PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Escuela de Ingeniería Eléctrica
PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Matías Cuevas Carreño

Diseño e Implementación de un Sistema de Localización en Robots Móviles mediante Sistemas de Radiofrecuencia

Informe Proyecto de Título de Ingeniero Civil Electrónico

Escuela de Ingeniería Eléctrica
Facultad de Ingeniería

Valparaíso, 05 de marzo de 2019

Diseño e implementación de un sistema de localización en robots móviles mediante sistemas de radiofrecuencia

Matías Manuel Ignacio Cuevas Carreño

Informe Final para optar al título de Ingeniero Civil Electrónico,

aprobada por la comisión de la

Escuela de Ingeniería Eléctrica de la

Facultad de Ingeniería de la

Pontificia Universidad Católica de Valparaíso

conformada por

Sr. Francisco Pizarro Torres

Profesor Guía

Sr. Gonzalo Farías Castro

Segundo Revisor

Sr. Sebastián Fingerhuth Massmann

Director de Escuela

Valparaíso, 05 de marzo de 2019

Dedicada a mi familia que a pesar de todos los momentos difíciles vividos confió en mí y me apoyo para poder finalizar de la mejor manera mis años universitarios, también a mis compañeros con los que superamos y compartimos metas, alegrías, frustraciones y tristezas.

Resumen

El informe que a continuación se presenta, tuvo como misión primaria la tentativa de explicar la problemática a la que busca dar solución, a saber, el asunto de la localización. En lo siguiente, se procedió analítica y dogmáticamente desde la perspectiva de aquellas áreas de la electrónica que permitieran dar una respuesta al problema, centrándose siempre en la opción más rápida y eficaz. Consideración especial recibió en este estudio el ruido que se produjo en la propagación de la señal y futura adquisición para su utilización en el sistema de estimación, planteando diversas técnicas para sopesarlo. Enseguida, se presentaron y ensayaron diversas tecnologías, con fundamento en la radiofrecuencia, que se han utilizado para la localización de dispositivos, junto con los diferentes resultados conseguidos por distintos autores que han centrado su investigación en la materia.

Proyectado el estado del arte, específicamente en lo referente a la localización por radiofrecuencia, parte del trabajo centró su foco en el estudio de la placa Raspberry Pi Zero W como reemplazo al robot Khepera IV. Se estudió además el sistema operativo de Linux y Python, logrando diseñar un prototipo de programa en el entorno de programación que permite caracterizar el ambiente, para luego obtener la posición aproximada del dispositivo; en seguida fue llevado a la placa Raspberry que fue instalada sobre el robot. Pese a esto, al realizar las pruebas, al girar la placa en su eje el valor de medición presento gran varianza, por lo que se modificó la placa para incluir una antena externa omnidireccional que mejoró el desempeño de la medición, enmendando el error a alrededor de 10 centímetros sin realizar aun el añadido de un algoritmo robusto para el seguimiento.

Surgió la necesidad de mejorar el tiempo de muestreo, por lo que se planteó el filtraje de la señal asumiendo un ruido gaussiano. Como respuesta al programa se tuvo un buen desempeño en términos de velocidades de muestreo, además al reducir la variación aleatoria de la muestra también se redujo el error en cierta proporción. Posteriormente con la presencia del erro en la distancia se analizó la opción de añadir el método de mínimos cuadrados para conseguir la estimación de la posición, donde se apreció que, al añadir un número considerable de referencias, la dispersión total en la estimación de la posición mejora considerablemente.

Palabras claves: Radiofrecuencia, localización, Robot móvil, Bluetooth, RSSI.

Abstract

The following report had as its primary mission the attempt to explain the problem to which it seeks to provide a solution, namely, the issue of location. In the following, we proceeded analytically and dogmatically from the perspective of those areas of electronics that would allow a response to the problem, always focusing on the fastest and most effective option. Special consideration was given in this study to the noise produced in the propagation of the signal and future acquisition for use in the estimation system, proposing various techniques for weighing it. This was followed by the presentation and testing of various technologies, based on radio frequency, which have been used to locate devices, together with the different results obtained by different authors who have focused their research on the subject.

Projected the state of the art, specifically regarding the location by radio frequency, part of the work focused on the study of the Raspberry Pi Zero W plate as a replacement for the Khepera IV robot. The Linux and Python operating systems were also studied, and a prototype program was designed in the programming environment to characterize the environment, and then the approximate position of the device was obtained; it was then taken to the Raspberry board that was installed on the robot. In spite of this, when performing the tests, when rotating the plate in its axis the measurement value presented great variance, reason why the plate was modified to include an omnidirectional external antenna that improved the performance of the measurement, correcting the error to around 10 centimeters without even performing the addition of a robust algorithm for the follow-up.

The need arose to improve the sampling time, so the filtering of the signal was raised assuming a Gaussian noise. As a response to the program, there was good performance in terms of sampling rates, and reducing the random variation of the sample also reduced the error by a certain proportion. Subsequently, with the presence of the error in the distance, the option of adding the method of least squares to obtain the estimation of the position was analyzed, where it was appreciated that, when adding a considerable number of references, the total dispersion in the estimation of the position varies considerably.

Keywords: Radio frequency, location, mobile robot, Bluetooth, RSSI.

Índice general

Introducción 1
Objetivos generales 3
Objetivos Específicos 3

1 Antecedentes Generales 4
1.1 Planteamiento de la situación a analizar 4
1.2 Soluciones al problema: Técnicas de localización. 4
1.2.1 Procesamiento de imágenes. 5
1.2.2 Radiofrecuencia 5
1.3 Técnicas de estimación de distancia 7
1.3.1 Tiempo de llegada ToA 7
1.3.2 Fuerza de señal recibida RSSI 7
1.3.3 Ángulo de llegada AoA 8
1.4 Técnicas de estimacion de posición 9
1.4.1 Trilateración 9
1.4.2 Triangulación 10
1.4.3 Otros sistemas 11
1.5 Métodos de apoyo 11
1.5.1 Maximum likelihood 11
1.5.2 Estimación Mínimos Cuadrados 12
1.5.3 Método del Centroide 13
1.5.4 Cramer-Rao Lower Bound (CRLB) 13
1.6 Tecnologías elaboradas. 14
1.6.1 Sistema de Posicionamiento Global 14
1.6.2 Ultra banda-ancha (UWB) 15
1.6.3 Comunicación inalámbrica 16
1.6.4 Bluetooth 17
1.6.5 Ultrasonido 17
1.6.6 RFID 18
1.6.7 Resultados experimentos investigados 18

2 Solución y tecnología asociada 20

2.1 Solución Propuesta 20
2.2 Bluetooth. 20
2.3 Técnica de estimación de posición 23
2.3.1 Técnica de Localización: RSSI. 23
2.3.2 Corrección de ruido gaussiano 23
2.3.3 Trilateración 24
2.3.4 Método de Mínimos Cuadrados 24
2.4 Robot Khepera 24
2.5 Raspberry Pi Zero W 26

3 Prototipo software 29
3.1 Ubuntu GBU/Linux 30
3.2 Beacons. 30
3.3 Descripción Programa 31
3.4 Caracterización de la señal 32
3.5 Estimación posición 35

4 Resultados Experimentales 37
4.1 Efecto filtro gaussiano 37
4.2 Prueba Estimación Posición 3 Dispositivos 39
4.3 Prueba Estimación Posición 7 Dispositivos 43
4.3.1 Prueba Detección mínima de 4 dispositivos 44
4.3.2 Prueba detección de 6 Dispositivos 46
4.3.3 Prueba Comparación Variación Numero de Dispositivos 47

Discusión y conclusiones 49

Bibliografía 52

Introducción

A raíz del avance exponencial del campo tecnológico, la automatización y la robótica se han cristalizado como vigorosos protagonistas en la consecución de diversas tareas y trabajos, buscando simplificar, apoyar o brindar mayor seguridad a los seres humanos en el rubro en que se desarrolle laboral, social, cultural, económicamente, etc. En este marco de ideas, y en la medida en que logros se cimentan y acrecientan, se busca adaptarlas a nuevas labores, aumentando también la cantidad de los robots autónomos en las diferentes áreas de trabajo. Así las cosas, se ha tornado imperativa la necesidad de una comunicación entre sí, en miras de mejorar el desempeño individual del robot en la tarea o trabajo que tenga como misión completar, incluso, permitir un trabajo cooperativo entre un gran número de robots que necesiten de cierta coordinación y sean conscientes de su entorno para la resolución de múltiples problemas. [1]

Es en este punto donde los robots se ven necesitados de la realización de un listado de tareas, priorizando su objetivo principal, pero sin descuidar las diferentes tareas necesarias para la realización de dicha tarea, se busca que tareas secundarias, de menor categoría, o que no añaden un valor agregado a la misión principal [2]–vale decir, que solo sirvan en la continuidad o seguridad propia de dicha tarea–, se realicen de la manera más óptima y breve posible. Una de las labores secundarias más importantes en el trabajo de los robots móviles es precisamente la localización y navegación de estos, debido a que se nos presenta con manifiesta importancia la correcta interacción de esta tecnología con su entorno, ya que solo será recordada en momentos de fallas o errores de medida que sobrepasen los límites permitidos. Así mismo, otro factor que se debe considerar es el límite respecto a la capacidad de procesamiento, tamaños, movilidad, fuentes de alimentación (baterías), entre otros.

Con este propósito, el presente estudio busca informar al lector acerca de distintas opciones tecnológicas (Ultra Wide Band, Bluetooth, Wifi, imágenes, etc.) y técnicas empleadas para la localización de un objeto de interés (LMS, CRLB, ML, etc.) y exponer acerca de la realización de un sistema eficaz para que el dispositivo móvil en cuestión (robot móvil) pueda realizar el trabajo de ubicarse en su entorno (gracias a las velocidades de procesamiento y capacidad de los nuevos dispositivos), en contraste a los métodos más comunes que utilizan un sistema de ayuda basado en una “torre de control” o computadora central, la que se encarga de establecer la comunicación entre dispositivos y de identificar la posición y dirección del robot, si bien este modelo simplifica el funcionamiento del programa interno del robot, añade costos extras al sistema al necesitar

equipo extra, una alimentación energética mayor y una comunicación extra entre el robot y la computadora central.

Explicado lo anterior, a través de la aplicación de un método descriptivo, analítico, comparativo y práctico, se proponen como objetivos del informe los que siguen: en primer lugar, presentar el estado del arte asociado a la localización de dispositivos; capítulo que implicará la descripción sobre las técnicas de localización más aplicadas, seguido de las tecnologías que estas emplean. En segundo lugar, la exposición del planteamiento general de la solución propuesta con base en el estudio anteriormente realizado; acápite que se destinará a un análisis más profundo de la tecnología Bluetooth y la técnica de estimación de distancia mediante RSSI, aplicado al robot Khepera IV presente en las dependencias del laboratorio, acompañado de un método de filtraje de la señal respecto al ruido a través del modelado gaussiano de la señal y la estimación de posición mediante trilateralización con apoyo del método de mínimos cuadrados.

En tercer lugar, producto de la incompatibilidad del robot con la tecnología a utilizar y nuevas tecnologías, se propone y entrega el estudio sobre un nuevo dispositivo, el mini computador Raspberry Pi Zero W que, gracias a su tamaño y a sus especificaciones técnicas, se convierte en un buen asistente en la localización para el robot Khepera IV. Se presenta además una modificación realizada en la placa que consiste en integrar una antena externa, con el fin de mejorar la resolución en el muestreo del RSSI, independizando la medición de la posición en la que se encuentre la placa Raspberry respecto a su centro.

En un cuarto punto, previo al desarrollo del programa, se estudiará el sistema operativo y el modelo de balizas de referencias, con el objeto de ampliar las herramientas en el desarrollo del sistema a crear para la localización. Simultáneamente, se confeccionará un programa que permita utilizar el Bluetooth, para alcanzar la finalidad consistente en la ubicación del robot. En conjunto con ello, se buscó que el programa permitiera la lectura de un gran número de balizas de referencia y una fácil inclusión de algoritmos para la optimización en la obtención de la posición. En un quinto punto, se exhibirá un análisis sobre el desempeño del prototipo del programa realizado para la localización de dispositivos con el fin de observar los tiempos de medición y el error en la estimación previo a la inclusión de algoritmos de corrección. Por último, se analizarán cómo influye el número de dispositivos en la rapidez de medición, el error de la estimación de la distancia entre beacon y robot, influyendo directamente en el error de estimación de posición y dispersión en la localización del mismo, entregando conclusiones y futuros proyectos que surgen a partir del modelo elaborado.

Objetivos generales

- Analizar las diferentes técnicas, tecnologías y sistemas para la localización de un dispositivo en miras al diseño de un sistema en base a radiofrecuencias.

Objetivos Específicos

- Estudio de la factibilidad de implementación del sistema a diseñar
- Diseñar un prototipo del programa de adquisición de datos.
- Implementación de programa en un dispositivo portable.
- Análisis del desempeño del programa y del dispositivo y buscar alternativas para la realización de mejoras.

1 Antecedentes Generales

1.1 Planteamiento de la situación a analizar

La Principal problemática en el desarrollo de robots que se hagan cargo de diversas tareas y trabajen en conjunto para conseguir desempeñar labores de mayor complejidad con o sin apoyo de otros robots, aterriza en la prioridad que se le debe asignar a cada una de las funciones presentes en los robots, cambiando de acuerdo a la realización de dichas funciones, realizando un monitoreo minucioso de que la conclusión sea correcta y eficaz. Aludiendo al monitoreo, este se puede realizar tanto presencial como remotamente; y es allí donde se encamina nuestra labor en miras a reducir este proceso sobre el robot, permitiendo que pueda discernir los problemas que se le presentan. Precisamente, uno de las dificultades que surgen es la convivencia del robot en un sistema que, sin lugar a dudas, presentará notables diferencias respecto del rubro en el que se inserta. Ahora, sin importar cuál sea el rubro, en todos ellos será necesaria la movilidad del robot en el entorno, junto con el logro de identificar donde se encuentra, todo con el fin de poder acceder a la posición correcta del robot, en cualquier momento, y que este cumpla su objetivo previamente programado.

Desde la perspectiva de un escenario atestado por limitaciones computacionales que deben enfrentan los robots, específicamente en lo que respecta la optimización de energía y tiempo de procesamiento digital, se busca priorizar la mayor cantidad de recursos para la tarea principal del usuario. De allí surge el imperativo de que el monitoreo sea lo más breve y óptimo posible, incluyendo el proceso de localización y movilización del robot. Bajo este concepto se busca conseguir, con la mayor precisión viable, la ubicación del robot empleado, a su vez, la menor cantidad de recursos disponibles.

1.2 Soluciones al problema: Técnicas de localización.

Hoy en día, la detección de objetos es primordial para cuidar de los robots móviles y/o de su entorno, buscando además que el gasto de recursos (en términos de tiempos y costos) sea el menor posible, teniendo en consideración que la tarea del robot no es la localización, sino aquella misión para el que fue diseñado. Para estos efectos, partiendo de la premisa de que el proceso debe otorgar una correcta información al usuario (al robot o al operario que lo requiera) sobre su localización, es que se obra con base en sensores que, junto a un algoritmo diseñado por el autor de dicho sensor, efectivamente permiten llevar a cabo dicha tarea. Frente a este desafío, la

atención en la actualidad se centra en dos grandes áreas, a saber, la radiofrecuencia y el procesamiento de imágenes.

1.2.1 Procesamiento de imágenes.

En relación con la obtención de la posición de un objeto, no obstante, existen diferentes variantes en los esquemas de localización elaborados, en general, se cuenta con un proceso común en cada uno de ellos. En concreto, este proceso consiste en conseguir una imagen donde se encuentra el objetivo a ubicar, para compararlo con una database que posee distintas imágenes previamente analizadas con base en distintos algoritmos, como los utilizados en [3] denominados por histogramas de color, descomposición wavelet y *shape matching*. Además, en [4] se plantea un método más moderno que desarrolla un dispositivo capaz de realizar una toma de la escena que permita obtener la posición aproximada con un error relativamente bajo. Ahora bien, el motivo que justifica que el procesamiento de imágenes no se haya masificado comercialmente para métodos de localización, es el gran consumo en términos de energía, dinero y tiempo de procesamiento digital que involucra. Hecha esta salvedad, se debe señalar que este sistema de detección se ha introducido, con intención de mejorarlo, en otros rubros como la seguridad y búsqueda de objetivos.

A modo de referencia se destaca el sistema implementado en el laboratorio de robótica en la Facultad de ingeniería de la Pontificia Universidad Católica de Valparaíso (ver Figura 1-1), donde se consiguió un error de estimación de posición menor a 5 centímetros con tiempos de respuesta cercanos a 0.3 segundos. [5]

Figura 1-1 Plataforma con sistema de localización por cámara en laboratorio PUCV [5]

1.2.2 Radiofrecuencia

El uso de la tecnología de radiofrecuencia, como medio para localizar objetos, es explicado de forma simple en el mundo de las ondas electromagnéticas de menor frecuencia, como lo son las

ondas acústicas; ello puntualmente a través de la eco localización que utilizan de forma biológica diversas especies en la naturaleza, como los murciélagos y los delfines, para localizar a sus presas y sus obstáculos. En el caso de los murciélagos, el animal envía un sonido (en el caso de los murciélagos de 20 kHz aproximadamente, fuera del rango audible) que al impactar sobre un objeto o un animal refleja el sonido de regreso al murciélago, lo que le sirve para conocer a que distancia se encuentra.

De manera análoga, se han desarrollado varias técnicas para replicar este efecto y/o modificarlo para conseguir detectar diversos obstáculos. Entre estas destacan el empleo de una antena emisora –la que busca la detección de obstáculos o ubicarse en su entorno– y antenas secundarias, las que representan una referencia con el fin de formalizar la localización. El ejemplo clásico para estos sistemas es la tecnología radar, donde en términos simples, se emite una señal en una determinada dirección (luego se replica proceso en otra dirección haciendo un barrido espacial) en la búsqueda de objetos (generalmente usado en la detección de vehículos), estimando la posición en relación al tiempo de retorno de la señal luego de que impacte en dicho objetivo (ver Figura 1-2). [6]

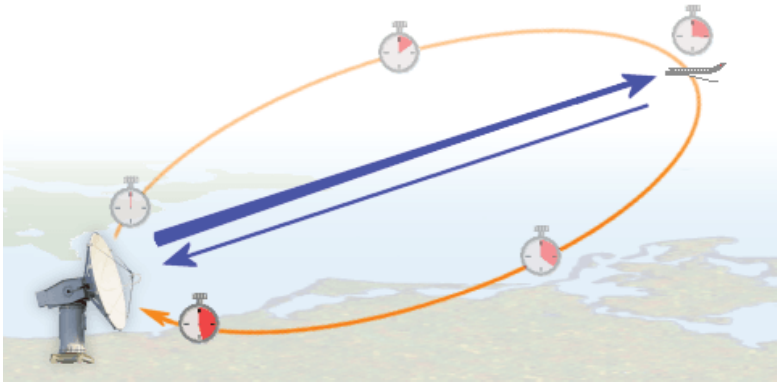

Figura 1-2 Tecnología radar (Fuente: www.radartutorial.eu)

A propósito, estos sistemas se desarrollan de muchas formas, entre las que destacan: un nodo central que procesa y localiza la posición de los robots o dispositivos enviándoles la posición de estos respecto a un espacio previamente definido; integrar el procesamiento de las distancias en el robot apoyándose de diversos sensores que indican el espacio donde se encuentra; y por último, el intercambio de información entre diferentes robots que comunican su posición referente a emisores fijos.

A su vez, los métodos utilizados para conseguir la locación se pueden englobar en dos etapas, i) identificar la distancia del robot a diferentes puntos de referencia, y ii) luego analizar la proximidad entre estos puntos para generar el mapa virtual y localizar al robot en el punto en que se encuentre. [7]

Por último, cuando se habla de utilizar tecnología con base en ondas electromagnéticas, se debe tener muy presente el control del ruido, pues en él se ven envueltos los sensores y antenas,

ocasionando que se obtenga muchas veces información errónea, contaminando las mediciones e impidiendo la consecución de la localización correcta.

1.3 Técnicas de estimación de distancia

Para evaluar la distancia entre las referencias y el robot móvil, se ha centrado la atención en diferentes características de la emisión y/o recepción de las señales, como el tiempo que demora en enviar o recibir la señal, o las propiedades en la atenuación de la señal en cuestión. Hoy en día, se trabaja en torno a 4 técnicas que aprovechan la comunicación para inquirir la distancia entre el robot y las referencias.

1.3.1 Tiempo de llegada ToA

Esta técnica hace uso de la comunicación emisor-receptor, midiendo el tiempo que demora el usuario en recibir una respuesta a la señal enviada a la referencia [8] [9] [10]. En este punto, es obligatorio considerar el tiempo de procesamiento en el receptor para una correcta medida. En concordancia con la naturaleza de la señal enviada y al conocimiento de la velocidad de esta (velocidad de la luz o del sonido), es posible calcular la distancia según la siguiente relación:

$$Distancia\ emisor - receptor\ [m] = Velocidad\ de\ la\ señal\ \left[\frac{m}{s}\right] * tiempo\ de\ llegada\ [s] \tag{1-1}$$

El tiempo de llegada ha demostrado ser una técnica precisa en trabajos a gran escala, sin embargo, al requerir de la sincronización entre los relojes presentes en emisor y receptor, es necesario realizar un trabajo extra que eleva el costo del equipamiento. Esta situación derivó en la técnica llamada **Diferencia de tiempos de llegada (TDoA)** la que, a diferencia del ToA, necesita la inclusión en el emisor de dos señales, realizando el análisis de la distancia a partir de estas dos muestras respecto a las velocidades conocidas. [9]

1.3.2 Fuerza de señal recibida RSSI

Con el objetivo de determinar la distancia entre el emisor y el receptor, esta técnica utiliza la intensidad de recepción de la señal, la que aprovecha la atenuación de la onda en la medida en que esta se aleja del emisor [11]. Al utilizar la atenuación de la señal, encontraremos muchos factores que pueden afectar el valor de este parámetro, vale decir, los obstáculos, las multi trayectorias, la temperatura, el ruido del ambiente, entre otros factores. Con todo, se debe destacar que una porción del ruido puede ser subsanada a partir del análisis de este factor al incluir al modelo de estimación de posición algoritmos probabilísticos [12], o acompañando a la comunicación de la señal con filtros de distintas categorías que reduzcan el ruido. Aun así, debido a la simpleza en la obtención de datos –gracias a la tecnología disponible en la actualidad, y que es un parámetro que el mismo sistema entrega al hacer contacto– es posible alcanzar un desempeño aceptable al acumular una muestra considerable.

1.3.3 Ángulo de llegada AoA.

El parámetro fundamental en el desarrollo de esta técnica es la obtención del ángulo de incidencia de la señal de comunicación entre el robot móvil (punto a localizar) y los diversos nodos de referencia que están asociados al sistema. En este sentido, y para detectar el ángulo de incidencia se han desarrollado una diversa cantidad de dispositivos, entre los que destacan diseños basados en arreglos de antenas y a través de plataformas rotativas [13].

