UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica

Trabajo de graduación presentado por Camilo Perafán Montoya para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

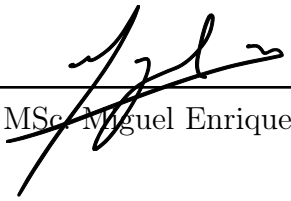
Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica

Trabajo de graduación presentado por Camilo Perafán Montoya para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022

Vo.Bo.:

(f) 

MSc. Miguel Enrique Zea Arenales

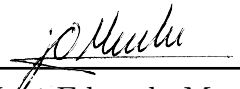
Tribunal Examinador:

(f) 

MSc. Miguel Enrique Zea Arenales

(f) 

MAEB Ing. Pablo Daniel Mazariegos De La Cerda

(f) 

MSc. José Eduardo Morales Espinoza

Fecha de aprobación: Guatemala, 5 de enero de 2022.

Prefacio

Quiero agradecer a toda mi familia, los cuales siempre han estado presentes durante este largo proceso y me han dado aliento, en especial en los momentos de duda. Particularmente quiero darles las gracias a mis padres, Carlos y Rosa, los cuales siempre me han ayudado a dar lo mejor de mí en cada proyecto que realizo. Sin el apoyo de ellos, todo lo que he logrado no sería posible.

Del mismo modo agradezco a mi asesor, Miguel Zea, el cual siempre me exhortó a realizar con excelencia todos los proyectos que abarca este trabajo de graduación. Finalmente quiero agradecer a todos mis amigos y compañeros de carrera, los cuales hicieron esta etapa inolvidable, y los animo a que se sigan esforzando, porque sé que cada uno de ellos llegará muy lejos.

Índice

Lista de figuras

Lista de cuadros

Resumen

En este trabajo de graduación se desarrolló una red de comunicación WiFi para varios agentes, que al funcionar en conjunto al sistema de captura de movimiento OptiTrack, formó un ecosistema de experimentación robótico denominado Robotat. Esta iteración significó un avance significativo con la versión anterior ya que este ecosistema se puede utilizar para diferentes agentes y no solo para un campo de investigación específico. Primeramente, se ensambló y configuró el sistema de captura de movimiento mencionado, para poder así obtener información acerca de la pose de los agentes que se encuentren dentro del área de trabajo. En la implementación de esta sección se pudo comprobar que las mediciones dadas por el sistema coincidían con las mediciones reales, dando como error máximo un valor de 5.28 %.

Posteriormente, se desarrolló una librería en el lenguaje de programación C para que así un microcontrolador, en este caso un ESP32, se pudiera conectar a la red WiFi del ecosistema y pudiera recibir los datos brindados por el OptiTrack por medio del protocolo MQTT, y enviar datos pertinentes por medio de este mismo protocolo. Cabe resaltar que también se desarrolló un programa en el lenguaje de programación Python, el cual fue usado en la computadora asignada al ecosistema. Este programa tomaba los valores de las poses de los agentes, provistos por el OptiTrack, y los publicaba a un broker haciendo uso del protocolo MQTT. En la implementación de esta sección se obtuvo confirmación de que el programa desarrollado en Python publicaba los datos obtenidos del OptiTrack al broker y el microcontrolador, implementando la librería desarrollada, los recibía y decodificaba. Además, se pudo verificar por medio de una medición de latencia, el número máximo de agentes que se podían conectar al ecosistema antes de que la frecuencia de recepción y decodificación de datos fuera menor a 10 Hz, dando un valor máximo de 11 agentes.

Finalmente, se desarrolló una antena inteligente, la cual permitió a cualquier agente no robótico interactuar con el ecosistema. En la implementación de esta sección se pudo comprobar que el incorporar una unidad de medición inercial y un filtro de Kalman permitiría reducir en cierta medida el ruido de los datos obtenidos por el OptiTrack, además de mejorar su exactitud.

Abstract

This thesis work shows the development of a WiFi communications network for several agents that, while working in conjunction with the OptiTrack motion capture system, created an ecosystem for robotic experimentation. This iteration of the Robotat ecosystem meant a significant advance with respect to the previous versions since this ecosystem can be used for different agents and not just for a specific research field. Firstly, the mentioned motion capture system was set up and assembled, in order to obtain information regarding the pose of the agents inside the experimentation area. In the implementation of this section, the data obtained from the system was compared to the real measurement, getting a maximum error of 5.28 %.

Subsequently, a library in the C programming language was developed, so that a microcontroller, in this case an ESP32, could connect to the WiFi network of the ecosystem and could receive the data captured by the OptiTrack, through the MQTT protocol, and send relevant data through the same protocol. It should also be noted that a program used in the computer assigned for the ecosystem was also developed, in this case using the Python programming language. This program took the data of the poses of the agents, provided by the OptiTrack, and published it to a broker, through the MQTT protocol. In the implementation of this section, confirmation was obtained that the Python developed program published the OptiTrack data to the broker and that a microcontroller, implementing the developed library, could recieve and decode said data. Furthermore, the maximum number of agents which could connect to the ecosystem before the recieving and decoding frequency was below 10 Hz was obtained, giving a maximum number of 11 agents.

Finally, a smart antenna was developed in order to allow non robotic systems to interact with the ecosystmem. In the implementation of this section, confirmation was obtained that the use of an inertial measuring unit and a Kalman filter could reduce the noise of the data obtained from the OptiTrack, while also improving the accuracy.

CAPÍTULO 1

Introducción

Este trabajo surge de la necesidad de que la Universidad del Valle de Guatemala, la cual se caracteriza por estar a la vanguardia de la tecnología en la región, tenga un lugar dedicado a la experimentación robótica. Proyectos desarrollados anteriormente, tal como *Diseño e implementación de una red de comunicación wifi e interfaz gráfica para una mesa de pruebas de robótica de enjambre* del año 2020, tenían como parte de sus objetivos desarrollar ecosistemas de pruebas y desarrollo. Sin embargo, estos proyectos solo se enfocaban en un área especifica de investigación, lo cual restringía las aplicaciones y experimentos que se podrían realizar en estos ecosistemas. Por lo tanto, la siguiente etapa de un ecosistema en la universidad debía estar enfocada en que su uso no estuviera restringido a solo un área de investigación. Consecuentemente, este documento presenta el desarrollo y montaje de una red de comunicación WiFi, la cual por medio del uso de un sistema de captura de movimiento, formaría un ecosistema robótico el cual permitiría a uno o varios agentes poder realizar diferentes tipos de experimentos.

Primeramente, se presenta el ensamblaje y configuración del sistema de captura de movimiento comprado por la universidad, el cual ya se había adquirido previo al comienzo de este trabajo de graduación. Esto permitió que se creara un ecosistema en donde se pudiera obtener información acerca de la pose de los agentes dentro de este. La siguiente sección se enfocó en el desarrollo de la red de comunicación WiFi, en donde se presentó la implementación del hardware y software utilizado para crearla. Finalmente, la última sección se enfocó en el desarrollo de una antena inteligente, la cual permitiría a cualquier sistema, electrónico o no electrónico, interactuar dentro del ecosistema robótico.

CAPÍTULO 2

Antecedentes

Para comprender de una manera clara los aspectos relevantes a tomar en cuenta en un ecosistema robótico, además de las ventajas y desventajas que ofrece la tecnología utilizada en estos, se investigaron tres áreas importantes. El primer antecedente presentado corresponde a la implementación de un ecosistema robótico en una universidad de Estados Unidos, lo cual brinda una idea acerca del tipo de tecnología que se puede implementar además de consideraciones importante que se deben tomar en cuenta en la fase de desarrollo. Por otra parte, el segundo antecedente presentado es la tesis de graduación del Ingeniero José Castañeda, en la cual se presenta la implementación de una red de comunicación para una mesa de pruebas de robótica de enjambre, la cual podría ser utilizada como base para así ser aplicada en un diferente ecosistema robótico. Finalmente, el tercer antecedente presentado es el uso de la captura de movimiento en la industria, lo cual brinda una mejor perspectiva acerca de las ventajas y desventajas que ofrecen estos sistemas además de las diferentes aplicaciones que puedan tener. A continuación se presentan más detalles acerca de cada uno de los antecedentes.

Robotarium del Georgia Institute of Technology

El Robotarium es un proyecto realizado por el Georgia Institute of Technology, con el propósito de crear un ecosistema de desarrollo multi-robot y que se pueda acceder de forma remota. En [1] se presenta esta iniciativa como una forma eficiente con la cual investigadores y estudiantes de diferentes partes del mundo podrían probar sus algoritmos y simulaciones de robótica, específicamente de robótica de enjambre.

En este artículo se presentan diferentes consideraciones que se tomaron en cuenta al momento del desarrollo de este proyecto, mostrándose a continuación las más significativas:

- Implementación de medidas de seguridad en el sistema, las cuales deben proteger tanto a este como a los agentes que se encuentren operando.
- Minimizar los costos y complejidad de mantenimiento de los agentes.

- Permitir la interacción intuitiva por medio de una recolección sencilla de datos.
- Permitir un cambio sencillo entre el desarrollo en simulaciones y la ejecución en los robots físicos.
- Permitir que el número de robots dentro del ecosistema puedan variar dependiendo de la aplicación a realizar.

Por otra parte, los robots utilizados dentro del Robotarium están equipados con los chips ESP8266, los cuales permiten comunicación WiFi adhiriéndose al estándar IEEE 802.11 B/G/N con un ancho de banda de 54 MBits/s. Este chip permite comunicación inalámbrica, estimación de la pose y control de bajo nivel en el robot. También utilizan una batería de polímero de litio de 400 mAh, la cual permite un tiempo de operación de 40 minutos en una sola carga y un tiempo de recarga de aproximadamente 45 minutos.

Finalmente, la posición global de todos los agentes dentro del sistema es obtenida por medio de un sistema de captura aéreo. Este sistema utiliza varias cámaras de captura de movimiento en conjunto con el uso de etiquetas ArUco, las cuales son unas etiquetas compuestas de un borde negro y una matriz binaria interna que determina el identificador del robot.

Fase previa del ecosistema

En [2] se presentó el diseño e implementación de una red WiFi e interfaz gráfica para una mesa de pruebas de robótica de enjambre, la cual fue desarrollada en la Universidad del Valle de Guatemala.

Este documento detalla la implementación del protocolo de comunicación MQTT, el cual presenta ventajas en los tipos de topología que pueden implementarse en el sistema. Estas topologías representan el tipo de comunicación que pueden tener los agentes dentro del sistema, ya sea solo comunicación directa con la computadora central o comunicación con la computadora central y con otros agentes dentro del sistema. Además, esta implementación de la comunicación presenta cierto tipo de flexibilidad ya que estas topologías pueden ser diseñadas por el usuario dentro de la interfaz gráfica y no están sujetas a las provistas por defecto.

Por otra parte, para determinar la pose de los agentes dentro del sistema se utilizó visión de computadora, la cual se realizó por medio de una cámara posicionada en la parte superior de la mesa de trabajo y la librería *Open CV* para el procesamiento de la imagen obtenida. Esto le permitiría al usuario de la plataforma poderle indicar a los agentes las trayectorias a seguir al implementar algún algoritmo específico.

Finalmente, también se presentó el software usado en los agentes, y como este se relaciona con el software del sistema, para así poder realizar las simulaciones y evaluar así una variedad de algoritmos de robótica de enjambre. El chip utilizado en los agentes fue un ESP8266 y el software de estos fue desarrollado en Micropython. El software consistía en el uso de tres archivos de trabajo, los cuales permitían la conexión a la red WiFi, la conexión al servidor MQTT y posterior obtención del diseño de red y finalmente, la realización de la rutina de

robótica de enjambre establecida. Cabe resaltar que el programa para los agentes era cargado de manera masiva e inalámbrica, debido a que esto recortaría el tiempo de preparación antes de la experimentación, ya que se debía cargar el mismo programa para todos los agentes dentro del ecosistema.

Uso de la captura de movimiento en robótica

El uso de captura de movimiento en la robótica es un componente de gran importancia para el desarrollo de robots autónomos e inteligentes. En los últimos años, el desarrollo de los sistemas de captura de movimiento ha avanzado de gran manera, y en la actualidad existen una gran variedad de técnicas disponibles que realizan este tipo de capturas. En el artículo *Human motion capture sensors and analysis in robotics* se presentó un análisis de las diferentes tecnologías disponibles para captura de movimiento y como estas pueden ser utilizadas en aplicaciones relacionadas con la robótica [3].

El artículo presenta las diferentes ventajas y desventajas que tiene cada una de las tecnologías disponibles. La tecnología más utilizada es la óptica, la cual se puede dividir en tres categorías: óptica pasiva, óptica activa y óptica sin marcadores. La primera presenta como ventajas un error menor a 1 mm y la habilidad de usar comunicación inalámbrica. Por otra parte, sus desventajas son que solo puede tomar medidas en un espacio limitado y la latencia en el post-procesamiento. La opción óptica activa presenta las mismas ventajas que la opción óptica pasiva, con la única diferencia que esta tiene un mayor rango de medición. Esta opción presenta las mismas desventajas que la opción óptica pasiva. Finalmente, la opción óptica sin marcadores presenta la ventaja que no debe usar ningún sensor, además de no estar restringida a un espacio específico. Sin embargo, esta opción presenta un alto ruido, no presenta valores en tiempo real y tiene un alto costo de post-producción.

Finalmente, el artículo indaga en las aplicaciones que esta tecnología puede tener en la robótica. Entre las aplicaciones más importantes, se encuentran la programación por demostración y la teleoperación. La primera aplicación se enfoca en la extracción de puntos importantes de una trayectoria, la cual fue realizada por un demostrador humano, para que así el efector final de cierto robot la pueda seguir. Por otra parte, la segunda aplicación se enfoca en el uso de esta tecnología para operar robots de manera remota dentro de un área específica. Un punto importante que menciona el documento es la necesidad de combinar diferentes sensores que puedan mejorar los problemas inherentes que tiene esta tecnología, indicando que para poder ofrecer información certera de las poses de los robots dentro de estos sistemas no se puede depender únicamente de la captura de movimiento.

CAPÍTULO 3

Justificación

En la actualidad, los ecosistemas de experimentación para robótica están tomando más y más relevancia, a medida que la complejidad de los robots usados aumenta. Tener a disposición un lugar especializado en donde se puedan realizar las pruebas de diferentes procesos robóticos permite al investigador examinar de una manera controlada los algoritmos desarrollados, creando no solo un lugar seguro para el experimento en cuestión, sino también agilizando el proceso de innovación. Sin embrago, los ecosistemas de experimentación actuales están enfocados solamente en cierto tipos de investigaciones robóticas, entre las cuales se destacan el desarrollo de robots móviles y la robótica de enjambre. Al estar los ecosistemas enfocados solamente en un área especifica de la robótica, el desarrollo de aplicaciones conjuntas entre robots se ve limitado y por ende la experimentación con situaciones reales más complejas no se pueda realizar.

Por otra parte, el desarrollo de ecosistemas de experimentación en Guatemala y la región es casi nulo, a pesar de que la robótica se ha ido adoptando cada vez más en la industria. Por tal razón es de suma importancia que la Universidad del Valle de Guatemala tome la iniciativa en la creación de este ecosistema, que permita no solo a los miembros de la universidad, sino a también a personas externas con experiencia en el área a aportar su conocimiento en el desarrollo e innovación en la robótica. La creación del ecosistema Robotat, el cual se encontrará en el laboratorio de robótica dentro del Centro de Innovación y Tecnología (CIT), será un gran paso en esta dirección. Una parte fundamental de este sistema es la red de comunicación empleada y el sistema usado para la recolección de datos, la cual permitiría al investigador tener más información acerca de uno o varios agentes que se encuentren dentro del ecosistema. Sin el establecimiento de esta red de comunicación, el desarrollo de aplicaciones conjuntas entre robots no podría realizarse, ya que aunque se pudiera saber externamente la pose de los robots, esta no podría ser transmitida a cada uno de ellos y por ende los robots no tendrían retroalimentación acerca de la pose de los demás relativa a la de él mismo.

Como se explicó en la sección de antecedentes, el tipo de comunicación más utilizada en este tipo de ambientes es la comunicación WiFi, que por medio de la implementación del protocolo de comunicación MQTT, permite una mayor flexibilidad en el número de agentes que se encuentren en el ecosistema y en el tipo de información que los agentes reciben. Por lo tanto, esto indica que el mejor diseño de red de comunicación para el ecosistema es el descrito anteriormente. Por otra parte, los sistemas de captura de movimiento permiten que de una manera sencilla se pueda obtener una gran cantidad de información acerca de la pose de los agentes que se encuentren dentro del ecosistema. Sin embargo, como se presentó en los antecedentes, dependiendo del sistema de marcadores utilizados, se presentarán diferentes inconvenientes en la lectura de los datos, de los cuales el más importante es la latencia post-procesamiento. Por lo tanto es necesario combinar la lectura de diferentes sensores para mejorar estos problemas, lo cual permitirá no solo obtener una gran cantidad de información del agente, sino también, que esta pueda ser trasmitida de la manera más eficiente posible.

Partiendo de lo mencionado anteriormente, en este trabajo de graduación se presenta la especificación e implementación de una red de comunicación WiFi, que en conjunto con el sistema de captura de movimiento adquirido por la universidad, forme un ecosistema de experimentación para robótica que permita la interacción entre varios agentes. La importancia de este proyecto radica en que este ecosistema permitirá que no solo se pueda realizar investigación y desarrollo en un área específica de la robótica, sino que se pueda experimentar con algoritmos más complejos que permitan la interacción entre diferentes plataformas robóticas tales como manipuladores seriales, drones o robot móviles.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Especificar e implementar una red de comunicación WiFi multiagente que opere en conjunto al sistema de captura de movimiento OptiTrack para formar un ecosistema de experimentación para robótica.

4.2. Objetivos específicos

- Ensamblar y configurar el sistema de captura de movimiento OptiTrack para que permita obtener información sobre la pose de los agentes que se encuentren dentro de su espacio de trabajo.
- Implementar una red de comunicación WiFi local que implemente el protocolo MQTT para la comunicación de múltiples clientes con un servidor empleando tópicos.
- Diseñar e implementar una antena inteligente que combine la información del sistema de captura de movimiento con la obtenida por una unidad de medición inercial, que permita la obtención e intercambio de información entre agentes mediante comunicación inalámbrica.

CAPÍTULO 5

Alcance

Este trabajo se centra principalmente en el desarrollo e implementación de una red de comunicación inalámbrica, que en conjunto con un sistema de captura de movimiento, permita tener un área para experimentación robótica. La diferencia con respecto a proyectos similares desarrollados en la universidad, tal como el presentado en los antecedentes, es que el Robotat se desarrolló para que pudiera ser utilizado por una gran variedad de robots, y así estos, si fuera el caso, pudieran realizar rutinas en conjunto. Consecuentemente, en este ecosistema no se tuvo la funcionalidad de cargar de manera inalámbrica los programas a los microcontroladores ya que se consideró que esta característica no aportaba mayor diferencia en el tiempo de preparación antes de la experimentación, al enfocarse principalmente este ecosistema en el uso de diferentes tipos de agentes. Además, en este proyecto se utilizó un sistema de captura de movimiento para obtener la pose de los agentes lo cual brinda un rango más amplio de análisis, comparado a algoritmos de visión de computadora, los cuales fueron utilizados en [2], como fue presentado en los antecedentes.

Por otra parte, el uso del sistema de captura de movimiento se restringió solo a cuerpos rígidos y no se utilizó otras funcionalidades de este, tal como los trajes de captura de movimiento para humanos. Por otra parte, la información de todos los agentes está siendo enviada en un mismo tópico y los agentes solo conocen la información de la pose de si mismos. Sin embargo, esta topología de red puede ser modificada. Finalmente, el desarrollo de la antena inteligente permitiría a cualquier sistema, independientemente que sea electrónico o no, conectarse al ecosistema, para que así la información obtenida pueda ser utilizada por el investigador para el propósito deseado. Cabe resaltar que las librerías que permitían la interacción de la IMU con el ESP32 fueron desarrolladas por el estudiante de ingeniería mecatrónica Hans Burmester para la tesis *Diseño de controlador de vuelo para cuadricóptero con capacidad de integrarse a ecosistema Robotat vía inalámbrica Wi-Fi*, y por lo tanto solo fueron implementadas en el proyecto. Además, el uso de filtros para la fusión de sensores se enfocó en la validación, ya que la implementación de este en el ESP32 estaba por fuera de los objetivos planteados.

Cabe resaltar que debido a la situación ocasionada por la pandemia de COVID-19, la experimentación y obtención de resultados fue afectada. Esto se debe a que no se pudo adquirir componentes esenciales de una manera oportuna, por lo que las pruebas relacionadas con la red de comunicación y el uso del sistema OptiTrack se vieron atrasadas. Además este acontecimiento limitó el tiempo de experimentación que se tuvo en la universidad para realizar así los proyectos que componían este trabajo de graduación.

