Universidad Rey Juan Carlos

Escuela Superior de Ciencias Experimentales y Tecnología

GRADO EN INGENIERÍA
DE TECNOLOGÍAS INDUSTRIALES

Trabajo de Fin de Grado

Sistema de navegación autónomo en entornos reales y simulados para situaciones de emergencia

Noelia Fernández Talavera

Directora: M.ª Cristina Rodríguez Sánchez

Codirector: Juan Jesús Roldán Gómez

Curso Académico 2020/21

Grado en Ingeniería de Tecnologías Industriales

Trabajo de Fin de Grado

El presente trabajo, titulado ***SISTEMA DE NAVEGACIÓN AUTÓNOMO EN ENTORNOS REALES Y SIMULADOS PARA SITUACIONES DE EMERGENCIA,*** constituye la memoria correspondiente a la asignatura Trabajo de Fin de Grado que presenta Dª. **Noelia Fernández Talavera** como parte de su formación para aspirar al Título de Graduada en Ingeniería de Tecnologías Industriales. Este trabajo ha sido realizado en la **Universidad Rey Juan Carlos** en el **Departamento de Matemática Aplicada, Ciencia e Ingeniería de los Materiales y Tecnología Electrónica** bajo la dirección de **M.ª Cristina Rodríguez Sánchez** y **Juan Jesús Roldán Gómez (**profesor ayudante doctor en el Departamento de Ingeniería Informática, EPS-UAM).

Móstoles, 14 de julio de 2021.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE PLANOS

ACRÓNIMOS Y ABREVIATURAS

- APTB → Asociación Profesional de Técnicos de Bomberos
- BBDD → Base de Datos
- BLE → Bluetooth Low Energy
- CAD → Diseño Asistido por Computadora
- eCO_2 → Dióxido de Carbono Equivalente Calculado
- EDT → Estructura de Desglose de Trabajo
- EI → Equipo de Intervención
- EE.UU → Estados Unidos
- HDMI → High Definition Multimedia Interface
- HTTP → Protocolo de Transferencia de Hipertexto
- I2C → Circuito Integrado
- IMU → Unidad de Medición Inercial
- IVA → Impuesto de Valor Añadido
- LAMP → Sistema de Infraestructura de Internet
- LabTel → Laboratorio de Tecnología Electrónica
- LED → Diodo Emisor de Luz
- MAC → Media Access Control
- MSCI → Modelos de Simulación Computacional de Incendios
- NFPA → National Fire Protection Association
- OS → Sistema Operativo/ Operating System
- PWM → Modulación por Ancho de Pulsos
- RAM → Memoria de Acceso Aleatorio
- RPM → Revoluciones Por Minuto
- SENiaLab → Laboratorio de desarrollo de Sistemas de Navegación Sensorial y de Sistemas de Monitorización.
- SLAM → Simultaneous Localization and Mapping
- SubTR→ Subtarea
- TFG → Trabajo de Fin de Grado
- TOF → Tiempo de vuelo
- TR → Tarea
- TVOC → Concentración de Compuestos Orgánicos Volátiles
- USB → Bus Universal en Serie
- URJC → Universidad Rey Juan Carlos
- UWB → Ultra-Wide Band/ Banda Ultra Ancha
- WiFi → Wireless Fidelity

AGRADECIMIENTOS

Para empezar, me gustaría agradecer a mi tutora Dra. Mª Cristina Rodríguez Sánchez su confianza en mí para lograr este desafío, su apoyo y su ayuda frente a los obstáculos en el transcurso de este trabajo. La oportunidad de colaborar y formar parte en uno de sus proyectos me ha enseñado que no existen límites en el aprendizaje.

Gracias también a mi cotutor Dr. Juan Jesús Roldán Gómez por transmitirme su conocimiento, experiencia y guiarme en campos tan desconocidos para mí como el de las simulaciones. Ha sido un gran descubrimiento como profesional y como persona, pues su trato siempre ha sido amable y su disposición inmediata para ayudarme cuando existían dificultades.

Por otro lado, dar las gracias al cuerpo de bomberos de Alcorcón y a todas las entidades dedicadas a la Prevención, Emergencias, Seguridad, Intervención y Evacuación por colaborar con nosotros, implicarse en el proceso de desarrollo del proyecto y acompañarnos en las pruebas.

De igual modo, gracias a los laboratorios LabTel, SENiaLab y al personal involucrado en los proyectos de investigación ya que sin ellos esta labor no hubiera sido posible.

A continuación, mostrar el mayor agradecimiento a mi familia, en especial a mis padres Rosa Mª Talavera Requena y Ángel Fernández Gonzalo, y a mi querida hermana Marina Fernández Talavera por haberme dado la oportunidad de llegar hasta aquí y confiar en mis capacidades. Me habéis enseñado lo que es el sacrificio, la constancia y el trabajo, pero también la importancia de estar rodeada de gente que me apoya, me quiere y cuida incluso en los peores momentos. Este logro que he conseguido es mérito vuestro. También quiero acordarme de mis abuelos que, a pesar de la distancia, han sido mis mayores referentes para lograr todos mis objetivos.

Por último, gracias a todos los profesores, compañeros y amigos que han formado parte de esta etapa tan enriquecedora de mi vida y de los que he aprendido tanto. En particular, gracias a Daniel, Jennifer y Benjamín por cogerme la mano, reír y llorar a mi lado.

RESUMEN

Los equipos de intervención (EI) actúan en situaciones hostiles y de baja visibilidad, donde la reducción de la siniestralidad y el tiempo de intervención son factores principales para garantizar mayor seguridad en la actuación. Para ello, el uso de la tecnología adecuada y la información a priori del entorno desempeñan un papel fundamental para asegurar su propio bienestar y el de las personas a evacuar.

En primer lugar, este Trabajo de Fin de Grado mejorará el diseño de un sistema autónomo previo, desarrollando e implementando un nuevo modelo de agente autónomo más robusto, mejorando el control remoto para ser usado por un bombero, e incorporando la tecnología de localización para interiores UWB que proporcionan unas balizas comerciales, pudiendo así identificar la posición exacta de cada amenaza. Además, el autómata podrá esquivar obstáculos y crear una ruta de evacuación alternativa hacia la salida para posibles víctimas.

En segundo lugar, se incorporarán sensores de monitorización del entorno que recogen datos de interés medioambiental como temperatura, eCO_2, % de humedad relativa, compuestos orgánicos volátiles, concentración bruta de H_2 y de etanol. Estos serán almacenados en una base de datos y posteriormente representados en una plataforma visual para su interpretación, evaluando así el nivel de riesgo para la salud humana que existe en la intervención y las posibles estrategias para afrontar la situación.

Por último, se abordará una nueva línea de investigación en el departamento con la creación de una simulación tanto del agente autónomo como del entorno de intervención, que presenta las mismas características que el escenario real. Una vez diseñado todo, por un lado, se programará la conducta del entorno incorporando fuego, obstáculos y personas, y por otro lado se programará el comportamiento del robot terrestre creando un modo manual controlado por teclado, un modo automático haciendo un recorrido similar al que se realizaría en el entorno real con las balizas de localización, y un modo de evacuación donde el autómata calculará la ruta más fiable y rápida hacia la salida. Su finalidad es estimar tiempos de intervención y crear recorridos alternativos más seguros identificando las zonas más peligrosas del espacio.

Desde hace más de una década, el área de Tecnología Electrónica junto con LabTel [1] y SENiaLab [2], lleva desarrollando proyectos de investigación y trabajos de fin de grado en este área de trabajo, donde se cuenta con la colaboración de los cuerpos de Prevención, Emergencias, Seguridad, Intervención y Evacuación, la Dirección de la Agencia de Seguridad y Emergencias de Madrid 112, la Asociación Profesional de Técnicos de Bomberos y la Jefatura de Bomberos de Alcorcón.

1. INTRODUCCIÓN

El presente Trabajo de Fin de Grado, forma parte de la línea de investigación de Sistemas inalámbricos y Aplicaciones Industriales del área de Tecnología Electrónica perteneciente al Departamento de Matemática Aplicada, Ciencia e Ingeniería de los Materiales y Tecnología Electrónica. Este departamento está involucrado en proyectos de localización sensorial y navegación en situaciones de emergencia asociados a la bioingeniería y accesibilidad, correspondiente al laboratorio LabTel [1] y SENiaLab [2]. Resultado de todos los trabajos y como evolución de un anterior proyecto denominado "Implementación de tecnología UWB para localización y guiado de interiores" [3] surgen nuevas necesidades y retos planteados fruto de la colaboración con la Asociación Nacional Profesional de Técnicos de Bomberos.

Algunas de las problemáticas que existen en las situaciones de emergencia con fuego son la imposibilidad de poder recorrer el espacio de manera segura, la falta de información del ambiente o la baja visibilidad. Esto dificulta las tareas de extinción de incendios que incluyen la prevención, vigilancia y extinción [4]. El mejor incendio es el que no se produce y, por ello, evitar su descontrol y conocer el comportamiento en función de su localización es imprescindible para su cese [5]. Por eso, en el presente Trabajo de Fin de Grado, se va a ofrecer una solución a estos problemas, analizando las tecnologías de navegación y geolocalización de focos de fuego, los sistemas actuales de monitorización de parámetros ambientales y el uso de simulaciones enfocadas a dar soporte a situaciones de emergencia.

En el Gráfico 1, se muestran los datos que proporciona el Ministerio para la Transición Ecológica y el Reto Demográfico en su informe sobre "Coordinación Nacional contra Incendios (2020)" [6] y los datos del "Estudio de Victimas de Incendios en España de la APTB, Fundación Mapfre (2020)" [7] sobre incendios y sus consecuencias. En la parte superior del gráfico se encuentran los datos reales mientras que en la parte inferior se representan los mismos en escala logarítmica para una mejor visualización. La tendencia es lineal en general, aunque destacan los incendios en interiores (34.029) por encima de los demás, siendo el 77 % de los accidentes que sufren los bomberos en edificaciones. Además, las víctimas mortales se cuantifican como 165 personas de las cuales 158 fueron en incendios y 7 en explosiones, lo que supone un aumento del 34,14 % más respecto al año anterior [8].

Los datos oficiales sobre incendios en el año 2020 aún no se han recopilado, sin embargo, lo que sí se sabe en referencia a los incendios en el ámbito forestal es que ha habido una reducción del 35 % respecto a la media del último decenio [9]. Además, este suceso ha sido muy

significativo ya que se inició una campaña de lucha contra incendios que, en combinación con la meteorología, ha hecho que este descenso se encuentre cerca del 50 %.

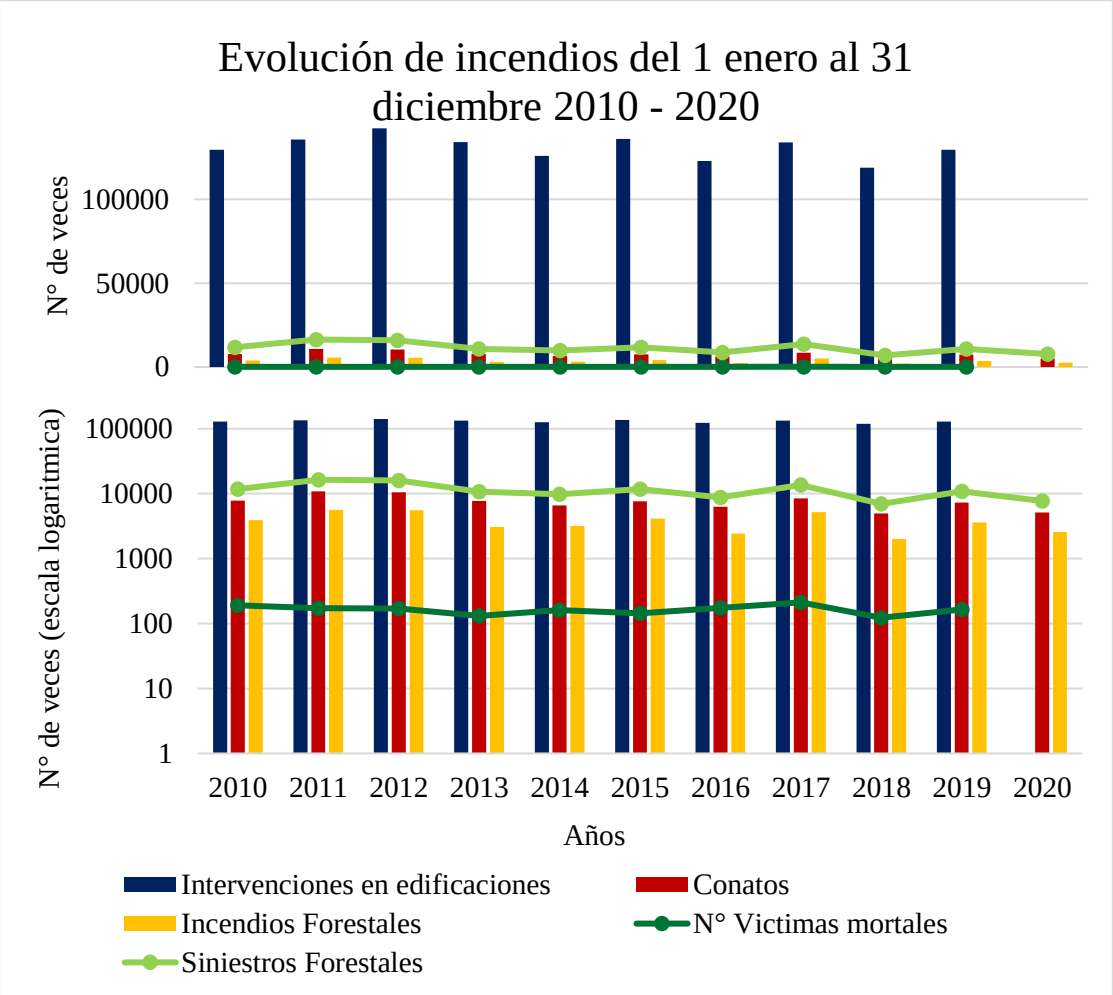


Gráfico 1. Evolución de incendios del 1 enero al 31 diciembre 2010-2020. Elaboración propia

Esto demuestra que se debe intentar solventar la problemática que existe en la siniestralidad del personal de emergencias. En la actualidad, el crecimiento industrial, el desarrollo y el acceso a mejores tecnologías han provocado que los sistemas de prevención y detección de incendios se hayan visto en la necesidad de empezar a implantar novedades tecnológicas que faciliten el trabajo y minimicen los daños con un aliado innovador, el robot [10]. Los robots pueden asumir riesgos que las personas no pueden, introduciéndose en espacios inseguros en los momentos más peligrosos y recabando datos que agilicen las labores de extinción. Algunos ejemplos son el SmokeBot [10] que es capaz de operar a través del humo, evaluando temperaturas, mediciones de gases y comportamiento del fuego; los drones [4, 10] cuya función es sobrevolar zonas afectadas e informar en tiempo real a los equipos de extinción mediante videos de alta resolución, o los robots bombero [10] que lanzan agua y otros fluidos para extinguir las llamas.

1.1 SISTEMAS DE NAVEGACIÓN Y LOCALIZACIÓN

La localización en interiores es uno de los grandes problemas a los que los equipos de intervención tienen que enfrentarse [11]. En interiores, conocer las lesiones que el fuego provoca en un edificio, generando daños estructurales o afectando a su estabilidad y su localización, es de vital importancia para los cuerpos de seguridad ya que de esto depende su supervivencia. Asimismo, el efecto que la temperatura produce sobre los materiales estructurales y la circulación del humo que afecta a la visibilidad y la ventilación, incrementan la dificultad de afrontar estas situaciones [12]. Las intervenciones en incendios no solo suponen la extinción del fuego, sino que también son aquellas acciones orientadas a la búsqueda y rescate de víctimas mediante planes de actuación eficientes. Por lo tanto, agregar medidas de prevención generales como sistemas de detección y monitorización de incendios, reduciría el número de accidentes potencialmente peligrosos [11] y el impacto económico negativo.

Para monitorizar el tiempo de navegación de un edificio en llamas, los investigadores del Kumon National Institute of Technology y la Seoul National University [13] han desarrollado un sistema de mapeo y localización para disminuir tiempos de rescate. Esta plataforma funciona con el algoritmo SLAM [14] [15] que consiste en la creación de un mapa de puntos del entorno. Sin embargo, para situaciones donde hay humo denso esta tecnología no es viable pues no es capaz de reproducir los interiores.

Por otro lado, existen tecnologías inalámbricas como WiFi [16], Bluetooth o Bluetooth Low Energy (BLE) [17] que se implementan en balizas con una estructura de nodos en estrella. Las balizas permanecen conectadas al dispositivo que se quiere localizar, obteniendo la posición de este con un error de 2 metros aproximadamente. El problema es que estos sistemas dependen de parámetros del entorno, provocando interferencias y errores en el posicionamiento. No obstante, existen dos sistemas cuya precisión en entornos hostiles es milimétrica y son los apropiados para desempeñar la función de localización y posicionamiento. Uno de ellos es el posicionamiento estático por ultrasonidos [18], que necesita una visión directa entre el emisor y el receptor, y los algoritmos no son necesariamente robustos para obtener buenos resultados interceptando obstáculos [19]. El otro es el sistema de Banda Ultra Ancha (UWB) que utiliza un ancho de pulsos ultracorto en un determinado ancho de banda para crear la localización; su ventaja es que es capaz de atravesar materiales y es inmune a interferencias de otras señales [16].

La robótica lleva unos años formando parte de la lucha contra los incendios forestales, siendo el estado de California pionero en encontrar soluciones para hacer frente a los incendios, que ya han quemado más de 400.000 hectáreas como consecuencia de la falta de lluvias, olas de calor o periodos de sequía [20]. Otro ejemplo es el robot bombero Colossus cuyo papel fue

protagonista en la extinción del fuego de la Catedral de Notre Dame en 2019 [21] o el incendio del Parque Nacional de Doñana en 2017 [10]. Sin embargo, son pocos los dispositivos que pueden acceder a un incendio en el interior de un edificio dada la inadecuada tecnología con la que están creados y la incapacidad que existe para visualizar el interior en presencia de humo denso.

Por todo ello, la solución para este primer problema es la creación de un sistema de seguimiento autónomo y toma de decisiones en tiempo real basado en tecnología de ultrasonidos para esquivar obstáculos y la UWB para el posicionamiento geográfico. Esto aportará información sobre recorridos alternativos de intervención, disminuirá tiempos de respuesta en misiones críticas y minimizará riesgos y accidentes derivados del desconocimiento del entorno hostil [11].

1.2 MONITORIZACIÓN DE PARÁMETROS AMBIENTALES DE INTERÉS

Los bomberos son los principales afectados en las intervenciones ya que es frecuente que se lesionen o incluso fallezcan a pesar de la existencia de múltiples protocolos de prevención. Se debe a que estos nunca tienen toda la información de la situación a la que se van a enfrentar, pudiéndose encontrar con explosiones de gas, humo denso, llamas, edificios bajo condiciones inseguras, construcciones en ruinas, deterioradas o dañadas [22], lo que produce una alta incertidumbre.

El conjunto de exposiciones potencialmente peligrosas que sufren los bomberos durante el desarrollo de su actividad es muy variado, tales como la exposición a sustancias peligrosas, ruido y riesgos biológicos; aspectos psicosociales, y factores ergonómicos y de carga física. Sin duda, la exposición a sustancias químicas peligrosas es lo más grave que soportan los bomberos porque tiene efectos de tipo respiratorio o generan cáncer [23]. Se producen por tareas de contacto directo con el fuego y humo, y las características que presenta este colectivo difieren de la forma habitual a la que se produce en los ambientes laborales. Los bomberos, salvo excepciones, no conocen el tipo de fuego ni los riesgos inherentes del mismo que se van a encontrar, y la posibilidad de medición de concentraciones de sustancias en el aire es muy limitada dadas las dificultades operativas. A pesar de que en ciertos países existen registros específicos con información apta para la actuación, los datos no siempre están disponibles ni actualizados.

Todos los años la NFPA [24] genera un informe sobre el número de bomberos que resultan lesionados, sufren enfermedades o mueren mientras están de servicio en los Estados Unidos (Gráfico 2). Aquí no solo se recogen los datos del momento de la intervención sino de posibles enfermedades que se generan a largo plazo como el cáncer, derivados de un tiempo de exposición prolongado en estos entornos.

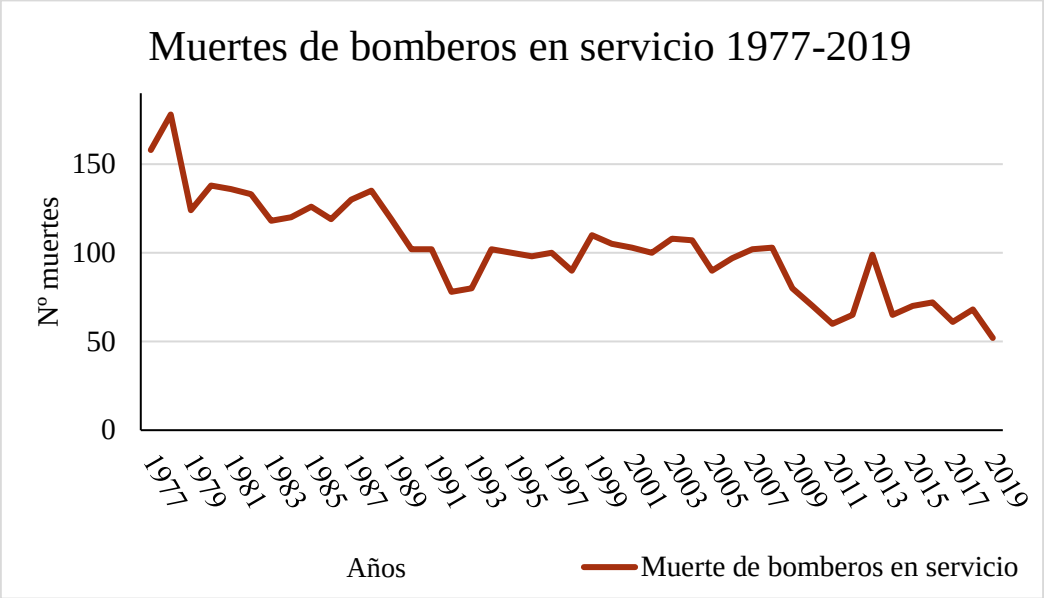


Gráfico 2. Número de muertes de bomberos 1977 – 2019, NFPA. Elaboración propia

En 2019 fallecieron 48 bomberos estando de servicio en los EE. UU, apreciándose una caída en las cifras respecto a los cuatro años anteriores. De entre el total de datos, las causas más destacables son: el 27 % de las muertes se produjeron en servicio y fueron causa del incendio o de explosiones por gases tóxicos, otro 19% corresponde a alarmas de emergencia y lo más significativo es que el 10 % de las muertes se produjeron practicando o entrenando [24].

Estos datos son recopilados bajo el estudio de bomberos con un rango de edad de entre los 21 años y los 81 años. Sin embargo, del porcentaje más alto de decesos es producido en bomberos por encima de los 60 años cuya principal causa es una parada cardiaca debido a enfermedades crónicas producidas por el desconocimiento de la calidad del aire en antiguas intervenciones.

Por todo esto, se demuestra la importancia de dotarles con la tecnología de comunicación y detección adecuada para conocer a priori las condiciones que van a afrontar, pudiendo generar así un plan de actuación eficiente durante las intervenciones y salvaguardar su seguridad.

La incorporación de dispositivos de detección de parámetros ambientales en interiores aún no está desarrollada de forma eficiente y por eso, en este trabajo se ofrecerá una alternativa que complementa información a este segundo problema, creando un agente autónomo que sea capaz de registrar estos parámetros en tiempo real para posteriormente visualizarlos. Así, de forma previa, durante y posterior a la intervención, los bomberos serán capaces de monitorizar las condiciones ambientales de un entorno hostil asociado a una emergencia. De la misma forma, se pueden minimizar riesgos mejorando la información sobre el escenario y reduciendo los tiempos de exposición en la intervención [11].

1.3 SIMULACIONES

Desde hace unos años, el uso del big data o los software de análisis de datos avanzados para monitorizar los sistemas de detección, ayudan a la coordinación en incendios para que las actuaciones sean globales, rápidas y eficientes [10]. La Inteligencia Artificial combinada con el 5G hacen que los sistemas de simulación y modelos predictivos tengan una función crítica, ya que integran datos de distintas fuentes, generan proyecciones sobre cómo se va a desarrollar el incendio una vez empezado y dónde es probable que se produzcan nuevos focos. Un ejemplo, es el software de simulación y predicción del Departamento Forestal y Protección contra Incendios del estado californiano Calfire [25]. Permite recrear operaciones así como el manejo de helicópteros en vuelos simulados y coordinar medios aéreos. Este sistema ha sido implantado en varios estados y empresas como Tecnosylva [26] y en algunas comunidades autónomas españolas como Andalucía.

Por el contrario, existen pocos modelos de simulación computacional de incendios (MSCI) para interiores. Uno de ellos es el Fire Dynamics Simulator (FDS) [27], un código usado para crear remolinos grandes, flujos de baja velocidad, transporte de humo y calor en incendios. Para ver los resultados es necesario hacer uso de un programa de simulación llamado Smokeview (SMV). Otro es el desarrollo de simulaciones de multitudes y evacuaciones como el Simulador de Movilidad de Persona en espacios Cerrados (SIMPCE) [28] o las simulaciones de comportamiento humano [29]. Sin embargo, estas simulaciones no permiten la visualización del entorno real, personas o del comportamiento del fuego dinámico de forma simultánea, y por lo tanto, no sirven para dar apoyo en situaciones de emergencia.

Para solventar este último problema, se propondrá la creación de una simulación de interiores para desarrollar mejores estrategias en la intervención y dar apoyo a los equipos de emergencia. Se llevará a cabo haciendo un estudio de los programas actuales de simulación de entorno, que ofrezcan la posibilidad de crear un entorno dinámico similar al real y sean compatibles con el diseño simulado de un autómata programado que replique los mismos movimientos que el agente autónomo real.