Aludiendo al segundo dispositivo, los diseños de plataformas rotatorias son una técnica que permite obtener el ángulo de incidencia de manera directa, y que consiste en el uso de una antena de tipo directivo que rota en su eje para ubicar la posición de mayor intensidad entre el emisor y el receptor. [13] [14] [15]

De manera puntual, existe una gran cantidad de diseños entorno a los arreglos de antenas para estimar el ángulo de incidencia; uno de ellos compara el tiempo de llegada de la señal en el arreglo (ToA) con la distancia conocida entre las antenas del arreglo (arreglo lineal), para realizar un trabajo geométrico y conseguir el ángulo de la señal, de forma simple se ilustra en la Figura 1-3 Trabajo geométrico conversión ToA a AoA . Se destaca que es necesario normalizar los ToA en relación al tiempo conseguido de la primera antena que detecte la señal. [16]

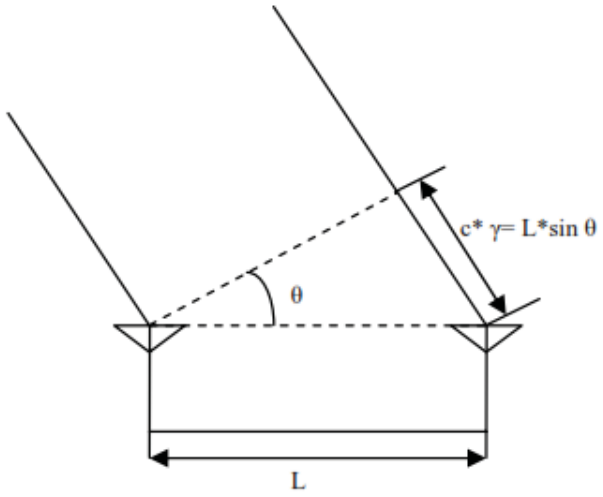

Figura 1-3 Trabajo geométrico conversión ToA a AoA [16]

Además, existen arreglos de antenas que utilizan dispositivos auxiliares o modifican el arreglo de antena para evitar el caso de una plataforma rotativa, como en el primer método. Así, por ejemplo, se tiene el proyecto en base a una femtocelda [17] que consiste en incluir múltiples antenas directivas con el fin de abarcar los 360° y determinar la dirección de la distancia a través del RSSI que recibe de la antena.

1.4 Técnicas de estimacion de posición

Una vez conseguida la distancia entre el robot y los nodos, es necesario analizar en qué lugar del área acotada se encuentra, con los datos ya conocidos. Con este fin, se han diseñado técnicas directas (como la triangulación y la trilateralización) que se complementan con métodos de apoyo que se rigen por algoritmos para responder ante diversas alteraciones de los datos muestreados (debido al ruido, multitrayectorias, atenuaciones no deseadas, etc.).

1.4.1 Trilateración

La trilateración consiste en la obtención de la posición espacial del robot respecto a la disposición de un mínimo de tres nodos de referencia, a través de circunferencias (en un plano de dos dimensiones, en 3 dimensiones se necesitan al menos 4 nodos) cuyo radio es la distancia obtenida por los métodos anteriores entre el robot y cada nodo (como se puede visualizar en la Figura 1-4). Para esta técnica se emplea un sistema de ecuaciones (1-2) donde se intersecta cada circunferencia (Destacando que el centro de cada nodo de referencia esta previamente configurado y conocido) obteniendo el punto de interés.

$$(x - x_1)^2 + (y - y_1)^2 = {m_1}^2 \tag{1-2}$$

$$(x - x_2)^2 + (y - y_2)^2 = {m_2}^2$$

$$(x - x_3)^2 + (y - y_3)^2 = {m_3}^2$$

$$\vdots$$

$$(x - x_n)^2 + (y - y_n)^2 = {m_n}^2$$

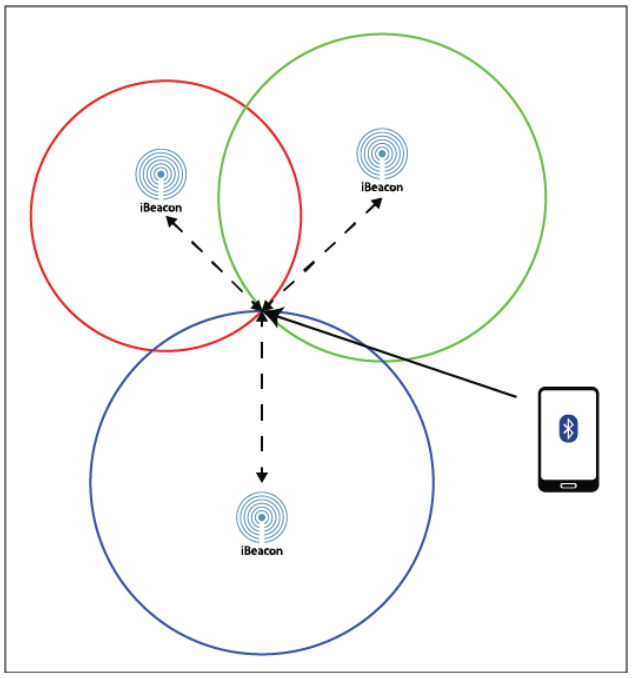

Figura 1-4 Técnica Trilateración de Posición (Fuente: www.iris.hdplus.es)

Se debe tener en consideración que, en el mundo práctico, el error que arrastra la estimación de la distancia obtenida entre el robot y cada nodo de referencia impiden la obtención exacta del punto de interés mediante la resolución directa de esta ecuación, lo que concluye en la creación de un área con posibles soluciones (ver Figura 1-5).

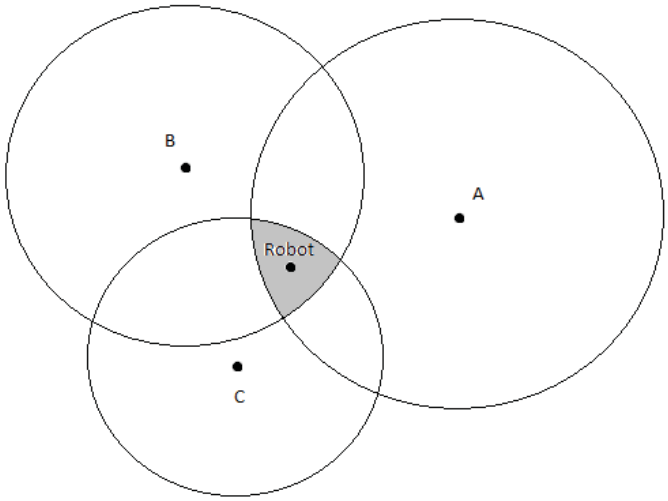

Figura 1-5 Error distancia robot-nodo referencial

1.4.2 Triangulación

A diferencia del método anterior, la triangulación calcula la distancia a través de ángulos obtenidos por el sistema, por lo que es posible reducir un nodo de referencia pues, para este caso, bastan dos nodos referenciales (en dos dimensiones) y la posición ya conocida de ellos, para luego realizar el trabajo trigonométrico con base en el teorema del seno y del coseno [18].

Gráficamente puede observarse en la Figura 1-6: a la izquierda se plantea el esquema en un plano 2D de la situación y a la derecha se plantea la simplificación acotada a un solo triangulo donde es conocida la distancia AB (nodos de referencia) y los ángulos α y β. Una vez conseguidas las distancias AC y BC se puede realizar el mismo planteamiento de la triangulación.

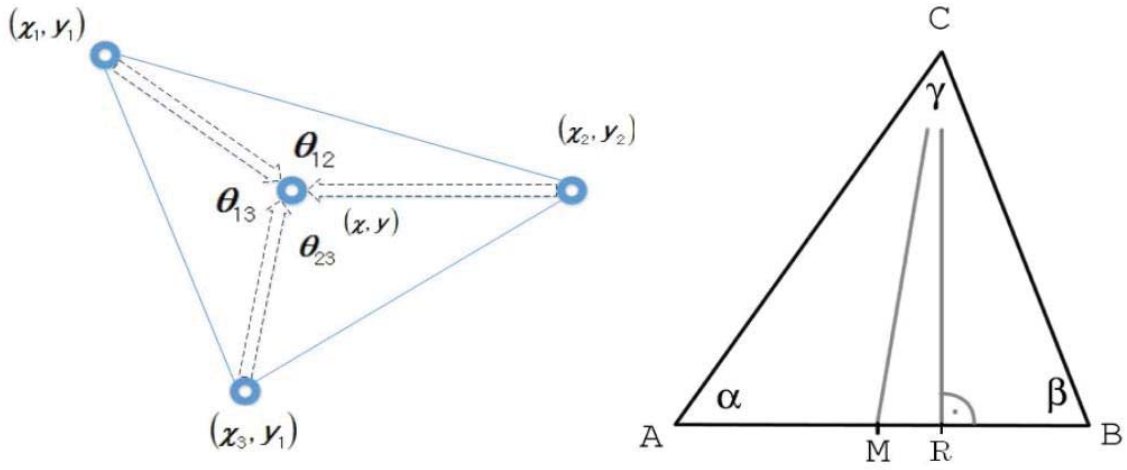

Figura 1-6 Triangulación con 3 nodos de referencia [18]

1.4.3 Otros sistemas

- La Multilateración es similar en su desarrollo a la trilateración, empero para obtener la distancia se utiliza el método de distancia TDoA.
- La Proximidad es una técnica que se basa en la detección de un objeto a poca distancia. Los métodos más simples son a través de contactos (como sensores de presión), monitoreo de puntos de accesos para corroborar si está en el rango de otro punto de acceso, o mediante sistemas automáticos de identificación (como el utilizado en las tarjetas de crédito)
- El Análisis de Escena, que es otra medida, consiste en almacenar información de distintos elementos localizados y comparar futuras variaciones en su posición. [7]

1.5 Métodos de apoyo

Como se ha mencionado en puntos anteriores, la estimación de la distancia entre el robot y los nodos de referencia acarrea distintos niveles de ruido, ocasionando que los sistemas planteados de manera teórica no puedan ser llevados directamente a la práctica. Por este motivo, y pensando en no integrar elementos físicos al sistema, se han desarrollado diversos modelos matemáticos fundados en algoritmos probabilísticos con el fin de mejorar la estimación de la posición del robot.

1.5.1 Maximum likelihood

Este método se utiliza partiendo del prisma de que el ruido ambiental generalmente se clasifica como una distribución gaussiana. [9] Esto nos conduce a replantear la variable medida añadiendo una distribución gaussiana:

$$z_i = h_i(x) + v \tag{1-3}$$

$$v \sim \mathcal{N}\left(0, \sigma_i^2\right) \qquad i = 1, \dots, M\ (total\ mediciones) \qquad x = [x, y]^T \in \mathbb{R}^2$$

Donde h_i representa las variable medidas dependiente de algún valor (x) y v representa una distribución gaussiana centrada en 0 con una desviación típica σ distinta para cada medición.

Para una medición gaussiana simple, se tiene que la función de verosimilitud (*Likelihood function*) es:

$$p(z_i|x) = \frac{1}{\sqrt{2\pi}\,\sigma_i} \exp\left(-\frac{1}{2\,\sigma_i^2} \cdot (z_i - h_i(x))^2\right) \tag{1-4}$$

Para definir la función para el sistema completo, se redefinen z_i y $h_i(x)$, además de crea la matriz diagonal R donde su diagonal está compuesta por las covarianzas σ_i^2:

$$p(z_t|x) = \frac{1}{\left|\sqrt{2\pi}\,R\right|} \exp\left(-\frac{1}{2} \cdot \left(z_t - h_t(x)\right)^T R^{-1} \left(z_t - h_t(x)\right)\right) \tag{1-5}$$

$$z_t = [z_1, \dots, z_M]^T \qquad h_t = [h, \dots, h_M]^T$$

$$R = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_M^2 \end{bmatrix}$$

Posteriormente, en la elaboración del algoritmo de resolución se busca minimizar el argumento de la función con base en el parámetro inicial del que depende $h_i(x)$.

1.5.2 Estimación Mínimos Cuadrados

Uno de los métodos más utilizados al emplear la técnica de trilateralización o triangulación para la estimación de la posición del objetivo, ya que permite estimar la posición a pesar de que el sistema de ecuaciones no tenga solución. El método comienza acomodando el sistema de ecuaciones (1-2) mediante manejo matemático se puede representar según [12]:

$$Y_{Nx1} = A_{NxM} \cdot X_{Mx1} \tag{1-6}$$

Donde Y y A indican vectores conocidos, obtenidos de reducir el sistema de ecuaciones, con N referente a la dimensión en la que esté trabajando (2D, 3D), M representa el número de ecuaciones (una vez reducido el sistema) destacando que a medida que el número de ecuaciones aumenta (consideración de nodos referenciales extras) el error disminuye, X representa el vector desconocido (para el caso de esta resolución corresponde a la posición del robot referente a las distancias calculadas).

Luego, el método prosigue explicando que la ecuación (1-6) presenta error producto a la no idealización de las mediciones, y se asocia a la siguiente suma cuadrática:

$$f(X) = (AX - Y)^2 = (AX - Y)^T (AX - Y) \tag{1-7}$$

Seguido, se expande la multiplicación y se procede a optimizar el error según igualar a cero la derivada de f(x) respecto a x, como se aprecia en la ecuación.

$$f(X) = (AX)^T AX - (AX)^T Y - Y^T AX - Y^T Y \tag{1-8}$$

$$\frac{df(X)}{dX} = 0 = \frac{X^T A^T AX - X^T A^T Y - Y^T AX - Y^T Y}{dX}$$

$$\frac{df(X)}{dX} = 0 = 2A^T AX - A^T Y - Y^T A$$

$$2A^T Y = 2A^T AX$$

De este modo se consigue un valor estimado de la posición del robot. [12]

$$X = (A^T A)^{-1} A^T Y \tag{1-9}$$

1.5.3 Método del Centroide

Este modelo parte del hecho que el sistema de ecuaciones no entrega una solución debido a la existencia de error, lo que determina que se trabaje en el área encerrada por las circunferencias trazadas en la Figura 1-7, de donde se puede estimar que el robot se encuentra en el centro del área anteriormente graficada. Esta resolución implica definir un algoritmo para obtener las intersecciones A, B, C y D, para luego promediar estos 4 puntos. [12]

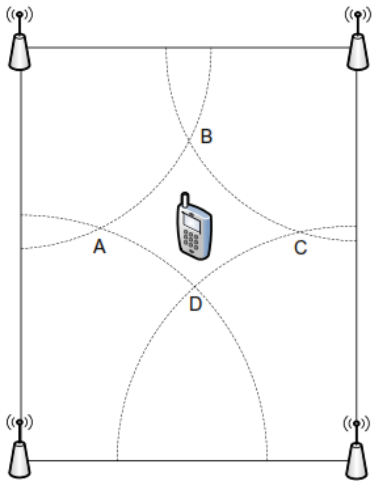

Figura 1-7 Método del centroide [12]

1.5.4 Cramer-Rao Lower Bound (CRLB)

El mérito del método de CRLB se centra en disminuir el error en la estimación de la posición, a través del trabajo sobre la señal y los diversos errores. Con todo, es necesario para desarrollar de forma práctica este método, que el sistema de adquisición de datos entregue una información más completa de la señal recibida pues, a más del máximo valor detectado, necesita conocer la amplitud y el desfase de las multi trayectorias que sean detectables. A pesar de lo anteriormente

mencionado en [19], puede observarse la aplicación de este método trabajando con la técnica ToA sobre una señal del tipo *Impulse Radio-Ultra Wideband* (IR-UWB) con simulación de ruido gaussiano. Este método fue trabajado sobre la base teóricamente planteada en [20], donde se plantean las bases del modelo y las diversas propiedades necesarias, que permiten la varianza al estimar parámetros desconocidos.

El modelo, como punto de partida, realiza el planteamiento matemático explicado en 1.5.1, efectuando una doble derivada en la ecuación (1-4), y un ligero arreglo para obtener la varianza de la señal determinística con base en parámetros desconocidos. Luego se plantea la matriz de Fisher que consiste en realizar derivadas parciales respecto a los parámetros a conseguir. Como resultado se obtiene la dispersión de cada parámetro, concluyendo en una mejor idea de cómo está actuando el ruido, en miras a desempeñar ajustes y disminuir finalmente su impacto. [16]

1.6 Tecnologías elaboradas.

1.6.1 Sistema de Posicionamiento Global

El Sistema de Posicionamiento Global (GPS) es la tecnología más usada para localización en exteriores, gracias al entramado de satélites alrededor del globo terrestre y a la baja taza de error que entrega en relación con las distancias que abarca (pese a abarcar el globo terrestre no sobrepasa los 50 metros). Otro rasgo relevante es que utiliza el método de localización por distancia de ToA, acompañado de la estimación de posición por triangulación. Cada satélite emite la señal en dos frecuencias de onda L1 (1,575.42 MHz) y L2 (1,227.6 MHz), las que permiten que esta atraviese la atmosfera y, a la vez, gracias a la altura en la que se encuentran los satélites, se consigue línea directa en ambientes exteriores.

Cada satélite envía su posición y hora exacta, con el fin de revisar el estado de sincronización entre el satélite y el receptor. Luego del ajuste y de captar cierto número de muestras de diferentes satélites, junto con un algoritmo interno basado en la técnica de triangulación (Figura 1-8) obtiene su posición en longitud, latitud y altitud. Dentro del sistema, además, se agregan filtros para disminuir la gran cantidad de errores que se añade producto del clima terrestre, de otras señales, entre otros. [8]

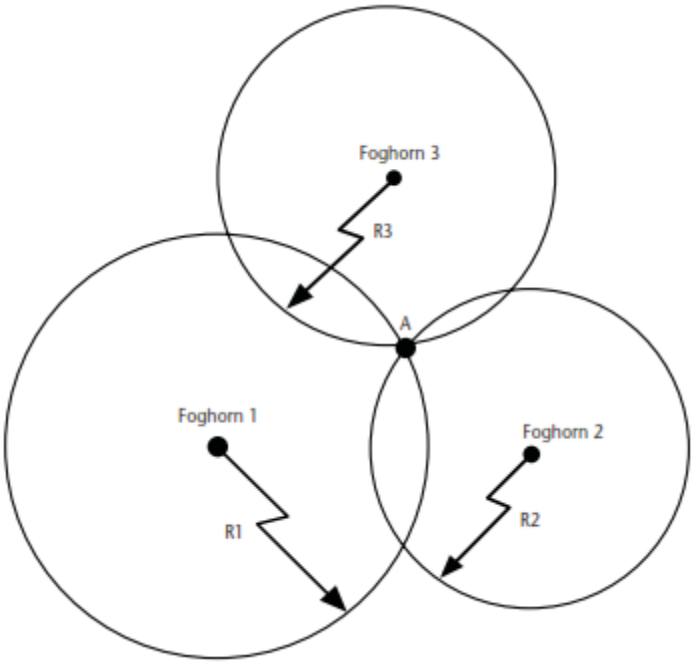

Figura 1-8 Triangulación GPS [8]

En particular, el mayor problema de este sistema se presenta en ambientes interiores (hogares, fábricas, etc.) donde el orden del lugar es más propenso a cambios, y las señales de los satélites se desvanecen o desvirtúan producto de los muros y/o techos. Ante ello, es necesaria la búsqueda de nuevas referencias o sistemas que permitan hacer consiente al receptor del espacio que ocupa en el entorno.

1.6.2 Ultra banda-ancha (UWB)

Los sistemas basados en UWB utilizan una extensa banda que se mueve entre los 3,1 GHz y los 10,6 GHZ, de ahí que permite un buen desempeño en velocidad de envío-recepción de información, a diferencia de los demás sistemas de comunicación en que los niveles de potencia que utilizan para la comunicación son realmente bajos. Se debe agregar, además, que la señal es enviada en banda base con una modulación por amplitud a través de pulsos, de modo tal que la tecnología necesaria para su implementación es económica y permite una mejor *performance* en el traspaso de información. En contraste con las mejoras de desempeño, el área de influencia de este sistema no sobrepasa los 20 metros y, sumado a su poca producción en serie, aún no se ha masificado su uso para comunicaciones.

Al tener bajos niveles de densidad de potencia en la emisión de las señales, conduce a un SNR bajo, lo que implica que se pueda confundir con el ruido ambiental (relación UWB *Spectrum* y *Noise floor* en Figura 1-9), entorpeciendo el reconocimiento de su comportamiento sin el equipo y la codificación adecuada. Por consiguiente, la seguridad en la comunicación es mayor que en otras tecnologías.

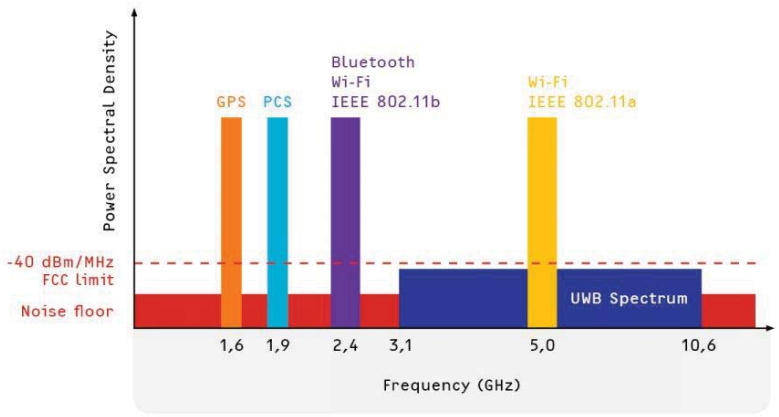

Figura 1-9 Espectro de frecuencias señales inalámbricas más utilizadas (Fuente: www.eliko.ee)

Dentro de sus características destaca el área de la localización, gracias a su alta frecuencia de transmisión, aumenta considerablemente la velocidad de transmisión, mejorando la precisión al poseer una mayor cantidad de datos. En [16] se plantea que, gracias a la velocidad y a la obtención simple del AoA, mejora la muestra en comparación a otras tecnologías, en [19] se incluye un análisis del efecto de la multi trayectoria y su influencia en la estimación de la posición.

Conjuntamente con lo dicho, este sistema es una buena opción para la localización de objetos, pues es posible subsanar el área de funcionamiento con la implementación de múltiples nodos. No obstante, ello, se ve limitada en la poca masificación de sus dispositivos, de modo tal que resulta interesante el seguimiento del progreso de esta tecnología.

1.6.3 Comunicación inalámbrica

La tecnología inalámbrica (también conocida como WIFI, WLAN, WSN, protocolo 802.11) permite utilizar un gran número de técnicas de localización, debido a su disponibilidad actual, ello pues existe un gran número de dispositivos que se comunican inalámbricamente en el mercado, incluida la masificación de computadores, notebooks y smartphones. En particular, transmite en frecuencias de 900 MHz, 2.4 GHz y 5.7 GHz [14], de manera que, en comparación con UWB, presenta una respuesta con un peor desempeño al momento de incluirlo en localización, pero con un mayor rango de funcionamiento al enviar información con niveles de potencias más altos y mayor disponibilidad de equipos.