CAPÍTULO 6

Marco teórico

6.1. Sistemas de captura de movimiento

Los sistemas de captura de movimiento son un tipo de tecnología que graba los movimientos de personas u objetos y transfiere los datos correspondientes de estos movimientos a una aplicación, presentando la Figura 1 un ejemplo de esto. Al modelar movimientos reales en un ecosistema virtual, esto permite que los datos puedan ser utilizados en diferentes áreas, desde aplicaciones de investigación en el área de ingeniería hasta aplicaciones en el cine y los videojuegos [4].

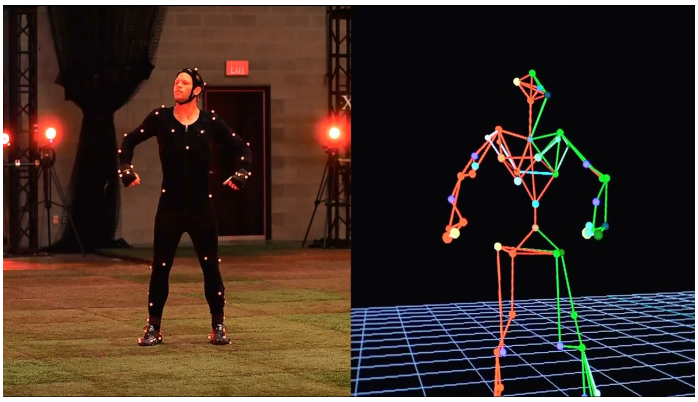


Figura 1: Sistema de captura de movimiento [5].

Existen diferentes técnicas de captura de movimiento, entre las cuales, las más utilizadas son [4]:

- Técnicas ópticas pasivas.
- Técnicas ópticas activas.

- Técnicas sin marcadores.
- Técnicas inerciales.

Las técnicas ópticas pasivas hacen uso de marcadores retroreflectivos que son posicionados en el objeto o persona a grabar. Estos reflejan la luz infrarroja generada por las cámaras especializadas, y una vez este reflejo es detectado por estas últimas, el programa lo utiliza para calcular la posición de los marcadores en el espacio tridimensional. Por otra parte, las técnicas ópticas activas hacen uso de los mismos procedimientos presentados en la técnica anterior, siendo la única diferencia que los marcadores no reflejan la luz sino que la emiten, lo cual también hace que estos requieran una fuente de poder para funcionar.

Por otra parte, en las técnicas no ópticas se encuentran las técnicas sin marcadores. Esta técnica se basa en el uso de cámaras con sensibilidad a la profundidad y algoritmos especializados para seguir y grabar los cuerpos u objetos. Este método puede llegar a ser más conveniente ya que no es necesario posicionar sensores en el cuerpo u objeto a analizar; sin embargo, el no hacer uso de sensores hace que esta técnica sea menos precisa que las técnicas ópticas u otras que sí usan sensores. Finalmente, las técnicas inerciales no necesitan cámaras para operar ya que hacen uso de unidades de medición inerciales (IMU), las cuales tienen giroscopios, magnetómetros y acelerómetros, y envían diferentes sets de datos que permiten determinar la posición del cuerpo u objeto en el espacio tridimensional.

6.2. Sistema OptiTrack

OptiTrack es una compañía que ofrece productos relacionados con la captura de movimientos, los cuales van desde las cámaras especializadas hasta el software que registra los datos enviados por el sistema de captura de movimiento. Este sistema fue el comprado por el Departamento de Ingeniería Electrónica, Mecatrónica y Biomédica de la Universidad del Valle de Guatemala, y por lo tanto será utilizado para el desarrollo del ecosistema Robotat [6].

Figura 2: Cámara *Prime*$^{\times}$*41* [7].

Las cámaras adquiridas fueron las *Prime*$^{\times}$*41*, mostrada en la Figura 2, y las especificaciones técnicas de estas se presenta a continuación [7]:

- **Resolución**: 2048x2048.

- **Velocidad de fotogramas**: 180 Hz.
- **Latencia**: 5.5 ms.
- **Precisión 3D**: +/- 0.10 mm.
- **Rango para marcadores pasivos**: 30 m.
- **Rango para marcadores activos**: 45 m.
- **Anillo de LEDs**:
  - 20 LEDs.
  - Luces del tipo infrarrojo de 850 nm.
- **Conexiones**:
  - Puerto de datos GigE.
  - Sincronización de la cámara por medio de Ethernet.
  - Alimentación por medio de PoE o PoE+.
- **Tamaño**: 12.6cm x 12.6 cm x 13.2 cm.
- **Peso**: 1.36 Kg.

En un sistema de captura de movimiento, una buena colocación de cámaras permitirá usar de manera eficiente las imágenes capturadas por cada una de estas, y por ende mejorará significativamente la calidad de la captura. La colocación de las cámaras dependerá de la aplicación, sin embargo, ciertos puntos se deben seguir cumpliendo independiente de esto. Primeramente, para calcular de manera precisa la localización 3D de marcadores es necesario que estos sean capturados simultáneamente por al menos dos cámaras del sistema. Por otra parte, es recomendado que las cámaras tengan una elevación considerable para maximizar así la cobertura del sistema. Finalmente, la separación entre cámaras debe ser consistente para que así estas no capturen imágenes tan similares y por ende, no contribuyan significativamente en la reconstrucción.

Por otra parte, para realizar una correcta calibración se debe primero preparar y optimizar el volumen de captura del sistema. Posteriormente, se debe aplicar un proceso de *masking*, el cual permite ignorar los reflejos que se puedan dar ya sea por ventanas o por las otras cámaras instaladas. Posteriormente se debe realizar el proceso de *wanding*, en el cual se mueve una vara de calibración en frente de las cámaras para que así se pueda calcular la posición y orientación en el espacio 3D de cada una de estas. Finalmente, al haber obtenido un resultado satisfactorio en el paso anterior y tener una reconstrucción fidedigna de la posición y orientación de las cámaras, se define el origen y el plano de tierra y se guarda el archivo de calibración, para que pueda ser reutilizado en otras sesiones de captura de movimiento.

Finalmente, el software usado para grabar la información obtenida de los cuerpos u objetos en el sistema es *Motive*. Este también permite realizar la calibración del sistema, además de que por medio de un Sistema de Desarrollo de Software (SDK) la información que este recolecte puede ser enviada a otras aplicaciones tales como MATLAB, para su posterior uso.

6.3. Comunicación WiFi

El WiFi es una tecnología de red inalámbrica, la cual permite que diferentes dispositivos puedan interactuar con Internet o solamente entre sí, permitiendo el intercambio de información y estableciendo así una red. Este se basa en ondas de radio, las cuales operan en las frecuencias de 2.4 GHz en el estándar 802.11 n y 5 GHz en el estándar 802.11 ac. El estándar IEEE 802.11 define los protocolos que permiten la comunicación con dispositivos inalámbricos, entre los cuales se incluyen routers y puntos de acceso inalámbrico. Esta tecnología está teniendo una gran aceptación en el desarrollo del área del internet de las cosas, la cual se enfoca en crear sistemas que puedan interactuar entre sí por medio del uso de una red ya sea privada o por medio de Internet. Sin embargo, una de las mayores desventajas de esta tecnología es su falta de seguridad, la cual puede poner en riesgo la información de los usuarios que se encuentren en la red. Por lo tanto, se han ido adoptando diferentes métodos de seguridad dentro de las conexiones, entre los cuales los más utilizados son: WEP, WPA y WPA2 [8] [9] [10].

- **WEP**: el primer cifrado desarrollado para la redes WiFi. Las claves en este tipo de cifrado eran de 64 bits para una contraseña de 8 caracteres, y de 128 bits para una contraseña de 14 caracteres. Este tipo de cifrado es fácil de configurar pero al ser tan antiguo no provee una buena seguridad.
- **WPA**: cifrado desarrollado en respuesta a las fallas y vulnerabilidades presentadas en el cifrado tipo WEP. Este salió al mercado con el cifrado TKIP (Temporal Key Integrity Protocol), el cual está obsoleto y es un riesgo a la seguridad de la red. Este tipo de seguridad funcionaba con una cifrado de 256 bits que requería una contraseña de 8 a 64 caracteres, y que permitía conectarse a la red a cualquier dispositivo que tuviera esta contraseña.
- **WPA2**: La principal diferencia con respecto a la versión anterior radica en el uso del AES, el cual permite tener claves más largas y seguras ya que realiza un cifrado por bloques. Además este también implementa CCMP, el cual es un protocolo mejorado de encriptación, el cual sustituye al antiguo TKIP.

6.4. Sistemas de transmisión de información

Al realizar un intercambio de datos entre equipos es necesario definir el sistema de transmisión y el método de acceso. Por lo tanto la ISO especificó el modelo de referencia ISO/OSI, el cual es un estándar que describe redes de comunicación y las diferentes partes en las que se divide. Este modelo presenta una serie de siete capas, independientes entre sí, las cuales son empleadas para construir un sistema de comunicación. El modelo OSI es un manual de buenas prácticas para definir un sistema de comunicación, por lo que no es obligatorio su cumplimiento. Las capas de este modelo son [11]:

- **Capa 7**: Capa de aplicación.
- **Capa 6**: Capa de presentación.

- **Capa 5**: Capa de sesión.
- **Capa 4**: Capa de transporte.
- **Capa 3**: Capa de red.
- **Capa 2**: Capa de enlace de datos.
- **Capa 1**: Capa física.

La Capa de Aplicación comprende todos los servicios de enlace con las diferentes aplicaciones de comunicación. Normalmente esta capa incluye protocolos de uso general, como la forma de iniciar y cerrar una sesión de comunicaciones. La Capa de Presentación convierte los datos de la Capa 7 al lenguaje definido para la transmisión y modifica los datos recibidos para que la aplicación reciba los datos conforme su necesidad.

Por otra parte, la Capa de Sesión sincroniza las relaciones de comunicación, permitiendo establecer una sesión de comunicación entre dos capas de aplicación, evitando que ambos lados realicen la misma operación simultáneamente, estableciendo puntos de chequeo de información y configurando el tipo de diálogo de la comunicación. La Capa de Transporte es la encargada de ofrecer al usuario un enlace entre nodos seguro, entregando datos libres de error a la Capa 5.

La Capa de Red es la encargada de la operatividad de la red, controlando la ruta de comunicación de entre el emisor y el receptor, seleccionando la ruta que deben seguir los datos para llegar. Por otra parte, la Capa de Enlace de Datos tiene como función asegurar la transmisión de los bits entre dos sistemas, recogiendo los datos de la Capa 3 y formando los paquetes de envío y viceversa. Esta Capa también identifica los agentes que se encuentran en la red usando su dirección de hardware asignada. Finalmente, la Capa Física se encarga de transmitir los bits en el orden definido en la Capa 2 a través del soporte físico. Esta capa también define los sistemas de modulación y demodulación de la señal transmitida y/o recibida, además de las señales de control que determinan el orden de transmisión.

6.5. Protocolo de transporte de información MQTT

MQTT (Message Que Telemetry Transport) es un protocolo de transporte de información Cliente/Servidor, el cual está basado en publicaciones y subscripciones a tópicos. Por lo tanto, cada vez que un mensaje sea publicado este será recibido por el resto de agentes adheridos al tópico correspondiente. El protocolo MQTT funciona sobre cualquier protocolo de red que tenga soporte bi-direccional y sin pérdida de datos [12].

Debido a que este protocolo desacopla el publicador del suscriptor, las conexiones con los clientes están manejadas por un broker, es decir un servidor. Los clientes son cualquier dispositivo que corre una librería MQTT y se conecta a un servidor MQTT a través de una red, por lo que tanto los publicadores como los suscriptores son clientes. Por otra parte, el broker es el encargado de recibir todos los mensajes para posteriormente filtrarlos y determinar quién está suscrito a cada mensaje, y finalmente enviar estos mensajes a los

correspondientes clientes suscritos [13]. Un ejemplo de uso del protocolo MQTT se presenta en la Figura 3.

Las características principales de este protocolo son:

- Uso de mensajes "broadcast"para subscripción y publicación de datos con independencia de aplicaciones.
- Transporte de mensajes de una manera transparente y optimizando el flujo de datos, reduciendo así el tráfico en la red.
- Posee un mecanismo que notifica cuando ocurren desconexiones inesperadas.

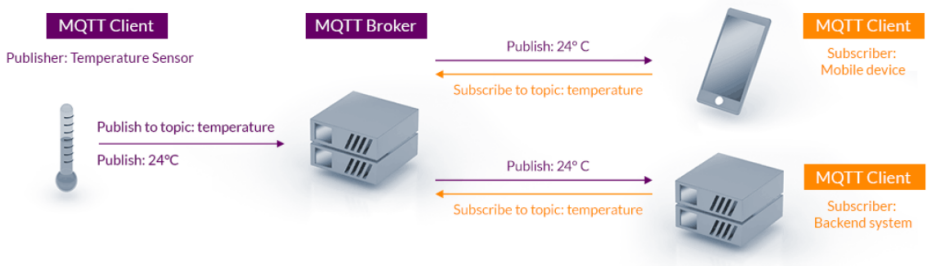


Figura 3: Protocolo MQTT [13].

6.6. Topologías de red

Las topologías de red son las diferentes estructuras de intercomunicación en la que las redes de transmisión de datos entre dispositivos pueden ser organizadas. Las topologías de red más comunes se presentan en la Figura 4, con una explicación mas detallada de estas presentada posteriormente [11]:

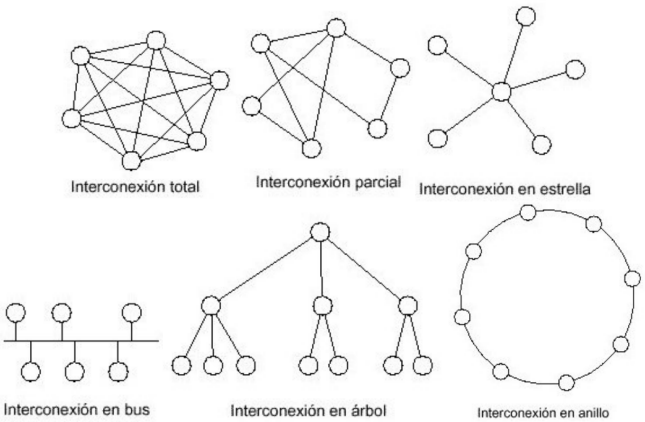


Figura 4: Topologías de red [11].

- **Red en estrella**: Cada nodo está conectado a un nodo central, el cual se encarga del control de acceso a la red por el resto de los nodos. Este tipo de red se utiliza

cuando los nodos externos no se encuentren muy distanciados del nodo central, ya que el encontrarse lejos puede elevar el costo de cableado.

- **Red en bus**: Todos los nodos se conectan a un único medio de comunicación (bus), por lo que la comunicación es visible para todos los nodos conectados. Esta topología permite agregar o quitar nodos sin interferir con el resto. También permite cubrir grandes distancias haciendo uso de repetidores y amplificadores, aunque los nodos deben estar conectados al bus principal por medio de segmentos cortos para así optimizar la velocidad de transmisión y recepción de datos.
- **Red de anillo**: Los nodos son conectados en serie alrededor de un anillo. Los mensajes son transmitidos en una dirección, pasando por todos los nodos necesarios hasta llegar al nodo destino. Una las mayores desventajas de esta topología es el retardo grande para un número de nodos elevados.
- **Red de árbol**: En esta topología se encadenan diferentes estructuras de bus de diferente longitud y características diferenciadas.
- **Red de malla**: Este tipo de topología proporciona múltiples enlaces físicos entre los nodos de la red, permitiendo así que existan múltiples caminos de interconexión entre dos nodos. Esta interconexión puede ser total si todos los nodos están conectados de forma directa entre ellos, mientras que es parcial cuando no todos los nodos pueden conectarse de manera directa.

6.7. Eclipse Mosquitto

Eclipse Mosquitto es un servidor de código abierto el cual permite implementar el protocolo MQTT en sus versiones 5.0, 3.1.1 y 3.1. Debido a que Mosquitto tiene un tamaño reducido, este puede ser implementado en una gran variedad de dispositivos, no exigiendo demasiados recursos para funcionar correctamente. Este puede ser implementado en diferentes lenguajes de programación, tales como *Python*, *C* y *C++*, por lo que provee una gran flexibilidad de implementación [14].

6.8. Placa de desarrollo ESP32

El ESP32 es un microcontrolador de bajo costo y bajo consumo que cuenta con la capacidad de implementar conexiones inalámbricas por medio de WiFi. Las características principales de este microcontrolador son [15]:

- **Velocidad de reloj**: 240 MHz.
- **Procesador**: Tensilica Xtensa 32-bit LX6.
- **ROM**: 448 KiB.
- **SRAM**: 520 KiB.

- **Tipos de comunicaciones admitidas**: I2C, SPI, CAN 2.0 y UART.
- **Conectividad inalámbrica**: WiFi por medio de la norma IEEE 802.11 B/G/N.
- **Voltaje de operación**: 2.2 a 3.6 V.
- **Temperatura de operación**: -40 °C a 125 °C.

Este microcontrolador puede ser programado en una gran variedad de lenguajes de programación, tales como *C++ de Arduino*, *Micropython* o *C*. Esto le brinda una mayor flexibilidad ya que permite escoger el lenguaje que más se adecue a la aplicación deseada, además de proveer una gran variedad de librerías que permiten la utilización de todos los recursos provistos por el microcontrolador.

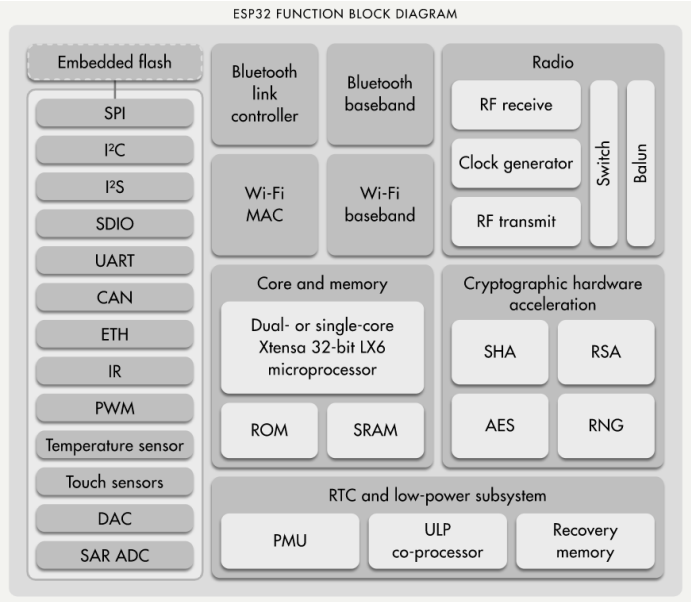


Figura 5: Diagrama de bloques del ESP32 [15].

6.9. Unidad de Medición Inercial (IMU)

Una unidad de medición inercial es un dispositivo que contiene una variedad de sensores y permite medir diferentes características de un objeto tales como su velocidad, dirección o aceleración. La IMU utiliza los diferentes datos de los sensores y los combina para así determinar de manera más precisa la característica deseada. Los sensores más comunes que se encuentran en una IMU son:

- **Acelerómetro**: Mide la velocidad y aceleración del objeto.
- **Giroscopio**: Mide la rotación y la tasa de rotación del objeto.
- **Magnetómetro**: Establece la dirección cardinal.

Sin embargo, las desventajas de estos sensores es que son susceptibles a errores que se pueden acumular con el tiempo. Esto es debido a que el dispositivo siempre está calculando los cambios con respecto a él mismo, y no con respecto a un marco inercial, lo cual hace que la IMU redondee pequeñas fracciones en su cálculo, lo cual con el tiempo puede llevar a error significativos en las mediciones. Incluso si estos errores son corregidos, es recomendable que las mediciones tomadas por la IMU sean complementadas con las mediciones de otros sensores [16].

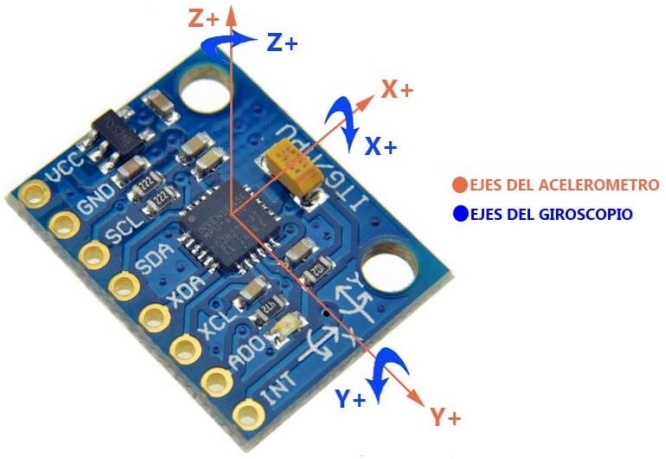


Figura 6: IMU 6050 de 6 Grados de Libertad [17].