Para concluir, se pone de manifiesto la necesidad de crear un agente autónomo que será un robot terrestre, capaz de monitorizar datos de interés ambiental y localización, que sean registrados y visibles en tiempo real, y diseñar de igual modo una simulación que replique las mismas condiciones que un incendio en interiores. Esto servirá para crear protocolos de acción más seguros y eficaces.

2. OBJETIVOS

El objetivo principal de este Trabajo de Fin de Grado es ofrecer una solución que mejore la solución previa de un sistema de navegación autónomo con mayor robustez, localizable en tiempo real y que permita la monitorización de un escenario hostil, todo ello aplicado tanto en un entorno simulado como real. Partiendo de este objetivo principal son dos los objetivos secundarios que se plantean en este proyecto.

Uno de ellos es la creación de un sistema de localización y navegación autónoma en espacios interiores para situaciones de emergencia donde sea necesario conocer parámetros de utilidad, como la posición en tiempo real del sistema autónomo y parámetros ambientales del entorno. Esto se conseguirá unificando en un solo componente independiente el sistema de la Banda Ultra Ancha (UWB) [30], que aportará comunicación para recibir información de la intervención con carácter preventivo y del espacio interior, y añadiendo componentes y sensores que permitan la monitorización de la calidad del ambiente. Los parámetros que se han seleccionado han sido la temperatura, % de humedad y calidad de aire (eCO_2, concentración de compuestos orgánicos volátiles, concentración bruta de hidrógeno y etanol). Además, este autómata terrestre deberá tener la capacidad de ser dirigido de manera remota por una persona y reconocer el entorno evitando colisionar con posibles obstáculos.

El otro objetivo es la implementación y diseño de un entorno simulado con características similares a las que se presentan en el entorno real para poder practicar y mejorar nuevos métodos de intervención, además de poder adaptar el comportamiento del sistema de navegación autónoma a cada escenario requerido, sin la necesidad de tenerlo físicamente.

Por lo tanto, para cumplir con estas motivaciones planteadas y con los objetivos establecidos se deben definir una serie de objetivos intermedios:

1. Documentación del estado actual del proyecto en lo relacionado con sistema de navegación autónomo [3] , analizando su comportamiento, evaluando posibles mejoras para una nueva versión y estudiar el funcionamiento y la integración del sistema de localización UWB [31].

2. Evaluación de las necesidades existentes en un entorno de intervención y estudio de los diferentes métodos de simulación para escoger el que mejor se ajuste a nuestras necesidades.

3. Creación y diseño de un agente autónomo simulado que presente las mismas características físicas que el agente autónomo real, además del diseño de cada componente o sensor que este contenga, respetando la misma arquitectura física que la estructura real.

4. Creación y diseño de un entorno simulado que presente las mismas características físicas (rugosidad de suelo, textura de paredes, mobiliario y obstáculos) que el entorno donde se va a llevar a cabo la intervención, integrando el diseño del agente autónomo simulado con los mismos sensores y comportamiento de movimiento (modo manual, autónomo y evacuación) que el real.

5. Evolución de una nueva versión de un agente autónomo guiado por sensores de ultrasonidos, añadiendo componentes que permitan mayor precisión en el trabajo requerido y creación de modos de movimiento manual, autónomo y evacuación.

6. Obtención de parámetros medioambientales de gran interés como temperatura del entorno, % de humedad, dióxido de carbono (eCO_2), concentración de compuestos orgánicos volátiles (TVOC), además de concentración bruta de hidrógeno (H_2) y etanol, para garantizar la seguridad de las personas en una intervención.

7. Diseño y desarrollo de una BBDD que contenga los datos obtenidos por las balizas de localización y los sensores, además de la comunicación entre sistemas a través de un ordenador de placa reducida (placa Raspberry Pi 4) de procesamiento de datos.

8. Exportación de los datos registrados en la BBDD a una herramienta de software libre llamada Grafana [32], con el objetivo de visualizar datos de serie temporales para un mejor análisis y comprensión.

9. Evaluación y comparación de las similitudes y diferencias del entorno real y simulado, y estudio de posibles mejoras.

10. Realización de pruebas en la torre de bomberos de Alcorcón y posterior evaluación del sistema y datos obtenidos.

3. METODOLOGÍA Y TECNOLOGÍAS DE DESARROLLO

A continuación, se explica el plan de trabajo, la metodología y las tecnologías de desarrollo para alcanzar los objetivos planteados. Para ello se establecerán los siguientes subcapítulos: organización y duración del proyecto, selección de las tecnologías de desarrollo y selección del hardware y software.

3.1 ORGANIZACIÓN Y DURACIÓN DEL PROYECTO

Para llevar a cabo este proyecto es necesario definir un sistema de organización, el cual estará inspirado en la Estructura de Desglose de Trabajo (EDT) explicada en la "Guía de los Fundamentos Para la Dirección de Proyectos (guía del PMBOK)" [33] y se define como *"La EDT es una descomposición jerárquica orientada al trabajo que será ejecutado por el equipo de proyecto para lograr los objetivos de este y crear los entregables requeridos*". Por lo tanto, procediendo con la estructura citada, se clasificará el trabajo en Tareas (TR) que se dividirán a su vez en subtareas (SubTR).

La Figura 1, muestra el organigrama de todas las TR y SubTR en las que está dividido este trabajo. Toda la información sobre lo que se desarrolla en cada una de ellas y su duración se encuentra en el "Anexo I. Organización y duración del proyecto" junto con el diagrama de Gantt.

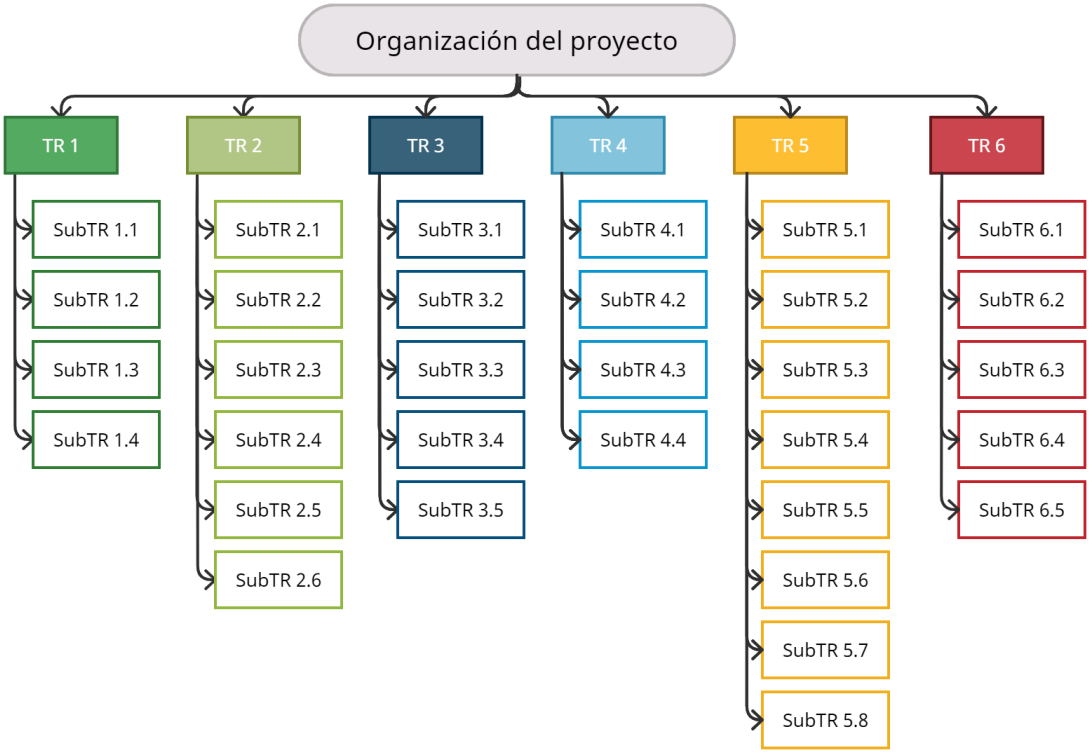


Figura 1. Organización del proyecto. Elaboración propia

3.2 FUNDAMENTOS TEÓRICOS DE LA BANDA ULTRA ANCHA (UWB)

En relación con la TR 1, se va a explicar la tecnología de localización citada en la SubTR 1.1. Para realizar un posicionamiento en interiores son necesarias dos cosas: mediciones y puntos de referencia desde donde se toman estas medidas. Para ello, el método de la trilateración (Figura 2) es el más utilizado en coordenadas bidimensionales ya que mide la distancia desde tres anclajes conocidos, creando tres círculos cuya intersección es el punto de posicionamiento buscado [34].

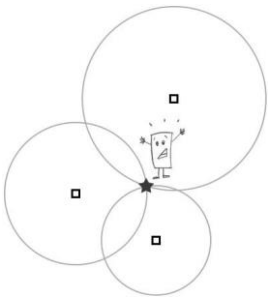


Figura 2. Sistema de medición basado en la trilateración. Cómo funciona el posicionamiento, Pozyx [31].

Para un posicionamiento tridimensional se usa el método de la multilateración que consta de cuatro anclajes. Actualmente, esta tecnología se usa en el Aeropuerto de Asturias [35] para identificar y localizar la posición relativa de las aeronaves. Sin embargo, en un espacio cerrado no se pueden crear círculos, así que lo que se usa para medir la posición entre anclajes son las ondas de radio que se envían de un módulo a otro y se mide el TOF, es decir, lo que tardan en llegar esas ondas [31]. Dado que las ondas viajan a la velocidad de la luz, el cálculo de la distancia sería algo tan sencillo como multiplicar el tiempo de vuelo por esa velocidad, pero esto no es así ya que la precisión que se requiere es centimétrica.

Por todo esto, la forma de ser más precisos es el uso del principio de incertidumbre de Heisenberg [36] que establece que es imposible conocer el tiempo de una señal y la frecuencia de una sinusoide de forma simultánea [37]. Sin embargo, si se combinan múltiples señales con diferentes frecuencias se puede crear un pulso definido tal y como se ve en la Figura 3.

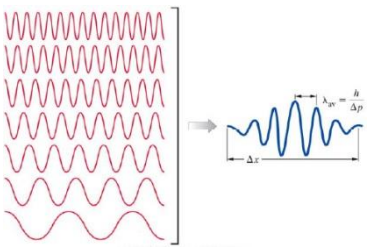


Figura 3. Principio de incertidumbre de Heisenberg. Coloide, Ciencia y Política por Aspirinasindical [37].

Ese conjunto de frecuencias que se usa para determinar el pulso se denomina ancho de banda Δf y, gracias a ello, se puede obtener el ancho de pulso Δt mediante la Ecuación 1:

Ecuación 1. Cálculo del ancho de pulso dado un ancho de banda

$$\Delta f \cdot \Delta t \geq \frac{1}{4\pi}$$

Lo que quiere decir esta fórmula es que para obtener un pulso estrecho con una sincronización muy precisa es necesario un ancho de banda muy grande, por ejemplo, los sistemas WFi usan anchos de banda de 20 MHz obteniendo anchos de pulso de más de 4 ns. A esto hay que añadir la existencia de reflejos, es decir, señales que rebotan en objetos del entorno cuyas frecuencias se pueden superponer a las ondas que provienen de la radio y crear mediciones erróneas. Para evitar esto, la señal de banda tiene que ser aún mayor.

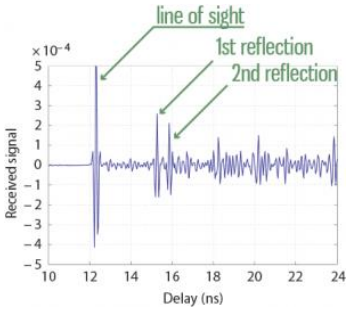


Figura 4. Señal de las ondas y posibles reflejos, Cómo funciona UWB, Pozyx [34].

Es aquí donde aparece la señal de Banda Ultra Ancha que usa el sistema Pozyx [38] ya que su ancho de banda es de 500MHz que dan como resultado un pulso de 0,16 ns de ancho. Es una precisión tan pequeña que el receptor distingue los reflejos que se puedan producir en interiores (Figura 4). Dado que es probable que haya otros sistemas en el entorno que usen este mismo rango de banda, para evitar interferencias la UWB transmite señales a muy baja potencia (por debajo de -41,3 dBm/MHz), que posiblemente sea una potencia tan baja que no sea capaz de llegar al receptor. Es por esto por lo que el transmisor envía un tren de pulsos, que crea un pulso acumulado capaz de evitar el ruido existente u obstáculos, como densidades de humo muy altas [11], convirtiendo a este sistema en un candidato perfecto que admite operar en las situaciones de emergencia correctamente.

Una de las ventajas de esta tecnología es que permite la transmisión de archivos de datos muy grandes a velocidades muy altas en distancias cortas [39]. La velocidad de transmisión que se da en la UWB suele ser de 110 Mbps para una distancia de 10 metros, pudiendo llegar a 480 Mbps para 1 metro [30]. Estas características satisfacen a la mayoría de las aplicaciones multimedia de audio, video y redes inalámbricas en general. Además, otra ventaja es que, dada su baja frecuencia de trabajo el nivel de penetración en objetos opacos es muy alta y su inmunidad a interferencias de otras señales, [16] lo que la convierte en un método óptimo para trabajar en interiores.

3.3 SELECCIÓN DE HARDWARE Y SOFTWARE

Este apartado se corresponde con los objetivos 1 y 2 implementados en las TR 1 y 2, donde se elegirán los sensores y componentes que cumplan los requisitos de los objetivos expresados, las placas de almacenamiento de códigos, el software de la BBDD y los programas para graficar datos. Además, se seleccionarán los programas para el diseño de la simulación del agente autónomo y del entorno que más se ajuste a las necesidades expuestas en la SubTR 5.1.

3.3.1 Sensores y componentes

Como ya se ha mencionado anteriormente, se busca monitorizar datos en situaciones de emergencia, alerta por ambiente tóxico, baja visibilidad por humo producido en incendios, etc. Por lo tanto, se tendrán que buscar sensores que capten los parámetros más relevantes en estas situaciones. El comportamiento de un incendio se ve definido por tres aspectos: las características del combustible, los focos de ignición o parámetros meteorológicos y la ubicación en el terreno [40]. Al centrarnos en espacios interiores como habitaciones, los últimos dos factores tienen un impacto pequeño, por lo que solo se mediría la temperatura para este proyecto, aunque la ventilación puede hacer variar estos parámetros como ocurre en túneles y galerías [41], y habría que hacer un enfoque diferente según el entorno de estudio.

Asimismo, el tipo de combustible hace que la calidad del aire se vea empeorada, así que este será otro parámetro para tener en cuenta. Por lo tanto, ahora se van a seleccionar los sensores que cumplan con los requisitos expuestos en la TR 2.

a) Sensor de temperatura

El sensor digital seleccionado ha sido el HTU21D-F (Figura 5) [42] que mide no solo la temperatura sino también la humedad relativa. La conexión es mediante I2C lo que hace que la obtención de datos sea más rápida y precisa en su envío.

Figura 5. Sensor de temperatura HTU21F-F, Mouser [42]

Cuenta con una librería específica hecha por el fabricante Adafruit [43] para una correcta calibración y lectura de datos, y se implementa en una placa electrónica de desarrollo de hardware libre denominada Arduino [44] que incorpora un microcontrolador.

La temperatura es una magnitud física que determina la energía interna de un cuerpo u objeto y se mide con un termómetro. Los seres humanos se ven afectados de una manera u otra por la temperatura ambiente, aunque hay otros factores influyentes como la edad, actividades a desarrollar o el momento del día. En general, la temperatura ambiente más confortable en reposo oscila entre los 18 ºC y 20 ºC [45]. Con temperaturas superiores, por encima de los 38 ºC en general, los efectos en la salud son varios, desde deshidratación, golpes de calor, síncopes o calambres hasta un agravamiento de posibles enfermedades previas, arritmias o muerte [45]. Para prevenir la aparición de estos efectos es necesario controlar el tiempo de exposición de los bomberos en el ambiente agresivo. En relación con este tiempo de exposición, en la Tabla 1, se encuentran los valores máximos de permanencia cuando existe un ambiente con altas temperaturas [41].

Temperatura ambiente	Tiempo de permanencia
0 ºC – 100 ºC	20 minutos
100 ºC – 160 ºC	15 minutos
160 ºC – 260 ºC	1 minuto
260 ºC – 1000 ºC	0 minutos

Tabla 1. Tiempo de exposición en un ambiente agresivo por temperatura. Elaboración propia

Vistos estos datos y bajo condiciones de trabajo duro y temperaturas elevadas, es aconsejable relevar cada 30 minutos los turnos de trabajo, procurando una frecuente hidratación.

Por humedad se entiende el vapor de agua persistente en la atmósfera y se puede medir mediante la humedad absoluta, que es el peso de vapor de agua por unidad de volumen de aire, y la humedad relativa, que es la cantidad de vapor de agua real que hay en el aire de manera porcentual. Este dato es relevante para las personas, ya que estas sufren un proceso de desprendimiento de calor y vapor por transpiración, que puede verse favorecido o no por la humedad que exista en el ambiente [46] . Unos niveles de entre 40 % y 60 % de humedad relativa son considerados óptimos para el bienestar de las personas. Los valores por debajo de esta franja establecida pueden ocasionar la aparición de virus, bacterias o problemas respiratorios mientras que valores muy altos genera la proliferación de hongos, trastornos alérgicos o moho [47].

b) Sensor de calidad de aire

El sensor digital SGP30 [48] seleccionado (Figura 6), utiliza un algoritmo de compensación de línea base dinámica y parámetros de calibración en el chip para proporcionar dos señales complementarias de calidad de aire. Al igual que el sensor anterior se comunica por I2C y los cuatro parámetros para la medición de la calidad de aire son: eCO_2, TVOC (compuestos

orgánicos volátiles), concentración bruta de H_2 y de etanol. Para el eCO_2 mide un rango de entre 400 ppm – 60.000 ppm y para TVOC el rango es de 0 ppb – 60.000 ppb.

Figura 6. Sensor de calidad de aire SGP30, Mouser [48].

También cuenta con una librería [43] para el correcto funcionamiento de sensor instalado en la placa Arduino [44] creada por el fabricante Adafruit [49].

El CO_2 no es un gas tóxico que pueda producir envenenamiento en su inhalación [50] ya que es producto de procesos donde tiene lugar la combustión. Los niveles comunes en interiores son de 350 ppm – 500 ppm. Al ser inodoro e incoloro lo que realmente produce es desplazamiento de oxígeno [51], por lo que en altas concentraciones en interiores como 30.000 ppm provoca asfixia. Pero no hace falta llegar a ese nivel de concentración en el aire para notar sus efectos ya que se ha estudiado que, en entornos laborales cerrados como oficinas, con una exposición diaria de 8 horas y una concentración de entre 2.000 ppm - 3.000 ppm provoca jaquecas y dolores de cabeza [52]. En la Tabla 2, se encuentran estos rangos.

Calidad de aire en interiores	Concentración (ppm)
Buena	350 ppm – 500 ppm
Moderada	550 ppm – 800 ppm
Mala	800 ppm – 1200 ppm
Muy mala	> 1200 ppm

Tabla 2. Concentraciones de CO_2 en interiores. Elaboración propia

Los TVOC son sustancias químicas que contienen carbono y contribuyen a una mala calidad en el aire interior ya que tienen mucha facilidad para convertirse en vapores y gases, volviéndose tóxicos [53]. Los niveles normales suelen encontrarse por debajo de los 120 ppb. En concentraciones superiores de entre 120 ppb y 1.200 ppb comienza a provocar malestar llegando a ser tóxico por encima de 10.000 ppb.

El etanol generalmente es un líquido incoloro, con un olor muy característico y volátil, lo que le hace ser uno de los vapores más pesados del aire [54]. Se suele utilizar como disolvente, por lo tanto, es altamente tóxico y siempre debe ser manejado con un equipo de protección personal. A partir de 1.000 ppm (1883,07 mg/m^3 a 1 atmósfera y 25 ºC) se obliga a llevar un equipo de respiración homologado. Afecta a las células epiteliales respiratorias provocando

dificultad al respirar, fatiga o asfixia. El valor que constituye un peligro para la salud humana es por encima de 3300 ppm (6218,20 mg/m^3 a 1 atmósfera y 25 ºC).

El H_2 es un gas asfixiante extremadamente inflamable que provoca mareo, sueño, vértigo y pérdida de consciencia en concentraciones por encima de 20.000 ppm. Tiene una reacción violenta con el aire, pudiendo explotar en caso de ser mezclado con oxidantes en incendios [55].

Tanto el etanol como el H_2 producen enfermedades pulmonares crónicas que, con un tiempo de exposición elevado, modifican el color en la piel mostrándose azulado [56]. Por eso es importante controlar estos factores en un incendio.

c) Controlador de motor

El componente SKU DRI0002 (Figura 7) seleccionado se utiliza para el control de los motores de una manera más fácil y segura, ya que usa un chip de alta potencia de LGS que permite el accionamiento directo de estos de forma bidireccional [57]. Soporta mayores tensiones debido a que puede disipar mucho calor e incorpora diodos de alta velocidad para su protección.

Figura 7. Controlador de motor SKU DRI0002, Mouser [57].

La parte lógica de entrada admite entre 6 V – 12 V, pero la parte impulsada del voltaje de entrada va de 4,8 V – 46 V. Funciona a 2 A y puede disipar 25 W (75 ºC). Además, tiene incorporado una modulación de ancho de pulso (PWM) que controla la cantidad de energía que se transmite. De este modo los motores siempre van a actuar del mismo modo independientemente de la carga de la fuente de alimentación.

d) Sensor ultrasonidos

El sensor SKU:SEN0304 (Figura 8) seleccionado es un módulo de sensor de distancia ultrasónica que se comunica por I2C, compatible con Arduino [44] y que sirve para detectar la distancia de los objetos que se encuentran alrededor. Tiene un rango de funcionamiento efectivo de 2 cm a 500 cm con una precisión del 1 % y un ángulo de dirección de 60º.

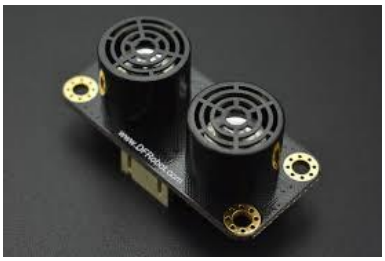


Figura 8. Sensor de ultrasonidos, Mouser [58].

El voltaje de suministro de este dispositivo es de 3.3 V – 5 V con una corriente de funcionamiento de 20 mA y una frecuencia de 50 Hz [58]. Tiene también un sensor de temperatura, pero no se va a usar ya que el sensor citado anteriormente para ello posee mayor precisión que el que este trae incorporado.

e) FLAG de Pozyx

Es un sistema de balizamiento para la ubicación en tiempo real que utiliza la tecnología de UWB (Figura 9). Se compone de cuatro balizas fijas para la pared y una baliza móvil [59].

Figura 9. Sistema de balizamiento Pozyx [60]

Tienen alimentación por USB de 3,3 V, un hardware que se compone de un microcontrolador STM34F4 y un chip DECAWAVE DW1000 que es el que se encarga de enviar y recibir las ondas de alta frecuencia que emiten los dispositivos. La empresa ha desarrollado un firmware que, mediante pulsos, trilateración y TOF comunican las balizas entre sí. Además, son compatibles con Arduino [44] ya que poseen sus propias librerías configuradas para que la obtención de datos sea lo más precisa posible, y la comunicación puede hacerse a través de pines o I2C, lo que facilita la introducción de nuevos sensores sin interferencias con la baliza móvil.

f) Motores

El motor seleccionado es el GM25-370 DC (Figura 10) micro motorreductor de imán permanente, de 12 V y 150 rpm.

Figura 10. Motor GM25-370 DC 12V 2WD, XiaoR Geek [61].

Es totalmente metálico con un bobinado de alambre de cobre puro, lo que disminuye el ruido, da un alto torque, una vida estable y larga, gira en ambos sentidos y tiene un consumo de energía muy bajo [61]. El eje es tipo D así que la instalación es más fácil y evita deslizamientos en el eje de giro respecto a otras piezas. En la Tabla 3, se pueden ven las características completas del motor.

V. funcionamiento	9 V	Juego axial del eje	0,05 mm/ 0,5 mm
Velocidad de salida	150 ±10 RPM	Ruido de carga	56 dB
Corriente sin carga	200 mA (máx.)	Longitud del eje	12 mm, tipo D 8 mm
Corriente a rotor bloqueado	4500 mA (máx.)	Corriente nominal de carga	1200 mA (máx.)
Torque de parada	9,5 Kg/cm	Tamaño del tornillo	M 3,0
Velocidad nominal	100 ± 10 %	Diámetro del eje	4 mm, grosor 3,5 mm
Par de carga	3000 gNaN	Codificador	2 pulsos/círculo

Tabla 3. Especificaciones de los motores. Elaboración propia

g) Chasis

En este caso (Figura 11) es de aleación de aluminio de alta resistencia que proporciona mayor durabilidad, más estabilidad que el plástico y produce poco ruido [62]. Tiene múltiples orificios para poder instalar diferentes componentes sin necesidad de hacer nuevos agujeros. Las orugas denominadas *forerunner* están fabricadas en plástico de ingeniería para una mayor amortiguación, son elásticas y tienen buen agarre tanto en superficies lisas como rugosas. Posee recambios de las orugas y de los tubos de giro de las ruedas.

Figura 11. Chasis HT, XiaoR Geek [62].

