hecho esto, comprobamos para las cámaras colocadas en las paredes horizontales qué rango de valores de “x”, para el valor de “y” en el que está localizado el niño, permite visualizar la cámara y así comparar si la posición “x” del individuo está dentro de ese rango de valores y por lo tanto, está dentro del ángulo de visión de la cámara. En el caso de las cámaras situadas en las paredes verticales tenemos que ver el rango de valores de “y” entre los cuales obtiene imagen la cámara para la posición “x” del individuo y comprobar si la posición “x” del niño está dentro de esos límites.

Cuando ya conocemos qué cámaras obtienen imagen del niño, calculamos la distancia de la pulsera a dichas cámaras y nos quedamos con la cámara que presente menor distancia. En caso de que ninguna cámara obtenga imagen del individuo, el programa devolverá un código de error.

3. Transmisión y recepción de vídeo

La obtención de los datos codificados desde una cámara es uno de los problemas más complicados a tratar. En primer lugar, para hacer funcionar las cámaras en la *Raspberry Pi* se necesita tener una versión del sistema operativo *Raspbian* antigua, pues los drivers modernos no funcionan con los modelos de cámaras utilizados.

Una vez conseguido vídeo desde una *Raspberry Pi*, el siguiente problema a tratar es la obtención de los datos de estos codificados en tiempo de ejecución (para la reducción de tamaño y posibilidad de envío por Internet). Las librerías que posibilitan la obtención de imagen de cámara en Java, no permiten esta operación, sólamente guardar los datos finales en un fichero o realizar *streaming* con ellos. Esto es un problema pues se necesitan tener en un vector de *bytes* para su envío mediante DDS (descartado modo *streaming* por HTTP o RTP) y leer y escribir un fichero de forma simultánea es un proceso lento y con muchos errores a solucionar. Ante este problema, hemos diseñado la solución que se puede encontrar en el siguiente diagrama.