Es fundando en esta masificación de equipos, y al sistema de ubicación de nuevos puntos de acceso, que se puede obtener información, como el tiempo de respuesta y la intensidad de la señal de conexión. En [21] se plantean métodos de auto sincronización para la técnica de ToA, reconocimiento del ángulo de incidencia (AoA), con el fin de aprovechar estas técnicas de un modo más simple ahorrando tiempo de procesamiento; al mismo tiempo en [22] se desarrolla un modelo de navegación en base al RSSI con buenos resultados.

1.6.4 Bluetooth

En cuanto a la tecnología Bluetooth (también conocida como WPAN), en concreto, y apuntando a sus especificaciones, transmite a 2.4 GHz con un alcance cercano a los 10 metros y con un bajo consumo debido a su transmisión de baja potencia. Así las cosas, fue pensado en sus inicios como medio de transmisión de datos, imágenes y/o audios en dispositivos móviles sin necesidad de conexiones por cable y a una distancia más cómoda, además gracias a estas características le fue permitido masificarse y estar presente en un gran número de dispositivos.

A diferencia de las tecnologías anteriormente mencionadas, debido a la forma que están diseñados los protocolos de comunicación y como están estructurados los dispositivos Bluetooth, los únicos parámetros que pueden ser usados para identificar la localización están asociados a la intensidad y calidad de la comunicación, aspecto que será tratado en el siguiente capítulo. De todas formas, derivado de su bajo coste de implementación y a la amplia oferta que existe en dispositivos Bluetooth, lo torna una buena opción para localización.

El parámetro RSSI en la tecnología Bluetooth está limitado al Rango de Potencia de Recepción Dorado (*Golden Receiver Power Range*, GRPR), el que entrega información de cuan confiable es la señal respecto a los niveles de la potencia recibida (P_{RX}). Se profundizará en las siguientes páginas sobre su efecto en la localización.

1.6.5 Ultrasonido

La tecnología de ultrasonido utiliza ondas que son enviadas aproximadamente a 40 kHz. Para ser más específico, este sistema utiliza el rebote de la señal en algún obstáculo y registra el tiempo que demora en ir y volver la señal al transmisor. Es ampliamente usado, por una parte, en medicina, pues permite efectuar una imagen fundado en los distintos niveles que registra al realizar múltiples mediciones de un lugar determinado.

Por otra parte, para uso doméstico se ve restringido en la búsqueda de eficiencia del consumo eléctrico, por lo que las distancias de funcionamiento no sobrepasan las decenas de metros. Aun así, es utilizado como método auxiliar a las tecnologías anteriormente enunciadas en técnicas como la TDoA. No obstante, ello, ya que es un sistema que no necesita utilizar nodos de referencia para estimar distancias, es posible su uso como apoyo para evitar obstáculos a distancias cercanas. Uno de los sistemas desarrollados se llama Cricket [23] [24] desarrollado por el MIT. Este sistema mezcla WSN (*Wireless Sensor Network*) y ultrasonido para desarrollar el posicionamiento en base a la técnica TDoA a través de la implementación de múltiples nodos referenciales (Figura 1-10).

Figura 1-10 Trasmisor-Receptor Cricket (Fuente: www.cricket.csail.mit.edu/)

1.6.6 RFID

La tecnología RFID (*Radio Frecuency identification*) consiste en la ubicación de una antena emisora-receptora y múltiples tags que contienen información grabadas en su interior (su rango de funcionamiento es de aproximadamente 6 metros).

Fundado en la simple elaboración de los tags, este sistema se observa viable para localización. Concretamente, los tags actúan como nodos referenciales instalándolos en posiciones determinadas, utilizando poco espacio. A su vez, la antena capta la intensidad de la comunicación con los tags, lo que permite utilizar la técnica de RSSI para estimar posición. Según el material investigado [25] [26] se indica que es un sistema aceptable en términos de ruidos y respuesta en ambientes sin excesiva cantidad de ruido o distorsión, lo que hace ineludible nuevos estudios sobre esta tecnología.

1.6.7 Resultados experimentos investigados

A continuación, se entrega un resumen de los resultados obtenidos en los diferentes trabajos (Tabla 1-1), en los que precisamente se cimienta la investigación realizada respecto de la localización de robots en interior mediante radiofrecuencia. Cabe destacar, además, que los niveles de error conseguidos (que en algunos casos alcanzan magnitudes de metros de distancia) son aceptables en relación con la superficie abarcada en el desarrollo de los variados experimentos llevados a efecto para cada tecnología empleada. Ahora bien, la gran magnitud de experimentos plantea la exigencia de obtener la localización en ambientes de gran tamaño; en contraste, se buscará analizar el efecto de la localización en espacios más reducidos, donde el error no debe superar el tamaño del robot en cuestión, buscando arribar a su utilidad en tareas sincronizadas y otorgarle cierta independencia a este.

Tabla 1-1 Resumen errores en estimación trabajos investigados

Experimento	Espacio Trabajado		Error Estimación
	X[m]	Y[m]	X[m]
Shankar [3] I	-	-	5-10%
Corso [4] I	-	-	6%
Zhou [11] B	6.8 m 1D		1.2 max
Yang [12] B	6	7	0.2
Albert [15] W	180 m^2		0.3 ~ 0.9
Ma and Law [16]	24	24	Simulación
Woo [18] R	7	6	<5%
Biswas [22] W	100	30	0.2 ~ 1.8
Burgard [25] R	5	5	1

W: Wireless B: Bluetooth I: Imagen U: UWB

2 Solución y tecnología asociada

2.1 Solución Propuesta

En el presente acápite se dará preferencia a la utilización de técnicas por radiofrecuencia sobre el procesamiento de imágenes, fundado en la búsqueda de un modelo de bajo consumo eléctrico y de bajo consumo a nivel de procesamiento digital. Junto con lo anterior, se perseguirá de reducir el gasto en equipamiento por medio del manejo de sistemas de comunicación de acceso masivos como Bluetooth y Wifi. Para este diseño se aspirará, en primer lugar, experimentar con tecnología Bluetooth partiendo de la gran variedad de dispositivos presentes en el mercado. Además, y en segundo lugar, se estudiará su desempeño con base en los nuevos avances pues, al ser una tecnología de bajo consumo, la detección presenta cierta variabilidad que puede afectar las mediciones.

A continuación, se prodigará un análisis más acabado respecto de la tecnología Bluetooth y de la técnica de localización RSSI. Asimismo, se realizará una introducción al robot Khepera IV, teniendo en consideración que se pretende que el robot sea consciente de su ubicación, con miras de ser empleado futuros proyectos en el desarrollo de tareas y/o interacción con otros robots en un ambiente común, junto al robot Khepera IV se presenta la placa Raspberry Pi como dispositivo auxiliar para realizar la localización del robot, debido a la incapacidad del robot de añadir nuevos dispositivos.

En lo que respecta a los nodos de referencia, existen en el mercado una amplia gama de dispositivos (mejor conocidos como “beacons o balizas”) que permiten repetir su señal para su detección y el análisis de su intensidad de recepción (RSSI). Producto del conocimiento de estos dispositivos, se utilizarán, en un principio, celulares smartphones ya que junto con poseer en su diseño antenas Bluetooth de tecnología avanzada, sumado al desarrollo del sistema operativo (Android), es posible simular el funcionamiento de los beacons, lo que facilitará el desarrollo del sistema; permitiendo, en último término, centrar la atención en el diseño del receptor de estas señales, que permita al robot estar al tanto de su posición.

2.2 Bluetooth.

En [27] se realizó un análisis de los diferentes parámetros de comunicación, en los que se da cuenta de la tecnología Bluetooth 1.2, buscando reducir las opciones para una correcta

localización, luego de realizar pruebas respecto a variaciones de distancia, clase de Bluetooth y movimiento del robot a localizar. Se destacó la respuesta del parámetro Rx (intensidad de recepción), que demostró cierta relación en torno a diferentes distancias, pero limitado, no obstante, a los rangos permitidos de transmisión del Bluetooth.

Posteriormente en [11] se efectuó el examen del parámetro RSSI. Así mismo, se planteó en su oportunidad su dependencia con el *Golden Receiver Power Range* (GRPR), que decide en qué niveles de potencia la conexión será viable para la comunicación. En esta línea, al observar la Figura 2-1, se llegó a la conclusión de que era necesario mantener la medición del RSSI sobre 0 para evitar la relación única entre RSSI y a la potencia recibida, dando como resultado un sistema que puede estimar la distancia a una taza de error de un metro aproximadamente.

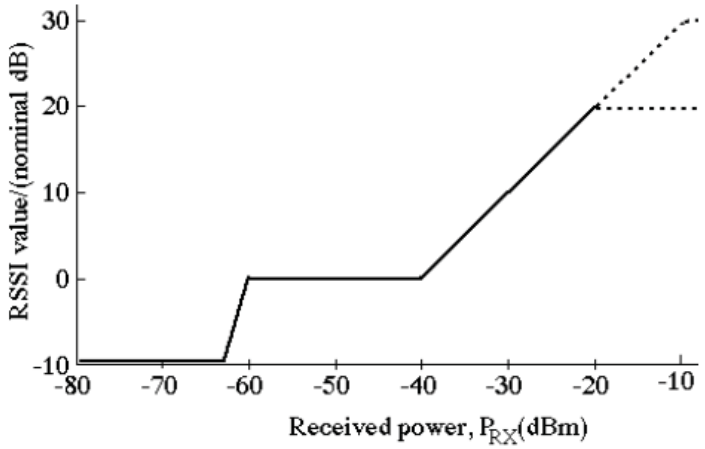

Figura 2-1 Efecto GRPR sobre RSSI Bluetooth [11]

Desde la versión 2 de Bluetooth, junto con la mejora de velocidad y potencia, prontamente en la versión 2.1 se hizo accesible al usuario la medición de la intensidad de recepción de conexión sin entablar una conexión directa, permitiendo agilizar y mejorar la obtención del parámetro RSSI, pero aun limitado al GRPR al heredar los protocolos del viejo sistema. En [12], al utilizar un nuevo algoritmo basado en RSSI con una versión más moderna de Bluetooth, el error estimado se redujo en comparación a [11], lo que permite estimar un buen precedente, aludiendo a los nuevos avances que han ocurrido.

Posteriormente, se siguió avanzando con el Bluetooth 3.0 que mejoró la velocidad de transmisión. Más aun, el verdadero cambio se originó con el lanzamiento de la versión 4.0 (cotidianamente conocida como "*Smart*" o "*Smart Ready*") que permitió aumentar de gran manera la velocidad de transmisión de datos concretos (en esta versión aún no se consiguieron cambios de consumo de energía al enviar audio, por lo que los audífonos o parlantes con Bluetooth continuaron con Bluetooth clásico) y la cobertura, disminuyendo a su vez el consumo de energía.

Los cambios anteriormente mencionados permitieron una revolución en el área del internet de las cosas (IoT) y la domótica, pues agregando esta tecnología a sus productos, y gracias a la gran compatibilidad con Android (desde el lanzamiento del Bluetooth Smart se fue incluyendo en

todos los dispositivos de comunicación como smartphones, notebooks, etc.), es efectivo crear una red de control de diferentes productos como electrodomésticos o de verificación de seguridad.

En efecto, para la nueva tecnología se modificaron los protocolos y perfiles encargados de Escanear, Reconocer, Entablar comunicación y Envío de datos. Concretamente, para entablar la comunicación se tienen diferentes capas de funcionamiento (Figura 2-2), entre las que se identifican capas físicas (como las antenas, los moduladores y demoduladores, entre otras) y capas virtuales; como el protocolo de control lógico y adaptación de enlace (L2CAP) que se divide en el protocolo de atributo (ATT) que está relacionado con la organización de los datos presentes en la comunicación (permisos de escritura o lectura, identificadores, etc.), y en el protocolo de gestión de la seguridad (SMP) que proporciona el marco para generar y distribuir las claves de seguridad entre pares.

Además, se creó un sistema de perfiles transversales a todas las capas (Figura 2-2) con el fin de facilitar información y direcciones. El más importante es el perfil genérico de acceso (GAP), que entrega los lineamientos generales de conexión, detección y difusión, para asegurar la interconectividad entre dispositivos de diferentes vendedores. A continuación, se tienen los perfiles genéricos de atributo (GATT), los que tienen un protocolo similar al ATT, pero que permiten construir cadenas de información y características de control. Se debe agregar que estos perfiles para el Bluetooth clásico eran de carácter general y de difícil modificación, por ello se cambió su programa para crear perfiles más específicos, ayudando a la eficiencia y al ahorro energético.

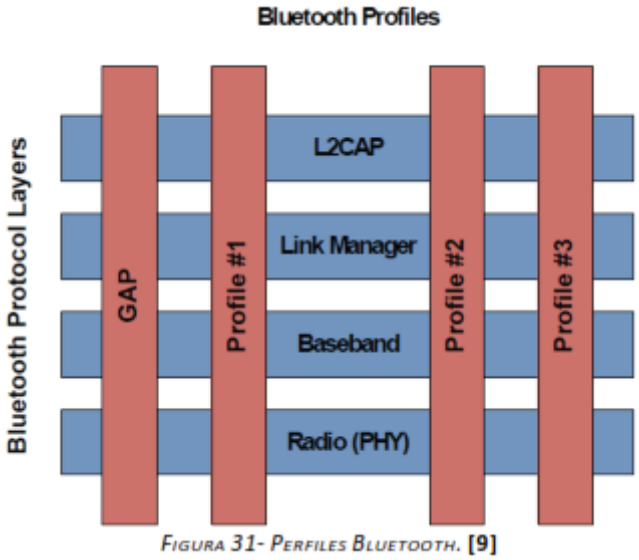

FIGURA 31- PERFILES BLUETOOTH. [9]

Figura 2-2 Protocolos comunicación Bluetooth 4.0 [28]

2.3 Técnica de estimación de posición

2.3.1 Técnica de Localización: RSSI.

Con respecto a la estimación de la distancia a raíz del RSSI, se propone en [12] un modelo de propagación considerando el efecto de la absorción y difracción en la señal transmitida, el que se rige por:

$$RSSI = -(10 \cdot n\, log_{10}\, d + A) \tag{2-1}$$

Donde A representa el nivel de RSSI a una distancia emisor-receptor de 1 metro, d se relación con la distancia emisor-receptor y n representa una constante propia del medio de transmisión.

Previo a la estimación de la distancia del robot en la escena, es necesario definir A y n, que son propios de los nodos referenciales. En este cauce, el primer paso es detectar el RSSI de cada nodo referencial a 1 metro de distancia para calcular A; luego, se volverá a realizar el mismo proceso de A, pero a diferentes distancias preestablecidas que serán analizadas en la siguiente relación:

$$n = \frac{1}{M} \cdot \sum_{i=1}^{M} \frac{E(RSSI_i) - A}{10 \cdot \log_{10} d_i} \tag{2-2}$$

$$E(RSSI_i) = \frac{1}{N} \cdot \sum_{f=1}^{N} RSSI_{fi}$$

Donde M indica la cantidad total de nodos referenciales a utilizar, N representa la cantidad de muestras a almacenar por cada posición de cada nodo, $E(RSSI_i)$ indica el promedio de las N muestras de intensidad medida en cada nodo i.

2.3.2 Corrección de ruido gaussiano

Partiendo del prisma de que el ruido ambiental generalmente se clasifica como una distribución gaussiana. Esto nos conduce a replantear la variable medida de RSSI añadiendo una distribución gaussiana normalizada P(x) (ya que es necesario referenciar respecto del máximo valor de las muestras) de la señal para n muestras de RSSI:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_{RSSI\, i}$$

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_{RSSI\, i} - \mu)^2$$

$$P(x) = e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{2-3}$$

Posteriormente se define el criterio de aceptación de valor de medida según $0.8 \leq P(x) \leq 1$, con el fin de que la muestra sea lo más estable posible.

2.3.3 Trilateración

Una vez obtenidos los valores de n y A, se reservarán estos valores en cada nodo referencial y serán comunicados al dispositivo a detectar, con la finalidad de estimar la posición según la ecuación (2-1). El siguiente punto a tratar será apreciar la posición acorde con el método de trilateración (capítulo 1.4.1) en dos dimensiones mediante la ecuación (1-2), donde x_i e y_i representan la ubicación del beacon i (con i número de beacons) y m_i la distancia estimada entre el robot a localizar y cada beacon, adjuntando el método de mínimos cuadrados, buscando una aproximación más exacta.

Para implementar este sistema de manera computacional, se trabajará en formato matricial, por lo que es necesario replantear el sistema según lo siguiente:

$$x' = x - x_1 \tag{2-4}$$

$$y' = y - y_1$$

$$x_i' = x_i - x_1 \qquad cuando\ i = 2,3,\dots,n$$

$$y_i{}' = y_i - y_1$$

$$\begin{bmatrix} x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} * \begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{1}{2} * \begin{bmatrix} {m_1}^2 - {m_2}^2 + {x_2'}^2 + {y_2'}^2 \\ \vdots \\ {m_1}^2 - {m_n}^2 + {x_n'}^2 + {y_n'}^2 \end{bmatrix}$$

2.3.4 Método de Mínimos Cuadrados

Como se introdujo en el capítulo 1.5, producto del error presente en la obtención de la distancia entre cada beacon y la referencia, se ve necesario el uso de técnicas de apoyo para la obtención de la posición. En esta oportunidad producto de la no disponibilidad de mayor equipamiento de beacons, se dio preferencia al método de mínimos cuadrados (1.5.2) sobre el método del centroide (1.5.3).

$$X = (A^T A)^{-1} A^T Y \tag{2-5}$$

2.4 Robot Khepera

Por lo que se refiere a las especificaciones del robot Khepera IV, disponible en las dependencias del laboratorio de robótica, corresponde a la penúltima versión desarrollada por la compañía K-Team para este modelo, posee un procesador Linux 800MHz ARM Cortex-A8 *Processor* y un micro controlador para el manejo de periféricos. Cuenta, además, con una memoria RAM de 512 MB, con una memoria flash de 512 MB y con la posibilidad de ser ampliada a 4 GB para datos. Así mismo, para su movilidad posee 2 motores DC y una amplia gama de sensores de proximidad (infrarrojos, ultrasonido, acelerómetros y giroscopios). En lo relativo a su comunicación, presenta dos puertos USB 2.0, una antena Wifi 802.11 b/g y una antena Bluetooth 2.0 EDR, además de la posibilidad de conectar módulos extras para ampliar sus capacidades. [29]

Figura 2-3 Khepera IV (Fuente: www.generationrobots.com)

Fundado en las especificaciones del Khepera IV, es posible deducir que el modelo del Bluetooth podría limitar la obtención del parámetro RSSI. En este orden de ideas, el primer trabajo relacionado con el robot fue la realización de un estudio acerca de su capacidad para reconocer dispositivos externos de comunicación buscando agregar un dispositivo USB Bluetooth de la versión 4.0, común en el mercado [30]. En concreto, el Hardware y sistema operativo es Gumstix Overo FireSTORM COM, el SO es una versión de Ubuntu GNU/Linux (del que se profundizará en el siguiente capítulo), por lo que considerando la fecha de adquisición de los robots del laboratorio (aproximadamente 2014) se hizo necesario analizar la versión del sistema con el fin de detectar la compatibilidad de drivers en la detección de dispositivos mediante puertos USB.

Acerca de la prueba de detección de dispositivos por vía USB, se utilizó un adaptador inalámbrico Wifi de posesión personal adquirido en el año 2013 [31]. En concreto, analizando los datos que se pueden obtener a través de comandos (dmesg output) fue posible conocer que el dispositivo estaba conectado al robot. Sin embargo, no fue viable la consecución del driver asociado para poder ser utilizado. Prosiguiendo en la búsqueda de una solución a este dilema, se verificó que el sistema estaba desactualizado y que era necesario actualizar el kernel del sistema operativo. Del manual [32] se alcanzó la forma de actualizar el kernel, no obstante, sorprendentemente al no tener éxito siguiendo los pasos recomendados, se procedió a comunicar con K-team en busca de ayuda, los que respondieron que en el año 2016 se cambió el hardware por lo que no es posible actualizar el kernel del robot. Ante ello, la única solución que se entregó fue adquirir uno nuevo. En resumidas cuentas, se optó por utilizar el computador pequeño Raspberry PI Zero W, buscando conseguir la posición y entregarla al robot de la forma más simple y ocupando el menor espacio posible.

2.5 Raspberry Pi Zero W

Raspberry Pi Zero W es un mini ordenador lanzado en 2017 como una actualización a la Raspberry Pi Zero en cuanto a conectividad, al agregar un chip de conectividad inalámbrica (resaltado a la derecha del procesador en la Figura 2-4) y la respectiva antena (resaltado bajo el procesador en la Figura 2-4) dotando al modelo de conexión Wi-Fi 802.11n y Bluetooth 4.0. No obstante ello, se conserva de la primera versión los demás componentes, a saber:

- CPU ARM Broadcom BCM2835 de un solo núcleo @ 1GHz ARM.
- 512 MB de RAM LPDDR2.
- Dos puertos micro-USB OTG
- Salida Micro-HDMI.
- Ranura para tarjeta Micro-SD.
- Conexión GPIO (es necesaria la colocación de pines para su utilización)

En términos de potencia de procesado, presenta una leve mejora en relación al robot Khepera. Así mismo, gracias a la actualización anteriormente mencionada, la versión de Bluetooth es compatible con la versión requerida para el sistema de localización a diseñar. Otro rasgo distintivo son sus dimensiones (65 mm x 30 mm x 5 mm) permitiendo su fácil acople al robot a localizar. En concreto, el sistema operativo a utilizar se denomina Raspbian, y consiste en un sistema basado en Linux acondicionado a las limitaciones de memoria de la placa y provisto de forma gratuita por los desarrolladores de Raspberry. El sistema operativo fue cargado en una tarjeta Micro SD de 16 Gb que, a su vez, sirve de dispositivo de memoria para la placa Raspberry pi.