El objetivo de haber seleccionado este chasis es que cumple con las expectativas de movimiento del laboratorio LabTel [1], pudiéndose fabricar o modificar nuevas piezas a través de empresas del sector acorde con el prototipo planteado en este proyecto, creándose así módulos independientes en función de las necesidades del momento.

h) Alimentación

Para dar funcionamiento a todo el sistema autónomo de navegación se ha optado por dividir la alimentación. Por un lado, se encuentran tres pilas ion-Litio recargables, de 3,7 V y 12.800 mAh de capacidad cada una, con una protección contra sobrecorriente; miden 65 mm de

altura y 17 mm de diámetro [63]. Estas irán destinadas a alimentar los motores del agente autónomo a través del controlador de motores.

Por otro lado, se encuentra una batería portátil de polímero de litio de 10.800 mAh [64], valor nominal eléctrico de salida de 5 V y un máximo de 2,4 A. También cuenta con una tecnología de seguridad múltiple de protección contra sobrecarga y cortocircuitos. Además, tiene cuatro Leds indicadores de carga en incrementos del 25 % para saber cuánta carga disponible hay. Posee 2 puertos de salida USB, una longitud de 123 mm, anchura de 71 mm, altura de 19 mm y su temperatura de funcionamiento oscila entre 0 ºC y 45 ºC. Esta se encargará de hacer funcionar la parte electrónica del sistema.

Figura 12. Batería Ansmann 10,8 Ah, y pilas recargables 12.800mAh 3,7V [63, 64]

La división de fuentes de alimentación entre los componentes físicos y electrónicos es una manera segura de evitar posibles sobrecargas y cortocircuitos. Además, garantiza una organización en la conexión de los componentes. En el "Anexo III. Autonomía del agente autónomo" se encuentra esta información de forma más detallada.

3.3.2 Adquisición, procesamiento y almacenamiento en la nube

Para este subapartado, se van a presentar las dos placas que se han usado para el control de los sensores y el almacenamiento de datos.

a) Arduino Uno Wifi Rev.2

La placa Arduino (Figura 13) es un componente electrónico de código abierto basada en un hardware y software libre por lo que el tipo de programación es muy accesible para quien quiera utilizarlo. Tiene un microcontrolador Microchip ATmega4809 incluido en un núcleo, con Bluetooth, BLE, WiFi, 14 pines de entrada y salida digital, de los cuales 5 pueden ser usados como PWM, 6 entradas analógicas, conexión USB y botón de reinicio [65].

Figura 13. Placa Arduino Uno Wifi Rev 2, Arduino [65].

En este trabajo se van a incluir dos placas de este tipo, una estará conectada a la baliza Pozyx [60] y a los sensores de temperatura, calidad de aire y ultrasonidos. La otra será la que ejecute el código del movimiento de los motores que enviará la señal al controlador.

b) Raspberry Pi 4

Es un ordenador de placa reducida de bajo coste que da órdenes a la placa Arduino Uno para el control de los motores, recibe los datos obtenidos por los sensores y sirve de comunicación con la base de datos (Figura 14).

Figura 14. Raspberry Pi 4, Página Raspberry [66].

Las características principales de este modelo (Raspberry Pi 4 Model B) son: 4 GB de memoria RAM, chip Broadcom con un procesador 64-bit quad-core Cortex-A72 de cuatro núcleos de 1,5 GHz, dos puertos USB 3.0, 2.0 y micro HDMI, WiFi, Bluetooth 5.0 y un puerto Gigabit Ethernet [66]. Esta unidad es suficiente para el trabajo que se va a desempeñar.

3.3.3 Visualización de datos

Para la visualización de los datos almacenados en la Raspberry Pi, se hace uso de un servidor llamado LAMP [67]. Es un conjunto de tecnologías que sirven para definir la infraestructura de un servidor web, que de manera conjunta desarrollan un sistema. Se compone de:

- Linux: así se denomina el sistema operativo que utiliza el tipo Unix, de código abierto, multiusuario y multiplataforma. Es un conjunto de varios proyectos que permite modificar su código fuente gracias al software libre.

- Apache: es un servidor HTTP de código abierto para las plataformas Unix, Microsoft Windows, etc. que presenta bases de datos configurables. Su tarea es la de realizar peticiones y acceso al sitio web en cuestión y realizar modificaciones.
- MySQL: es el sistema más popular del mundo para gestionar bases de datos por su rapidez y versatilidad en modificación de los diversos campos de la base de datos.
- PHP: es el lenguaje de programación que se usa para hacer desarrollos web.

Para poder visualizar los datos de la BBDD, se va a utilizar una herramienta de software libre llamada Grafana [32]. Posee una configuración multiventana diseñada por el usuario acorde a las características de los datos obtenidos y así poder tener series temporales definidas.

3.3.4 Simulación

Se hace un estudio de las características que presentan diferentes programas de simulación en base a las características la realidad. Aquí se seleccionarán las herramientas elegidas para simular el agente autónomo y el entorno, referidas a la SubTR 5.1.

a) Agente autónomo terrestre

Es necesaria una herramienta de diseño CAD que permita crear con mucho detalle todos los componentes del robot terrestre en 3D, y que su tipo de archivo sea compatible con otros lenguajes de diseño para poder combinarse. A lo largo de la carrera se estudió AutoCAD [68], un software con altas capacidades de edición en dibujo digital no solo para diseños en 2D o 3D, sino también para planos.

Dado que ya se tenían los conocimientos suficientes para el uso de esta herramienta, es la seleccionada para la tarea.

AutoCAD es un software de diseño que ha evolucionado hasta el punto de poder crear y editar geometrías en 2D, modelos de 3D con sólidos, superficies, estructuras, piezas parametrizadas, bocetos y dibujos [68]. Además, se han creado unas variantes destinadas a proyectos de carácter urbanístico, industrial, civil o mecánico.

b) Entorno

Para la creación del entorno existen muchas posibilidades válidas las cuales se desconocen su funcionamiento, por lo tanto, es necesario un tiempo de aprendizaje. El objetivo aquí es conseguir la plataforma que ofrezca la opción más realista en cuanto a simulación del entorno. Lo primero es definir las características que presenta contexto de intervención. Se trata de dos habitaciones diáfanas con paredes y columnas de hormigón, donde hay obstáculos como

colchones, sillas o pallets de madera distribuidos por todo el espacio. En el "Anexo X. Planos" se encuentran las medidas de las estancias a diseñar. Se evalúan las distintas posibilidades.

- Gazebo: es un simulador dinámico 3D que permite realizar simulaciones de robots articulados o entornos realistas [69]. También es un software libre compatible con ROS, por lo que puede ser ejecutable desde esta plataforma.

Al inicio se comenzó a usar Gazebo para la creación de un robot terrestre y entorno. Sin embargo, se vio que el diseño no estaba muy conseguido ya que se creaba con formas geométricas muy simples y la renderización no era la esperada, por eso se decidió buscar otra plataforma de diseño.

- ROS: es una plataforma de software y código abierto que utiliza librerías de programación, monitorización y herramientas de visualización para hacer robots y entornos [70]. Se centra en la creación de paquetes modulares de programación de robots, con la intención de que cualquier persona pueda acceder a esos paquetes ya configurados y modificarlos según sus propias necesidades. No se descarta su uso en un futuro dadas las amplias posibilidades que ofrece respecto a la programación de robots simulados
- Unity: es una plataforma de desarrollo de videojuegos con un software con rutinas de programación que permiten un diseño interactivo de un entorno real. Las características que tiene son muy amplias como sonidos, animaciones, leyes de la física, inteligencia artificial, programación por scripting basada en C# y la posibilidad de crear gráficos en 2D y 3D [71]. Además, otra de las características más importantes es que permite importar archivos 3D de otros lenguajes, lo que le hace compatible con AutoCAD. Por lo tanto, es el programa seleccionado para crear la simulación del entorno.

4. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE MONITORIZACIÓN

4.1 DISEÑO E IMPLEMENTACIÓN DE LA TECNOLOGÍA UWB

En este primer punto, se va a explicar el desarrollo de la TR 1. En 2020, Diego Salas Prieto realizó el TFG en la URJC "Implementación de tecnología UWB para localización y guiado de interiores" [3] donde estudia las posibles tecnologías para la localización de interiores y selecciona la que mejor se adapta al uso en situaciones de emergencia hostiles. Ese Trabajo de Fin de Grado pertenece a la misma línea de investigación que este proyecto.

Tras analizar todas las posibilidades, el autor del trabajo anterior determina que las balizas comerciales Pozyx [60] basadas en la localización UWB [30] son la mejor opción porque tienen mucha precisión en entornos con humo frío y fuego. Este *kit* cuenta con cuatro balizas fijas a las que también llamamos *anchors* y una baliza móvil o maestra llamada *flag*.

4.1.1 Hardware de las balizas

En el Trabajo de Fin de Grado actual, la localización se va a realizar en un espacio tridimensional, por lo tanto, hay que seguir una serie de pautas a la hora de colocar las balizas fijas [72]. Deben ser fijadas en las paredes del recinto donde se va a hacer la localización de la baliza móvil, en una línea de visión cómoda para el usuario, pero cada una debe estar a diferente altura, nunca en la misma línea. También tienen que estar colocadas verticalmente con la antena en la parte superior y el enchufe hacia abajo [73]. La Figura 15, es un ejemplo de posicionamiento:

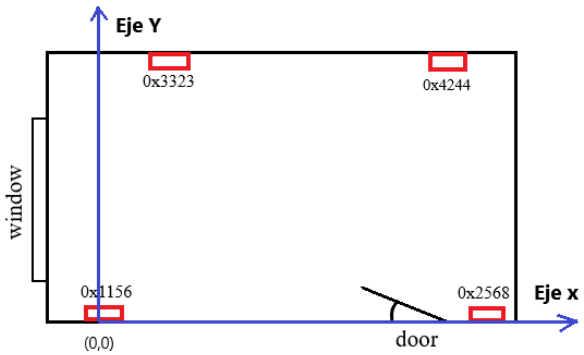


Figura 15. Ejemplo de posicionamiento de las balizas Pozyx, Elaboración propia

Es importante que dos de las balizas se encuentren en la misma pared, y que se tome una de ellas como eje de referencia (0,0) para que las coordenadas del resto tengan coherencia con el resultado de las mediciones.

4.1.2 Software de las balizas

Una vez se haya hecho la instalación, se procede a la configuración del código. En la página web de las balizas Pozyx [59] se pueden encontrar una serie de recursos y documentación para la puesta en marcha de su producto. En uno de los módulos de aprendizaje está la función de posicionamiento que se necesita para este proyecto en el entorno de programación Arduino.

Para empezar a programar es imprescindible descargar las librerías [43] que se indican, que permiten la comunicación entre la Arduino y las balizas; estas son "*Pozyx.h"*, *"Pozyx_definitions.h" y "Wire.h"*. Ahora es necesario declarar las variables que se van a usar a lo largo del código y que van a permitir el cálculo de la posición. "*num_anchors"* contendrá el número de balizas totales que se usa, ya que este es un sistema de multilateración que permite la adaptación a más balizas de posicionamiento. Además, es necesario saber el código de configuración de cada baliza, que se guardará en "*anchors[num_anchors]"* en forma de vector ordenado. Lo siguiente es guardar en tres variables las posiciones *X, Y* y *Z* de las balizas en mm con estructura de vector, con "*anchors_x[num_anchors]"*, "*anchors_y[num_anchors]"* y "*anchors_z[num_anchor]"*. Es importante destacar que el orden de las posiciones *X*, *Y* y *Z* debe ser el mismo que el orden de las balizas, de lo contrario el código mandaría un mensaje de error ya que detecta que los valores obtenidos no son coherentes. Por último, en la variable "*height"* se guarda, en mm, la altura a la que se encuentra la baliza móvil.

En el microcontrolador de la placa de Arduino que se usará, se programa una función de inicialización, *setup,* en la que se inicializan las variables y solo se ejecuta una vez; y un bucle *loop* que es donde se programa lo que se quiere hacer y se ejecuta constantemente [74]. Sin embargo, también se pueden crear funciones individuales con comportamientos determinados que pueden ser llamados desde el bucle *loop* tantas veces como se quiera, sin necesidad de escribir repetidamente la misma orden, manteniendo el código organizado y depurado. Por todo ello se han creado funciones separadas.

- *"Pozyx.begin"*: inicializa la comunicación y funcionamiento de las balizas.
- *"Pozyx.crealDevices"*: elimina los datos anteriores de la memoria de las balizas para iniciar la calibración con los nuevos datos.
- *"Pozyx.setPositionAlgorith"*: indica el tipo de localización que se va a realizar, en nuestro caso 3D.
- *"Pozyx.doPositioning"*: función principal que devuelve la posición de la baliza móvil usando los datos de las balizas fijas y el algoritmo de posicionamiento.

- *"PrintErrorCode"*: imprime por pantalla un mensaje de fallo en caso de que la señal de las balizas tenga algún error de configuración. Crea un bucle que está constantemente comprobando si las señales que llegan son correctas.
- *"PrintCalibrationResult"*: informa del estado de la calibración de las balizas, indicando con un mensaje si es correcta o no.
- *"SetAnchorsManual"*: guarda en variables las posiciones de las balizas facilitadas por el usuario.
- *"PrintCoordinates"*: saca por pantalla la posición en tiempo real de la baliza móvil.

Estas son las funciones más importantes para la localización, su explicación y programación más detallada se encuentra en el "Anexo V. Códigos para el funcionamiento del agente autónomo" y en el repositorio de GitHub [75] "TFG-Noelia-Fernández-Talavera".

4.2 ESTUDIO DEL ANTIGUO MODELO DE AGENTE AUTÓNOMO

Diego Salas Prieto en su TFG [3] también reutilizó un agente autónomo procedente de un compañero anterior a él para comprobar la eficiencia del sistema de localización por UWB [31]. Este modelo constaba de sensores de ultrasonidos para evitar colisiones de forma autónoma, pero el modo autónomo no se utilizaba ya que el autómata presentaba un comportamiento anómalo y no respondía a las necesidades. Además, el movimiento del agente autónomo estaba programado con tiempos de espera calculados visualmente lo que provocaba que a medida que las baterías disminuyeran su carga, el agente autónomo no siempre actuara de la forma esperada.

Debido a esto, se decidió crear un nuevo modelo para poder solventar errores y añadir nuevas necesidades que se habían creado. Se van a enumerar los requisitos:

- Incorporación de un elemento que pueda controlar los motores de un modo más preciso.
- Incorporación de sensores medioambientales.
- Creación de nuevas funciones de movimiento.
- Creación de un modo manual más exacto y eficiente para acortar tiempos de intervención.
- Creación de un modo automático y de evacuación.
- Aumento de la vida funcional del agente autónomo.
- Montaje modular con posibilidad de incluir nuevos elementos en un futuro.
- Creación de un sistema de visualización de datos en tiempo real.

Estas mejoras son la referencia para el diseño del nuevo agente autónomo. Todo esto corresponde a la SubTR 2.1 y 2.2.

4.3 DISEÑO E IMPLEMENTACIÓN DEL NUEVO MODELO DE AGENTE AUTÓNOMO

Este punto corresponde al resto de la TR 2, donde se explica cómo se ha creado el agente autónomo terrestre tanto en hardware como en software. En la Figura 16, se indica el proceso de ensamblaje del autómata.

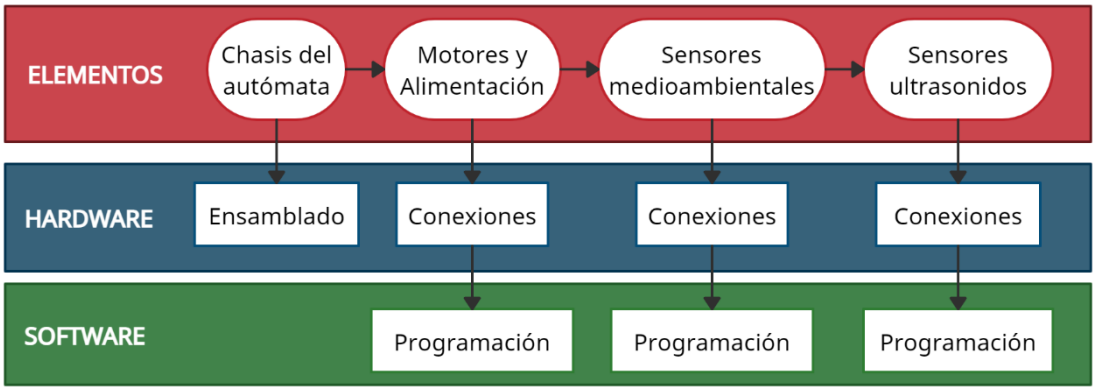


Figura 16. Proceso de creación del agente autónomo. Elaboración propia

4.3.1 Chasis, motores y alimentación

a) Hardware

Lo primero que se hace es el ensamblado del chasis y motores del nuevo agente autónomo especificado en la SubTR 2.3. Su diseño es estéticamente diferente al modelo anterior ya que está compuesto de metal, tiene solo dos motores que mueven dos orugas que se comunican con varias ruedas, mientras que el modelo anterior tenía cuatro motores y cuatro ruedas. El proceso de montaje es el siguiente:

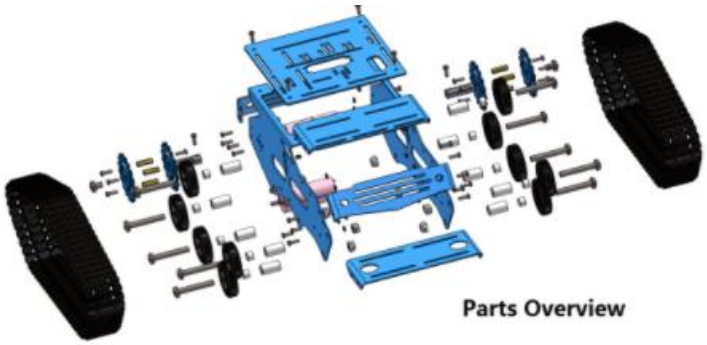


Figura 17. Montaje despiezado del chasis del autómata, XiaoR Geek [62]

Lo siguiente es conectar al controlador de motor la alimentación y la placa Arduino, ya que es el elemento que gestiona el suministro de energía. La alimentación se divide en el control de los motores con pilas recargables y el control de la electrónica con batería de litio, como se ha mencionado en el subapartado “3.3.1. Sensores y componentes”, en la sección “h) Alimentación”.

Los motores se conectan a los pines M1 y M2 del controlador. Las pilas se conectan a los pines VS y GND; y la batería recargable se conecta a VD y GND. Para la placa Arduino se conectan los pines E1, M1, E2 y M2 a 4 pines digitales que se quieran. Por último, se tiene que desconectar el *jumper* del controlador porque es un pasador eléctrico, hace de unión en la alimentación y produciría un cortocircuito en el sistema.

b) Software

Tras el montaje, se procede a programar el movimiento del agente autónomo como se define en la SubTR 2.5 y 2.6. La creación de las funciones de movimiento se ha hecho pensando en la rugosidad del suelo, la resistencia al movimiento que el agente autónomo pueda presentar en diferentes terrenos y en la precisión en el posicionamiento. El código que se ha creado es muy eficiente dado que se puede indicar la velocidad a la que se quiere mover el autómata terrestre, limitado por las *rpm* que permita hacer el controlador. El rango de la velocidad se encuentra entre 0 y 255 *rpm*, pudiéndose modificar de forma sencilla. Las funciones creadas son:

- Adelante: el agente autónomo avanza a 150 rpm hacia delante.
- Atrás: el agente autónomo retrocede hacia atrás a 150 rpm.
- Derecha: el agente autónomo gira a la derecha en el sitio. Se hace moviendo en sentido horario el motor izquierdo y en sentido antihorario el motor derecho, ambos a 200 rpm.
- Izquierda: el agente autónomo gira a la izquierda en el sitio. Se hace moviendo en sentido antihorario el motor izquierdo y en sentido horario el motor derecho, ambos a 200 rpm.
- Aproximación a la derecha: el agente autónomo avanza hacia adelante a la vez que gira ligeramente a la derecha. Esta función es útil cuando se quiere redirigir el autómata hacia la derecha sin necesidad de hacer un gran giro. Para ello ambos motores giran hacia delante pero el izquierdo a mayor velocidad que el derecho.
- Aproximación a la izquierda: el agente autónomo avanza hacia adelante a la vez que gira ligeramente a la izquierda. Esta función es útil cuando se quiere redirigir el autómata hacia la izquierda sin necesidad de hacer un gran giro. Para ello ambos motores giran hacia delante pero el derecho a mayor velocidad que el izquierdo.
- Aproximación detrás a la derecha: el agente autónomo avanza hacia atrás a la vez que gira ligeramente hacia la derecha. Con esta función se puede recolocar la dirección de movimiento del autómata en caso de que no se pueda hacer un giro por encontrarse cerca de una pared.
- Aproximación detrás a la izquierda: el agente autónomo avanza hacia atrás a la vez que gira ligeramente hacia la izquierda. Con esta función se puede recolocar la dirección de

movimiento del autómata en caso de que no se pueda hacer un giro por encontrarse cerca de una pared.

- Parar: los motores se paran.
- Turbo: esta función se ha creado cuando es necesario que el agente autónomo avance con mayor velocidad hacia delante, ya que gira a 250 rpm. Es útil cuando la dirección del agente autónomo no necesita ser modificada y se quiere acortar el tiempo del recorrido.

Estas funciones pretenden abarcar todas las necesidades de movimiento, en todas las direcciones, y con variaciones en la velocidad para ajustarse a los distintos terrenos. De esta forma se tiene el control total del agente autónomo. Con esto ya se pueden programar los tres modos de comportamiento que se requieren: modo manual, modo automático y modo evacuación. Todas ellas son dirigidas por la Rasperry Pi [66], que es el centro de entrada y salida de datos del sistema.

4.3.2 Configuración de sensores

a) Hardware de los sensores medioambientales

En este punto se configuran los sensores medioambientales indicados en la SubTR 2.4. Los sensores de temperatura y calidad de aire usan conexiones I2C, lo que quiere decir que es un bus de comunicación donde se requieren solamente dos cables para el funcionamiento [76]. Cada bus posee una dirección de envío, y la conexión se hace con dos pines, uno para la señal de reloj (CLK) y otro para el envío de datos (SDA). Para realizar las conexiones es necesario incluir unas resistencias de *pull up* de 4k7 entre los pines que transmiten la información y el voltaje (Vcc) para evitar un funcionamiento variable debido a las velocidades y distancias de transmisión.

b) Software de los sensores medioambientales

Para la programación de estos sensores se deben incluir las librerías [43] del fabricante que son "*Wire.h"*, "*Adafruit_HTU21DF.h"* para el sensor de temperatura y humedad, y "*Adafruit_SGP30.h"* para el sensor de calidad de aire.

Tras esto, hay una función para calibrar los datos en función de unos parámetros, porque los valores que leen directamente los sensores no son los definitivos y tienen un formato que hay que convertir. Además, también hay unos bucles para comprobar si el sensor está instalado correctamente o en su defecto aparecerá un mensaje de error.

Y, por último, se crean las variables necesarias para poder almacenar las mediciones de los sensores ya convertidas para su correcto tratamiento: "temp" es la que guarda los datos de temperatura en ºC, "rel_hum" los de humedad relativa dada en %, "TVOC" guarda los datos de los compuestos orgánicos volátiles en ppb, "eCO2" tiene los de dióxido de carbono en ppm y

"Crudo H_2" y "Crudo Ethanol" guarda el crudo de hidrógeno y de etanol respectivamente. El código se explica detalladamente en el "Anexo V. Códigos para el funcionamiento del agente autónomo" y en el repositorio de GitHub [75] "TFG-Noelia-Fernández-Talavera".

c) Hardware de los sensores de ultrasonidos

Estos sensores también tienen conexiones por I2C, lo que implica que tan solo se requieren dos pines para conectar los 6 sensores. Esta comunicación necesita de pocos cables, y en este caso las conexiones se pueden hacer en los mismos pines (CKL y SDA) a los que están conectados los sensores medioambientales, y usar las mismas resistencias de *pull-up*.

Además, para su montaje en el autómata ha sido necesario diseñar un soporte con una impresora 3D [77]. El proceso de diseño y los detalles se encuentran en el "Anexo VI. Simulación del agente autónomo" y el resultado de las medidas y ensamblado en el "Anexo X. Planos".

d) Software de los sensores de ultrasonidos

Para concluir, se configuran los códigos de funcionamiento. Al tener 6 sensores con la misma dirección I2C, hay que cambiar las direcciones para que su lectura no se superponga. A continuación, se incluye la librería correspondiente [43], se declaran identificadores específicos y se inicializan las variables que guardarán la distancia y la dirección de los sensores. Hecho esto, en la función *setup* se abren los puertos de I2C y serial, y se indica el rango de alcance del sensor con los comandos "*MEASURE_MODE_PASSIVE | MEASURE_RANG_500"*. Por último, se crea una variable que vaya leyendo los bytes que recibe el sensor en función de la dirección I2C. En el bucle *loop* que se va a ejecutar de forma cíclica, se llama a los identificadores con los valores leídos y se guardan en las variables declaradas de distancia "dist*"*. El código se explica detalladamente en el "Anexo V. Códigos para el funcionamiento del agente autónomo" y en el repositorio de GitHub [75] "TFG-Noelia-Fernández-Talavera".

4.4 DISEÑO E IMPLEMENTACIÓN DEL ENSAMBLADO DE MÓDULOS

En este punto se va a desarrollar todo lo relacionado con la TR 3. Es necesario incluir el módulo de posicionamiento dentro del agente autónomo y configurarlo con el resto de los componentes para poder crear los modos de movimiento. Para ello es imprescindible saber en todo momento que posición tiene el autómata dentro de una habitación y que obstáculos se puede encontrar a su paso. En la Figura 18, se define qué elementos reciben información desde la Raspberry Pi y cuales le envían información.

La placa Raspberry Pi recibe información constante de la posición del agente autónomo, de los sensores medioambientales y de ultrasonidos, así que se decide unificar todo en una única

placa Arduino para que el envío de datos sea conjunto y se guarden en la base de datos con la misma marca temporal. De este modo se sabe qué condiciones de temperatura y aire hay en cada posición de la habitación, pudiendo identificar zonas peligrosas dentro de la misma.

