UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación de un sistema de localización en tiempo real en espacios cerrados para el robot Rover UVG

Trabajo de graduación presentado por Carmen Natalia de León Bercián
para optar al grado académico de Licenciada en Ingeniería Mecatrónica

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación de un sistema de localización en tiempo real en espacios cerrados para el robot Rover UVG

Trabajo de graduación presentado por Carmen Natalia de León Bercián
para optar al grado académico de Licenciada en Ingeniería Mecatrónica

Guatemala,

2023

Vo.Bo.:

(f) 

Ing. Kurt Kellner

Tribunal Examinador:

(f) 

Ing. Kurt Kellner

(f) 

MSc. Carlos Esquit

(f) 

MAEB. Pablo Mazariegos

Fecha de aprobación: Guatemala, 07 de enero de 2023.

Prefacio

La elaboración de este trabajo de graduación tiene como fin montar un sistema de localización en tiempo real con módulos DWM1001 que sea compatible con el sistema operativo para robots para así, incorporarlo al Rover UVG. Precisamente, una de las motivaciones para realizar este trabajo es que, en conjunto, con otros seis estudiantes se está llevando a cabo el proyecto del Rover UVG. La idea de este es poder unir todos los trabajos de graduación y obtener un robot móvil explorador capaz de movilizarse de un punto a otro con ayuda de sensores, cámaras y sistemas de localización.

Quiero agradecer a Dios y principalmente a mis papás por darme la oportunidad de realizar mis estudios en la Universidad del Valle de Guatemala, por estar a mi lado en todo momento, apoyarme, escucharme, animarme y más que todo por creer en mi. Agradezco a mi hermana que su apoyo y ánimo fue incondicional y a mi hermano por mostrar siempre interés.

Le agradezco a mi asesor Ing. Kurt Kellner por compartir sus conocimientos, ayudarme a solucionar los problemas y apoyarme a lo largo del proceso de mi tesis y la carrera. Agradezco a Dr. Luis Alberto Rivera por ser un gran catedrático y siempre ayudarme y a MSc. Miguel Zea por introducirme al mundo de la robótica y sistemas de control. Quiero agradecer a todas las personas que conocí durante los años en la Universidad y principalmente a mis amigos, que hicieron este tiempo más alegre y bonito.

Índice

Lista de figuras

Lista de cuadros

Resumen

El trabajo de graduación propuesto tiene como objetivo principal realizar un sistema de localización en tiempo real en un espacio cerrado para su implementación en el robot Rover UVG. Para esto se implementaron módulos MDEK1001 de la empresa Qorvo con los que se creó una red de localización con múltiples módulos instalados en el espacio y un módulo colocado en el robot móvil.

Adicionalmente, la solución se implementó en el sistema operativo robótico (ROS por sus siglas en inglés) para que pudiera integrarse con los demás sensores y módulos desarrollados en otros trabajos de graduación. El objetivo de implementar este sistema operativo es proveerle de funcionalidad y modularidad al robot Rover UVG.

Se evaluó la precisión y exactitud del sistema. Para hacerlo se midió la posición en diversos puntos dentro de la red obteniendo así una exactitud de $\pm 18.15cm$ y una desviación estándar relativa de 0.04 %. Finalmente, se compararon los resultados con los obtenidos en el sistema de seguimiento 3D y captura de movimiento OptriTrack. El objetivo de la comparación era determinar que método brinda información más precisa y exacta para el control del robot.

Abstract

The main objective of the proposed graduation work was to implement a real-time location system in a closed space oriented to the Rover UVG robot. To do this, MDEK1001 modules from Qorvo were implemented to create a localization Network with modules placed all over the area and one module placed on the mobile robot.

Additionally, the system was implemented in a *Robotic Operating System* so that it could be integrated with the other sensors and modules developed in other graduation works. The objective for implementing the operating system was to provide functionality and modularity to the Rover UVG robot.

Accuracy and precision of the system were evaluated. To do so, several and specific coordinates were measured within the Network, giving as a result a $\pm 18.15cm$ of accuracy and a standard deviation of 0.04. Finally, the modules were compared with a 3D tracking system and precision motion capture called OptriTrack. The intention was to determine which method provides the most accurate and precise information for the robot.

CAPÍTULO 1

Introducción

Los sistemas de localización en tiempo real son soluciones tecnológicas que permiten identificar y localizar objetos o personas en tiempo real. En una implementación típica se colocan *tags* y *anchors*, en donde los primeros van en las personas u objetos en movimiento y los segundos son la tecnología estática que sirve de referencia. Los dispositivos pueden configurarse como transmisores o receptores capaces de crear una red y localizar los objetos en movimiento.

En el presente trabajo de graduación se implementó un sistema de localización en tiempo real orientado al robot Rover UVG. Se utilizaron los módulos MDEK1001 de la empresa Qorvo. Estos utilizan tecnología de banda ultra-ancha (UWB, por sus siglas en inglés). Para hacer pruebas se utilizó una plataforma robótica con los *anchors* colocados en su periferia. La red formada consistió de 6 *anchors* capaces de identificar la posición del módulo colocado en el robot móvil.

También se implementó el sistema de localización en el entorno del Sistema Operativo Robótico (*ROS* por su siglas en inglés). La implementación de ROS permite incorporar sensores, módulos, sistemas de localización etc. al robot de forma eficaz y modular.

Se hizo una comparación de los módulos MDEK1001 contra el sistema de captura 3D: Optitrack. La prueba realizada fue medir los mismos puntos al mismo tiempo con ambos sistemas de localización y corroborar qué datos eran más precisos y exactos. Finalmente, se hizo otro experimento para corroborar los datos obtenidos por los módulos. Este consistió en medir manualmente diversas coordenadas en la plataforma robótica y luego colocar el *tag* en esos mismos puntos. Con esto se logró evaluar qué tan preciso y exacto es el sistema para una aplicación en robótica.

CAPÍTULO 2

Antecedentes

2.1. Sistema de localización en tiempo real de la empresa *Eliko*

La empresa de *Eliko* [1] desarrolló una tecnología de localización basada en comunicación inalámbrica por banda ultra-ancha (UWB por sus siglas en inglés). Esta consta de *anchors*, *tags*, un servidor y el software de rango. Alcanza diferentes niveles de precisión con una detección de proximidad de hasta $3cm$, capaz de utilizarse en exteriores e interiores y con facilidad de configuración según el uso que se le quiere dar.

Los *anchors* son dispositivos de referencia que se colocan alrededor de toda el área de localización, en puntos, de preferencia visibles y accesibles. La cantidad mínima que se debe colocar son 4. Estos tienen un alcance operativo de hasta 70 metros, dimensiones de $110x90x70mm$ (largo x ancho x alto) y contienen accesorios que facilitan montarlos en las paredes o techo. Se puede observar el dispositivo en la Figura 1.

Figura 1: Anchors de la empresa Eliko

Los *tags* son pequeños dispositivos móviles que se colocan en cualquier objeto que se requiera rastrear. Estos constantemente están actualizando su posición y mandando la información a los *anchors* por medio de tecnología de radio de banda ultra-ancha. Con la

información recibida cada *anchor* puede medir la distancia que hay entre su posición y el objeto móvil en ese preciso momento. Posteriormente se calculan las coordenadas del objeto dentro del área de localización y se hacen visibles para el usuario por medio del software gráfico. Estos dispositivos tienen un alcance operativo de hasta 70 metros, dimensiones de $63x42x13mm$ (largo x ancho x alto), luces LEDS configurables, motor de vibración y dispositivos externos para alarmas. Se puede observar el dispositivo en la Figura 2

Figura 2: Tag de la empresa Eliko

2.2. Localización de posición de banda ultra-ancha para un entorno de oficina utilizando módulos DWM1001-DEV

En [2] Jake Uribe realizó un proyecto con el objetivo de determinar la viabilidad y precisión de hacer un sistema en tiempo real para tener control de las personas dentro de una oficina. Decidió utilizar la tecnología DWM1001 de la empresa Decawave para determinar si es capaz de localizar a los trabajadores con una precisión de las medidas de un escritorio. Esta tecnología cuenta con dispositivos que pueden configurarse como *anchors* o como *tags*. Los primeros son los dispositivos estáticos que se colocan alrededor de toda el área y los *tags* son los dispositivos móviles que se colocan en lo que se quiere localizar, en este caso sería para que cada persona tenga uno.

Para implementar el sistema se eligió el 4to piso del *Seamans Center Anne* en Iowa, Estados Unidos. Se colocaron 3 *anchors* en habitaciones accesibles y concurridas. Se realizaron dos pruebas, una con un *tag* estático y la otra con un *tag* móvil. La primera prueba consistió en dejar el *tag* estático sobre un escritorio para recolectar 1100 datos. En la segunda prueba el *tag* lo llevó una persona caminando de este a oeste, dentro de la oficina, a un paso uniforme y se recolectaron 400 datos. Se obtuvo como resultado que es posible realizar un sistema de localización en tiempo real para monitorear personas dentro de una oficina utilizando la tecnología DWM1001 de Decawave con una precisión de $\pm 20cm$.

2.3. Diseño, implementación y mejoras del robot explorador modular *ROVER* en UVG

En la fase anterior se implementaron componentes mecánicos, electrónicos y software al robot ROVER de UVG.

En el proyecto de Héctor Sagastume [3] se realizaron cambios a la estructura principal del robot. La intención era tener una estructura que cubriera los componentes internos, se adaptara a las llantas que tenía y su material fuera liviano, resistente y duradero (teniendo un mantenimiento adecuado). Para el control y movimiento se utilizaron 3 controladores: Arduino, Raspberry Pi y Pixhawk. Este último incluye GPS, giroscopio, acelerómetro y barómetro en una sola plataforma. Adicionalmente se colocó un sensor ultrasónico HC-SR04, un sensor de humedad y temperatura DHT11 y se cambiaron los motores que controlan las llantas por unos con mayor velocidad, pero menor torque.

Los materiales utilizados fueron acero para la estructura y aluminio para las placas que cubrían el robot. La estructura terminada se unió a las llantas, se colocaron los sensores externos y se acomodaron los componentes internos dentro de la estructura del robot. El resultado se puede observar en la Figura 3. Por último, se probó el funcionamiento del robot moviéndolo hacia adelante y hacia atrás en las instalaciones de la Universidad del Valle de Guatemala.

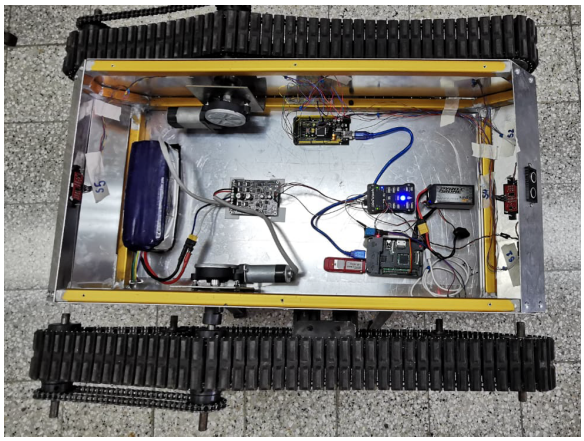


Figura 3: Resultado final del robot Rover en la fase anterior.

En el proyecto de Javier Archila [4] se trabajó el software y electrónica del robot para controlar su movimiento. Para lograrlo se utilizó el control de navegación *Pixhawk* y una Raspberry Pi como controlador principal. Estos controladores se conectaron por medio de una señal de *Wifi* generada por el módulo de la Raspberry Pi, debido a esto su alcance era limitado. Sin embargo, al comunicarse con *Wifi* fue posible usar un control RF inalámbrico. Las pruebas realizadas consistieron en conectar el *Pixhack* a los drivers del motor y movilizarlo con el control RF por terrenos planos, no uniformes y subiendo las banquetas de las calles, lo cual se puede observar en la Figura 4

Figura 4: Pruebas de movimiento del ROVER en la fase anterior.

2.4. Implementación del algoritmo *Particle swarm optimization* (PSO) dentro del entorno de *Robotic operating System* (ROS) en UVG

El objetivo del proyecto de Marco Izeppi [5] fue comprobar la viabilidad de utilizar ROS para la implementación del algoritmo PSO. Para lograrlo se hicieron pruebas iniciales, implementando el algoritmo de Dijkstra y el algoritmo de D^*. Para el adecuado funcionamiento de los algoritmos se crearon y se utilizaron nodos disponibles en ROS. Adicionalmente se utilizaron simuladores 3D como Gazebo y Rviz, siendo este último compatible con ROS, para poder visualizar el comportamiento del robot con los algoritmos implementados.

Para la implementación del algoritmo PSO dentro de ROS se crearon 3 robots con base en el modelo United Robotics Description Format (URDF) del robot Turtlebot, el cual viene dentro de la instalación de ROS. Para el proyecto se trabajó con el lenguaje de programación Python para crear el código general de cada robot y crear un nodo global que tomaba todos los parámetros independientes de cada robot y los comparaba. Como resultado se obtuvo la mejor posición global para sincronizar a todo el enjambre de robots en un mismo punto. Se puede observar en la Figura 5 la simulación de ROS como todos los robots convergen en el mismo punto.

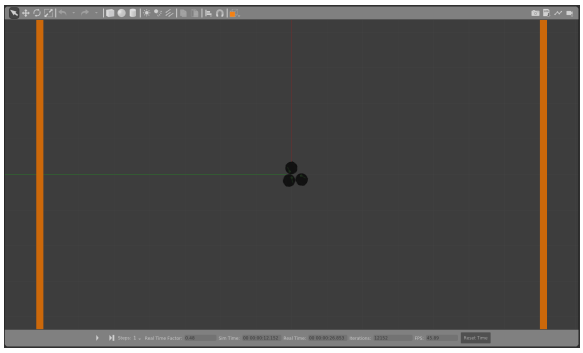


Figura 5: Todos los robots en un mismo punto con algoritmo PSO en ROS.

2.5. Implementación de un sistema de control para un robot secador de café en UVG

El proyecto de Álvaro Torres [6] consistió en implementar un robot móvil con un sistema de control que cumpliera con la función específica de secado en una finca de café. Se diseñó un prototipo funcional en el cual se implementaron todos los módulos y controladores a utilizar en el robot real, este se puede observar en la Figura 6. En este modelo se realizaron pruebas por separado de cada módulo para asegurar su correcto funcionamiento de cuando se integraran todos. Estos son el módulo DWM1001-DEV, Encoders, ATMega256 y DC Motor Driver 2x15A.

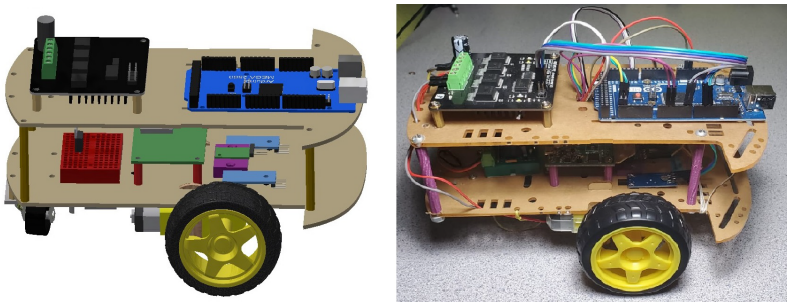


Figura 6: Modelo 3D y prototipo del 2WD Smart Robot completo

Los módulos DWM1001-DEV se utilizaron para hacer un sistema de localización en tiempo real o por sus siglas en inglés RTLS. Colocando uno de los módulos anclado al robot (funcionando como *tag*) y otros 3 módulos colocados en las esquinas del área de trabajo (funcionando como *anchors*). Se realizó la configuración de hardware y software de cada uno de los DWM1001-DEV para que llevaran a cabo el rol de *anchor* o *tag* respectivamente. Posteriormente se hicieron pruebas en un área pequeña, moviendo al robot de forma lineal a un paso uniforme por medio de jalar una bolsa plástica que se encontraba debajo como se muestra en la Figura7.

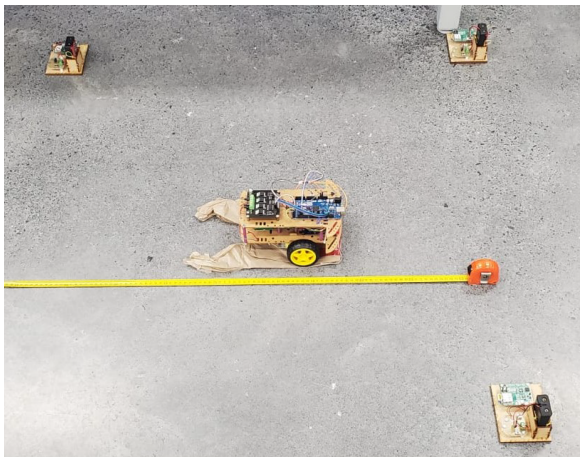


Figura 7: Manipulación del robot para escenarios en pruebas en movimiento

Luego de realizar diversas pruebas se llegó a la conclusión que utilizando solo 3 módulos como *anchors* el resultado de la posición no era precisa. Dado esto, se optó por usar la

distancia vectorial que se recibía de los cada uno de los anchors para triangular la ubicación real del robot por medio de software.

Al calcular la posición de esta forma se estaban obteniendo valores bastante dispersos. El rango normal estaba en valores cercanos a 500 (valores en bruto recibidos del *tag*), sin embargo, se estaban obteniendo valores muy altos o bajos en comparación al rango normal. Por lo que, finalmente, para erradicar datos erróneos que estuvieran fuera de un rango de ± 10 cm se realizó un filtro pasa banda en software, de forma que se obtuvo datos más precisos. Sin embargo, no se pude establecer una posición exacta a pesar de que los valores estaban cercanos al rango normal. Debido a esto el control del robot y el sistema RTLS no se implementaron en conjunto, sino que el control se aplicó a cada uno de los motores por separado.

CAPÍTULO 3

Justificación

En la fase anterior del proyecto del robot Rover UVG se modificó la estructura mecánica, se mejoró el mecanismo de potencia y se implementó el software de control y movimiento de este. Sin embargo, el alcance fue que únicamente se moviera hacia adelante y hacia atrás con un control remoto. El siguiente paso del proyecto es aportar a que el robot sea autónomo y funcione en espacios cerrados. Por esta razón, se requiere implementar un sistema de localización en tiempo real para el mismo.

En otro trabajo de graduación, en la fase anterior se implementó un sistema de localización utilizando módulos DWM1001 que funcionan con tecnología de banda ultra-ancha. Sin embargo, solo se contaba con 4 módulos, 3 configurados como *anchors* y uno como *tag*. Dado esto, los datos obtenidos directamente de los módulos no fueron exactos ni precisos. Para solucionar esto se utilizó la distancia vectorial para triangular la posición del robot en conjunto con filtro pasa banda en software para obtener datos congruentes. A pesar de esto, los datos siguieron sin ser precisos por lo que no fue posible aplicar el sistema de control en el robot.

Tomando esto en consideración, en este proyecto se busca realizar una red de localización en tiempo real para un espacio cerrado colocando al menos 4 *anchors* en el área y un *tag* en el robot. Además, se busca que el sistema sea compatible con ROS para una mejor integración de los distintos módulos del robot. Lo que se quiere obtener de los módulos son datos precisos de la posición para la implementación en el robot.

CAPÍTULO 4

Objetivos

Objetivo general

Implementar un sistema de localización en tiempo real basado en los módulos DWM1001C y ROS para el robot Rover UVG.

Objetivos específicos

- Realizar una red de localización para entornos cerrados con un mínimo de un *tag* y múltiples *anchors*.
- Adaptar el hardware y firmware del sistema de localización para que sea compatible con ROS.
- Evaluar la precisión y exactitud del sistema de localización en tiempo real.

CAPÍTULO 5

Alcance

El alcance de este trabajo abarcó la implementación de un sistema de localización en tiempo real con módulos MDEK1001. En la red de posicionamiento se consideraron únicamente 2 dimensiones, es decir, el movimiento del robot solo fue en x & y. La red contaba con 6 *anchors* posicionados en forma rectangular, en donde la coordenada $(0, 0)$ coincidía con el módulo de la esquina inferior izquierda.

Los experimentos se realizaron en un ambiente controlado sobre una plataforma robótica que limitó el área de prueba. En estos el *tag* siempre tuvo línea de vista directa a todos los *anchors*. Adicionalmente, por la configuración de la red se tomaron sólo coordenadas positivas.

La programación compatible con ROS fue capaz de comunicarse con los módulos MDEK1001 vía puerto UART utilizando el formato TLV. Debido a la configuración de la red los algoritmos implementados procesan únicamente coordenadas positivas.

La unidad de medición utilizada para las pruebas de precisión y exactitud fue centímetros debido a las futuras posibles aplicaciones del proyecto y las capacidades de medición de los módulos implementados.

CAPÍTULO 6

Marco teórico

6.1. Sistema de localización en tiempo real

Conocido por sus siglas en inglés como RTLS (Real-time locating system). Estos sistemas son una combinación de hardware y software que trabajan en conjunto para crear una red inalámbrica capaz de monitorear el movimiento de objetos dentro de una ubicación específica. Están conformados por dispositivos con tecnología de comunicación inalámbrica que permite que se comuniquen entre ellos. Se crea la red inalámbrica al colocar dispositivos de lectura en puntos claves alrededor del área y un dispositivo en el objeto que se quiere monitorear.

El dispositivo que se coloca en el objeto se conoce como *tag*. Este es la parte móvil del sistema, encargado de mandar continuamente la información de su posición por medio de radio señales a los demás dispositivos. Por otra parte, se le conoce como *anchors* a los dispositivos de lectura colocados alrededor del área de trabajo. Estos son la parte estática del sistema que reciben las radio señales y las trasmiten a un servidor. En el servidor la información es procesada en software y finalmente se envía a una computadora o celular para que se pueda visualizar la posición en tiempo real de los objetos. [7]

En la Figura 8 se puede observar cómo se conforma un RTLS básico. En este caso, el sistema cuenta con 3 módulos como *anchors* y 2 módulos como *tags*. Adicionalmente, tiene un módulo configurado como *gateway* que permite la comunicación inalámbrica con una computadora o dispositivo para trasmitir la información obtenida de la posición de los *tags*.

6.2. Tecnologías de comunicación inalámbrica

6.2.1. Tecnología de banda ultra-ancha (UWB)

La tecnología de banda ultra-ancha (UWB, por sus siglas en inglés) es una tecnología de trasmisión de corto alcance que utiliza una gran porción del espectro de radio. Ocupa un

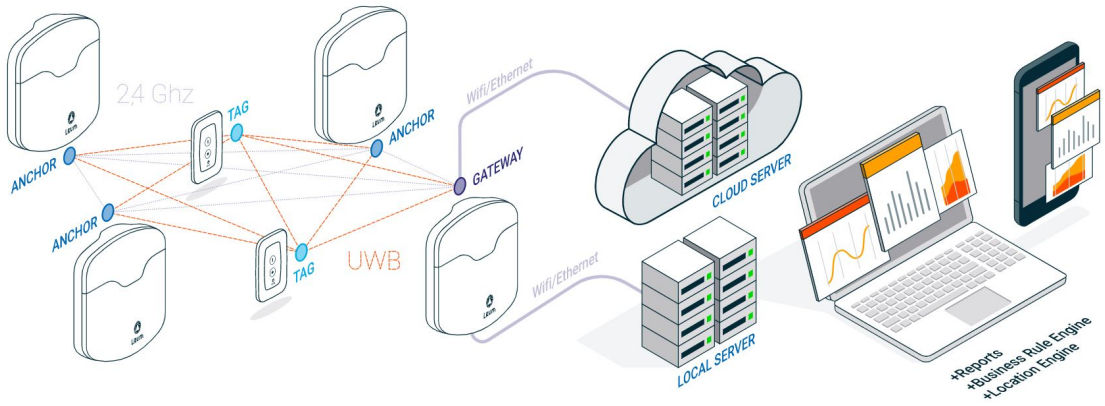


Figura 8: Diagrama de funcionamiento de un RTLS [7]

ancho de banda mayor al 25 % o más de 1.5Ghz, lo que es mucho mayor que la tecnología radio celular actual. [8]

La señal UWB está diseñada para ocupar bandas de frecuencia asignadas a otros servicios sin causar interferencias ostensibles. Estos sistemas emiten una potencia muy baja que, al repartirse en su ancho de banda tan grande, produce una densidad espectral de potencia de pocos MHz. Estos niveles de señal se parecen a los niveles de ruido que soportan los receptores de otros sistemas. Por lo tanto, se puede decir que la tecnología UWB consiste en repartir su capacidad de interferencia entre todos los demás sistemas dando, por consiguiente, que interfiera poco en ellos. [8]

Hoy en día se está decidiendo utilizar esta tecnología debido a las ventajas que posee: [9]

- Alta velocidad de trasmisión de datos: Los sistemas UWB pueden soportar más de 500 Mbps dentro de 10 metros.
- Poco desvanecimiento de la señal: Son inmunes al desvanecimiento de su señal en entornos multiruta densos.
- Seguridad: Debido a que los sistemas operan en frecuencias por debajo del ruido, son extremadamente difíciles de detectar. Por ende, es más difícil que se pierda información.
- Alcance de alta precisión: Tienen una buena resolución en el dominio del tiempo, por lo que ofrecen una resolución de centímetros para ubicar objetos.
- Poca pérdida al atravesar objetos: Con esta tecnología es posible penetrar obstáculos, permitiendo trabajar con líneas de trabajo directo o no.
- Arquitectura: Los sistemas se pueden implementar digitalmente con un *chip* pequeño, de bajo costo y potencia. Esto es esencial para dispositivos móviles.

6.3. Módulo DWM1001-DEV

Los módulos DWM1001C, desarrollados por la empresa Decawave, integra antenas UWB y Bluetooth, todos los circuitos RF, un microcontrolador nRF52832 de Nordic Semiconductor y un acelerómetro de 3 ejes. Estos módulos están diseñados para utilizarse en sistemas de localización en tiempo real. Configurando cada uno como un RTLS *anchor* o *tag*, es decir, el mismo módulo se configura según la necesidad. El módulo se puede observar en la Figura 9.