6.10. Fusión de sensores

La fusión de sensores es una metodología que permite sobreponerse a las limitaciones que tienen los sensores por separado, al obtener y mezclar datos de diferentes sensores para crear así una información más certera. Esta información más robusta puede ser posteriormente utilizada para hacer cálculos o tomar decisiones [18].

Existen varios algoritmos para la fusión, siendo los más utilizados:

- Filtro complementario.
- Filtro de Kalman.

El filtro complementario es un tipo de filtro usado cuando se tienen dos mediciones diferentes de dos sensores que estiman a una misma variable y las propiedades del ruido hacen que las mediciones provean información correcta en diferentes regiones de frecuencia. Por lo tanto, cada uno de los datos medidos entra a su filtro correspondiente y posteriormente se suma para así obtener el valor de la variable deseada [19].

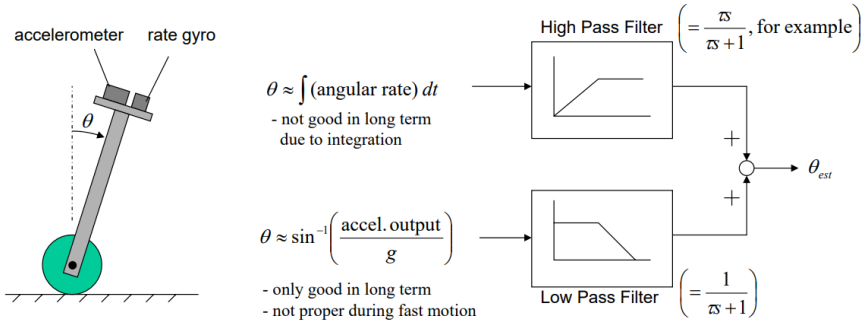


Figura 7: Ejemplo de uso de un filtro complementario para determinar el ángulo de inclinación [19].

Por otra parte, el filtro de Kalman es un conjunto de ecuaciones matemáticas que implementan un estimador óptimo del tipo predictor-corrector. Por lo tanto este procesa todas las medidas disponibles, independientemente de su precisión, y estima el valor actual de la variable de interés [20]. Esto se puede realizar debido a:

- El conocimiento del sistema y los dispositivos dinámicos de medida.
- La descripción estadística de los ruidos, errores de medida e incertidumbre en los modelos dinámicos.

El filtro de Kalman tiene varios planteamientos, cambiando el tipo de medidas con el que puede trabajar, ya sean discretas o continuas. Por otra parte, este es un algoritmo recursivo, por lo que no requiere almacenar todos los datos previos para poder procesarlos de nuevo cada vez que una nueva medida sea tomada.

El filtro de Kalman usado para fusión de sensores es solo un caso especial del filtro de Kalman general, el cual puede ser utilizado para otras aplicaciones tales como observadores de estado. Primeramente, antes de definir el algoritmo del filtro, es necesario definir las ecuaciones que definen un sistema dinámico [21].

Dejando que $x_t \in \mathbb{R}^k$, para $t = 1,2,3,...$ denote los estados y $z_t \in \mathbb{R}^d$, para $t = 1,2,3,...$ denote la mediciones que evolucionan acorde al sistema dinámico LTI:

$$x_t = Fx_{t-1} + \delta_t, \tag{1}$$

$$z_t = Hx_t + \epsilon_t. \tag{2}$$

para $t = 1,2,3,...$ Se asume que los términos de ruido δ_t y ϵ_t tienen media de cero y covarianzas $Q \in \mathbb{R}^{k\times k}$ y $R \in \mathbb{R}^{d\times d}$, respectivamente. También se asume que el estado inicial x_0 y todos los términos de ruido blanco no correlacionado son mutuamente independientes. Esto finalmente indica que (1) y (2) son el modelo del proceso y el modelo de las mediciones respectivamente.

El filtro de Kalman entonces es un método para realizar un estimación secuencial en el modelo (1) y (2). Este utiliza estimaciones pasadas $\hat{x}_1, ..., \hat{x}_t$ y mediciones $z_1, ..., z_{t+1}$ para

formar así un estimado $\hat{x}_{t+1}$ del estado x_{t+1} por medio de:

$$\bar{x}_{t+1} = F\hat{x}_t, \tag{3}$$

$$\hat{x}_{t+1} = \bar{x}_{t+1} + K_{t+1}(z_{t+1} - H\bar{x}_{t+1}), \tag{4}$$

donde $K_{t+1} \in \mathbb{R}^{k\times d}$ es la ganancia de Kalman en el tiempo $t+1$.

6.11. Normas para el diseño de PCBs

La norma ANSI IPC-2221 presenta los principios y los materiales del diseño físico genérico de circuitos impresos. Primeramente, se presentan las reglas que rigen el enrutamiento de los conductores y la ubicación de los componentes, para así evitar que se presenten perturbaciones electromecánicas [22] [23]. Las reglas principales que rigen el enrutamiento de los conductores son:

- Debe procurase que la pista conductora sea lo más corta posible entre componentes.
- Evitar esquinas totalmente rectas, usando mejor esquinas con ángulos obtusos.
- Usar un espaciado considerable entre conductores y componentes, y sólo utilizar el espaciado mínimo cuando el espacio sea limitado.

Por otra parte, las recomendaciones relacionadas con la ubicación de los componentes son:

- Los componentes más importantes del circuito deben colocarse en primer lugar para así utilizar menos longitud de pistas.
- Ciertos componentes pueden influir en el correcto funcionamiento de componentes cercanos, por lo que estos deben estar lo suficientemente separados para reducir el efecto negativo.
- Los componentes deben ubicarse de manera vertical u horizontal para así ofrecer un mejor resultado con respecto a la presentación.

Por otra parte, el ancho de la pista conductora está determinada por tres factores: la densidad del empaquetado del componente, el espaciado mínimo entre conductores y/o componentes y las limitaciones geométricas que este pueda tener debido al contorno de los componentes. Este valor por ende depende de la corriente que puede fluir en la pista y el calor máximo que este puede permitir. Por lo tanto, se busca que la pista sea lo más ancha posible para así las señales que pasen por esta sean de baja impedancia.

CAPÍTULO 7

Sistema de captura de movimiento OptiTrack

Una parte fundamental del ecosistema Robotat es el sistema de captura de movimiento. Este es capaz de brindar información acerca de la pose de los robots que se encuentren dentro del ecosistema y, sin el uso de este, las siguientes fases del proyecto no tendrían sentido. Consecuentemente el objetivo principal de este capítulo fue configurar el hardware y software del sistema de captura de movimiento, tal que pudiera proveer información fidedigna acerca de la pose de un agente que se encuentre dentro del ecosistema.

7.1. Metodología

En esta sección se describen los pasos realizados en el montaje de todos los componentes necesarios para el buen funcionamiento del sistema de captura de movimiento, dividiendo estos pasos en instalación de hardware e instalación de software. Cabe resaltar que este sistema fue adquirido previo al inicio de este proyecto de graduación, por lo que no se realizó un trade study para determinar la mejor opción a escoger en los sistemas de captura de movimiento.

7.1.1. Instalación de hardware

Primeramente fue necesario instalar todos los componentes de hardware antes de poder usar los componentes de software, ya que el uso satisfactorio de estos últimos dependía de la correcta configuración de los primeros. El hardware instalado durante el montaje del ecosistema fue:

- 6 cámaras Prime$^{\times}$41.
- 6 abrazaderas Manfrotto.
- 6 trípodes de cámara.
- 1 switch NETGEAR ProSafe GS728TPPv2.
- 1 tarjeta de red Intel I210-T1.

El primer componente instalado fue la tarjeta de red, presentada en la Figura 8. Esta fue posicionada en la computadora provista por el departamento para el ecosistema y tenía como objetivo proveer un puerto ethernet extra para conectar el switch.

Figura 8: Tarjeta de red Intel I210-T1 [24].

Posteriormente, se instalaron las abrazaderas en los trípodes provistos y se posicionaron las cámaras en estas abrazaderas, como se presenta en la Figura 9. Los trípodes utilizados podían alcanzar una altura de 3 metros, soportar como máximo un peso de 140 libras y estaban compuestos de dos tubos los cuales permitían el ajuste de la altura por medio de la selección del posicionamiento de un pin en uno de los agujeros que se encontraban en los tubos. Por otra parte, el uso de las abrazaderas permitió tener un mayor control en el posicionamiento de la cámara y en la inclinación de esta.

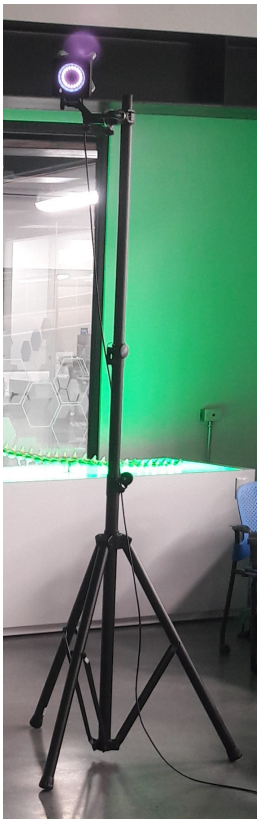


Figura 9: Posicionamiento de una de las cámaras del ecosistema.

Al tener las cámaras instaladas en sus respectivos trípodes, se procedió a posicionarlas en la configuración deseada. Para una correcta configuración de las cámaras fue necesario definir tres aspectos clave: la elevación, el enfoque y la inclinación de estas.

Con base en las recomendaciones de la compañía, se utilizaron elevaciones altas de las cámaras para así maximizar la cobertura de captura en el volumen, configurando el tubo inferior con dos agujeros y el tubo superior con siete agujeros. Para estandarizar la inclinación de las cámaras primeramente se utilizó un nivel para asegurar que las cámaras estuvieran rectas y posteriormente se inclinó toda la estructura 30° con respecto a la rueda presentada en la Figura 10. Por otra parte, se modificó el enfoque de las cámaras para que pudieran visualizar los marcadores de manera clara, comprobando esto por medio de la visualización de la imagen en escala de grises como se presenta en la Figura 11. Esto se realizó girando el anillo de enfoque, como se presenta en la Figura 12, a favor de las agujas del reloj hasta el valor máximo y el anillo de F-Stop hasta el valor de 4. Finalmente, para la colocación de las cámaras se decidió realizar un posicionamiento rectangular, el cual permitió tener la mayor cobertura del espacio posible, dadas las circunstancias de espacio presentadas en el laboratorio. La configuración mencionada se puede observar en la Figura 13 y la Figura 14.

Figura 10: Inclinación de la cámara.

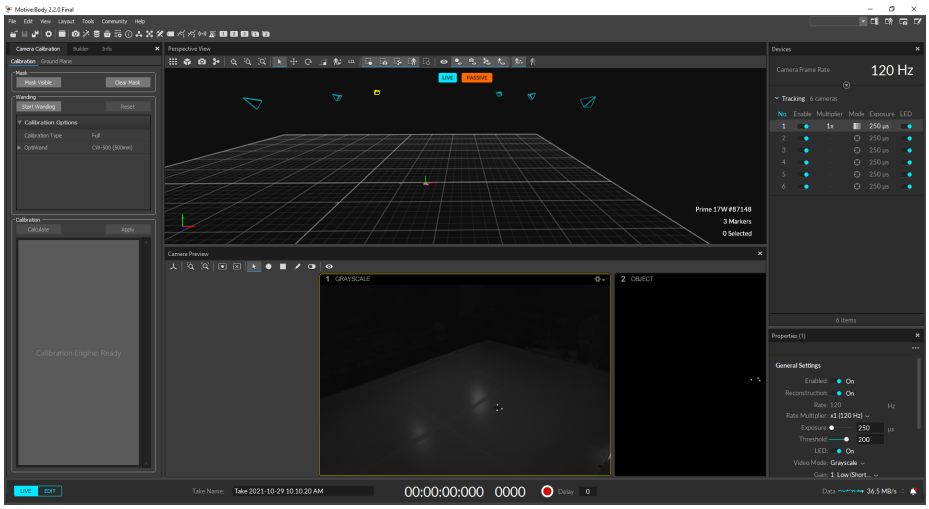


Figura 11: Visualización en escala de grises de la imagen capturada por una de las cámaras.

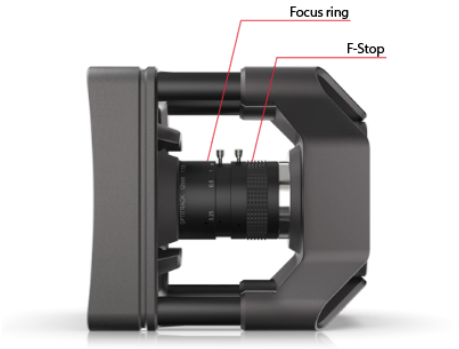


Figura 12: Anillo de enfoque en la cámara Prime$^{\times}$41 [6].

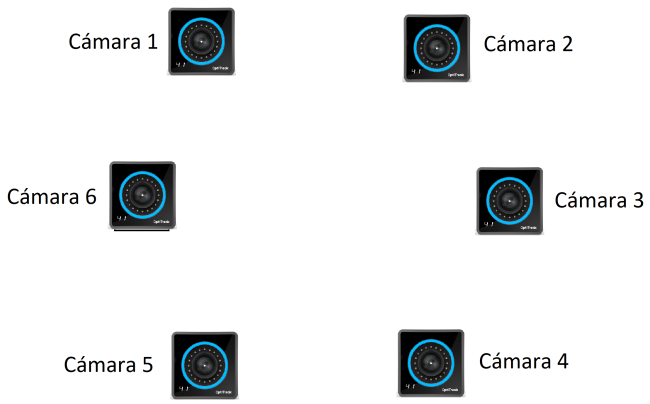


Figura 13: Vista en planta de la colocación de las cámaras en el ecosistema.

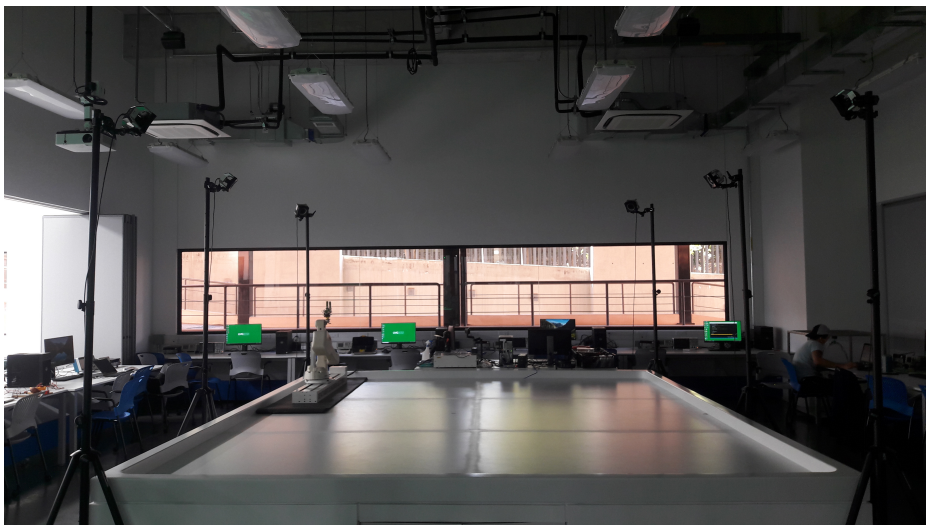


Figura 14: Vista frontal de la colocación de las cámaras en el ecosistema.

Finalmente, se instaló el switch, el cual permitió que el software instalado en la computadora asignada al ecosistema pudiera obtener y leer la información capturada por las cámaras, además de ser la fuente de alimentación de estas últimas. La conexión entre las cámaras y el switch y entre el switch y la computadora se realizó por medio de un cable Ethernet categoría 6, mostrándose esta conexión en la Figura 15.

Figura 15: Conexiones de las cámaras y computadora al switch NETGEAR ProSafe GS728TPPv2.

7.1.2. Instalación de software

Al ya haber ensamblado los componentes del ecosistema relacionados con el hardware, se procedió a descargar y configurar el software, el cual iba a ser implementado en la computadora asignada. El software instalado durante el montaje del ecosistema fue:

- Programa Motive para captura de movimiento óptico.
- Kit de desarrollo de software NatNet.

El primer componente implementado fue el programa Motive. Este fue el encargado de poder interpretar la información capturada por las cámaras y de guardar los diferentes archivos del sistema de captura de movimiento, entre los cuales se encontraban los archivos de calibración, los cuales serán explicados en la subsección 7.1.3. El software fue descargado desde la página oficial de OptiTrack y para poder utilizarlo fue necesario conectar a la computadora la llave de hardware provista en la compra, ya que esta última habilitaba la licencia del software. En la Figura 16 se presenta la pantalla inicial de Motive al abrir el programa con el switch y la llave de hardware conectados.

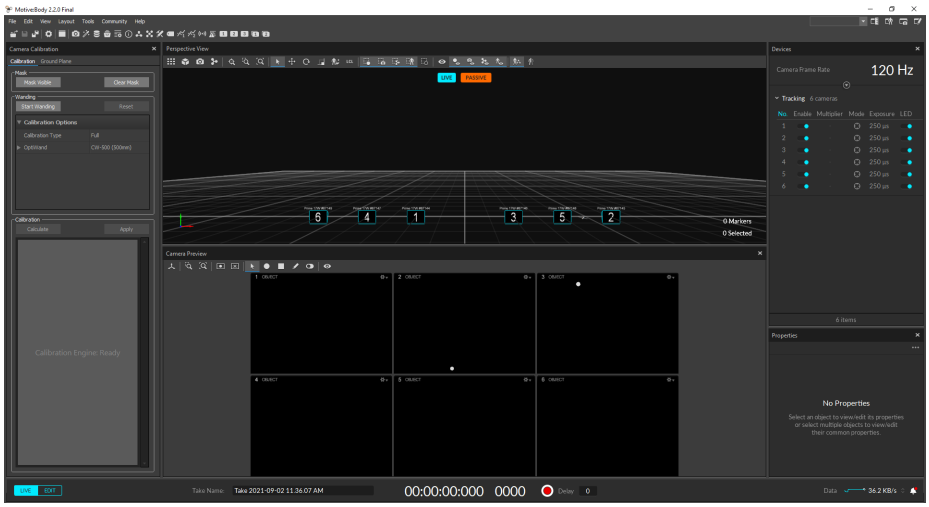


Figura 16: Pantalla de inicio en Motive.

Por otra parte, el kit de desarrollo de software NatNet fue utilizado con el propósito de usar la información provista por el OptiTrack en otra aplicación, pudiéndolo unir así con la red de comunicación WiFi desarrollada. Al descargar este kit se obtuvieron diferentes ejemplos de uso para diferentes lenguajes de programación, tal como MATLAB y Python. Se decidió utilizar el kit de desarrollo para Python, en vez de MATLAB, debido a que este último no permitía realizar una aplicación a tiempo real, además de restringir el broker que se podía utilizar para el sistema de comunicación.

La arquitectura de NatNet funciona con el concepto de servidor y cliente. En este caso, el servidor fue el programa Motive, mientras que el cliente fue la aplicación de Python desarrollada. Como se puede observar en la Figura 17, el programa de Python crea un cliente que se conecta al servidor y por medio de las funciones definidas, determina qué hacer con la información obtenida. El paquete de información obtenido contiene el identificador único del agente, el cual es creado automáticamente en Motive, un vector de tamaño 3 que contiene la información de la posición del agente en metros, y un vector de tamaño 4 que contiene la información de la orientación del agente en el formato de cuaterniones unitarios.

```python
# Uses the Python NatNetClient.py library to establish a connection
# (by creating a NatNetClient), and receive data via a NatNet connection
#and decode it using the NatNetClient library.
from NatNetClient import NatNetClient

# This is a callback function that gets connected to the NatNet client and
#called once per mocap frame.
def receiveNewFrame( frameNumber, markerSetCount, unlabeledMarkersCount,
                     rigidBodyCount, skeletonCount, labeledMarkerCount,
                     timecode, timecodeSub, timestamp, isRecording,
                     trackedModelsChanged ):
    print( "Received frame", frameNumber )

# This is a callback function that gets connected to the NatNet client. It is
#called once per rigid body per frame
def receiveRigidBodyFrame( id, position, rotation ):
    print( "Received frame for rigid body", id )
    print( "Received position for rigid body", position )
    print( "Received rotation for rigid body", rotation )

# This will create a new NatNet client
streamingClient = NatNetClient()

# Configure the streaming client to call our rigid body handler on the
#emulator to send data out.
streamingClient.newFrameListener = receiveNewFrame
streamingClient.rigidBodyListener = receiveRigidBodyFrame

# Start up the streaming client now that the callbacks are set up.
# This will run perpetually, and operate on a separate thread.
streamingClient.run()
```

Figura 17: Código de ejemplo del uso de NatNet para Python.