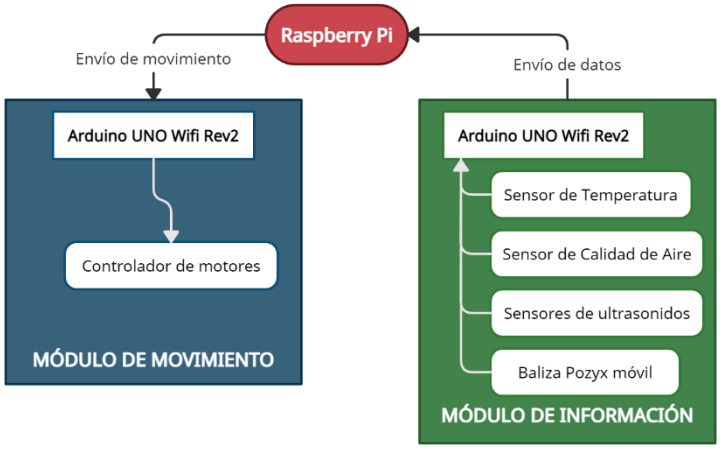


Figura 18. Ensamblado de módulos. Elaboración propia

Por otro lado, se encuentra otra placa Arduino que recibe de manera continuada órdenes de movimiento de la Raspberry Pi y que envía información al controlador para mover los motores, o que necesita coger información de la base de datos sobre la geolocalización. Además, debe comparar la información que están obteniendo los sensores de ultrasonidos sobre las distancias de los objetos que hay alrededor con la posición del agente autónomo para evitar que se choque con paredes u obstáculos

Ambas Arduinos se pueden conectar a la Raspberry Pi que a su vez está alimentada con la batería de litio, por lo tanto, es necesario hacer una serie de cambios en las conexiones y en los códigos de todos los elementos.

4.4.1 Lectura de sensores y tratamiento de datos

En el código de lectura de los sensores se decide crear un solo script con las mismas inicializaciones que se han visto en cada uno de los sensores individualmente en el apartado "4.3.2 Configuración de sensores". El reto aquí es conseguir que todos los datos que se recogen salgan por pantalla a la vez. Para que esto ocurra, hay que meter las funciones de detección ambiental en el mismo bucle que la localización de las balizas ya que si no estos datos nunca serán leídos.

Tras comprobar que el código funciona, se debe crear una única cadena de caracteres o *string* donde se encontrarán todos los datos que se quieren guardar en la base de datos. Esto es necesario porque es el método de comunicación que existe entre el tipo de programación que tiene Arduino y la programación Python, que es la que utiliza la Raspberry Pi para enviar los datos a la base de datos. La cadena o *string* es la siguiente:

"cad" = "#1:" + String(coor.x) + "#2:" + String(coor.y) +"#3:" + String(dato)+".....+@";

Esto se hace así, introduciendo símbolos como # o @ para poder localizar dentro de la cadena el dato que se quiere. Así ya estaría el código preparado para poder comunicar el sistema con la BBDD que se creará más adelante. Este funcionamiento se explicará mejor en el "Anexo V. Códigos para el funcionamiento del agente autónomo" y también se encuentra en el repositorio GitHub [75] llamado "TFG -Noelia-Fernández-Talavera".

4.4.2 Modos de movimiento

En este punto se van a explicar los tres modos de movimiento (manual, automático y evacuación) donde se ve la necesidad de reconfigurar desde cero el movimiento del nuevo agente autónomo respecto al modelo anterior.

a) Modo manual

En este modo se quiere que una persona controle el agente autónomo mediante teclado. Esto se consigue creando una variable en Arduino que guarda la decisión que tome la Raspberry Pi mediante el teclado y crear un bucle para seleccionar la elección del usuario. Las teclas que se usan para mover el agente autónomo son:

Movimiento hacia delante	W
Movimiento hacia atrás	S
Girar a la derecha	D
Girar a la izquierda	A
Aproximación a la derecha	E
Aproximación a la izquierda	Q
Aproximación trasera a la derecha	C
Aproximación trasera a la izquierda	Z
Turbo	T
Parada	X

Tabla 4. Movimiento por teclado del agente autónomo real. Elaboración propia

Lo primero es crear el código de movimiento en Arduino. Se inicializan las variables que mueven los motores y que se corresponden con los pines de conexión de la placa. En la función *setup* de inicialización se indica que esas variables sean de salida. A continuación, se crean funciones independientes de movimiento, donde se establece la velocidad a la que deben girar los motores. Por último, en el bucle *loop* que se repite constantemente, se indica la lectura del puerto de entrada de la Raspberry Pi, guardando en la variable "c" la elección de la tecla seleccionada.

Lo siguiente es crear bucles "*if"* y "*elseif"* donde se encuentra que función de movimiento se ejecuta dependiendo de la tecla pulsada guardada en *"*c*"*.

A continuación, se creará un código en Python que se comunique con el código de Arduino. Para ello, lo primero es indicar que el puerto serie de lectura de ambos sitios funcionen a la misma velocidad, en este caso a 9600 baudios. Ahora, al igual que antes, se crea un bucle donde se guarda la tecla pulsada y con el comando "*arduino.write(comando*)" se envía la elección al código de Arduino. Este código se encuentra explicado de forma detallada en "Anexo V. Códigos para el funcionamiento del agente autónomo" GitHub [75] llamado "TFG -Noelia-Fernández-Talavera".

b) Modo automático

Este modo de funcionamiento consiste en que el agente autónomo debe recorrer el espacio yendo de una baliza hacia otra, haciendo un recorrido cerrado. Para ello se hace uso del sistema UWB, donde se conoce la posición que ocupa cada baliza en la pared y la posición del agente autónomo ya que lleva la baliza móvil instalada. La forma de crear el código es parecida a una máquina de estados, comparando la posición de la baliza móvil con la baliza fija a la que le toque llegar. En el momento en el que las coordenadas de ambas balizas son iguales, teniendo en cuenta un margen de distancia en el rango de aproximación, el autómata se dirige a la siguiente baliza para realizar la misma operación igualando sus posiciones con el resto de *anchors* (Figura 19).

Dado que se necesita la posición del agente autónomo de manera continua, pero este dato se recoge en una placa diferente a la placa que genera las decisiones de movimiento, se sopesan varias opciones para conseguir este dato.

- BBDD: se puede descargar la posición de la BBDD y usarla en el código de movimiento. El problema aquí es que la Raspberry Pi, que es la que maneja la información, está haciendo varias acciones a la vez, y para descargar datos necesita parar el resto de las funciones durante algunos segundos, lo que hace que se pierda información. Esta opción no es válida.
- Conexión directa entre Arduinos: se pueden conectar por I2C o por cable, pero estas tienen un único puerto de conexión que ya está siendo utilizado para comunicarse con la Raspberry Pi, por lo tanto, esta opción tampoco es válida.
- Puerto Serie Virtual: la comunicación será maestro-esclavo, pudiendo crear un puerto serie adicional en la Arduino esclavo para recibir datos sin que éstos se intercepten con el resto de las funciones que se ejecutan. Para ello se incluye la librería de los puertos serie "*SoftwareSerial.h"* y se inicializa un puerto indicando los dos pines que se necesitan

para la conexión. Una vez hecho esto ya se puede obtener la posición del agente autónomo.

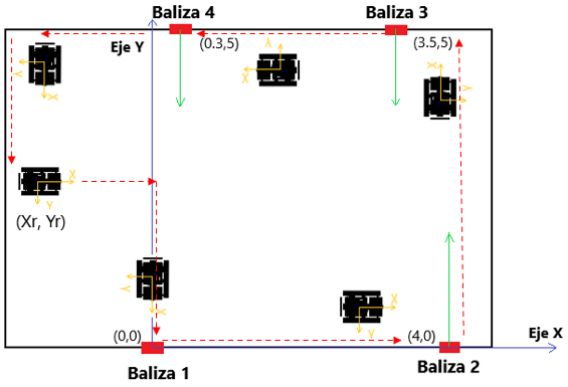


Figura 19. Recorrido modo automático. Elaboración propia

Desde la posición de inicio (Xr,Yr), el agente autónomo compara su coordenada Xr con la coordenada X del eje de coordenadas globales, en caso de que sean diferentes se mueve hasta esa posición. Tras esto hace lo mismo comparando la posición Yr para que sea igual que la de la primera baliza de la pared. En este momento, la comparación vuelve a ser la misma, pero con la posición de la baliza 2 y así sucesivamente hasta llegar otra vez al punto de inicio. El código se encuentra más detallado en el "Anexo V. Códigos para el funcionamiento del agente autónomo" y en el repositorio de GitHub [75] llamado "TFG -Noelia-Fernández-Talavera".

c) Modo evacuación

Para este modo, el método de funcionamiento es parecido al anterior, pero teniendo en cuenta sólo la posición del agente autónomo dentro del espacio, y el punto de evacuación al que se debe llegar (Figura 20).

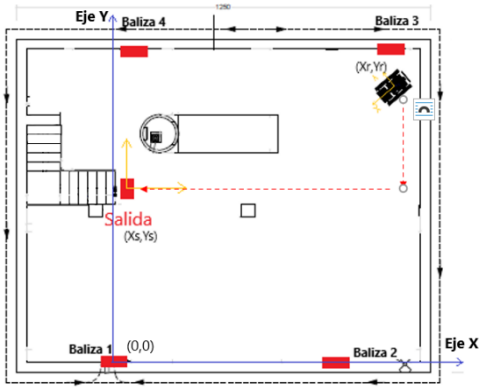


Figura 20. Recorrido modo evacuación. Elaboración propia

Se conocen las coordenadas del agente autónomo (Xr,Yr) y las de la salida de evacuación (Xs,Ys), así que lo primero que hace el agente autónomo es comparar su coordenada Xr con la de la salida, y en función de la lectura actuará, recorriendo la primera parte del trayecto en rojo en la Figura 20. Cuando estos valores sean iguales y dentro de un rango de margen comparará la

coordenada Yr y operará del mismo modo, recorriendo la segunda parte del camino llegando a su destino. El código se encuentra más detallado en "Anexo V. Códigos para el funcionamiento del agente autónomo" y en el repositorio GitHub [75] llamado "TFG -Noelia-Fernández-Talavera".

4.4.3 Estructura final del agente autónomo

Una vez se han explicado los pasos seguidos para el ensamblaje y programación del autómata, se van a enseñar las partes en las que está dividido. Uno de los requisitos del agente autónomo es que fuese modular, con capacidad de incluir elementos nuevos en un futuro si así fuera necesario. Por esta razón, el robot terrestre tiene cuatro niveles tal y como se observa en la Figura 21, cuyos componentes están distribuidos de la manera más cómoda y eficiente posible.

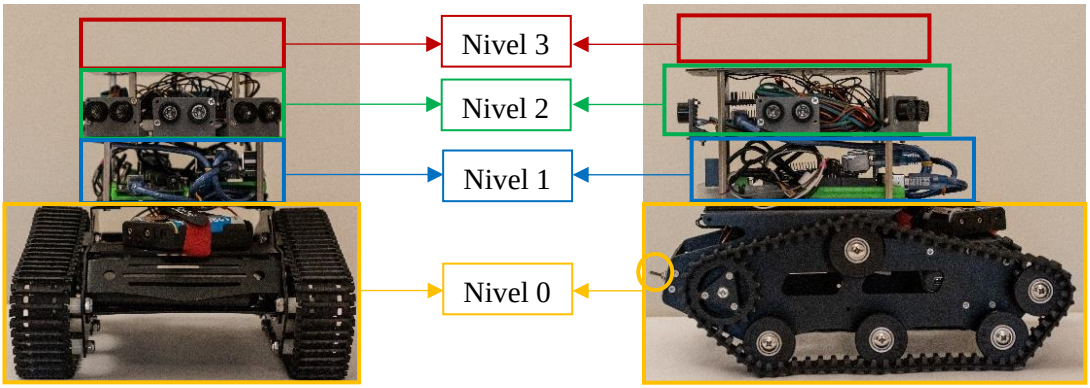


Figura 21. Niveles del agente autónomo. Elaboración propia

- Nivel 0: se encuentra la parte principal del chasis [62], dos motores [61], una batería de litio [64] escondida entre dos placas del propio chasis y tres pilas recargables [63] para un acceso fácil (parte delantera). De esta zona parte toda la alimentación del sistema que se acciona mediante un interruptor que se encuentra en la parte trasera.
- Nivel 1: está instalado el controlador de motores [57], una placa Arduino [65] y la Raspberry Pi [66]. Al ser un nivel tapado, se protege el centro del sistema de control que es la Raspberry y el elemento de control de los motores ya que es muy sensible.
- Nivel 2: en este nivel hay seis sensores de ultrasonidos [58], dos sensores [42, 48] y una placa Arduino [65] conectada a la baliza Pozyx móvil [60]. Este nivel tiene la opción de ser o no tapado, por lo que los sensores que necesitan señales del exterior pueden recibir sin interferencias de otros niveles los datos requeridos. Además, se puede ejecutar de manera independiente al movimiento del agente autónomo.
- Nivel 3: destinado a albergar elementos adicionales como cámara térmica, UDO, otro tipo de balizas como la Pycom o Deep Beacon, etc.

En la Figura 22, se encuentra el autómata montado con todos los niveles completos. El peso total del conjunto se encuentra en "Anexo II. Peso del agente autónomo y rozamiento".

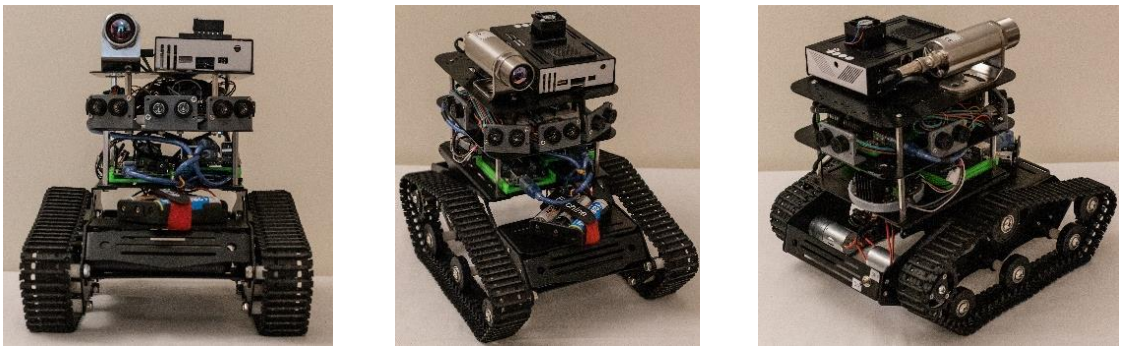


Figura 22. Agente autónomo montado. Elaboración propia

Para ver el funcionamiento conjunto del sistema se desarrolla un diagrama de interacciones de los componentes y el ensamblado que se ha hecho de los módulos (Figura 23).

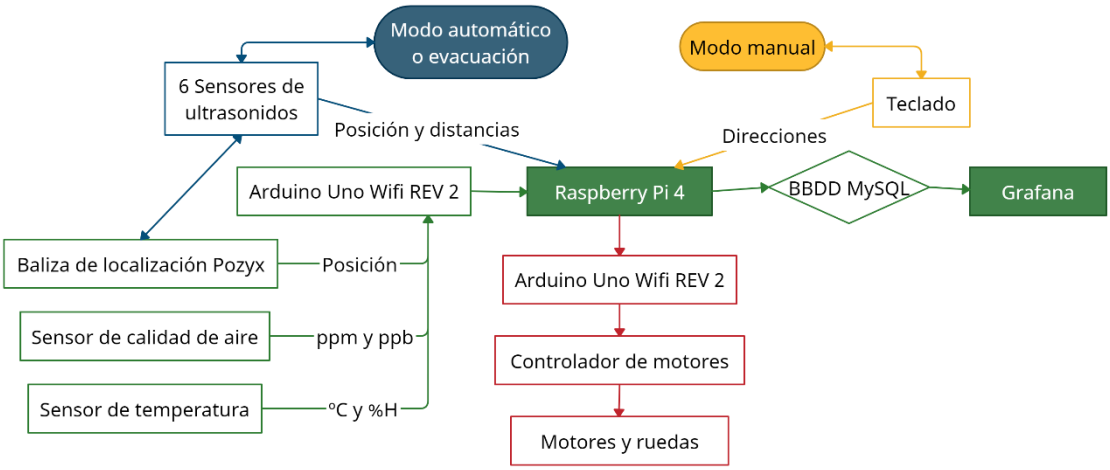


Figura 23. Diagrama de funcionamiento del sistema. Elaboración propia

El camino con los componentes verdes es el de la información, que se está ejecutando constantemente independientemente del modo de movimiento del agente autónomo ya que el sistema es modular. Aquí se recogen los datos de los sensores y se suben a la BBDD para visualizarlos posteriormente.

El camino con los componentes rojos es el del movimiento del autómata, pero en función del modo seleccionado, manual (amarillo) controlado por el teclado, modo automático o evacuación (azul) controlado por los ultrasonidos y la posición, se comportará de una manera o de otra. Una de las ventajas del modo de movimiento del robot terrestre es que se puede replicar el modo automático o de evacuación de forma manual, lo que permite mayor flexibilidad en las pruebas de campo.

Destacar que la Raspberry Pi está ejecutando la subida de datos a la BBDD y el control de movimiento de manera simultánea y de forma constante, con una velocidad de milisegundos, lo que hace que no se pierda ningún dato y los cambios de movimientos sean instantáneos.

4.5 DISEÑO E IMPLEMENTACIÓN DE LA BASE DE DATOS

Esta sección está dedicada la explicación y funcionamiento de la BBDD de la TR 4. La finalidad de tener una base de datos es el almacenamiento de los parámetros de interés para poder acceder a ellos en cualquier momento vía internet. Para eso es necesario tener un servidor que albergue todos los datos, así que se va a crear un servidor con la infraestructura LAMP [67] como se ha visto en el punto "3.3.3. Visualización de datos". En la Figura 24 se encuentra el proceso que sufren los datos desde que son registrados hasta que se visualizan.

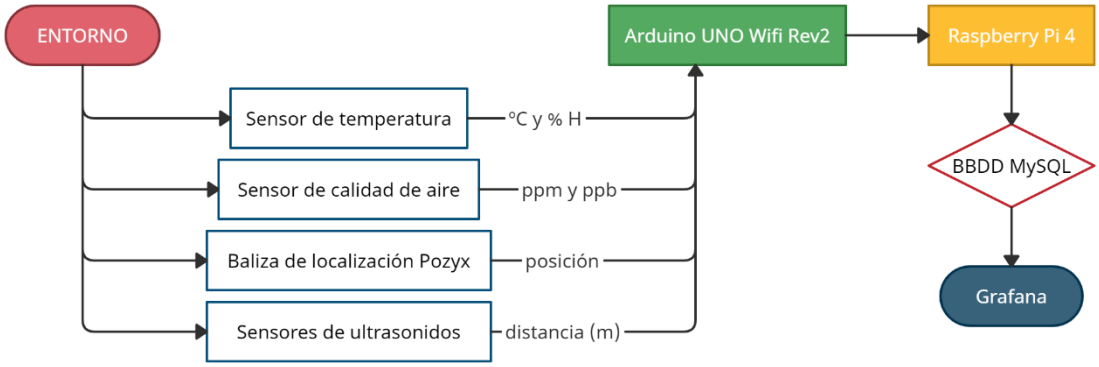


Figura 24. Tratamiento de los datos. Elaboración propia.

4.5.1 Diseño y puesta en marcha de la Base de Datos

El centro de comunicaciones de datos del sistema es la Raspberry Pi y para que esto sea posible es esencial configurarla con todos los elementos necesarios para esta tarea. Lo primero es instalar LAMP [67] con el sistema operativo Raspbian en la Raspberry Pi [66] y el programa Python, indicado en la SubTR 4.1.

Tras esto se crea una base de datos local que solo funcionará mientras esté encendida la conexión a internet, pero que se podrá acceder a ella desde cualquier dispositivo con red a través de la herramienta *phpMyAdmin*. En este momento se debe insertar el usuario y la contraseña, y se debe crear una nueva base de datos a la que habrá que ponerle nombre, para nuestro caso se llama "TFGRobot*"* como se ve en la Figura 25.

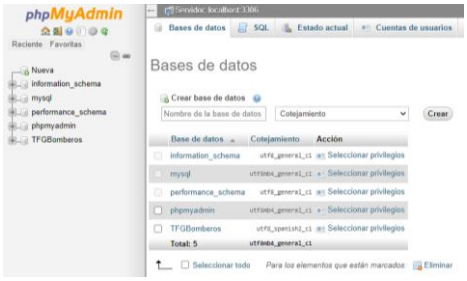


Figura 25. Creación de la BBDD. Elaboración propia

Lo siguiente es crear una tabla (Figura 26) que contenga los datos que se quieren recoger, indicando el tipo de dato que se guardará en la pestaña de "Estructura*"*.

Figura 26. Configuración de la tabla de la BBDD. Elaboración propia

En la Tabla 5, se ven todas las variables creadas, el tipo de variable y lo que registran.

Variable	Tipo de dato	Descripción
Id	int (11)	Número entero de 4 bytes con una longitud de 11 caracteres
coorx	float	Número de coma flotante pequeño, guarda la coordenada *X*
coory	float	Número de coma flotante pequeño, guarda la coordenada *Y*
coorz	float	Número de coma flotante pequeño, guarda la coordenada *Z*
Temp	float	Número de coma flotante pequeño que registra la temperatura
Humedad	float	Número de coma flotante pequeño que registra el % humedad
TVOC	int (8)	Número entero de 4 bytes con una longitud de 8 caracteres que guarda los compuestos orgánicos volátiles
eCO2	int (5)	Número entero de 4 bytes con una longitud de 5 caracteres que registra el eCO_2
H2	int (5)	Número entero de 4 bytes con una longitud de 5 caracteres que registra el crudo de H_2
Ethanol	int (5)	Número entero de 4 bytes con una longitud de 5 caracteres que registra el crudo de etanol
dist	Int (3)	Número entero de 4 bytes con una longitud de 3 caracteres que registra la distancia del sensor delantero
fecha	Timestamp(3)	Marca temporal con precisión de ms que guarda la hora y el día

Tabla 5. Tipos de variables de la BBDD. Elaboración propia

Una vez configurado todo, al conectar la Raspberry Pi y empezarse a subir los datos, en la pestaña "Examinar" se podrá visualizar la fecha y la hora a la que los datos fueros registrados (Figura 27). Esto es importante ya que la marca temporal tendrá que estar bien configurada para que en Grafana [32] se puedan visualizar con la misma fecha y hora. Estos son los pasos a seguir para completar la SubTR 4.2.

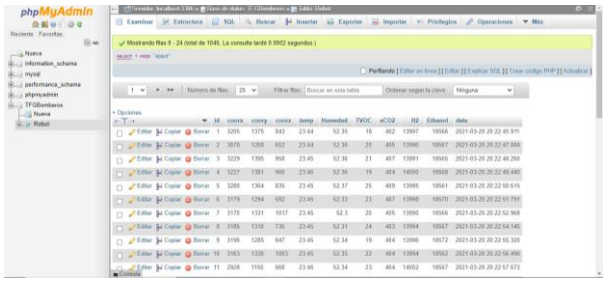


Figura 27. Datos subidos en tiempo real. Elaboración propia

4.5.2 Creación del código de comunicación entre el agente autónomo y la BBDD

En este apartado correspondiente la SubTR 4.3 se va a explicar cómo se crea la comunicación entre la placa de gestión de datos y la BBDD. Como se vio en el apartado "4.4.1 Lectura de sensores y tratamiento de datos", se dejaron los datos preparados para poder subirlos a la BBDD. Hay que crear el código de Python que se ejecuta en la Raspberry Pi para que esto ocurra. En el inicio del código de Python se encuentran las siguientes instrucciones que sirven para conectar el puerto serie de Arduino y Python, y enlazarlo todo con la BBDD MySQL mediante los comandos "*import serial"* e *"import mysql.connector"*. La siguiente instrucción es la que determina a qué velocidad lee la placa Arduino los datos, en este caso son 115200 baudios con el comando *"serial.Serial("/dev/ttyACM0", 115200)*"; por último se crea un vector vacío con el mismo número de datos que se quiere obtener, *"values = [0,0,0,0,0,0,0,0]"*.

A partir de aquí, se desarrollan las funciones que van a comunicar los datos con la base de datos; *"cad_proc(string)"* es una función que se encarga de encontrar los diferentes símbolos citados anteriormente, separa los datos de la cadena y los guarda en otra variable manipulable para este lenguaje de programación. Lo siguiente es tratar los datos obtenidos y los subirlos a la base de datos para su almacenamiento mediante la función "*send_mysql*". Cabe destacar que es necesario incluir el usuario, contraseña, *host* y nombre de la base de datos dentro de las instrucciones de esta función ya que, si no, no sabría donde guardar los datos.

Este funcionamiento se explicará mejor en el "Anexo V. Códigos para el funcionamiento del agente autónomo" que también se encuentra en el repositorio de GitHub [75] llamado "TFG -Noelia-Fernández-Talavera".

4.5.3 Configuración Grafana

A continuación, se completa la SubTR 4.4, de forma que se instala Grafana en la Raspberry Pi, siguiendo las instrucciones que aparecen en su página oficial [78]. Se accede a través de la URL *" http://localhost:3000/"* y se introduce el usuario y la contraseña. Seguidamente, es necesario indicar la base de datos de la que tiene que obtener la información para poder graficar, con la opción "*add data surce"* (Figura 28) donde se abre una nueva ventana. Se deberá rellenar el tipo de base de datos que se va a usar, en nuestro caso MySQL, el nombre de la base de datos que se ha creado, la contraseña y el usuario.