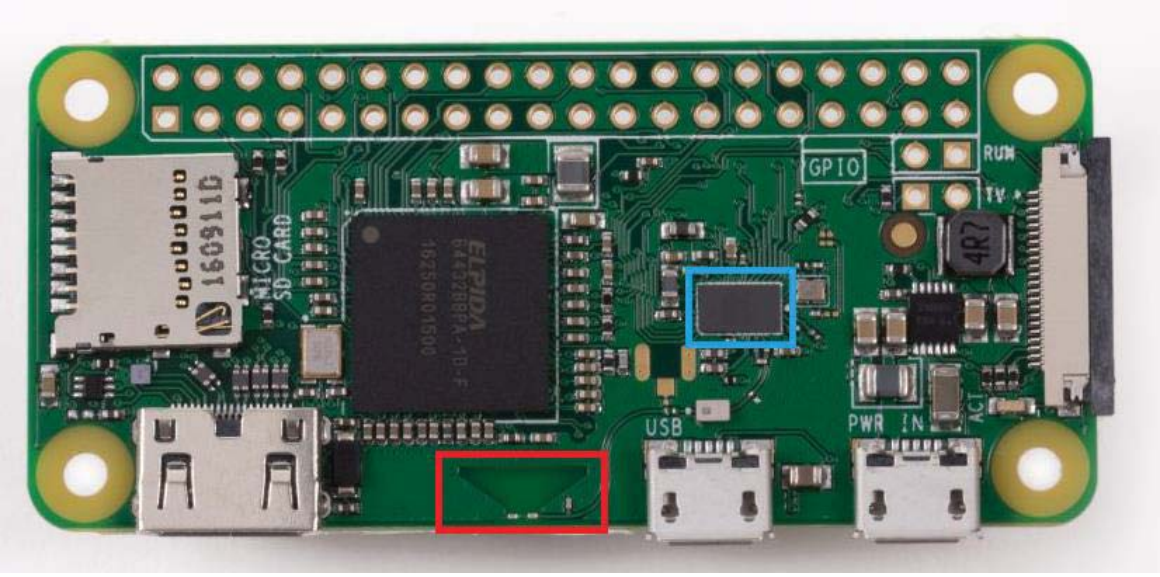

Figura 2-4 Raspberry Pi Zero W (fuente: www.raspberrypi.org)

Si bien la Raspberry presenta una buena conexión inalámbrica, al momento de implementar un sistema de localización, con base en la atenuación de su conexión, la ubicación de la antena

presentó problema, pues al realizar un giro los elementos adyacentes (Procesador, resistencias, conectores, etc) provocaron diversas atenuaciones extras no cuantificables. Frente a ello, resultó menester la búsqueda de una alternativa a la antena. Bajo esta premisa es posible centrarse en el conector libre que trae la placa (Véase Figura 2-5), que en un comienzo era utilizado como medio de diagnóstico de conexión para los fabricantes, pero que gracias a su utilidad no fue modificado.

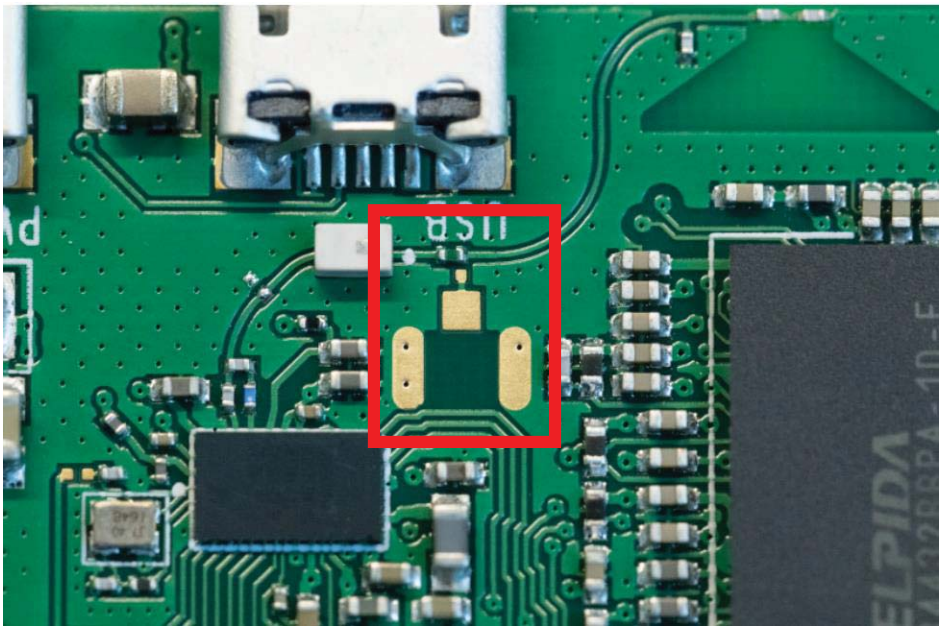

Figura 2-5 Conexión antena externa Raspberry pi zero https://www.briandorey.com

Para realizar la adaptación de la antena se realizó la adquisición de un conector u.fl, un cable con adaptación de u.fl a SMA y una antena omnidireccional apta para 2.4 GHz. En esta senda, el uso de la antena omnidireccional se justifica ya que para cualquier dirección en que se gire la antena (en el plano transversal) se detecta el mismo nivel de intensidad de la señal. La modificación realizada en la placa se observa en la Figura 2-6, donde se rotó la resistencia de 0 ohms (conector smd destacado en rojo) e inclusión del conector u.fl (destacado en color azul). Como extra se diseñó una base plástica como soporte de la antena y la placa Raspberry (Vease Figura 2-7 a.) con ayuda del software FreeCAD, concluyendo con el montaje de la placa Raspberry y la antena en la base impresa (Figura 2-7 b.). La base fue diseñada pensando en sacar el máximo partido al sistema y su inclusión en la parte superior del robot Khepera IV.

Al añadir una antena externa a la Raspberry pi zero, el consumo energético es superior, por lo que se debe tener precaución al momento de alimentar la placa. Pensando en la adquisición de una batería para alimentar el sistema sobre el robot Khepera se tendrá que considerar un mínimo de 3000 mAh.

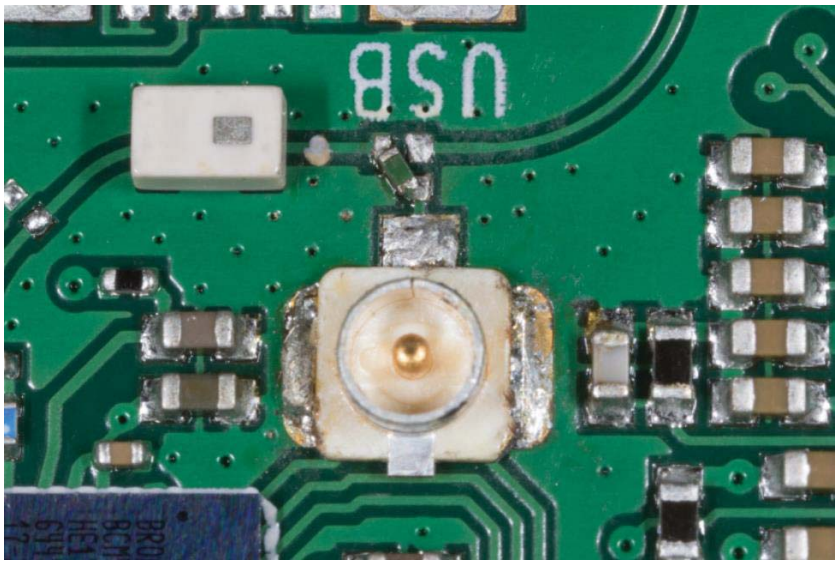

Figura 2-6 Modificación Placa Raspberry pi zero para conexión antena externa

Figura 2-7 a) Base plástica Raspberry-antena b) Montaje sistema Raspberry-antena

3 Prototipo software

Dando inicio a este tópico, se desarrollaron dos programas en el software de programación Python, presente de forma gratuita en el sistema operativo Ubuntu GBU/Linux (más conocido como Linux). Al mismo tiempo, como dispositivo a localizar, como se informó se tendrá el robot Khepera IV, donde la placa Raspberry Pi Zero W se ubicará en la parte superior. Además, se utilizarán celulares smartphones (Iphone 6, Iphone 6S, LG K10, LG Magna, Xperia M5, Samsung j6, Samsung j7, todos con bluetooth versión 4.0 o superior) junto con la aplicación Beacon Simulator (App gratuita que permite simular balizas independientes en la plataforma Android, esto es apreciable en la Figura 3-1) y Beacon sim (App gratuita para plataforma Apple).

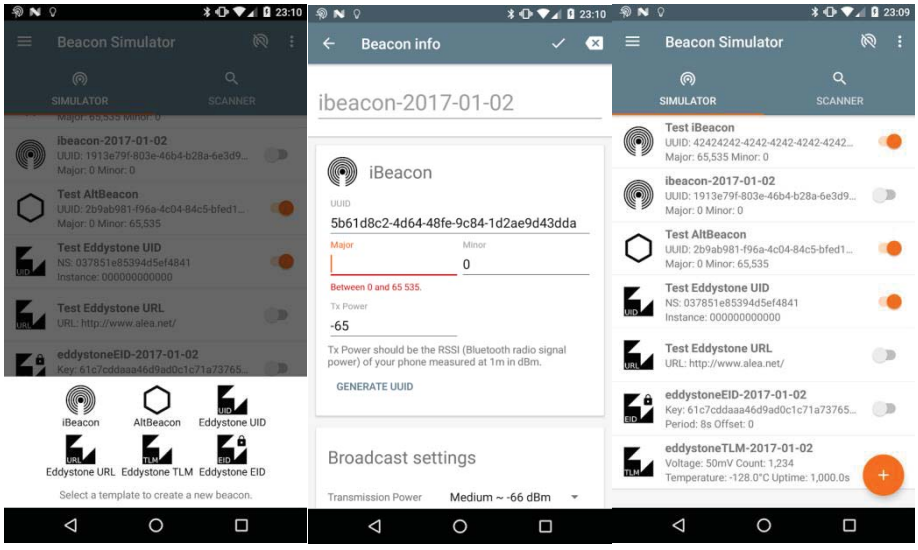

Figura 3-1 Interfaz App Beacon Simulator (Fuente: www.play.google.com)

Cabe destacar que cuando se implementó por primera vez el programa en la placa Raspberry, se probó alimentando con una batería externa de 1 [mA] y 5 [V], empero al observar que el valor de RSSI presentaba una gran variabilidad, se alternó por una conexión cableado de 2.4 [mA] y 5 [V] mejorando la obtención del RSSI. Luego al comprobar que el valor de medición era más estable, se observó que el sistema no funcionaba de manera correcta al girar la Raspberry en su eje, por lo que se añadió una antena externa.

Previo a la explicación del programa de búsqueda de posición, es necesario explayarse sobre el sistema operativo de Linux, ello pues, por un lado, es conveniente la realización del trabajo en este sistema; por el otro, para conseguir simular el funcionamiento de beacon para los dispositivos de referencia.

3.1 Ubuntu GBU/Linux

Es el sistema operativo por antonomasia en lo que a programación se refiere, en términos simples, es el intermediario del hardware del equipo y las aplicaciones que se busquen ejecutar, todo a través de líneas de comandos. Se caracteriza por ser un sistema operativo de código abierto y gratuito, contando con el plus de funcionar con base en paquetes que se han ido construyendo por los mismos usuarios que, a través de la comunicación de sus aplicaciones creadas, y por los diseñadores de dispositivos, mejoran y aportan nuevas formas de procesamiento u optimizan lo ya establecido. Precisamente, gracias a esta simpleza de programación, compilación y ejecución, es que se ha exportado a múltiples variaciones del mismo sistema operativo, [33] permitiendo ser añadidos en sistemas embebidos como lo son el robot Khepera IV, Raspberry PI, entre otros.

Ahora bien, el entorno de Linux es bastante amplio, por lo que es necesario acotarlo a lo que el proyecto amerita. Como se planteó al inicio del capítulo, el escenario de programación se centrará en Python, fundado en su fácil exportación a las distintas variaciones del sistema operativo y a la alta disponibilidad de códigos presentes en el mundo cibernético. Para estos efectos, la principal librería a utilizar es Bluez, cuyo uso se condice con la tecnología Bluetooth y cuya inclusión al mundo de Linux, prácticamente desde los inicios de la tecnología en el mundo del SO, permite un gran manejo de protocolos, perfiles, comunicación y manejo de tramas detectadas a través del Bluetooth. Así las cosas, se ha trabajado para mantenerlo vigente, lo que tiene como resultado una tecnología que soporta de buena forma las modificaciones en las versiones más modernas del Bluetooth, y que aprovecha las múltiples cualidades que entrega, como es la utilización en la localización. [34]

3.2 Beacons.

Los Beacons o balizas son dispositivos que funcionan mediante la tecnología Bluetooth, y que hacen referencia a dispositivos pequeños que emiten su señal de forma continua, con miras de utilizar la intensidad de esta señal, al ser detectada para la localización del que lo necesite. Además, imita el funcionamiento del GPS, pero a menor escala, con un bajo consumo de energía, y permite su localización en espacios pequeños.

Hay que mencionar, así mismo, que existen múltiples diseños de balizas, entre los que destacan iBeacon de la compañía Apple y Eddystone de Google. Ambos utilizan un patrón de comunicación similar asociado a los protocolos Bluetooth, donde resalta el patrón de identificación de la baliza y la intensidad de conexión (según el diseño y el uso de la baliza es posible asociar otros datos). En concreto, y acotando los modelos de balizas, para el proyecto se utilizará el patrón de los iBeacon.

Como instruye Wade Wegner [35], para la identificación de la baliza por el dispositivo se utiliza una dirección de 16 Bytes denominada UUID que permite distinguir entre un iBeacon y otro, además es posible configurar los iBeacon en grupos, determinando el conjunto a través de dos bytes (major); un último par de bytes se asocia al número del iBeacon (minor), para luego añadir a estos bytes de identificación, 2 bytes para comunicar el valor de la potencia de la comunicación (como el nivel de potencia se mide en [dBm] el número se arregla para transformarlo a hexadecimal). Precisamente, en la Figura 3-2 se observa el esquema general de una trama Bluetooth, donde los 22 bytes asociados al iBeacon se añaden al final de la data, y se incluyen otros datos como identificadores del productor, conjuntamente con datos necesarios y requeridos por los protocolos internos y/propios de la tecnología. Por último, es necesario recalcar que el sistema almacena la trama de forma inversa, ya que guarda al final los primeros datos, y al principio los últimos datos que recibe. Se apreciará en el programa que se trabajará, de forma inversa, parte de la trama con el fin de ahorrar código y tiempo.

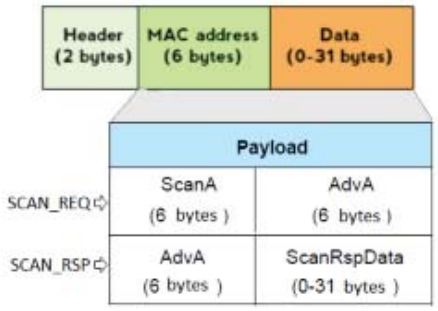

Figura 3-2 Paquete o trama Bluetooth [28]

3.3 Descripción Programa

El programa consta de dos partes, la primera consiste en la caracterización del ambiente a través de la realización de múltiples muestras de la señal (consiguiendo el valor del RSSI) para distancias conocidas de los nodos de referencias (beacons), y por medio de un proceso simple, conseguir el parámetro A y n necesarios para obtener la distancia respecto al RSSI. La segunda parte requiere del establecimiento de los nodos en sus respectivas posiciones para analizar las tramas y calcular la distancia respecto a los valores que registre en tiempo real. (véase Figura 3-3)

La base del programa se centra en el código de mylittle-domoticz/Presence-detection-beacon/test_beacon.py, que fue desarrollada por jmleglise, el 25 de mayo de 2016, siendo parte de un proyecto de domótica. En resumidas cuentas, el programa parte analizando si el dispositivo Bluetooth se encuentra disponible y luego, por intermedio de un loop identifica nuevos dispositivos, analizando las tramas de Bluetooth y para finalizar las trabaja para obtener la dirección MAC, el valor RSSI, el tiempo de detección y si contiene datos.

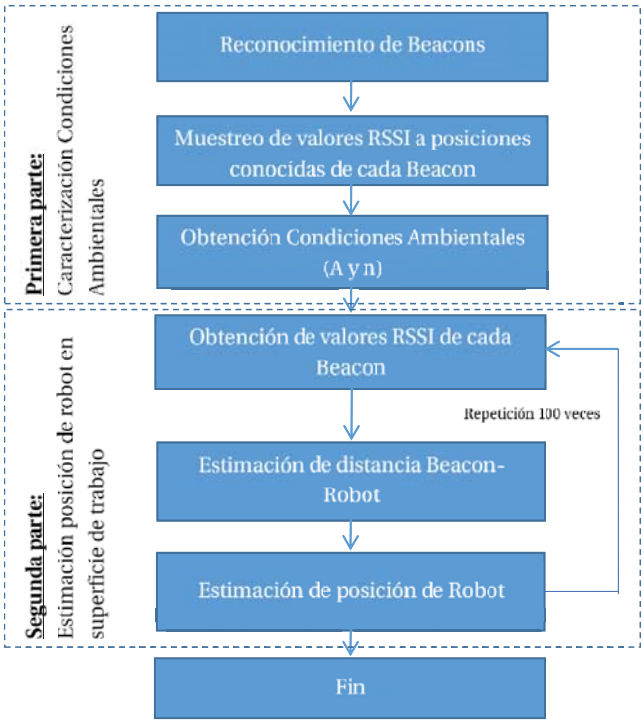

Figura 3-3 Diagrama Descripción de Programa

3.4 Caracterización de la señal

El programa da comienzo creando la matriz donde se almacenarán los datos de interés de los beacon a utilizar como referencia para el sistema en la segunda parte del programa, definiendo los encabezados con los que se reconocerá cada columna cuando se guarden los datos en el archivo tabla.csv (los encabezados son dirección UUID, minor, major, A, n, Posición X, Posición Y). Se declinó por este formato de archivo pues permite almacenar de manera rápida y ligera tablas con datos (almacenando números, letras y caracteres).

Prosiguiendo, realiza los llamados a la librería Bluez detectando la trama de Bluetooth que se encuentra disponible en ese momento, para luego filtrar si es una trama válida para conseguir los valores que se persiguen. Una vez la dirección es válida, para la obtención de datos se dispone de la función partida y la función seleccionar_MAC, las que en su inicio contienen una serie de códigos asociados para verificar que el dispositivo Bluetooth se encuentra activado y disponible para su utilización.

Específicamente, la función partida (Figura 3-4) comienza con la activación de Bluetooth, detección de trama y filtrado de la trama, siendo el objetivo de la función conseguir la dirección UUID en cada trama que detecte (la que se encontrará en los bytes del 19 al 35 –16 bytes–). Cabe destacar que en dispositivos que no son beacons, en este lugar de la trama, se entregará información que, para el caso de este trabajo, no son de importancia, por lo que regresará a la función cada vez que detecte una trama diferente a las anteriormente detectadas, almacenándose en la lista revisar, aumentando en último término la variable contador (determina el número total de beacons analizados).

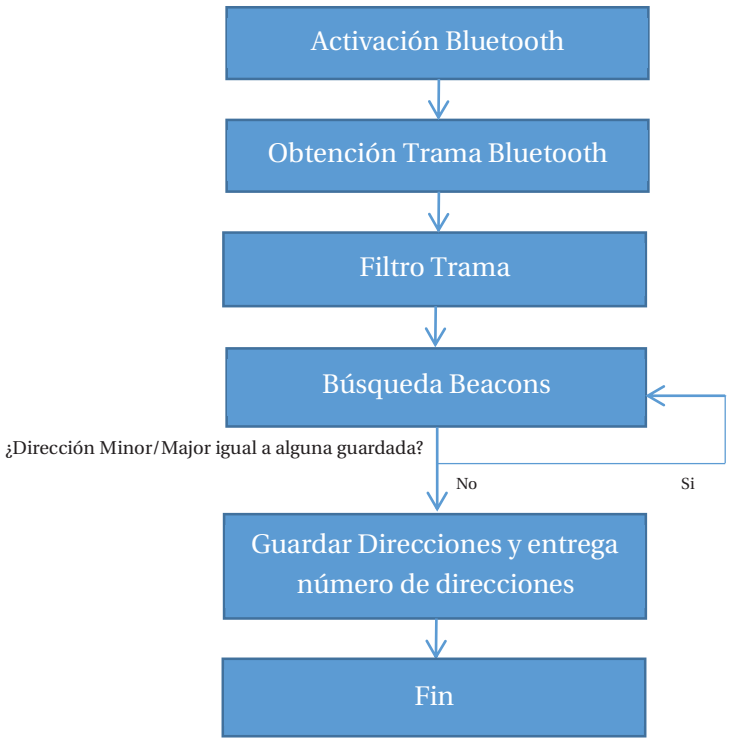

Figura 3-4 Función Partida

Al igual que la función anterior, la del seleccionar_MAC (Figura 3-5) comienza con la activación de Bluetooth, detección de trama y filtrado de la trama. Esta función necesita como dato de referencia la UUID de un dispositivo para entregar sus parámetros, consiguiéndose en la primera parte de la función de la trama los valores de MAC, UUID, minor, major y el RSSI. Siguiendo con la comparación, si el valor de UUID es igual al valor entregado antes de invocar la función, y si ha pasado un lapso pequeño de muestreo (se hace necesario este tiempo para permitir al Bluetooth se estabilice y entregue mejores valores), de ser acertado, almacena en la matriz MATRIZ y, una vez terminado el llenado de esta, promedia los valores almacenados, obtiene la desviación estándar y según la ecuación de distribución gaussiana (2-3) elimina los valores que sean menores que 0.8, finalizando con la entrega del promedio de la muestra filtrada al usuario.

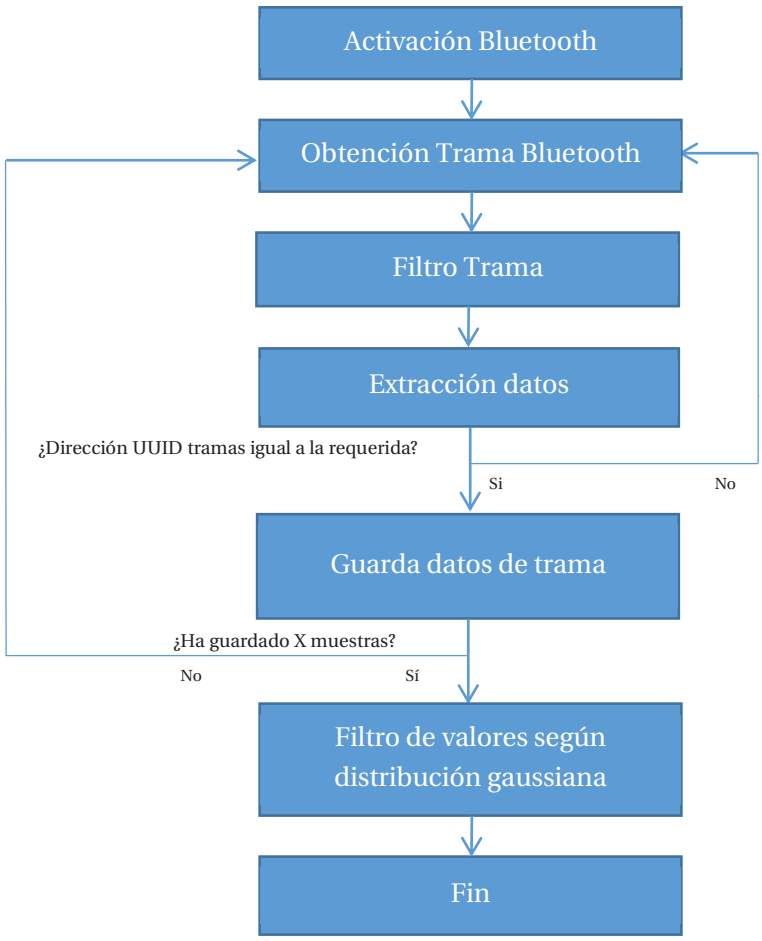

Figura 3-5 Función Seleccionar MAC

Como puede observarse en la Figura 3-6, el programa comienza invocando la función partida para posteriormente mostrarlas al momento de elegir la dirección (línea 14), a continuación, requiere el número de beacons que se utilizarán como referencia; seguido crea una matriz con el número de filas asociado al total de beacons y comienza el llenado de la matriz. Para cada beacon se pide ingresar la dirección y la posición física donde se ubicará, luego comienza con el valor de A (explicado en el capítulo anterior como el valor RSSI a 1 metro de distancia) invocando la función seleccionar_MAC. Consecutivamente para calcular el valor de n (factor que dimensiona como detecta el dispositivo el ambiente) pide ingresar el número de mediciones (mientras más muestras se realicen menor es el impacto del ruido) e invoca nuevamente a la función seleccionar_MAC para realizar el cálculo matemático para obtener n y, una vez finalizado el muestreo, promediar todos los n previstos. Concluida la obtención de A y n, almacena todos los datos en matriztotal e imprime en pantalla los valores. Ultima el programa guardando en archivo la matriz MATRIZTOTAL con todos los datos calculados precedentemente.