7.1.3. Calibración del sistema

Al ya tener ensamblados y configurados los componentes de hardware y software del sistema, se procedió a realizar la calibración de este. Este es un proceso fundamental para la obtención de medidas fidedignas en la pose de los agentes, ya que la calibración permite calcular la posición y orientación de cada una de las cámaras, tomar en cuenta la distorsión de la imágenes capturadas y, finalmente, construir el volumen de captura 3D en el programa Motive.

Primeramente, se debió realizar un proceso de enmascarado el cual permitió eliminar cualquier reflejo externo que pudiera afectar la calibración. Posteriormente, se realizó el proceso de *wanding* usando el instrumento presentado en la Figura 18, el cual consiste en mover este instrumento varias veces en frente de las cámaras. Esto permitió a las cámaras capturar marcos de muestreo que les permitiera calcular su respectiva posición y orientación en el espacio 3D. Cabe resaltar que se siguieron las siguientes recomendaciones durante el proceso de *wanding*:

- Se evitó mover muy rápido el instrumento mostrado en la Figura 18, ya que si esto se hacía, se corría el riesgo de generar muestras malas.
- Se evitó utilizar ropa o accesorios reflectivos.
- No se tomaron más de 10 000 muestras, ya que si se tomaban más, esto afectaría negativamente el resultado de la calibración.
- Se utilizó el LED indicador de la cámara para verificar si se cubrió todo el espacio visto por cada cámara.

Para comenzar estos dos procesos se presionó el icono indicado en la barra superior del programa, como se presenta en la Figura 19, y se procedió a presionar el botón de *Start Wanding*.

Figura 18: Instrumento usado para el proceso de *wanding*.

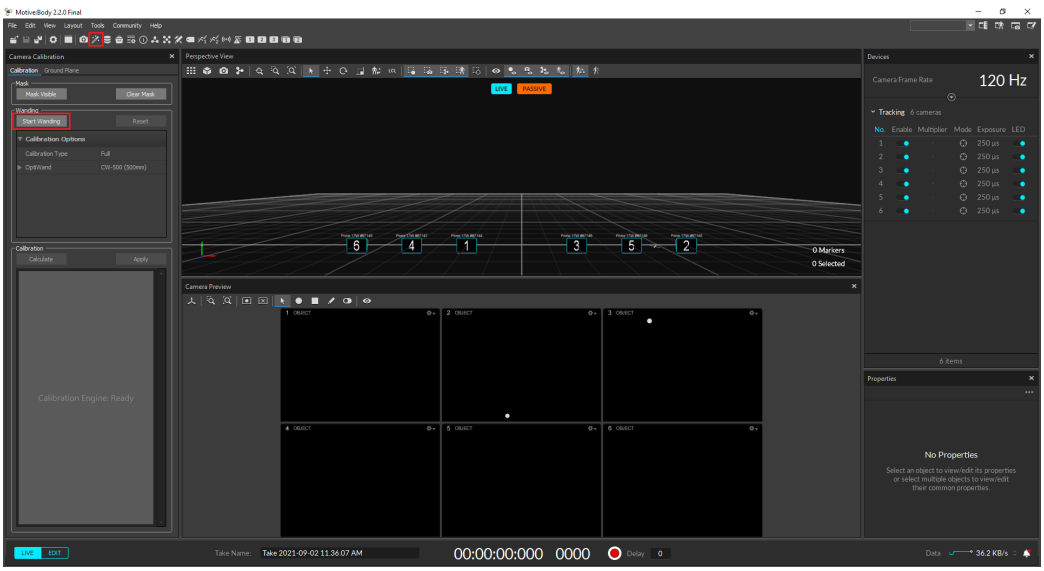


Figura 19: Pantalla para inicialización del proceso de *wanding*.

El resultado de realizar por un tiempo de 5 minutos el proceso de *wanding* se puede observar en la Figura 20. Posteriormente, para continuar el proceso de calibración se presionó el botón de *Calculate*, mostrado también en la misma figura. Esta acción abrió una nueva pantalla en la cual se presentó el resultado y la calidad de la calibración. Como se puede

observar en la Figura 21, el resultado obtenido en la calibración fue “excepcional”, por lo que se considera que el proceso de calibración fue realizado de manera correcta.

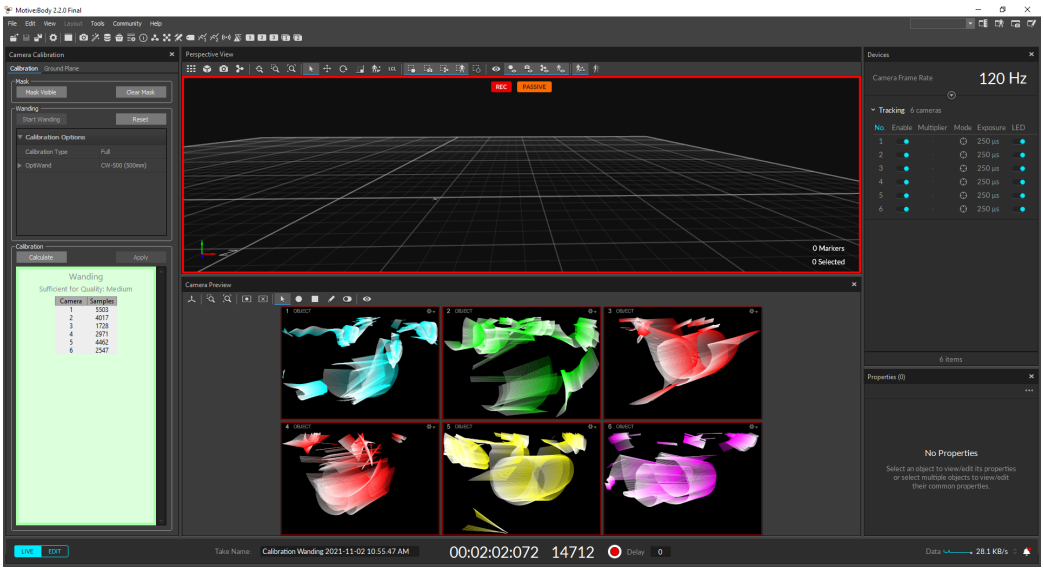


Figura 20: Resultado obtenido al realizar el proceso de *wanding*.

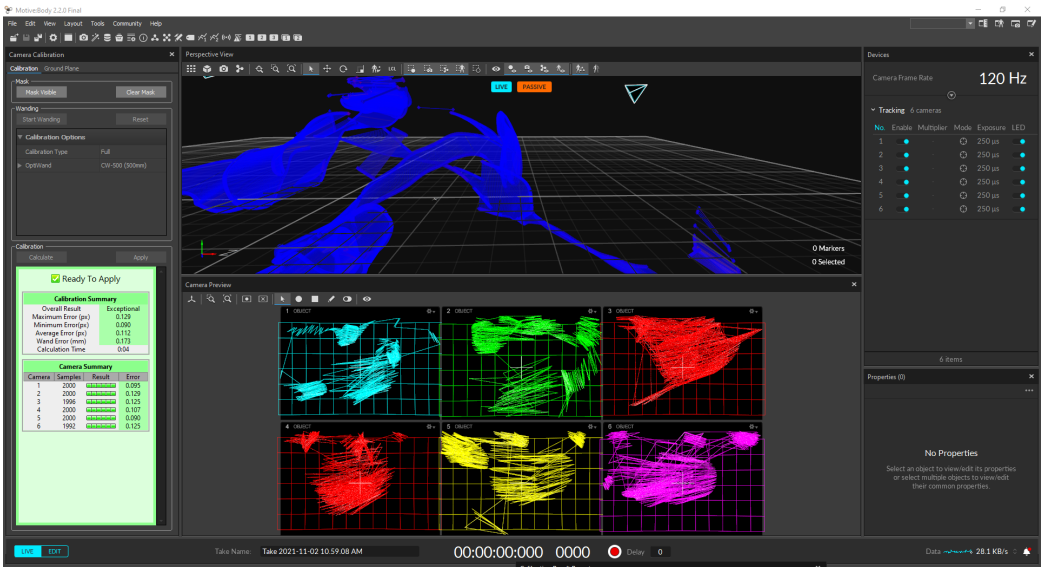


Figura 21: Resultado del proceso de calibración.

Al aplicar a Motive el resultado de calibración obtenido, este abrió una nueva pantalla en la cual se presentaron las opciones a usar para el proceso de definición del plano del suelo, el cual se usó para definir el origen del sistema de captura de movimiento. Este proceso consistió en posicionar el instrumento presentado en la Figura 22 en el suelo del ecosistema y presionar el botón de *Set Ground Plane*, mostrado en la Figura 23. Al presionar este botón, se generó una ventana emergente en donde se procedió a guardar el archivo de calibración, el cual es de extensión .CAL. Este tipo de archivos guarda el resultado de calibración obtenido, evitando así que se deba hacer una nueva calibración cada vez que se desee usar el sistema de captura de movimiento. El resultado final de la calibración del sistema se presenta en la Figura 24, en el cual se puede observar que el posicionamiento de las cámaras se asemeja a la configuración planificada. Cabe resaltar que en el ecosistema, el eje y, el cual es el eje verde presentado en la Figura 24, es perpendicular a la mesa de trabajo mientras que los otros dos ejes son paralelos a esta mesa y su dirección dependerá de la dirección que tomen los ejes de la escuadra, mostrada en la Figura 22, siendo la barra más larga el eje z y la barra más corta el eje x.

Figura 22: Escuadra usada para definir el plano de suelo.

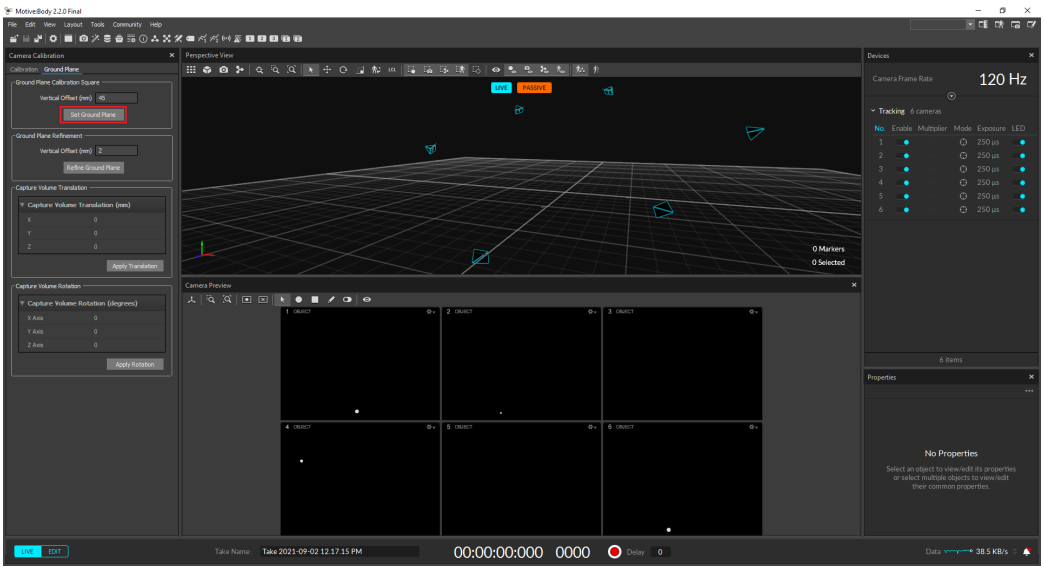


Figura 23: Definición del plano del suelo.

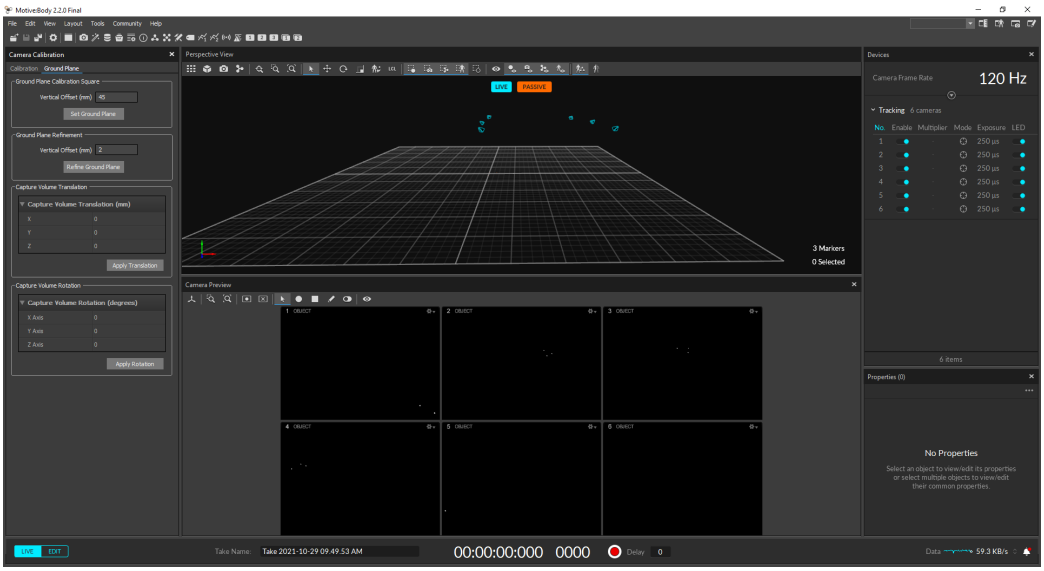


Figura 24: Resultado final de la calibración del sistema.

7.2. Resultados

Para probar el funcionamiento del sistema montado, se realizó la creación de un cuerpo rígido y se revisaron los valores de la pose mostrados. Primeramente, se ingresó un cuerpo rígido al ecosistema utilizando marcadores pasivos y los soportes presentados en la Figura 25. Cabe resaltar que cada cuerpo rígido que ingrese al ecosistema debe tener un posicionamiento único de marcadores, ya que por medio de esto el sistema determina que los cuerpos rígidos son diferentes. Si se utilizaran solo los soportes provistos para realizar el posicionamiento de los marcadores, se podría tener en total a 20 agentes al mismo tiempo dentro del ecosistema.

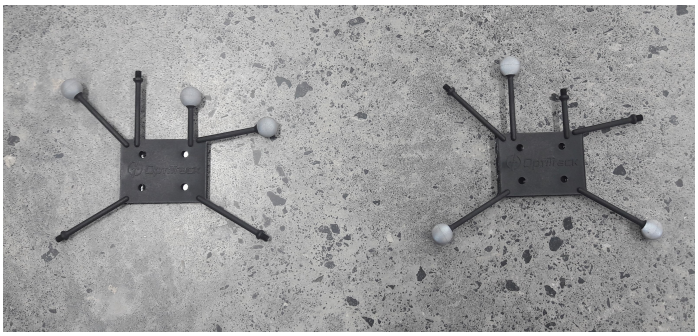


Figura 25: Cuerpo rígidos usados en la prueba del ecosistema.

Al ya haber un cuerpo rígido dentro del ecosistema, se ingresó a la ventana de creación de objeto, oprimiendo el botón en la barra superior presentada en la Figura 26. Posteriormente se escogieron los marcadores presentes, dejando oprimido el click derecho del *mouse* y arrastrandolo para cubirir el área en donde se encontraban estos, para que así el cuerpo rígido fuera reconocido en Motive. Finalmente, se oprimió el botón de *Create*, presentado

también en la Figura 26, para generar así el cuerpo rígido.

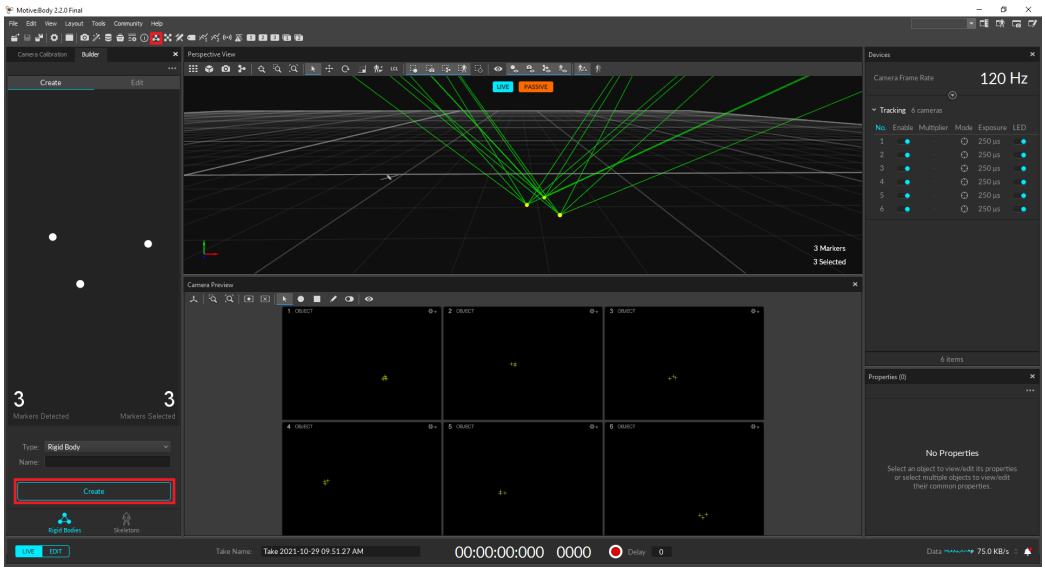


Figura 26: Creación de un cuerpo rígido.

Finalmente, para visualizar los valores de la pose del cuerpo rígido creado, se oprimió el botón en la barra superior presentado en la Figura 27. Esta nueva ventana generada presentó información acerca de la posición del agente además de la orientación de este, como también se puede observar en la misma figura.

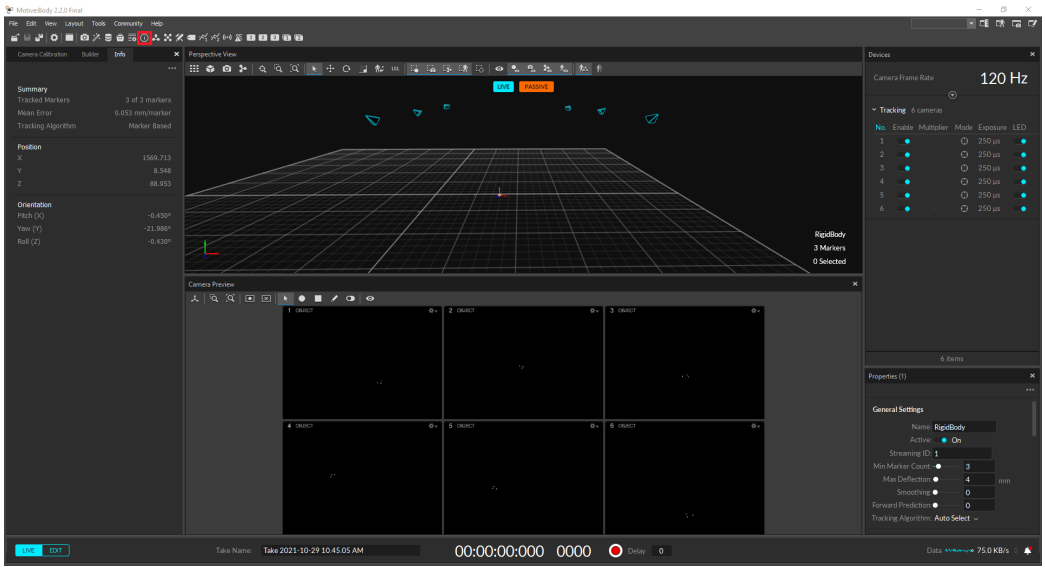


Figura 27: Pose del cuerpo rígido dentro del ecosistema.

Ya teniendo un cuerpo rígido dentro del ecosistema se procedió a ponerlo en diferentes posiciones y se comparó la posición dada por el sistema OptiTrack con las mediciones reales, las cuales fueron medidas con un metro. Se realizó la comparación en cinco posiciones y los resultados obtenidos se presentan en el Cuadro 1.

Corrida	**Posición**	**Posición OptiTrack (mm)**	**Incert. OptiTrack (mm)**	**Posición Real (mm)**	**Incert. metro (mm)**	**Error (%)**
1	x	1569.713	0.435	1570.00	0.50	0.02
	y	8.548		8.90		3.96
	z	88.953		89.00		0.05
2	x	2162.821	0.435	2163.00	0.50	0.01
	y	6.317		6.00		5.28
	z	857.54		857.20		0.04
3	x	1582.759	0.435	1583.00	0.50	0.02
	y	8.110		8.30		2.29
	z	-1090.26		-1090.00		0.02
4	x	140.141	0.435	140.00	0.50	0.10
	y	6.952		7.20		3.44
	z	-14.140		-14.00		0.97
5	x	3160.634	0.435	3161.00	0.50	0.01
	y	9.017		8.70		3.64
	z	-1747.450		-1747.20		0.01

Cuadro 1: Comparación de mediciones reales y mediciones dadas por el sistema OptiTrack.