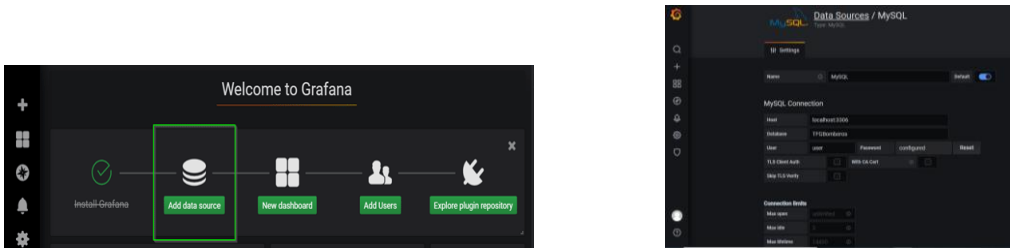


Figura 28. Conexión y configuración de Grafana y la BBDD. Elaboración propia

Ahora es el momento de crear un tablero o *dashboard* donde incluir todas las gráficas que se necesiten. En la parte izquierda de la ventana (Figura 29) se encuentra la opción "*create"* donde se habre un desplegable con la opción *dashboard* que se deberá seleccionar. Seguidamente se abrirá una ventana con la opción *"add new panel"* para ir añadiendo gráficas (Figura 29).

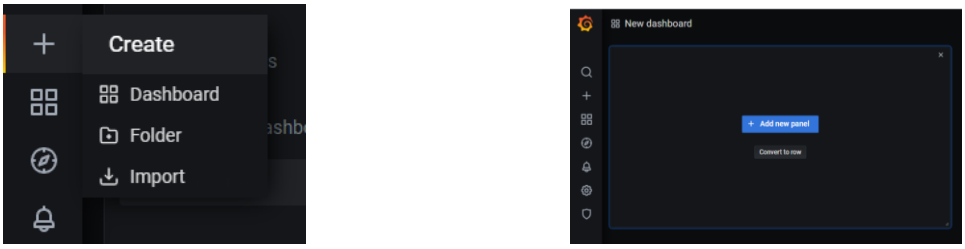


Figura 29. Creación del dashboard. Elaboración propia

La configuración de las gráficas es la que se observa en la Figura 30. Se puede elegir entre muchos tipos de paneles. En la parte inferior es necesario indicar en *"FROM"* el nombre de la BBDD de donde tiene que coger la información. En el eje horizontal denominado "*time column"* se va a graficar el tiempo y será necesario seleccionar "*date"*. Para el eje vertical *"column"* se escogerá el dato que se desee de entre las opciones "eCO2", "temp", " humedad", "TVOC", etc.

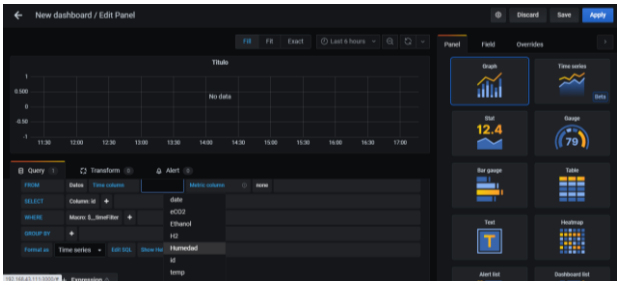


Figura 30. Configuración de un panel del dashboard. Elaboración propia.

Por último, para guardar los cambios se elige la opción de "*apply"*. El resultado final es el que se ve en la Figura 31.

Figura 31. Resultado final del dashboard. Elaboración propia

Grafana tiene múltiples opciones para la configuración de tablas, pudiendo visualizar valores de máximos, mínimos, medias, valor actual he incluso pudiendo hacer rangos de valores de concentraciones con colores para interpretar visualmente en tiempo real y de forma más rápida si los datos obtenidos son tóxicos o normales.

4.6 DISEÑO E IMPLEMENTACIÓN DE LA SIMULACIÓN

Dar apoyo autónomo en situaciones de emergencia mediante un dispositivo móvil que recoja datos de interés del entorno hostil donde las condiciones pueden ser peligrosas para un bombero, y poder anticipar métodos de intervención resulta muy útil para evitar posibles consecuencias graves. En esta línea de investigación en la que se desarrolla este proyecto se encuentran muchas personas realizando sistemas que favorezcan y ayuden al trabajo de los profesionales de emergencia.

Sin embargo, se vio necesario dar un paso más y abrirse a la posibilidad de apoyarse en tecnologías que permitieran crear situaciones críticas para poder anticipar posibles soluciones o métodos de evacuación, esta es la razón de crear simulaciones.

La Jefatura de Bomberos de Alcorcón realiza pruebas de entrenamiento en la torre de bomberos del (CUS), donde existen obstáculos, capacidad de hacer fuego y usan humo frío para simular situaciones de emergencia. No obstante, no tienen la posibilidad de poder hacer pruebas con el robot autónomo ya que constantemente se están implementando mejoras.

Una manera de solventar esta situación es la de replicar en una simulación los espacios que suelen utilizar, con los mismos obstáculos y condiciones físicas que tienen en la realidad, e incluir un robot simulado con las mismas características y funcionamiento que el real. Por eso en los siguientes puntos se va a desarrollar como se ha llevado a cabo la TR 5.

4.6.1 Simulación del agente autónomo

Lo primero de todo es recrear un agente autónomo con los mismos componentes que los que se tienen en la realidad. Como se estudió en el subapartado “3.3.4 Simulación” se ha seleccionado el programa AutoCAD [68] perteneciente a la multinacional Autodesk, un entorno de edición para el dibujo digital capaz de hacer geometrías de alambre, mallas, tridimensional y planos. Este software se encuentra disponible para MAC y Windows y admite muchas interfaces de programación como Visual LISP, NET, etc.

En la Figura 32, se muestra el resultado del diseño para la simulación del agente autónomo con todos los niveles, componentes electrónicos y detalles que tiene el real. Al ser un proceso

muy detallado, se explica todos los pasos que se han seguido para la creación de los componentes y del chasis en el "Anexo VI. Simulación del agente autónomo". Además, en el repositorio de GitHub [75] existe una galería de imágenes de todos los elementos y el montaje.

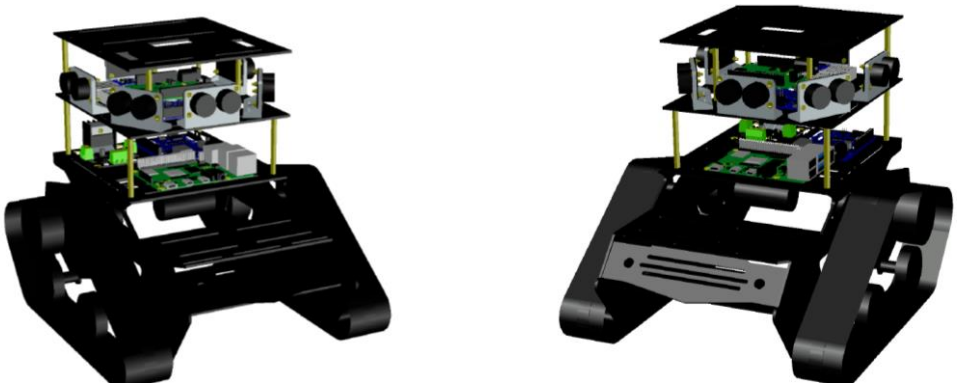


Figura 32. Diseño final del agente autónomo en AutoCAD. Elaboración propia

4.6.2 Simulación del entorno

Como ya se ha comentado también antes en el subapartado "b) Entorno", se va a realizar una simulación de entorno con Unity [71], que incorpora un IDE que integra todas las tareas que están relacionadas con el desarrollo de un videojuego. La lógica del videojuego se hace mediante lenguajes como C#, Javascript o Lua. La torre de bomberos tiene siete plantas, una planta baja y un sótano. Lo primero que se va a hacer es definir qué estancias se van a diseñar. Como las pruebas las suelen realizar en el sótano y la primera planta, esos van a ser los entonos replicados.

En la Figura 33, se encuentra el resultado del diseño del entorno de pruebas realizado en Unity. Dado que la explicación de la creación del entorno y funcionamiento del agente autónomo es muy extensa, se detallan todos los pasos en el "Anexo VII. Simulación del entorno". Además, en el repositorio de GitHub [75] existe una galería de imágenes de todas las estancias y obstáculos diseñados.

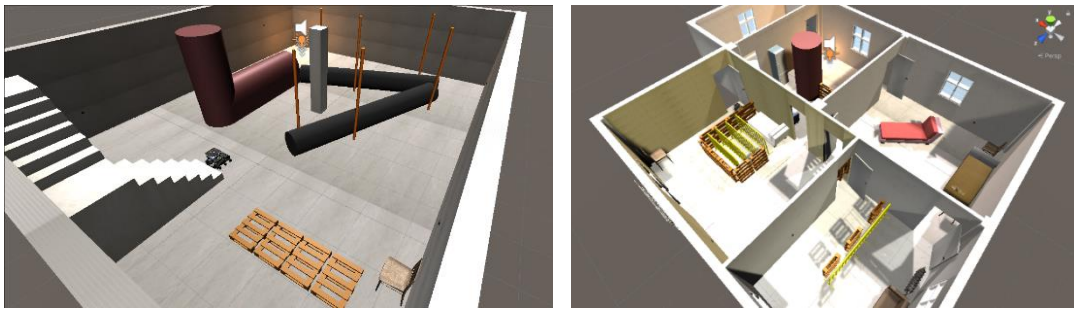


Figura 33. Diseño final del sótano y 1º planta del CUS de la simulación. Elaboración propia

4.7 DISEÑO DE PRUEBAS

Se van a realizar dos pruebas en la torre de bomberos del Centro Unificado de Seguridad de Alcorcón bajo la supervisión de la tutora y dos compañeros del laboratorio SENiaLab [2], donde se comprobará el funcionamiento del agente autónomo en situaciones que replican una emergencia real.

En este apartado se explica en que van a consistir las pruebas, el diseño, montaje de estas y las condiciones en las que se van a realizar.

4.7.1 Diseño de la 1ª prueba en la Torre de Bomberos del Centro Unificado de Seguridad (CUS)

Se programa la primera prueba el día 30 de abril con la finalidad de comprobar la respuesta del sistema de movimiento, sensores y recogida de datos en un ambiente de condiciones normales. Para ello, las pruebas se realizan en el sótano del CUS donde se colocarán las balizas encargadas de la monitorización de la posición del agente autónomo (Figura 34) a lo largo del perímetro del sótano según las posiciones indicadas en la Tabla 6.

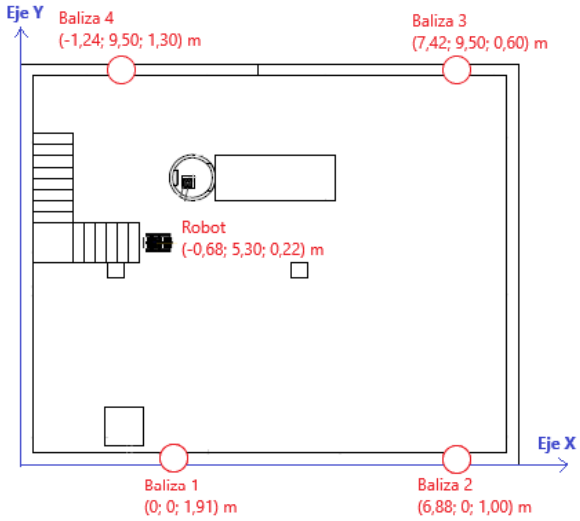


Figura 34. Posicionamiento de las Pozyx en el sótano del CUS. Elaboración propia

Código id baliza	Posición X (metros)	Posición Y(metros)	Posición Z (metros)
0x6e44	0,00	0,00	1,91
0x6e71	6,88	0,00	1,00
0x6e7d	7,42	9,50	0,60
0x6932	-1,24	9,50	1,30
Móvil (robot)	-	-	0,22

Tabla 6. Posición de las Pozyx en las paredes del sótano del CUS. Elaboración propia

Se precede a enumerar los diferentes requisitos que se van a evaluar:

- El área de estudio es de 126,25 m^2, casi cuatro veces superior al recomendado por el fabricante, por lo que se estudiará el alcance del sistema de posicionamiento y la fiabilidad de los datos obtenidos.
- El agente autónomo deberá hacer el recorrido de las balizas, yendo de una a otra por orden, en el menor tiempo posible y siguiendo la trayectoria más directa entre objetivos, volviendo a su posición inicial. Esto tendrá que hacerlo de manera manual y autónoma.
- Se probará el modo evacuación colocando el robot terrestre en cualquier parte del sótano y tendrá que ser capaz de llegar hasta la salida (Figura 35).

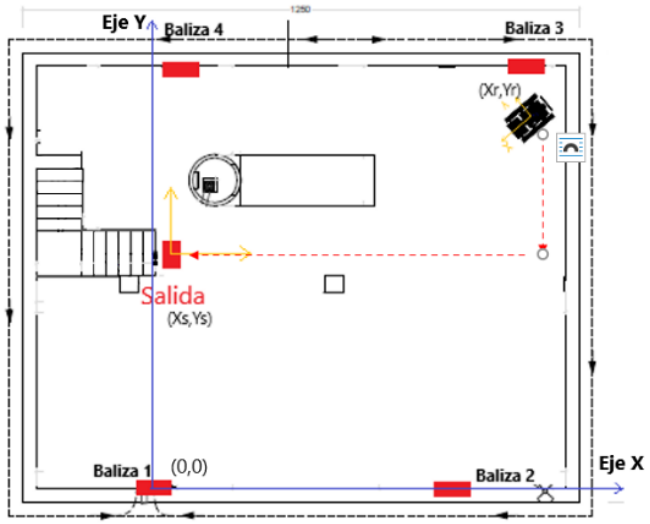


Figura 35. Modo evacuación en el sótano del CUS. elaboración propia

- Además, se comprobará el comportamiento de las orugas del autómata con el suelo para evaluar el rozamiento y velocidad de este.
- Se cotejará si el agente autónomo es capaz de superar obstáculos a diferente altura, rugosidad y elementos cerrados como túneles o tuberías.
- Se estudiará el correcto registro y visualización de los datos ambientales en tiempo real.
- Se comprobará si el peso del agente autónomo excede el límite de carga de los motores y si tiene la autonomía suficiente para realizar una intervención de 3 horas que es la duración media que los bomberos suelen realizar.

4.7.2 Diseño de la 2ª prueba en el Centro Unificado de Seguridad (CUS)

Una segunda prueba es realizada el día 24 de mayo para volver a evaluar el comportamiento del agente autónomo en condiciones reales de emergencia, incluyendo fuego y humo frío.

La Tabla 7, define las posiciones de las balizas en la pared. No están colocadas como el fabricante recomienda (dos balizas en la misma pared) ya que en recorrido que se quiere hacer transcurre a lo largo de cuatro habitaciones, no en una sola como ocurría en la primera prueba.

Código id baliza	Posición X (metros)	Posición Y(metros)	Posición Z (metros)
0x6e44	0,00	0,00	1,82
0x6e71	2,55	2,10	0,63
0x6e7d	5,00	10,10	0,91
0x6932	-4,37	6,64	1,65
Móvil (robot)	-	-	0,22

Tabla 7. Posición de las Pozyx en las paredes de la 1º planta del CUS. Elaboración propia

Además, en la Figura 36, se determina dónde se va a localizar el fuego creado con un hornillo y una sartén, y una máquina de humo frío cuya densidad impide la visión.

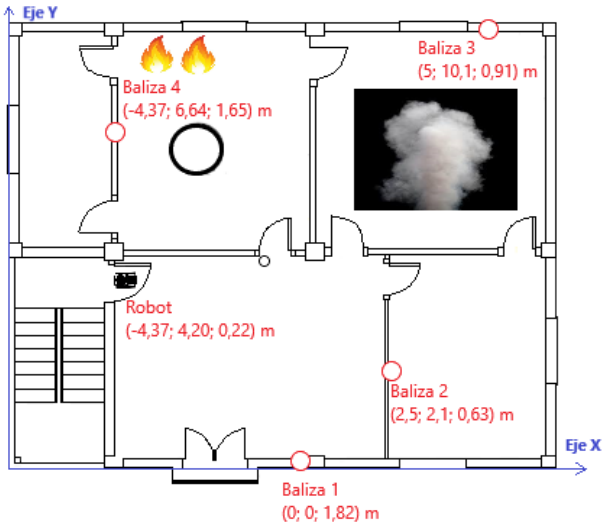


Figura 36. Posicionamiento de las Pozyx en la 1º planta del CUS. Elaboración propia

Esta vez se van a incluir nuevos desafíos:

- El agente autónomo deberá replicar el recorrido de las balizas en la primera planta del CUS donde hay más habitaciones y obstáculos en el camino, comprobando la agilidad y la recogida de datos en tiempo real de los parámetros medioambientales.
- El recorrido deberá hacerse en condiciones de humo frío y fuego.
- Se montará en el último nivel modular del agente autónomo una cámara térmica para comprobar el guiado manual por video, peso y autonomía de este.

5. RESULTADOS

En este apartado, se llevará a cabo la TR 6 donde se expondrán y analizarán los resultados de las pruebas realizadas en dos escenarios diferentes. Aquí se incluirán comparativas de cálculos reales y teóricos, parámetros estadísticos, representaciones de datos y evaluación de similitudes y diferencias de los distintos entornos para poder elaborar unas conclusiones posteriormente. La información del peso del agente autónomo, autonomía teórica, presupuesto de materiales y personal involucrado se encuentran en el "Anexo II. Peso del agente autónomo y rozamiento", "Anexo III. Autonomía del agente autónomo" y "Anexo IV. Presupuesto".

5.1 RESULTADOS DE LA 1ª PRUEBA EN EL CUS

5.1.1 Guiado y trayectoria

Para evaluar la precisión del sistema de posicionamiento se ha realizado un recorrido determinado dentro del sótano del CUS. El agente autónomo debe llegar a las balizas colocadas en la pared, quedándose dentro de un radio máximo de 50 cm de distancia de estas, en el menor tiempo y de la forma más directa posible de manera manual. Se realizaron un total de 7 repeticiones con una media de 1 minuto y 4 segundos de duración.

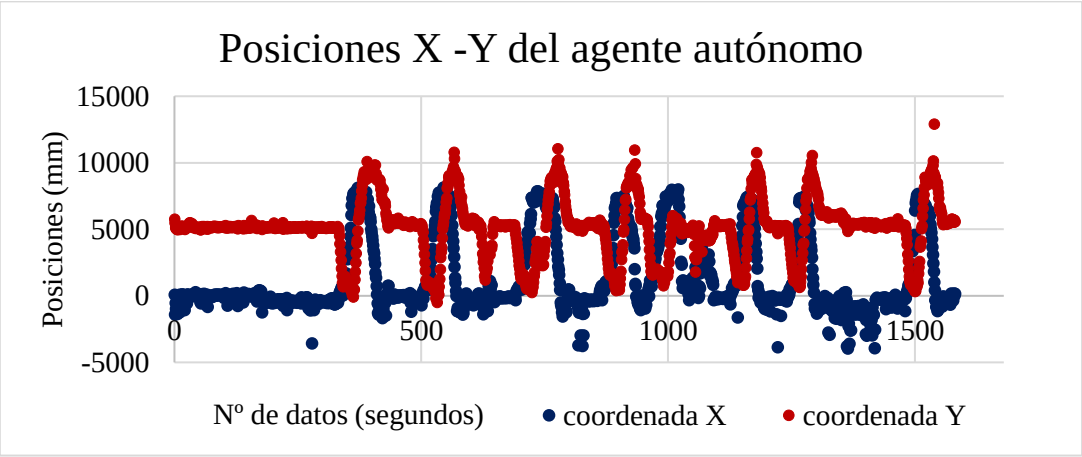


Gráfico 3. Datos de posición del agente autónomo durante las pruebas en el sótano del CUS.

En primer lugar, se van a estudiar los datos de posición recogidos a lo largo de toda la prueba. En el primer tramo del Gráfico 3, el agente autónomo estaba quieto dado que se estaba haciendo un reconocimiento visual del terreno, definiendo las posiciones donde se tendría que parar el autómata; por eso los datos son estáticos. Tras esto se observa como las posiciones se modifican, generándose un mismo patrón que se repite. En algunas zonas el patrón es diferente ya que se adaptó la trayectoria del agente autónomo con el fin de comprobar su funcionamiento con obstáculos.

Para obtener resultados más realistas se han eliminado algunos puntos con datos anómalos y se ha seleccionado la prueba que mejor tiempo y trayectoria registró.

En segundo lugar, se va a estudiar el recorrido que hizo el agente autónomo en el sótano del CUS. En la Figura 37, la trayectoria teórica que el robot terrestre debe hacer es verde y los puntos huecos las posiciones recogidas cada segundo por las balizas Pozyx [60]. El recorrido se hizo en 57 segundos. Los vídeos del recorrido se pueden visualizar en el repositorio de GitHub [75] llamado *"*TFG -Noelia-Fernández-Talavera*"*.

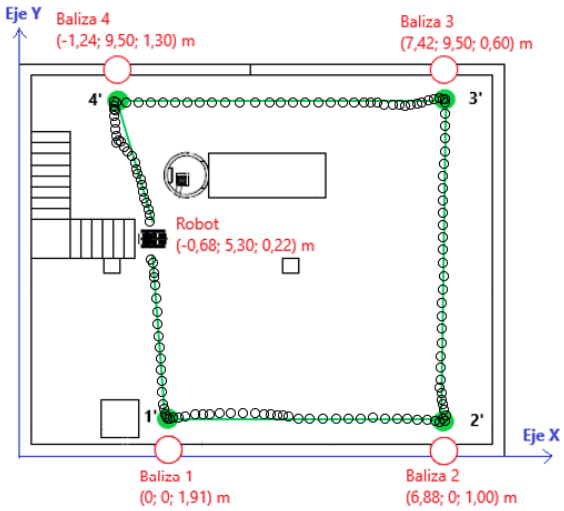


Figura 37. Recorrido teórico y real del autómata en el sótano del CUS. Elaboración propia

Si nos fijamos en el primer tramo (Robot – 1'), el autómata posee un movimiento uniforme y recto en el avance hacia delante, viéndose puntos más separados en la línea recta producido por un aumento de la velocidad, con desviaciones mínimas de 10 cm. Al llegar a la primera baliza, el giro se hace sobre el eje *Z*, registrándose la misma posición con una cierta variación. Esto se muestra en forma de aglomeración de varios datos sobre el mismo punto, hasta llegar al ángulo de giro necesario para poder dirigirse a la siguiente baliza.

Evaluando el inicio del segundo tramo, visualmente y de modo manual es complicado direccionar el agente autónomo en una trayectoria directa hacia el siguiente objetivo. Por eso se hace uso de las aproximaciones creadas, en donde el agente terrestre disminuye su velocidad y gira muy sutilmente hacia donde se requiera. En la Figura 37, se percibe que al salir de cada baliza los puntos se encuentran superpuestos y desviados de la trayectoria teórica, pero una vez encontrada la dirección deseada se vuelve a aumentar la velocidad.

En los tramos siguientes, 2'- 3' y 3'- 4' las trayectorias rectas tienen más distancia, lo que refleja el uso de la máxima velocidad y los redireccionamientos que se realizaron.

El último tramo desde 4’ hasta el punto de partida (Robot), es el que presenta una desviación mayor de hasta 30 cm respecto a la trayectoria real. Es una zona con unas dimensiones menores que en el resto del espacio y con posibles elementos de choque cerca, así que se disminuye la velocidad y se hace uso de las aproximaciones para redirigir al autómata lentamente hacia la posición de partida.

Los modos autónomo y evacuación presentaban un comportamiento anómalo dado que el tiempo de envío de datos no fue lo suficientemente rápido como para que el movimiento fuera el esperado. Por eso, estas pruebas se han podido llevar a cabo haciendo uso del modo manual.

5.1.2 Obstáculos

En este subapartado, se va a explicar el comportamiento del agente autónomo cuando se encuentra obstáculos en el recorrido. Para empezar, en las tres últimas pruebas se colocó una madera de 1 m^2 de superficie y 2,5 cm de altura entre los puntos 2’ - 3’ para comprobar las capacidades de este. Pudo superar el obstáculo sin complicaciones haciendo uso del comando “turbo*”* donde la velocidad de giro del motor es la máxima.

En segundo lugar, se incluyeron piedras para modificar el agarre del suelo entre los puntos 3’ – 4’ a lo que el robot terrestre respondió de forma favorable, enfrentándose las características del terreno y modificando un poco su trayectoria con ciertos tamaños de guijarros, pudiendo ser reconducido posteriormente con el uso de las funciones de aproximación.

Por último, se utilizó un tubo de 0,5 metros de diámetro y 6 metros de largo en el tramo 2’ – 3’. El agente terrestre pudo recorrerlo internamente a máxima velocidad, prosiguiendo con el resto del recorrido sin presentar dificultades.

5.1.3 Monitorización de datos ambientales

En esta subsección, se analizan los datos que han sido almacenados en la BBDD que fue definida en el punto “4.5.1.Diseño y puesta en marcha de la Base de Datos”. Los parámetros recogidos en la BBDD y graficados en tiempo real en Grafana muestran las características del entorno (Figura 38).

El sótano es un sitio cerrado con escasa ventilación. A lo largo de la prueba se mantuvo una temperatura media de aproximadamente 16,25 °C con una desviación de 0,17 °C y una moda de 16,31 °C. Para la humedad relativa el valor medio es de 71,35 % con una desviación del 0,83 % y una moda de 71,77 %. Ese día había llovido horas antes y además en el sótano hay una arqueta de bombero por donde a menudo entra agua llegando a inundar la estancia 1 metro de altura, así que es normal que la humedad se encuentre en valores superiores a los normales.