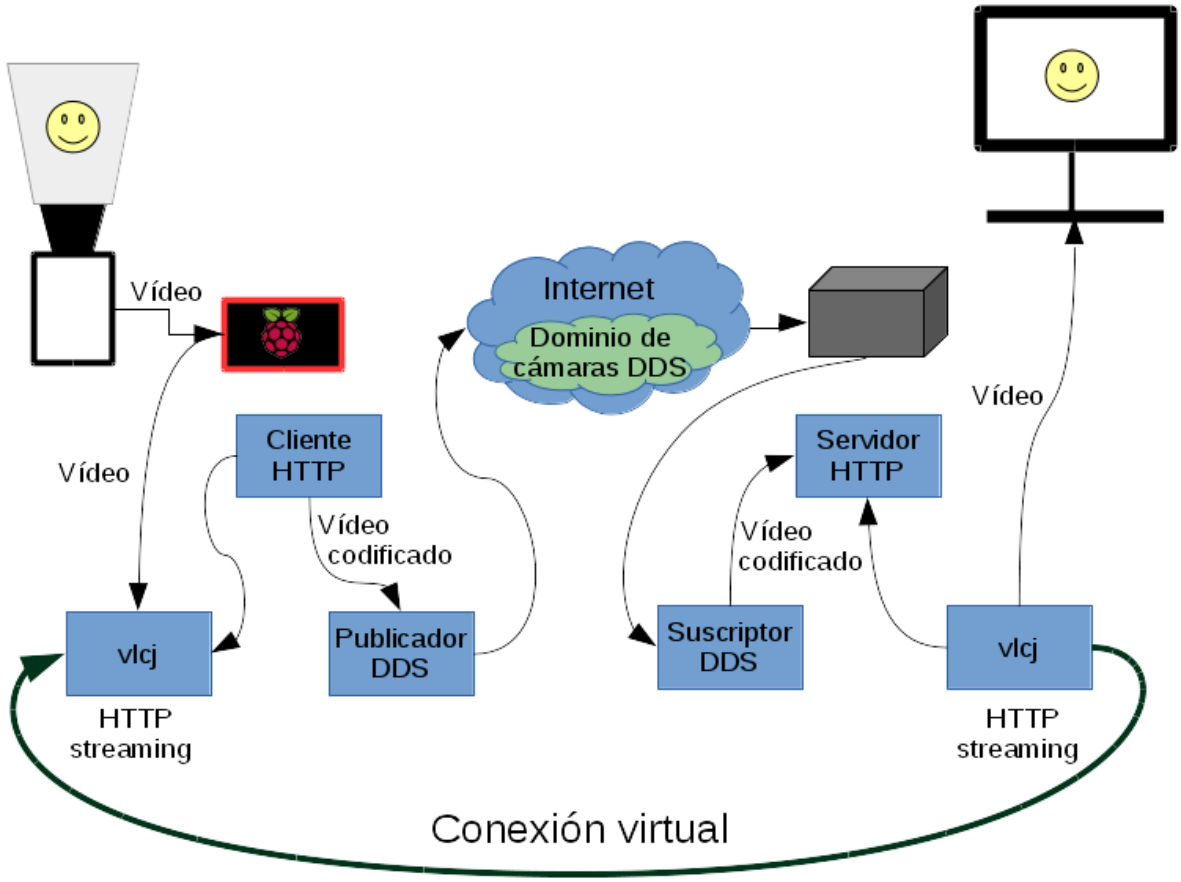

Figura 8. Esquema de las interconexiones para la transmisión y recepción de video

El objetivo de la solución es realizar *streaming* con la librería *vlcj* usando *DDS*. Para ello, mediante esta librería se obtendrá el vídeo de la cámara y se le ordenará que los codifique y realice un *streaming* por HTTP con ellos. A continuación, en ese mismo dispositivo que será una *Raspberry Pi*, se creará un cliente HTTP que se conecte al *streaming*. De esta forma, este cliente creará a su vez un publicador de DDS y todos los datos que reciba del *streaming* los podrá redirigir como publicador.

En el lado de la aplicación del cliente se realizará el proceso contrario. Se creará un suscriptor de DDS y un servidor HTTP. Los datos que recibe de DDS los pasará al servidor de HTTP. De esta forma se le indicará a la librería *vlcj* que reproduzca un *streaming* HTTP conectándose al servidor HTTP que se ejecuta en esa misma máquina, que a su vez le enviará los datos que recibe por DDS. El objetivo final es realizar un túnel de los datos codificados mediante DDS de forma que se está simulando una conexión directa de *streaming* entre ambas librerías *vlcj*.

4. Ampliaciones y actualizaciones para la fase final

Una vez explicados los objetivos realizados en esta fase, pasamos a explicar los avances que esperamos desarrollar en la última fase del Desafío Tecnológico:

- En primer lugar, en cuanto a la transmisión de datos de los sensores *Bluetooth* a la *Raspberry Pi* establecida como principal, se añadirá una actualización en la que, si dicha *Raspberry Pi* deja de estar operativa, se establezca una de las otras como principal, de forma que siempre haya una realizando los cálculos de la principal.

- Respecto al cálculo de la selección de cámara y posicionamiento, desarrollaremos un software más sotisficado que nos permita leer los datos obtenidos de los sensores *Bluetooth* usando Java, para no necesitar hacer uso de un *socket* para transmitirlos desde el *script* en Python. También utilizaremos una configuración más eficiente del *Bluetooth* con la que podamos obtener la información necesaria para identificar la pulsera y la potencia recibida en la misma, utilizando un menor consumo de energía. Además realizaremos un programa para paliar lo máximo posible los datos erróneos que puedan interferir en la triangulación y así obtener una localización más precisa del niño dentro de la habitación.

- En cuanto a la recepción y transmisión de vídeo, implementaremos una aplicación de cliente para escritorio y dispositivos móviles con el fin de facilitar la utilización de nuestro producto. Adicionalmente, en este ámbito también se introducirá la posibilidad de que los usuarios de nuestro servicio puedan seleccionar otra cámara (entre las que están enviando video del niño) de la que recibir la imagen de forma que dicha selección, predomine sobre la cámara indicada como la adecuada para transmitir video en relación con el niño seleccionado.