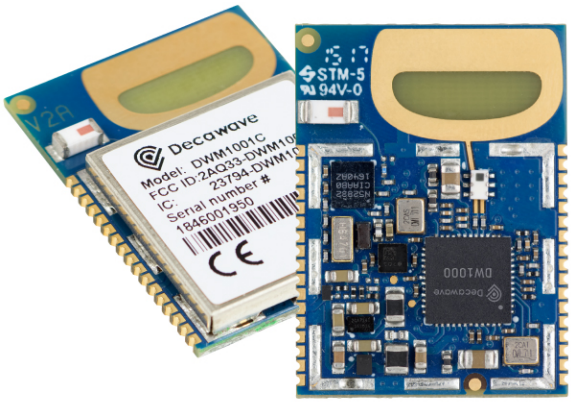


Figura 9: Módulo DWM1001C de la empresa Decawave [10]

Adicionalmente la empresa proporciona un *DWM1001 Development Board*, el cual incluye el módulo DWM1001C, conector de batería y circuito de carga, LEDs, botones, Raspberry-Pi y conector USB. Además, la plataforma incluye un módulo J-Link OB que agrega capacidades de depuración y puerto COM virtual. Este se puede observar en la Figura 10 [10]

Figura 10: *DWM1001 Development Board* de la empresa Decawave [10]

6.3.1. Tecnología de comunicación utilizada

La empresa Decawave utiliza la tecnología de Banda Ultra-ancha (UWB). Está basada en los estándares IEEE 802.15.4a y 802.15.4z que permiten una medición muy precisa del tiempo de vuelo de la señal de radio. Esto da como resultado una medición de distancia/ubicación con precisión centimétrica. En la empresa, las propiedades físicas de la señal UWB se definieron específicamente para lograr una ubicación y comunicación en tiempo real, ultra precisas y ultra confiables. En la Figura 11 se puede observar una comparación que hizo la empresa Decawave de su tecnología UWB contra diferentes tecnologías para comprobar que es la mejor opción. [10]

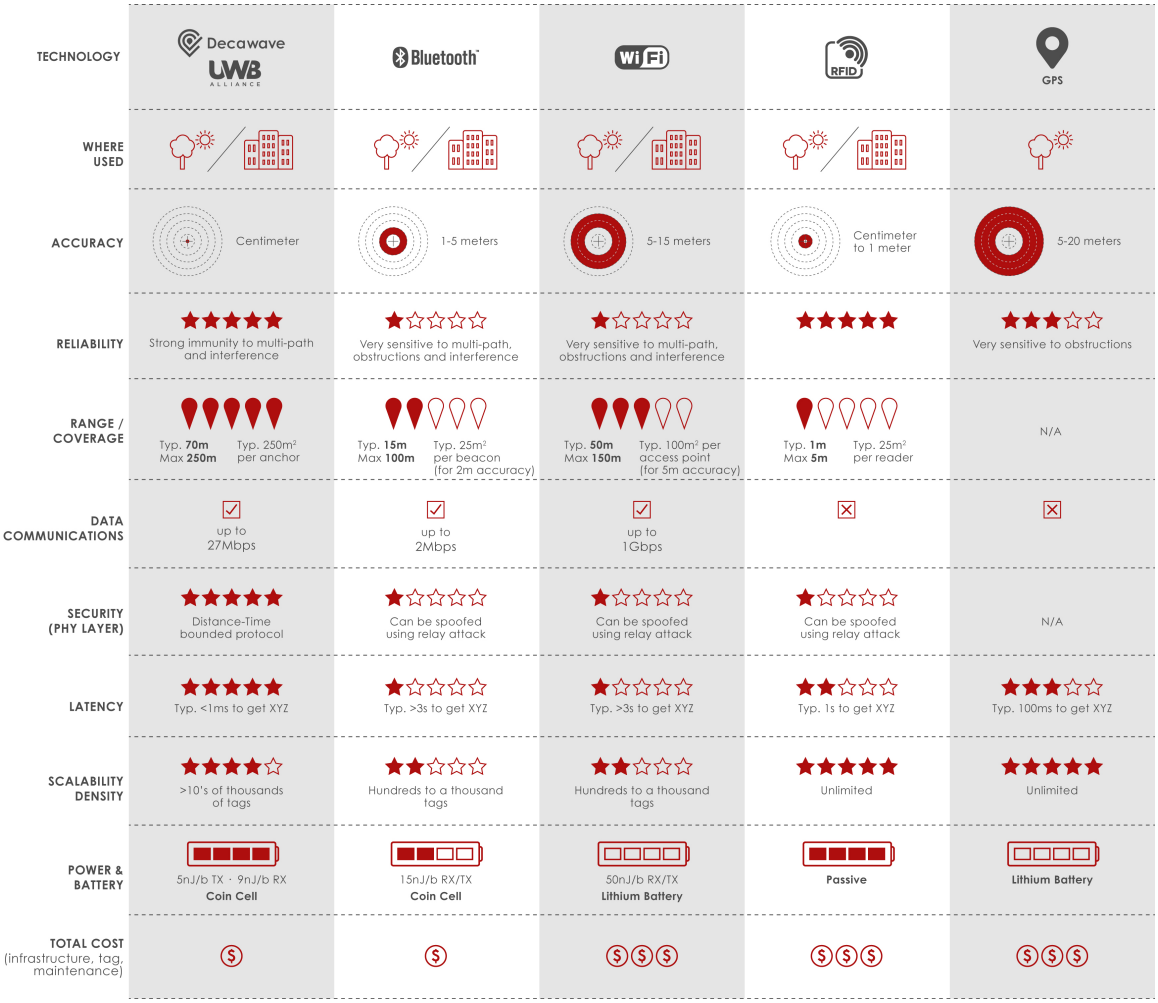


Figura 11: Comparación de tecnología UWB, Bluetooth, Wifi, RFID y GPS [10]

6.3.2. Funcionamiento

Para poder implementar un sistema RTLS con los módulos de Decawave, inicialmente se debe configurar cada módulo DWM1001C como *anchor* o *tag*. Siendo los anchores los dispositivos lectores estáticos que se van a encargar de leer la información enviada por los

tags. Estos últimos son los que se deben colocar en el objeto que se requiere monitorear. La comunicación entre ellos es por medio de UWB y al recibir el tiempo que se tarda en llegar la señal de los *tags* a los *anchors* se determina a que distancia se encuentran. Finalmente, la información se manda a una computadora o celular para poder visualizar la posición de los objetos. [10]

En la Figura 12 se visualiza un diagrama de cómo funciona un RTLS utilizando módulos DWM1001. En este caso se tiene 2 *tags* y 8 *anchors*, también tiene un *gateway* que permite mandar la información recopilada a una computadora. Adicionalmente, esta tecnología permite mandar información de los *tags* directamente a un dispositivo Android por medio Bluetooth.

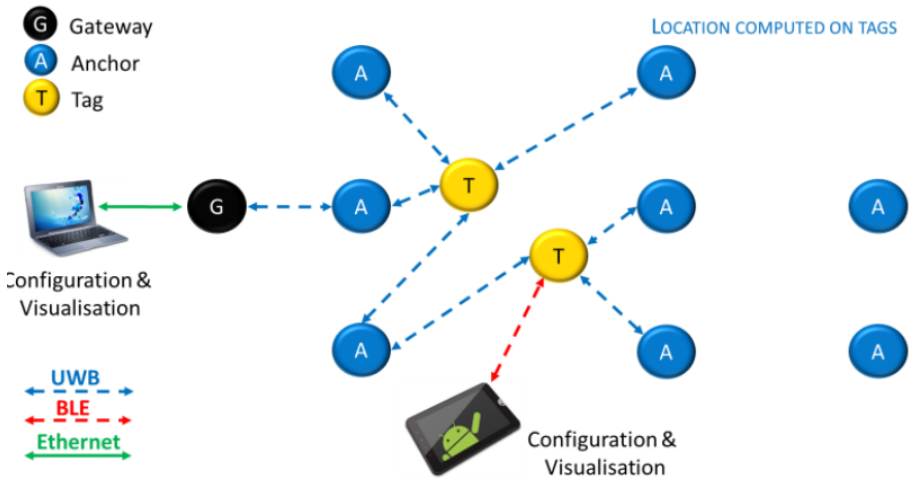


Figura 12: Diagrama de funcionamiento de un RTLS con módulos DWM1001C [10]

6.4. Esquema de codificación TLV (Type - Length - Value)

En español se conoce como Tipo - Longitud - Valor. Este esquema de codificación se utiliza dentro de los protocolos de comunicación para trasmitir y recibir información. La información TLV se compone de 3 partes: [11]

1. Tipo (*Type*): Indica que tipo de información es.
2. Longitud (*Length*): Indica la longitud del valor, es decir, cuantos bytes son del dato que realmente interesa.
3. Valor (*Value*): Es la información o dato que se requiere.

En la Figura 13 se observa un ejemplo del comando *dwm_ gpio_ cfg_ output* utilizado en los módulos DWM1001 para indicar que se quiere colocar en HIGH el pin 13. Dado esto,el primer hexadecimal indica el tipo de instrucción, el segundo indica que la longitud es de 2 bytes y los siguientes 2 bytes son el valor como tal.

TLV request			
Type	Length	Value	
		gpio_idx	gpio_value
0x28	0x02	0x0d	0x01

Figura 13: Ejemplo del formato TLV con comandos del módulo DWM1001 [10]

6.5. Formato endian

Este es un atributo de los datos que describe el orden de los bytes. Al intercambiar datos de varios bytes se debe conocer el convenio de clasificación, de otra forma se puede malinterpretar la información. [12]

A continuación se presentan los formatos de orden de bytes.

6.5.1. Big endian

En este el byte más significativo se almacena en primer lugar, los demás bytes van en orden de significado descendente. Por ejemplo, para una palabra de cuatro bytes, el orden de bytes es 0, 1, 2, 3. Para una palabra de dos bytes, es 0, 1. [12]

6.5.2. Little endian

En este el byte menos significativo se almacena en primer lugar, los demás bytes van en orden de significado ascendente. Por ejemplo, para una palabra de cuatro bytes, el orden de bytes es 3, 2, 1, 0. Para una palabra de dos bytes, es 1, 0. [12]

En la Figura 14 se puede observar de forma gráfica las diferencias entre ambos formatos de orden de bytes.

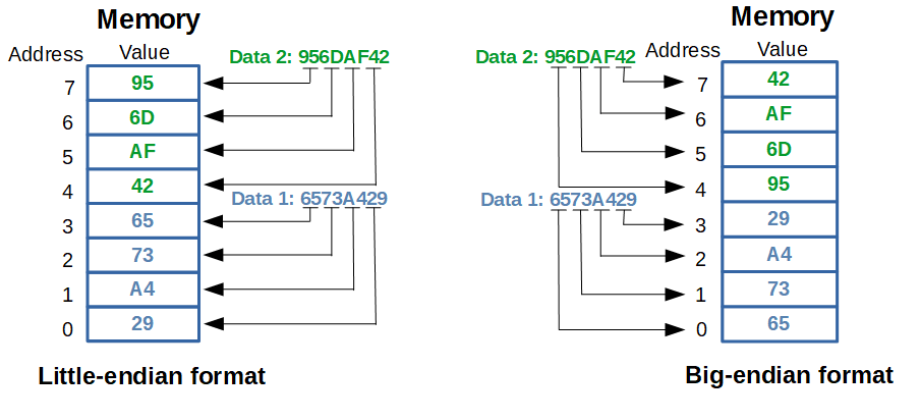


Figura 14: Comparación Big endian y Little endian [13]

6.6. ROS - Robotic Operating System

Según [14]: *El sistema operativo de robot (ROS) es un conjunto de bibliotecas de software y herramientas que lo ayudan a crear aplicaciones de robot. Desde controladores hasta algoritmos de última generación y potentes herramientas para desarrolladores.*

ROS brinda los servicios estándar de un sistema operativo como el control de dispositivos de bajo nivel, trasmisión de mensajes entre procesos, mantenimiento de paquetes e implementaciones de funciones. Se basa en una arquitectura de grafos donde el procesamiento toma lugar en los nodos.

ROS está orientado para un sistema UNIX (Ubuntu (Linux)) principalmente. Sin embargo, ahora con ROS2 es posible utilizar otros sistemas operativos como Fedora, Mac OS X, Arch, Gentoo, OpenSUSE, Slackware, Debian o Microsoft Windows, pero son considerados como 'experimentales'. Con estos no se tienen las mismas funciones que al utilizarlo con Linux.

En ROS se conocen como nodos a los programas ejecutables que realizan una tarea específica. Estos pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros. Cada función o tarea que se requiere que realice el robot se realiza en un nodo diferente. Entre ellos se pueden comunicar mandando mensajes por medio de *topics*. Una vez teniendo todo los nodos necesarios, para que se pueda unificar todo se requiere de un proceso llamado *ROS Master*. Este se encarga de registrar los nodos y configurar la comunicación de nodo a nodo para los *topics*. [15].

En la Figura 15 es posible ver como la comunicación entre 2 nodos en ROS. Esa es la comunicación más básica, pero para generar proyectos en ROS, se sigue el mismo esquema.

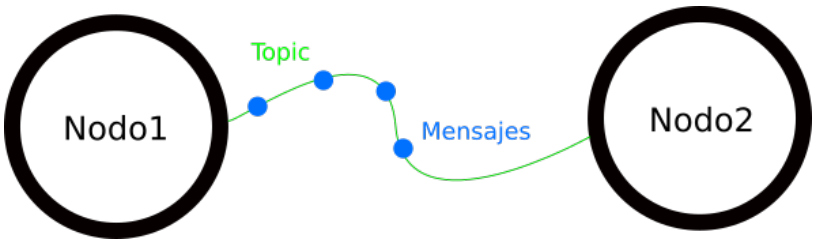


Figura 15: Diagrama de funcionamiento de nodos en ROS [16]

6.7. Máquina virtual

Una máquina virtual es un ordenador de software que brinda la misma funcionalidad que un ordenador físico. Estas ejecutan aplicaciones, programas y un sistema operativo. En otras palabras son archivos informáticos que se ejecutan en un ordenados físico y se comportan como uno. Son un un sistema independiente.

La máquinas virtuales se crean en un entorno informático denominado *host*. En un *host* pueden existir varias máquinas simultáneamente las cuales pueden realizar tareas específicas que podrían ser de riesgo si se ejecutaran en el *host*. Ejemplos de estas tareas es el acceso

a datos con virus o pruebas de sistemas operativos. Como las máquinas están aisladas del resto del sistema, el software de su interior no afecta al ordenador físico. [17]. En la Figura 16 se ve una representación animada de lo que significa tener varias máquinas virtuales dentro de un mismo ordenador físico.

Ventajas

- Tiene opciones de recuperación ante algún desastre o modificación de aplicaciones
- Son de gestión y mantención fácil.
- Permiten ejecutar varios sistemas operativos en un mismo ordenador

Desventajas

- Ejecutar varias máquinas virtuales en un mismo ordenador físico puede generar que el rendimiento sea inestable.
- Son menos eficientes y más lentas que una máquina física

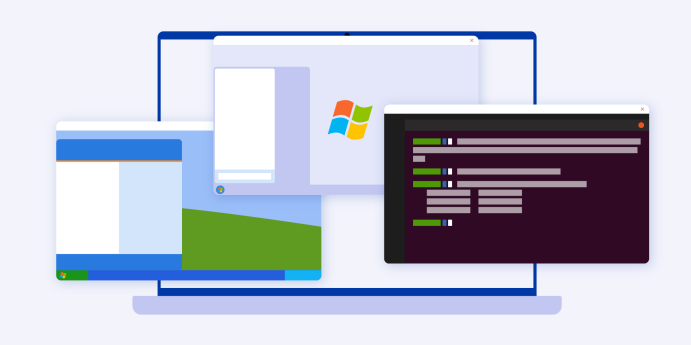


Figura 16: Representación gráfica de un ordenador físico con varias máquinas virtuales [18]

6.8. Optitrack

Es un sistema de captura de movimiento de precisión. Puede servir para producción virtual, ciencia del movimiento, animación, robótica y realidad virtual. Optitrack ofrece la combinación ideal entre precisión de medición y flujos de trabajo simples y fáciles de usar. Esto brindan a los investigadores, ingenieros y biomecánicos datos de seguimiento 3D ideales para estudiar. 18 [19]

Sus cámaras van desde la serie Flex más asequible y ampliamente utilizada de la industria hasta la serie Prime de gran volumen con mejor rendimiento, esta última se pude observar en la Figura 17. Las cámaras ofrecen el seguimiento de objetos y personas en tiempo real con el mejor rendimiento disponible en la actualidad. Ofrecen una exactitud de pocisión de $\pm 0.2mm$, una latencia de $< 9ms$ y una exactitud de rotación de $\pm 0.1deg$.

Este sistema se puede utilizar para crear sistemas de localización en tiempo real por medio de colocar un marker en el objeto que se quiere localizar y las cámaras en conjunto con el software brindan las coordenadas x, y, z & la orientación del objeto en cuestión como se puede observar en la Figura 18

Figura 17: Cámaras prime utilizadas en el sistema de Optitrack [19]

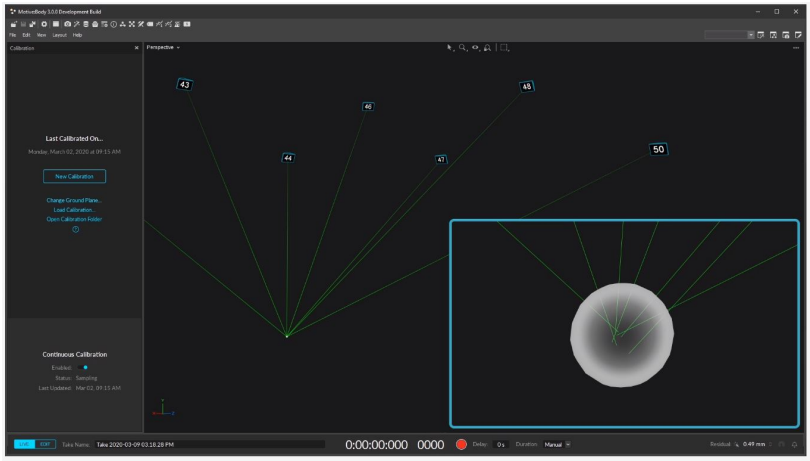


Figura 18: Localización en tiempo real con el sistema de Optitrack [19]

6.9. Desviación estándar

La desviación estándar es una medida de la dispersión de los datos, entre más grande es la dispersión mayor es la desviación estándar. Si no hay ninguna variación en los datos, es decir, son todos iguales, la desviación estándar sería cero. Esta cuantifica la dispersión alrededor de la media aritmética, indica de la media de distancias que tienen los datos respecto de su media aritmética.[20].

Para poder calcular la desviación estándar se utiliza la fórmula mostrada en la Figura 19 en donde se explica que significa cada variable y tiene una representación gráfica de las

operaciones que se realizan. Es bueno conocer la fórmula, sin embargo, hoy en día se tiene la facilidad de contar con programas como Excel que lo calculan automáticamente.

Figura 19: Fórmula y explicación de variables de la desviación estándar o típica [21]

CAPÍTULO 7

Módulos DWM1001-DEV

Para poder desarrollar el sistema de localización en tiempo real se utilizaron los módulos MDEK1001 – DEV que ofrece la empresa Qorvo como se puede observar en la Figura 20. Estos módulos cuentan con tecnología de banda ultra-ancha que permite la comunicación entre ellos. La empresa brinda kits de 12 módulos para poder crear una red de localización con una cantidad suficiente y además, que estén protegidos por el estuche.

Figura 20: Módulos MDK1001 - DEV de la empresa Qorvo utilizados en el trabajo [10]

7.1. Fuentes de alimentación de los módulos

Los módulos requieren de 5V y como máximo 500mA para funcionar, por lo que hay varias formas de alimentarlos. Entre estas pueden ser una batería, una batería portátil (*power bank*), conexión a la computadora y conexión directa al toma corriente. Si se escoge alguna de las últimas tres opciones es necesario contar con un cable micro USB por cada módulo.

Se optó por instalarlas con cables micro USB y conexión directa a un tomacorriente. Esto con la intención de tener que desmontar las baterías diariamente. Además se tuvo problemas con la corriente por lo que las baterías ya no eran opción viable. (Esto se explica a mayor detalle en el capítulo 9).

7.2. Montaje de módulos

7.2.1. Etiquetas física de los módulos

Para un sistema de localización basado en *anchors* y *tags* es necesario identificar correctamente cada módulo, tanto en la app como en físico también. Además, estos son todos iguales físicamente, por lo que es difícil mantener el control de todos.

Dado esto se decidió enumerar los módulos y colocarles un pequeño pedazo de papel para diferenciarlos, esto se puede observar en la Figura 21. De esta manera era posible identificarlos desde lejos y tener mejor visualización de la red.

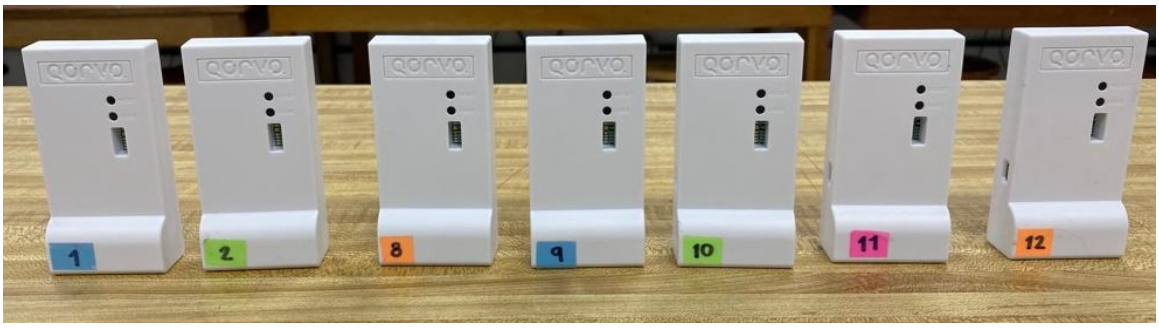


Figura 21: Módulos etiquetados para poder identificarlos

7.2.2. Cantidad y posición de módulos para una red

Para que un sistema de localización con módulos MDEK1001 funciones, el proveedor indica que deben ser mínimo 4 módulos configurados como *anchors*. Además, estos deben estar colocados idealmente, en forma cuadrada o rectangular alrededor del área. En este trabajo no se probaron geometrías distintas a las mencionadas.

Dada estas indicaciones los módulos se colocaron en formas rectangulares alrededor de los salones en que se hicieron pruebas. Se identificó que el sistema funciona de forma correcta

con 6 módulos colocados de forma genérica como se muestra en la Figura 22 en donde A significa *Anchor* y la T significa *Tag*.

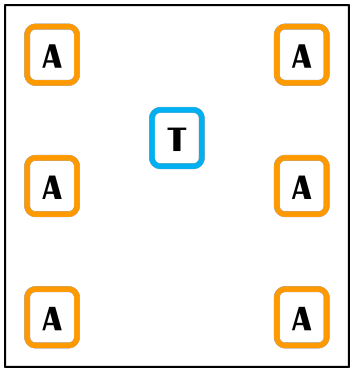


Figura 22: Forma rectangular en que se colocan los módulos para formar una red

7.3. Creación de una red

Para crear una red con los módulos se requiere de configurar cada uno de los módulos y agregarlos a la red. Esto se puede realizar desde una aplicación de Android o en la computadora desde el *Shell Command*.

En este trabajo de graduación se realizó por medio de la aplicación Android y se explica a continuación.

7.3.1. Configuración de los módulos

Los pasos se pueden seguir en la guía rápida que brinda el proveedor Qorvo. A continuación se brinda el link y código QR para poder verla. Link para ver guía rápido. [10]

Figura 23: Código QR para ver guía rápida de instalación

También es necesario descargar la aplicación Android. A continuación se brinda el link y código QR para poder descargarla. El archivo .apk se debe descargar directamente en el

dispositivo Android. Link para descargar aplicación. [10]

Figura 24: Código QR para archivo de aplicación Android

Para poder configurar la red se siguieron estos pasos:

1. Se seleccionó el módulo que iba a ser *tag* y los otros que iban a ser los *anchors*. Dependiendo de la prueba que era se tenían 4 o 6 *anchors*.
2. Se colocaron los *anchors* alrededor de toda el área en forma cuadrada o rectangular.
3. Se le colocaron los cables micro USB a cada módulo y se conectaron a una fuente de alimentación.
4. Se conectó el *tag* a la computadora.
5. Se descargó la aplicación en un teléfono Android. Esta aplicación no se encuentra en Google Play, sino que debe descargarse el archivo .apk desde el navegador.
6. Se siguieron los pasos de la guía para crear una red de la guía y se le colocó el nombre de "Prueba UVG #", en donde el # es el número de prueba realizada en ese momento.
7. Seguido de esto se siguió los pasos para configurar cada uno de los módulos como *anchors* y se les colocó el de A# en donde el # es el mismo número que tenían en su etiqueta física.
8. Se siguió los pasos para configurar el módulo como *tag*, este se identificó como T#, en donde # corresponde al número de la etiqueta física.
9. Es importante colocar la misma tasa de actualización normal y estática en la configuración del *tag*. Estas fueron de $100ms/Hz$

Una vez terminados estos pasos, la red ya está lista.

7.3.2. Verificación de funcionamiento del sistema en aplicación Android

Para revisar que todos los módulos de la red están conectados y funcionando correctamente hay 2 formas de hacerlo dentro de la aplicación.

1. **Listado de módulos**
   En esta forma la aplicación lista todos los módulos encontrados dentro de la red en cuestión. Indica si están configurados como *anchors* o como *tags* y si están activos o no. Para revisar que realmente están activos debe aparecer las 3 líneas de de señal y el botón de edición está activo como se muestra en la Figura 25. Adicionalmente, si estos están activos se puede presionar sobre ellos y se despliega toda la información de ese módulos como se muestra en la Figura 26. En este caso se está mostrando la información del módulo que es Iniciador:. Por eso, en donde dice *Initiator* aparece *enable*. En el resto de *anchors* dice *disabled*.