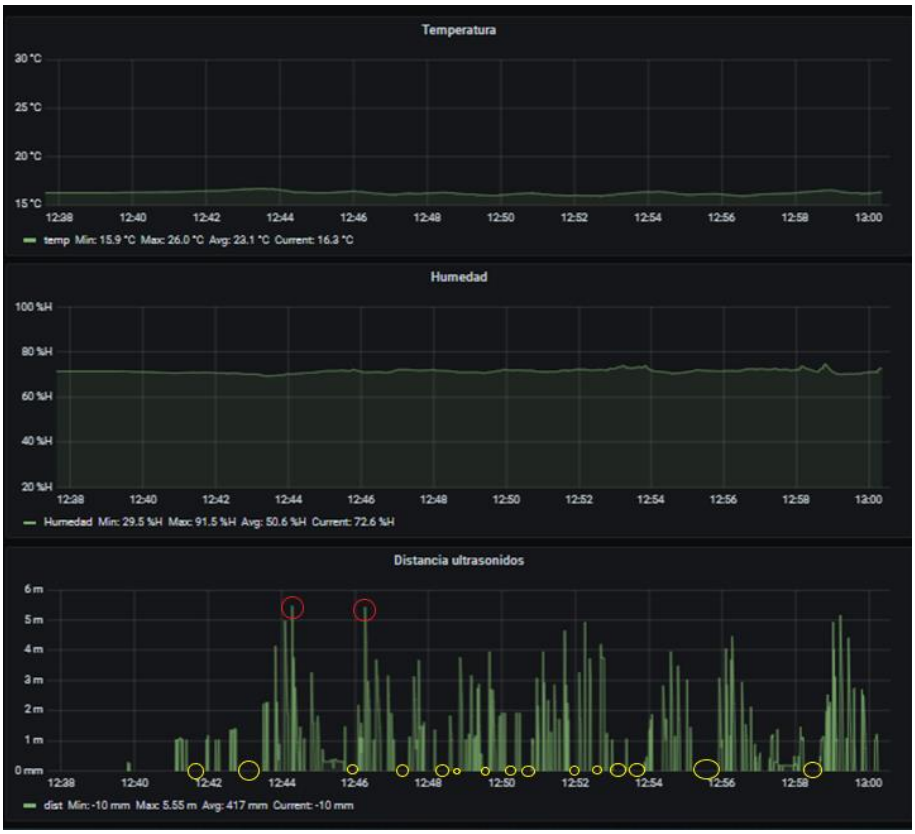


Figura 38. Datos de la 1º prueba (temperatura, humedad y distancia). Elaboración propia

Se ha incluido también la distancia que capta el sensor de ultrasonidos de la parte delantera del agente autónomo para comprobar la existencia de obstáculos frontales. La media obtenida es de 431,81 mm. En este caso existe una variación de datos muy amplia porque el sensor solo tiene un alcance de 500 cm, así que, si la medida al objeto es mayor, el registro es nulo.

En la Figura 39, se encuentran los datos leídos de calidad del aire del sótano. En la parte de la izquierda de esta, se muestran las gráficas que recogen el registro de todo el tiempo que duró la prueba. A la derecha, hay alertas instantáneas en tiempo real que avisan si los valores son peligrosos o superiores a los valores normales.

Para comenzar, se evalúa el parámetro de eCO_2. Los datos que se obtienen son estables con una media de 403,14 ppm, desviación típica de 8,54 ppm y moda de 400 ppm. En general el valor es constante puesto que no se dieron condiciones de combustión.

En segundo lugar, nos fijamos en los valores de TVOC. Son valores pequeños con una media de 18,75 ppb, una desviación de 23,79 ppb y como moda un valor nulo de 0 ppb. Existen picos en la gráfica que indican cuando el agente autónomo se alejaba o acercaba a la única ventilación que existía en la sala. Los resultados muestran condiciones de una sala donde no suele haber tránsito de personas durante tiempos prolongados.

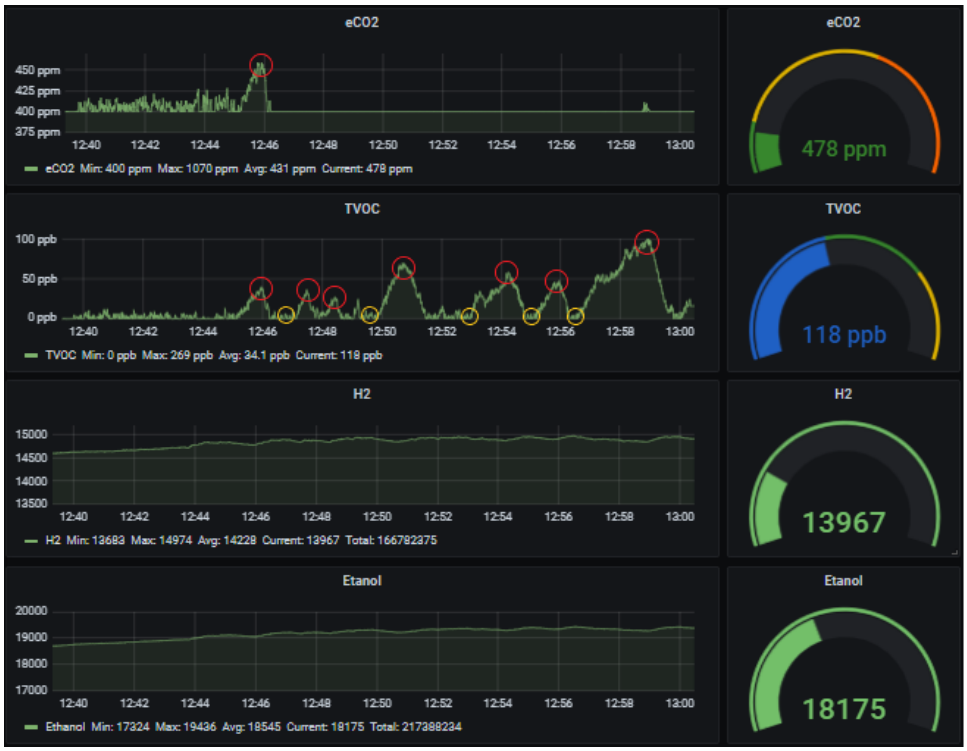


Figura 39. Datos de la 1º prueba de calidad del aire. Elaboración propia.

Por último, se evalúan los parámetros de concentración bruta de H_2 y etanol. Estos valores no se pueden interpretar directamente ya que son mediciones que el sensor toma para realizar los cálculos de eCO_2 y TVOC. Tampoco se puede evaluar si superan o no niveles de riesgo para la salud humana. Sin embargo, si se puede ver la tendencia que tienen para determinar si hay coherencia en la toma de datos. La media de H_2 es de 14.837,72 uds. con la desviación de 102,95 uds. y una moda de 14.932 uds.; y el etanol tiene una media de 19.180,00 ppm, una desviación de 205,17 uds. y moda de 19.344 uds.. Los valores aparentemente son lógicos porque no suele haber personas dentro del sótano. Sin embargo, se ve una ligera tendencia al alza en los datos debido a que durante las pruebas había tres personas en el sótano y se produjeron productos exhalados al respirar. Esto va a permitir generar una referencia en la toma de datos para condiciones donde no hay fuego ni humo.

5.1.4 Autonomía

Al inicio y final de cada repetición se midió el voltaje de las pilas para saber cuánto se consumía en cada recorrido. En la Gráfico 4, se indica la reducción del voltaje de las pilas con cada intento. Se ha sacado la línea de tendencia que define la disposición de los datos y con ello se podrá calcular cuantas veces se puede hacer el mismo recorrido. El valor R^2 que se obtiene del ajuste de la ecuación con los datos es de 0,97, lo que indica que la línea de tendencia se ajusta adecuadamente a los valores recogidos.

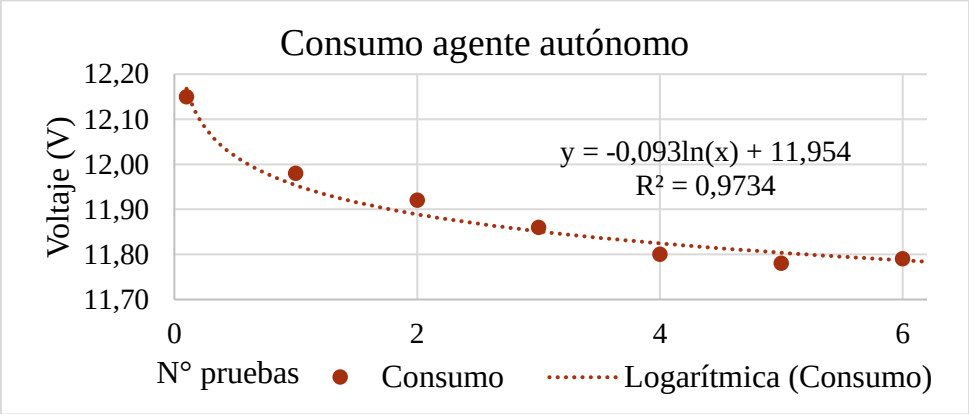


Gráfico 4. Consumo de las pilas en la 1º prueba en el sótano del CUS. Elaboración propia

También se puede calcular cómo el agente autónomo tiene más consumo en este terreno respecto a los cálculos teóricos (Ecuación 2) donde el rozamiento era mínimo.

Ecuación 2. Cálculo de la resistencia e intensidad real del autómata en el sótano del CUS.

$$R_{real}(\Omega) = \frac{V(V)}{I(A)} = \frac{11{,}88\ V}{0{,}505\ A} = 23{,}53\ \Omega \qquad I_{real}(mAh) = \frac{V(V)}{R(\Omega)} = \frac{11{,}88\ V}{21{,}98\ A} = 0{,}540\ mAh$$

Usando el valor real del voltaje y los valores teóricos de resistencia e intensidad se puede comprobar como estos datos son mayores. Sin embargo, estos valores medidos con el multímetro no son fiables del todo porque cada comando de movimiento tiene un consumo distinto y no siempre se usan los mismos durante el mismo tiempo en cada intento. Por eso en el "Anexo III. Autonomía del agente autónomo" se puede encontrar el estudio de autonomía teórico y real medido con un sensor específico para esta tarea.

5.2 Resultados de la 2ª prueba en el CUS

5.2.1 Guiado y trayectoria

Esta segunda prueba se ha realizado en la primera planta del CUS. El agente autónomo debe recorrer el espacio siguiendo el orden de las balizas colocadas en la pared, quedándose a un máximo de 50 cm de ellas y obtener un registro de datos ambientales en presencia de fuego y humo colocados en diferentes habitaciones. Para ello se han tomados las posiciones del autómata manualmente ya que las balizas están colocadas en diferentes habitaciones, y no se cumplen las normas de posicionamiento establecidas por el fabricante. Se realizaron 9 repeticiones, 3 de ellas con fuego y humo, con una media de 2 minutos y 29 segundos de duración. Los vídeos del recorrido se pueden visualizar en el repositorio de GitHub [75] llamado "TFG -Noelia-Fernández-Talavera".

En la Figura 40, se estudia la trayectoria teórica que se debía seguir en color verde, y los puntos huecos son las posiciones reales del agente autónomo. El recorrido tuvo una duración de 2 minutos y 17 segundos, que fue la repetición más rápida y eficiente.

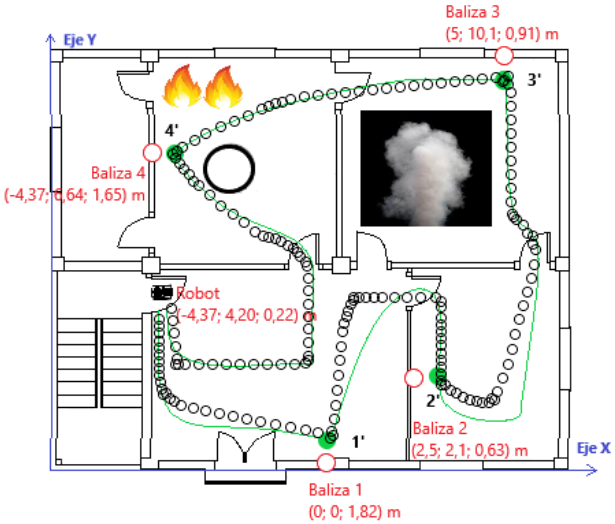


Figura 40. Recorrido teórico y real del autómata en la 1º planta del CUS. Elaboración propia.

Fijándonos en el primer tramo (Robot – 1'), el agente autónomo va a una velocidad lenta donde se ve una aglomeración de puntos, realizando un giro y posteriormente aumentando la velocidad hasta llegar a la baliza en cuestión. Al llegar a la primera baliza, gira sobre el eje Z hasta alcanzar el ángulo de giro para empezar el siguiente tramo. Del punto 1' hasta el 2', el autómata avanza a velocidad rápida hasta encontrarse alineado con la puerta que lleva a la siguiente habitación, gira 90°, avanza rápido y vuelve a girar 90º hasta llegar a la segunda baliza.

El inicio del tramo 2' – 3' es el más complicado ya que hay que girar el robot terrestre 180° para que salga por una puerta hacia la siguiente habitación. Para ello se realiza una aproximación, que permite girar y avanzar el agente autónomo, un giro mayor en el sitio y avanzar a máxima velocidad hasta la puerta. Al entrar en la habitación contigua con humo, el agente autónomo hace una aproximación y avanza rápido hasta llegar a la baliza.

El tramo 3' – 4' es el más sencillo y rápido porque es casi recto. Solo se usa la velocidad máxima y una pequeña aproximación para evitar el fuego que se encuentra al lado derecho del autómata, por eso los huecos están más espaciados que en el resto de recorrido.

El último tramo es el que va desde el punto 4' hasta la posición inicial (Robot). Se realiza un giro en el sitio para orientar el agente terrestre hacia la puerta que lleva a la habitación inicial, y en el trayecto se usan varias aproximaciones para evitar obstáculos. La llegar a la habitación final, se realizan dos giros de 90° grados para dejar a la derecha un obstáculo y se avanza hacia adelante a la máxima velocidad.

5.2.2 Obstáculos

En este apartado, se van a evaluar los obstáculos que se ha encontrado el agente autónomo a lo largo de su recorrido. Principalmente, han sido vallas, pallets de madera, sillones y sillas que se han colocado en zonas estratégicas de las habitaciones para replicar situaciones reales de viviendas. Además, se ha incluido fuego en una habitación y humo frío en otra, que dificultaba la visibilidad dentro de ese espacio. De esta manera se ha comprobado si las funciones de movimiento programadas son útiles.

En primer lugar, los giros en el sitio son idóneos para su uso en espacios reducidos donde el movimiento está limitado. En segundo lugar, el uso de las aproximaciones es útil donde es necesario hacer una ligera corrección en la dirección de avance y dirigirse al objetivo. Por último, la programación de un modo con mayor velocidad permite disminuir el tiempo de intervención y acceder a un objetivo de manera más rápida.

5.2.3 Monitorización de datos ambientales

En este subapartado, se van a analizar los datos medioambientales almacenados en la BBDD y graficado en tiempo real por la herramienta Grafana [32] que muestran las características del entorno de pruebas.

La primera planta del CUS es un conjunto de habitaciones interconectadas con ventilación en cada habitación, cuyo techo se encuentra a 6,85 metros del suelo donde se ha edificado la torre.

La Figura 41, recoge los datos de toda la prueba. La temperatura media fue de 27,59 °C, con una moda de 28,06 °C y una desviación de 0,99 °C. Se observa un claro aumento de la temperatura de casi 12 °C respecto a la primera prueba ya que en este caso existía un foco de calor constante que aumentaba la temperatura de una de las habitaciones. Sin embargo, este fuego controlado era muy pequeño en comparación con los m^2 disponibles, por lo que el calentamiento de la estancia se producía de forma progresiva tal y como se observa en la primera gráfica, y el valor máximo que se alcanzo fue de casi 30 °C.

Los datos de humedad son de 34,97 % de media, 36,66 % de moda y una desviación de 2,61 %, datos por debajo de los valores recomendados y menores que los de la primera prueba. Esto se debe a que el aumento de la temperatura produce un entorno más seco y menos húmedo, viendo su tendencia a la baja en la segunda gráfica de la Figura 41, donde se alcanza un valor mínimo del 30,57 % de humedad.

Los datos de la distancia de ultrasonidos son muy variables con un amplio rango de datos ya que en este caso había más obstáculos que en la primera prueba. La media se corresponde con

un valor de 1.160,25 mm y una desviación de 1357,00 mm, lo que confirma la variabilidad de los datos. Además, también se aprecian mediciones nulas que indican que existe una medida superior al rango máximo de medición del sensor.

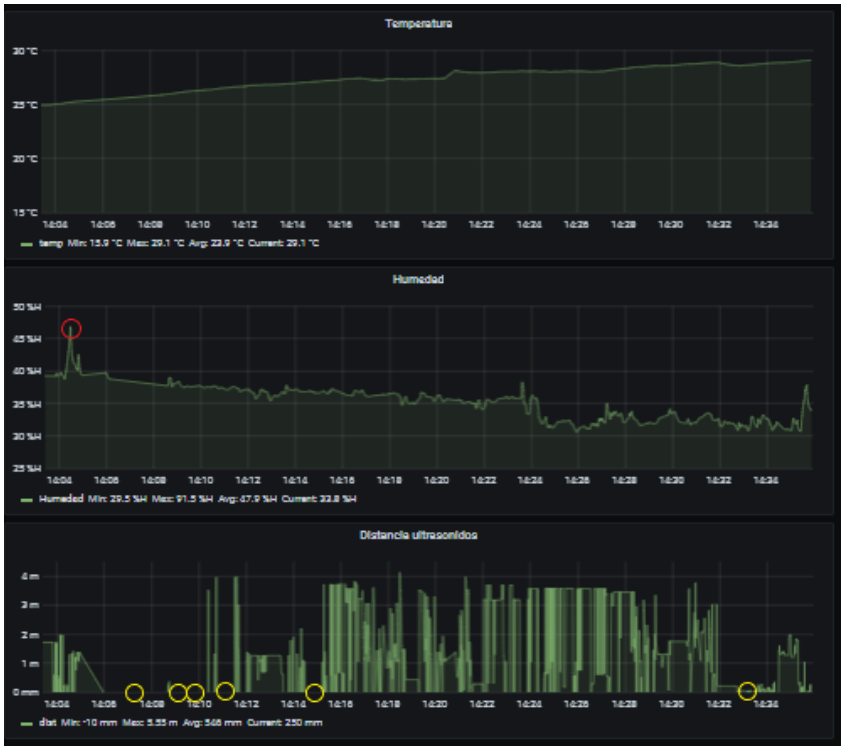


Figura 41. Datos de la 2º prueba (temperatura, humedad y distancia). Elaboración propia

En la Figura 42, se encuentran los datos de la calidad de aire durante la segunda prueba. Las gráficas de la izquierda muestran el registro de datos del tiempo total del ensayo mientras que los indicadores de la derecha son alertas instantáneas que avisan si los valores leídos se encuentran en niveles perjudiciales para la salud.

En primer lugar, se va a evaluar el parámetro de eCO_2. Su tendencia va en aumento mostrando picos en momentos determinados que indican cuando el agente autónomo se encontraba cerca del foco del fuego. La media es de 525,00 ppm con una moda de 400 ppm y una desviación típica de 121,96 ppm. Estos valores son más altos que los recogidos en la primera prueba, pudiéndose comprobar que la medición de este parámetro es útil en presencia de fuego. Además, en varias ocasiones se llegó a superar la concentración admisible en interiores de 550 ppm, llegando al máximo de 1070 ppm.

En segundo lugar, se evalúan los TVOC. Al igual que el parámetro anterior, a medida que discurre el tiempo van aumentando los compuestos orgánicos volátiles en el aire, observándose unos picos en los mismos puntos que donde se producían las concentraciones máximas de eCO_2, que era donde se encontraba el foco de fuego. La media ha sido de 82,76 ppb, con una moda de 0

ppb y una desviación de 60,17 ppb. A pesar de que los valores son aceptables, la desviación indica que el rango de valores es muy amplio, llegándose a alcanzar el valor máximo de 269 ppb perjudicial para la salud durante un tiempo prolongado.

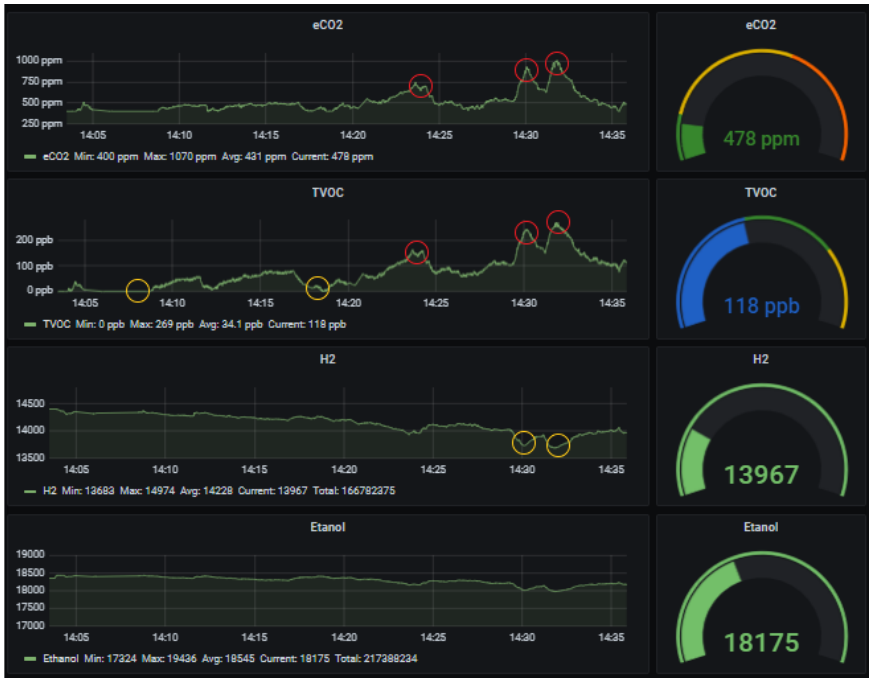


Figura 42. Datos de la 2º prueba de calidad de aire. Elaboración propia.

Por último, se estudian los datos de concentración bruta de H_2 y etanol. Como se ha explicado antes, estos parámetros son necesarios para que el sensor pueda determinar los valores de eCO_2 y TVOC. Aunque estos no se puedan comparar con valores normales o peligrosos para la salud, si se pueden interpretar para verificar una toma de datos realista. El H_2 tiene un valor medio de 14.122,94 uds., una moda de 14.283 uds., y una desviación amplia de 169,02 uds. Este parámetro tiene una tendencia decreciente dado que el en entorno se estaba consumiendo oxigeno debido a la combustión, llegando al valor mínimo de 14.331 uds. Para el etanol se obtiene una media de 18.271,41 uds., una moda de 18.297 uds. y una desviación de 100,82 uds. Este dato se mantiene prácticamente constante, aunque hacia la mitad de esta se observa un ligero descenso, con un valor mínimo de 17.972 uds.

5.2.4 Autonomía

Para esta prueba, no se midió el voltaje de las pilas para comprobar el consumo del recorrido dado que en la primera prueba se observó que este método no era fiable. Por ello, los datos de consumo se encuentran en el "Anexo III. Autonomía del agente autónomo" donde los consumos fueron tomados con un sensor de corriente lineal.

5.3 COMPARATIVA ENTRE ENTORNO SIMULADO Y REAL

En este apartado, se ejecuta el recorrido real en la simulación, colocando las balizas a las mismas distancias que las reales y adaptando la velocidad y rugosidad del suelo para que las condiciones sean similares a las reales. En el repositorio de GitHub [75] se pueden visualizar los vídeos de la simulación. En la Figura 43, se indica el recorrido teórico en verde y real (puntos huecos) que ha hecho el agente autónomo.

Figura 43. Recorrido teórico y real del autómata en Unity. Elaboración propia

a) Similitudes

La primera similitud que se observa es la estética del entorno y agente autónomo. Todos los componentes son similares a los reales, con la mismas dimensiones, colores y respetando el mismo orden en el ensamblaje de estos. También se ha diseñado al robot terrestre con el mismo peso que el real y con el mismo rozamiento que este sufre en el terreno de las pruebas.

La segunda es el manejo del agente autónomo, ya que tanto en el entorno simulado como en la realidad esto se realiza por teclado, y las teclas de movimiento son iguales en ambas situaciones. Además, se han programado las mismas funciones de movimiento en ambos casos, a excepcion de las aproximaciones. Sin embargo, esta posibilidad de realizar aproximaciones existe en la simulación pulsando dos teclas a la vez (delante-derecha o delante-izquierda) para mover en diagonal al automata y así poder redirigirlo.

La tercera es que el agente terrestre gira sobre el eje *Z*, pudiendo girar los mismos grados en la situacion real y simulada. Adicionalmente, se ha incluido la función de una cámara de 360° en la simulacion para que el campo de visión en ambas situaciones sea la misma.

Otra similitud es que se ha incluido un sensor de ultrasonidos en la parte delantera del agente autonomo para que mida la distancia en tiempo real de los objetos que tiene delante. De igual modo se ha incluido fuego en movimiento con sonido para simular un incendio.

Por último, la velocidad del autómata en la realidad y en la simulación se puede modificar para que ambas sean iguales y se puedan ajustar a la situación. De esta forma, las pruebas que se realicen en la simulación para disminuir tiempos de intervencion serán semejantes a los reales.

b) Diferencias

La diferencia principal que existe es el hecho de que todos los comandos de movimiento poseen la misma velocidad en la simulación, por eso los puntos tienen la misma distancia. Además, el giro que se produce del agete autónomo es instantáneo y más rápido que el real.

La siguiente diferencia es que no tiene sensores ambientales, por lo que no se pueden obtener parametros de interés ambiental, aunque si se cuenta con sensores de ultrasonidos. Por último, el fuego no avanza como lo haría en una situación real y no se ha incluido el humo frío.