Como se puede observar en el Cuadro 1, el porcentaje de error más alto fue de 5.28 %. Debido a que la métrica de evaluación de resultados definía que los resultados no debían variar en un valor mayor al 10 %, se aceptaron los resultados como válidos. Consecuentemente, se concluye que el sistema se ensambló y calibró de manera correcta, permitiendo que las siguientes fases del proyecto puedan obtener información fidedigna acerca de la pose de cualquier agente que se encuentre en el ecosistema.

CAPÍTULO 8

Red de comunicación del ecosistema Robotat

La red de comunicación fue el siguiente paso en la creación del ecosistema Robotat. La correcta implementación de esta red permitió que la información obtenida del sistema de captura de movimiento pudiera ser trasmitida a los diferentes agentes y que estos pudieran también trasmitir información relevante a la experimentación. Consecuentemente, el objetivo principal de este capítulo fue diseñar e implementar una red de comunicación WiFi que permitiera el intercambio bi-direccional de información entre la computadora dedicada al uso del sistema de captura de movimiento y los agentes.

8.1. Metodología

En esta sección se describen los pasos realizados en la instalación del hardware necesario para la red, así como también para el desarrollo de las librerías usadas tanto en el microcontrolador, en este caso un ESP32, como en la computadora encargada de recolectar los datos del sistema de captura de movimiento.

8.1.1. Instalación del hardware

Primeramente fue necesario instalar y configurar un dispositivo que actuara como un punto de acceso y creara así la red WiFi en la que se conectarían la computadora y los agentes. En este caso se utilizó un router para generar la red. Cabe resaltar que este router fue adquirido previo al comienzo de este proyecto de graduación, por lo que no se realizó un trade study para determinar la mejor opción para este tipo de aplicación.

El router utilizado fue el NETGEAR MBR624GU, y este se puede observar en la Figura 28. Este router opera en la banda de frecuencia de 2.4 GHz y utiliza los algoritmos de encriptación WPA-PSK y WPA2-PSK. Posteriormente a ser instalado, este fue configurado para que tuviera un nombre y contraseña diferente a la provista por defecto. El nombre fue cambiado a *WiFi_Robotat* y su contraseña se actualizó a *R0bot4tUVG*, como se puede observar en la Figura 29. Es pertinente resaltar que el router utilizaba por defecto el algoritmo de encriptación de red WPA-PSK. Sin embargo, como se presentó en los antecedentes, este algoritmo está obsoleto y es poco seguro, por lo que se cambió al algoritmo WPA2-PSK.

Figura 28: Router utilizado para generar la red WiFi en el ecosistema.

Figura 29: Configuración del nombre y contraseña del router.

Por otra parte, también era necesario que la computadora encargada de interactuar con el OptiTrack estuviera conectada a la red WiFi. Sin embargo, al ser una computadora de escritorio esta no tenía la capacidad de conectarse a esta red. Consecuentemente, fue necesario instalarle un adaptador WiFi a esta para que así estuviera en la misma red que los agentes. Cabe resaltar que no se recomienda desinstalar los drivers y programas que utiliza este adaptador ya que se corre el riesgo a que se modifiqué la dirección IP del broker y por ende sea necesario modificar la librería usada en los programas tanto del ESP32 como de la computadora. La dirección IP a utilizar en la configuración del broker debe ser la misma que toma la computadora al conectarse a la red WiFi.

8.1.2. Desarrollo del programa de integración con el OptiTrack

El desarrollo de un programa externo que interactúe con la información proporcionada por el OptiTrack surge de la necesidad de que los agentes dentro del ecosistema puedan tener esta información para así poder utilizarla en alguna experimentación planeada. Consecuentemente, esta subsección se enfoca en el desarrollo de dicho programa.

Primeramente, se debía determinar el tipo de topología por defecto. Se consideró que la topología ideal, la cual se podía utilizar como base para posteriormente desarrollar experimentaciones más complejas, sería la topología de bus. En esta topología solo existe un tópico en el cual el publicador, en este caso la computadora, manda la información de la pose a todos los agentes dentro del ecosistema. Esta topología puede ser observada en la Figura 30. Cabe resaltar que a pesar de que todos los agentes reciben la información de todos, por defecto los agentes solo se preocupan de su propia información, como se explicará en la subsección 8.1.3.

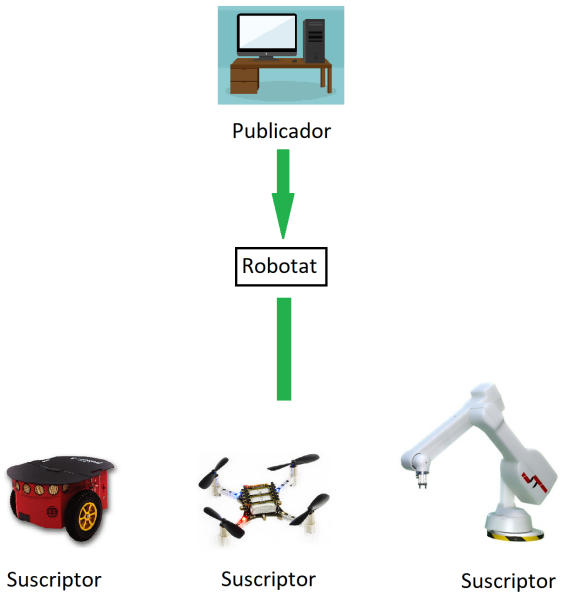


Figura 30: Topología por defecto usada en el ecosistema Robotat.

Posteriormente, se debió definir el tipo de información que la aplicación debía de enviar. En este caso, ya que este ecosistema debía interactuar con varios proyectos que se desarrollaron en conjunto, se definió que la información relevante a enviar sería:

1. Número de datos a enviar.
2. Bandera de uso del identificador.
3. Identificador.
4. Posición.
5. Orientación.
6. Número de puntos en la trayectoria.
7. Modo de operación.
8. Velocidad de operación.
9. Tipo de modo de ruta.
10. Estado de la pinza.
11. Configuración.
12. Posición del efector final.
13. Puntos de la trayectoria.

Cabe resaltar que la información provista desde los datos 6 en adelante son exclusivas para el uso del manipulador serial R17, en concordancia a lo requerido por el proyecto realizado por José Pellecer. Debido a que se utilizaron dos lenguajes de programación distintos para los agentes y la aplicación, se decidió enviar estos datos en formato de caracteres (*string*), y ya en la librería del ESP32 crear una función que decodificara ese mensaje.

El siguiente paso se enfocó en la configuración de todos los componentes de software necesarios para poder implementar la aplicación. Fue necesario descargar la versión más actualizada de Eclipse Mosquitto, la cual serviría para inicializar el broker. Sin embrago, se realizaron modificaciones al archivo de configuración de Mosquitto, para que la inicialización del broker se realizara de la manera correcta. En la Figura 31 se muestran las dos lineas de código añadidas. La primera habilita el broker en un puerto y dirección IP especifica, mientras que la segunda permite que clientes externos a la computadora se puedan conectar a ese broker. Finalmente para inicializar el broker se debió ejecutar el comando presentado en la Figura 32, en la carpeta donde se instaló Eclipse Mosquitto. El conjunto del primer y segundo comando inicializaban el broker con el archivo de configuración modificado, mientras que el tercer comando, permitía visualizar con más detalle los eventos que se realizaban en el broker.

```
# Config file for mosquitto
#
# See mosquitto.conf(5) for more information.
#
# Default values are shown, uncomment to change.
#
# Use the # character to indicate a comment, but only if it is the
# very first character on the line.

# =================================================================
# General configuration
# =================================================================

listener 1883 192.168.0.2
allow_anonymous true
```

Figura 31: Código añadido a la configuración de Eclipse Mosquitto.

```
C:\Program Files\mosquitto>mosquitto -c mosquitto.conf -v
```

Figura 32: Código para la inicialización.

El desarrollo de la aplicación se realizó en el ambiente de Anaconda, ya que se consideró que en este, la instalación y actualización de librearías de Python se realizaba de una manera fácil y eficiente. La única librería necesaria para implementar el protocolo MQTT en esta aplicación fue *paho mqtt*, la cual se instaló en Anaconda por medio del comando "conda install -c conda-forge paho-mqtt".

Ya teniendo todo el software previo instalado se prosiguió a desarrollar la aplicación. Esta tenía como objetivo poder tomar los datos provistos por el OptiTrack y por medio del protocolo MQTT, enviarlos a cada uno de los agentes dentro del ecosistema. Por lo tanto, se tomó como base el código presentado en la Figura 17, ya que este ya obtiene en Python los valores de las poses de los agentes por lo que lo único necesario a añadir fue la red comunicación por medio del protocolo MQTT. En este código se añadió la conexión de la aplicación como cliente del broker y otras utilidades que sirvieron para tener un mayor control sobre la experimentación. Una de estas utilidades consistió en un diccionario de Python, en el cual estaban guardados todos los *ids* usados durante una experimentación. Esto hacía que cuando los agentes recibieran la información de las poses, no pudieran dos agentes utilizar el mismo identificador. Finalmente, la publicación de los datos al tópico general se realizó cada vez que un nuevo *frame* de información llegara a la aplicación, proveniente de OptiTrack.

8.1.3. Desarrollo de la librería para la red de comunicación

La librería desarrollada para el ESP32 fue realizada en el lenguaje de programación C, utilizando también el SDK ESP-IDF. Este último provee diferentes funcionalidades las cuales permiten habilitar diferentes módulos del microcontrolador, para poder así implementarlos en diferentes aplicaciones.

Primeramente se debió determinar de que manera el ESP32 se podía conectar a un broker por medio del protocolo MQTT. Posteriormente, se debió encontrar la manera de decodificar los datos obtenidos a través de este protocolo para así poder guardarlos en una variable que

pudiera ser utilizada en cualquier aplicación. Finalmente, se debió determinar en qué puntos del encendido se debían hacer las suscripciones y/o publicaciones a los diferentes tópicos que estuvieren operativos durante la experimentación.

El ESP32, como cualquier otro dispositivo, debe estar conectado a la misma red que el broker para así poder ser un cliente de este. Por lo tanto, fue necesario que el microcontrolador se conectara primeramente a la red WiFi para que así después de una correcta vinculación, se conectara al broker. Por otra parte, la carga útil enviada desde el broker, llega en formato de caracteres (*string*), por lo que fue necesario determinar una manera de separar los datos y convertirlos a un formato útil el cual fue un vector de tamaño variable en donde se sabía a priori en que posiciones se encontraban los diferentes datos enviados. Finalmente, se determinó que las suscripciones se debían realizar al momento de que el microcontrolador se conectara de manera exitosa al broker, para evitar problemas inesperados mientras que las publicaciones se podrían realizar en el *loop* principal del código.

La función encargada de la separación de los datos de la carga útil no sólo se ocupa de lo mencionado anteriormente, sino que también realiza diferentes utilidades importantes. Primeramente, esta función se encarga de determinar si el *id* recibido en la carga útil no esta en uso. Si este es el caso, el ESP32 manda un mensaje a la computadora en el cual se le indica que a partir de ese momento el microcontrolador estará recibiendo la información enviada con ese *id* y por lo tanto, ningún otro agente debería tener asignado dicho identificador. Por otra parte, esta función determina si el *id* recibido en la carga útil corresponde a el identificador guardado en el paso anterior. Si este es el caso, la función procede a realizar la separación de datos, de lo contrario, la omite. Finalmente, la función también guarda en la memoria flash el id escogido, por lo que este valor no va a ser susceptible a reseteos o a desconexiones de la alimentación. Cabe resaltar que la única manera de borrar el identificador seleccionado es por medio del uso de la herramienta *esptool*. Esta se instala desde la ventana de comandos y el código utilizado para borrar la memoria flash se puede observar en la Figura 33, en donde el valor de X debe ser reemplazado por el número de puerto en donde se encuentre conectado el ESP32. Estas funcionalidades extra hacen que la librería sea más robusta y eficiente, ya que toma medidas defensivas y solo realiza las operaciones necesarias para asegurar así un correcto funcionamiento de la red.

```
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>esptool.py --chip esp32 --port COMX erase_flash
```

Figura 33: Comando utilizado para borrar la memoria flash.

En la siguiente tabla se presenta la librería realizada, incluyendo todas las funciones que se desarrollaron, los parámetros de estas y una breve explicación de su uso.

Librería Robotat

Función	Parámetros	Descripción
robotat_connect	void [1]	Configura e inicializa la conexiones WiFi y MQTT con los parámetros indicados en los macros.
robotat_publish	Tópico al cual publicar Mensaje a enviar	Publica un mensaje a través de MQTT al tópico indicado.
robotat_get_data	void	Obtiene el *array* de datos enviados por medio de MQTT para que este puede ser utilizado en otras partes del código.
robotat_get_status	void	Regresa el estado en el que se encuentra el agente con respecto a la red de comunicación.

Cuadro 2: Estados de los agentes en la red de comunicación.

Por otra parte, en la siguiente tabla se presentan los estados en los que se puede encontrar el agente, además de una breve explicación de estos.

Estados Robotat

Estado	Descripción
ROBOTAT_CONNECTING_MQTT	El dispositivo se está conectando a la red del Robotat (específicamente al broker MQTT).
ROBOTAT_CONNECTION_ERROR_WIFI	El dispositivo no se pudo conectar a la red del Robotat (específicamente a la red WiFi).

Continúa en la siguiente página

[1] El término *void* tiene como significado la no necesidad de un parámetro

Cuadro 3 – *Continuación de la página previa*

Estado	Descripción
ROBOTAT_CONNECTION_ERROR_MQTT	El dispositivo no se pudo conectar a la red del Robotat (específicamente al broker MQTT).
ROBOTAT_CONNECTION_SUCCESS_WIFI	El dispositivo se conectó correctamente a la red del Robotat (específicamente a la red WiFi).
ROBOTAT_CONNECTION_SUCCESS_MQTT	El dispositivo se conectó correctamente a la red del Robotat (específicamente al broker MQTT).
ROBOTAT_DISCONNECTED_MQTT	El dispositivo se desconectó de la red del Robotat (específicamente al broker MQTT).
ROBOTAT_SUBSCRIBED_TO_TOPIC	El dispositivo se conectó a un tópico de la red del Robotat.
ROBOTAT_MESSAGE_PUBLISHED	El dispositivo publicó un mensaje a un tópico en la red del Robotat.
ROBOTAT_DATA_RECEIVED	El dispositivo recibió un mensaje enviado de un tópico en la red del Robotat.

Cuadro 3: Funciones desarrolladas para la librería Robotat.

8.2. Resultados

Para probar el correcto funcionamiento de la librería y programa desarrollado, se desarrollaron dos pruebas: una prueba con solo un agente conectado y otra con dos agentes conectados, lo cual serviría para determinar si la red podía manejar múltiples agentes. Además, también se desarrolló una prueba de latencia de la red de comunicación, la cual permitiría determinar el número de agentes que podían estar conectados a la red antes de que la frecuencia de recepción y decodificación de datos fuera menor a 10 Hz. Se llegó a esta frecuencia debido a que se quiere que el sistema opere como mínimo a la misma frecuencia en la que un GPS envía datos a un agente.

8.2.1. Verificación del programa y librería desarrollados

Primeramente, se realizaron pruebas de la red de comunicación con un solo agente conectado. Esto se efectuó posicionando un cuerpo rígido con marcadores dentro de la plataforma para crearlo así en *Motive*, como se puede observar en la Figura 34. Posteriormente, se ejecutó el programa desarrollado en Python y se encendió el ESP32, el cual ya estaba utilizando la librería desarrollada.

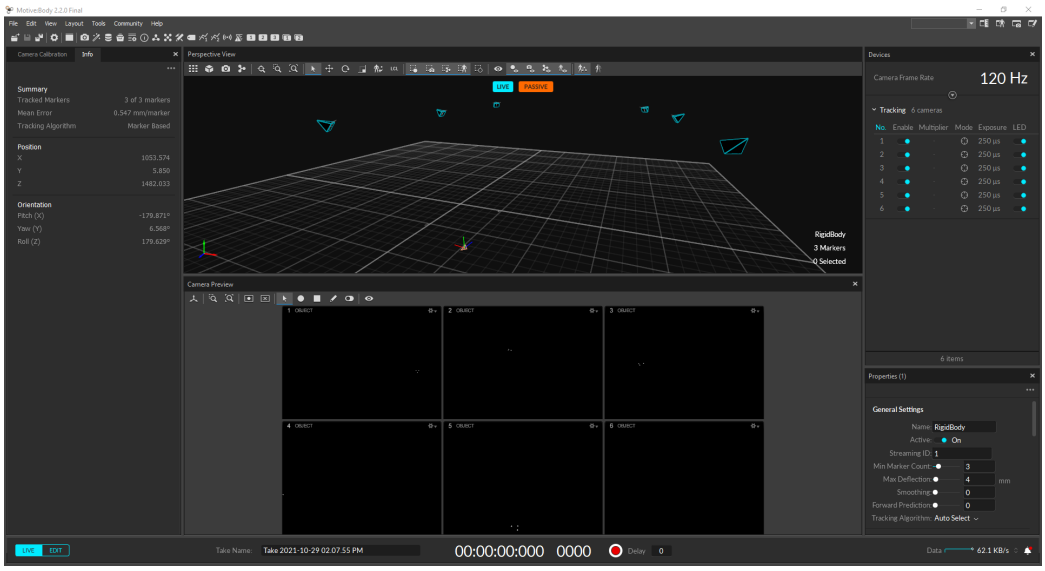


Figura 34: Datos de la pose del agente conectado.

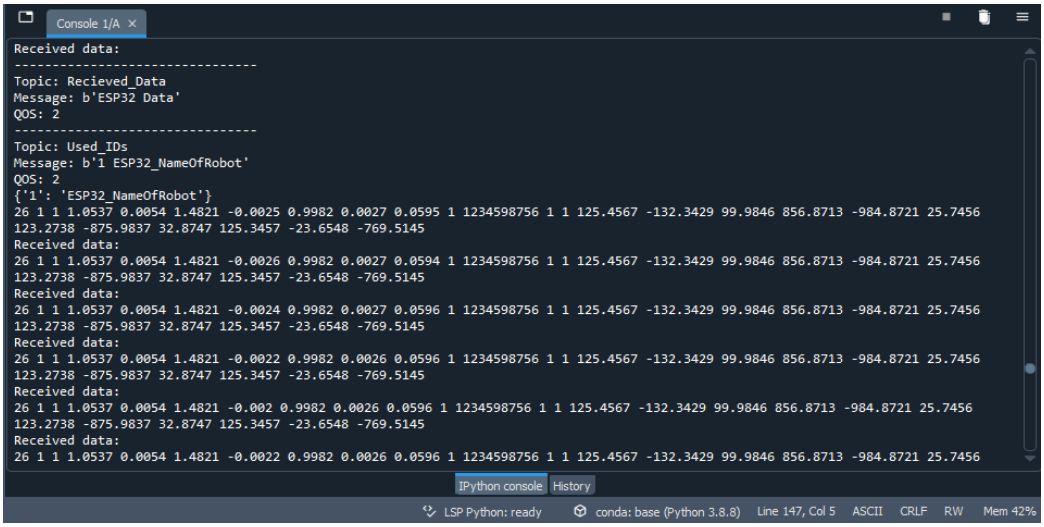


Figura 35: Datos enviados desde el programa de Python hacia los agentes conectados.

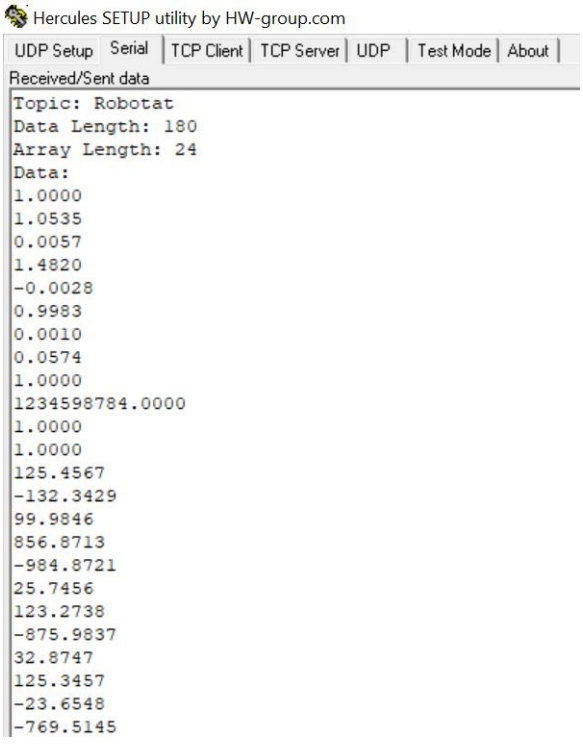


Figura 36: Datos obtenidos en el ESP32, visualizados a través del monitor serial.