  Sería interesante la posibilidad de conocer qué cámaras hay capturando, en tiempo de ejecución, con idea de mejorar la escalabilidad del sistema. Para ello se procederá a cambiar levemente la implementación del programa de la *Raspberry Pi* principal, de forma que preguntase (en el dominio en el que se encuentran emitiendo las cámaras) de qué tópicos disponemos.

- Utilizaremos también mecanismos para facilitar el mantenimiento del sistema, de forma que, en caso que se publiquen versiones nuevas de los distintos programas que utilizamos, se actualicen de forma automática o notifiquen de la posibilidad de actualizar.

- Con respecto a la conexión de los suscriptores (padres) con los publicadores, implementaremos en un servidor el plugin *RTI WAN Serve*r con el que realizaremos la comunicación fuera del área local.

- Se buscará una alternativa a la solución planteada en el apartado 3 de codificación de vídeo. El método del túnel por DDS aunque funciona presenta una serie de inconvenientes como un retardo adicional en la comunicación.

- Se estudiará sustituir *Raspberry Pi* por mini PCs con más potencia, de menor tamaño y precio similar, como el *MK802 II de Rickomagic* u otro similar.

5. Diseño y construcción de la pulsera *Locaviewer*

En este apartado vamos a comentar el modelo de la pulsera que hemos implementado y que permitirá devolver el dato del RSSI recibido para poder realizar los cálculos con los que detectamos la cámara adecuada para proporcionar el vídeo del niño que tiene dicha pulsera.

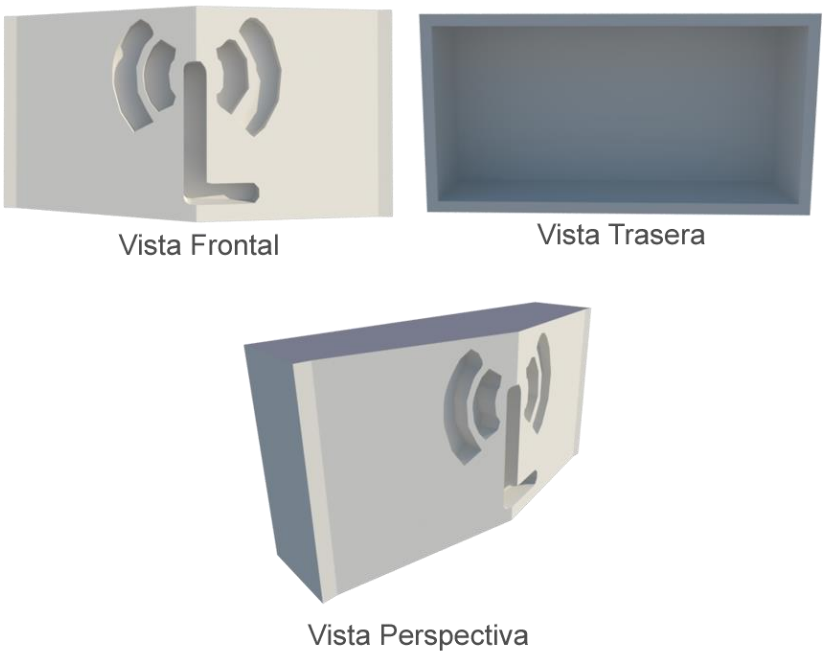

Figura 9. Diseño en 3D de la pulsera Locaviewer

Los componentes de la pulsera son:

- Carcasa de plástico fabricada mediante nuestra impresora 3D
- 2 pilas de botón de 3V
- 1 Modulo Bluetooth HC-06
- Correa adaptable con velcro

La carcasa se ha diseñado en un programa de CAD de forma adecuada al tamaño del HC-06 y al espacio ocupado por las pilas de botón. En la figura 9 se muestra el diseño final del prototipo.