5.4 PRUEBAS EN EL ENTORNO SIMULADO

En este último subapartado, se han probado en el entorno simulado el resto de los modos (automático y evacuación) para comprobar la eficiencia de la programación simulada y la posibilidad de que sirva de apoyo en situaciones reales. Para ello, se han usado dos escenarios, el sótano y la primera planta del CUS. En el repositorio de GitHub [75] se pueden visualizar los vídeos de las simulaciones

5.4.1 Modo Automático

Para este modo, se ha programado una maquina de estados donde el agente autónomo debe recorrer el entorno de una baliza a otra, esquivando los obstáculos que se encuentra a su paso y encontrando el camino más rápido. Ademas también tendrá que evitar el fuego simulado.

Figura 44. Trayectorias de modo automático en Unity. Elaboración propia

En el sótano la duración del recorrido es de 44 segundos y el de la primera planta es de 1 minuto y 44 segundos. El comportamiento del agente autónomo es preciso aproximandose mucho a las paredes donde se encuentran las balizas y los cambios de direcciones son muy rápidos.

Esquiva los objetos modificando su trayectoria sin que se produzcan colisiones con elementos de alrededor y el movimiento general del autómata es fluido y realista.

5.4.2 Modo evacuación

Este modo consiste en encontrar el camino más seguro y rápido hacia un objetivo, en este caso una persona. Se debe tener en cuenta el fuego y el mobiliario que el robot terrestre se puede encontrar a su paso. Para ello se usa una función de navegación que analiza el espacio disponible por el que se puede mover el agente autónomo, eliminando áreas donde haya objetos, obstáculos y fuego. Esta función es útil para diseñar estrategias donde la minimización del tiempo de intervención es el objetivo principal, además de agilizar el rescate y guiado de las victimas hacia la salida más cercana. En la Figura 45, se puede ver en azul el área que el autómata tiene para llegar hasta el objetivo.

Figura 45. Zonas de navegación en Unity. Elaboración propia

Como se puede observar en la Figura 46, el agente autónomo se dirige hacia la persona por el camino más rápido evitando el fuego. Una de las ventajas que tiene este modo es que se puede modificar la posición del objetivo en cualquier momento, independientemente de si el robot terrestre ha iniciado ya su trayectoria o no. La programación de la simulación permite recalcular en tiempo real la posición a la que se ha de llegar, creando así un nuevo camino hacia el objetivo.

Figura 46. Trayectorias de evacuación en Unity. Elaboración propia

El diseño y la programación del entorno simulado de encuentra en “Anexo VII. Simulación del entorno”, “Anexo VIII. Códigos de Unity C#” y repositorio GitHub [75] llamado “TFG -Noelia-Fernández-Talavera”.

6. CONCLUSIONES

Este Trabajo de Fin de Grado surge de la motivación creada por la línea de investigación de sistemas de localización sensorial y navegación en situaciones de emergencia que dirigen los laboratorios LabTel [1] y SENiaLab [2] del área de Tecnología Electrónica. Se ha propuesto una solución para la monitorización y localización de riesgos ambientales de interiores en situaciones de emergencia mediante la creación de un agente autónomo. También se ha llevado a cabo una simulación del entorno y del robot terrestre para la estimación de tiempos de intervención y creación de recorridos más seguros en entornos hostiles. Para ello se ha colaborado con entidades dedicadas a la Prevención, Emergencias, Seguridad, Intervención y Evacuación como la Asociación Profesional de Técnicos de Bomberos, la Jefatura de Bomberos de Alcorcón y la Agencia de Seguridad y Emergencias de Madrid 112.

En primer lugar, se ha mejorado el anterior sistema de localización y navegación mediante el montaje de un nuevo agente autónomo localizable en tiempo real que monitoriza el escenario de trabajo. Según los resultados, este autómata presenta versatilidad de funciones, velocidades y movimientos que le permiten adaptarse a las características del entorno. Además, se ha disminuido el consumo de sus elementos aumentando así su autonomía de funcionamiento.

En segundo lugar, se ha dotado al sistema de elementos que detectan parámetros ambientales de interés, establecidos por las entidades que participan en el proyecto. Así, se han incluido sensores de temperatura, % de humedad relativa y calidad de aire (eCO_2, TVOC, concentración bruta de H_2 y etanol). Estos datos se han almacenado en una BBDD y han sido representados en una plataforma para visualizarlos en tiempo real. Interpretando los resultados, las mediciones registradas indican las condiciones ambientales propias de incendios, señalando valores críticos que permiten identificar la localización del foco del fuego de manera precisa.

En tercer lugar, y como nueva línea de investigación, se ha diseñado una simulación del agente autónomo y del entorno de intervención para validar el sistema real sin necesidad de contar con él físicamente. Con este fin, se ha replicado el entorno incorporando obstáculos, personas, fuego, texturas y rugosidades iguales a las reales; y se ha programado el autómata con los mismos modos de movimiento, peso y rozamiento que el verdadero. Los resultados muestran cómo el agente autónomo simulado recorre el espacio con el mismo comportamiento que el real, además de encontrar nuevos trayectos más seguros y rápidos para la evacuación de víctimas.

Por último, se está escribiendo un artículo científico en *MDPI Open Access Journals*, *Special Issue* para la divulgación de estas soluciones y resultados obtenidos. Así los equipos de intervención podrán afrontar las situaciones de emergencia de una forma más eficiente y fiable.

7. FUTURAS LÍNEAS DE INVESTIGACIÓN

En este proyecto, se ha abierto la línea de investigación de una plataforma de simulación que sirva de apoyo en intervenciones además del sistema autónomo real. Actualmente, la configuración del entorno real y simulado es diferente ya que los lenguajes de programación no son iguales, y el entorno simulado no tiene la posibilidad de incluir ciertos sensores. Asimismo, el guiado autónomo o la evacuación del autómata real tienen un funcionamiento muy primario, ya que los sistemas de detección de objetos y localización internos no tienen la suficiente velocidad de respuesta necesaria para estas situaciones en las que la información anticipada es esencial para desarrollar una buena estrategia de intervención.

Es por esto por lo que se debería plantear un nuevo sistema de localización de interiores como escáneres 3D, que consisten en la creación de una nube de puntos de cartografía 3D [79], sistemas IMU para determinar la orientación de forma precisa o el uso de cámaras térmicas que distingan personas, obstáculos y fuego. Además, la posibilidad de incorporar apoyo aéreo con drones sería un progreso porque la información obtenida en este ámbito supondría la creación de una intervención más segura y completa [4]. Esto permitiría hacer una monitorización del eje horizontal y vertical, o recogida de información visual de modo aéreo.

La mejora en la simulación también es necesaria, haciendo que el comportamiento del fuego sea dinámico y no estático como hasta ahora, de modo que evolucione en el interior de igual forma que la que se espera en la realidad, atendiendo a los parámetros de evolución externa estudiados.

La demanda constante de simulaciones más realistas con características más amplias ha provocado que Unity haya agregado y mejorado funcionalidades como un *solver used* para motores físicos, robots articulados, prototipos de diseños industriales con movimientos y comportamientos realistas, además de integrar paquetes de comunicación bidireccional con ROS para migrar de forma fácil modelos de ROS a Unity y viceversa [80]. También se pueden generar scripts desde ROS con C# que puedan ser usados en Unity, modificar los sistemas de coordenadas de ambos programas para que exista un entendimiento entre ambos y la existencia de archivos URDF (Formato de descripción de robot universal) que sean compatibles [81]. La gran ventaja del uso de ROS para la simulación del comportamiento del autómata es que es extrapolable al agente autónomo real, siendo su comportamiento el mismo en ambos ámbitos.

8. BIBLIOGRAFÍA

[1] *Laboratorio de Tecnología Electrónica - LabTel - Universidad Rey Juan Carlos* [en línea]. [accedido. 2021-03-12]. Disponible en: https://www.urjc.es/empresas-e-instituciones/949-laboratorio-de-tecnologia-electronica-labtel#ensayos-servicios-que-ofrece

[2] *Laboratorio de Desarrollo de Sistemas de Navegación Sensorial y de Sistemas de Monitorización. (SENiaLab) - Universidad Rey Juan Carlos* [en línea]. [accedido. 2021-03-12]. Disponible en: https://www.urjc.es/empresas-e-instituciones/5877-laboratorio-de-desarrollo-de-sistemas-de-navegacion-sensorial-y-de-sistemas-de-monitorizacion-senialab

[3] SALAS PRIETO, Diego. *Implementación de tecnología UWB para localización y guiado de interiores*. B.m., 2019. Universidad Rey Juan Carlos.

[4] ROLDÁN-GÓMEZ, Juan Jesús, Eduardo GONZÁLEZ-GIRONDA y Antonio BARRIENTOS. A survey on robotic technologies for forest firefighting: Applying drone swarms to improve firefighters' efficiency and safety. *Applied Sciences (Switzerland)* [en línea]. 2021, **11**(1), 1-18. ISSN 20763417. Disponible en: doi:10.3390/app11010363

[5] CASEY C.GRANT. CONCEPTOS BASICOS. En: *Enciclopedia de la salud y seguridad en el trabajo*. sin fecha, p. 41.1-41.31.

[6] *SUBDIRECCIÓN GENERAL DE POLÍTICA FORESTAL Y LUCHA CONTRA LA DESERTIFICACIÓN Centro de Coordinación de la Información Nacional sobre Incendios Forestales*. 2020.

[7] APTB, Fundación MAPFRE. Estudio de víctimas de incendios en España. *Noviembre* [en línea]. 2020 [accedido. 2021-03-15]. Disponible en: https://www.tecnifuego.org/recursos/arxius/20201217_11062020_Estudio_victimas_incendios_en_2019_APTB_y_MAPFRE.pdf

[8] APTB, Fundación MAPFRE. *APTB Victimas de Incendios en España* [en línea]. 2020 [accedido. 2021-03-17]. Disponible en: www.fundacionmapfre.org

[9] INTERVENCIÓN, L A. Peligroso reto para Bomberos de Burgos : camión volcado junto a la vía del tren. 2021.

[10] COTTES. *Robots contra incendios: tecnología e innovación en la lucha contra el fuego y el humo.* [en línea]. 11. junio 2020 [accedido. 2021-03-20]. Disponible en: https://www.cottesgroup.com/blog/robots-contra-incendios

[11] RODRIGUEZ-SANCHEZ, M. Cristina, Luis FERNÁNDEZ-JIMÉNEZ, Antonio R. JIMÉNEZ, Joaquin VAQUERO, Susana BORROMEO y Jose L. LÁZARO-GALILEA. Helpresponder: system for the security of first responder interventions. *Sensors* [en línea]. 2021, **21**(8), 2614. ISSN 14248220. Disponible en: doi:10.3390/s21082614

[12] BELLO SÁNCHEZ, Belén. *Guía visual para bomberos*. 2017.

[13] LEE, Seunghwan, Hanjun KIM, Beomhee LEE, M. Andreasi BASSI, M. A. LOPEZ, L. CONFALONE, R. M. GAUDIO, L. LOMBARDO y D. LAURITANO. *An Efficient Rescue System with Online Multi-Agent SLAM Framework* [en línea]. 2020. Disponible en: doi:10.3390/s20010235

[14] RODRIGO FRANCISCO, Munguía-Alcalá y Grau-Saldes ANTONI. SLAM con mediciones angulares: método por triangulación estocástica. *Ingeniería, Investigación y Tecnología* [en línea]. 2013, **14**(2), 257-274 [accedido. 2021-03-22]. ISSN 14057743. Disponible en: doi:10.1016/s1405-7743(13)72241-8

[15] ESFAHLANI, Shabnam Sadeghi. Mixed reality and remote sensing application of unmanned aerial vehicle in fire and smoke detection. *Journal of Industrial Information Integration* [en línea]. 2019, **15**, 42-49. ISSN 2452414X. Disponible en: doi:10.1016/j.jii.2019.04.006

[16] ZAFARI, Faheem, Athanasios GKELIAS, Kin K. LEUNG, Zafari FAHEEM, Athanasios GKELIAS y Kin K. LEUNG. *A Survey of Indoor Localization Systems and Technologies* [en línea]. B.m.: Institute of Electrical and Electronics Engineers Inc. 2019. ISSN 1553877X. Disponible en: doi:10.1109/COMST.2019.2911558

[17] HERRERA VARGAS, Milan. *Degree Programme in Information Technology | Internet Technology, INDOOR NAVIGATION USING BLUETOOTH LOW ENERGY (BLE) BEACONS*. 2016.

[18] PRIETO, José Carlos, Christophe CROUX y Antonio Ramón JIMÉNEZ. RoPEUS: A new robust algorithm for static positioning in ultrasonic systems. *Sensors* [en línea]. 2009, **9**(6), 4211-4229. ISSN 14248220. Disponible en: doi:10.3390/s90604211

[19] GUALDA, David, María Carmen PÉREZ-RUBIO, Jesús UREÑA, Sergio PÉREZ-BACHILLER, José Manuel VILLADANGOS, Álvaro HERNÁNDEZ, Juan Jesús GARCÍA y Ana JIMÉNEZ. Locate-us: Indoor positioning for mobile devices using encoded ultrasonic signals, inertial sensors and graph-matching. *Sensors* [en línea]. 2021, **21**(6), 1-25 [accedido. 2021-03-23]. ISSN 14248220. Disponible en: doi:10.3390/s21061950

[20] RODELLA, Francesco. *California: robots contra los megaincendios ,Sacyr* [en línea]. 2021 [accedido. 2021-03-24]. Disponible en: https://www.sacyr.com/-/california-recluta-a-robots-y-algoritmos-contra-los-megaincendios

[21] P. COLLINS, Andrew. *Así es Colossus, el robot de bomberos que ha salvado Notre Dame* [en línea]. 17. abril 2019 [accedido. 2021-03-24]. Disponible en: https://es.gizmodo.com/colossus-asi-es-el-robot-de-los-bomberos-de-paris-que-1834102424

[22] CDC. *Prevención de muertes y lesiones de bomberos mediante el uso de principios de gestión de riesgos en incendios de estructuras* [en línea]. 2015 [accedido. 2021-03-24]. Disponible en: www.cdc.gov/niosh

[23] JOSÉ, María y López JACOB. *Enfermedades de los Bomberos. Una revisión de la literatura a demanda de la Federación de Servicios y Administraciones Públicas de CC.OO*. 2004.

[24] FAHY, Rita F, Jay T PETRILLO y Joseph L MOLIS. *Firefighter Fatalities in the United States, 2019*. 2020.

[25] *Wildfire Analyst, Tecnosylva* [en línea]. 2018 [accedido. 2021-03-24]. Disponible en: https://tecnosylva.es/wildfire-analyst

[26] *Tecnosylva, Soluciones avanzadas para incendios forestales* [en línea]. [accedido. 2021-03-25]. Disponible en: https://tecnosylva.es/

[27] *Fire Dynamics Simulator (FDS) and Smokeview (SMV)* [en línea]. [accedido. 2021-04-02]. Disponible en: https://pages.nist.gov/fds-smv/

[28] CAMILO, Andrés, Galvis RODRÍGUEZ y Juan FELIPE GONZÁLEZ GÓMEZ. *SIMPCE (Simulador de Movilidad de Personas en espacios Cerrados)* [en línea]. 2009 [accedido. 2021-04-02]. Disponible en: http://pegasus.javeriana.edu.co/~CIS0910TK01/

[29] BUI, T D, T V HO, Q T HA, Arief RAHMAN, Ahmad Kamil MAHMOOD y Etienne SCHNEIDER. *Using Agent-Based Simulation of Human Behavior to Reduce Evacuation Time*. 2008.

[30] MILLÁN TEJEDOR, Ramón Jesús. Qué es... UWB (Ultra Wide Band). *BIT* [en línea]. 2004, **147**(COIT & AEIT) [accedido. 2021-04-07]. Disponible en: https://www.ramonmillan.com/tutoriales/ultrawideband.php

[31] POZYX. *¿Cómo funciona el posicionamiento UWB?, Pozyx* [en línea]. [accedido. 2021-04-07]. Disponible en: https://pozyx.io/uwb-technology/how-does-positioning-work/

[32] *Características de Grafana® | Laboratorios Grafana* [en línea]. [accedido. 2021-05-12]. Disponible en: https://grafana.com/grafana/

[33] *La Herramienta Esencial para Todo Director de Proyecto* [en línea]. sin fecha [accedido. 2021-04-12]. ISBN 9781628250091. Disponible en: www.PMI.org

[34] POZYX. *¿Cómo funciona la banda ultraancha (UWB)?, Pozyx* [en línea]. [accedido. 2021-04-07]. Disponible en: https://pozyx.io/uwb-technology/how-does-uwb-work/

[35] *ENAIRE pone en servicio el nuevo sistema de vigilancia por Multilateración en el Aeropuerto de Asturias* [en línea]. 2018 [accedido. 2021-04-08]. Disponible en: https://www.enaire.es/es_ES/2018_09_20/ndp_092018_enaire_pone_marcha_sistema_multilateracion_asturias

[36] *El principio de incertidumbre de Heisenberg , Superconductividad (ICMM-CSIC)* [en línea]. [accedido. 2021-04-14]. Disponible en: https://wp.icmm.csic.es/superconductividad/fisica-cuantica-y-transiciones/fisica-cuantica/principio-de-incertidumbre-de-heisenberg/

[37] ASPIRINASINDICAL. *Entendiendo el principio de incertidumbre de Heisenberg (I), Coloide* [en línea]. 1. octubre 2011 [accedido. 2021-04-15]. Disponible en: https://coloide.wordpress.com/2011/10/01/entendiendo-el-principio-de-incertidumbre-de-heisenberg-i/

[38] *Pozyx* [en línea]. [accedido. 2021-04-07]. Disponible en: https://pozyx.io/

[39] SALAZAR, Jordi. *Redes Inalámbricas* [en línea]. 2012. ISBN 9781447152927. Disponible en: http://www3.uah.es/vivatacademia/ficheros/n54/redesinalam.PDF

[40] ANDALUCÍA, Junta de. Guía técnica de seguridad contra incendios. 1377, 68-70.

[41] GARCÍA, Carlos Barberán y J. Andrés VIÉITEIZ MARTÍN. *Incendio en túneles y galerías, Criterios de intervención.* [en línea]. N1º. Madrid: España, 2009. ISBN 978-84-686-9284-5. Disponible en: www.aptb.org

[42] *HTU21D-F Temperature & Humidity Sensor - Adafruit, Mouser* [en línea].

[accedido. 2021-04-20]. Disponible en: https://www.mouser.es/new/adafruit/adafruit-htu21d-f-sensor/

[43] *Instalación de bibliotecas Arduino adicionales, Arduino* [en línea]. [accedido. 2021-04-20]. Disponible en: https://www.arduino.cc/en/Guide/Libraries

[44] *Arduino - Inicio* [en línea]. [accedido. 2021-04-20]. Disponible en: https://www.arduino.cc/

[45] *Efectos en salud relacionados con la temperatura* [en línea]. [accedido. 2021-04-27]. Disponible en: http://www.oscc.gob.es/es/general/salud_cambio_climatico/altas_temperaturas_es.htm

[46] *Ministerio de Sanidad, Consumo y Bienestar Social* [en línea]. [accedido. 2021-04-24]. Disponible en: https://www.mscbs.gob.es/

[47] *¿Cuál debe ser el nivel de humedad ideal en casa?* [en línea]. [accedido. 2021-04-24]. Disponible en: https://www.siberzone.es/blog-sistemas-ventilacion/cual-debe-ser-el-nivel-de-humedad-ideal-en-una-casa/

[48] *SGP30 Air Quality Sensor Breakout - VOC and eCO2 - Adafruit | Mouser* [en línea]. [accedido. 2021-04-20]. Disponible en: https://www.mouser.es/new/adafruit/adafruit-sgp30-sensor/

[49] *Adafruit Industries, kits y electrónicos de bricolaje únicos y divertidos* [en línea]. [accedido. 2021-04-20]. Disponible en: https://www.adafruit.com/

[50] JOSÉ BERENGUER SUBILS LICENCIADA EN CIENCIAS QUÍMICAS FÉLIX BERNAL DOMÍNGUEZ, Mª. *NTP 549: El dióxido de carbono en la evaluación de la calidad del aire interior.* sin fecha.

[51] VALLE, Francisco Alonso. *NTP 340: Riesgo de asfixia por suboxigenación en la utilización de gases inertes*. sin fecha.

[52] *Dióxido de carbono CO2 | Instituto para la Salud Geoambiental* [en línea]. [accedido. 2021-04-24]. Disponible en: https://www.saludgeoambiental.org/dioxido-carbono-co2

[53] Cómo medir la calidad del aire | Soloelectronicos.com. *Soloelectronicos* [en línea]. 21. octubre 2020 [accedido. 2021-04-24]. Disponible en: https://soloelectronicos.com/2020/10/21/como-medir-la-calidad-del-aire/

[54] DEPARTAMENTO DE SALUD DE NUEVA JERSEY. *Hoja informativa sobre sustancias peligrosas*. 2016.

[55] *HIDROGENO (H 2), Extremadamente Inflamable, Dirección General de Protección Civil y Emergencias* [en línea]. [accedido. 2021-05-01]. Disponible en: http://www.proteccioncivil.es/riesgos/quimicos/fichas-toxicologicas/extremadamente-inflamable/-/asset_publisher/O3eU8FspXzkM/content/hidrogeno-h-2?inheritRedirect=false

[56] *Calidad del aire. Datos en tiempo real - Portal de datos abiertos del Ayuntamiento de Madrid* [en línea]. [accedido. 2021-05-12]. Disponible en: https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5

a0/?vgnextoid=41e01e007c9db410VgnVCM2000000c205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD#

[57] *DRI0002 DFRobot, Mouser España* [en línea]. [accedido. 2021-04-20]. Disponible en: https://www.mouser.es/ProductDetail/DFRobot/DRI0002?qs=Zcin8yvlhnNyAlF94b10eg%3D%3D&mgh=1&vip=1&gclid=CjwKCAjw3pWDBhB3EiwAV1c5rAmdBRkywZBnfHB3Q9RKWHbsY4HblBsDuF6fCl1CaQdxoQ1az84EwRoCncAQAvD_BwE

[58] *SEN0304 DFRobot | Mouser España* [en línea]. [accedido. 2021-04-20]. Disponible en: https://www.mouser.es/ProductDetail/DFRobot/SEN0304?qs=2WXlatMagcGrpauaC%252BoNdA%3D%3D

[59] *Soluciones RTLS de banda ultraancha | Pozyx* [en línea]. [accedido. 2021-04-07]. Disponible en: https://pozyx.io/

[60] *Ancla del creador | Pozyx* [en línea]. [accedido. 2021-04-12]. Disponible en: https://store.pozyx.io/product/11-001-0001-creator-anchor-69

[61] *GM25-370 Motor DC 12V 350rpm Micro Motor de engranajes de imán permanente - XiaoR Geek Official Store* [en línea]. [accedido. 2021-05-12]. Disponible en: http://www.xiaorgeek.com/store/robot-accessories/gm25-370-motor-dc-12v-350rpm-permanent-magnet-micro-gear-motor.html

[62] *Kit de chasis de tanque TH multicolor con motores 2WD - Tienda oficial XiaoR Geek* [en línea]. [accedido. 2021-05-12]. Disponible en: http://www.xiaorgeek.com/store/robot-accessories/multi-color-th-tank-chassis-kit-with-2wd-motors.html

[63] *Tienda de componentes electrónicos en Móstoles* [en línea]. [accedido. 2021-05-12]. Disponible en: https://www.amitronica.es/componentes-electronicos

[64] *Power bank Powerbank 10.8, 10.8Ah, 2 puertos USB | RS Components* [en línea]. [accedido. 2021-05-20]. Disponible en: https://es.rs-online.com/web/p/power-banks/1350999/

[65] *ARDUINO UNO WiFi REV2 | Tienda oficial Arduino* [en línea]. [accedido. 2021-05-21]. Disponible en: https://store.arduino.cc/arduino-uno-wifi-rev2

[66] *Comprar una Raspberry Pi 4 Modelo B - Raspberry Pi* [en línea]. [accedido. 2021-05-21]. Disponible en: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[67] RAÍL JESÚS. *¿Qué es LAMP?, Redes* [en línea]. 18. octubre 2017 [accedido. 2021-06-12]. Disponible en: http://rauljesus.xyz/redes/lamp/queEs/

[68] *Software AutoCAD | Obtener precios y comprar software oficial de AutoCAD 2022* [en línea]. [accedido. 2021-05-12]. Disponible en: https://www.autodesk.es/products/autocad/overview?term=1-YEAR

[69] *Kiosko* [en línea]. [accedido. 2021-06-16]. Disponible en: http://gazebosim.org/

[70] *ROS.org | Sobre ROS* [en línea]. [accedido. 2021-06-16]. Disponible en: https://www.ros.org/about-ros/

[71] *Plataforma de desarrollo en tiempo real de Unity | Motor de VR y AR en 3D y 2D* [en línea]. [accedido. 2021-06-12]. Disponible en: https://unity.com/es

[72] POZYX. *¿Dónde colocar los anclajes UWB?, Pozyx* [en línea]. [accedido. 2021-04-07]. Disponible en: https://pozyx.io/uwb-technology/where-to-place-uwb-anchors/

[73] *Tutorial 2: Listo para localizar (Arduino)* [en línea]. [accedido. 2021-04-07]. Disponible en: https://docs.pozyx.io/creator/latest/arduino/tutorial-2-ready-to-localize-arduino

[74] *Referencia de Arduino - Referencia de Arduino* [en línea]. [accedido. 2021-04-20]. Disponible en: https://www.arduino.cc/reference/en/

[75] TALAVERA, Noelia Fernández. *Noelia-vera/TFG-Noelia-Fernandez-Talavera* [en línea]. 2021 [accedido. 2021-06-12]. Disponible en: https://github.com/Noelia-vera/TFG-Noelia-Fernandez-Talavera

[76] *El bus I2C en Arduino* [en línea]. [accedido. 2021-05-20]. Disponible en: https://www.luisllamas.es/arduino-i2c/

[77] *Ultimaker 3* [en línea]. [accedido. 2021-05-20]. Disponible en: https://ultimaker.com/es/3d-printers/ultimaker-3

[78] *Instalar Grafana en Raspberry Pi | Laboratorios Grafana* [en línea]. [accedido. 2021-04-08]. Disponible en: https://grafana.com/tutorials/install-grafana-on-raspberry-pi/

[79] *¿Cómo funciona el mapeo de los UAV 3DLiDAR ? - Recursos y conocimientos | YellowScan* [en línea]. [accedido. 2021-06-18]. Disponible en: https://www.yellowscan-lidar.com/es/knowledge/how-lidar-works/

[80] BARCELO, Christian. *Simulación de Robots en Unity usando Ros , Local User Groups - ROS Spanish Users Group - ROS Discourse* [en línea]. 5. febrero 2021 [accedido. 2021-06-17]. Disponible en: https://discourse.ros.org/t/simulacion-de-robots-en-unity-usando-ros/18821

[81] GREENE, Cameron, Jacob PLATIN, Michael PIÑOL, Amanda TRANG, Vidur VIJ y Sarah GIBSON. *Robotics simulation in Unity is as easy as 1, 2, 3!. Unity Technologies Blog* [en línea]. 19. noviembre 2020 [accedido. 2021-06-18]. Disponible en: https://blogs.unity3d.com/2020/11/19/robotics-simulation-in-unity-is-as-easy-as-1-2-3/

[82] *Sensor de corriente ACS714 - 30A Pololu 1187 | BricoGeek.com* [en línea]. [accedido. 2021-04-20]. Disponible en: https://tienda.bricogeek.com/sensores/367-sensor-de-corriente-acs714-30a.html?gclid=CjwKCAjwgZuDBhBTEiwAXNofRPPzodYQsgkGU-rduoD3fE2bP7izIl9Nk7GrCctXQ1p-ONhkisun1hoCdhwQAvD_BwE

[83] *254 Adafruit | Mouser España* [en línea]. [accedido. 2021-04-20]. Disponible en: https://www.mouser.es/ProductDetail/Adafruit/254?qs=GURawfaeGuAkwqCF4BmPzA%3D%3D&mgh=1&vip=1&gclid=CjwKCAjw3pWDBhB3EiwAV1c5rHSoHYgGiVpBW2-gRpp4D9LLwD-YaYvNDwKyayfUxeJaHQTh-UFLdBoCpGkQAvD_BwE

[84] *SQF-MSDU1-2G-21C Advantech | Mouser España* [en línea]. [accedido. 2021-04-20]. Disponible en: https://www.mouser.es/ProductDetail/Advantech/SQF-MSDU1-2G-21C?qs=sGAEpiMZZMu3sxpa5v1qriqoT2hAgVGPvIZXYrhdGhA%3D

[85] RAFA GARRIDO. Ley de Ohm fundamental en electricidad y electrónica. *ElectroHobby* [en línea]. 2013 [accedido. 2021-04-20]. Disponible en: https://www.electrohobby.org/ley-de-ohm/

ANEXO I. ORGANIZACIÓN Y DURACIÓN DEL PROYECTO

ANEXO I.I ORGANIZACIÓN DEL PROYECTO

Para llevar a cabo este proyecto es necesario definir un sistema de organización, el cual estará inspirado en la Estructura de Desglose de Trabajo (EDT) explicada en la "Guía de los Fundamentos Para la Dirección de Proyectos (guía del PMBOK)" [33] y se define como *"La EDT es una descomposición jerárquica orientada al trabajo que será ejecutado por el equipo de proyecto para lograr los objetivos de este y crear los entregables requeridos*". Por lo tanto, procediendo con la estructura citada, se clasificará el trabajo en Tareas (TR) que se dividirán a su vez en subtareas SubTR.