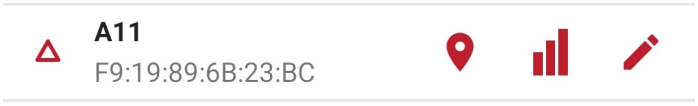


Figura 25: Visualización de un módulo activo en la aplicación

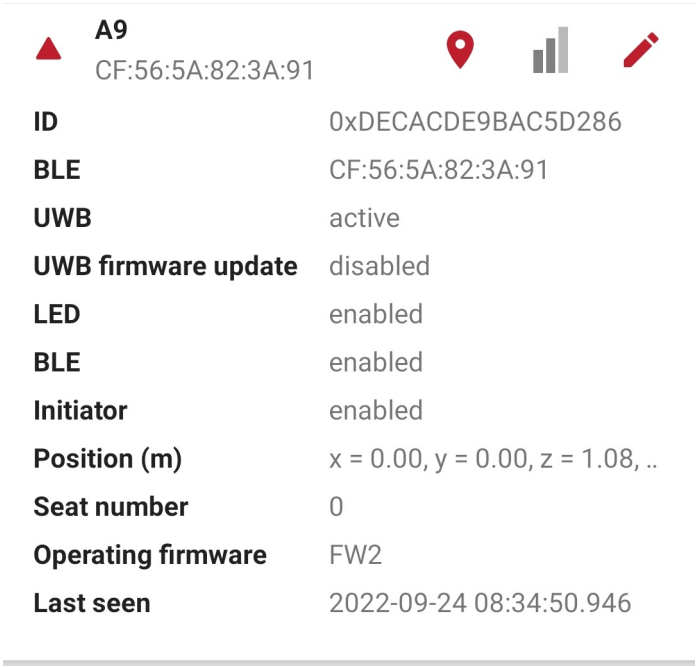


Figura 26: Visualización de información de un módulo activo en la aplicación

En la forma de listado lo que se muestra es la posición en coordenadas y son las del *tag* va variando conforme se va movilizando. Esta forma realmente sirve para la configuración y visualización de la información de todos los módulos, no tanto para conocer la posición.

En la Figura 27 se puede observar el listado de módulos dentro de la red llamada "*Prueba 4 UVG*"la cual está conformada por 6 *anchors* y 1 *tags*.

2. **Cuadrícula gráfica**
   En esta forma la aplicación muestra una visualización gráfica de los módulos en una cuadrícula como se puede observar en la Figura 28.

   En base en cómo fueron configurados, es decir, si son *anchors* o si son *tags* tienen distinta geometría. Los primeros se visualizan como un triangulo y los segundos como

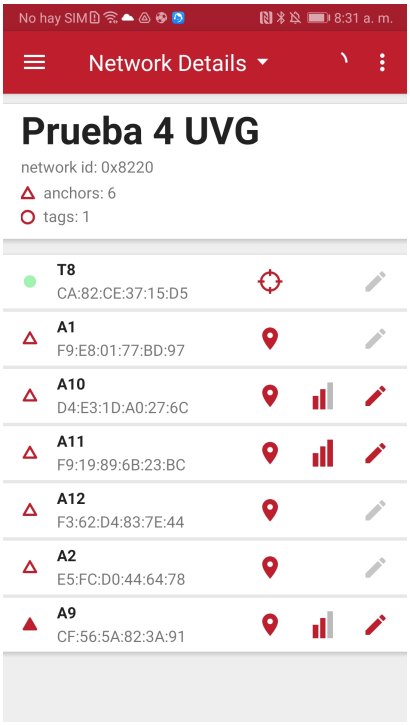


Figura 27: Visualización de listado de módulos en aplicación Android

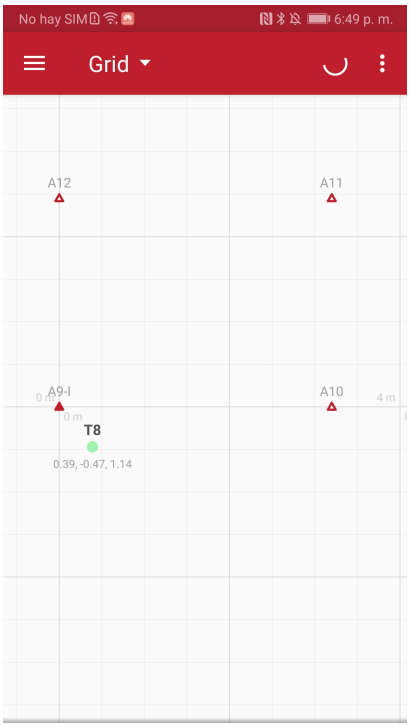


Figura 28: Visualización de los de módulos en forma gráfica en aplicación Android

un círculo, el cuál es el que se va moviendo conforme e módulo físico también lo hace. Además, muestra debajo del círculo las coordenadas X, Y & Z de su posición actual.

Adicionalmente, se puede habilitar la opción de observar la distancia a la que esta el *tag* de cada uno de los *anchors* por medio de líneas rectas como se ve en la Figura 29. Estas se van actualizando, al igual que la posición, conforme se mueve en tiempo real el módulo.

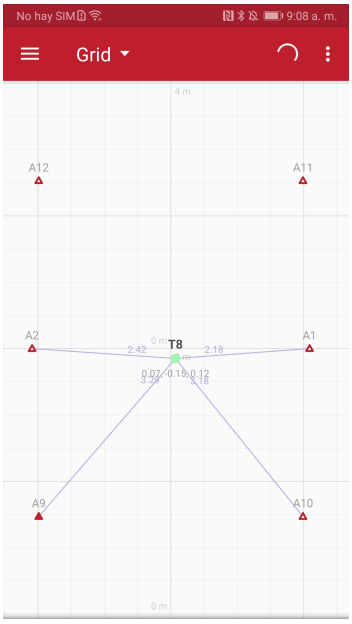


Figura 29: Visualización de los de módulos en forma gráfica en aplicación con líneas de distancia

Con estas dos formas de visualizar en la aplicación fue sencillo corroborar que lo módulos funcionaban, si estaban activos, que configuración tenían y si la posición del *tag* en la aplicación era congruente a donde se encontraba el módulo en la vida real.

7.4. Obtención de datos

En este trabajo de graduación se trabajó en el sistema operativo de Linux: ya que era un requisito para unirlo con ROS. Para hacerlo, se trabajó en una máquina virtual con Ubuntu: 20.4 (Esto se explicará a mayor detalle en el capítulo 10)

Dado esto, a continuación se explica como se hizo la obtención de datos dentro de ese sistema.

7.4.1. Shell comand

Para poder recibir los datos de esta manera se necesita una terminal serial para poder interactuar con los módulos. En este caso se utilizó *Moserial termial* en donde la pantalla inicial se muestra en la Figura 30.

Para poder comunicarse con los módulos es necesario conectar a la computadora el módulo que está configurado como *tag*. Dentro de la terminal se escoge el puerto USB donde

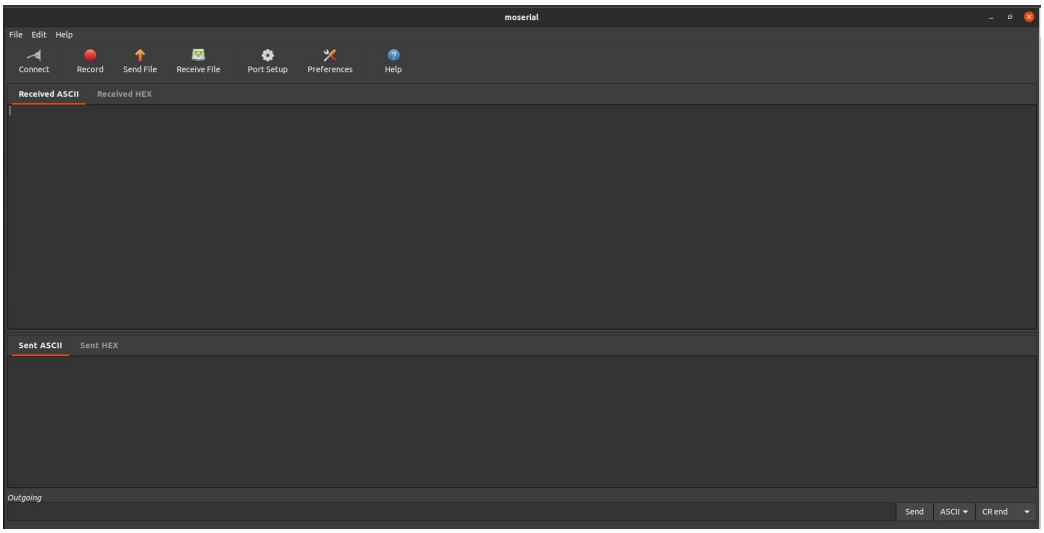


Figura 30: Pantalla inicial de la terminal serial *Moserial*

está el módulo y se coloca que se quiere recibir data en formato ASCII como se muestra en la Figura 31. También se debe colocar que se quiere enviar data en formato ASCII y con una secuencia de **CR end** como se muestra en la Figura 32.

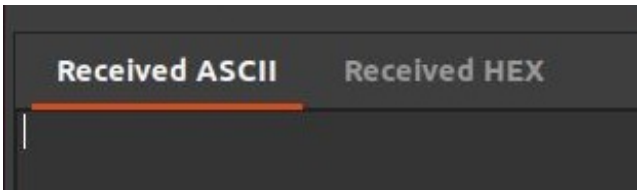


Figura 31: Formato en que se quiere recibir la data de los módulos en *Moserial*

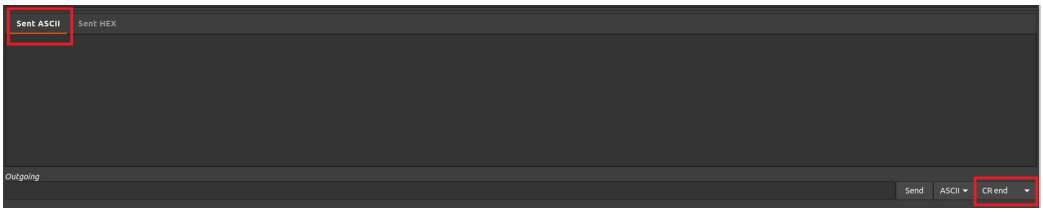


Figura 32: Formato en que se quiere mandar la data a los módulos en *Moserial*

Luego, dentro de la terminal presionar conectar y esperar que lo haga. Cuando ya está conectado se debe presionar *enter* 2 veces en menos de 1 segundo para que el módulo lo reciba y entre en el modo *Shell command*. Si ingresó correctamente al modo debería aparecer un menú como el que se muestra en la Figura 33.

Lo primero que se recomienda hacer es escribir el comando *help* o un signo de interrogación para que aparezca un listado de todos los comando que se tienen disponible.

Algunos comandos importantes son:

- si: Información del módulo.

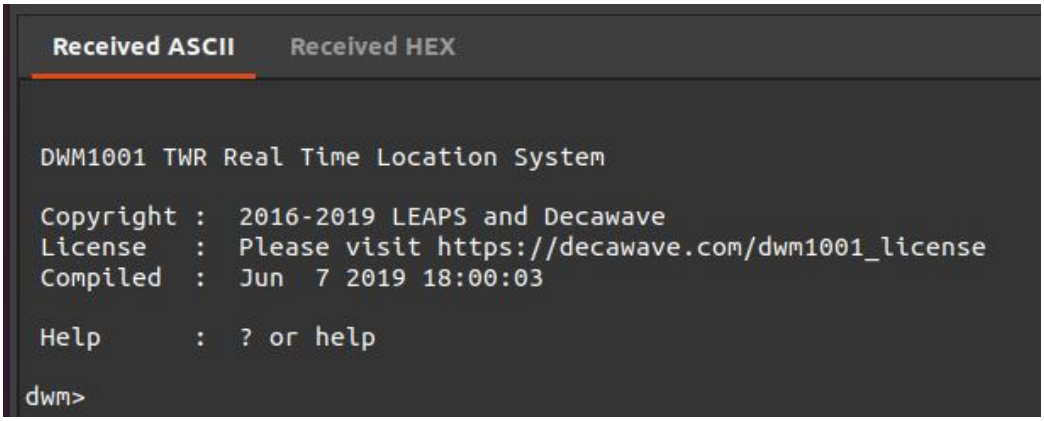


Figura 33: Menú de los módulos DWM en el modo Shell command

- reset: Reiniciar el sistema.
- frst: Reset de fábrica del módulo.
- les: Muestra la distancia del *tag* a cada uno de los *anchors* y su posición actual.
- apg: Muestra la posición en milímetros del *tag*.
- nmg: muestra si el módulo es *tag* o *anchor*
- la: Muestra lista de todos los *anchors* que conforman la red, su posición e información de cada uno de ellos.
- quit: se sale del modo.

El comando ***apg*** es el que nos sirve para visualizar la posición en que se encuentra el *tag*. Es importante que toda los *anchors* estén conectados y activos para poder obtener las coordenadas.

El formato en que muestra la posición es el siguiente:

$$apg : x : 0\ y : 0\ z : 0\ qf : 100$$

En donde cada coordenada está dada en milímetros y el último dato indica el factor de calidad, es decir, en un rango de 1-100 que tan buena es la medida que se está proporcionando. En la Figura 34 se muestra un ejemplo de una medición realizada.

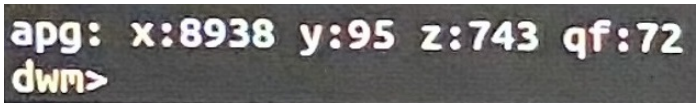


Figura 34: Formato en que se recibe la posición del *tag* en el shell command

Si se quiere corroborar la actualización en tiempo real se manda el comando *apg* y luego se puede dejar presionado la tecla *enter* ya que esto permite repetir el comando anterior. Entonces, se puede ir moviendo el módulo y va a ir apareciendo en la consola las posiciones por las que va pasando. Esta forma fue la que se utilizó para corroborar la actualización de los datos en tiempo real. (se explica en el capítulo 9).

7.4.2. Comunicación con formato TLV

Esta otra forma de obtención de datos que provee la tecnología de los módulos DWM consiste en comunicación UART: o SPI: utilizando el formato TLV: (Tipo - Longitud - Valor). Se debe enviar un comando en ese formato y los módulos se encargan de enviar de regreso otro dato en el mismo formato con la información solicitada.

La forma en que presenta los datos esta tecnología es en bytes en sistema hexadecimal. Un único byte para el tipo de dato, un único byte para la longitud. La cantidad de bytes para los valores varía en base a los que sean necesarios. Esta cantidad de bytes es la que indica el 2do byte de longitud.

En la Figura 35 a continuación se muestra un ejemplo del formato TLV de un comando que utilizan los módulos DWM.

TLV request			
Type	Length	Value	
		gpio_idx	gpio_value
0x28	0x02	0x0d	0x01

Figura 35: Ejemplo de un comando en formato TLV utilizado en los módulos DWM

Obtener la posición

Para poder obtener la posición del *tag* se debe mandar el comando: $0x02$ $0x00$, el cual indica que el tipo es 0x02 y como no se va a enviar ningún dato la longitud es 0.

Como respuesta se obtiene primero la información de si hay un error o no Por lo que se recibe $0x41$ $0x01$ $0x00$, en donde, si el último byte es 0 significa que no hay ningún error. Seguido de esos 3 bytes ya se recibe la data sobre la posición por lo que se obtiene como respuesta el byte de tipo: $0x41$ y el byte de longitud $0x0D$. Este último indica la data de la posición va a constar de 13 bytes. En estos 13 bytes los primeros 4 representan la coordenada x, los siguientes 4 representan la coordenada y, lo siguientes 4 representan la coordenada y y el último byte representa el factor de calidad. A continuación se muestra en la Figura 36 un ejemplo del comando que se envía y la respuesta, en formato TLV, que se recibe para obtener la posición.

Es importante mencionar que los bytes obtenidos del valor de la posición están en Little endian:. Esto significa que de esos 4 bytes de cada coordenada el byte más a la derecha es el más significativo. Por lo que, para obtener la posición en milímetros hay que hacer la conversión tomando esto en cuenta.

A continuación se muestra un ejemplo de como convertir 4 bytes en un valor en decimales con 2 procedimientos distintos. Tomando como bytes de ejemplo $0x01$ $0x0A$ $0x20$ $0x0D$

Procedimiento 1

En la Figura 37 se puede observar como se realiza la conversión tomando cada Nibble:

Example:

TLV request	
Type	**Length**
0x02	0x00

TLV response								
Type	**Length**	**Value**	**Type**	**Length**	**Value**			
		err_code			**Position**			
					32 bit value in little endian is x coordinate in millimeters	32 bit value in little endian is y coordinate in millimeters	32 bit value in little endian is z coordinate in millimeters	8 bit value is quality factor in percents (0-100)
0x40	0x01	0x00	0x41	0x0D	0x79 0x00 0x00 0x00 0x32 0x00 0x00 0x00 0xfb 0x00 0x00 0x00 0x64			

Figura 36: Ejemplo comando y respuesta para obtener la posición del módulo

por separado.

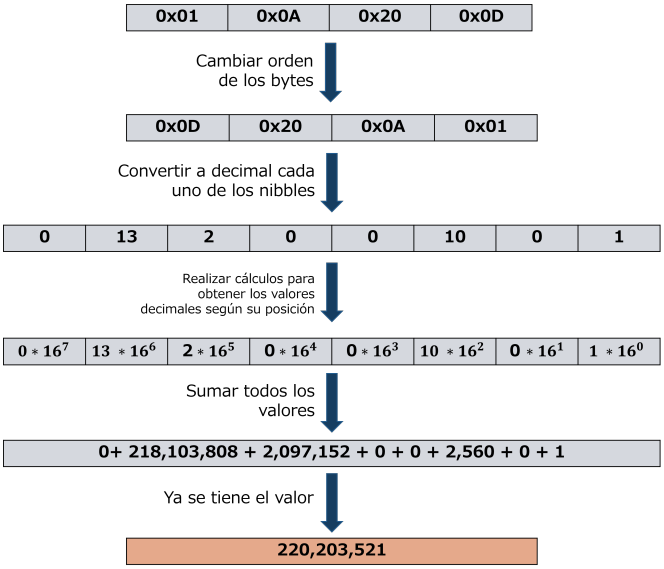


Figura 37: Ejemplo de procedimiento 1 de conversión de un valor de 4 bytes a decimal en Little endian

Procedimiento 2

En la Figura 38 se puede observar como se realiza la conversión tomando cada byte por separado.

Implementación en el proyecto

Uno de los objetivos de este trabajo de graduación era hacer el sistema de localización compatible con ROS. Para eso se debía generar un archivo en C: o en Python: que fuera capaz de obtener la posición del *tag*.

Se decidió hacer el archivo en Python y con una comunicación serial con el *tag*. Se realizó

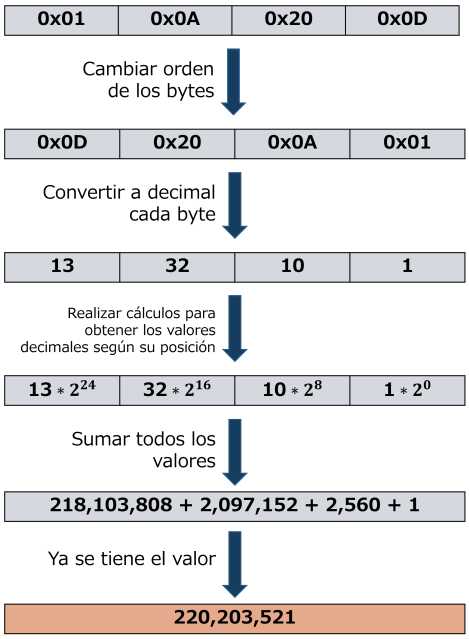


Figura 38: Ejemplo de procedimiento 2 de conversión de un valor de 4 bytes a decimal en Little endian

una función para obtener la posición de módulo llamada *dwm_pos_get*. Esta función se encargaba de enviar los bytes de comando $0x02$ $0x00$ y luego leer los 18 bytes que el módulo respondía. De estos 18, los primeros 5 eran $0x41$ $0x01$ $0x00$ $0x41$ $0x0D$, que indicaban que no había error y que los siguientes 13 bytes eran de las coordenadas y el factor de calidad.

Para obtener las coordenadas en milímetros se creó una función llamada *get_coord* en donde se utilizó el procedimiento 2 mostrado en la Figura 38 para convertir cada coordenada. Adicionalmente, se hizo el código de la conversión a metros o centímetro, lo cual se puede modificar según se requiera. En la Figura 39 se muestra como se recibían los dato en la consola de Python.

Example:

TLV request	
Type	**Length**
0x02	0x00

TLV response								
Type	**Length**	**Value**	**Type**	**Length**	**Value**			
		err_code			**Position**			
					32 bit value in little endian is x coordinate in millimeters	32 bit value in little endian is y coordinate in millimeters	32 bit value in little endian is z coordinate in millimeters	8 bit value is quality factor in percents (0-100)
0x40	0x01	0x00	0x41	0x0D	0x79 0x00 0x00 0x00 0x32 0x00 0x00 0x00 0xfb 0x00 0x00 0x00 0x64			

Figura 39: Obtención de datos por comunicación serial y formato TLV

Para poder recibir la posición en tiempo real lo que se hizo fue colocar esas funciones

dentro de un *Loop while:*. Esto generaba que se estuviera mandando repetidamente el comando de posición al módulo y este regresaba los 18 bytes constantemente. Luego, la otra función encargaba de hacer la conversión a milímetros y la programación ya era capaz de devolver la posición como se requería para hacerlo compatible con ROS.

El código de las funciones se encuentran en anexos.

CAPÍTULO 8

Pruebas del RTLS con los módulos MDEK1001

8.1. Prueba 1: Verificación de funcionamiento de la red

El objetivo de la primera prueba era comprobar si los módulos funcionaban y si se mostraba correctamente la posición desde la aplicación. Para esto se creo una pequeña red con 4 *anchors* y un *tag* para poder visualizarla en la aplicación. A continuación se muestra la metodología de esta prueba.

8.1.1. Montaje

Para la prueba se implementó una red, con la forma y medidas que se muestran en la Figura 40. Los módulos destinados a ser *anchors* se colocaron sobre unos bancos de altura $0.513m$ y se alimentaron conectándolos al tomacorriente con ayuda de extensiones como se muestra en la Figura 41.

8.1.2. Configuración

Una vez colocados los módulos que en la posición deseada, se prosiguió a configurarlos desde la app Android. Para esto se siguieron los pasos explicados en la sección de configuración del capítulo anterior. En este caso los módulos que fueron *anchors* tenían las etiquetas de A1, A2, A3 y A4 como se puede observar en la Figura 44. El *tag* tenía la etiqueta de T8.

Con esos pasos se consiguió crear una red llamada "Prueba1 - UVG"la cual consistía en 4 *anchors* (A1, A2 ,A3 y A4) y 1 *tag* (T8). Siendo el *anchor* A1 el iniciador.

Dentro de la configuración de *tag* se dejó los parámetros de tasa de actualización normal

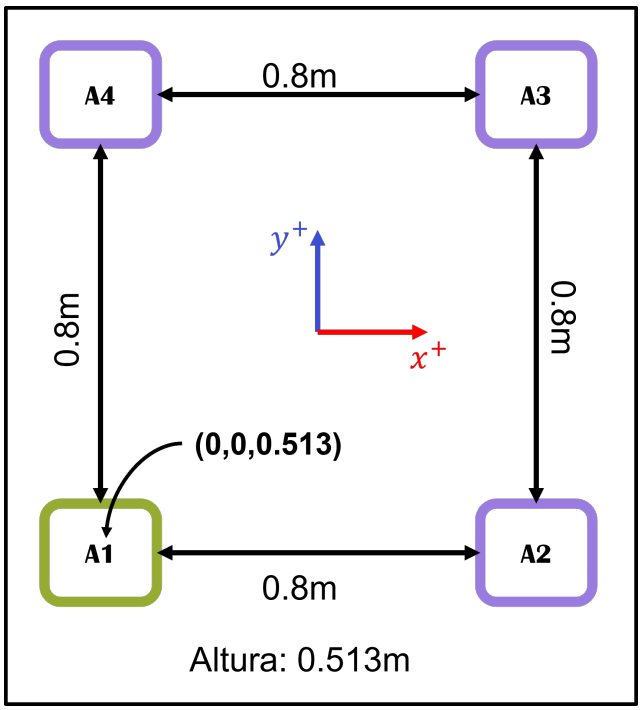


Figura 40: Forma y dimensiones de la red en la prueba 1

(a) Montaje visto desde arriba

(b) Montaje visto desde el lado

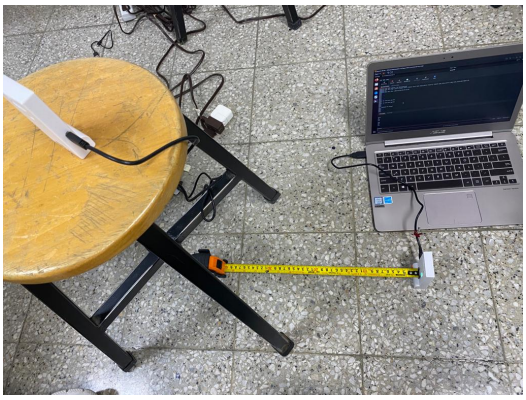
Figura 41: Montaje real de la prueba 1

y estática como venía predeterminado. Esto fue con valores de: $100ms/Hz$ y $10s/0.1Hz$ respectivamente.

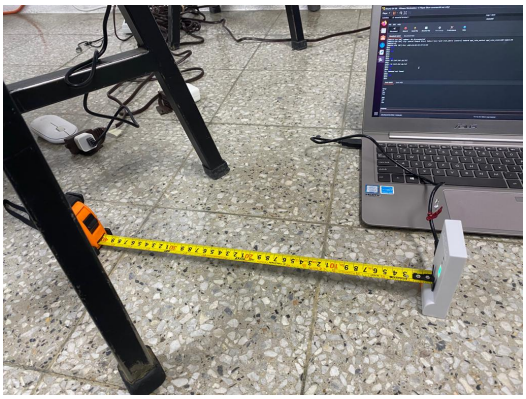
8.1.3. Verificación de la posición

Para hacer las pruebas y verificar que la posición era la correcta se conectó el módulo de *tag* a la computadora para alimentarlo y comunicarse con este. Luego se colocó el módulo en la posición $(0.4, 0, 0)$ como se muestra en la Figura42. Seguido de eso se verificó que posición

se mostraba tanto en la aplicación como en el *Shell command*.



(a) Distancia a la que se colocó el módulo visto desde arriba



(b) Distancia medida por el metro

Figura 42: Distancia a la que se colocó el módulo para hacer verificar posición

Primero se verificó los datos desde la aplicación. Dejando el módulo totalmente estático se observó que marcaba una posición de: $(0.36, 0.08, -0.04)$. Lo que indicaba que estaba midiendo la posición de forma acertada. No era totalmente exacto, pero el proovedor hace la indica que lo módulos dan una medida en un radio de 10cm de la medida exacta.

Por otro lado, para probar en el *Shell command* se obtuvo una posición de: $(0.416, 0.0256, 0.0623)$. Dado esto se pudo identificar que había una pequeña diferencia entre las dos formas de obtener valores. Además que en el *Shell command* las coordenadas tenían más números decimales, lo que indica que es más preciso en cuanto las coordenadas.

Con esta prueba fue evidente notar que el poder visualizar la posición de forma gráfica, como en la aplicación, ayuda a entender mejor el espacio.