Como se puede observar en la Figura 35, el programa desarrollado en Python toma los datos del agente que están siendo trasmitidos desde *Motive*. Cabe resaltar que la orientación del agente en el programa desarrollado en Python está dada en cuaterniones unitarios, con orden de rotación XYZ. Por otra parte, en la Figura 36 se puede observar que el microcontrolador ESP32 se conecta de manera satisfactoria a la red y toma el identificador que esté disponible y consecuentemente, empieza a recibir los datos correspondientes.

Posteriormente, se realizó la misma prueba pero en este caso se conectaron dos agentes, lo cual daría una idea de el funcionamiento de la red al momento de interactuar con multiples agentes.

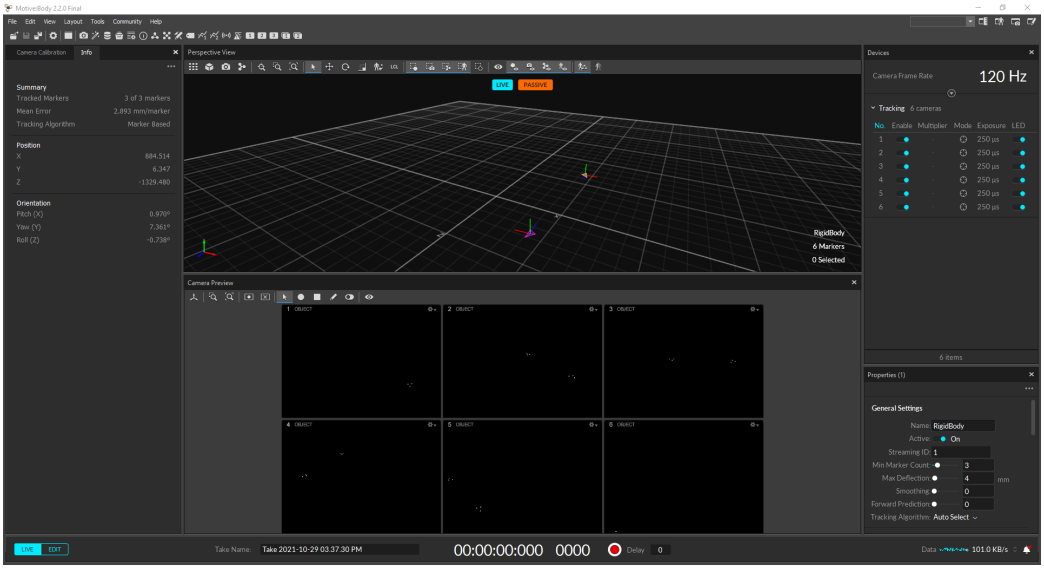


Figura 37: Datos de la pose del agente conectado con identificador 1.

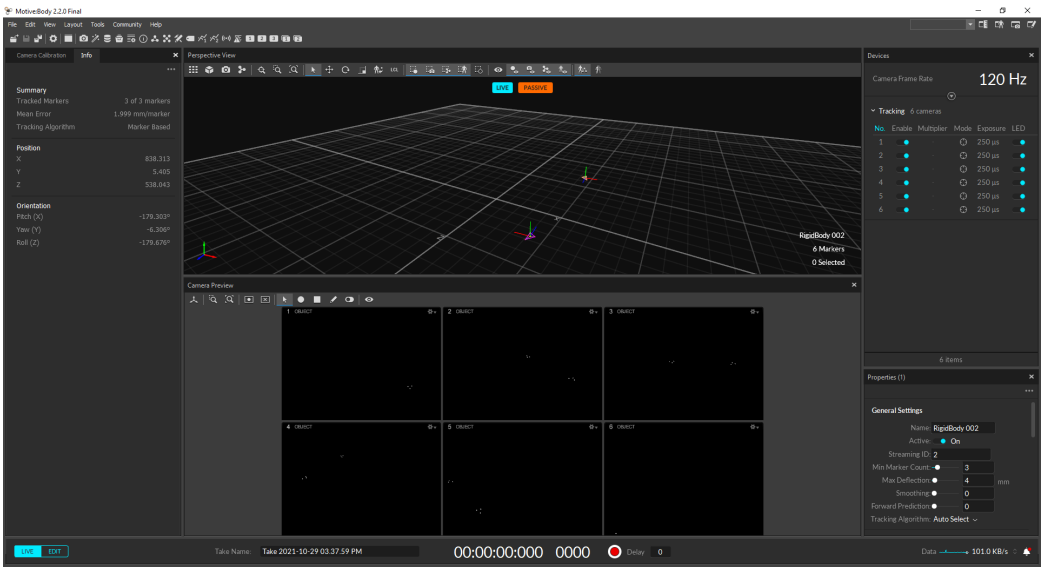


Figura 38: Datos de la pose del agente conectado con identificador 2.

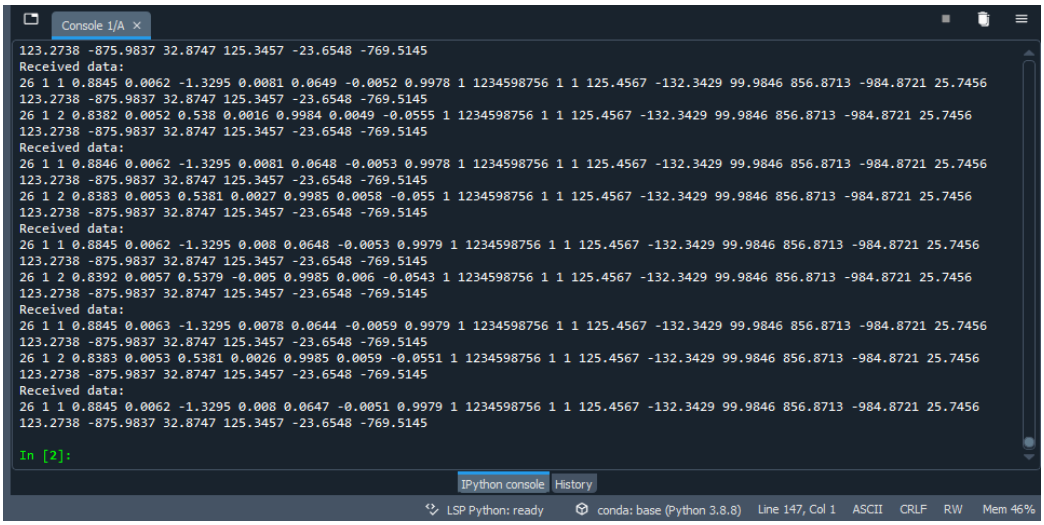


Figura 39: Datos enviados desde el programa de Python hacia los agentes conectados.

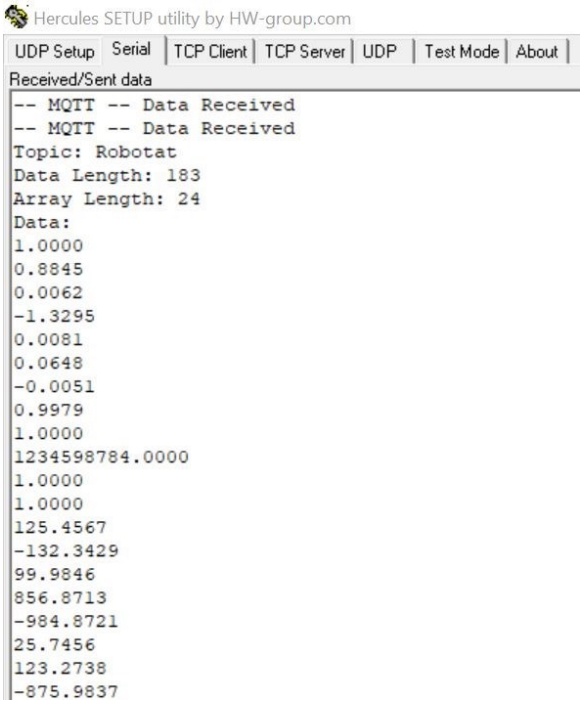


Figura 40: Datos obtenidos en uno de los ESP32, visualizados a través del monitor serial.

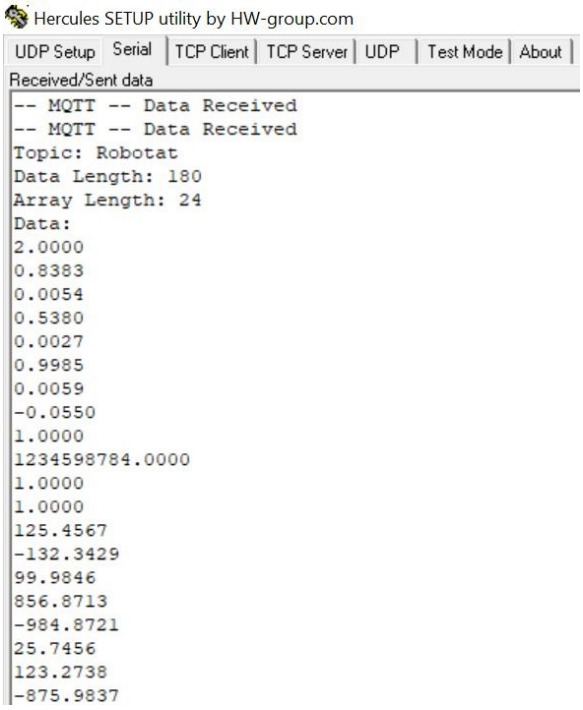


Figura 41: Datos obtenidos en uno de los ESP32, visualizados a través del monitor serial.

Como se puede observar en la Figura 39, ahora se realiza el envío de dos paquetes de información, ya que existen dos agentes creados en el ecosistema. Consecuentemente, se puede visualizar en las Figuras 40 y 41 que cada uno de los ESP32 se conectó a un identificador diferente y empezó a recibir la información de la pose correspondiente.

Estas dos pruebas finalmente indican que la implementación de la red de comunicación se realizó con éxito y esta red puede interactuar con varios agentes al mismo tiempo.

8.2.2. Medición de latencia

La medición de latencia se realizó por medio de la implementación de *timers* en el microcontrolador ESP32, los cuales medían el tiempo que pasaba entre la recepción y decodificación de los paquetes de información, los cuales eran enviados desde el programa desarrollado en Python. El proceso de esta medición consistió en ir conectando más agentes a la red hasta que se pudiera realizar una regresión a los datos obtenidos. Estos datos son presentados en el Cuadro 4 y en la Figura 42.

Número de agentes	**Período de envío de paquetes (μs)**	**Prom. período de envío de paquetes (μs)**	**Frecuencia de recepción (Hz)**
1	10243 10247 10239 10444 10066	0.010	97.582
2	13176 12900 12593 13045 13743	0.013	76.386
3	17128 16893 16825 16914 16847	0.017	59.097
4	22376 23966 22253 22113 22600	0.023	44.128
5	30352 32851 30666 32841 33449	0.032	31.219
6	41842 38804 39735 36538 40506	0.039	25.326
7	40623 38474 43861 41397 40456	0.041	24.413
8	51769 49500 50492 50085 51453	0.051	19.740
9	51082 55469 50979 56799 52442	0.053	18.743
10	57007 59047 57838 61143 58991	0.059	17.005

Cuadro 4: Medición de frecuencia de recepción y decodificación de paquetes de información.

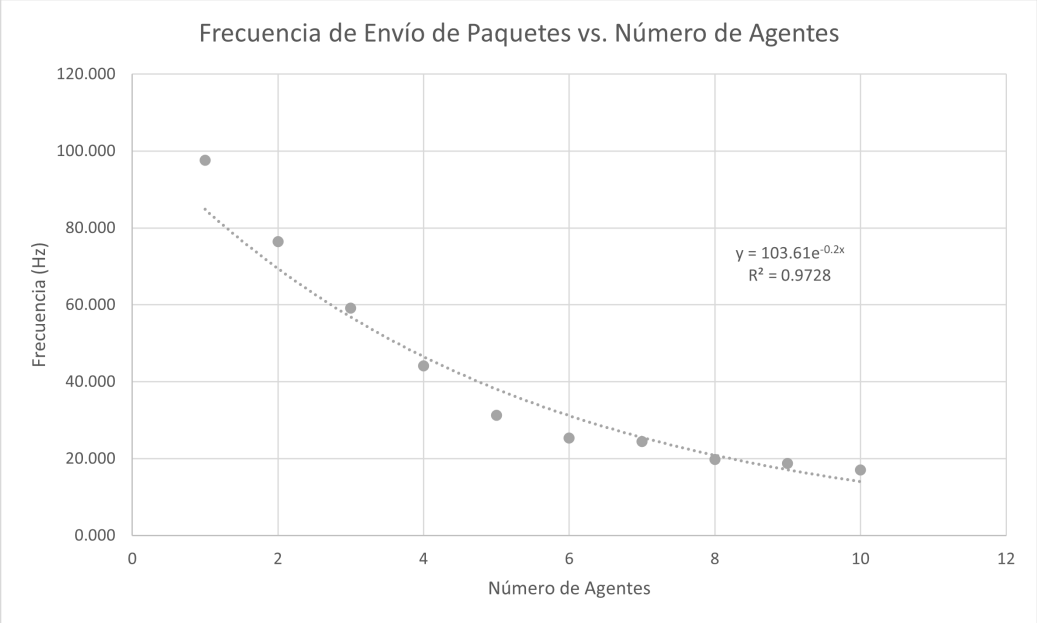


Figura 42: Gráfica de los datos obtenidos del Cuadro 4

Como se puede observar en la Figura 42, al aplicar una regresión exponencial a los datos obtenidos se obtuvo un coeficiente de correlación de 0.9728, indicando que la ecuación exponencial obtenida es una buena aproximación, siendo esta:

$$y = 103.61e^{-0.2x}. \tag{5}$$

Consecuentemente, al despejar 5 para un valor de 10 Hz se obtiene que el número de agentes máximo es de 11 agentes. Estos resultados por lo tanto brindan un mayor entendimiento acerca de las limitantes que tiene la red de comunicación y como la topología escogida tiene un efecto en la rapidez en la que se obtienen y decodifican los paquetes de datos.

CAPÍTULO 9

Desarrollo e implementación de antena inteligente

A pesar de que el ecosistema Robotat ya permitía que diferentes agentes robóticos se pudieran conectar e interactuar entre si, era necesario ampliar esta funcionalidad a agentes no robóticos para así tener una plataforma de experimentación más completa. Consecuentemente, el objetivo principal de este capítulo fue diseñar e implementar una antena inteligente, la cual permitiera a agentes no robóticos interactuar dentro de la plataforma Robotat.

9.1. Metodología

En esta sección se describen los pasos realizados durante el diseño, implementación y verificación de la antena inteligente. Estos incluyen los trade studies realizados para la elección de ciertos componentes clave, el diseño de la placa y el estuche, y finalmente, la definición del filtro de Kalman a utilizar para la fusión de sensores.

9.1.1. Realización de los Trade Studies para la elección de componentes clave

La realización de trade studies permitió determinar las mejores opciones para tres elementos clave de la antena inteligente: la forma de acoplamiento, la batería a utilizar y los soportes de marcadores a utilizar. Por lo tanto, para lograr resultados más certeros, se utilizó la calculadora en línea ofrecida por la página MechaniCalc [25].

Forma de acoplamiento

En este análisis se quería verificar cual era la mejor opción para acoplar la antena a los dispositivos no robóticos. Las dos opciones a analizar fueron:

- Acople por medio de correas.
- Acople por medio de velcro.

El trade study toma en cuenta ciertos aspectos críticos para determinar cual es la mejor opción, por lo que una buena definición de aspectos esenciales permitirá obtener resultados más definitivos. Los criterios utilizados para este análisis se presentan a continuación:

- **Seguridad**: Este criterio describe que tan fuerte el elemento se puede acoplar a la superficie, y si este se mantendrá incluso si hay movimiento.
- **Facilidad**: Este criterio describe que tan fácil se puede acoplar el elemento a la superficie.
- **Adaptación**: Este criterio describe si es necesario adaptar la superficie a la que el elemento se acoplará.
- **Tamaño**: Este criterio describe que tan grande es el elemento utilizado para el acoplamiento.
- **Disponibilidad**: Este criterio describe que tan fácil es encontrar el elemento en el mercado local.
- **Precio**: Este criterio describe el precio del elemento.

Posteriormente, a cada uno de estos criterios se le debe asignar un grado de importancia. Esto permite determinar que tanto afecta un criterio a la ponderación general. La calculadora utilizada para realizar este análisis utilizaba diferentes métodos para este análisis, además de calificar la importancia de un criterio en un escala del 1 al 9, siendo 1 la menor importancia y 9 la mayor importancia. La importancia escogida para los criterios se presenta a continuación:

Criterio	**Importancia**	**Ponderación AHP (%)**	**Ponderación por pares (%)**	**Ponderación lineal (%)**
Seguridad	9	38.14	37.94	22.50
Facilidad	6	9.66	9.69	15.00
Adaptación	7	15.66	15.69	17.50
Tamaño	8	24.84	24.91	20.00
Disponibilidad	5	5.85	5.88	12.50
Precio	5	5.85	5.88	12.50

Cuadro 5: Pesos asignados a los criterios.

Como se puede observar en el Cuadro 5, el criterio de mayor importancia fue la seguridad, seguido del tamaño y la adaptación. Por otra parte, los criterios menos significativos fueron la disponibilidad y el precio.

Cada uno de los criterios fueron evaluados por medio de dos tasas de rendimiento: un rendimiento cualitativo y un rendimiento cuantitativo. La primer tasa de rendimiento describe a una evaluación subjetiva del rendimiento de cierto criterio, siendo un valor de 1 muy deficiente y un valor de 5 excelente. Por otra parte, la segunda tasa de rendimiento toma en cuenta valores numéricos en donde el valor más bajo es el mejor. Consecuentemente, las tasas de rendimiento de los criterios se presenta a continuación:

Alternativa	**Valor**	**Norma**
Velcro	3	0.429
Correas	4	0.571

Cuadro 6: Rendimiento de cada alternativa para el criterio de seguridad (Tasa de rendimiento - Cualitativa).

Alternativa	**Valor**	**Norma**
Velcro	4	0.500
Correas	4	0.500

Cuadro 7: Rendimiento de cada alternativa para el criterio de facilidad (Tasa de rendimiento - Cualitativa).

Alternativa	**Valor**	**Norma**
Velcro	1	0.167
Correas	5	0.833

Cuadro 8: Rendimiento de cada alternativa para el criterio de adaptación (Tasa de rendimiento - Cualitativa).

Cabe resaltar que para el Cuadro 8 un valor de 5 representa que no necesita adaptación mientras que un valor de 1 representa la necesidad de adaptación.

Alternativa	**Valor**	**Norma**
Velcro	5	0.714
Correas	2	0.286

Cuadro 9: Rendimiento de cada alternativa para el criterio de tamaño (Tasa de rendimiento - Cualitativa).

Cabe resaltar que para el Cuadro 9 un valor de 5 representa que el tamaño del acople es pequeño mientras que un valor de 1 representa que el tamaño del acople es grande.

Alternativa	**Valor**	**Norma**
Velcro	4	0.571
Correas	3	0.429

Cuadro 10: Rendimiento de cada alternativa para el criterio de disponibilidad (Tasa de rendimiento - Cualitativa).

Cabe resaltar que para el Cuadro 10 un valor de 5 representa gran disponibilidad en el mercado local mientras que un valor de 1 representa poca disponibilidad en este mismo mercado.

Alternativa	Valor	Norma
Velcro	30	0.625
Correas	50	0.375

Cuadro 11: Rendimiento de cada alternativa para el criterio de precio (Tasa de rendimiento - Cuantitativa [Valor más bajo mejor]).

Finalmente, los resultados obtenidos se muestran en el Cuadro 12 y la Figura 43. En estos se puede observar que la mejor forma de acople, tomando en cuenta todos los criterios y tasas de rendimiento escogidas, es el uso de **correas**.

Alternativa	Suma de puntación ponderada	Puntuación final
Velcro	0.485	94
Correas	515	100

Cuadro 12: Resultados finales del trade study del acople.

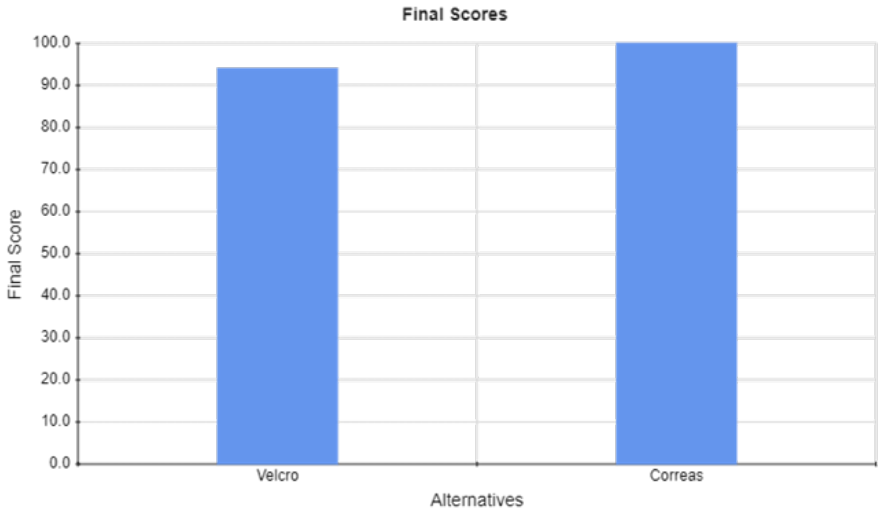


Figura 43: Resultados finales del trade study del acople.