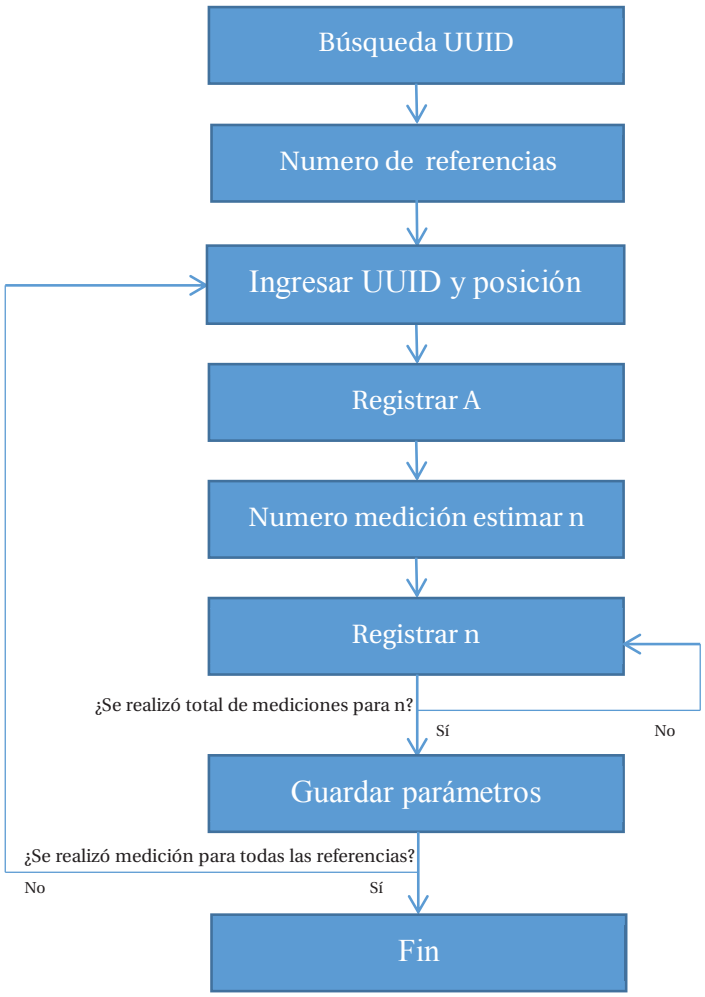

Figura 3-6 Estructura Programa Condiciones ambientales

3.5 Estimación posición

La segunda parte del programa (Figura 3-7) calcula en tiempo real la distancia entre dispositivos con base en los parámetros almacenados en la primera parte del programa. Para llevar acabo dicho objetivo, el primer paso es conseguir los valores guardados previamente para las condiciones ambientales, además de establecer la nueva matriz con datos importantes; como apoyo en la estimación de la. A continuación, el programa emula la función seleccionar_MAC, pero esta vez restringe el análisis de tramas a comparar con las direcciones de los beacons almacenados en el primer programa.

Al detectar una trama perteneciente a algún beacon se guarda en memoria el promedio filtrado según la distribución gaussiana de x muestras del valor de RSSI, es probable que en el momento de la muestra para algún equipo el filtro elimine todos los valores, por lo que se añade la condición previa a la estimación de distancia un número mínimo de valores válidos. Luego se calcula mediante la transformación matemática del valor RSSI en distancia para cada beacon previamente incluido. Además, previo a calcular el valor de las distancias, y guardar en una matriz

estos valores, se añadió una condición que filtra el valor de la posición acotándola a la superficie que abarca el sistema.

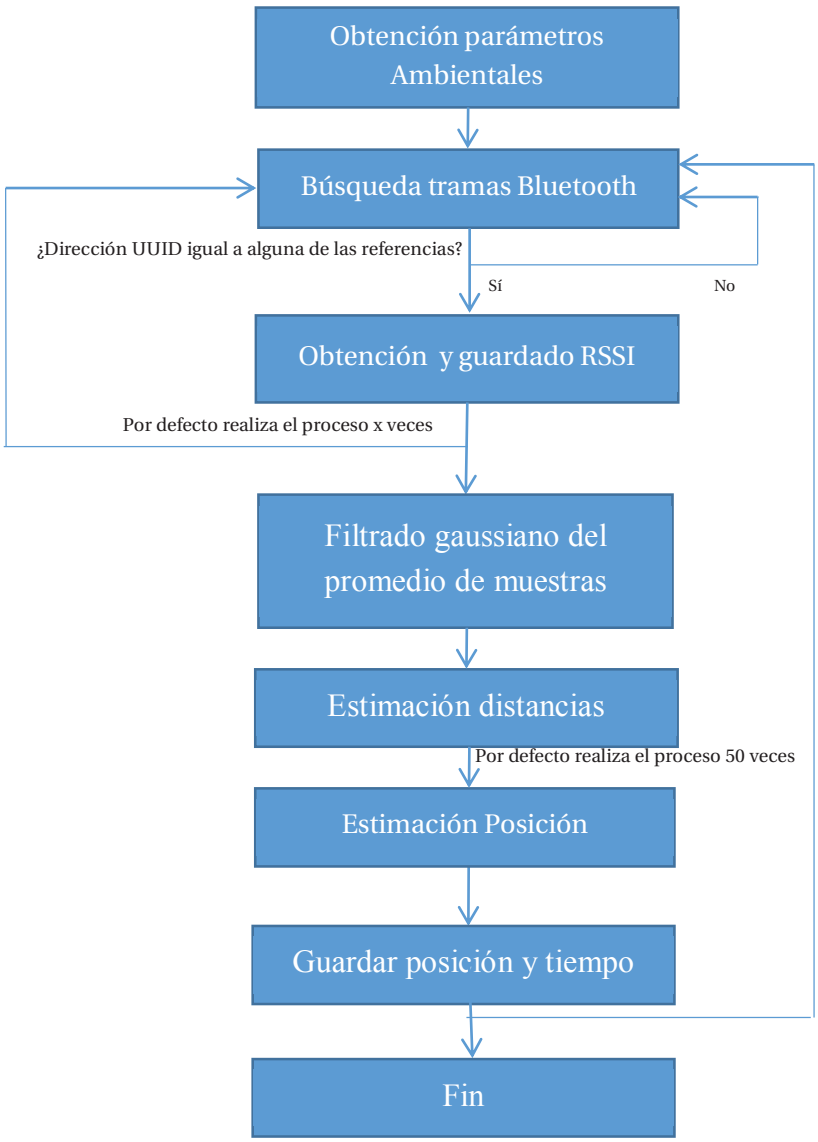

Figura 3-7 Estructura Programa Distancia

Una vez que se consigue la matriz de distancias, se planteó un apartado de estimación de posición mediante el método de triangulación, por lo que se diseñó una subrutina para la solución del sistema de ecuaciones para el método de trilateralización (planteado en el capítulo 2.3), mediante la adecuación al sistema de matrices planteado en la ecuación (1-6) propuesta en el capítulo 2.3, pensando en la futura implementación de algoritmos para la estimación de la posición del robot.

4 Resultados Experimentales

Para las mediciones, como se informó, se utilizó como objeto a localizar la placa Raspberry Pi Zero W modificada controlada a través de Wifi con un notebook modelo hp envy 15 j108la, los celulares smartphone que fueron nombrados a comienzos del capítulo. Donde se realizaron pruebas para analizar el desempeño de estos equipos en la localización, variando de 3 dispositivos a los 7 celulares nombrados.

4.1 Efecto filtro gaussiano

Una primera prueba se realizó con el teléfono Xperia y la placa Raspberry, con el fin de analizar el funcionamiento del filtro a través de distribución gaussiana (capítulo 2.3.2) en la búsqueda del aumento de la estabilidad de la medición, para la medición se desplazó la placa desde 10 centímetros a 200 centímetros de distancia del celular Xperia con variaciones de 10 centímetros (siguiendo Figura 4-1), realizando 10 muestras de RSSI en cada posición. Se filtró cada muestra según el modelo gaussiano. Luego la medición a 1 metro se asignó como A según el modelo de RSSI (2-1), y se calculó el valor de n para cada posición, una vez calculadas las condiciones ambientales, se comparó el modelo matemático asumiendo mismas lecturas de RSSI.

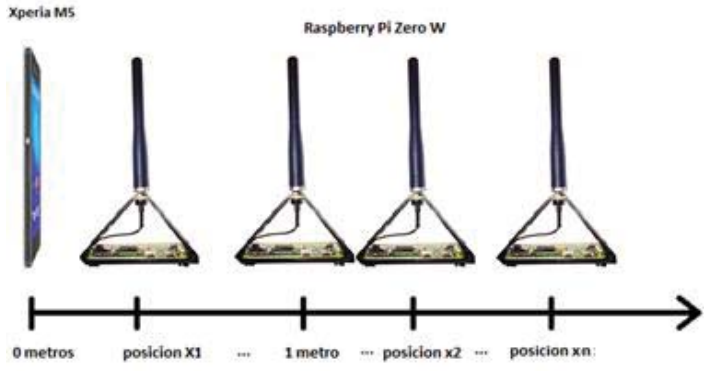

Figura 4-1 Prueba filtro gaussiano

Como podemos observar en Figura 4-2 la oscilación presentada por la curva del promedio de la variable medida sin filtrar es mayor que la oscilación de la curva del promedio de la variable filtrada, por lo que responde favorablemente al objetivo perseguido en la utilización del filtraje,

lamentablemente se observa que para las distancias superior a 150 centímetros se presenta una oscilación que el filtraje no es capaz de corregir. Ahora bien si se observa Figura 4-3 el error en la estimación de la distancia según el n calculado para la muestra filtrada hasta los 150 centímetros se mantiene menor a 20 centímetros, en comparación a la estimación de la distancia según el n calculado para la muestra sin filtrar que supera en algunos casos los 60 centímetros, además se observa que para distancias superiores a 150 centímetros el error se dispara producto de la oscilación en la medición.

Como conclusión se deduce que en el proceso de estimación de A y n es necesario realizar un muestreo amplio en distancia, con el fin de analizar si en algún tramo la señal se ve influenciada por algún fenómeno como las multitrayectorias.

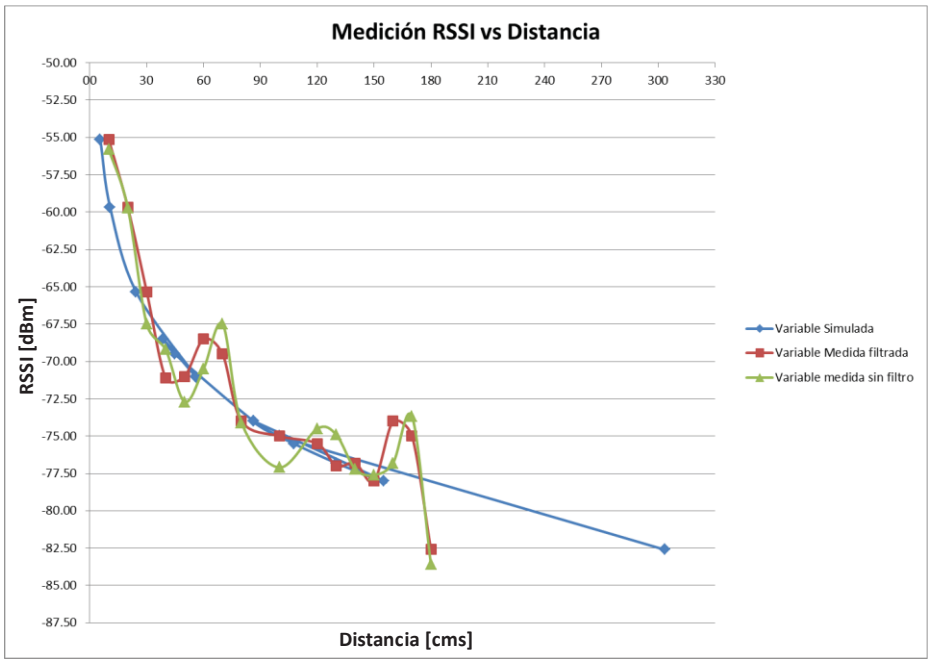

Figura 4-2 Comparación RSSI medido vs RSSI estimado

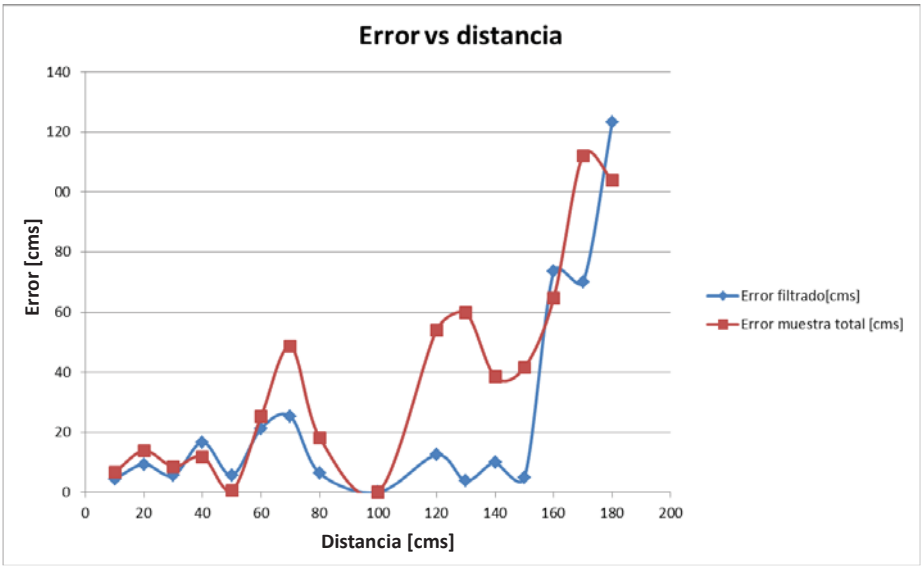

Figura 4-3 Error en centímetros variable medida filtrada vs variable medida no filtrada

4.2 Prueba Estimación Posición 3 Dispositivos

Los dispositivos se dispusieron en el laboratorio de robótica, específicamente en la plataforma asignada para el manejo de los robots (de dimensiones 2 metros de largo y 1.5 metros de ancho), donde se realizaron pruebas con 3 equipos (Xperia M5, Iphone 6 y LG K10) y 6 dispositivos (Iphone 6, Iphone 6S, LG K10, Xperia M5, Samsung j6, Samsung j7). Para la primera etapa del programa (obtención de condiciones ambientales), se dispuso cada Smartphone como se observa en la Figura 4-4, y se desplazó la placa Raspberry en distancias conocidas (30 [cm], 50 [cm] ,130 [cm], 180 [cm]); así mismo, para determinar n en cada beacon se realizaron 2 muestras en cada posición, destacando que en cada medición de RSSI se registraron 30 valores que fueron posteriormente filtrados y se estimaron 25 posiciones en cada experimento.

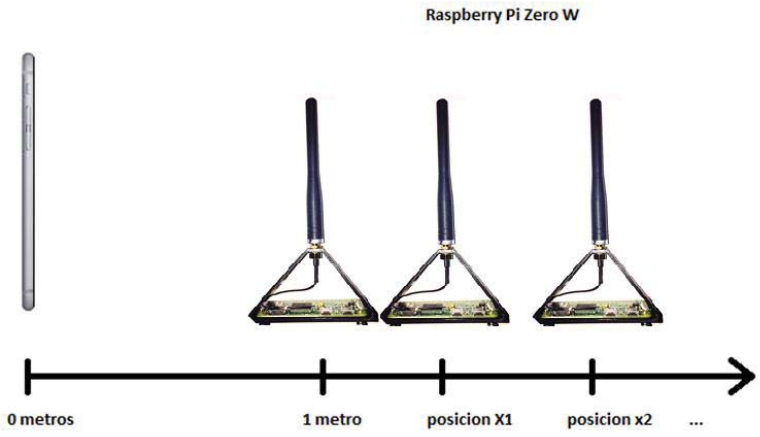

Figura 4-4 Medición distancia

Como se observa en el esquema en la Figura 4-5, el primer experimento utiliza tres dispositivos con el fin de analizar el peor caso (mínimo número de referencias para estimar posición, en una superficie de 1.6 metros por 1.2 metros). Se ubicó cada celular en una esquina del perímetro presente en la Figura 4-5 (se declaró que el celular marca Apple estaría en la coordenada (0 [cm], 0 [cm]), el celular marca Sony en (160 [cm], 0 [cm]) y el teléfono LG en (0 [cm], 120 [cm])), luego se dispuso la Raspberry Pi entre los dispositivos para realizar el muestreo de distancia de forma estacionaria en las coordenadas (70 [cm], 60 [cm]), (110 [cm], 0 [cm]) y (110 [cm], 100 [cm]). Como se observa en la Figura 4-6 y Figura 4-7 se entrega los valores resultantes de la estimación de la posición.

Al centrar la atención en el lugar 70,60, en ambas muestras se consigue que las mayor cantidad de las muestras está centrada en (80,60) por lo que el error respecto a la posición original del robot será aproximadamente de 10 centímetros, para las muestras en la posición (110,0), en ambas mediciones presenta una acumulación entorno a (100,0) por lo que el error también será aproximadamente de 10 centímetros, por último en la medición (110,100), se tiene que la dispersión en la estimación es demasiado alta para realizar algún análisis de error respecto a la posición real del robot. Si analizamos el promedio de la muestra total (presentes en Tabla 4-1 y Tabla 4-2), el error es aproximadamente 10 centímetros, por lo que se puede plantear la hipótesis de que un mayor números de dispositivos mejorara la dispersión y el error.

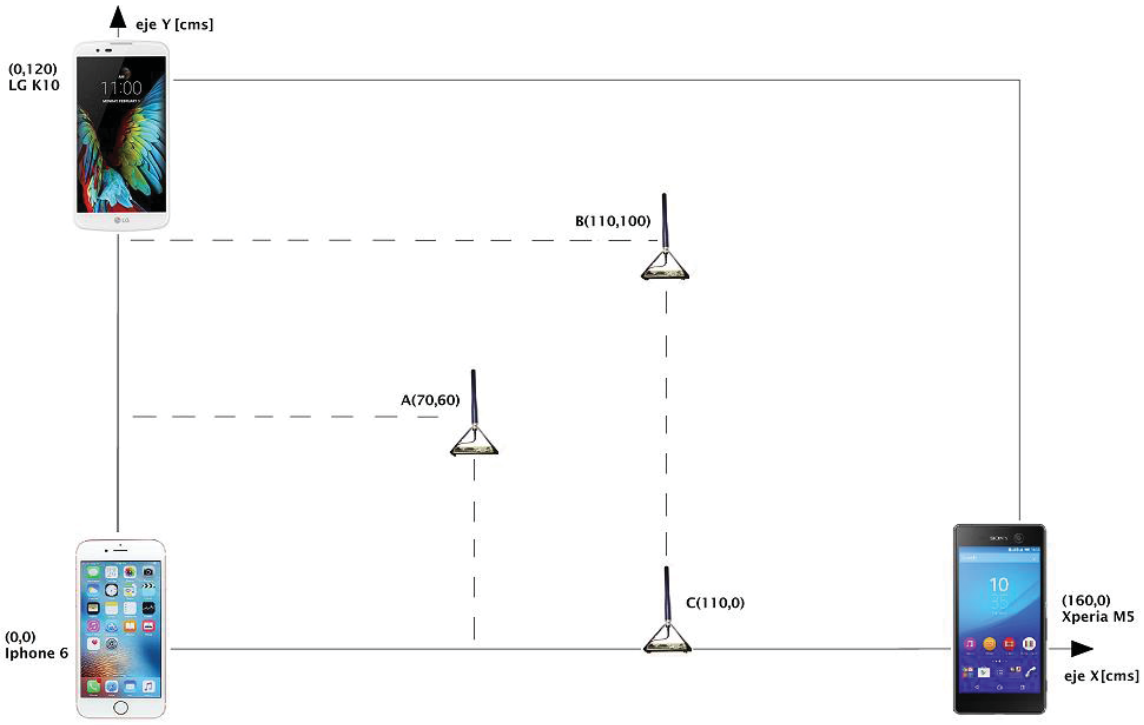

Figura 4-5 Planteamiento experimento versión 1 (3 dispositivos)

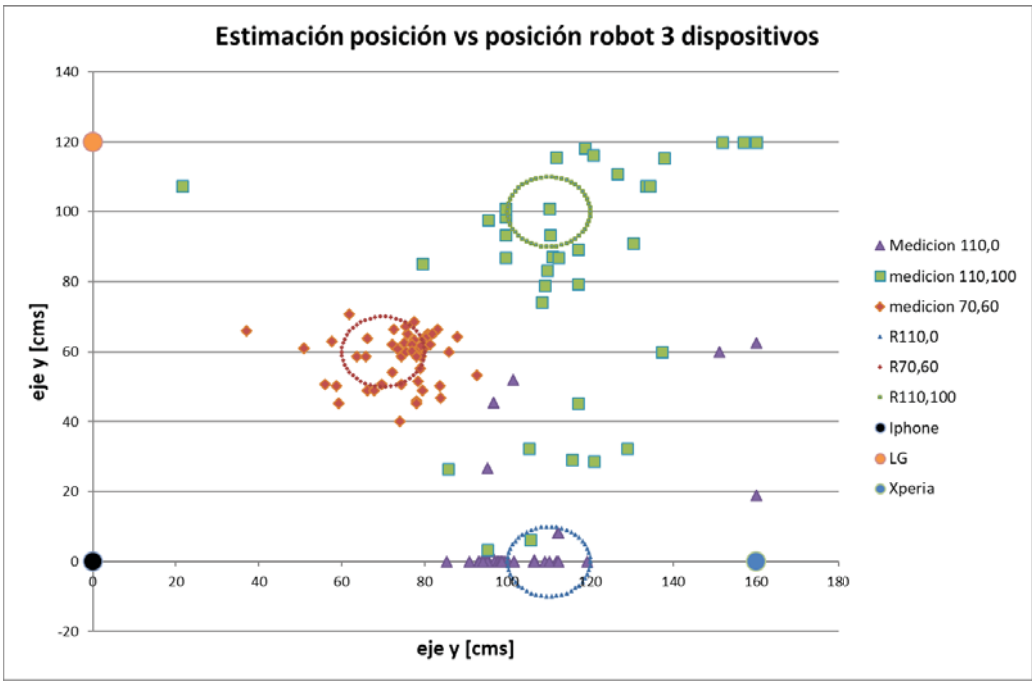

Figura 4-6 Estimación de posición robot para variaciones en su ubicación versión 1