8.1.4. Prueba en tiempo real

Para poder probar que tan rápido se actualizaban los datos lo que se hizo fue mover el módulo dentro del cuadrado que forma la red. Mientras se iba moviendo el módulo se iba revisando que tan rápido cambiaba las coordenadas y el *tag* gráfico dentro de la cuadrícula en la aplicación.

Se movió en distintas direcciones para hacer pequeñas pruebas e identificar si algún eje tenía más retardo que otro. Se Las pruebas que se realizaron fue:

1. Deslizamiento en línea recta sobre el eje x: Se inició en el *anchor* A1 y se movilizó hasta el *anchor* A2.
2. Deslizamiento en línea recta sobre el eje y: Se inició en el *anchor* A1 y se movilizó hasta el *anchor* A4.
3. Deslizamiento en línea recta horizontal: Se inició en el *anchor* A1 y se movilizó hasta el *anchor* A3.

4. Deslizamiento alrededor de todo el cuadrado: Se inició en el *anchor* A1, luego al *anchor* A2, seguido al *anchor* A3 y por último hasta el *anchor* A4.

5. Deslizamiento aleatorio: Se movió para distintos lados el módulo sin un orden respectivo.

En la siguiente figura se muestra como se fue movilizando el módulo dentro del espacio de la red. Se tenía conectado a la computadora ya que eso permitía encender el módulo y visualizar la información en el *Shell command*.

(a) Movilización de módulo por una parte del área

(b) Distancia medida por el metro

Figura 43: Movilización de módulo por una parte del área

8.1.5. Resultados

Al final de esta prueba se identificó que actualización de los datos en tiempo real no era lo suficientemente rápida, si el módulo se movía de un *anchor* a otro *anchor* este no se actualizaba hasta luego de 3 segundos o más.

Dado que este sistema de localización en tiempo real se quería implementar en el robot Rover UVG, al cual se le tenía que aplicar control punto a punto, una actualización tan lenta como este es inservible.

8.2. Prueba 2: Solucionar la lenta actualización de datos

El objetivo de la segunda prueba era identificar y solucionar el problema de la lenta actualización de los datos.

8.2.1. Montaje

Para la prueba se montó una red, con la forma y medidas que se muestran en la Figura 40. Los módulos destinados a ser *anchors* se colocaron sobre cada CPU de las computadoras

disponibles en el laboratorio. Para conectaros se utilizaron únicamente los cables micro USB directo de los módulos a la computadora como se muestra en la Figura 41.

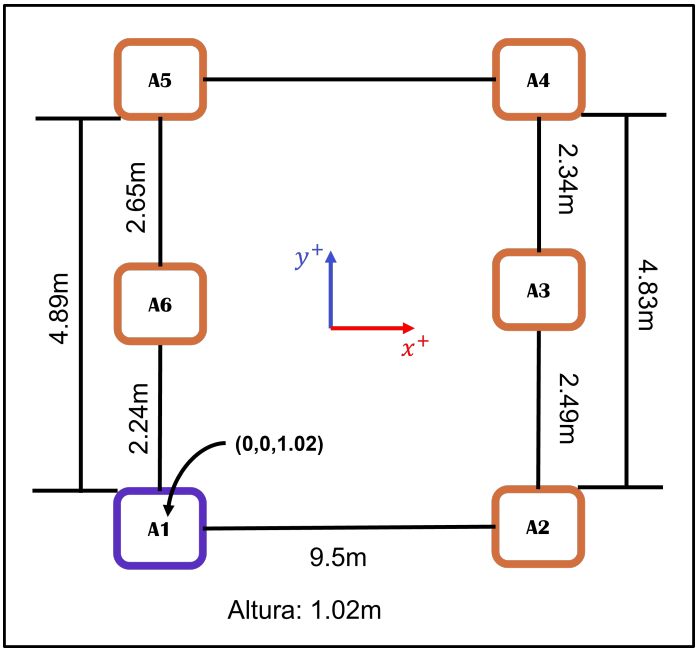
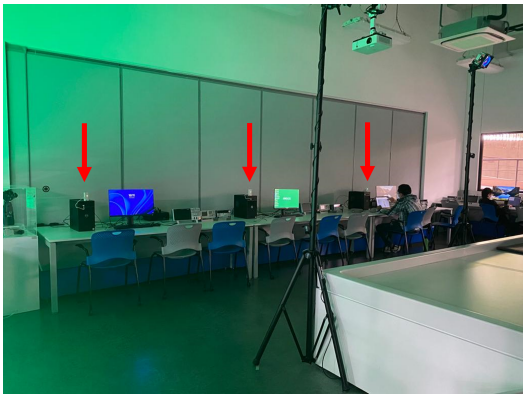


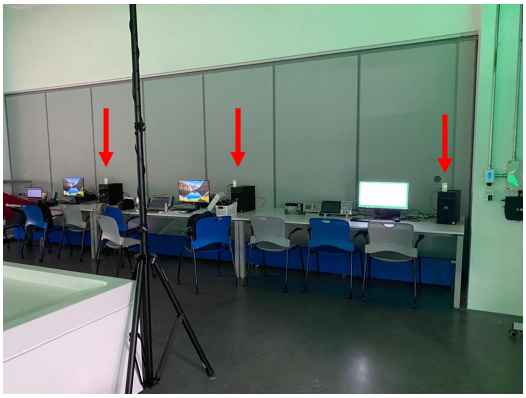
Figura 44: Forma y dimensiones de la red en la prueba 2

Figura 45: Cómo se colocaron los módulos en la prueba 2

A continuación se muestran imágenes de donde fueron colocados los módulos dentro del salón de prueba. La Figura 46b muestra en donde se colocaron los módulos dentro del salón y la Figura 46a muestra los otros 3 módulos colocados del otro lado del salón.



(a) Posición en que se colocaron los módulos del lado izquierdo



(b) Posición en que se colocaron los módulos del lado izquierdo

Figura 46: Posición en donde se colocaron los módulos para la prueba 2

8.2.2. Configuración

Una vez colocados los módulos que en la posición deseada, se prosiguió a configurarlos desde la aplicación Android. Para esto se siguieron los pasos explicados en la sección de configuración del capítulo anterior. En este caso los módulos que fueron *anchors* tenían las etiquetas de A1, A2, A3, A4, A5 y A6 como se puede observar en la Figura 44. El *tag* tenía la etiqueta de T8.

Con esos pasos se consiguió crear una red llamada "Prueba2 - UVG"la cual consistía en 6 *anchors* (A1, A2, A3, A4, A5 y A6) y 1 *tag* (T8). Siendo el *anchor* A1 el iniciador.

Dentro de la configuración de *tag* se cambiaron los parámetros de tasa de actualización normal y estática a valores de: $100ms/Hz$ y $100ms/Hz$ respectivamente. Se identificó que al realizar el cambio de la tasa de actualización normal en la aplicación, la actualización de datos se realizaba cada fracción de segundo y ya no cada 3. Esto significa que se resolvió el problema de la prueba 1.

Esto se debe a que, el DWM1001 tiene incorporado un acelerómetro que detecta cuando el módulo está estático y activa el uso de la tasa de actualización estacionaria. Pero, si esta no es mayor o igual a la normal el módulo no detecta que está inmóvil y por eso la actualización se tarda demasiado.

8.2.3. Pruebas en tiempo real

Para realizar las pruebas en tiempo real se conectó el *tag* a la computadora para alimentarlo y poder comunicarse con el por medio del *Shell command*. Asimismo se abrió la aplicación en el teléfono para poder tener visualización gráfica de donde está el *tag* y que tan rápido se actualiza.

Para poder recibir información en el *Shell command* se conectó el módulo, se conectó con *Moserial*, se presionó 2 veces *enter* para entrar al modo de *Shell command*. Una vez ya nos salió el menú se ingresa el comando: ***apg*** el cual nos regresa las coordenadas x, y & z.

Una vez se ingresó un comando, si se presiona *enter* nuevamente se repite el comando anterior, por lo que ese se utilizó para verificar la actualización de los datos.

Lo que se hizo fue caminar alrededor de todo el salón con la computadora, el módulo y el teléfono para poder verificar de las dos formas, simultáneamente, que valores de las coordenadas se obtenían. Se dejó presionado la tecla *enter* para poder ver en tiempo real como iba cambiando en el *Shell command* y se fue observando la aplicación para ver si se movía el círculo acorde a donde yo me estaba movilizando.

El recorrido inició en el módulo A9, es decir, el *anchor* iniciado. Luego me fui moviendo de *anchor* en *anchor* en la siguiente secuencia: A2 - A3 - A4 - A3 - A2 - A1 - A6 - A5 - A6 - A1. Dado que conocía la medida a la que estaba cada *anchor* del iniciador se podía ir observando si efectivamente la posición iba cambiando y tenía congruencia con las medidas.

A continuación se muestra imágenes de como fue que se tomó la computadora, módulo y teléfono para caminar por todo el salón.

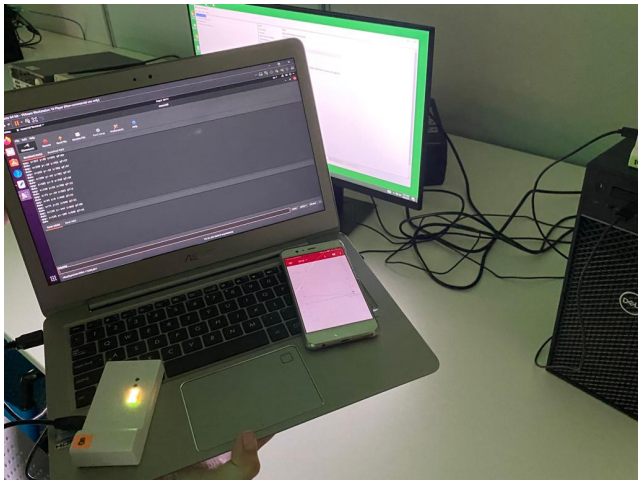


Figura 47: Muestra 1 de como se realizó la prueba en un lado del salón

8.2.4. Resultados

Durante la prueba fue posible observar que el círculo en la cuadrícula de la aplicación ya se movía conforme yo me movía alrededor del salón. Asimismo las coordenadas que se muestran, tanto en la aplicación como en la computadora, iban cambiando conforme yo me movía.

Con esto se puede concluir que es imperativo que el valor de la tasa de actualización estática sea mayor o igual a la normal. En este caso, ambos son $100ms/Hz$ y funcionó correctamente.

Adicionalmente se volvió a verificar si estaba dando valores congruentes a la posición en donde yo me encontraba. Se hizo la prueba de observar que valores daba cerca del *anchor* A1, que es la posición $(0, 0, 1.02)$ y cerca del *anchor* A2 que es la posición $(9.5, 0, 1.02)$.

En la Figura 49a se puede observar que el círculo verde que representa el *tag* se encuentra

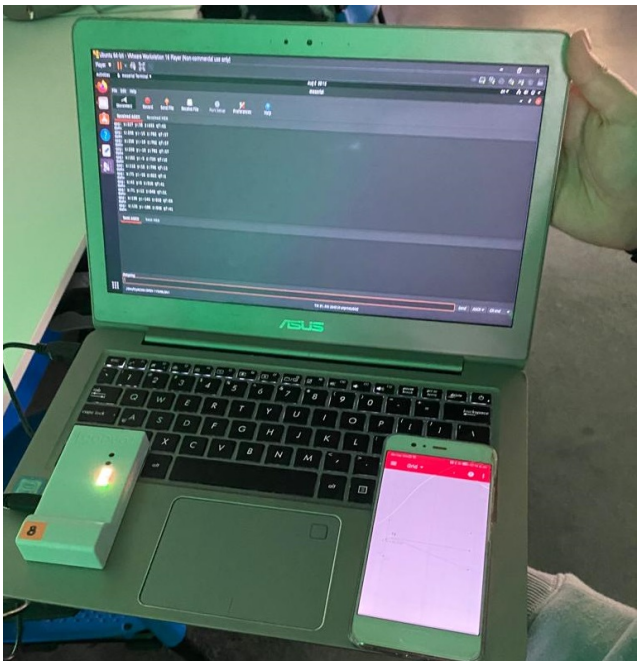
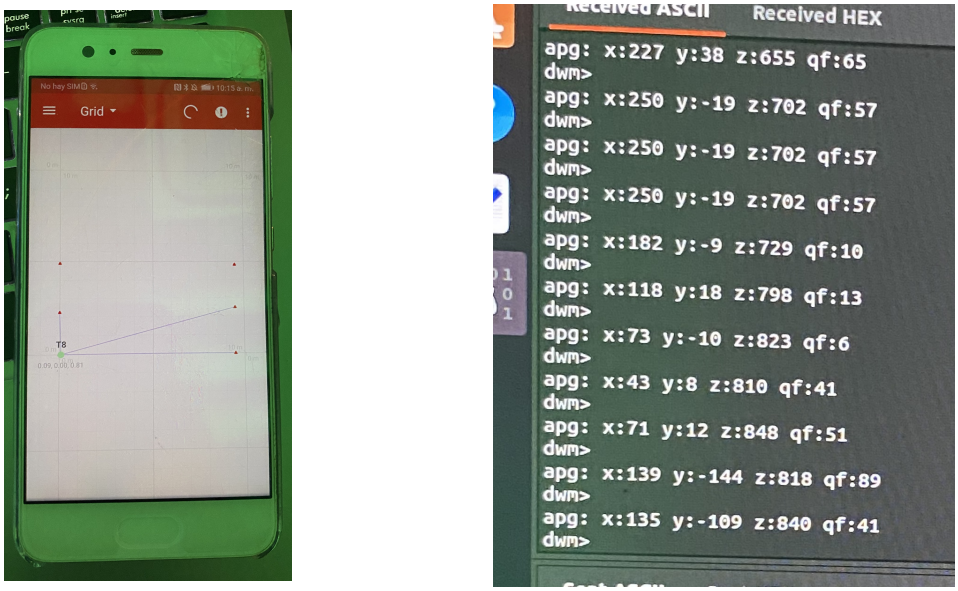


Figura 48: Muestra 2 de como se realizó la prueba en un lado del salón

en el *anchor* A1 y la posición que indica es $(0.09, 0.00, 0.81)$. Por otro lado en la Figura 49b hay un listado de la posiciones, en milímetros, que se fueron obteniendo al mantener presionado *enter*. La medida con mayor factor de calidad indica que la posición es $(135, -144, 818)$ en milímetros o $(0.135, -0.144, 0.818)$ en metros.



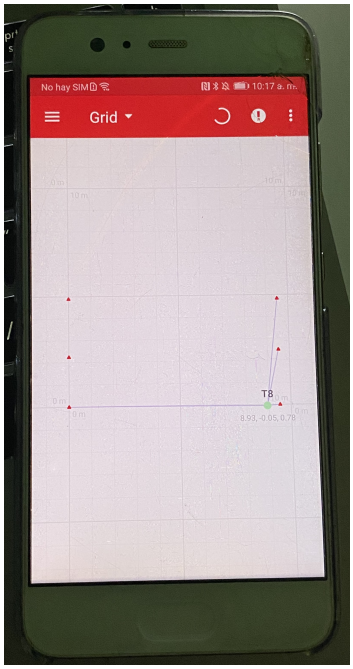
(a) Posición obtenida en la aplicación cerca del *anchor* A1

(b) Posición obtenida en el *Shell command* cerca del *anchor* A1

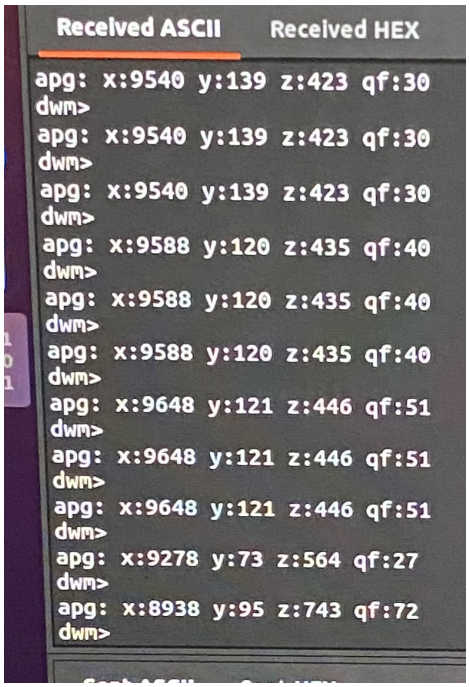
Figura 49: Posición obtenida por ambos métodos cerca del *anchor* 1 en la prueba 2

En la Figura 50a se puede observar que el círculo verde se encuentra casi en el *anchor* 2

y la posición que indica es $(8.93, -0.05, 0.78)$. Por otro lado en la Figura 50a hay un listado de la posiciones, en milímetros, que se fueron obteniendo al mantener presionado *enter*. La medida con mayor factor de calidad indica que la posición es $(8938, 95, 743)$ en milímetros o $(8.938, -0.095, 0.743)$ en metros.



(a) Posición obtenida en la aplicación cerca del *anchor* A2



(b) Posición obtenida en el *Shell command* cerca del *anchor* A2

Figura 50: Posición obtenida por ambos métodos cerca del *anchor* 2 en la prueba 2

8.3. Prueba 3: Verificación de actualización de datos en tiempo real

El objetivo de esta prueba era verificar si realmente los datos se estaban actualizando los suficientemente rápido. Esto se quería verificar por medio movilizarse alrededor de todo el área en donde estaba activa la red mientras se iba obteniendo la posición en tiempo real. Esos datos se querían graficar para ver si la trayectoria concordaba con lo que se caminó alrededor del área.

8.3.1. Montaje

Para la prueba se montó una red, con la forma y medidas que se muestran en la Figura 51. Los módulos destinados a ser *anchors* se colocaron sobre los CPUs de las computadoras disponibles en el laboratorio. Para conectarlos se utilizaron únicamente los cables micro USB directo de los módulos a las computadoras. En esta ocasión los módulos no se colocaron en una forma rectangular completamente. El *anchor* 11 está movido debido a que era ahí

únicamente donde había espacio disponible.

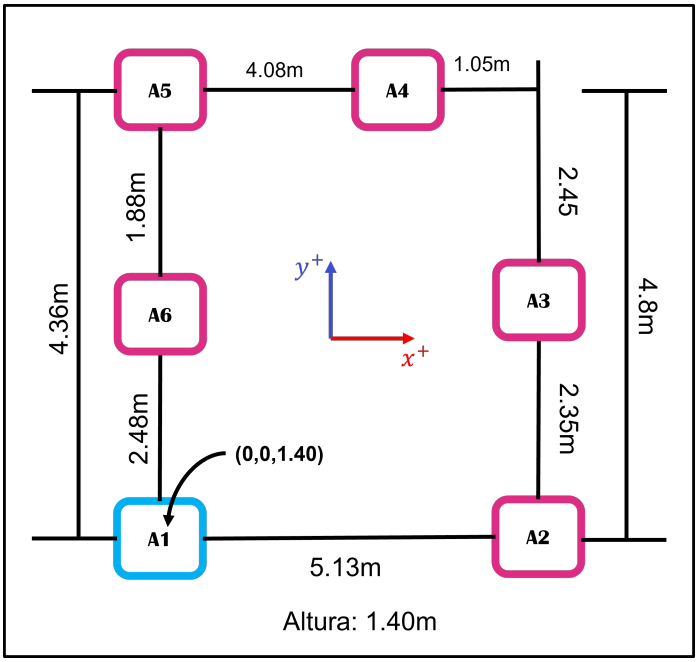


Figura 51: Forma y dimensiones de la red en la prueba 3

A continuación se muestran imágenes de donde fueron colocados los módulos dentro del salón de prueba. La Figura 52 muestra el salón completo en donde se colocó la red. Dado que es difícil diferencia en donde están colocados los módulos. La Figura 53b muestra en donde se colocaron los módulos del lado izquierdo dentro del salón y la Figura53a muestra los otros 3 módulos colocados del otro lado del salón.

8.3.2. Configuración

Una vez colocados los módulos que en la posición deseada, se prosiguió a configurarlos desde la app Android. Como se utilizaron los mismo módulos que la prueba pasada, la configuración de cada uno se mantuvo. De lo único que si se requirió cambio es en las posiciones en que se encontraba cada *anchor* debido a que las dimensiones del laboratorio cambiaron. El *tag* seguía siendo el mismo, con la etiqueta de T8 y la configuración de los parámetros de tasa de actualización normal y estática a valores de: $100ms/Hz$ y $100ms/Hz$ respectivamente.

Con esos pasos se consiguió crear una red llamada "Prueba3 - UVG"la cual consistía en 6 *anchors* (A1, A2, A3, A3 ,A5 y A6) y 1 *tag* (T8). Siendo el *anchor* A1 el iniciador.

8.3.3. Prueba en tiempo real

Para realizar la prueba en tiempo real se conectó el *tag* a la computadora y se puso en modo *Shell command*. Dentro de este se utilizó el comando *apg* para obtener la posición en milímetros. Dejando presionado *enter* se fue obteniendo la posición en tiempo real.

Figura 52: Salón en donde se realizó la prueba 3

(a) Posición en que se colocaron los módulos del lado izquierdo

(b) Posición en que se colocaron los módulos del lado derecho

Figura 53: Posición en donde se colocaron los módulos para la prueba 3

Se inició en el *anchor* iniciador, lo cual es el A1. Luego se prosiguió a caminar por todo el salón siguiendo una trayectoria por las orillas de la red y finalmente alrededor de la mesa en el centro que se puede observar en la Figura 52.

El módulo se iba sosteniendo con la mano, por lo que, no fue en una posición constante que se movilizó. Además, en el momento que se realizó la prueba el salón estaba lleno de gente por lo que se perjudicó en dos cosas. Primero en que no se pudiera mover totalmente estable el módulo. Segundo, que el *tag* no se movió exactamente al lado de los *anchors* , sino que se mantuvo, mínimo, separado de las mesas que se observan frente a los a las computadoras en la Figuras. 53b y 53a

Al terminar de caminar se tomaron los datos recopilados, en el *Shell Command*, de las coordenada x & y para graficarlas y visualizar la trayectoria que se formaba con los puntos.

Se decidió hacer un "diagrama de dispersión.[en] el programa de Excel para poder visualizar

la trayectoria. Se utilizó este diagrama ya que este utiliza las coordenadas cartesianas para mostrar los valores de dos variables para un conjunto de datos.

Graficación de puntos

En la Figura 54 se puede visualizar la trayectoria que se formó con los puntos obtenidos de los módulos. Los punto de los ejes coinciden con las dimensiones de la red, tanto de ancho como de largo. Los cuadrados rosados indican los *anchors*, simulando estar en la misma posición en que estaban en la vida real y el cuadro celeste indica el *anchor* iniciador.

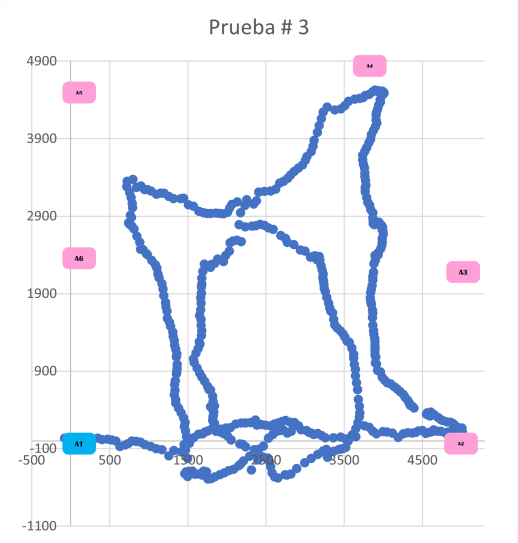


Figura 54: Gráfica obtenida de la prueba 3

Resultados

Luego de obtener la gráfica se pudo observar que la línea de la trayectoria es continua, por lo que nos indica que los módulos eran capaces de brindar una posición en todo momento. Los datos si se actualizaban lo suficientemente rápido como para que no hayan espacios en la trayectoria.

Otro cosa que fue posible notar, es que, a pesar de la geometría en que se colocaron los *anchors* el sistema si era capaz de brindar una posición congruente. Esto quiere decir, que los módulos pueden no colocarse en un rectángulo exacto para poder funcionar de una manera correcta,

Dado estos resultados se puede concluir que los módulos si eran aptos para implementarlo en un robot móvil como lo era el robot Rover UVG al que se le debía implementar el sistema.

8.4. Prueba 4: Prueba de campo y función de auto posicionamiento de los módulos

El objetivo de esta prueba fue probar el sistema de localización en un área mayor a la plataforma robótica. Se quería corroborar que las mediciones aún fueran congruentes. Asimismo se quería determinar si la funcionalidad de auto posición que traen los módulos configurados como *anchors*, si funcionaba y daba mediciones entre ellos correctos. Se realizaron 2 intentos colocando los módulos en distintas posiciones. A continuación se desglosan ambos intentos.

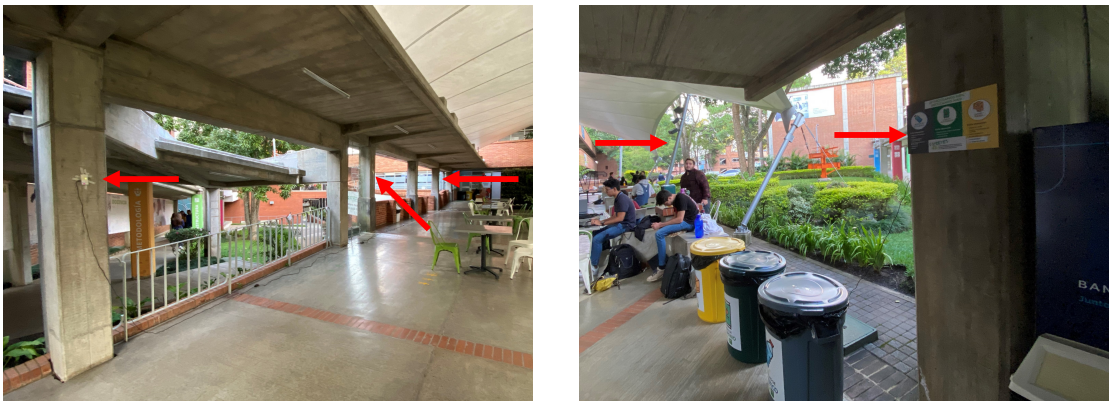
8.4.1. Intento 1: Prueba alrededor de toda la plaza

Montaje

Inicialmente se colocaron los módulos alrededor de toda la plaza Paiz Riera de la Universidad del Valle de Guatemala sobre las columnas y soportes. En la Figura 55 se muestra como es que fueron colocados los módulos. Adicionalmente en las siguientes figuras 56a y 56b se muestra en donde fueron colocados los módulos en la plaza (lado izquierdo y derecho respectivamente).