Elección de batería

Primeramente, para poder evaluar de mejor manera los criterios planteados para este trade study, se realizó un *power budget* para determinar que tanta corriente necesitaban todos los componentes de la antena inteligente. Dicho análisis se presenta a continuación:

	Requerimiento de corriente nominal (A)	**Requerimiento de corriente pico (A)**
ESP32	0.08	0.3008
MPU-9250	0.003930	-
TP4056 (Cargador)	0.000154	0.000507
Conversor DC-DC	0.001	0.001
Total (A)		0.306237
Total (mA)		**306.237**

Cuadro 13: Resultados del *power budget* realizado.

Posteriormente, se procedió a realizar el trade study. En este análisis se quería verificar cual era la mejor opción de baterías para proveerle energía a la antena inteligente. Las cuatro opciones a analizar fueron:

- Uso de baterías Li-Ion.
- Uso de baterías LiPo.
- Uso de baterías LiFePO4.
- Uso de baterías Ni-MH.

Los criterios utilizados para este análisis se presentan a continuación:

- **Disponibilidad**: Este criterio describe que tan fácil es conseguir el componente en el mercado local..
- **Voltaje**: Este criterio describe el valor del voltaje de trabajo de las baterías (solo una celda).
- **Tiempo de vida**: Este criterio describe que tantas recargas de batería se pueden realizar antes de que sea necesario reemplazar el componente.
- **Tamaño**: Este criterio describe que tan compacta es la batería.
- **Corriente**: Este criterio describe el valor de corriente que puede suministrar la batería (de los valores disponibles en el mercado).
- **Cargador**: Este criterio describe la disponibilidad de cargadores que puedan ser instalados en la placa.

La importancia escogida para los criterios se presenta a continuación:

Criterio	Importancia	Ponderación AHP (%)	Ponderación por pares (%)	Ponderación lineal (%)
Disponibilidad	7	11.12	11.14	15.91
Voltaje	8	19.76	19.85	18.18
Tiempo de vida	5	4.74	4.69	11.36
Tamaño	9	33.50	33.35	20.45
Corriente	8	19.76	19.85	18.18
Cargador	7	11.12	11.14	15.91

Cuadro 14: Pesos asignados a los criterios.

Como se puede observar en el Cuadro 14, el criterio de mayor importancia fue el tamaño, seguido del voltaje y la corriente. Por otra parte, los criterios menos significativos fueron el de tiempo de vida y disponibilidad.

Las tasas de rendimiento de los criterios se presenta a continuación.

Alternativa	Valor	Norma
Baterías Li-Ion	5	0.357
Baterías LiPo	3	0.214
Baterías LiFePO4	1	0.071
Baterías Ni-MH	5	0.357

Cuadro 15: Rendimiento de cada alternativa para el criterio de disponibilidad (Tasa de rendimiento - Cualitativa).

Alternativa	Valor	Norma
Baterías Li-Ion	5	0.250
Baterías LiPo	5	0.250
Baterías LiFePO4	5	0.250
Baterías Ni-MH	5	0.250

Cuadro 16: Rendimiento de cada alternativa para el criterio de voltaje (Tasa de rendimiento - Cualitativa).

Cabe resaltar que para el Cuadro 16 se busca que los valores de voltaje que provean las baterías estén lo más cercanas a 3.3V o en su defecto 5V.

Alternativa	Valor	Norma
Baterías Li-Ion	5000	0.313
Baterías LiPo	5000	0.313
Baterías LiFePO4	5000	0.313
Baterías Ni-MH	1000	0.063

Cuadro 17: Rendimiento de cada alternativa para el criterio de tiempo de vida (Tasa de rendimiento - Cuantitativa [Valor más alto mejor]).

Cabe resaltar que para el Cuadro 17 el tiempo de vida se mide en número de recargas.

Alternativa	Valor	Norma
Baterías Li-Ion	19	0.225
Baterías LiPo	5	0.423
Baterías LiFePO4	30	0.070
Baterías Ni-MH	15	0.282

Cuadro 18: Rendimiento de cada alternativa para el criterio de tamaño (Tasa de rendimiento - Cuantitativa [Valor más bajo mejor]).

Cabe resaltar que para el Cuadro 18 se define al tamaño como el grosor de la batería (en mm).

Alternativa	Valor	Norma
Baterías Li-Ion	3000	0.160
Baterías LiPo	8000	0.428
Baterías LiFePO4	5000	0.267
Baterías Ni-MH	2700	0.144

Cuadro 19: Rendimiento de cada alternativa para el criterio de corriente (Tasa de rendimiento - Cuantitativa [Valor más alto mejor]).

Cabe resaltar que para el Cuadro 19 se describe el valor máximo de mAh que pueden brindar las baterías disponibles en el mercado.

Alternativa	Valor	Norma
Baterías Li-Ion	5	0.250
Baterías LiPo	5	0.250
Baterías LiFePO4	5	0.250
Baterías Ni-MH	5	0.250

Cuadro 20: Rendimiento de cada alternativa para el criterio de corriente (Tasa de rendimiento - Cualitativa).

Finalmente, los resultados obtenidos se muestran en el Cuadro 21 y la Figura 44. En estos se puede observar que la mejor batería, tomando en cuenta todos los criterios y tasas de rendimiento escogidas, es la **batería LiPo**.

Alternativa	Suma de puntación ponderada	Puntuación final
Baterías Li-Ion	0.239	70
Baterías LiPo	0.342	100
Baterías LiFePO4	0.176	51
Baterías Ni-MH	0.243	71

Cuadro 21: Resultados finales del trade study de las baterías.

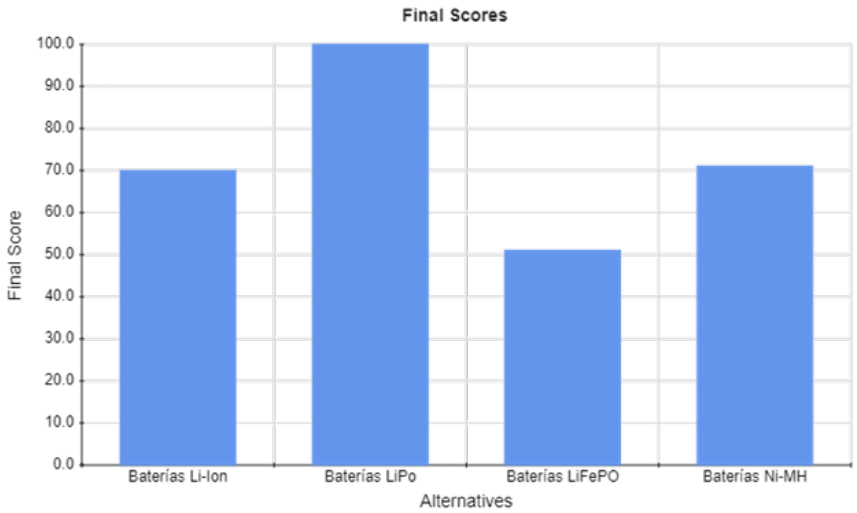


Figura 44: Resultados finales del trade study de las baterías.

Elección de soportes para marcadores

En este análisis se quería verificar cual era la mejor opción para acoplar los marcadores a la antena inteligente. Las tres opciones a analizar fueron:

- Soportes comprados junto con el sistema OptiTrack (OT).
- Descarga de soportes desde la página de Thingiverse (TV).
- Mezcla de diseño propio (OD) en conjunto con piezas encontradas en la página de Thingiverse (TV).

Los criterios utilizados para este análisis se presentan a continuación:

- **Disponibilidad**: Este criterio describe que tan rápido se puede diseñar y manufacturar el soporte.
- **Adaptabilidad**: Este criterio describe que tan fácil es realizar cambios de diseño a la forma de los soportes.
- **Tamaño**: Este criterio describe que tan grande es el diseño de los soportes.
- **Factibilidad**: Este criterio describe que tan factible es la fabricación dado el el tamaño de la rosca de los marcadores (M4).

La importancia escogida para los criterios se presenta a continuación:

Criterio	Importancia	Ponderación AHP (%)	Ponderación por pares (%)	Ponderación lineal (%)
Disponibilidad	7	12.23	12.22	21.88
Adaptabilidad	9	42.36	42.31	28.13
Tamaño	8	22.70	22.74	25.00
Factibilidad	8	22.70	22.74	25.00

Cuadro 22: Pesos asignados a los criterios.

Como se puede observar en el Cuadro 22, el criterio de mayor importancia fue la adaptabilidad, seguido del tamaño y la factibilidad. Por otra parte, el criterio menos significativo fue el de disponibilidad.

Las tasas de rendimiento de los criterios se presenta a continuación.

Alternativa	Valor	Norma
Soportes OT	5	0.417
Thingiverse	4	0.333
TV + OD	3	0.250

Cuadro 23: Rendimiento de cada alternativa para el criterio de disponibilidad (Tasa de rendimiento - Cualitativa).

Alternativa	Valor	Norma
Soportes OT	1	0.111
Thingiverse	3	0.333
TV + OD	5	0.556

Cuadro 24: Rendimiento de cada alternativa para el criterio de adaptabilidad (Tasa de rendimiento - Cualitativa).

Alternativa	Valor	Norma
Soportes OT	3	0.231
Thingiverse	5	0.385
TV + OD	5	0.385

Cuadro 25: Rendimiento de cada alternativa para el criterio de tamaño (Tasa de rendimiento - Cualitativa).

Cabe resaltar que para el Cuadro 25 un valor de 5 describe un tamaño pequeño mientras que un valor de 1 describe un tamaño grande.

Alternativa	Valor	Norma
Soportes OT	5	0.556
Thingiverse	2	0.222
TV + OD	2	0.222

Cuadro 26: Rendimiento de cada alternativa para el criterio de factibilidad (Tasa de rendimiento - Cualitativa).

Cabe resaltar que para el Cuadro 26 un valor de 5 describe que la fabricación no se necesita realizar o no existe ningún inconveniente en esta mientras que un valor de 1 describe que la fabricación no se puede realizar.

Finalmente, los resultados obtenidos se muestran en el Cuadro 27 y la Figura 45. En estos se puede observar que la mejor opción, tomando en cuenta todos los criterios y tasas de rendimiento escogidas, es el uso de **soportes de diseño propio con algunas piezas obtenidas de Thingiverse**.

Alternativa	Suma de puntación ponderada	Puntuación final
Soportes OT	0.277	69
Thingiverse	0.320	79
TV + OD	0.404	100

Cuadro 27: Resultados finales del trade study de soportes.

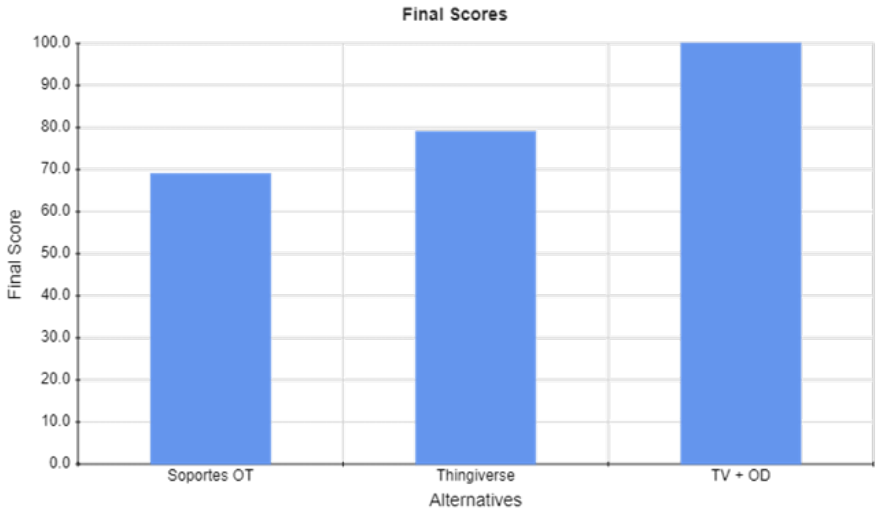


Figura 45: Resultados finales del trade study de soportes.

9.1.2. Diseño de placa

Ya habiendo seleccionado los componentes clave que estarían en la antena inteligente, se procedió a diseñar la placa en la que se posicionarían estos componentes. Debido a que esta debía ser posicionada en otros objetos, se buscó que la placa fuera lo más compacta posible. La placa fue diseñada en el programa *Altium Designer*, fue fabricada en la fresadora LPKF S103 disponible en la universidad y tuvo un tamaño de 75 mm x 75 mm.

Función de componentes de la placa

- **Switch SPDT**: Permite habilitar o deshabilitar la alimentación de la batería a la antena inteligente.
- **TP4056**: Módulo que permite cargar la batería.

- **Conversor DC-DC de 5V**: Convierte el voltaje de la batería (3.7V) a un voltaje que pueda alimentar al microcontrolador ESP32.
- **ESP32**: Es el encargado de conectarse a la red de comunicación y obtener los datos de orientación brindados por la IMU.
- **MPU-9250**: Proporciona los datos de orientación de la antena, haciendo uso de un magnetómetro, giroscopio y acelerómetro.
- **Bornera de 2 pines**: Permite conectar los cables de la batería a la placa.
- **Batería LiPo de 3.7V y 2000 mAh**: Dispositivo que le brinda energía a la antena inteligente.

Esquemático

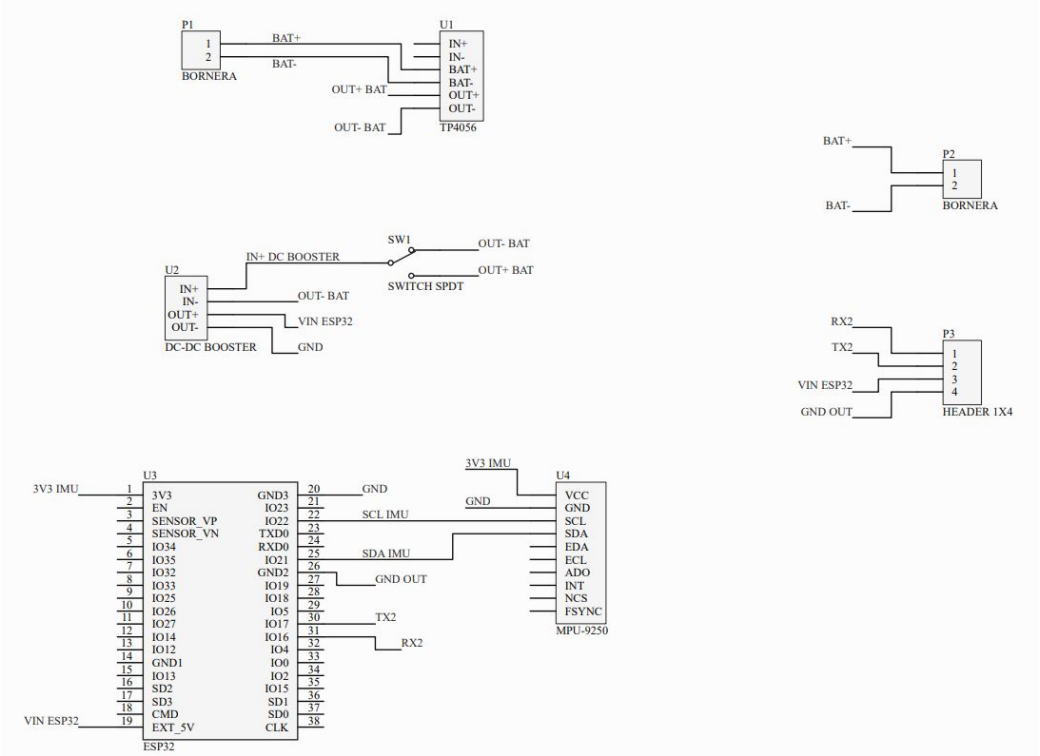


Figura 46: Esquemático de la antena inteligente.

La Figura 46 muestra de forma gráfica las conexiones de todos los elementos de la placa. Este puede ser visualizado de arriba para abajo, donde primeramente, se muestra la conexión de la bornera de la batería con el módulo cargador TP4056. De este último sale las conexiones de voltaje y tierra al conversor DC-DC de 5V y al switch SPDT, el cual determinaba si el voltaje de la batería pasaba a los otros componentes o no. Finalmente, el conversor DC-DC se conecta a los pines de alimentación del ESP32 y este último se conecta a la IMU. Cabe resaltar que se tiene unos pines y una bornera extra, los cuales pueden ser utilizados para realizar diferentes mediciones o alimentar otros componentes.

Wire Routing

Al ya tener el esquemático, se procedió a realizar el *wire routing* del PCB. Sin embargo, para realizar esto era necesario realizar primero el cálculo del tamaño que debían tener las pistas y la separación entre estas. Ya que no se estaba trabajando con dispositivos de potencia, se tomaron las siguientes consideraciones:

- Un paso de corriente de 307 mA.
- Un aumento máximo de temperatura de 10°C.
- Un espesor de cobre de 1 oz/ft^2 (tomando en cuenta la disponibilidad de placas en la UVG).
- Un temperatura ambiente de 25°C.
- Una longitud promedio de conductor de 2 pulgadas.

Los resultados de estos cálculos son presentados en la Figura 47

ANSI PCB TRACE WIDTH CALCULATOR

Input Data			Results Data				
				Internal Traces		External Traces	
Field	Value	Units	Trace Data	Value	Units	Value	Units
Current (max. 35A)	307	mA	Required Trace Width	0.16	mm	0.06	mm
Temperature Rise (max. 100°C)	10	°C	Cross-section Area	0.01	mm²	0	mm²
Cu thickness	1	oz/ft²	Resistance	0.17	Ω Ohms	0.45	Ω Ohms
Ambient Temperature	25	°C	Voltage Drop	0.05	Volts	0.14	Volts
Conductor Length	2	inches	Loss	0.02	Watts	0.04	Watts
Peak Voltage	5	Volts	Required Track Clearance	0.61	mm		

Figura 47: Cálculo de ancho y separación de pistas.

Como se puede observar en la Figura 47, el ancho mínimo de pistas fue de **0.06 mm** mientras que la separación mínima entre pistas fue de **0.61 mm**. Consecuentemente, estas reglas fueron actualizadas en *Altium Designer* y se procedió a diseñar la placa, dando como resultado la placa de una sola cara, mostrada en las Figuras 48 y 49.

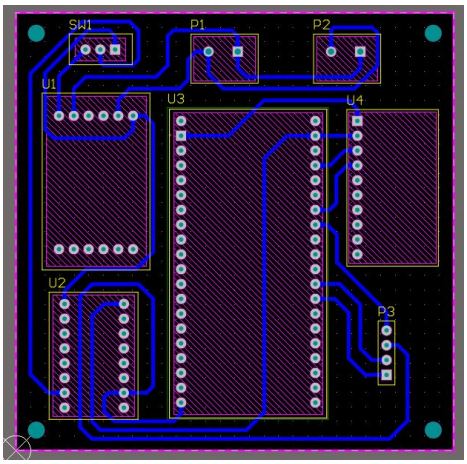


Figura 48: Vista de la capa inferior de la placa diseñada.

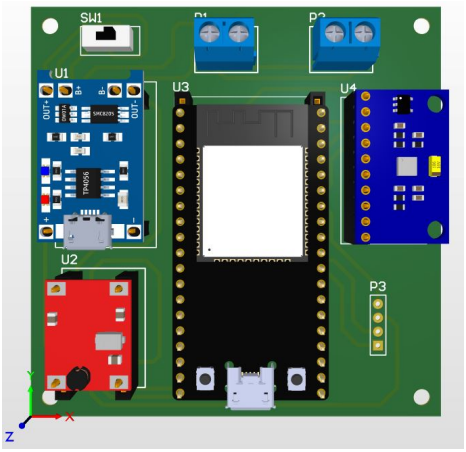


Figura 49: Vista en 3D de la placa diseñada.