Para obtener este diseño, se han ido desarrollando diferentes versiones con el fin de obtener las mejores formas y medidas de nuestra pulsera. En la siguiente figura se pueden ver los prototipos de carcasa desechados:

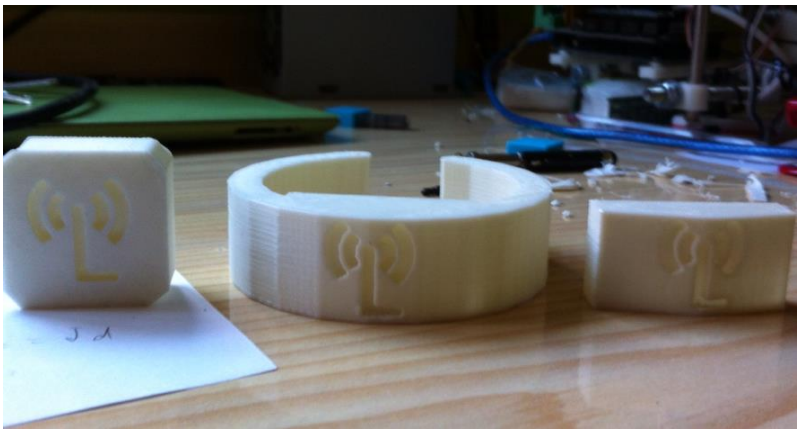

Figura 10. Prototipos de pulsera desechados

Para la fase final es probable que se rediseñe la pulsera con el fin de que se proporcionen nuevas formas de adjuntar a los niños el módulo *Bluetooth* como un collar, una chapa, etc y que sea lo más pequeña y cómoda posible para los niños.

Tras ello se ha fabricado la pulsera mediante nuestra impresora 3D con plástico ABS blanco con una resolución de capa de 0.2mm.

En la siguiente figura se observa el prototipo final de nuestra pulsera *Locaviewer*:

Figura 11. Modelo de la pulsera Locaviewer impresa en 3D

Y en el interior de este modelo se han insertado las pilas de botón junto al módulo *Bluetooth* comentado anteriormente, tal y como se muestra en la figura 12:

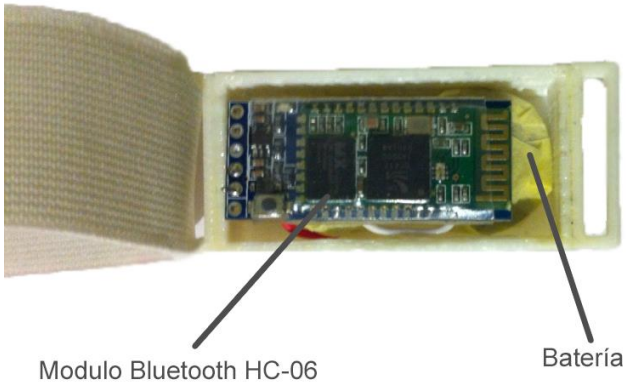

Figura 12. Interior de la pulsera Locaviewer

6. Conclusión

En esta fase, hemos realizado un sistema con el que, a partir de los datos de potencia recibidos por un mínimo de 3 sensores *Bluetooth* y las posiciones de los mismos y de las cámaras, podemos obtener cuál de estas últimas es la más adecuada para grabar a cada uno de los niños detectados (gracias a una pulsera que nosotros mismos hemos diseñado, la cual cuenta con un dispositivo *Bluetooth* integrado). Por otra parte, también hemos realizado la obtención del vídeo codificado de las cámaras conectadas a una *Raspberry Pi* y la transmisión en *streaming* por DDS.

Finalmente hemos comentado los avances y mejoras que vamos a realizar para poder obtener el producto que queremos ofrecer a nuestros clientes con la mayor fiabilidad y calidad posibles.

7. Referencias

- RTI Connext DDS - User's Manual version 5
- DDS Protocol - Specification
- “Bluetooth Indoor Positioning using RSSI and Least Square Estimation”. Autores: Yapeng Wang, Shusheng Shi, Xu Yang1 y Athen Ma
- Tesis de Anne Fransses: *Impact of multiple inquirers on the Bluetooth discovery process*
- Specification of the Bluetooth System V4
- “Bluetooth : connect without cables”. Autores: Jennifer Bray y Charles F. Sturman
- Bluetooth revealed : the insider's guide to an open specification for global wireless communications. Autores: Brent A. Miller y Chatschik Bisdikian.