Figura 55: Forma de colocar los módulos sobre las columnas y soportes

A continuación en la 57 se muestra la distribución total de los módulos alrededor de toda las plaza. En donde el módulo del centro (flecha morada) fue el *anchor* iniciador.



(a) Posición en que se colocaron los módulos del lado izquierdo

(b) Posición en que se colocaron los módulos del lado derecho

Figura 56: Posición en donde se colocaron los módulos para la prueba 4

Figura 57: Distribución total del montaje de la prueba 4 - intento 1

Configuración

Una vez colocados los módulos en toda la plaza, se prosiguió a configurarlos desde la app Android. Como se utilizaron los mismo módulos que la prueba pasada, la configuración de cada uno se mantuvo. El *tag* seguía siendo el mismo, con la etiqueta de T8 y la configuración de los parámetros de tasa de actualización normal y estática a valores de: $100ms/Hz$ y $100ms/Hz$ respectivamente.

De lo único que si se requirió cambio es en las posiciones en que se encontraba cada *anchor*. En esta ocasión como estaban tan separados se utilizó la función de **auto posicionamiento** que ya traen integrado.

Auto-posicionamiento

Se utilizó la función de auto-posicionamiento por 2 razones. La primera fue porque los módulos estaban muy separados entre sí por lo que no se podría medir manualmente su posición. La segunda razón y la más importante fue para probar esa funcionalidad y ver que tan congruentes eran las posiciones de cada *anchor* provista.

Para realizar esto se siguió el manual de instrucciones que se puede escanear en la Figura 23. En este proceso se colocan los módulos en orden de como están colocados en la vida real empezando por el módulo que es el iniciador. Con esto la aplicación y los módulos solos comienzan a leer la posición que hay entre cada uno de ellos. Al finalizar brindan una posición estimada de cada *anchor* e incluso es posible tener una visualización gráfica en la aplicación de como estan posicionados.

Resultados

Luego de hacer el auto-posicionamiento fue posible ver que las posiciones obtenida no tenían congruencia con como estaban colocados los módulos en la plaza. La forma que generaba la red no era parecida a la que se tenía colocada realmente.

Dado esto fue posible identificar que para que funcione correctamente este recurso cada módulo debe tener al menos otros 3 módulos en línea de visión directa. Es decir, cada uno de los módulos necesita que hayan otros 3 sin interferencia para que las medidas si sean correctas, lo que no ocurría en este intento porqye habían columnas atravesadas.

Es mejor si todos los módulos se pueden observar entre ellos para que las medidas si sean congruentes.

8.4.2. Intento 2: Prueba con módulos colocados de forma controlada

El objetivo de esta prueba era realizar una red en un área mayor a la plataforma robótica para corroborar si las mediciones se realizaban correctamente. Adicionalmente, como ya se había intentado antes hacerlo en la plaza pero la función de auto-posicionamiento no funcionó correctamente. Es esta ocasión lo que si se quería lograr es que cada uno de los módulos si tuvieran línea directa en su visión con otros 3 módulos.

Montaje

Se colocaron cada uno de los módulos sobre una silla como se ve en la 58 para poder movilizarlos y alejarlos entre sí lo más posible. Para conectarlos se utilizaron cables micro USB y extensiones para poder alcanzar los tomacorrientes de la plaza. En esta ocasión los módulos no se colocaron en una forma rectangular sino que fue más en forma circular o hexagonal.

Figura 58: Forma de montaje de los módulos en la plaza

Configuración

La numeración y configuración de los *anchors* permaneció igual, lo único que cambiaba es que ahora ya no se tenía la posición de cada módulo.

Con esos pasos se consiguió crear una red llamada "Prueba4 - UVG"la cual consistía en 6 *anchors* (A1, A2, A3, A3 ,A5 y A6) y 1 *tag* (T8). Siendo el *anchor* A1 el iniciador.

Auto-posicionamiento

Al colocar los módulos en forma de círculo se prosiguió a realizar el auto-posicionamiento. Este se va ejecutando y va midiendo las distancias entre cada módulo como se muestra en la Figura 59.

Al finalizar se pudo previsualizar la forma que generaba la red (dentro de la aplicación), la cual fue en forma casi hexagonal como se puede ver en la Figura 60. Esto era congruente con la posición en lo que realmente estaban los DWM1001.

Prueba en tiempo real

Para corroborar que la red si era capaz de brindar una posición congruente y en tiempo real del *tag* se hizo la prueba de movilizarse dentro de la red y observar los datos obtenidos en la aplicación y en el *Shell Command*.

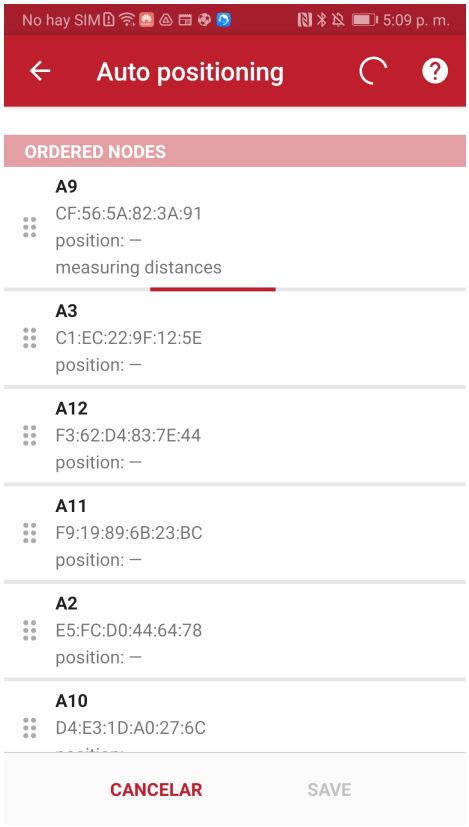


Figura 59: Ejecución de auto-posicionamiento en la aplicación Android

En la aplicación se verificó que la posición del *tag* si se actualizara con rapidez y que mostrara congruencia con la posición en la que se encontraba. Es decir, si el *tag* se encontraba cerca del *anchor* 1, que en la aplicación si se viera eso mismo.

En la Figura 61a se muestra la medición en la aplicación cuando el *tag* se encontraba cerca del *anchor* y las coordenadas que indica son:

$$x : -0.04m$$

$$y : 0.00m$$

$$z : 0.85m$$

Por otro lado, en la Figura 61b se muestra resaltado en rojo la medición obtenida desde el *Shell Command* los cual es (ya convertido a metros):

$$x : -0.07m$$

$$y : -0.08m$$

$$z : 0.877m$$

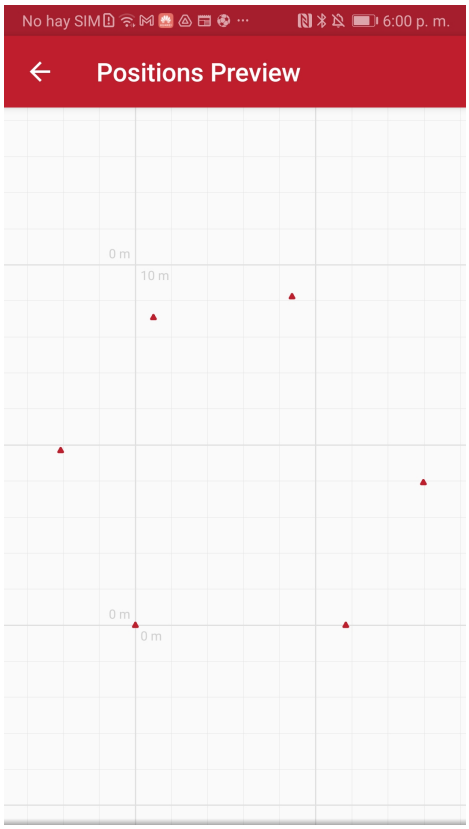
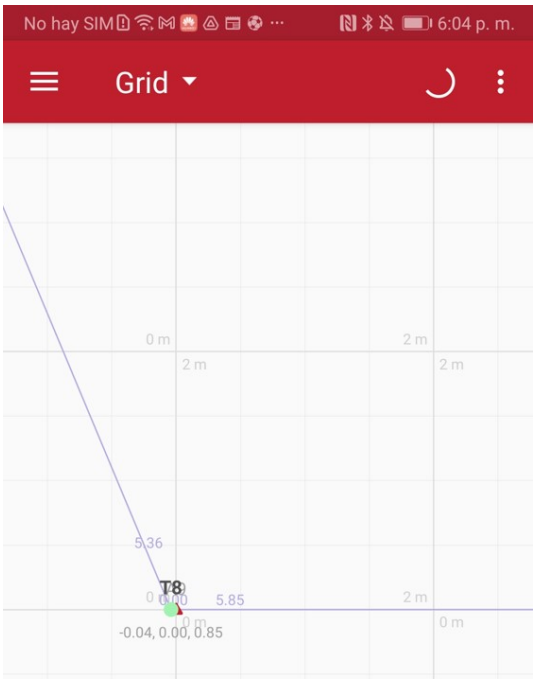
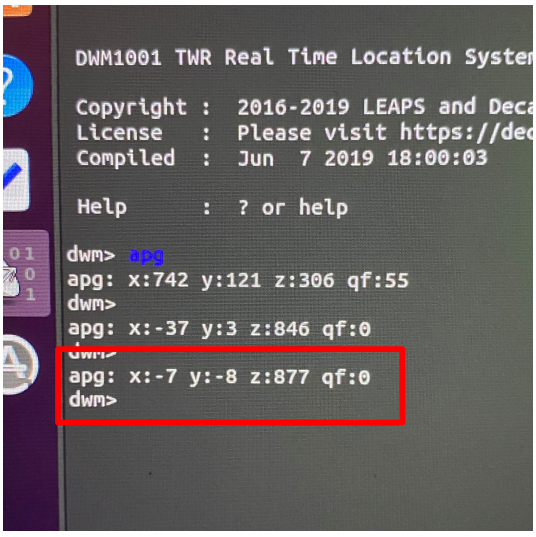


Figura 60: Previsualización de las posiciones de cada módulo



(a) Medición de un punto visto en la aplicación



(b) Medición de un punto visto en el *Shell Command*

Figura 61: Mediciones de un punto en la prueba de la plaza

CAPÍTULO 9

Proceso de alimentación y montaje de los módulos

La forma de alimentar los módulos y la forma en que se montaban dependía totalmente del lugar en donde se iban a hacer las pruebas y movilización del robot Rover UVG. Esta era una plataforma robótica ubicada en un laboratorio de la Universidad del Valle de Guatemala.

Esta es una plataforma, mostrada en la Figura 62 en alto tenía las dimensiones de 5 x 4 x 1 metros de largo, ancho y alto respectivamente. Estaba rodeada por cámaras del sistema de captura de movimiento Optitrack: y tiene fuentes y conexión al tomacorriente debajo de ella.

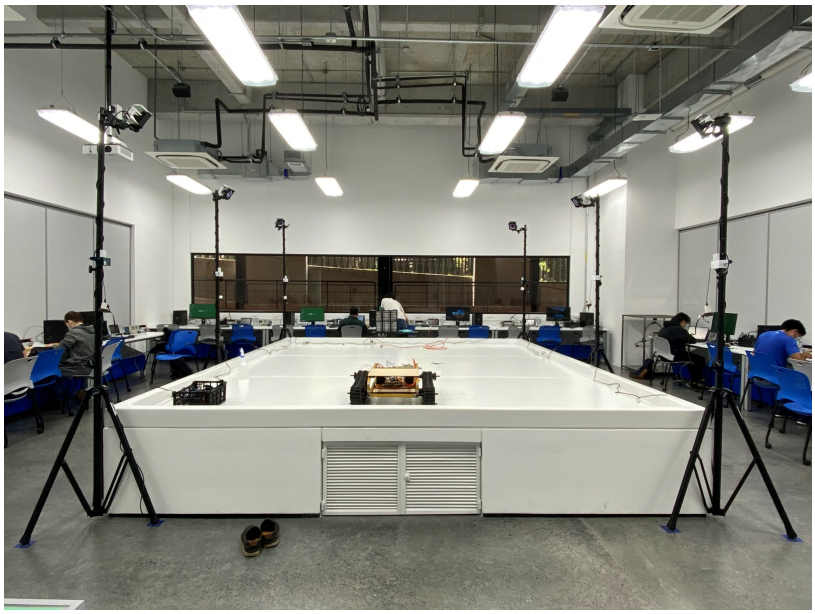


Figura 62: Plataforma robótica en donde se montaron los módulos

9.1. Proceso de alimentación y diseño de agarradores

9.1.1. Cable micro USB cortos y extensiones

Durante la familiarización con los módulos y posteriormente, en todas las pruebas realizadas (explicadas en el capítulo 8) se utilizó cable micro USB y dos formas de conectarlos.

1. Con extensiones y cubos de carga. Los cables se conectaban a los módulos y la entrada USB a un cubo de carga, estos ya se conectaban a las extensiones directas del tomacorriente.
2. A una computadora. Los cables se conectaban a los módulos y la entrada USB directo a una computadora de las computadores del laboratorio.

AL utilizar estas 2 formas los módulos siempre funcionaron de forma correcta.

9.1.2. Lugar en donde se iban a montar

Para el proyecto general del robot Rover UVG, en otro trabajo de graduación, se va a utilizar el sistema de Captura Optitrack. Por esta razón se decidió que mis módulos deberían ir en los mismo parales de las cámaras que el Optitrack. Estos serían los parales que se observan el la Figura 62.

Son 6 parales colocados en forma rectangular alrededor de la plataforma por lo que, aproximadamente, forman un espacio de 5 X 4 metros. Tomando el espacio en donde se debían montar se buscó la forma de alimentarlos y diseñar uno agarradores a estos parales.

9.1.3. Baterías portátiles

Dado que se quería que el sistema de localización funcionara de forma autónoma y además se quería que estuviera presentable visualmente. Dada esta última razón no se quería colocar las extensiones ya que podían estorbar el camino y no se iba a ver bien.

Por estos requerimientos se llego a la idea de usar baterías portátiles. Las baterías escogidas se muestran el la Figura 63. Estas entregaban 5V y 1A, lo que era más que suficiente para alimentar los módulos porque estos requerían 5V y 500m como mínimo. Otra razón por las que se escogieron fue por sus dimensiones. Estas eran pequeñas y poco pesadas para poder ponerlas montadas junto a los módulos en los parales del Optitrack. Sus dimensiones eran de: 3cm x 2.17cm x 9.02cm de largo, ancho y alto respectivamente.

9.1.4. Diseño e impresión 3D de agarradores

Tomando en consideración la batería portátil explicada anteriormente y los parales en donde se iban a colocar. Se diseñó un agarrador que se ajustara al diámetro de los parales

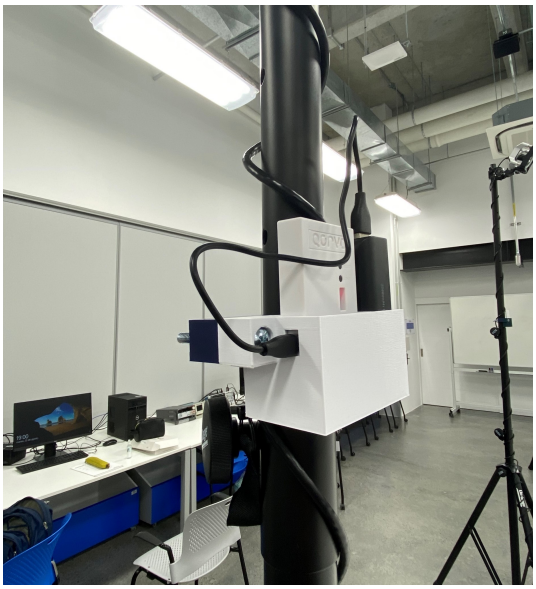
Figura 63: Baterías portátiles utilizadas para alimentar los módulos

y que fuera capaz de sostener el módulo y batería juntos. En la imágenes a continuación se muestra como quedó instalado y colocado.

Figura 64: Agarrador instalado con módulo y batería portátil visto de enfrente

9.1.5. Pruebas en plataforma

En el momento que se instalaron los módulos como se muestra anteriormente se realizó la prueba de funcionamiento de una vez. Para encender cada módulo se debía conectar el cable micro USB de este a la batería. Y como a plataforma es grande, cada módulo se fue encendiendo uno por uno conforme se iban conectando.



(a) Agarrador instalado con módulo y batería portátil visto del lado izquierdo

(b) Agarrador instalado con módulo y batería portátil visto del lado derecho

Figura 65: Agarradores co módulo y batería instadalos

Al encender todos luego de un rato los módulos se comenzaban a apagar en el mismo orden en que se fueron encendiendo. Se hizo pruebas de comenzar conectando otro módulos e ir los encendiendo en orden distinto y se obtuvo el mismo resultado. Luego de varias pruebas no se logró mantener la red activa por lo que se decidió quitarla y hacer pruebas solo con los módulos y baterías aparte.

9.1.6. Problemas encontrados

Se hicieron pruebas con solo los módulos y las baterías como se muestra en la Figura 66, en donde se fueron encendiendo en distinto orden.

Figura 66: Prueba con módulo y baterías para ver porqué se apagaban

Luego de varias pruebas se logró evidenciar que luego de 40 segundos los módulos empezaban a apagarse uno a uno. Dado esto se decidió usar un medidor de voltaje y corriente USB para corroborar lo que estaba ocurriendo al conectarlos.

Al medir voltaje de las baterías, este si estaba dando los 5V que decía. Sin embargo, al conectar el módulo y medir la corriente lo que ocurría es que, en el momento de conectarlo demandaba 0.09A (90mA). Pero esta medida era, literal, solo en el momento que se conectaba el módulo luego de eso la corriente que mostraba el medidor era 0.

Con esto pudimos darnos cuenta que los módulos demandaban tan poca corriente que las baterías los dejaban de detectar y por eso, luego de 40 segundos se apagaban.

9.1.7. Solución final

Debido a que las baterías no servían para mantener encendida la red se optó por directamente tomacorriente cada módulo.

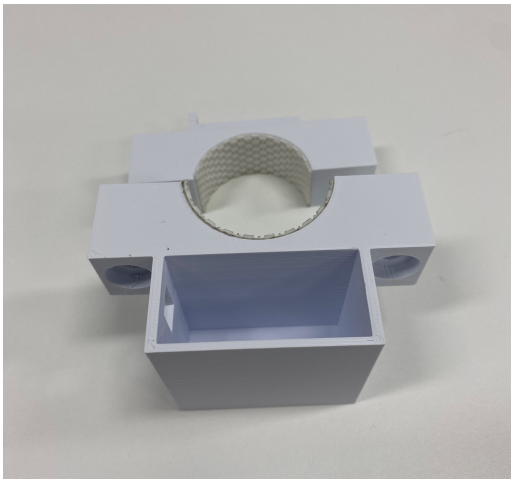
Se encontró una solución para encender la red y que se vieran bien visualmente. Esta consistió en conseguir 6 cables micro USB de 3.5 metros color negro para enrollarlos en los parales de la plataforma. Al llegar al piso, cada cable se conectó una extensión de 4 metros que se unificaron todas por debajo de la plataforma en una regleta. En la Figura 67 se muestra como quedó conectado el módulo al cable sobre el paral.

Figura 67: Módulo colocado con el cable alrededor del paral

Para poder encender la red, se conectó la regleta a un conector Wi-fi. Con este se pro-

gramó para opder tener un icono en el escritorio de la computadora que permite encenderlo desde ahí.

Adicionalmente se diseñó un nuevo agarrador quitando el espacio en donde iba la batería. En las figuras 68 se muestra el nuevo diseño impreso en 3D y colocado en el paral.



(a) Agarrador sin el espacio para la batería portátil impreso en 3D

(b) Agarrador sin espacio para batería colocado en el paral

Figura 68: Agarrador de solo el módulo impreso en 3D e intalado

CAPÍTULO 10

Integración del sistema en el entorno de ROS 2

10.1. Creación del nodo del sistema

El objetivo de esta fase del robot Rover UVG es hacer control punto a punto. Para esto es necesaria la odometría, conocer la posición de donde se está el robot respecto a un punto inicial. Dado esto se quiere implementar los sensores, módulos y el sistema de localización en el entorno de ROS 2. Con la intención de que la fusión y control del robot fuera sencilla.

Era importante que todos los miembros del proyecto unificaran las versiones del sistema operativo y de ROS. Entonces, para este trabajo se instaló la distribución Foxy de ROS 2 en una máquina virtual con Ubuntu 20.04.

Lo primero que se hace es crear un *Workspace:* en el entorno de ROS, esta se crea como una carpeta dentro de la computadora. Seguido se crea un *Package:* que contiene un Nodo:, este es un archivo en Python o en lenguaje c. Lo importante para este trabajo de graduación era el nodo ya que en este se realizó la programación para obtener la posición de los módulos.

Para poder unir el sistema de localización al proyecto del robot Rover UVG se requiere de comandos y funciones especiales dentro del archivo. La intención del nodo era publicar mensajes que contenían la posición del robot para que se pueda localizar este sobre un espacio determinado. Estos mensajes se publican en un *Topic:*, el cual es de odometría para este proyecto.

Dentro de las funciones especiales esta la de *Init* la cual configura al nodo para que pueda ser identificado por el nodo máster. Dentro de esta función se define:

1. Nombre del nodo
2. Nombre del *topic*

3. Frecuencia con la que se mandan los valores de la posición

4. Tipo de mensaje, el cual es odometría en este caso.

La otra función importante es la de *Publish* la cual se encarga de guardar la información obtenida por los sensores y llenar los campos necesarios del mensaje de odometría. Los campos son: las coordenadas x, y, z y la orientación, en este caso los valores del último campo que son el cuaternión se dejan en blanco porque los módulos no lo proporcionan.

10.2. Integración del nodo al proyecto del robot Rover UVG

El proyecto del robot Rover en el entorno de ROS 2 fue trabajado por otro compañero y la forma en que está organizado es la siguiente. Existe el *Workspace* del robot, en donde se crean los distintos *packages* según las partes del proyecto. En este caso, el paquete en donde se ingresa el nodo de los módulos se encuentra en el *package* de odometría.

El nombre del nodo es *Nodo_ dwm1001* el cual publica la información de la posición a un *topic* de odometría llamadao *odom_ nati*. Una vez ya fue capaz de enviar mensajes con la posición al *topic*, se integró al espacio de trabajo del robot Rover UVG. Esto quiere decir que ya, desde cualquier nodo que esté en el , se puede suscribir al *topic* de odometría y obtener la posición. En la Figura 69 se muestra un diagrama de como está integrado el nodo de odometría de los módulos DWM1001 y el *topic* creado.

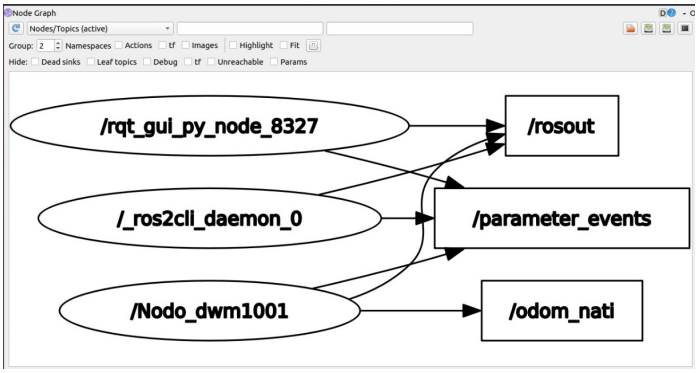


Figura 69: Diagrama en donde ya está integrado el nodo de odometría de los módulos DWM

10.3. Prueba de suscripción al *topic* de odometría

En otro trabajo de graduación se estaba trabajando el control del robot, es decir, ahí estaba el proyecto completo. Por esta razón se hizo una prueba en la computadora del compañero que desarrolló este nodo máster.

Lo que se realizó fue probar si el nodo del control del robot era capaz de suscribirse al *topic* de odometría y acceder a la posición brindada por los módulos. Se colocó el moódulo en aproximadamente una posición de $x = 0.46m$ & $y = 3.85m$). Se midió la posición y se

obtuvo la posición mostrada en la Figura 70. Las variables de orientación está en blanco porque los módulo no proveen esta información.

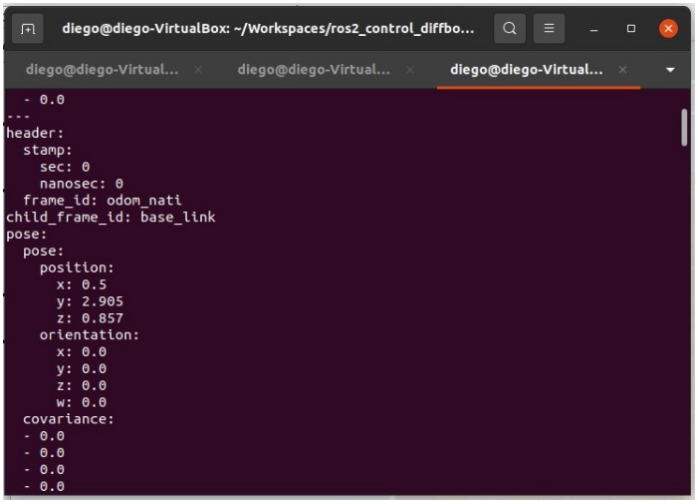


Figura 70: Pantalla mostrando los valores obtenidos de los módulos dentro del entorn de ROS 2

CAPÍTULO 11

Evaluación de precisión y exactitud de los módulos DWM1001

Para evaluar la precisión y exactitud de los módulo se midieron varios puntos en un área específica con un metro y luego se realizaron mediciones con los módulos en los mismo puntos. A continuación se explica como se montó, realizó y los resultados obtenidos.

11.1. Montaje de prueba

La prueba se realizó en la plataforma robótica mencionada en el Capítulo 9, esta se puede observar en la Figura 62. Se colocaron los módulos montados en los agarradores diseñados formando una red alrededor de la plataforma. Se puede observar el montaje en la Figura 71

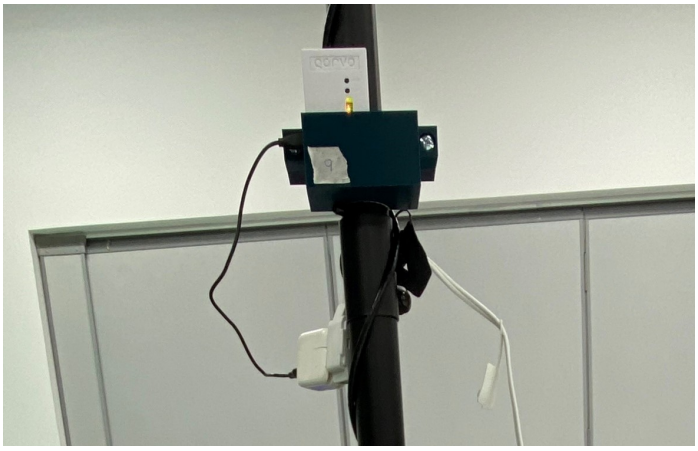


Figura 71: Montaje de los módulos en prueba de precisión y exactitud

11.2. Metodología

Para realizar la prueba se escogieron 13 puntos distintos en toda la plataforma de pruebas. Para cada uno se midió la coordenada X &Y de con una cinta métrica, tomando como referencia el origen en el módulo de la esquina inferior izquierda. En la Figura 72 se muestra un ejemplo de como se midió la coordenada Y de un punto específico.