9.1.3. Diseño de estuche

El estuche diseñado debió permitir posicionar la placa diseñada, además de poder acoplar los soportes de los marcadores y la correa de acople. Este estuche fue diseñado en *Autodesk Inventor* y fue fabricado en acrílico por medio de corte láser. El resultado de este diseño es presentado en la Figura 50

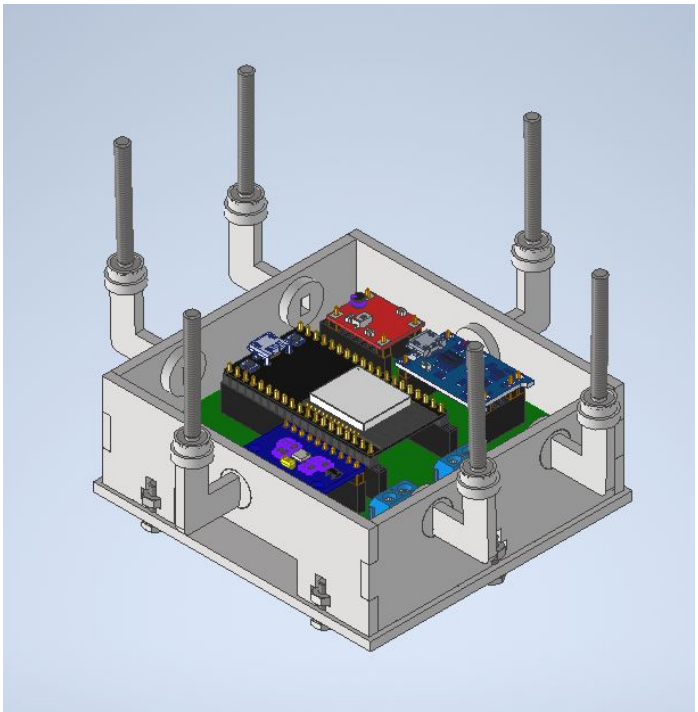


Figura 50: Estuche diseñado para la antena inteligente.

Cabe resaltar que finalmente, no se utilizaron piezas descargadas desde Thingiverse, en especifico para el acople de los marcadores. Esto es debido a que la impresora 3D disponible en la UVG no podía realizar impresiones con la dimensiones requeridas, ya que la rosca era demasiado pequeña. Consecuentemente, se procedió a comprar tornillos plásticos M4 los cuales se pegaron a los soportes y se acoplaron a los marcadores en la punta superior. El resultado final del estuche es presentado en la Figura 51.

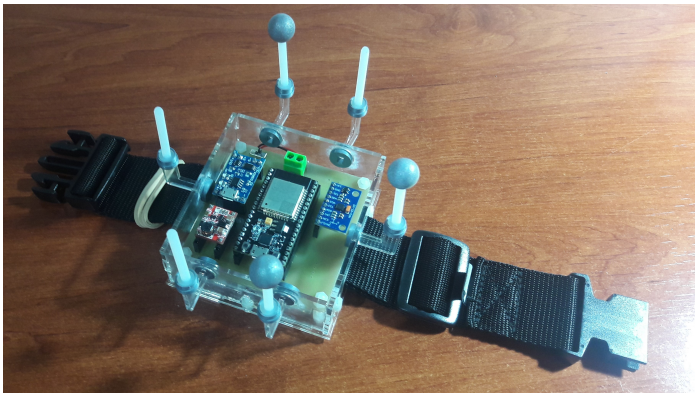


Figura 51: Estuche diseñado para la antena inteligente.

9.1.4. Definición de filtro de Kalman para esta aplicación

Ya que la antena inteligente no tenía un sistema dinámico como tal, se procedió a hacer una derivación de este para así poder aplicar una fusión de sensores por medio del filtro de Kalman. Primeramente, se definieron las variables de estado como se presenta en la Figura 52, los cuales son los valores de *roll*, *pitch* y *yaw* respectivamente.

$$x = \begin{bmatrix} \Phi \\ \Theta \\ \Psi \end{bmatrix}$$

Figura 52: Variables de estado del sistema.

Consecuentemente, las variables de entrada se definieron como se presenta en la Figura 53. Esto permitió obtener un sistema dinámico de un sistema que no tiene sistema dinámico.

$$u = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\Phi} \\ \dot{\Theta} \\ \dot{\Psi} \end{bmatrix}$$

Figura 53: Variables de entrada del sistema.

Posteriormente se definió a $\dot{x}$ como:

$$\dot{x} = u. \tag{6}$$

Ya teniendo (6) se procedió a realizar la discretización del sistema por medio de Forward Euler, dando como resultado:

$$x_k = x_{k-1} + \Delta t u_k. \tag{7}$$

$$y_k = x_k + v_k. \tag{8}$$

Con $A = I$, $B = \Delta t I$ y $C = I$.

Posteriormente, se debieron definir las desviaciones estándar que se encontraban dentro de la matriz de covarianza del ruido de medición y del error de estimación, de las cuales se presenta su forma a continuación:

$$Q_v = \begin{bmatrix} \sigma_{\Phi IMU} & 0 & 0 \\ 0 & \sigma_{\Theta IMU} & 0 \\ 0 & 0 & \sigma_{\Psi IMU} \end{bmatrix}$$

Figura 54: Matriz de covarianza del ruido de medición.

$$Q_w = \begin{bmatrix} \sigma_{\Phi OT} & 0 & 0 \\ 0 & \sigma_{\Theta OT} & 0 \\ 0 & 0 & \sigma_{\Psi OT} \end{bmatrix}$$

Figura 55: Matriz de covarianza del error de estimación.

Las desviaciones de la matriz presentada en la Figura 54 fueron encontradas dejando estática a la antena inteligente y guardando los datos de *Roll*, *Pitch* y *Yaw* provenientes de la IMU, para posteriormente obtener la desviación estándar de cada uno de estos datos. Por otra parte, las desviaciones de la matriz presentada en la Figura 55 se encontraron por medio del siguiente proceso:

Ya que se tenía el error de las mediciones del OptiTrack (0.435 mm), se procedió a sumar y restar este valor a la matriz de rotación esperada, la cual en este caso era la matriz identidad de 3X3. Esto dio como resultado tres matrices:

- Matriz identidad de 3X3
- Matriz identidad de 3X3 + el error de medición
- Matriz identidad de 3X3 - el error de medición

Posteriormente se convirtieron estas matrices de rotación a su contra parte de *roll*, *pitch* y *yaw*. Finalmente se restaron las dos matrices que contenían el error con la matriz de rotación esperada y se encontró la media entre los dos valores obtenidos.

Finalmente, el filtro de Kalman propuesto se presenta a continuación:

Predicción

$$\hat{x}_{K|K-1} = \hat{x}_{K-1|K-1} + \Delta t u_K, \tag{9}$$

$$P_{K|K-1} = P_{K-1|K-1} + F^T Q_w F. \tag{10}$$

Corrección

$$z_K = y_K - \hat{x}_{K|K-1}, \tag{11}$$

$$S_K = Q_v + P_{K|K-1}, \tag{12}$$

$$L_K = P_{K|K-1} S_K^{-1}, \tag{13}$$

$$\hat{x}_{K|K} = \hat{x}_{K|K-1} + L_K z_K, \tag{14}$$

$$P_{K|K} = (I - L_K) P_{K|K-1}. \tag{15}$$

Cabe resaltar que en este caso, el valor de $\hat{x}$ será el de los datos tomados por la IMU debido a que estos datos son obtenidos de una manera más frecuente que los obtenidos por el sistema OptiTrack. Por otra parte, ya que no se tiene certeza acerca de las condiciones iniciales del sistema, se asume un valor de σ_e grande en el valor inicial de $P_{K|K}$, el cual en este caso fue de 10.

9.2. Resultados

Ya que la conexión a la red de comunicación ya se había probado en el Capítulo 8, en este capítulo solo se procedió a comparar los valores de orientación de la IMU y el sistema OptiTrack, además de determinar si la aplicación de fusión de sensores era viable y mejoraba la exactitud de los valores de orientación obtenidos.

9.2.1. Comparación de datos de la IMU y el sistema OptiTrack

En esta subsección se quería comprobar que los datos obtenidos por la IMU posicionada en la antena inteligente, coincidieran con los datos obtenidos a través del sistema OptiTrack, variando como máximo en un error absoluto de 8°. Esto se hizo realizando diferentes movimientos dentro del ecosistema y verificando los valores de orientación dados por la IMU y el OptiTrack. Los resultados de esta prueba se muestran en el Cuadro 28. Cabe resaltar que la orientación de los ejes vistos en planta en la IMU (X y Y) no es igual a la orientación

de los ejes vistos en planta en OptiTrack (X y Z) por lo que el error debe ser tomado del valor absoluto de cada uno de los valores, ya que los signos del *pitch* y *roll* tienen el signo cambiado con respecto a los obtenidos en el OptiTrack.

Corrida	**Orientación**	**Orientación OptiTrack (°)**	**Orientación IMU (°)**	**Error (°)**
1	pitch	5.718	-6.283	0.57
	yaw	-9.912	-9.805	0.11
	roll	-0.410	-2.598	2.19
2	pitch	-16.21	16.697	0.49
	yaw	-3.323	-2.748	0.58
	roll	1.933	-0.499	1.43
3	pitch	-5.328	-1.679	3.65
	yaw	-15.343	-17.360	2.02
	roll	-29.026	25.825	3.20
4	pitch	-25.663	22.270	3.39
	yaw	-7.834	-9.581	1.75
	roll	-20.851	21.044	0.19
5	pitch	-0.763	-1.402	0.64
	yaw	23.299	22.537	0.76
	roll	-20.747	18.355	2.39

Cuadro 28: Comparación de mediciones de orientación dadas por la IMU y mediciones dadas por el sistema OptiTrack.

Como se puede observar en el Cuadro 28, el error absoluto más grande fue de 3.65°. Debido a que la métrica de evaluación de resultados definía que los resultados no debían variar en un valor mayor al 8°, se aceptaron los resultados como válidos. Consecuentemente, se concluye que los datos obtenidos por la IMU y el Optitrack son válidos y son candidatos a poder ser utilizados en la fusión de sensores.

9.2.2. Verificación de uso del filtro de Kalman

En esta subsección se quería verificar si el uso de fusión de sensores por medio de un filtro de Kalman mejoraba de alguna manera la exactitud de las mediciones de orientación obtenidas. Esto se realizó por medio de un programa desarrollado en MATLAB, en el cual se implementaba el filtro de Kalman propuesto, obteniendo los resultados presentados a continuación.

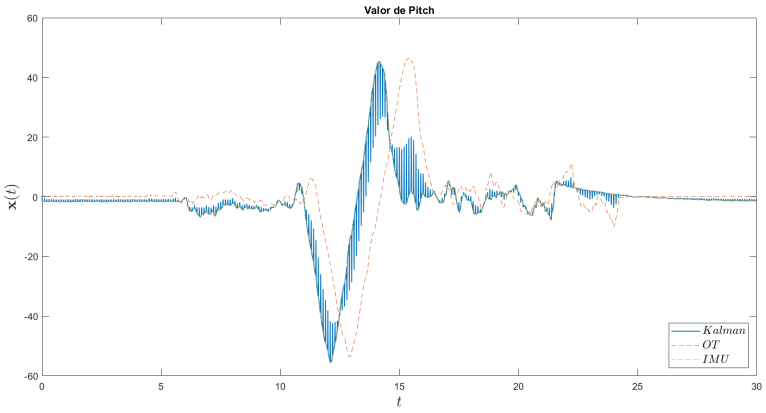


Figura 56: Valor de *pitch* utilizando fusión de sensores.

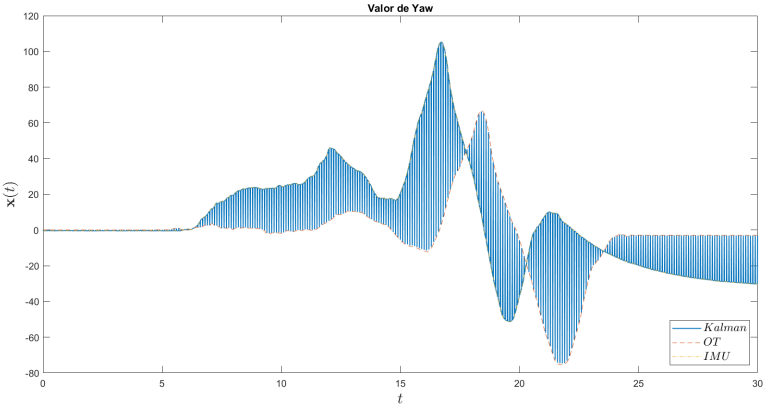


Figura 57: Valor de *yaw* utilizando fusión de sensores.

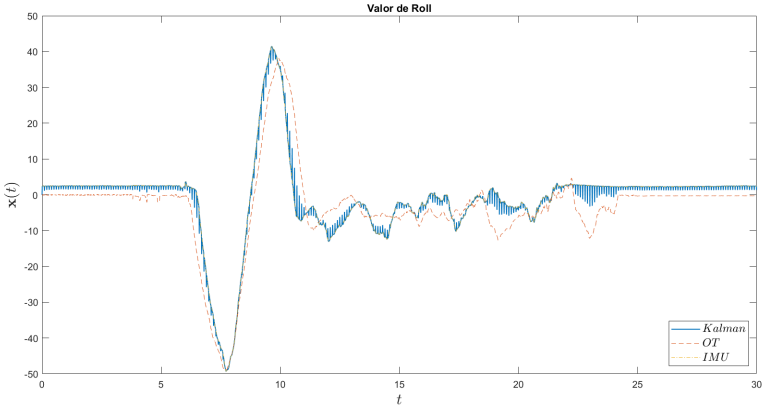


Figura 58: Valor de *roll* utilizando fusión de sensores.

Como se puede observar en las Figuras 56, 57 y 58, la implementación de la fusión de sensores tiene cierto efecto en las mediciones de los ángulos de orientación, mejorando en cierta partes el ruido obtenido por las mediciones por separado. Consecuentemente, se

concluye que la implementación de fusión de sensores brindaría una mejoría a los datos obtenidos, proveyendo una información más clara de la orientación en la que se encuentre el objeto en el que esté posicionada la antena inteligente.

CAPÍTULO 10

Conclusiones

- Se logró ensamblar, calibrar y configurar el sistema de captura de movimiento OptiTrack, permitiendo así la obtención de información de la pose de los agentes que se encuentren en el ecosistema.
- Se desarrolló e implementó un programa que permitiera, por medio de una red WiFi y el protocolo MQTT, enviar información obtenida del OptiTrack a los agentes además de recibir información relevante que los agentes envíen.
- Se desarrolló e implementó una librería que permitiera que un ESP32 pudiera conectarse e interactuar dentro la red de comunicación WiFi.
- Se diseñó e implementó una antena inteligente la cual se puede conectar al sistema de comunicación desarrollado y permite a sistemas no robóticos interactuar en el ecosistema.
- Se comprobó que el uso de un filtro de Kalman para la antena inteligente proveía una mejora en los datos obtenidos, proveyendo una información más clara de la orientación en la que se encuentre el objeto en el que esté posicionada la antena inteligente.

CAPÍTULO 11

Recomendaciones

- Ampliar la funcionalidad del sistema de captura de movimiento para que el ecosistema pueda admitir experimentos más complejos en los que se pueda incluir la captura de movimiento en humanos.
- Probar si el uso de JSON en vez de strings en el envío de datos tiene algún efecto en la rapidez en la que se efectúe el data parsing.
- Utilizar chips discretos en la antena inteligente para tratar de disminuir el tamaño de esta.
- Si se desea expandir el análisis a un conjunto de cuerpos rígidos (esqueletos), es necesario modificar la altura e inclinación de las cámaras para que así estas tengan mayor rango de visión
- Analizar si tiene algún efecto en la latencia si se cambia la topología de bus a estrella.
- Diseñar una estructura que permita que las cámaras sean menos propensas a golpes y por ende a descalibraciones.
- Utilizar persianas en las ventanas del laboratorio para evitar que luces externas interfieran con los marcadores durante las mediciones.

CAPÍTULO 12

Bibliografía

[1] D. P. y P. Glotfelter y L. Wang y M. Mote y A. Ames y E. Feron y M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," en *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, págs. 1699-1706. DOI: 10.1109/ICRA.2017.7989200.

[2] J. Castañeda, "Diseño e implementación de una red de comunicación wifi e interfaz gráfica para una mesa de pruebas de robótica de enjambre," Tésis de Pregrado, Departamento de Ingeniería Mecatrónica. Universidad del Valle de Guatemala, Diciembre de 2020.

[3] M. F. y Z. Pan y D. Stirling y F. Naghdy, "Human motion capture sensors and analysis in robotics," en *Industrial Robot*, vol. 38, 2011, págs. 163-171. DOI: 10.1108/01439911111106372.

[4] Mo-Sys, *What is motion capture and how does it work?* https://www.mo-sys.com/what-is-motion-capture-and-how-does-it-work/, (Accedida el 21/05/2021).

[5] IDIS, *Motion Capture*, https://proyectoidis.org/motion-capture/, (Accedida el 21/05/2021).

[6] N. Point, *OptiTrack Documentation*, https://v22.wiki.optitrack.com/index.php?title=OptiTrack_Documentation_Wiki, (Accedida el 21/05/2021).

[7] ——, *Prime$^{\times}$41 – Specs*, https://optitrack.com/cameras/primex-41/specs.html, (Accedida el 21/05/2021).

[8] CISCO, *¿Qué es la tecnología wifi? Definición y tipos*, https://www.cisco.com/c/es_mx/products/wireless/what-is-wifi.html#~preguntas-y-respuestas, (Accedida el 21/05/2021).

[9] A. Zone, *Qué es el WiFi y cómo funciona para conectar todo a Internet*, https://www.adslzone.net/reportajes/tecnologia/que-es-wifi-como-funciona/, (Accedida el 19/09/2021).

[10] J. Jiménez, *Cifrados Wi-Fi inseguros que se pueden romper hoy en día*, https://www.redeszone.net/tutoriales/redes-wifi/tipos-cifrados-wifi-inseguros/, (Accedida el 19/09/2021).

[11] U. de Valencia, *Tema 2. Redes de Comunicación: Topología y enlaces*, https://www.uv.es/rosado/courses/sid/Capitulo2_rev0.pdf, (Accedida el 21/05/2021).

[12] TST-Sistemas, *Protocolo MQTT*, http://www.tst-sistemas.es/mqtt/, (Accedida el 21/05/2021).

[13] MQTT, *MQTT - The Standard for IoT Messaging*, https://mqtt.org/, (Accedida el 21/05/2021).

[14] Eclipse, *Eclipse Mosquitto*, https://mosquitto.org/, (Accedida el 21/05/2021).

[15] ESP32, *The Internet of Things with ESP32*, http://esp32.net/, (Accedida el 21/05/2021).

[16] Arrow, *What is IMU? Inertial Measurement Unit Working*, https://www.arrow.com/en/research-and-events/articles/imu-principles-and-applications, (Accedida el 21/05/2021).

[17] UNIT-Electronics, *IMU MPU6050*, https://uelectronics.com/producto/imu-mpu6050-6-grados-de-libertad/, (Accedida el 21/05/2021).

[18] Fierce-Electronics, *What is sensor fusion?* https://www.fierceelectronics.com/sensors/what-sensor-fusion, (Accedida el 21/05/2021).

[19] M.I.T, *16.333: Lecture 15*, https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-333-aircraft-stability-and-control-fall-2004/lecture-notes/lecture_15.pdf, (Accedida el 21/05/2021).

[20] U. de Sevilla, *El Filtro de Kalman*, http://bibing.us.es/proyectos/abreproy/11611/fichero/Memoria%252FAp%C3%A9ndice+B+-+El+Filtro+de+Kalman.pdf+, (Accedida el 23/05/2021).

[21] M. J. y Roni Rosenfeld y David Farrow y Ryan J. Tibshirani, *Kalman Filter, Sensor Fusion, and Constrained Regression: Equivalences and Insights*, https://www.stat.cmu.edu/~ryantibs/papers/sensorfus.pdf, (Accedida el 23/05/2021).

[22] IPC, *IPC-2221*, http://jp.jetpcb.com/Cht/Document/ipc-2221all.pdf, (Accedida el 21/05/2021).

[23] C. Ayala, *Protocolo para diseño de tarjetas de circuitos en máquina ruteadora*, http://bibliotecadigital.usbcali.edu.co/bitstream/10819/6149/1/Tarjetas_Circuitos_Ruteadora_Ayala_2018.pdf, (Accedida el 21/05/2021).

[24] N. Point, *OptiTrack Accesories*, https://optitrack.com/accessories/, (Accedida el 20/09/2021).

[25] MechaniCalc, *Trade Study*, https://mechanicalc.com/calculators/trade-study/, (Accedida el 13/10/2021).

CAPÍTULO 13

Anexos

Todos los archivos relacionados con el desarrollo de la red de comunicación y la antena inteligente se encuentran en el siguiente repositorio:

https://github.com/mezea-uvg/robotat/tree/17092-Dev

CAPÍTULO 14

Glosario

Broker: Servidor que recibe los mensajes publicados por medio del protocolo MQTT y permite que los clientes suscritos, dependiendo del tópico, reciban los mensajes correspondientes. 17, 41–46

Dirección IP: Conjunto de números que identifica a un dispositivo conectado a una red. 41, 42

Router: Dispositivo de hardware que permite que varios dispositivos se puedan conectar entre si por medio de una red. 39

Trade study: Análisis entre varias opciones que permite determinar qué opción es la mejor basándose en un sistema de méritos. 25, 39, 56