Tabla 4-1 Valores promedio y desviación estándar posición y tiempo versión 1

Experimento	Medida X		Medida Y		Tiempo
	μ[cm]	σ[cm]	μ[cm]	σ[cm]	μ[s]
Medición 110,0	105	17	8	22	2.2
Medición 110,100	127	29	93	32	3.22
Medición 70,60	74	10	58	7	3.08

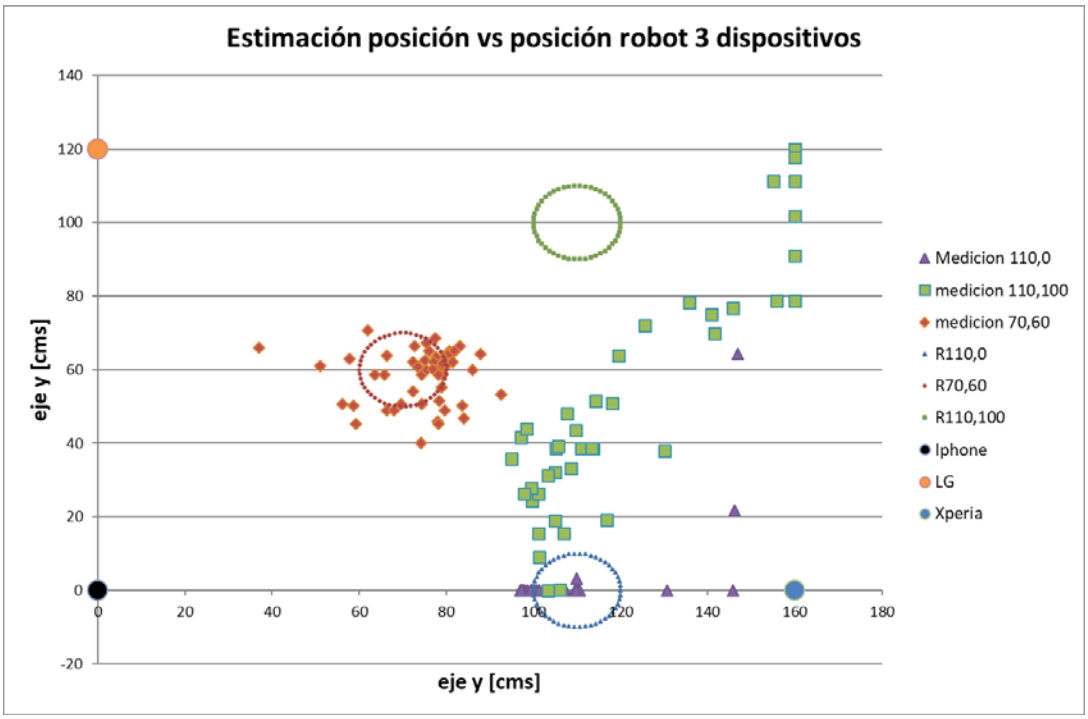

Figura 4-7 Estimación de posición robot para variaciones en su ubicación versión 1.1

Tabla 4-2 Valores promedio y desviación estándar posición y tiempo versión 1.1

Experimento	Medida X		Medida Y		Tiempo
	μ[cm]	σ[cm]	μ[cm]	σ[cm]	μ[s]
Medición 110,0	105	12	2	10	2.74
Medición 110,100	128	25	63	39	2.16
Medición 70,60	74	10	58	7	3.08

4.3 Prueba Estimación Posición 7 Dispositivos

Para la prueba realizada con 7 dispositivos dispuestos según la Figura 4-8, se realizaron cuatro tipos de mediciones, la primera permitiendo la detección de un mínimo de 4 dispositivos (luego de procesar las muestras en el filtro y un máximo de 6 dispositivos despreciando el celular LG Magna), la segunda con 6 dispositivos, la tercera medición con una detección mínima de 5 dispositivos (con un total de 7 dispositivos) y la última medición con el total de 7 dispositivos; cabe destacar además que para las primeras dos mediciones se utilizaron 6 muestras por dispositivo, y para las otras dos se utilizaron 7 y 9 mediciones respectivamente. En esta oportunidad debido al número de dispositivos se repartieron en una superficie de 1.7 metros de largo por 1.3 metros de ancho (según Tabla 4-3). En esta oportunidad se realizaron pruebas en las coordenadas (100,30), (100,80) y (40,80).

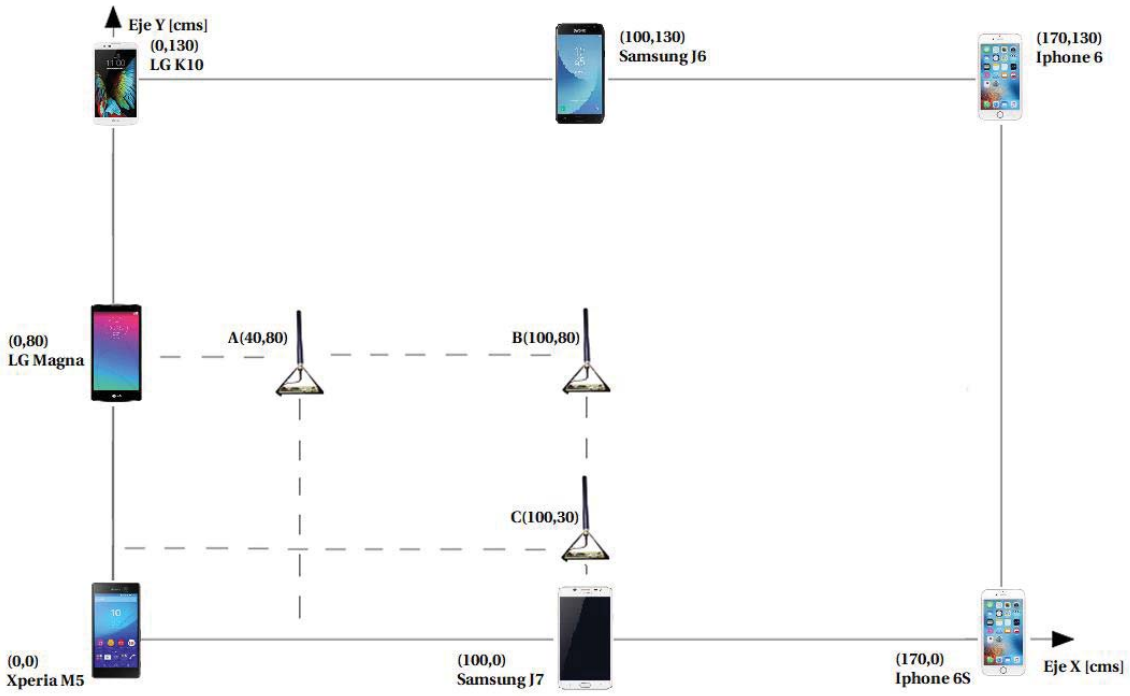

Figura 4-8 Disposición dispositivos en superficie experimento 2 (7 dispositivos)

Tabla 4-3 Disposición dispositivos en superficie experimento 2

Celulares	Ubicación equipos	
	x[cm]	y[cm]
Xperia M5	0	0
LG K10	0	130
Iphone 6	170	130
Iphone 6S	170	0
Samsung J6	100	130
Samsung j7	100	0
LG Magna	0	80

4.3.1 Prueba Detección mínima de 4 dispositivos

En la Figura 4-9, se presentan las estimaciones realizadas para una detección valida de valores RSSI filtrados mínima de 4 dispositivos y máxima de 6 dispositivos, registrando 6 valores de RSSI en cada referencia, además es posible percatarse a simple vista que al añadir un mayor número de dispositivos (en comparación con Figura 4-6 y Figura 4-7) la desviación estándar de la medición se reduce en gran medida, si se analiza la muestra 100,80 v1 y 100, 80 v2, se observa un comportamiento similar entre ellos (baja dispersión y un error respecto a la posición real cercano a los 20 centímetros), observando la posición 100,30 y 40,80 se observa que al desplazarse del centro en la dirección del eje y,(de 100,80 a 100,30) la existencia de los teléfonos ubicados en el centro (posición 100,0 y 100,130) favorecen a que la dispersión se mantenga constante, en cambio al desplazarse en dirección del eje X (de 100,80 a 40,80) la dispersión aumenta considerablemente.

Por ultimo en la Tabla 4-4, se resume el promedio, la desviación estándar y el error en el eje X y el eje Y, en esta tabla se destaca que gracias a la existencia de los celulares en la posición 100,0 y 100,130, la desviación en el eje X se encuentra en un rango aceptable en contraste de la desviación en el eje Y donde la desviación es mayor. Además, el impacto de la desviación se observa también en el tiempo entre estimación de posición, donde a medida que el robot se aleja del centro el tiempo aumenta.

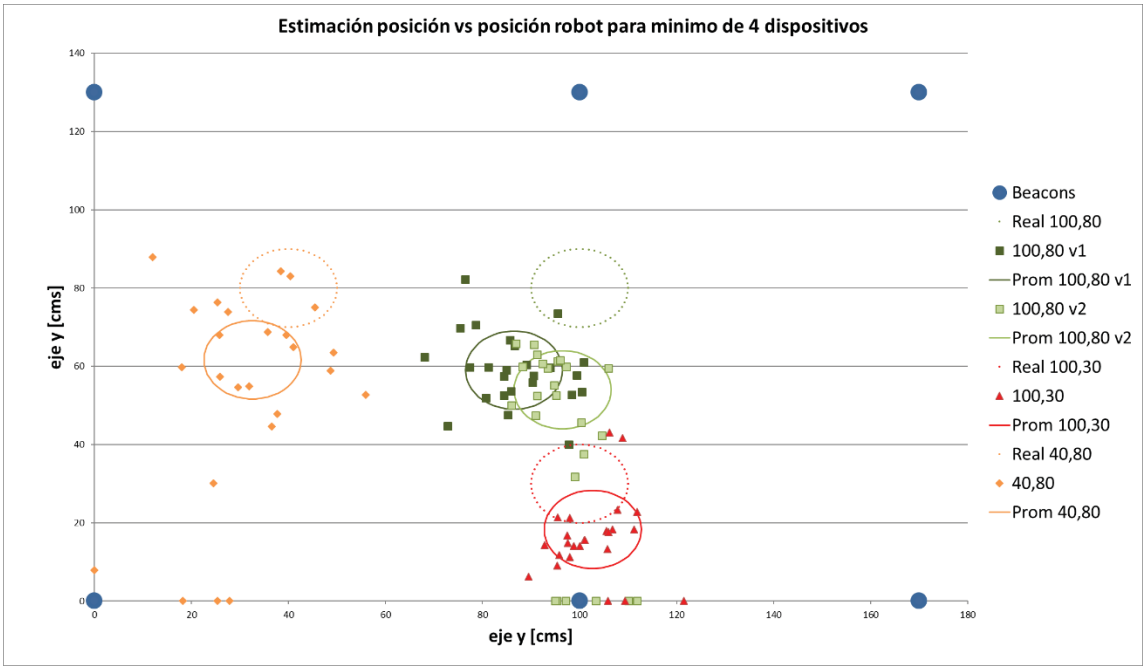

Figura 4-9 Estimación de posición robot para variaciones en su ubicación versión 2

Tabla 4-4 Valores promedio y desviación estándar posición y tiempo versión 2

Experimento	Medida X			Medida Y			Tiempo
	μ[cm]	σ[cm]	Error μ[cm]	μ[cm]	σ[cm]	Error μ[cm]	μ[s]
Medición 100,80	87	9	14	59	9	21	1.92
Medición 100,80	97	7	6	41	25	39	2.52
Medición 40,80	31	13	12	54	27	27	4.72
Medición 100,30	103	7	6	16	10	16	2.36

4.3.2 Prueba detección de 6 Dispositivos

En la Figura 4-10, se presenta la estimación de posición para 6 dispositivos, registrando 6 mediciones de RSSI en cada referencia, en paralelo es claramente apreciable que al utilizar el total de referencias disponibles, la desviación disminuye a la mitad en relación al eje X (donde se encuentran los equipos auxiliares) pero el tiempo necesario para estimar la posición aumenta en una proporción cercana. Otro dato destacable es que también es posible estimar valores más alejados al centro con una desviación estándar aceptable.

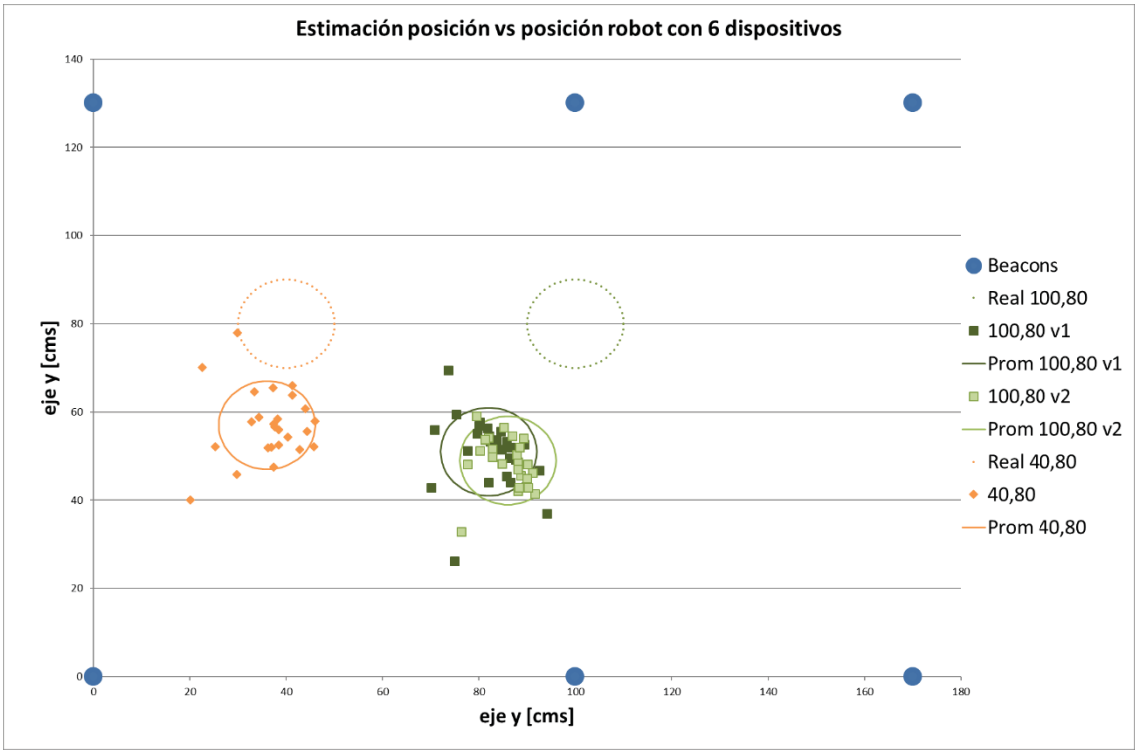

Figura 4-10 Estimación de posición robot para variaciones en su ubicación versión 2.2

Tabla 4-5 Valores promedio y desviación estándar posición y tiempo versión 2.2

Experimento	Medida X			Medida Y			Tiempo
	μ[cm]	σ[cm]	Error μ[cm]	μ[cm]	σ[cm]	Error μ[cm]	μ[s]
Medición 100,80	82	6	18	51	8	29	13.64
Medición 100,80	86	4	14	49	6	31	5.64
Medición 40,80	36	7	6	57	8	23	6.28

4.3.3 Prueba Comparación Variación Numero de Dispositivos

Con el propósito de comprobar el impacto del número de dispositivos y aumentar el número para el proceso de adquisición de valores RSSI, se realizó una segunda tanda de mediciones, esta vez con siete dispositivos, para el que se desarrollaron dos mediciones, la primera estimando la posición con un mínimo de 5 dispositivos validos luego del filtro, y la segunda con el total de referencias (7 dispositivos), además en esta oportunidad se buscó adquirir 9 muestras.

En la Figura 4-11, se agruparon las mediciones realizadas con el robot en la posición 100,80 para los cuatro casos medidos (mínimo de 4 dispositivos y 7 muestras; 5 dispositivos y 9 muestras; 6 dispositivos y 7 muestras; y dispositivos y 9 muestras). La primera observación que se observa es que al agregar un dispositivo en el eje Y (0,80) el error promedio respecto a la posición real se reduce, pero al ser solo un equipo la dispersión aumenta levemente en el eje X, pero se reduce en el eje Y. centrando la atención en la medición para 6 y 7 dispositivos se observa que al aumentar el número de valores de RSSI para estimar las distancias el tiempo de muestreo se reduce considerablemente, siendo comparable el tiempo promedio de 7 dispositivos con el de 4 dispositivos. Lamentablemente al realizar pruebas en los extremos de la superficie (cercano a un beacon) la medición de los beacons más lejanos presentaba demasiada variabilidad.

Analizando la información de la Tabla 4-6, al aumentar el numero de dispositivos el error de posición promedio se encuentra entre 6 y 17 centímetros, con un tiempo de respuesta cercano a los 3 segundos, lo que al ser comparado con el sistema de posición mediante procesamiento de imágenes, el sistema desarrollado mediante radiofrecuencia contiene una taza de error mayor pero esperable debido a la variabilidad de mediciones producto del ruido ambiental. Por otra parte, es necesario evaluar el código elaborado del programa y analizar el sistema de muestreo Bluetooth con el fin de reducir el tiempo de muestreo ya que es 10 veces mayor que el sistema desarrollado mediante procesamiento de imágenes.

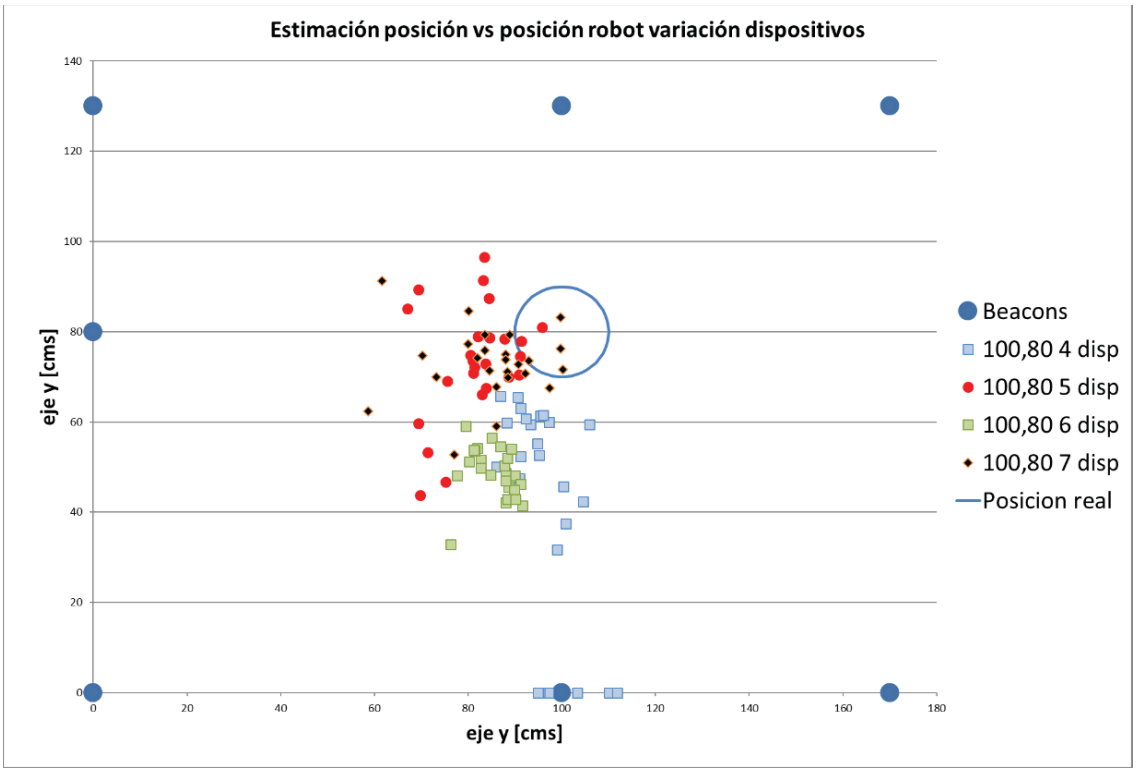

Figura 4-11 Grafico impacto número de dispositivos en medición

Tabla 4-6 Valores promedio y desviación estándar posición y tiempo comparación número de dispositivos

Experimento	Medida X			Medida Y			Tiempo
	μ[cm]	σ[cm]	Error μ[cm]	μ[cm]	σ[cm]	Error μ[cm]	μ[s]
100,80 4 disp.	97	6	7	41	39	25	2.52
100,80 5 disp.	81	19	8	73	11	13	15.36
100,80 6 disp.	86	14	4	49	31	6	5.64
100,80 7 disp.	85	15	11	73	9	8	2.92

Discusión y conclusiones

Arribados a este punto, el resultado de la investigación desarrollada se entrega al lector, junto con las técnicas empleadas en los principales sistemas creados para dar respuesta a la estimación de la localización de dispositivos, concretamente a través de radiofrecuencia; que van sufriendo ajustes en la medida que la misma tecnología sigue la línea de la progresión. Ello ha permitido, sin lugar a dudas, la elaboración de sistemas más exactos y con mejores respuestas a las variaciones en el entorno en que el robot se encuentra inmerso.

Ahora bien, como patrón de elección de una tecnología sobre otra, se hace necesaria la evaluación de los criterios del interesado, entre los se encuentran la disposición de capital, la precisión de posición, la disponibilidad de alimentación eléctrica, el alcance del sistema, el gasto computacional y el ambiente en el que se pretende realizar la implementación del sistema. A raíz de lo anterior, el método que ha demostrado mayor adaptabilidad al medio es la UWB pues, gracias a su funcionamiento, permite agilizar el procesamiento digital al entregar datos más confiables o con mayor información en contraposición al resto. Sin embargo, se ve mermada su elección por su baja disponibilidad en el mercado (mayor costo y espacio), frente a otras tecnologías como el Wifi o Bluetooth. Asimismo, es imperativo profundizar más en su desempeño frente a ambientes con un ruido mayor.