Figura 72: Medición de coordenada Y de un punto dentro del área

Se hizo estas mediciones en 13 puntos distintos, y se colocó un papel en el punto exacto con las coordenadas escritas. La medida de la coordenada X coincidía con el centro del cuadrado papel y la medida de la coordenada Y coincidía con la orilla superior del papel. Con esto se aseguró que el módulo se colocaba en el lugar exacto donde se midió.

Al final se obtuvo una distribución de puntos como se muestra en la Figura 73, esta imagen fue alterada con los puntos morados para mejorar la visualización de los puntos medidos. En cada punto morado había un papel pegado a la plataforma con las coordenadas en metros y en milímetros escrito.

Una vez se tuvo los 13 puntos medidos se prosiguió a hacer medidas con los módulos DMW1001. Se colocó el *tag* encima del papel, en la posición exacta para que coincidiera con las mediciones manuales. Se utilizó el Shell Command desde la computadora para poder obtener las mediciones. Se obtuvo una cantidad aproximada de 200 mediciones por punto para tener resultados más certeros. Estas mediciones se trasladaron a Excel para poder graficarlos y obtener información de interés como: la media, desviación estándar, error de porcentaje y en cuanto difería de la medida real el promedio.

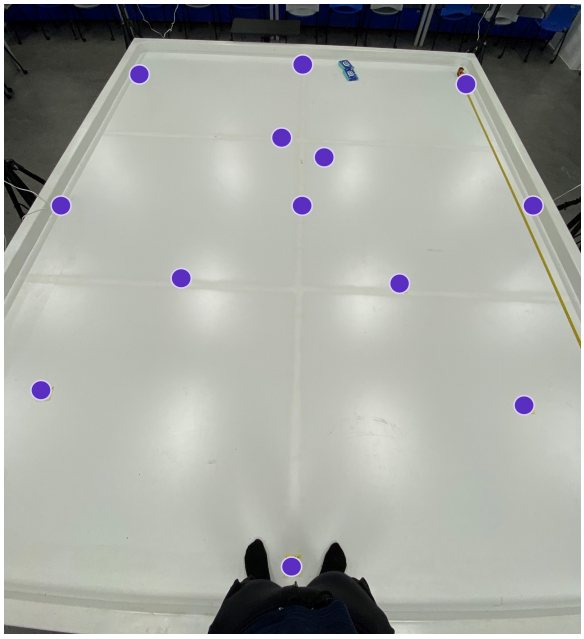


Figura 73: Distribución de puntos medidos en prueba de precisión y exactitud

Es importante mencionar que al utilizar una cinta métrica como instrumento de medición, se tiene una precisión en base a la cifra más pequeña que puede medir. En este caso la precisión de es un milímetro, por lo tanto el error instrumental es de 0.5 milímetros. Saber esto indica que las medidas manuales fueron tomadas como las reales, pero no eran 100 % la medida exacta.

11.2.1. Cálculo de precisión

Para verificar la precisión se tomaron entre 180 - 200 mediciones por punto y se calculó la desviación estándar y relativa. Esta se utilizó para cuantificar la dispersión del conjunto de datos numéricos medidos con los módulos DWM1001.

Para obtener esas desviaciones se utilizaron funciones de Excel descritas en el Cuadro 1. La desviación estándar, entre más cercana es a cero significa que menos dispersión tienen los datos entre sí.

11.2.2. Cálculo de exactitud

Para verificar la exactitud se calculó el error porcentual tomando como valor teórico la medida manual y el valor experimental el promedio de los valore medidos con los módulos DWM. La ecuación utilizada se describe en el Cuadro 1.

Adicionalmente se obtuvo la medida más alejada o desviada de la coordenada real para obtener el radio dentro del que los datos se obtuvieron. Esto fue con la intención de comparar con lo que el proveedor indicia de los módulos, lo cual es que tienen un exactitud $< 10cm$.

11.2.3. Funciones de Excel

En el Cuadro 1 se muestran las funciones y ecuaciones utilizadas en Excel para calcular lo mencionado anteriormente. Se indican tanto en español como en inglés en caso se tenga versiones distintas.

Funciones utilizadas en Excel		
	Español	Inglés
Promedio	=PROMEDIO	=AVERAGE
Valor absoluto	=ABS	=ABS
Desviación estándar de población	=DesvEstP	=STDEV.P
Desviación estándar relativa	$= \frac{DesvEstP()}{ABS(PROMEDIO())}$	$= \frac{STDEV.P()}{ABS(AVERAGE())}$
Error porcentual	$= \frac{Coordenadareal - PROMEDIO}{Coordenada}$ %	$= \frac{Coordenada - AVERAGE}{Coordenadareal}$ %

Cuadro 1: Funciones utilizadas en Excel para obtener precisión y exactitud

11.3. Gráfica de puntos medidos manualmente

A continuación se muestran la representación gráfica de los puntos medidos manualmente. En la Figura 74 se muestra el nombre de cada uno de los puntos medidos para que se puedan identificar. Estos son las mediciones que se consideraron como reales.

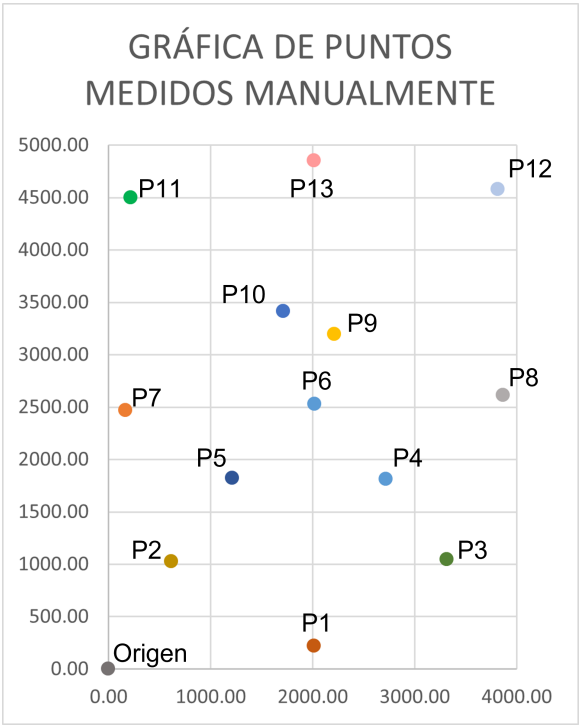


Figura 74: Gráfica de distribución de puntos medidos manualmente

11.4. Resultados

A continuación se muestra los resultados obtenidos en el origen y en otros 2 puntos medidos. Se muestran únicamente los resultados de estos 3 puntos para mostrar lo que se obtuvo para verificar precisión y exactitud. Sin embargo, se obtuvo la misma información de los otros 11 puntos medidos. Para poder ver el resto de gráficas y desglose de resultados revisar anexos.

11.4.1. Punto de origen

Se hizo 200 mediciones en este mismo punto. En el Cuadro 2 se muestra la comparación entre las medidas obtenidas manualmente y el promedio de las 200 mediciones realizadas con los módulos.

Mediciones en milímetros		
	X	Y
Coordenada real	10.0	10.0
Promedio de coordenada medida	−117.51	241.88

Cuadro 2: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el origen

Adicionalmente en el Cuadro 3 se muestra las mismas medidas, pero en centímetros y en metros.

Mediciones en centímetros y metros				
	X (cm)	Y (cm)	X (m)	Y (m)
Coordenada real	1.0	1.0	0.01	0.01
Coordenadas medida	−11.751	24.188	−0.118	0.242

Cuadro 3: Coordenada real y medida en centímetros y metros en el origen

Precisión en punto de origen

El origen estaba localizado en el módulo DWM de la esquina inferior izquierda. En ese punto se obtuvo una desviación estándar y relativa como se muestra en el Cuadro 4. Esto indica que el porcentaje no es mayor al 15 % lo cual indica que las mediciones fueron precisas.

Parámetros de precisión		
	X	Y
Desviación estándar	15.44	22.64
Desviación estándar relativa	13.14 %	9.36 %

Cuadro 4: Desviaciones estándar en el punto de origen

Exactitud en punto de origen

La posición medida en el origen tuvo un error porcentual bastante elevado, ya que obre pasa los mil en la coordenada X y los dos mil en la coordenada Y. Esto indica que las medidas no son exactas en comparación a lo que se esperaba: 10 cm en ambas coordenadas.

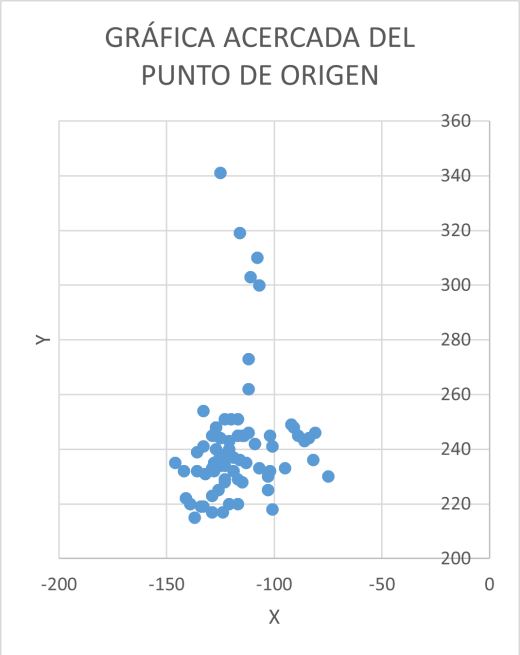
Además se puede observar que el valor más alejado en ambas coordenadas sobrepasan los 10 cm que indica el proveedor.

Parámetros de exactitud		
	X	Y
Error porcentual	1275.13 %	2318.75 %
Desviación máxima de la medición real (cm)	15.6	33.10

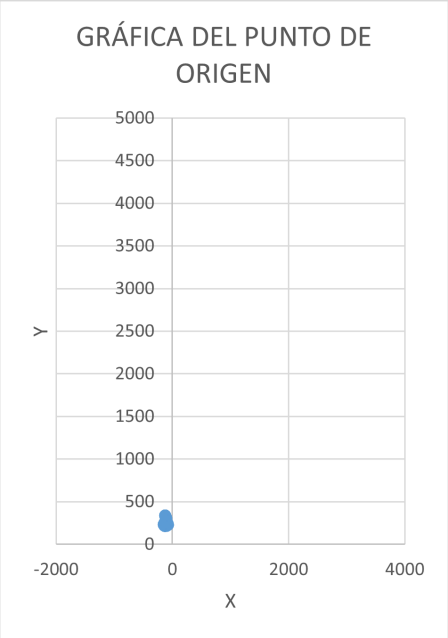
Cuadro 5: Error porcentual y rango de medición en el origen

Gráficas de mediciones

A continuación se muestran 2 gráficas de todos las mediciones realizadas para este punto. En la Figura 75a (izquierda) se muestran las 200 mediciones realizadas para poder apreciar la dispersión entre ellos. En la Figura 75b (derecha) se muestran las mismas mediciones pero los ejes tienen los valores iguales a las medidas de la plataforma robótica (4000 x 5000 milímetros). Esto es para poder apreciar que tanto coincide con el punto medido manualmente.



(a) Gráfica de los puntos del origen en una escala pequeña para ver la dispersión



(b) ráfica de los puntos del origen con una escala qe simula las dimensiones de la plataforma robótica

Figura 75: Gráficas de los puntos del origen

11.4.2. Punto 1

Se hizo 190 mediciones en este mismo punto. En el cuadro 6 se muestra la comparación entre las medidas obtenidas manualmente y el promedio de las 190 mediciones realizadas con los módulos.

Mediciones en milímetros del punto 1		
	X	Y
Coordenada real	2015.0	220.0
Promedio de coordenada medida	1934.92	413.17

Cuadro 6: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 1

Adicionalmente en el Cuadro 7 se muestra las mismas medidas pero en centímetros y en metros.

Mediciones en centímetros y metros				
	X (cm)	Y (cm)	X (m)	Y (m)
Coordenada real	201.5	22.0	2.015	0.22
Coordenadas medida	193.50	41.32	1.94	0.41

Cuadro 7: Coordenada real y medida en centímetros y metros en el punto 1

Precisión en punto 1

El punto 1 tiene mayor movimiento en el eje X y se puede ver en el Cuadro 8 que esta coordenada tiene menor desviación. Se puede observar que la coordenada Y tiene una desviación relativa de 25.57 % mientras la coordenada X casi es cero.

Parámetros de precisión		
	X	Y
Desviación estándar	51.47	105.61
Desviación estándar relativa	2.66 %	25.56 %

Cuadro 8: Desviaciones estándar en el punto 1

Exactitud en punto 1

La posición medida en el origen tuvo un error porcentual bastante elevado en la coordenada Y a diferencia de la coordenada X que es casi 4 %. Esto indica que la coordenada X si es exacta, mientras la Y no lo es.

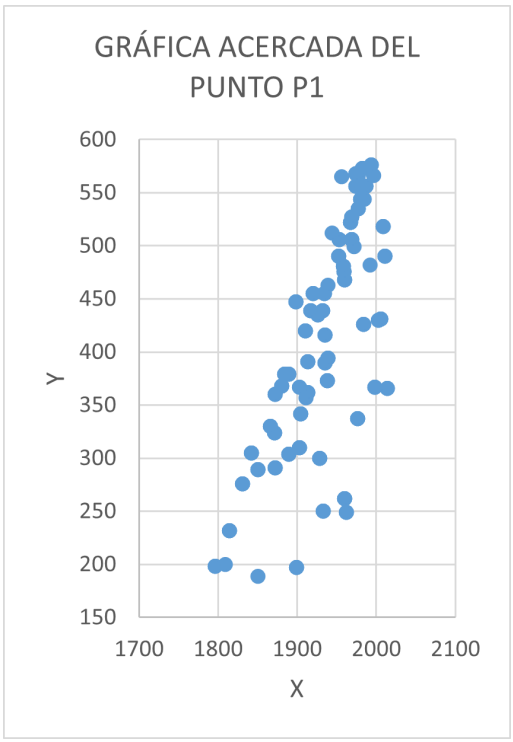
Sin embargo, se puede observar que el valor más alejado en ambas coordenadas sobrepasan los 10 cm que indica el proveedor.

Parámetros de exactitud		
	X	Y
Error porcentual	3.97 %	87.80 %
Desviación máxima de la medición real (cm)	21.9	35.6

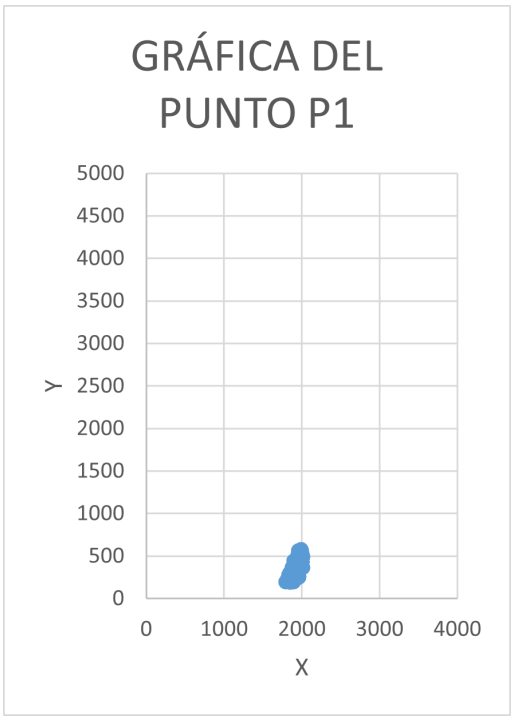
Cuadro 9: Error porcentual y desviación máxima de medición en el punto 1

Gráficas de mediciones

A continuación se muestran 2 gráficas de todos las mediciones realizadas para este punto. En la Figura 76a (izquierda) se muestran las 200 mediciones realizadas para poder apreciar la dispersión entre ellos. En la Figura 76b (derecha) se muestran las mismas mediciones pero los ejes tienen los valores iguales a las medidas de la plataforma robótica (4000 x 5000 milímetros). Esto es para poder apreciar que tanto coincide con el punto medido manualmente.



(a) Gráfica de los puntos del P1 en una escala pequeña para ver la dispersión



(b) Gráfica de los puntos del P1 con una escala qe simula las dimensiones de la plataforma robótica

Figura 76: Gráficas de los puntos del P1

11.4.3. Punto 6

Se hizo 190 mediciones en este mismo punto. En el Cuadro 10 se muestra la comparación entre las medidas obtenidas manualmente y el promedio de las 190 mediciones realizadas con los módulos.

Mediciones en milímetros del punto 6		
	X	Y
Coordenada real	2015.0	2530.0
Promedio de coordenada medida	2011.24	2568.03

Cuadro 10: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 6

Adicionalmente en el Cuadro 11 se muestra las mismas medidas pero en centímetros y en metros.

Mediciones en centímetros y metros				
	X (cm)	Y (cm)	X (m)	Y (m)
Coordenada real	201.5	253.0	2.015	2.530
Coordenadas medida	201.12	256.80	2.01	2.57

Cuadro 11: Coordenada real y medida en centímetros y metros en el punto 6

Precisión en punto 6

En este punto se obtuvo una desviación estándar y relativa como se muestra en el Cuadro 12. Estos valores indica que la precisión es de casi 99 % debido a que la desviación relativa es casi de 1.

Parámetros de precisión		
	X	Y
Desviación estándar	14.20	16.67
Desviación estándar relativa	0.71 %	0.65 %

Cuadro 12: Desviaciones estándar en el punto 6

Exactitud en punto 6

La posición medida en el centro tuvo un error porcentual bastante bajo en ambas coordenadas, ninguno sobrepasa el 2 % lo que nos indica que las mediciones fueron bastante exactas. Los valores obtenidos se pueden observar en el Cuadro 13 en la coordenada Y a diferencia de la coordenada X que es casi 4 %.

Además se puede observar que los valores más alejados no sobre pasan los 10 cm que indica el proveedor por lo que las mediciones en este punto fueron muy buenas.

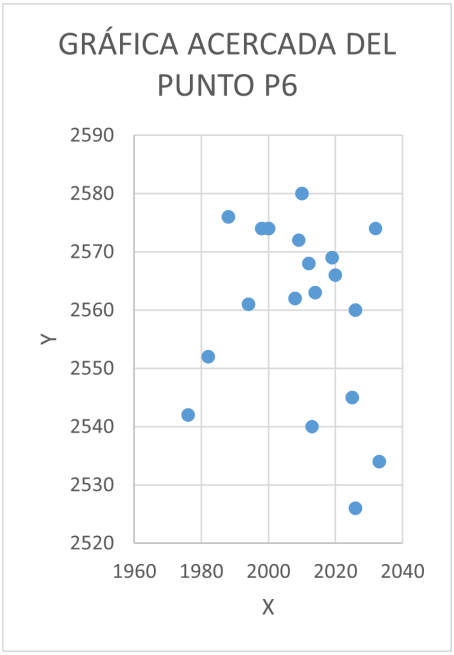
Parámetros de exactitud		
	X	Y
Error porcentual	0.19 %	1.50 %
Desviación máxima de la medición real (cm)	3.90	8.30

Cuadro 13: Error porcentual y desviación máxima de medición en el punto 6

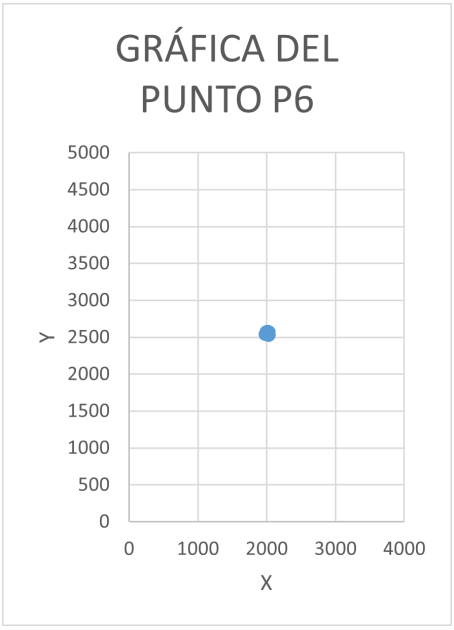
Gráficas de mediciones

A continuación se muestran 2 gráficas de todos las mediciones realizadas para este punto. En la Figura 77a (izquierda) se muestran las 200 mediciones realizadas para poder apreciar la dispersión entre ellos. En la Figura 77b (derecha) se muestran las mismas mediciones pero

los ejes tienen los valores iguales a las medidas de la plataforma robótica (4000 x 5000 milímetros). Esto es para poder apreciar que tanto coincide con el punto medido manualmente.



(a) Gráfica de los puntos del P6 en una escala pequeña para ver la dispersión

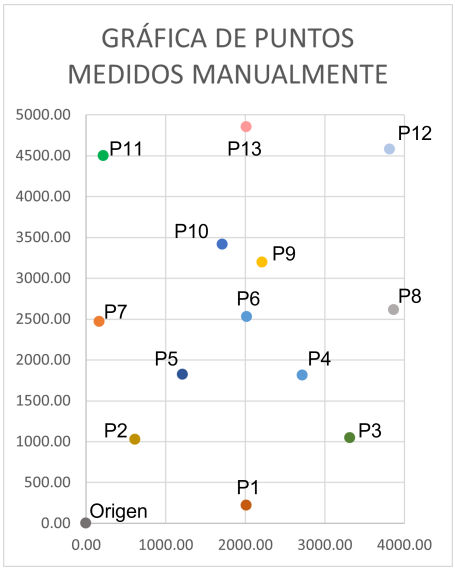


(b) Gráfica de los puntos del P6 con una escala que simula las dimensiones de la plataforma robótica

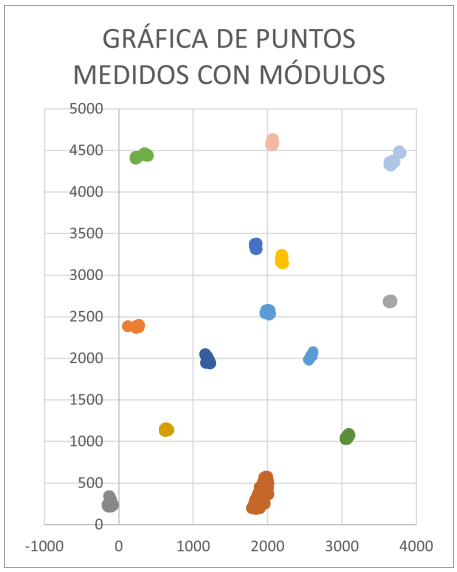
Figura 77: Gráficas de los puntos del P6

11.4.4. Comparación de mediciones con módulos y manuales

A continuación se muestran una gráficas que permiten comparar la posición de los puntos medidos manualmente y los datos obtenidos con los módulos. En la Figura 78.



(a) Visualización de los puntos medidos manualmente



(b) Visualización de los puntos medidos con los módulos DWM

Figura 78: Comparación entre las medidas manuales y con los módulos DWM

11.5. Resumen de resultados

Tomando en consideración las mediciones realizadas en los 13 puntos se obtuvo un promedio de estas. En las tablas a continuación se muestran los parámetros de precisión y exactitud promedio.

Parámetros de precisión promedio		
	X	Y
Desviación estándar	24.80	33.39
Desviación estándar relativa	0.03 %	0.04 %

Cuadro 14: Desviaciones estándar promedio de los 13 puntos

Parámetros de exactitud promedio		
	X	Y
Error porcentual	11.95 %	10.98 %
Desviación máxima de la medición real (cm)	14.45	18.15

Cuadro 15: Error porcentual y desviación máxima de medición promedio

CAPÍTULO 12

Comparación de los módulos DWM1001 vs Optitrack

El sistema de captura y movimiento 3D Optritrack también es utilizado para medir la posición de del robot Rover UVG por lo que se realizó una comparación entre los dos sistemas de localización. El objetivo era corroborar la precisión y exactitud de los dos sistemas y ver el más funcional para la implementación robótica.

12.1. Montaje de prueba

La prueba se realizó en la plataforma robótica mencionada en el Capítulo 9, esta se puede observar en la Figura 62. Se colocaron los módulos montados en los agarradores diseñados formando una red alrededor de la plataforma. Se puede observar el montaje en la Figura 79

12.2. Metodología

Para realizar la prueba se tenía que asegurar que las mediciones obtenidas por los módulos DWM y por el OptiTrack fuera realizadas en el mismo punto. Por esta razón se colocó el *marker* del OptiTrck encima del *tag* de los módulos DWM como se muestra en le Figura 80

Luego se tomaron mediciones en 4 puntos distintos. Con los módulos se tomaron 100 datos por puntos aproximadamente para comparar el promedio con la medida que daba el OptiTrack. Ya con esos valores se obtuvo el error porcentual para identificar que tanto variaba la medida de los módulos contra el otro sistema.

Algo importante que se tuvo que tener en consideración fue que el origen en ambos sistemas no era el mismo. En el caso de los módulos era la esquina inferior izquierda de la

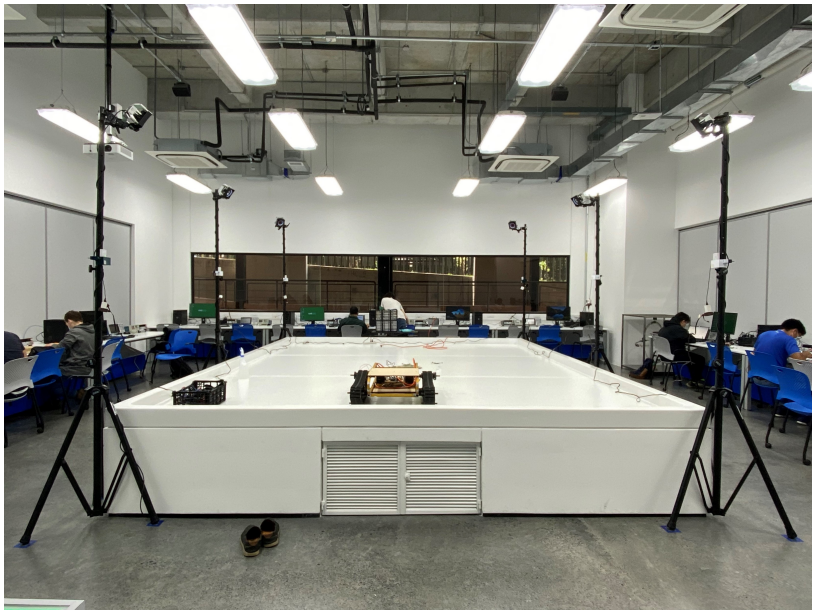


Figura 79: Montaje de los módulos en prueba de comparación con OptiTrack

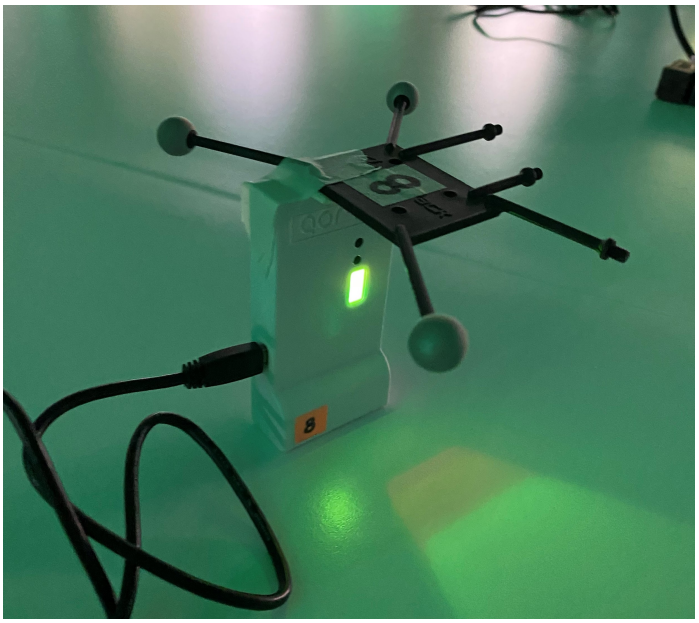


Figura 80: Colocación del *marker* y *tag* para realizar mediciones

plataforma y en el Optitrack era el centro de la plataforma. Dado eso se tuvo que hacer la conversión de un sistema a otro para obtener las coordenadas en los puntos.

Los pasos a seguir para convertir de un sistema a otro son los siguientes:

1. Obtener, con los módulos DWM, las medidas del centro de la plataforma, es decir el origen para el sistema Optitrack.
2. Obtener la posición del *marker* en cualquier punto dentro del área con el OptiTrack.

3. Obtener la posición del *tag* en el mismo punto que se midió el paso anterior.

4. A la medición del centro obtenida con los módulos restarle o sumarle las coordenadas X & Y del punto (según el cuadrante en que se encuentre el punto), obtenido con el OptiTrack.

5. Esos valores obtenidos ya se pueden comparar con la medición del punto obtenida con los módulos.

12.3. Resultados de mediciones

A continuación se va a mostrar los resultados obtenidos en el punto 1. Solo se muestra este punto pero el mismo procedimiento y resultados se obtuvo de los otros 3 puntos, si se quiere ver el desglose de estos puntos referirse a anexos.

12.3.1. Punto 1

Comparación de mediciones

Comparación de medidas		
	X	Y
Medición con DWM del centro (mm)	2092	3188
Medición con Optitrack	247.2	−193.5
Medición con módulos DWM	1886.89	3258.37
Medición real (Conversión de medida del OptiTrack)	1844.80	3381.50
Error porcentual	2.23 %	3.78 %

Cuadro 16: Medidas que sirvieron para hacer comparación del punto 1

Gráfica de comparación

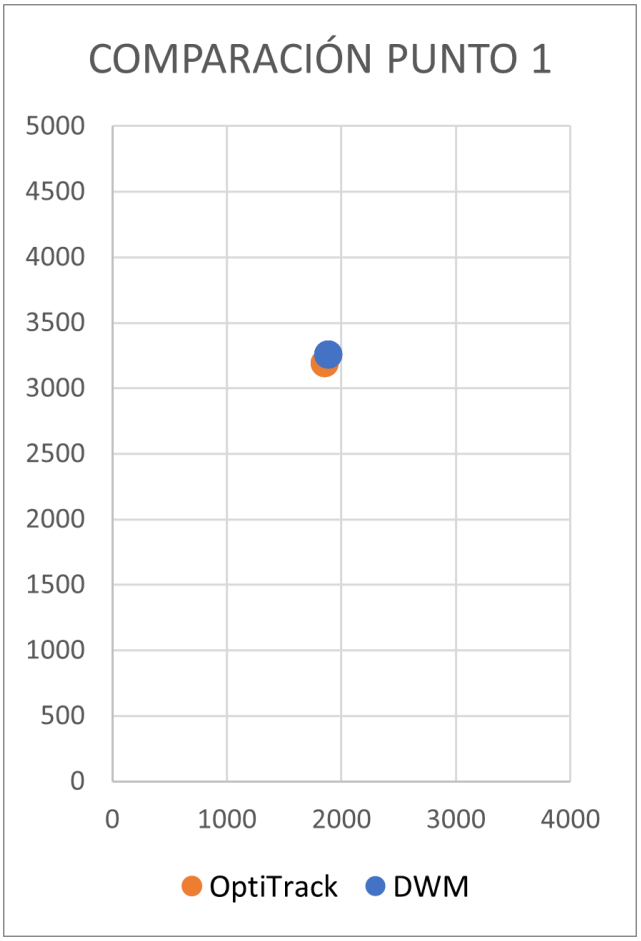


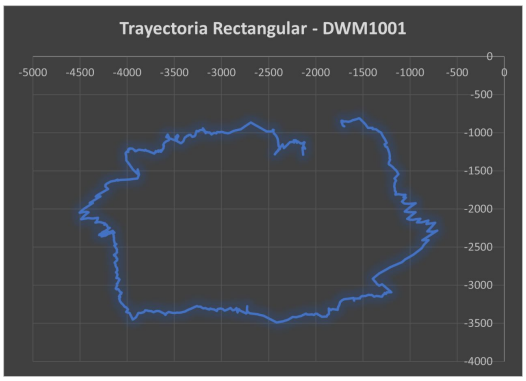
Figura 81: Comparación gráfica del punto medido con OptiTrack y los módulos DWM1001

12.4. Trayectorias en tiempo real

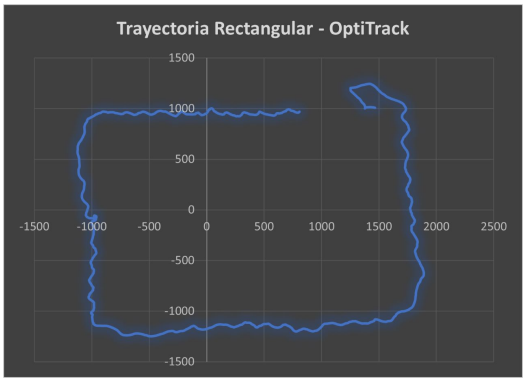
Para corroborar las mediciones en tiempo real se movió el *marker* del Optitrack y el *tag* de los módulos DWM al rededor de toda el área formado 2 trayectorias. Esto se hizo con el objetivo de ver la diferencias de medición en tiempo real de ambos sistemas.

12.4.1. Trayectoria alrededor del área

En las figuras 82 se muestra la comparación de la trayectoria obtenida al realizar una trayectoria en forma de cuadro alrededor del área.



(a) Trayectoria cuadrada obtenida con los módulos DWM

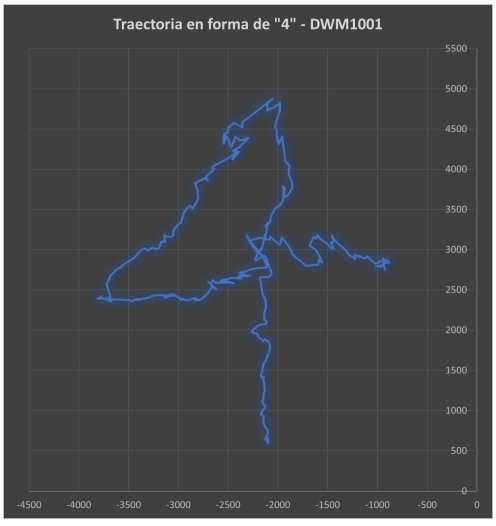


(b) Trayectoria cuadrada obtenida con OptiTrack

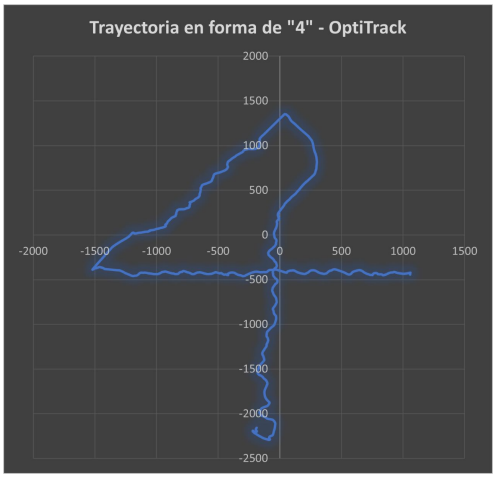
Figura 82: Comparación de trayectorias cuadradas medidas con los 2 sistemas

12.4.2. Trayectoria en forma de 4

En las figuras 83 se muestra la comparación de la trayectoria obtenida al realizar una trayectoria en forma del número 4 dentro del área.



(a) Trayectoria en forma de número 4 obtenida con los módulos DWM



(b) Trayectoria en forma de número 4 obtenida con OptiTrack

Figura 83: Comparación de trayectorias en forma de número 4 medidas con los 2 sistemas

CAPÍTULO 13

Integración al robot Rover UVG

Para concluir este trabajo de graduación el sistema de localización en tiempo real realizado con los módulos MDEK1001 se integró al robot Rover UVG por medio de ROS2. El nodo del sistema se ingreso en el proyecto completo del robot, realizado en otro trabajo de graduación, en el cual se encontraban todos los paquetes y nodos de las cámaras, sensores y sistemas de localización. Adicionalmente, en el otro trabajo de graduación se realizó la simulación del robot dentro del programa de visualización 3D llamado Rviz (este es especifico para utilizar con ROS).

Para unificar el todos los sensores, cámaras y sistemas en el robot y que este tuviera movilidad se utilizó una Raspberry pi 4. En esta se realizó el proyecto en ROS2 del robot y vía comunicación serial por conexión USB se conectó el *tag*. De esta manera el módulo iba colocado en el robot conectado directamente a la Raspberry pi en donde constantemente se le iba solicitando la posición.

13.1. Colocación del módulo al robot Rover UVG

Para poder colocar el *tag* sobre el Rover se diseñó un agarrador como se muestra en la Figura 84, el agarrador que tiene encima era para poder colocar el marker del OptiTrack y asegurarse que estaban colocados en el mismo punto. Para poder conectarse con las Raspberry pi 4, se le realizó un agujero por donde pasó el cable como se muestra en la Figura 85.

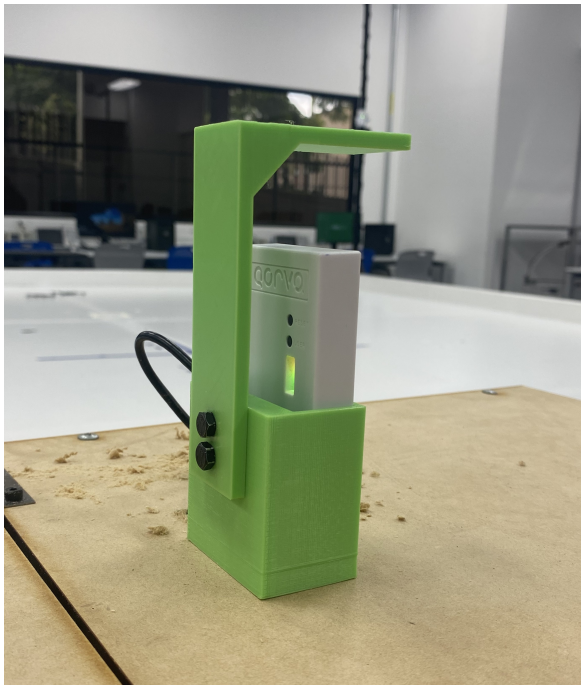


Figura 84: Colocación del *tag* sobre el robot Rover UVG

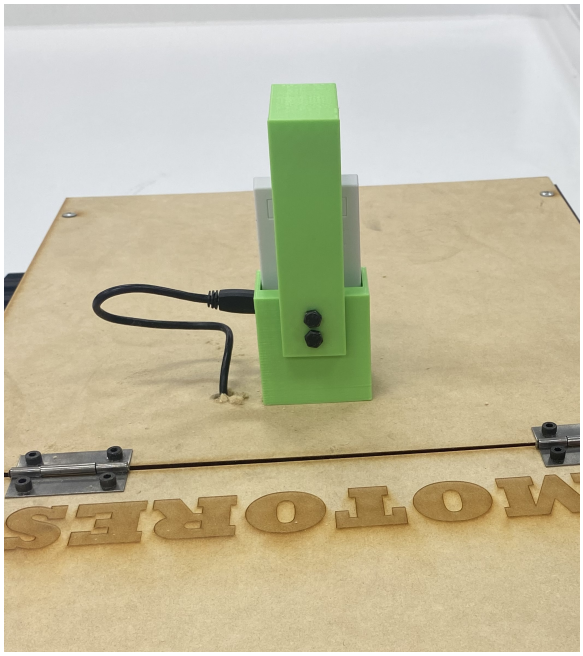


Figura 85: Conexión de módulo con Raspberry pi 4

13.2. Demostración de integración del sistema

Para corroborar que el sistema de localización funcionaba correctamente al unificarse al robot se hizo la siguiente prueba:

1. Se colocó el *tag* sobre el robot Rover UVG como se muestra en las figuras 84 y 85.
2. En el proyecto del Rover en ROS se le solicitó al *tag* la posición
3. Con la posición obtenida se inicializó la simulación del robot la cual se movía en base a lo que indicaba el *tag*.
4. Se grabó en tiempo real al robot en movimiento.
5. Se grabó pantalla donde se mostraba la simulación y la posición.
6. Se grabó la visualización del sistema en la aplicación Android.

A continuación se puede ver la demostración en donde están los 3 videos unidos.

Link para ver demostración.

Figura 86: Código QR para ver el demo

CAPÍTULO 14

Conclusiones

- Se implementó un sistema de localización en tiempo real utilizando módulos MDEK1001 para un robot explorador. El cual funciona con 6 *anchors* colocados en forma rectangular con el origen en el módulo de la esquina inferior izquierda.
- El sistema funciona únicamente si se tiene una tasa de actualización normal y estática igual a $100ms/Hz$. Esto permite una actualización de datos en una fracción de segundo.
- Si se quiere hacer una red más amplia es posible hacerlo usando la función de autoposicionamiento. Si y solo si cada módulo tiene al menos una línea de visión directa con otros 3 módulos de la red.
- Se implementó un nodo en lenguaje Python en el entorno de ROS 2 capaz de comunicarse vía UART con los módulos MDEK1001 utilizando formato TLV para enviar y recibir la información de las coordenadas.
- Los módulos tienen una exactitud de $\pm 18.15cm$
- La desviación estándar relativa para corroborar precisión es de 0.04 %.
- El sistema de captura OptiTrack provee mediciones en tiempo real menos dispersas y más uniformes que el sistema de localización conformado con módulos MDEK1001.

CAPÍTULO 15

Recomendaciones

Con base en los resultados obtenidos en esta tesis y para futuros trabajos se recomienda:

- Alimentar los módulos MDEK1001 con conexión directa a un tomacorriente o una computadora debido a la poca corriente que demandan.
- Realizar pruebas en entornos donde el robot se movilice en coordenadas negativas, es decir, por fuera de la red creada con los *anchors*.
- Realizar redes de módulos más grandes utilizando la opción de auto-posicionamiento que brindan los módulos. Ya se hizo la prueba de un círculo de 10 metros de diámetro por lo que se podría aumentar el tamaño poco a poco. De igual forma se puede probar agregar más *anchors* a la red y comprobar si eso permite cubrir más área.
- Dado que la aplicación que se le está dando al sistema es para un robot móvil. Sería ideal probar en entornos no controlados la precisión y exactitud de este. Se pueden hacer pruebas agregando obstáculos entre el *tag* y los *anchors* y luego probar con paredes de por medio.
- Imprimir en 3D los 5 agarradores de los módulos MDEK1001 que faltan. Estos son especializados para solo sostener los módulos, ya no tienen un espacio extra en donde se iba a colocar las baterías portátiles anteriormente.

CAPÍTULO 16

Bibliografía

[1] *Eliko UWB RTLS - The most accurate and reliable tracking*, en, mayo de 2021. dirección: `https://eliko.tech/uwb-rtls-ultra-wideband-real-time-location-system/` (visitado 21-04-2022).

[2] J. Uribe, «Evaluation of Ultra-Wideband Position Localization for an Indoor Office Environment,» *Evaluation*, 2018.

[3] H. J. S. Pinto, «Diseño Mecánico, Selección de Motores e Implementación de Sensores para un Robot Explorador Modular,» Universidad Del Valle de Guatemala, 2021.

[4] J. E. A. Murillo, «Diseño e implementación de capacidades automáticas de navegación para un Robot Explorador Modular,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[5] M. A. Izeppi, «Aplicación de Herramientas de Aprendizaje Reforzado y Aprendizaje Profundo en Simulaciones de Robótica de Enjambre con Restricciones Físicas,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[6] A. J. T. Gonzáles, «Diseño e implementación de un sistema de control y navegación automático para el robot secador de café,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[7] *What is RTLS - Real-Time Location System? | Litum*, en-US, Section: Blog, jul. de 2021. dirección: `https://litum.com/blog/what-is-rtls-real-time-location-system-rfid/` (visitado 04-05-2022).

[8] J. Diaz, «Tecnología Ultra-Wideband (UWB) La revolución a corto alcance,» *enter for Communications and Signal Processing Reseach New Jersey Institute ofTechnology*, 2003.

[9] V. Kumar, «Ultra Wide Band (UWB) Communication and its applications,» *In proceedings of 8th National Level Science Symposium., Rajkot, India.*, vol. 2, págs. 34-38, feb. de 2015.

[10] *DWM1001C Module*, en-US. dirección: `https://www.decawave.com/product/dwm1001-module/` (visitado 06-05-2022).

[11] M. Mosko, I. Solis y C. Wood, «Content-centric networking (CCNx) messages in TLV format,» inf. téc., 2019.

[12] *IBM Docs*, es-ES, jul. de 2022. dirección: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/es/db2-for-zos/12?topic=data-endianness (visitado 15-08-2022).

[13] Y. Stoyanov, *Little-Endian vs Big-Endian in Embedded Systems*, en-US, jun. de 2020. dirección: https://open4tech.com/little-endian-vs-big-endian-in-embedded-systems/ (visitado 15-08-2022).

[14] *ROS: Home*. dirección: https://www.ros.org/ (visitado 06-05-2022).

[15] M. J. Torres, C. León y P. Cardena, «ROS sistema operativo para robótica, nociones y aplicaciones,» *Revista Colombiana de Tecnologías de Avanzada*, pág. 7, 2014.

[16] *4 Nodos, Topics y Mensajes. Turtlesim*, es, ago. de 2020. dirección: http://rostutorial.com/4-nodos-topics-y-mensajes-turtlesim/ (visitado 07-05-2022).

[17] *What is a Virtual Machine? | VMware Glossary*, es-ES. dirección: https://www.vmware.com/topics/glossary/content/virtual-machine.html (visitado 25-09-2022).

[18] S. R. Campos, *Características y ventajas de sistemas de máquinas virtuales*, es-ES, mar. de 2022. dirección: https://www.docpath.com/caracteristicas-y-ventajas-de-sistemas-de-maquinas-virtuales/?lang=es (visitado 25-09-2022).

[19] *Motion Capture Systems*, en. dirección: http://optitrack.com/index.html (visitado 15-08-2022).

[20] *Desviación típica (o desviación estándar) | Análisis de resultados: conceptos estadísticos y resultados*. dirección: https://formacion.intef.es/pluginfile.php/49844/mod_imscp/content/4/desviacin_tpica_o_desviacin_estndar.html (visitado 25-09-2022).

[21] J. F. López, *Desviación típica - Definición, qué es y concepto*, es. dirección: https://economipedia.com/definiciones/desviacion-tipica.html (visitado 25-09-2022).

CAPÍTULO 17

Anexos

17.1. Comandos en el Shell Command

En esta sección se muestran los comandos a los que se tiene acceso desde el *Shell Command* para porder comunicarse con los módulos DWM1001. A continuación se muestra el listado literal:

DWM1001 TWR Real Time Location System

Copyright : 2016-2019 LEAPS and Decawave
License : Please visit https://decawave.com/dwm1001_license
Compiled : Jun 7 2019 18:00:03

Help : ? or help

Usage: <command> [arg0] [arg1] ...
Build-in commands:

** Command group: Base **
?: this help
help: this help
quit: quit

** Command group: GPIO **
gc: GPIO clear
gg: GPIO get
gs: GPIO set
gt: GPIO toggle

** Command group: SYS **
f: Show free memory on the he
reset: Reboot the system
si: System info
ut: Show device uptime
frst: Factory reset

** Command group: SENS **
twi: General purpose TWI read
aid: Read ACC device ID
av: Read ACC values
scs: Stationary config set

scg: Stationary config get

** Command group: LE **
les: Show meas. and pos.
lec: Show meas. and pos. in CSV
lep: Show pos. in CSV

** Command group: UWB **
utpg: Get TxPwr
utps: Set TxPwr

** Command group: UWBMAC **
nmg: Get node mode
nmp: Set UWB mode to passive
nmo: Set UWB mode to off
nma: Set mode to AN
nmi: Set mode to ANI
nmt: Set mode to TN
nmtl: Set mode to TN-LP
nmb: Set mode to BN
la: Show AN list
lb: Show BN list
nis: Set Network ID
nls: Set node label
udi: Show incoming IoT data
uui: Send IoT data
stg: Get stats
stc: Clear stats

** Command group: API **
tlv: Send TLV frame
aurs: Set upd rate
aurg: Get upd rate
apg: Get pos
aps: Set pos
acas: Set anchor config
acts: Set tag config
aks: Set encryption key
akc: Clear encryption key
ans: Set NVM usr data
anc: Clear NVM usr data
ang: Get NVM usr data

** Tips **
Press Enter to repeat the last command

17.2. Código de los módulos DWM1001

Link para ver los códigos de Python.

Figura 87: Código QR para ver los archivos de los códigos

17.3. CAD de agarradores

Link para ver archivos CAD de los agarradores.