Así mismo, se demostró también que la tecnología Wifi permite entregar una respuesta decente a la localización, al contar con la opción de múltiples accesos. No obstante, se ve limitada su puesta en práctica fundado en el consumo de energía que precisa al ser solo utilizado como localizador, esto en relación a otras tecnologías. Otro punto de relevancia en contra de esta tecnología es la variación de potencia de los puntos de acceso en relación al número de dispositivos conectados. Constatado lo anterior, se arriba a la conclusión de que la tecnología más amigable en temas de costos y consumo es la tecnología Bluetooth la que, no siendo necesaria una estimación tan precisa, se convierte en un eficiente aliado teniendo en consideración los avances en su desarrollo. En resumidas cuentas, estas dos últimas tecnologías se ven favorablemente ampliada su utilidad gracias a la gran masificación y considerable disponibilidad que gozan en el mercado.

Paralelamente, una tecnología moderna que entregó buen desempeño en la puesta en marcha del presente estudio fue la tecnología RFID la que, no obstante, aún se encuentra en desarrollo y en investigación en entornos con ruidos u obstáculos. Hemos de destacar que, Junto con UWB, podrían representar una buena alternativa de masificación y estudio, al mejorar sus efectos y sus

respuestas. En esta línea, y otra forma de utilizar estos sistemas es desarrollarlos en conjunto, todo con el propósito de aprovechar las distintas cualidades que tienen y mejorar la respuesta del sistema entorno a perturbación u obstáculos.

Por consiguiente, habida consideración de lo expuesto anteriormente, en los primeros pasos del proyecto se obtuvo como resultado el desarrollo del programa base para la obtención de la distancia en la plataforma de Linux. Al utilizar Linux es posible exportar el programa a otros dispositivos que funcionan bajo sistemas derivados del sistema operativo de Linux realizando pequeños ajustes para la compatibilidad, esto se llevó a cabo al exportar el programa al mini computador Raspberry Pi Zero W. Como contrapartida de los logros alcanzados, a partir de la información con ahínco recaba, se enfrentaron retrasos debido a la capacitación sobre esta nueva plataforma, destacando el estudio de Raspberry Pi, del sistema operativo y las diversas opciones para lograr un manejo de forma remota y simple del programa inserto en el dispositivo.

En sus inicios, el programa demostró en este nuevo dispositivo una respuesta rápida en la estimación de la distancia. Sin embargo, al utilizar un dispositivo de menor potencia (cambiando de un notebook a Raspberry) se observó que la variación en la medición de RSSI aumentó. Además, fue posible visualizar la variación de posición del robot en una dimensión, aun con una tasa de error importante. Para este caso, como primera solución a esta tasa de error, se modificó la placa Raspberry adjuntando una antena externa omnidireccional, que permitió independizar el muestreo del valor RSSI de la orientación de la placa Raspberry; igualmente mejoró la tasa de error en algunos centímetros. Trabajado en una dimensión se buscó el desarrollo del programa ampliando el sistema a un plano de dos dimensiones, preparándolo para la inclusión de algoritmos con fuente en arreglos matriciales y de compensación de ruido.

Producto de la variación de la señal al momento de obtener muestras de los diferentes dispositivos, se decidió añadir un filtro de ruido asumiendo distribución gaussiana, dando como resultado la estabilización de la frecuencia de muestreo y reducción de la variabilidad en la obtención del valor RSSI derivado del ruido ambiental y de equipos no ideales, además se acompañó con el método de mínimos cuadrados, que da respuesta al cálculo del sistema de trilateralización para estimar la posición que en presencia de ruido falla en su objetivo de localizar al robot. Una vez desarrollado el software de estimación de posición, se realizaron pruebas variando el número de mediciones para estimar un óptimo en términos de velocidad de estimación de posición, el cual siendo superior a nueve e inferior a 15, mejoraba la acción del filtro permitiendo conseguir un muestreo aproximadamente constante.

Al variar el número de dispositivos para estimar la posición (desde 3 dispositivos en adelante), se consigue que la dispersión y el error de la posición estimada se reducen considerablemente, pero a su vez al utilizar un mayor número de dispositivos es necesario caracterizar más características ambientales, por lo que el factor humano se vuelve un agente de riesgo en la efectividad de la obtención de posición. Un factor clave es evaluar los límites del sistema, como por ejemplo cómo reacciona al aumentar la distancia, porque como se visualizó a medida que el dispositivo se alejaba de una referencia (smartphones), el modelo dejaba de ser logarítmico, este proceso se da debido a que el patrón de radiación de los celulares no es omnidireccional como lo es la antena

anexada a la placa Raspberry, por lo que se puede plantear que las mediciones y resultados anteriormente conseguidos son el peor caso que se puede conseguir para un sistema de localización, sin desprestigiar que los resultados conseguidos son cercanos al objetivo de estimar la posición acotado al tamaño del robot a localizar.

A partir de lo anterior, se concluye una primera etapa en el proyecto de localización de robots móviles en interior, siendo capaz de desarrollar un software que permite estimar la posición del dispositivo en cuestión de manera autónoma, independizándose de un sistema central, a través de balizas que serán utilizadas como referencias y que pueden ser tanto diseñadas como representadas por otros dispositivos como es el caso de los teléfonos, los que si bien son un buen instrumento de apoyo en la realización del software, es necesario evaluar más a fondo la variación del valor de RSSI respecto a la distancia, con el fin de adecuar el modelo planteado para la distancia en base a condiciones ambientales y el valor RSSI a las variaciones en la antena del dispositivo. Centrando la atención en el desarrollo de las balizas se abre una puerta con la modificación de la placa Raspberry al añadir una antena externa del tipo omnidireccional.

Por otra parte, se recalca la capacidad del código de poder ser modificado en su sistema de muestreo de señal, en el cálculo del modelo base para estimar distancia beacon-robot o para cambiar o mejorar el algoritmo de estimación de posición. Por último, en miras a mejorar el sistema de estimación de posición, se plantea la posibilidad de elaborar un algoritmo que permita actualizar las condiciones ambientales a medida que el tiempo avanza y sopesar los cambios en el espacio y probar el funcionamiento del sistema para seguir variaciones en tiempo real de la posición del robot, analizando la posibilidad de añadir un sistema que ajuste la dispersión de la estimación a la dirección real del robot.

Bibliografía

[1] W. B. Arthur y W. Polak, «The evolution of technology within a simple computer model,» *Complexity,* vol. 11, nº 5, pp. 22-31, 2006.

[2] L. Steels, «Evolving grounded communication for robots,» *Trends in Cognitive Sciences,* vol. 7, nº 7, pp. 308-312, 2003.

[3] N. Ravi, P. Shankar, A. Frankel, A. Elgammal y L. Iftode, «Indoor Localization Using Camera Phones,» de *Seventh IEEE Workshop on Mobile Computing Systems & Applications (WMCSA'06 Supplement)*, Orcas Island, WA, 2006.

[4] J. Z. Liang, N. Corso, E. Turner y A. Zakhor, «Reduced-complexity data acquisition system for image-based localization in indoor environments,» de *International Conference on Indoor Positioning and Indoor Navigation*, Montbeliard-Belfort, 2013.

[5] E. A. P. Hormazábal, «MODELADO, CONTROL Y SIMULACIÓN EN ROBÓTICA MÓVIL,» PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO, VALPARAISO, 2016.

[6] M. Skolnik, Radar Handbook, United States of America: Mc Graw Hill, 2008.

[7] E. García, «Técnicas de Localización en Redes inalámbricas de Sensores,» 2008.

[8] E. Kaplan y C. Hegarty, Understanding GPS: principles and applications, 2 ed., Norwood: Artech House, 2006.

[9] R. Kaune, «Accuracy Studies for TDOA and TOA Localization,» de *15th International Conference on Information Fusion*, Singapore, 2012.

[10] J. H. e. al, «On the Cramer-Rao Lower Bounds of Ranging Based on IR-UWB TOA Estimation in Wirelessbody Area Networks,» de *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, Francia, 2018.

[11] S. Zhou y J. K. Pollard, «Position measurement using Bluetooth,» *IEEE Transactions on Consumer Electronics,* vol. 52, nº 2, pp. 555-558, mayo 2006, DOI: 10.1109/TCE.2006.1649679.

[12] Y. Wang, X. Yang, Y. Zhao, Y. Liu y L. Cuthbert, «Bluetooth positioning using RSSI and triangulation methods,» de *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, 2013, doi: 10.1109/CCNC.2013.6488558.

[13] M. I. Jais, P. Ehkan, R. B. Ahmad, I. Ismail, T. Sabapathy y M. Jusoh, «Review of angle of arrival (AOA) estimations through received signal strength indication (RSSI) for wireless sensors network (WSN),» de *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*, Kuching, 2015.

[14] Q. H. Spencer, B. D. Jeffs, M. A. Jensen y A. L. Swindlehurst, «Modeling the statistical time and angle of arrival characteristics of an indoor multipath channel,» *IEEE Journal on Selected Areas in Communications,* vol. 18, nº 3, pp. 347-360, Marzo 2000, DOI: 10.1109/49.840194.

[15] J. Schilling, A. Albert, P. Biber y A. Graefenstein, «Wireless node localization based on RSSI using a rotating antenna on a mobile robot,» de *2009 6th Workshop on Positioning, Navigation and Communication*, Hannover, 2009.

[16] J. Xu, M. Ma y C. L. Law, «AOA Cooperative Position Localization,» de *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, New Orleans, 2008.

[17] Ahmed, Ismail y Islam, «Estimating DoA From Radio Frequency RSSI Measurements Using Multi-Element Femtocell Configuration,» *IEEE Sensors Journal,* vol. 15, nº 4, pp. 2087-2092, abril 2015, DOI: 10.1109/JSEN.2014.2371475.

[18] B. Lee, D. M. Woo y M. K. P. a. S. Kim, «Development of self-localizer using collaboration of trilateration and triangulation,» de *2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Xiamen, 2014.

[19] J. Zhang, R. A. Kennedy y T. D. Abhayapala, «Cramer-Rao lower bounds for the time delay estimation of UWB signals,» *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577),* vol. 6, pp. 3424-3428, junio 2004, doi: 10.1109/ICC.2004.1313180.

[20] S. M. Kay, « Fundamentals of statistical signal processing : estimation theory l,» de *Fundamentals of statistical signal processing : estimation theory l*, Englewood Cliffs, PTR Prentice-Hal, 1993, pp. 27-82.

[21] C. Yang y H. r. Shao, «WiFi-based indoor positioning,» *IEEE Communications Magazine,* vol. 53, nº 3, pp. 150-157, Marzo 2015, DOI: 10.1109/MCOM.2015.7060497.

[22] J. Biswas y M. Veloso, «WiFi localization and navigation for autonomous indoor mobile robots,» de *2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK, 2010.

[23] Y. Wang, S. Goddard y L. C. Perez, «A study on the cricket location-support system communication protocols,» de *2007 IEEE International Conference on Electro/Information Technology*, Chicago, IL, 2007.

[24] S. Zhou, H. Feng y R. Yuan, «Error Compensation for Cricket Indoor Location System,» de *2009 International Conference on Parallel and Distributed Computing, Applications and Technologies*, Higashi Hiroshima, 2009.

[25] D. Hahnel, W. Burgard, D. Fox, K. Fishkin y M. Philipose, «Mapping and localization with RFID technology," Robotics and Automation,» de *Robotics and Automation 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, New Orleans, LA, 2004.

[26] T. F. Kennedy, R. S. Provence, J. L. Broyan, P. W. Fink, P. H. Ngo y L. D. Rodriguez, «Topic models for RFID data modeling and localization,» de *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, 2017.

[27] A. Hossain y W. Soh, «A Comprehensive Study of Bluetooth Signal Parameters for Localization,» de *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, Athens, 2007.

[28] L. G. Gutierrez y F. F. Lanas, «Diseño y estudio de un sistema de comunicación inalámbrico basado en tecnología Bluetooth low energy con desarrolllo de protocolo propio de enrutamiento,» Pamplona, España, 2016.

[29] K-team. [En línea]. Available: https://www.k-team.com/mobile-robotics-products/khepera-iv/specifications.

[30] [En línea]. Available: https://www.pcfactory.cl/producto/17726-adaptador-usb-bluetooth-micro-4-0.

[31] tp-link. [En línea]. Available: https://www.tp-link.com/ar/products/details/TL-WN722N.html.

[32] K.team, *Khepera IV user manual 3.0,* 2017.

[33] M. J. Busta, Agosto 2015. [En línea]. Available: https://www.hostname.cl/blog/que-es-linux-y-como-funciona-en-tu-servidor-dedicado.

[34] M. Holtmann. [En línea]. Available: http://www.bluez.org/about/.

[35] W. Wegner, Mayo 2014. [En línea]. Available: http://www.wadewegner.com/2014/05/create-an-ibeacon-transmitter-with-the-raspberry-pi/.

[36] A. T. Council, Earthquake damage evaluation data for California. Technical report, Seismic Safety Commission, Applied Technology Council (ATC), California, 1995.

[37] S. Fingerhuth, «Integridad académica,» Pontificia Universidad Católica de Valparaíso, Julio 2014. [En línea]. Available: http://integridadacademica.cl/. [Último acceso: 20 10 2015].

[38] E. P. Wigner, «Theory of traveling wave optical laser,» *Phys. Rev.,* vol. 134, pp. A635-A646, 1965.

[39] M. Shell, «Preparation of papers for IEEE TRANSACTION and JOURNALS,» May 2007.

[40] A. G. Tsipkin, V. Vodnev, G. G. Tsipkin y A. I. Samojv, Fórmulas matemáticas: álgebra, geometría, análisis matemático., Mir, 1998.

[41] Oficina Internacional de Pesas y Medidas, «El Sistema Internacional de Unidades SI,» 2006. [En línea]. Available: http://www2.cem.es:8081/cem/es_ES/documentacion/generales/SIU8edes.pdf. [Último acceso: 2 January 2015].

[42] Alexander, C. K., Sadiku y M.N.O., Circuits, Fundamentals of Electric, McGraw-Hill College, 2003.

[43] R. K. Munro y E. T. J. v. Weert, «Informatics and the Digital Society: Social, ethical and cognitive issues: IFIP TC3/WG3.1&3.2 Open Conference on Social,» de *Ethical and Cognitive Issues of Informatics and ICT*, Dortmund, Germany, 2003.

A Tabla de Figuras

Figura 1-1 Plataforma con sistema de localización por cámara en laboratorio PUCV [5] 5
Figura 1-2 Tecnología radar (Fuente: www.radartutorial.eu) 6
Figura 1-3 Trabajo geométrico conversión ToA a AoA [16] 8
Figura 1-4 Técnica Trilateración de Posición (Fuente: www.iris.hdplus.es) 9
Figura 1-5 Error distancia robot-nodo referencial 10
Figura 1-6 Triangulación con 3 nodos de referencia [18] 11
Figura 1-7 Método del centroide [12] 13
Figura 1-8 Triangulación GPS [8] 15
Figura 1-9 Espectro de frecuencias señales inalámbricas más utilizadas (Fuente: www.eliko.ee) 16
Figura 1-10 Trasmisor-Receptor Cricket (Fuente: www.cricket.csail.mit.edu/) 18
Figura 2-1 Efecto GRPR sobre RSSI Bluetooth [11] 21
Figura 2-2 Protocolos comunicación Bluetooth 4.0 [28] 22
Figura 2-3 Khepera IV (Fuente: www.generationrobots.com) 25
Figura 2-4 Raspberry Pi Zero W (fuente: www.raspberrypi.org) 26
Figura 2-5 Conexión antena externa Raspberry pi zero https://www.briandorey.com 27
Figura 2-6 Modificación Placa Raspberry pi zero para conexión antena externa 28
Figura 2-7 a) Base plástica Raspberry-antena b) Montaje sistema Raspberry-antena 28
Figura 3-1 Interfaz App Beacon Simulator (Fuente: www.play.google.com) 29
Figura 3-2 Paquete o trama Bluetooth [28] 31
Figura 3-3 Diagrama Descripción de Programa 32
Figura 3-4 Función Partida 33
Figura 3-5 Función Seleccionar MAC 34
Figura 3-6 Estructura Programa Condiciones ambientales 35
Figura 3-7 Estructura Programa Distancia 36
Figura 4-1 Prueba filtro gaussiano 37
Figura 4-2 Comparación RSSI medido vs RSSI estimado 38
Figura 4-3 Error en centímetros variable medida filtrada vs variable medida no filtrada 39
Figura 4-4 Medición distancia 39
Figura 4-5 Planteamiento experimento versión 1 (3 dispositivos) 40
Figura 4-6 Estimación de posición robot para variaciones en su ubicación versión 1 41
Figura 4-7 Estimación de posición robot para variaciones en su ubicación versión 1.1 42

Figura 4-8 Disposición dispositivos en superficie experimento 2 (7 dispositivos) 43
Figura 4-9 Estimación de posición robot para variaciones en su ubicación versión 2 45
Figura 4-10 Estimación de posición robot para variaciones en su ubicación versión 2.2 46
Figura 4-11 Grafico impacto número de dispositivos en medición..... 48

B Codigos Programas Desarrollados

B.1 Condiciones Ambientales

```
1
2
3  TAG_DATA = [
4             ]
5
6
7  import logging
8
9
10 logLevel=logging.DEBUG
11 #logLevel=logging.CRITICAL
12
13 # --------------------------------------------------------------------------------
14 #                                   Librerias
15
16 import os
17 import subprocess
18 import sys
19 import struct
20 import bluetooth._bluetooth as bluez
21 import time
22 import requests
23 import signal
24 import threading
25 import math
26 import csv
27 # --------------------------------------------------------------------------------
28 #       Limpieza pantalla y creacion de carpeta para nuevas pruebas
29 os.system('clear')
30
31 #buscar carpeta
32
33
34 fecha= time.strftime(" %I:%M,%d_%b ", time.gmtime())
35 #Para ejecutar nueva medicion debe aumentar numeroprueba
36 numeroprueba=1
37 direccionprueba= r'/home/pi/Desktop/bluetooth/pruebas'+numeroprueba
38 nuevaruta =direccionprueba+'/muestra'+fecha
39 if not os.path.exists(nuevaruta): os.makedirs(nuevaruta)
40
41
42 # --------------------------------------------------------------------------------
43 #numero de muestras a utilizar y creacion de matriz para guardado en memoria de condiciones
44 ambientales
45
46 muestras=30
47 limite=5
48 headers = ('address','UUID','minormajor','A','n','posicion_x','posicion_y')
49 matriz = [range(4) for i in range(100)]
```

```
50 revisar=[range(1) for i in range(20)]
51 with open(nuevaruta+'/tabla.csv', 'w') as csvfile:
52     writer = csv.writer(csvfile)
53     writer.writerow(headers)
54
55 # --------------------------------------------------------------------------------
56 #           codigos coomparacion para deteccion de tramas bluetooth
57
58 LE_META_EVENT = 0x3e
59 OGF_LE_CTL=0x08
60 OCF_LE_SET_SCAN_ENABLE=0x000C
61 EVT_LE_CONN_COMPLETE=0x01
62 EVT_LE_ADVERTISING_REPORT=0x02
63
64 #---------------------------------------------------------------------------------
65 #           definicion estructura de codificacion trama bluetooth
66
67 def print_packet(pkt):
68     for c in pkt:
69         sys.stdout.write("%02x " % struct.unpack("B",c)[0])
70
71 def packed_bdaddr_to_string0(bdaddr_packed):
72     return ':'.join('%02x'%i for i in struct.unpack("<BBBB", bdaddr_packed[::-1]))
73
74 def packed_bdaddr_to_string(bdaddr_packed):
75     return ':'.join('%02x'%i for i in struct.unpack("<BBBBBB", bdaddr_packed[::-1]))
76
77 def packed_bdaddr_to_string1(bdaddr_packed):
78     return ' '.join('%02x'%i for i in struct.unpack("<BBBBBBBBBBBBBBBB", bdaddr_packed[::-
79 1]))
80
81 def hci_disable_le_scan(sock):
82     hci_toggle_le_scan(sock, 0x00)
83
84 def hci_toggle_le_scan(sock, enable):
85     cmd_pkt = struct.pack("<BB", enable, 0x00)
86     bluez.hci_send_cmd(sock, OGF_LE_CTL, OCF_LE_SET_SCAN_ENABLE, cmd_pkt)
87
88 def handler(signum = None, frame = None):
89     time.sleep(1)  #here check if process is done
90     sys.exit(0)
91
92 # --------------------------------------------------------------------------------
93 # comprobacion texto ingresado correctamente
94 def teclado():
95         comprobacion='0'
96         word=0
97         while(comprobacion!='y' and comprobacion!='Y'):
98                 word=raw_input()
99                 print ('are you sure (y/n):')
100                comprobacion=raw_input()
101        return word
102
103 # --------------------------------------------------------------------------------
104 #funcion para conseguir tramas acorde a una direccion previamente asignada
105
106 def seleccionar_MAC(IP):
107        IP=IP.lower()
108        if IP=='fail':
109                sys.exit()
110        #Reset Bluetooth interface, hci0
111        os.system("sudo hciconfig hci0 down")
112        os.system("sudo hciconfig hci0 up")
113
114        #Make sure device is up
115        interface    =    subprocess.Popen(["sudo    hciconfig"],    stdout=subprocess.PIPE,
116 shell=True)
117        (output, err) = interface.communicate()
118
119        if "RUNNING" in output: #Check return of hciconfig to make sure it's up
```

```
120                 a=1
121         #    logging.debug('Ok hci0 interface Up n running !')
122         else:
123             logging.critical('Error : hci0 interface not Running. Do you have a BLE device
124 connected to hci0 ? Check with hciconfig !')
125             sys.exit(1)
126 
127         devId = 0
128         try:
129             sock = bluez.hci_open_dev(devId)
130 
131             logging.debug('Connect to bluetooth device %i',devId)
132         except:
133             logging.critical('Unable to connect to bluetooth device...')
134             sys.exit(1)
135 
136         old_filter = sock.getsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, 14)
137         hci_toggle_le_scan(sock, 0x01)
138     # funcion detecta y filtra tramas incompletas
139         j=0
140         while (j<muestras+limite):
141             restriccion=0
142             while restriccion < 120:
143                     old_filter = sock.getsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, 14)
144                     flt = bluez.hci_filter_new()
145                     bluez.hci_filter_all_events(flt)
146                     bluez.hci_filter_set_ptype(flt, bluez.HCI_EVENT_PKT)
147                     sock.setsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, flt )
148 
149                     pkt = sock.recv(255)
150                     #revisar= ' '.join("{0:02x}".format(ord(x)) for x in pkt[39:35:-1])
151                     restriccion=' '.join("{0:02x}".format(ord(x)) for x in pkt[::-1])
152                     restriccion= len(restriccion)
153          #parametros revision bluetooth de trama valida
154             ptype, event, plen = struct.unpack("BBB", pkt[:3])
155 
156             if event == bluez.EVT_INQUIRY_RESULT_WITH_RSSI:
157                     i =0
158             elif event == bluez.EVT_NUM_COMP_PKTS:
159                     i =0
160             elif event == bluez.EVT_DISCONN_COMPLETE:
161                     i =0
162             elif event == LE_META_EVENT:
163                     subevent, = struct.unpack("B", pkt[3])
164                     pkt = pkt[4:]
165                     if subevent == EVT_LE_CONN_COMPLETE:
166                         le_handle_connection_complete(pkt)
167                     elif subevent == EVT_LE_ADVERTISING_REPORT:
168                         num_reports = struct.unpack("B", pkt[0])[0]
169                         report_pkt_offset = 0
170                         for i in range(0, num_reports):
171 
172 macAdressSeen=packed_bdaddr_to_string(pkt[report_pkt_offset + 3:report_pkt_offset + 9])
173                                     found=0
174                                     for tag in TAG_DATA:
175                                         if macAdressSeen.lower() == tag[1].lower():
176                                             elapsed_time=time.time()-tag[3]  # lastseen
177 
178                                             if elapsed_time>tag[2] :
179                                                 tag[2]=elapsed_time
180                                 #luego de despreciar primerar muestras, compara trama con
181 direccion asignada y guarda en memoria de ser valido
182                                             if j>=limite:
183 
184         ibeacon=packed_bdaddr_to_string1(pkt[report_pkt_offset  +  16:report_pkt_offset  +
185 32])
186 
187                                                 if ibeacon==IP:
188                                                         rssi=''.join(c     for     c     in
189 str(struct.unpack("b", pkt[report_pkt_offset -1])) if c in '-0123456789')
190                                                         #print ibeacon
```

```
191
192 mima=packed_bdaddr_to_string0(pkt[report_pkt_offset + 32:report_pkt_offset + 36])
193                                                     j=j-limite
194                                                     matriz[j][3] = mima
195                                                     matriz[j][2] = ibeacon
196                                                     matriz[j][1] = rssi
197                                                     matriz[j][0] = tag[0]
198                                                     j=j+limite
199                                                 if IP!=ibeacon:
200                                                     j=j-1
201                                             j=j+1
202                                             tag[3]=time.time()   # update lastseen
203                                               found=1
204
205                                     if found==0 :
206
207 TAG_DATA.append([macAdressSeen,macAdressSeen,0,time.time()])
208         #                                           logging.debug('New Beacon %s Detected ',
209 macAdressSeen)
210             sock.setsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, old_filter )
211
212 # ---------------------------------------------------------------------------------------
213 #                               accion de filtro gaussiano
214         suma=0
215         for x in range (0,muestras):
216                 f=open(nuevaruta+'/'+ibeacon+'medicion.txt', 'a')
217                 f.write(matriz[x][1]+'\n')
218                 f.close()
219                matriz[x][1]=float(matriz[x][1])
220                suma=suma+matriz[x][1]
221         promedio=suma/muestras
222         suma=0
223         for x in range (0,muestras):
224                 suma=suma+(matriz[x][1]-promedio)**2
225         sigma=suma/muestras
226         cont=0
227         suma=0
228          for x in range (0,muestras):
229                 errores=math.exp(-1*(matriz[x][1]-promedio)**2/(2*sigma))
230                 if errores>0.8:
231                      cont=cont+1
232                      suma=suma+matriz[x][1]
233          promedio=suma/cont
234          return promedio
235
236 # ---------------------------------------------------------------------------------------
237 #          Detecta numero dispositivos en el entorno y entrega direccion
238 def partida():
239         #Reset Bluetooth interface, hci0
240         os.system("sudo hciconfig hci0 down")
241         os.system("sudo hciconfig hci0 up")
242
243         #Make sure device is up
244         interface    =    subprocess.Popen(["sudo    hciconfig"],    stdout=subprocess.PIPE,
245 shell=True)
246         (output, err) = interface.communicate()
247
248         if "RUNNING" in output: #Check return of hciconfig to make sure it's up
249                 a=1
250         #    logging.debug('Ok hci0 interface Up n running !')
251         else:
252             logging.critical('Error : hci0 interface not Running. Do you have a BLE device
253 connected to hci0 ? Check with hciconfig !')
254             sys.exit(1)
255
256         devId = 0
257         try:
258             sock = bluez.hci_open_dev(devId)
259
260             logging.debug('Connect to bluetooth device %i',devId)
261         except:
```

```
262             logging.critical('Unable to connect to bluetooth device...')
263             sys.exit(1)
264 
265         old_filter = sock.getsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, 14)
266         hci_toggle_le_scan(sock, 0x01)
267         muestra=80
268         j=0
269         cont=0
270         while (j<muestra+15):
271             restriccion=0
272             while restriccion < 100:
273                     old_filter = sock.getsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, 14)
274                     flt = bluez.hci_filter_new()
275                     bluez.hci_filter_all_events(flt)
276                     bluez.hci_filter_set_ptype(flt, bluez.HCI_EVENT_PKT)
277                     sock.setsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, flt )
278 
279                     pkt = sock.recv(255)
280                     restriccion=' '.join("{0:02x}".format(ord(x)) for x in pkt[::-1])
281                     restriccion= len(restriccion)
282 
283             if (j>=15):
284                     j=j-15
285                     mima=' '.join("{0:02x}".format(ord(x)) for x in pkt[35:19:-1])
286                     x=0
287                     if j==0:
288                         revisar[0]=mima
289                     while (x<j):
290                                 if mima==revisar[x]:
291                                         x=j
292                                 elif cont==x:
293                                         cont=cont+1
294                                         revisar[cont]=mima
295                                         x=j
296                                 x=x+1
297                     j=j+15
298             j=j+1
299         return cont
300 
301 # ------------------------------------------------------------------------------
302 #               inicio de programa condiciones ambientales
303 
304 # se pide el numero de referencias a utilizar para estimacion de posicion
305 print ('Number of references: ')
306 referencias=teclado()
307 referencias=int(referencias)
308 
309 #busca dispositivos disponibles para seleccionar direccion
310 cont=partida()
311 matriztotal = [range(7) for i in range(referencias)]
312 
313 #estima condiciones ambientales para cada referencia
314 for x2 in range (0,referencias):
315     #muestra en pantalla direcciones disponibles, luego pide direccion a muestrear y su
316 posicion en metros
317         for x1 in range (0,cont+1):
318                         print revisar[x1]
319         print ("select Address {}:".format(x2+1))
320         IP=teclado()
321         print ("Reference position x,y:")
322         pos_xy=teclado()
323         coma=pos_xy.find(',')
324         pos_x=pos_xy[0:coma]
325         pos_y=pos_xy[coma+1:]
326     #se pide colocar dispositivo a un metro de referecia y presionar cualquier tecla para
327 obtener A
328         print ("Obtain A parameter at 1 meter of distance, press any key")
329         distancia=raw_input()
330         A=seleccionar_MAC(IP)
331         A=float(A)
```

```
332         print (A)
333
334     #Se pide numero de posiciones a utilizar para estimar n
335         print ("number of measurements to estimate n: ")
336         num=teclado()
337         num=int(num)
338     #luego se pide desplazar posicion del robot a distancias diferentes de 1 metro e indicar
339 en metros la distancia,
340     #para finalizar con la obtencion de n
341         suma=0
342         for x in range(0,num):
343                 print ("Move the device and write the distance of the measurement in meters:
344 ")
345                 distancia=teclado()
346                 distancia=float(distancia)
347                 RSSI=seleccionar_MAC(IP)
348                 logar=math.log10(distancia)
349                 n=(RSSI-A)/(10*logar)
350
351                 print n
352                 suma=suma+n
353 # -----------------------------------------------------------------------------------
354 #               promedia valor de n y guarda todos los datos en memoria
355         npromedio=suma/num
355         matriztotal[x2][0]=matriz[x2][0]
356         matriztotal[x2][1]=IP
357         matriztotal[x2][2]=matriz[x2][3]
358         matriztotal[x2][3]=A
359         matriztotal[x2][4]=npromedio
360         matriztotal[x2][5]=pos_x
361         matriztotal[x2][6]=pos_y
362         print ('A: {} n: {}'.format(A,npromedio)   )
363
364
365 with open(nuevaruta+'/tabla.csv', 'a') as csvfile:
366     writer = csv.writer(csvfile)
367     writer.writerows(matriztotal)
368
369 guardardireccion=open(direccionprueba+'/direccion.txt',"w")
370 guardardireccion.write(nuevaruta)
371 guardardireccion.close()
372
373
374
375
376
377
378
379
380
381
```

B.2 Estimación Posición

```
1
2  import logging
3
4  # choose between DEBUG (log every information) or CRITICAL (only error)
5  logLevel=logging.DEBUG
6  # ---------------------------------------------------------------------------
7  #                                  Librerias
8  import os
9  import subprocess
10 import sys
11 import struct
12 import bluetooth._bluetooth as bluez
13 import time
14 import requests
15 import signal
16 import threading
17 import csv
18 import math
19 import numpy as np
20 # ---------------------------------------------------------------------------
21 #       Limpieza pantalla y busqueda de condiciones ambientales calculadas
22
23 os.system('clear')
24
25 #buscar carpeta contenedora en memoria, donde es necesario asignar numeroprueba
26 #segun la muestra de condiciones ambientales a utilizar
27 numeroprueba=1
28 opendir=open('/home/pi/Desktop/bluetooth/pruebas'+numeroprueba+'/direccion.txt',"r")
29 direccion=opendir.read()
30
31 limite=7                # Numero de muestras para el promedio de RSSI
32 muestras=25             # cantidad en veces para la estimacion de posicion
33 # ---------------------------------------------------------------------------
34 #                 obtener condiciones ambientales de memoria
35
36 matriz2 = [range(7) for i in range(100)]
37 with open(direccion+'/tabla.csv') as csvfile:
38     reader = csv.DictReader(csvfile)
39     i=0
40     for row in reader:
41         matriz2[i][0] = row['address']
42
43         matriz2[i][1] = row['UUID']
44         print (matriz2[i][1])
45
46         matriz2[i][2] = row['minormajor']
47
48         matriz2[i][3] = row['A']
49         matriz2[i][3] = float(matriz2[i][3])
50
51         matriz2[i][4] = row['n']
52         matriz2[i][4] = float(matriz2[i][4])
53
54         matriz2[i][5] = row['posicion_x']
55         matriz2[i][5] = float(matriz2[i][5])
56
57         matriz2[i][6] = row['posicion_y']
58         matriz2[i][6] = float(matriz2[i][6])
59
60         i = i+1
61 tamano = i
62 # ---------------------------------------------------------------------------
63 #               Creacion matrices nulas para el sistema
64
65 pos = np.zeros((tamano,2))
66 m = np.zeros((tamano,1))
67 matrizrssi = np.zeros((tamano,7))
```

```
68  matrssi = np.zeros((tamano,limite))
69  promrssi = np.zeros((tamano,1))
70  # ------------------------------------------------------------------------------------
71  #                   LLenado matriz de posicion de referencias
72  for x in range(0,tamano):
73          matrizrssi[x,5] = 0
74          pos[x,0] = matriz2[x][5]
75          pos[x,1] = matriz2[x][6]
76  # ------------------------------------------------------------------------------------
77  #                   Creacion archivo guardado de posicion
78  posicion_medida = raw_input('ingrese posicion: ')
79  limx = 1.7
80  limy = 1.3
81
82  headers = ('pos x', 'pos y', 'hora')
83  with open(direccion+'/pos_'+posicion_medida+'.csv', 'w') as csvfile:
84      writer = csv.writer(csvfile)
85      writer.writerow(headers)
86  # ------------------------------------------------------------------------------------
87  #           codigos coomparacion para deteccion de tramas bluetooth
88  LE_META_EVENT = 0x3e
89  OGF_LE_CTL = 0x08
90  OCF_LE_SET_SCAN_ENABLE = 0x000C
91  EVT_LE_CONN_COMPLETE = 0x01
92  EVT_LE_ADVERTISING_REPORT = 0x02
93
94  # ------------------------------------------------------------------------------------
95  # A partir de la matriz de posicion de referencias y distancias estimadas, realiza la
96  transformacion
97  # de la ecuacion de trilateralizacion y obtiene posicion mediante minimos cuadrados
98  def estimar_posicion(pos,m):
99      #print pos
100     print m
101     #funcion multiplicar matriz A y B comparando que sean matrices validas
102     def multiplicacion(a,b):
103         fila = a.shape[0]
104         cola = a.shape[1]
105         filb = b.shape[0]
106         colb = b.shape[1]
107         if cola==filb:
108             prueba = np.zeros((fila,colb))
109             x = 0
110             z = 0
111             for x in range(0,fila):
112                 for z in range(0,colb):
113                     for j in range(0,filb):
114                         #print('x {} z {} j {}'.format(x,z,j))
115                         prueba[x,z] = prueba[x,z]+a[x,j]*b[j,z]
116             return prueba
117         else:
118             print('matriz distinto rango')
119             return
120 #revisa si se tienen al menos dos ecuaciones
121     rev = 0
122     filpos = pos.shape[0]-1
123     A = pos
124     B = m
125     if filpos==1:
126         filpos = filpos+1
127         A = pos
128         B[0] = m[0]**2
129         B[1] = m[1]**2
130         rev = 1
131
132 #si se tienen mas de 2 se reduce la matriz seguna la primera ecuacion
133     if rev==0:
134             A = np.zeros((filpos,2))
135             B = np.zeros((filpos,1))
136             x1 = pos[0,0]
137             y1 = pos[0,1]
```

```
138             for x in range(0,filpos):
139                     A[x,0] = pos[x+1,0] - x1
140                     A[x,1] = pos[x+1,1] - y1
141                     B[x,0] = 0.5*(m[0]**2 - m[x+1]**2 + A[x,0]**2 + A[x,1]**2)
142
143 #metodo de minimos cuadrados
144     X = np.zeros((3,1))
145     A_t = A.transpose()
146     AtA = multiplicacion(A_t,A)
147     AtA_inv = np.linalg.inv(AtA)
148     coef = multiplicacion(AtA_inv,A_t)
149     X = multiplicacion(coef,B)
150     for x in range(0,1):
151             if rev==0:
152                 X[x] = X[x] + pos[0,x]
153     return X
154
155
156 # --------------------------------------------------------------------------------
157 #           definicion estructura de codificacion trama bluetooth
158 def print_packet(pkt):
159     for c in pkt:
160         sys.stdout.write("%02x " % struct.unpack("B",c)[0])
161
162 def packed_bdaddr_to_string0(bdaddr_packed):
163     return ':'.join('%02x'%i for i in struct.unpack("<BBBB", bdaddr_packed[::-1]))
164
165 def packed_bdaddr_to_string(bdaddr_packed):
166     return ':'.join('%02x'%i for i in struct.unpack("<BBBBBB", bdaddr_packed[::-1]))
167
168 def packed_bdaddr_to_string1(bdaddr_packed):
169     return    '    '.join('%02x'%i    for    i    in    struct.unpack("<BBBBBBBBBBBBBBBB",
170 bdaddr_packed[::-1]))
171
172 def hci_disable_le_scan(sock):
173     hci_toggle_le_scan(sock, 0x00)
174
175 def hci_toggle_le_scan(sock, enable):
176     cmd_pkt = struct.pack("<BB", enable, 0x00)
177     bluez.hci_send_cmd(sock, OGF_LE_CTL, OCF_LE_SET_SCAN_ENABLE, cmd_pkt)
178
179 def handler(signum = None, frame = None):
180     time.sleep(1)  #here check if process is done
181     sys.exit(0)
182
183 # --------------------------------------------------------------------------------
184 #                                 accion de filtro gaussiano
185 def arregloerror():
186         promrssi=np.zeros((tamano,1))
187
188         for i in range(0,tamano):
189             suma=0
190             rev=1
191             for x in range (0,limite):
192                     matrssi[i][x] = float(matrssi[i][x])
193                     suma = suma+matrssi[i][x]
194             promedio=suma/limite
195             suma=0
196             for x in range (0,limite):
197                     suma=suma+(matrssi[i][x]-promedio)**2
198             sigma=suma/muestras
199             cont=0
200             suma=0
201             for x in range (0,limite):
202                     errores=math.exp(-1*(matrssi[i][x]-promedio)**2/(2*sigma))
203                     if errores>0.8:
204                         cont=cont+1
205                         suma=suma+matrssi[i][x]
206
207             if cont>0:
208                  promrssi[i]=suma/cont
```

```
209         return promrssi
210
211
212 # ------------------------------------------------------------------------------
213 #           activacion bluetooth y de proceso de deteccion de tramas
214 for sig in [signal.SIGTERM, signal.SIGINT, signal.SIGHUP, signal.SIGQUIT]:
215     signal.signal(sig, handler)
216
217
218
219 FORMAT = '%(asctime)s - %(name)s - %(levelname)s - %(message)s'
220 if globals().has_key('logOutFilename') :
221     logging.basicConfig(format=FORMAT,filename=logOutFilename,level=logLevel)
222 else:
223     logging.basicConfig(format=FORMAT,level=logLevel)
224
225 #Reset Bluetooth interface, hci0
226 os.system("sudo hciconfig hci0 down")
227 os.system("sudo hciconfig hci0 up")
228
229 #Make sure device is up
230 interface = subprocess.Popen(["sudo hciconfig"], stdout=subprocess.PIPE, shell=True)
231 (output, err) = interface.communicate()
232
233 if "RUNNING" in output: #Check return of hciconfig to make sure it's up
234        a=1
235 #    logging.debug('Ok hci0 interface Up n running !')
236 else:
237     logging.critical('Error : hci0 interface not Running. Do you have a BLE device
238 connected to hci0 ? Check with hciconfig !')
239     sys.exit(1)
240
241 devId = 0
242 try:
243     sock = bluez.hci_open_dev(devId)
244
245     logging.debug('Connect to bluetooth device %i',devId)
246 except:
247     logging.critical('Unable to connect to bluetooth device...')
248     sys.exit(1)
249
250 old_filter = sock.getsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, 14)
251 hci_toggle_le_scan(sock, 0x01)
252
253 # ------------------------------------------------------------------------------
254 #                   Inicio programa de obtencion de posicion
255 program=0
256 while(program<muestras):
257         #print (program)
258        contador=0
259        for x in range(0,tamano):
260                matrizrssi[x,1]=0
261               matrizrssi[x,0]=0
262     # funcion detecta y filtra tramas incompletas
263         while (contador<tamano):
264             restriccion=0
265             while restriccion < 120:
266                     old_filter = sock.getsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, 14)
267                     flt = bluez.hci_filter_new()
268                     bluez.hci_filter_all_events(flt)
269                     bluez.hci_filter_set_ptype(flt, bluez.HCI_EVENT_PKT)
270                     sock.setsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, flt )
271
272                     pkt = sock.recv(255)
273                     restriccion=' '.join("{0:02x}".format(ord(x)) for x in pkt[::-1])
274                     restriccion= len(restriccion)
275
276         #parametros revision bluetooth de trama valida
277             ptype, event, plen = struct.unpack("BBB", pkt[:3])
278
279             if event == bluez.EVT_INQUIRY_RESULT_WITH_RSSI:
```

```
                    i =0
            elif event == bluez.EVT_NUM_COMP_PKTS:
                    i =0
            elif event == bluez.EVT_DISCONN_COMPLETE:
                    i =0
            elif event == LE_META_EVENT:
                    subevent, = struct.unpack("B", pkt[3])
                    pkt = pkt[4:]
                    if subevent == EVT_LE_ADVERTISING_REPORT:
                        num_reports = struct.unpack("B", pkt[0])[0]
                        report_pkt_offset = 0
                # para trama valida se revisa si se han obtenido todas las muestras para
los dispositivos
                        for i in range(0, num_reports):
                                if contador<limite:
        # si no se has obtenido registra direccion, compara con referencias y si es valida
guarda en matrizrssi

        ibeacon=packed_bdaddr_to_string1(pkt[report_pkt_offset + 16:report_pkt_offset +
32])
                                                                for  x  in  range(0,tamano):

                                                                if    matriz2[x][1]==ibeacon
and matrizrssi[x,1]<limite:
                                                                        rssi=''.join(c for c
in str(struct.unpack("b", pkt[report_pkt_offset -1])) if c in '-0123456789')

        rssi=float(rssi)
                                                                        matrizrssi[x,0]    =
matrizrssi[x,0]+rssi
                                                                        j=matrizrssi[x,1]
                                                                        j=int(j)
                                                                        matrssi[x,j]=rssi

                                                                        matrizrssi[x,1]    =
matrizrssi[x,1]+1
                                                                        matrizrssi[x,3]    =
matriz2[x][3]
                                                                        matrizrssi[x,4]    =
matriz2[x][4]
                                                                if matrizrssi[x,1]==limite:
                                                                        contador=contador+1

        matrizrssi[x,1]=limite+1


            sock.setsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, old_filter )

# --------------------------------------------------------------------------------
#ejecuta filtro gaussiano y revisa el numero de muestras validas (distintas de cero)
luego del filtro
        indice=0

        promrssi=arregloerror()
        for x in range(0,tamano):

               if promrssi[x] != 0:
                   indice = indice+1

# --------------------------------------------------------------------------------
#bloque para calcular posicion con el numero total de dispositivos
#        if indice==tamano:
#                with open(direccion+'/posrssi_'+posicion_medida+'.csv', 'a') as csvfile:
#                 writer = csv.writer(csvfile)
#                 writer.writerow(promrssi.transpose())
#             for x in range(0,tamano):
#                 rssi=promrssi[x]
#               m[x]=10**((rssi-matrizrssi[x,3])/(10*matrizrssi[x,4]))
#        final= estimar_posicion(pos,m)
# --------------------------------------------------------------------------------
```

```
#bloque para calcular posicion asignando un numero minimo de dispositivos
        if indice >= 4:
            aux = 0
            pos2 = np.zeros((indice,2))
            m2 = np.zeros((indice,1))
            with open(direccion+'/posrssi_'+posicion_medida+'.csv', 'a') as csvfile:
                    writer = csv.writer(csvfile)
                    writer.writerow(promrssi.transpose())
            for x in range(0,tamano):

                rssi = promrssi[x]
                if rssi != 0:
                    m2[aux] = 10**((rssi-matrizrssi[x,3])/(10*matrizrssi[x,4]))
                    pos2[aux,0] = matriz2[x][5]
                    pos2[aux,1] = matriz2[x][6]
                    aux = aux+1
            final= estimar_posicion(pos2,m2)
# ------------------------------------------------------------------------------
#limita posicion con tamaño de superficie, pide tiempo en sistema y guarda en memoria
            guardar=np.zeros((1,3))
            if final[0]<0:
                final[0]=0
            if final[0]>limx:
                final[0]=limx
            if final[1]<0:
                final[1]=0
            if final[1]>limy:
                final[1]=limy
            guardar[0,0]=final[0]
            guardar[0,1]=final[1]
            guardar[0,2]=int(time.strftime("%I%M%S", time.gmtime()))
            with open(direccion+'/pos_'+posicion_medida+'.csv', 'a') as csvfile:
                writer = csv.writer(csvfile)
                writer.writerow(guardar)

            print ('pos {},{} time {}'.format(guardar[0,0],guardar[0,1],guardar[0,2]))
            program=program+1
```