Figura 88: Código QR para ver los archivos CAD de los agarradores

17.4. Cuadros y tablas de comprobación de precisión y exactitud en plataforma robótica

17.4.1. Precisión y exactitud del punto 2

Mediciones en milímetros del punto 2		
	X	Y
Coordenada real	615.0	1025.0
Promedio de coordenada medida	647.90	1149.36

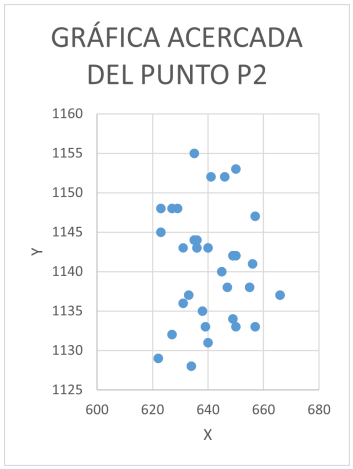
Cuadro 17: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 2

Parámetros de precisión		
	X	Y
Desviación estándar	16.86	23.73
Desviación estándar relativa	2.60 %	2.06 %

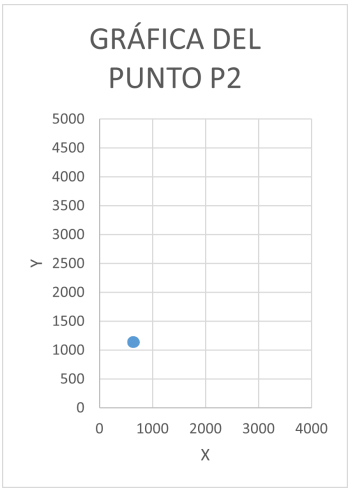
Cuadro 18: Desviaciones estándar en el punto 2

Parámetros de exactitud		
	X	Y
Error porcentual	5.35 %	12.13 %
Desviación máxima de la medición real (cm)	7.80	21.70

Cuadro 19: Error porcentual y desviación máxima de medición en el punto 2



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 89: Gráficas de los puntos del P2

17.4.2. Precisión y exactitud del punto 3

Mediciones en milímetros del punto 3		
	X	Y
Coordenada real	3315.0	1045.0
Promedio de coordenada medida	3083.04	1065.68

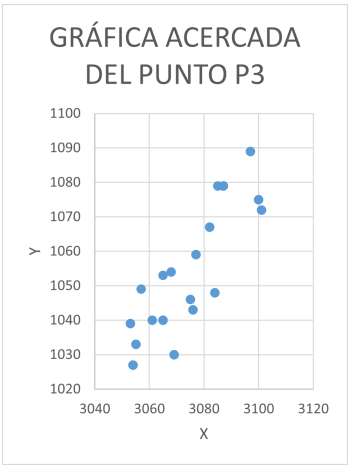
Cuadro 20: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 3

Parámetros de precisión		
	X	Y
Desviación estándar	14.31	22.33
Desviación estándar relativa	0.46 %	2.09 %

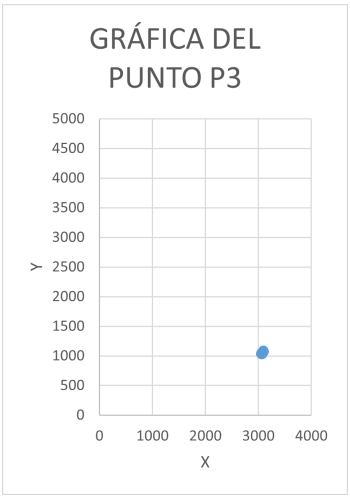
Cuadro 21: Desviaciones estándar en el punto 3

Parámetros de exactitud		
	X	Y
Error porcentual	7.00 %	1.98 %
Desviación máxima de la medición real (cm)	27.00	5.00

Cuadro 22: Error porcentual y desviación máxima de medición en el punto 3



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 90: Gráficas de los puntos del P3

17.4.3. Precisión y exactitud del punto 4

Mediciones en milímetros del punto 4		
	X	Y
Coordenada real	2715.0	1813.0
Promedio de coordenada medida	2597.86	1980.30

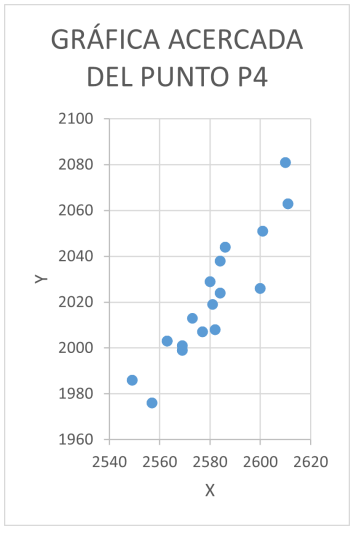
Cuadro 23: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 4

Parámetros de precisión		
	X	Y
Desviación estándar	25.05	43.47
Desviación estándar relativa	0.96 %	2.20 %

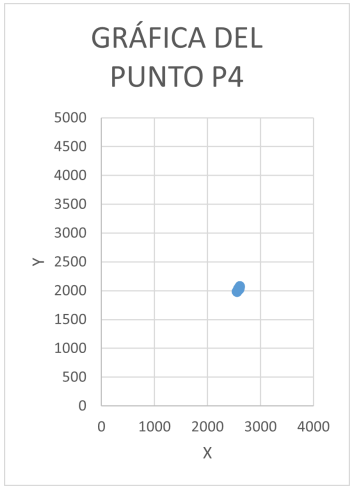
Cuadro 24: Desviaciones estándar en el punto 4

Parámetros de exactitud		
	X	Y
Error porcentual	4.31 %	9.23 %
Desviación máxima de la medición real (cm)	16.60	26.80

Cuadro 25: Error porcentual y desviación máxima de medición en el punto 4



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 91: Gráficas de los puntos del P4

17.4.4. Precisión y exactitud del punto 5

Mediciones en milímetros del punto 5		
	X	Y
Coordenada real	1215.0	1820.0
Promedio de coordenada medida	1207.81	1982.43

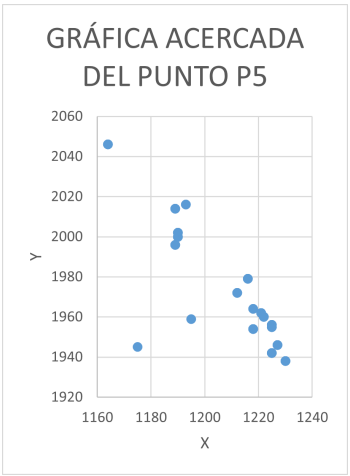
Cuadro 26: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 5

Parámetros de precisión		
	X	Y
Desviación estándar	17.38	34.71
Desviación estándar relativa	1.44 %	1.75 %

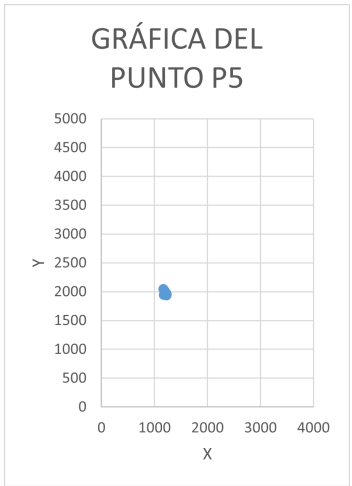
Cuadro 27: Desviaciones estándar en el punto 5

Parámetros de exactitud		
	X	Y
Error porcentual	0.59 %	8.92 %
Desviación máxima de la medición real (cm)	7.00	23.20

Cuadro 28: Error porcentual y desviación máxima de medición en el punto 5



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 92: Gráficas de los puntos del P5

17.4.5. Precisión y exactitud del punto 7

Mediciones en milímetros del punto 7		
	X	Y
Coordenada real	165.00	2470.00
Promedio de coordenada medida	253.76	2383.02

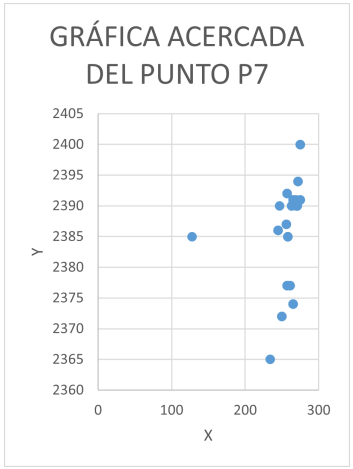
Cuadro 29: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 7

Parámetros de precisión		
	X	Y
Desviación estándar	34.57	15.74
Desviación estándar relativa	13.62 %	0.661 %

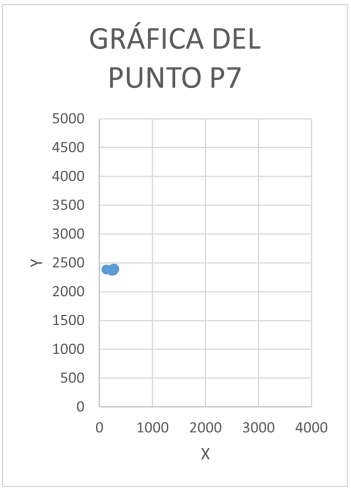
Cuadro 30: Desviaciones estándar en el punto 7

Parámetros de exactitud		
	X	Y
Error porcentual	53.79 %	3.52 %
Desviación máxima de la medición real (cm)	13.50	11.80

Cuadro 31: Error porcentual y desviación máxima de medición en el punto 7



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 93: Gráficas de los puntos del P7

17.4.6. Precisión y exactitud del punto 8

Mediciones en milímetros del punto 8		
	X	Y
Coordenada real	3865.0	2610.0
Promedio de coordenada medida	3649.46	2682.72

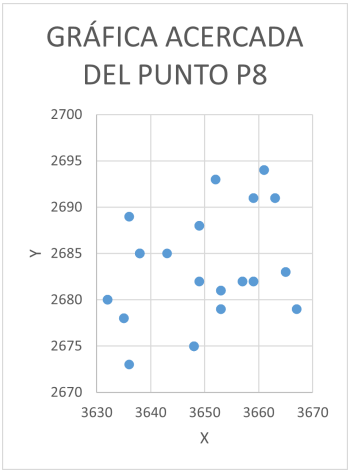
Cuadro 32: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 8

Parámetros de precisión		
	X	Y
Desviación estándar	13.81	9.83
Desviación estándar relativa	0.38 %	0.37 %

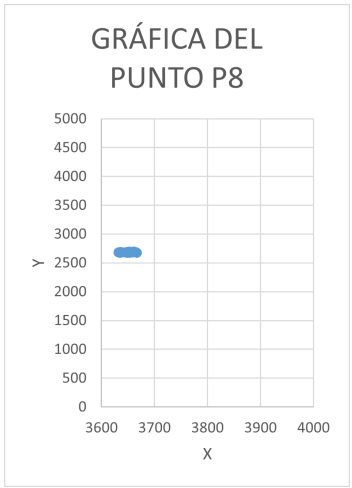
Cuadro 33: Desviaciones estándar en el punto 8

Parámetros de exactitud		
	X	Y
Error porcentual	5.58 %	2.79 %
Desviación máxima de la medición real (cm)	25.10	9.60

Cuadro 34: Error porcentual y desviación máxima de medición en el punto 8



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 94: Gráficas de los puntos del P8

17.4.7. Precisión y exactitud del punto 9

Mediciones en milímetros del punto 9		
	X	Y
Coordenada real	2215.00	3196.00
Promedio de coordenada medida	2198.16	3149.80

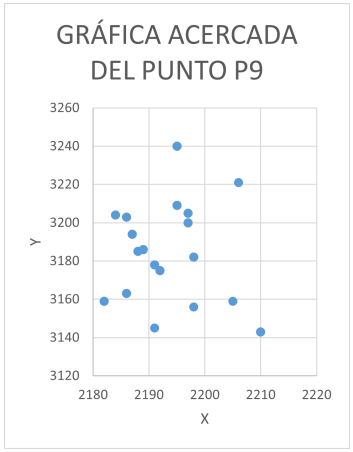
Cuadro 35: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 9

Parámetros de precisión		
	X	Y
Desviación estándar	14.20	45.10
Desviación estándar relativa	0.65 %	1.43 %

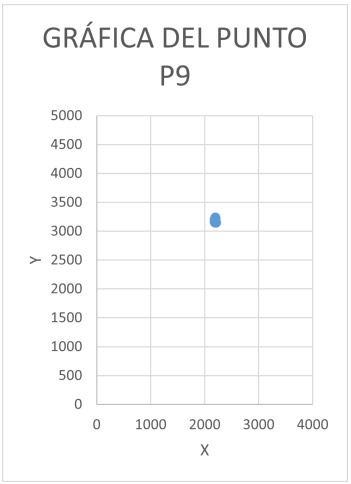
Cuadro 36: Desviaciones estándar en el punto 9

Parámetros de exactitud		
	X	Y
Error porcentual	0.76 %	1.45 %
Desviación máxima de la medición real (cm)	5.20	15.00

Cuadro 37: Error porcentual y desviación máxima de medición en el punto 9



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 95: Gráficas de los puntos del P9

17.4.8. Precisión y exactitud del punto 10

Mediciones en milímetros del punto 10		
	X	Y
Coordenada real	1715.00	3412.00
Promedio de coordenada medida	1848.81	3333.42

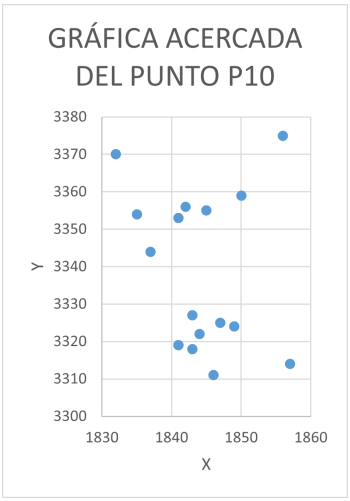
Cuadro 38: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 10

Parámetros de precisión		
	X	Y
Desviación estándar	10.31	19.01
Desviación estándar relativa	0.56 %	0.57 %

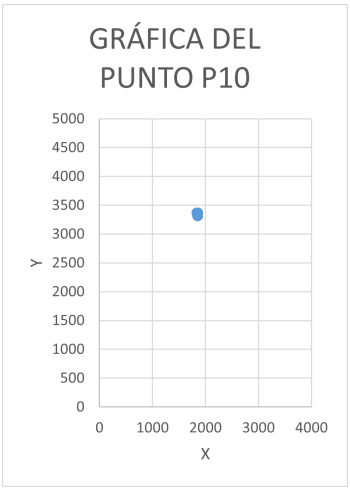
Cuadro 39: Desviaciones estándar en el punto 10

Parámetros de exactitud		
	X	Y
Error porcentual	7.80 %	2.30 %
Desviación máxima de la medición real (cm)	16.60	12.30

Cuadro 40: Error porcentual y desviación máxima de medición en el punto 10



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 96: Gráficas de los puntos del P10

17.4.9. Precisión y exactitud del punto 11

Mediciones en milímetros del punto 3		
	X	Y
Coordenada real	215.00	4500.00
Promedio de coordenada medida	348.23	4450.63

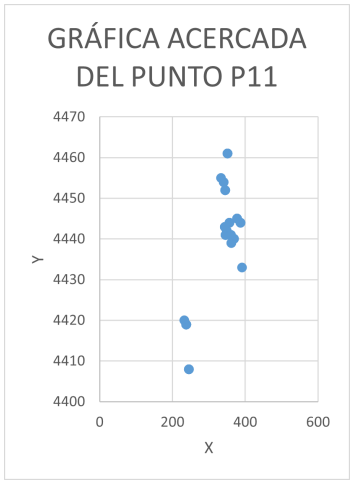
Cuadro 41: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 11

Parámetros de precisión		
	X	Y
Desviación estándar	30.66	13.79
Desviación estándar relativa	8.81 %	0.31 %

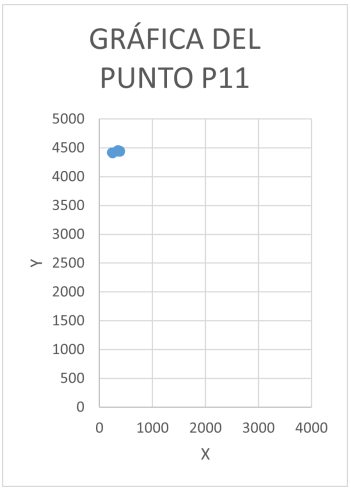
Cuadro 42: Desviaciones estándar en el punto 11

Parámetros de exactitud		
	X	Y
Error porcentual	61.97 %	1.10 %
Desviación máxima de la medición real (cm)	19.90	9.20

Cuadro 43: Error porcentual y desviación máxima de medición en el punto 11



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 97: Gráficas de los puntos del P11

17.4.10. Precisión y exactitud del punto 12

Mediciones en milímetros del punto 3		
	X	Y
Coordenada real	3815.00	4577.00
Promedio de coordenada medida	3758.38	4376.71

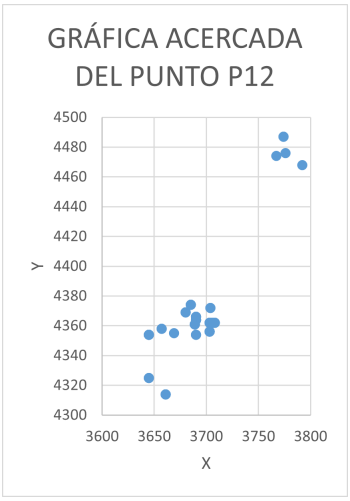
Cuadro 44: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 12

Parámetros de precisión		
	X	Y
Desviación estándar	62.30	30.50
Desviación estándar relativa	1.66 %	0.70 %

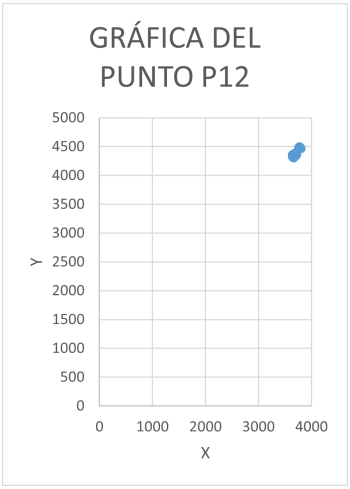
Cuadro 45: Desviaciones estándar en el punto 12

Parámetros de exactitud		
	X	Y
Error porcentual	1.48 %	4.38 %
Desviación máxima de la medición real (cm)	17.00	56.30

Cuadro 46: Error porcentual y desviación máxima de medición en el punto 12



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 98: Gráficas de los puntos del P12

17.4.11. Precisión y exactitud del punto 13

Mediciones en milímetros del punto 3		
	X	Y
Coordenada real	2015.00	4850.00
Promedio de coordenada medida	2066.69	4579.33

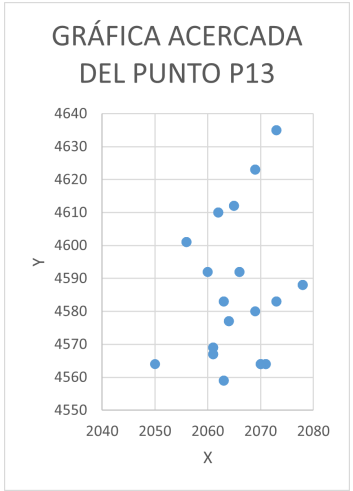
Cuadro 47: Tabla de comparación entre la coordenada real y la medida con los módulos DWM1001 en el punto 13

Parámetros de precisión		
	X	Y
Desviación estándar	26.68	64.28
Desviación estándar relativa	1.29 %	1.404 %

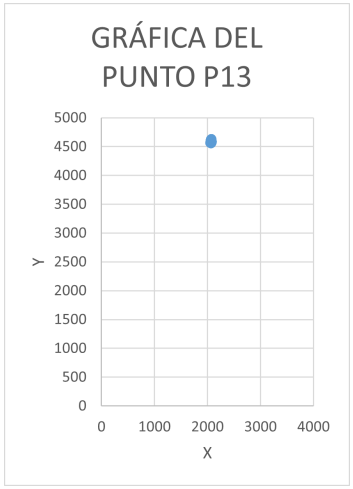
Cuadro 48: Desviaciones estándar en el punto 13

Parámetros de exactitud		
	X	Y
Error porcentual	2.57 %	5.58 %
Desviación máxima de la medición real (cm)	6.30	31.10

Cuadro 49: Error porcentual y desviación máxima de medición en el punto 13



(a) Gráfica en una escala pequeña para ver la dispersión



(b) Gráfica con una escala que simula las dimensiones de la plataforma robótica

Figura 99: Gráficas de los puntos del P13

17.5. Tablas y gráficas de comparación entre el sistema OptiTrack y los módulos DWM1001

17.5.1. Comparación de mediciones punto 2

Comparación de medidas		
	X	Y
Medición con DWM del centro (mm)	2092	3188
Medición con Optitrack	−903.8	1294
Medición con módulos DWM	3135.54	1634.98
Medición real (Conversión de medida del OptiTrack)	2995.80	1806.00
Error porcentual	4.46 %	10.46 %

Cuadro 50: Medidas que sirvieron para hacer comparación del punto 2

17.5.2. Comparación de mediciones punto 3

Comparación de medidas		
	X	Y
Medición con DWM del centro (mm)	2092	3188
Medición con Optitrack	1643.50	2763.3
Medición con módulos DWM	525.74	370.17
Medición real (Conversión de medida del OptiTrack)	448.50	424.70
Error porcentual	14.69 %	14.73 %

Cuadro 51: Medidas que sirvieron para hacer comparación del punto 3

17.5.3. Comparación de mediciones punto 4

Comparación de medidas		
	X	Y
Medición con DWM del centro (mm)	2092	3188
Medición con Optitrack	−865.10	−710.30
Medición con módulos DWM	2925.09	3793.03
Medición real (Conversión de medida del OptiTrack)	2957.10	3898.30
Error porcentual	1.09 %	2.78 %

Cuadro 52: Medidas que sirvieron para hacer comparación del punto 4

17.5.4. Gráfica de comparación punto 2

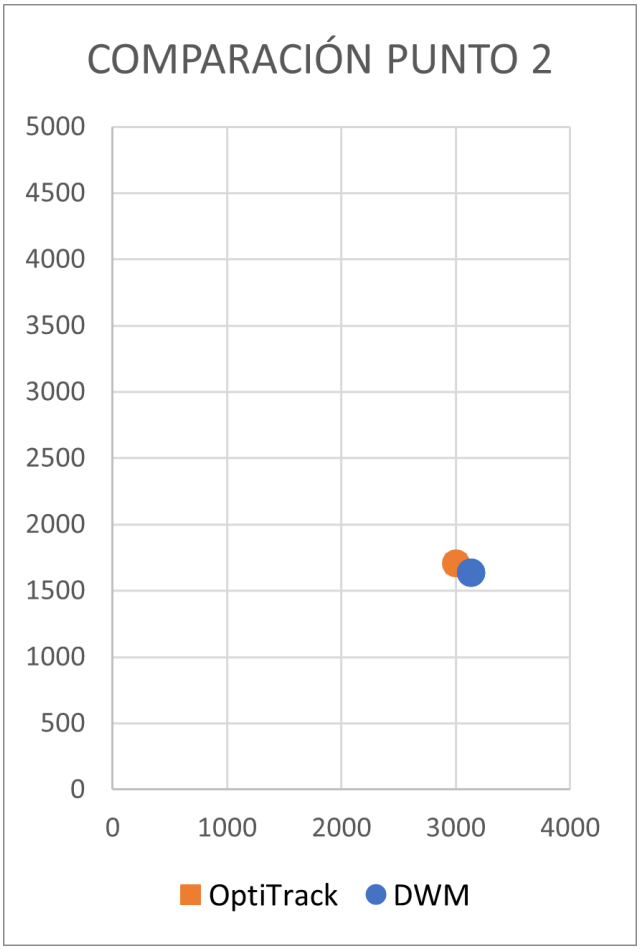


Figura 100: Comparación gráfica del punto 2 medido con OptiTrack y los módulos DWM1001

17.5.5. Gráfica de comparación punto 3

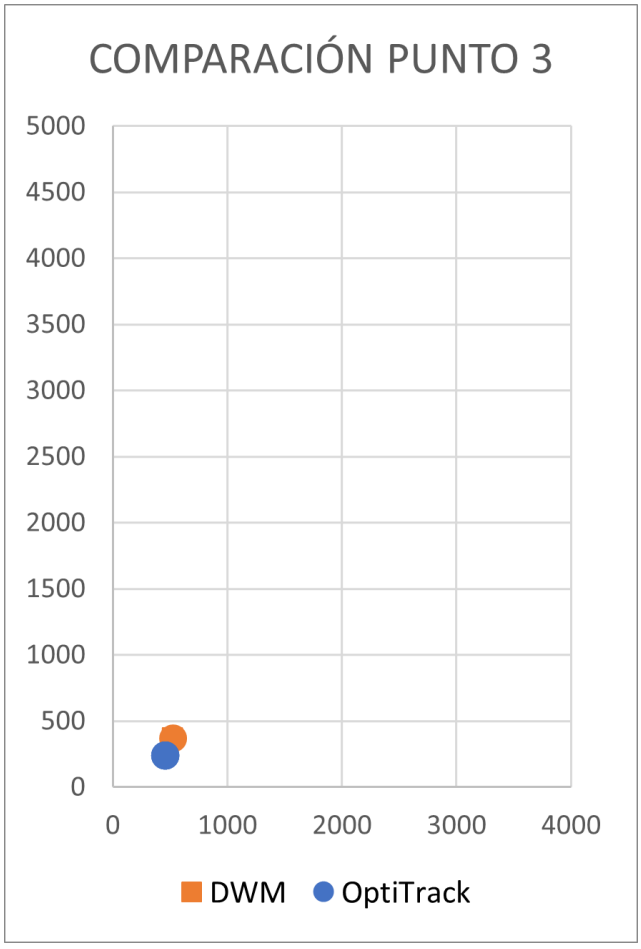


Figura 101: Comparación gráfica del punto 3 medido con OptiTrack y los módulos DWM1001

17.5.6. Gráfica de comparación punto 4

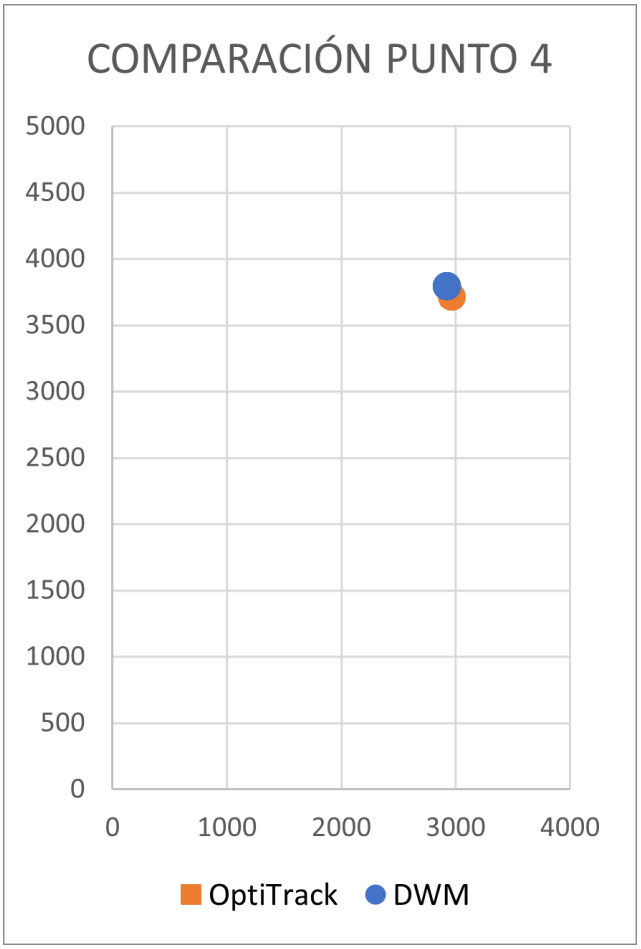


Figura 102: Comparación gráfica del punto 4 medido con OptiTrack y los módulos DWM1001

CAPÍTULO 18

Glosario

c: Es un lenguaje de programación estructurado en el cual se pueden desarrollar aplicaciones y sistemas operativos. 35

Iniciador: Es uno de los módulos DWM1001 configurado como *anchor* que se encarga de corroborar que todos los demás *anchors* tengan el ID correcto de la Network. 29

Linux: Es un sistema operativo de software libre con código abierto. Es bastante robusto, estable y rápido. En este sistema operativo es en donde ROS tiene más funcionalidades y es más eficiente. 31

Little endian: Formato donde el byte menos significativo se almacena en primer lugar, los demás bytes van en orden de significado ascendente. 34

Loop while: Se utiliza para repetir un pedazo de código un número de veces ilimitadas hasta que se cumpla una condición específica. 37

Nibble: En arquitectura de computadoras, es un conjunto de 4 dígitos binarios (bits) o medio octeto. Es la segunda unidad de información más pequeña en la trasmisión y almacenamiento de datos. 34

Nodo: Es un ejecutable que utiliza ROS para comunicarse con con otros nodos. 63

Optitrack: Es un sistema de captura de movimiento 3D de precisión. 57

Package: El software de ROS está organizado en paquetes. Estos son los que contienen los nodos, librerías independientes,documento de configuración o cualquier documento que sea útil para el módulo. 63

Python: Es un lenguaje de programación orientado a objetos que hace énfasis en la legibilidad de su código. 35

SPI: Del inglés *Serial Peripheral Interface* es un protocolo de comunicación que permite la transferencia de información dúplex entre un dispositivo principal y otros periféricos. 34

TLV: Tipo - Longitud - Valor. Es un esquema de codificación se utiliza dentro de los protocolos de comunicación para trasmitir y recibir información. 34

Topic: Los nodos pueden mandar mensajes a los topics, lo que permite que otros nodos se suscriban a estos y reciban los mensajes. 63

UART: Transmisor-receptor asíncrono universal. Este define un protocolo para intercambiar datos en serie entre dos dispositivo. 34

Ubuntu: Es el sistema operativo moderno de código abierto en Linux. Vienen con todas las aplicaciones necesarias para poder trabajar en la computadora. 31

Workspace: Es un conjunto de directorios o carpetas donde se almacenan los paquetes y nodos del entorno. 63