TR 1: Diseño y desarrollo del módulo IPS. (Duración aproximada de 23 días):

Esta primera tarea está destinada a analizar la validez de la tecnología de la Ultra -Wide Band [30] en el contexto del proyecto. Para ello, se va a hacer uso de unas balizas comerciales basadas en el posicionamiento de interiores. Esta tarea se corresponde con el objetivo 1. Se definen las subtareas:

- **SubTR 1.1** (4 días): Familiarización con la tecnología UWB [34], análisis del funcionamiento de las balizas comerciales Pozyx [38].
- **SubTR 1.2** (8 días): Comprensión de los distintos métodos y códigos de localización. Selección de las partes de interés a incorporar en el proyecto.
- **SubTR 1.3** (3 días): Implementación de posibles mejoras en el código.
- **SubTR 1.4** (8 días): Realización de pruebas de calibración y precisión de parámetros.

Resultados esperados: Obtención de los conocimientos necesarios para poder aplicar esta tecnología, pudiendo hacerse las pertinentes modificaciones para que se adapte al agente autónomo en un entorno con más elementos electrónicos sin errores ni interferencias.

TR 2: Diseño y ensamblado del nuevo módulo móvil/Agente autónomo. (Duración aproximada de 2 meses y 4 días):

Se creará e implementará un nuevo diseño de un agente autónomo a través de un robot terrestre con tecnología de mayores prestaciones, teniendo en cuenta la existencia de un agente autónomo desarrollado en un trabajo de fin de grado anterior, observando las nuevas necesidades que este no podía cubrir. Esta tarea se corresponde con los objetivos 1, 5 y 6. Se definen las siguientes subtareas:

- **SubTR 2.1** (12 días): Estudio de los componentes y funcionamiento del primer modelo de agente autónomo que se desarrolló en un trabajo previo [3], comprensión de los códigos de funcionamiento y análisis de las posibles limitaciones como autonomía, movimiento manual y automático, peso de carga y falta de sensores de adquisición para monitorizar parámetros relevantes del entorno.
- **SubTR 2.2** (10 días): Análisis y toma de decisiones de las posibles mejoras, incluyendo elementos de control nuevos, modificación de la alimentación antigua por otra con mayor capacidad, sensores medioambientales y nueva estructura.
- **SubTR 2.3** (10 días): Diseño y ensamblado del chasis del nuevo agente autónomo, comprobación del funcionamiento de los motores y nueva alimentación.
- **SubTR 2.4** (15 días): Creación y configuración de los códigos de todos los sensores del agente autónomo y posterior comprobación de datos obtenidos.
- **SubTR 2.5** (8 días): Creación y configuración de las funciones de movimiento.
- **SubTR 2.6** (8 días): Instalación del sistema operativo adecuado en el dispositivo que procesará los datos para controlar los códigos de movimiento de manera remota. Creación de un código de conexión entre todos los sistemas de manejo de órdenes y posterior comprobación de un buen funcionamiento. Puesta en marcha del sistema.

Resultados esperados: creación de un agente autónomo operativo que cubra todas las necesidades de movimiento y obtención de variables medioambientales, minimizando tiempos de envío de datos para una respuesta inmediata frente a los estímulos externos.

TR 3: Implementación del módulo IPS en el agente autónomo. (Duración aproximada de 1 mes y 22 días):

El objetivo de esta tarea es unir el agente autónomo con el módulo UWB de posicionamiento para conocer la localización del robot terrestre en todo momento. Además, se reestructurarán los códigos para depurarlos y monitorizar todos los datos deseados. También se diseñará un sistema de conexión y alimentación conjunto para dar servicio a todo el sistema. Esta tarea engloba y mejora los objetivos 5 y 6. Las subtareas son las siguientes:

- **SubTR 3.1** (3 días): Incorporación del módulo UWB en el chasis del agente autónomo, unificando en un único módulo todos los sensores.
- **SubTR 3.2** (15 días): Modificación de los códigos para el tratamiento del envío de datos a un ordenador de placa reducida (Raspberry Pi 4) y posterior almacenamiento.
- **SubTR 3.3** (8 días): Reestructuración de las conexiones de alimentación para evitar posibles sobrecargas.

- **SubTR 3.4** (16 días): Creación y desarrollo de los modos de movimiento del agente autónomo: manual, automático y evacuación.
- **SubTR 3.5** (10 días): Comprobación del funcionamiento y si fuera necesario reprogramación del código.

Resultados esperados: Obtención de un sistema autónomo completo capaz de esquivar obstáculos a la vez que es monitorizado en tiempo real y obtiene datos de posición y medioambientales.

TR 4: Diseño, desarrollo de la gestión y visualización de datos. (Duración aproximada de 19 días):

A pesar de que hay creadas BBDD previas de anteriores trabajos de fin de grado, existían ciertas carencias para poder llegar a los objetivos establecidos. Por lo tanto, esta tarea está destinada a crear una nueva BBDD con las tablas necesarias para el almacenamiento de todos los datos obtenidos. También se procederá a diseñar un entorno para la visualización de los mismos con una herramienta llamada Grafana [32], la cual facilitará el visionado de los parámetros. Esta tarea se corresponde con los objetivos 7 y 8. Se definen las siguientes subtareas:

- **SubTR 4.1** (5 días): Instalación de las herramientas que permitan crear una BBDD.
- **SubTR 4.2** (3 días): Diseño de las tablas que contendrán los valores recibidos por los sensores.
- **SubTR 4.3** (7 días): Creación de un código de comunicación entre el agente autónomo y la base de datos.
- **SubTR 4.4** (4 días): Creación y configuración del entorno de Grafana [32] para visualizar en gráficas los datos almacenados en la BBDD.

Resultados esperados: creación de un sistema completo de almacenamiento y visualización de datos en tiempo real.

TR 5: Diseño y creación de un entorno simulado. (Duración aproximada de 4 meses):

Aunque este proyecto sea un avance respecto a anteriores trabajos de fin de grado mediante la incorporación de nuevas tecnologías y peculiaridades, se había creado la necesidad de tener un entorno y un robot terrestre simulado con las mismas características que los reales para poder mejorar los mismos sin necesidad de probarlo físicamente en ellos. Por ello, esta tarea engloba la creación de un contexto simulado y hace referencia a los objetivos 2, 3, 4 y 9. Para poder llevarlo a cabo las subtareas son estas:

- **SubTR 5.1** (3 días): Evaluación del entorno y agente autónomo real, y enumeración de los requisitos que la simulación debe cumplir.
- **SubTR 5.2** (8 días): Estudio de los distintos programas de simulación existentes y selección del más adecuado a los requisitos expuestos.
- **SubTR 5.3** (16 días): Diseño y creación en CAD del agente autónomo y todos los componentes que lo constituyen.
- **SubTR 5.4** (8 días): Creación de los planos de todos los componentes del agente autónomo.
- **SubTR 5.5** (44 días): Aprendizaje del sistema de simulación seleccionado.
- **SubTR 5.6** (16 días): Diseño y creación del entorno simulado, incluyendo mobiliario, obstáculos, condiciones físicas como el fuego, rozamiento, rugosidad del suelo y paredes.
- **SubTR 5.7** (18 días): Exportación del diseño CAD del agente autónomo al entorno de simulación, programación de las funciones de movimiento de este e incluir sensores, haciendo el sistema autónomo simulado lo más parecido posible al real.
- **SubTR 5.8** (9 días): Evaluación del comportamiento simulado y real para incluir posibles mejoras.

Resultados esperados: obtención de todos los conocimientos necesarios para poder crear un entorno simulado capaz de comportarse de forma similar al entorno real y diseño de un agente autónomo simulado con las mismas capacidades que el real, de modo que se puedan incluir mejoras y técnicas de intervención sin necesidad de probarlo en el entorno físico.

TR 6: Pruebas del sistema completo y análisis de los resultados. (Duración aproximada de 24 días):

Por último, esta tarea consiste en probar y validar en escenarios reales el sistema autónomo con el fin de evaluar su eficiencia. Estos entornos pueden ser hostiles o libres de emergencia. Se corresponde con el objetivo 10.

- **SubTR 6.1** (1 día): Primera prueba en la torre de bomberos para comprobar el funcionamiento del agente autónomo, los sensores, obstáculos, subida de datos, autonomía y peso.
- **SubTR 6.2 (**20 días**):** Corrección de posibles fallos en el agente autónomo terrestre.
- **SubTR 6.3** (1 día): Segunda prueba en la torre de bomberos para comprobar el funcionamiento del agente autónomo, los sensores, subida de datos, autonomía y peso con la cámara térmica, en condiciones de humo frío y fuego.
- **SubTR 6.4 (**2 días**):** Comparativa de los resultados reales y simulados.
- **SubTR 6.5 (**10 días): Pruebas en la simulación de los modos automático y de evacuación.

ANEXO I.II DURACIÓN DEL PROYECTO

Tras haber enumerado todas las tareas y subtareas con una estimación aproximada de su duración, la siguiente tabla recoge toda esa información para plasmar las fechas de realización.

Tareas	Subtareas	Fecha de inicio	Fecha final	Duración (días)
Diseño y desarrollo del módulo IPS	Análisis tecnologías UWB	22/07/2020	26/07/2020	4
	Comprensión código	27/07/2020	04/08/2020	8
	Implementación de mejoras	05/08/2020	08/08/2020	3
	Pruebas de calibración	09/08/2020	17/08/2020	8
Diseño y montaje de un nuevo módulo móvil/ Agente autónomo	Estudio y comprensión del antiguo agente autónomo	18/08/2020	30/08/2020	12
	Mejoras y elementos nuevos	31/08/2020	10/09/2020	10
	Ensamblado del nuevo agente autónomo	11/09/2020	21/09/2020	10
	Creación códigos de sensores	22/09/2020	07/10/2020	15
	Creación de las funciones de movimiento	08/10/2020	16/10/2020	8
	Instalación OS y configuración Raspberry	17/10/2020	25/10/2020	8
Implementación del módulo IPS en el agente autónomo	Unión del agente autónomo y el sistema UWB	26/10/2020	29/10/2020	3
	Modificación códigos	30/10/2020	14/11/2020	15
	Reestructuración conexiones	15/11/2020	23/11/2020	8
	Creación de modos de movimiento	24/11/2020	10/12/2020	16
	Comprobación funcionamiento	11/12/2020	21/12/2020	10
Diseño y desarrollo de la gestión y visualización de datos	Instalación herramientas BBDD	22/12/2020	27/12/2020	5
	Diseño de tablas de la BBDD	28/12/2020	31/12/2020	3

	Códigos de comunicación	01/01/2021	08/01/2021	7
	Configuración de Grafana	09/01/2021	13/01/2021	4
Diseño y creación de un entorno y agente autónomo simulado	Evaluación y requisitos del entorno	14/01/2021	17/01/2021	3
	Evaluación programas simulación	18/01/2021	26/01/2021	8
	Diseño CAD agente autónomo	27/01/2021	12/02/2021	16
	Creación de planos	20/03/2021	28/03/2021	8
	Aprendizaje del programa de simulación	13/02/2021	29/03/2021	44
	Diseño del entorno	30/03/2021	15/04/2021	16
	Unificación de ambas simulaciones	16/04/2021	04/05/2021	18
	Evaluación real/simulado	05/05/2021	14/05/2021	9
Pruebas del sistema completo y análisis de los resultados.	Primera prueba	29/04/2021	30/04/2021	1
	Corrección de posibles fallos	01/05/2021	21/05/2021	20
	Segunda prueba	24/05/2021	25/05/2021	1
	Comparativa real y simulación	26/05/2021	28/05/2021	2
	Pruebas en la simulación del resto de modos	01/06/2021	11/06/2021	10

Tabla 8. Tabla de fechas y duración de desarrollo del proyecto. Elaboración propia.

Por lo tanto, la duración total de este proyecto según la tabla es de 313 días. La duración de tiempo desde la fecha de inicio hasta la fecha de finalización es de 324 días.

En la Figura 47, se representa el diagrama de Gantt en donde se indica la duración y las fechas de inicio y fin de cada subtarea para una mejor comprensión de los datos.

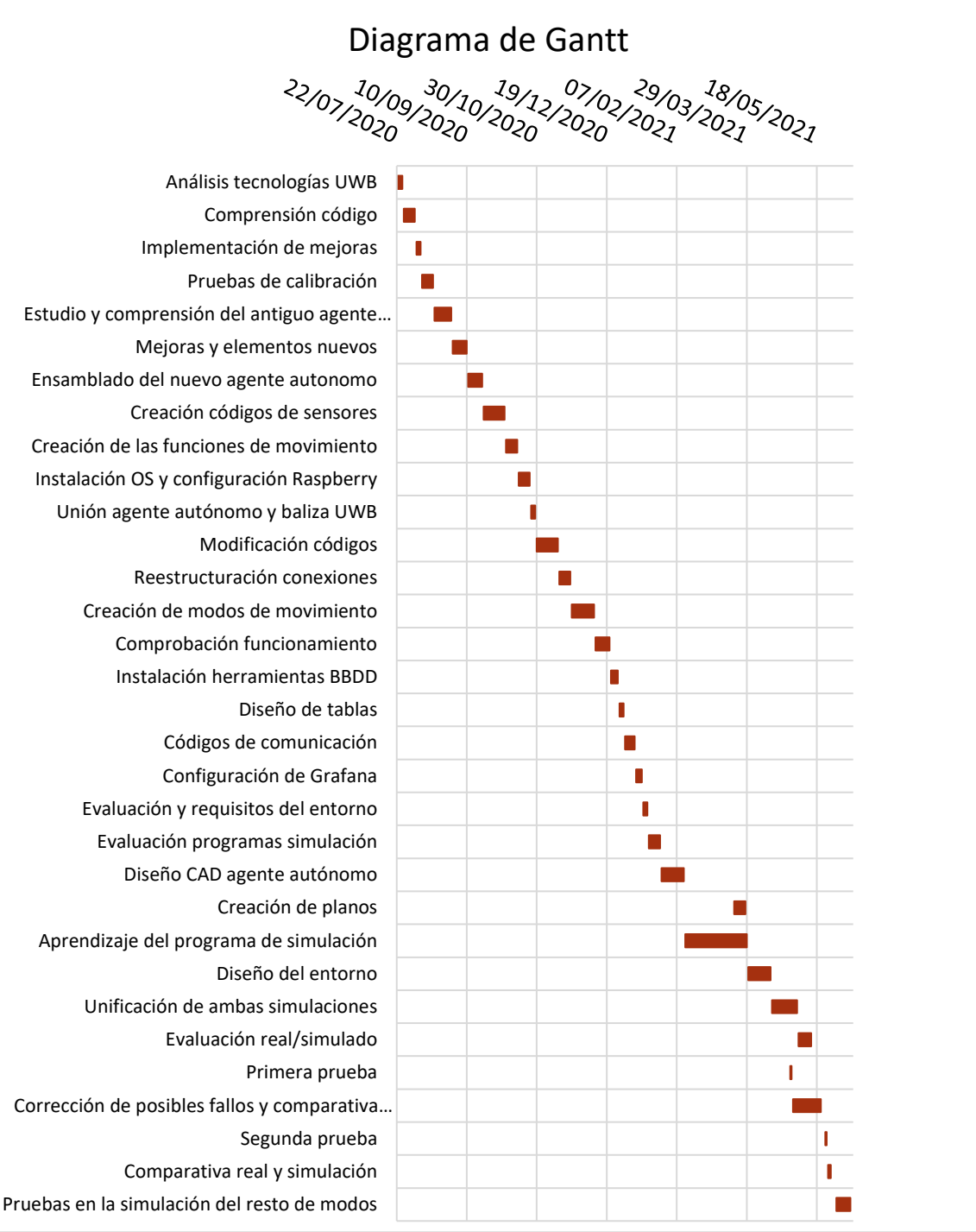


Figura 47. Diagrama de Gantt del proyecto. Elaboración propia

Resaltar que en la situación sanitaria de la COVID-19 en la que se ha desarrollado este Trabajo de Fin de Grado, ha sido más complicado de lo habitual coordinar las tareas de envío de material, solventar problemas o realización de pruebas en espacios cerrados dados los límites de aforo impuestos, lo que ha repercutido en que algunas tareas se hayan demorado en el tiempo más de lo previsto o esperado en circunstancias de normalidad.

ANEXO II. PESO DEL AGENTE AUTÓNOMO Y ROZAMIENTO

a) Peso del agente autónomo

Conocer el peso del agente autónomo es importante para determinar cuanta carga pueden manejar los motores sin verse modificado el comportamiento de este. En la Tabla 9, se encuentran los componentes que definen el agente autónomo obteniéndose un total de 2 kilos y 70 gramos.

Dispositivo	Unidades	Peso/Ud (gramos)	Peso Total (gramos)
Chasis + motores	1	1155	1155
Columnas de unión	8	12	96
Techo extra	2	95	190
Placa Arduino UNO Wif Rev 2	2	36	36
Raspberry pi 4	1	47	47
Controlador de motor	1	30	30
Batería	1	215	215
Pilas recargables	3	34	102
Soporte pilas	1	31	31
Protoboard	1	50	50
Sensor de ultrasonidos + soporte	6	19	114
Sensor temperatura y humedad	1	2	2
Sensor gas	1	2	2
Total	-	-	**2.070**

Tabla 9. Peso del agente autónomo. Elaboración propia

Además, como el diseño es modular se pueden incluir más elementos encima de los niveles existentes. Se puede añadir una UDO y cámara térmica para hacer un guiado de forma visual, como ha ocurrido en una de las pruebas. También se puede incorporar una baliza Deep Beacon que es un dispositivo de adquisición y control multipropósito que se está desarrollando en otro proyecto del laboratorio. En la de en la Tabla 10, se encuentra el peso de estos.

Dispositivo	Unidades	Peso/Ud (gramos)	Peso Total (gramos)
UDO	1	120	120
Cargador	1	100	100
Carcasa	1	140	140
Cámara térmica	1	185	185
Pedestal	1	30	30
Baliza Deep Beacon	1	358	358
Total	-	-	**933**

Tabla 10. Peso de elementos adicionales. Elaboración propia

b) Rozamiento

El par de los motores es de 2,3 kg · cm, por lo que el par total del agente autónomo es de 4,6 kg · cm. Si se pasan los *cm* a *metros* y se multiplica por la gravedad se obtienen las unidades habituales en las que se encuentra esa medida.

Ecuación 3. Cálculo del par de los motores

$$4{,}6\ kg \cdot \text{ cm } \frac{1\ m}{100\ cm} \cdot 9{,}8\ \frac{N}{kg} = 0{,}45\ Nm$$

El par se define como la fuerza que hay que aplicar por la distancia al centro del eje de aplicación, que en caso del agente autónomo es la distancia desde el centro del eje del motor al punto de contacto de la rueda con el suelo que son 8 cm. De la "Ecuación 4. Cálculo del par máximo", se conoce el par de los motores y la distancia, así que se puede despejar la fuerza.

Ecuación 4. Cálculo del par máximo

$$T\ (Nm) = Fuerza\ (N) \cdot \text{Distancia (m)}$$

$$Fuerza\ (N) = \frac{T\ (Nm)}{\text{Distancia (m)}} = \frac{0{,}45\ (Nm)}{0{,}08(\text{m})} = 5{,}63\ N$$

Con esta ecuación se saca la fuerza lineal que hay que aplicar para que su avance sea un movimiento de rodadura sin deslizamiento. Sin embargo, existe una fuerza de rozamiento que varía en función del tipo de suelo donde se realiza la intervención. Se dibuja el diagrama de solido rígido (Figura 48) para ver las fuerzas que están involucradas en este movimiento.

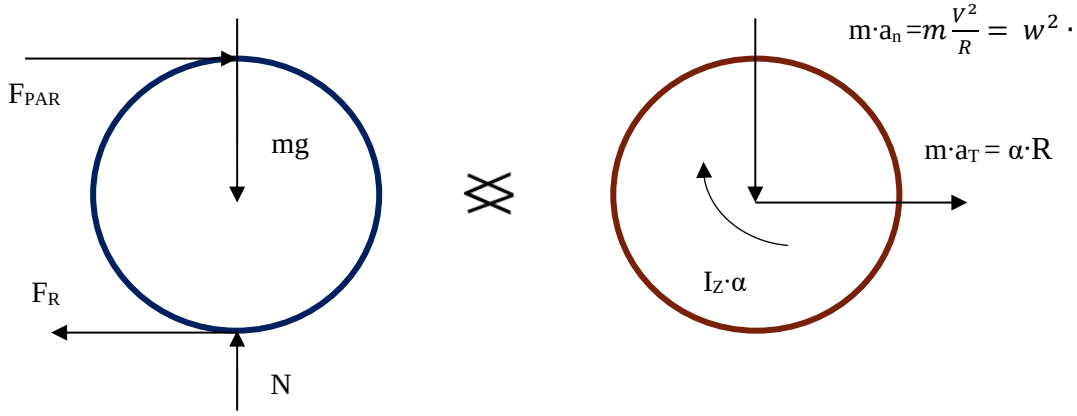


Figura 48. Diagrama de sólido rígido. Elaboración propia

Por otro lado, en las pruebas se midieron las velocidades que presenta el agente autónomo con el suelo en cuestión. Haciendo uso de la Ecuación 5, los datos obtenidos fueron los siguientes:

Ecuación 5. Velocidad del agente autónomo

$$Velocidad\ \left(\frac{m}{s}\right) = \frac{distancia(metros)}{tiempo\ (segundos)}$$

Comando	Distancia (metros)	Tiempo (segundos)	Velocidad (m/s)
Adelante	5,00	20,84	0,24
Trubo	5,00	10,05	0,50

Tabla 11. Velocidades del agente autónomo. Elaboración propia

Teniendo los datos de peso, radio, velocidad, se calculan los sumatorios de fuerzas y momentos obtener μ y α.

Ecuación 6. Sumatorio de fuerzas y momentos

$$+\uparrow \sum F_x = 0 \ \rightarrow\ -\mu \cdot N\left(g \cdot \frac{m}{s^2}\right) + F_{PAR}\,(N) = m \cdot \alpha\ \left(\frac{m}{s^2}\right) \cdot R\ (m)$$

$$\overset{+}{\rightarrow} \sum F_y = 0 \ \rightarrow\ \ N\left(g \cdot \frac{m}{s^2}\right) - m\ (g) \cdot g\left(\frac{m}{s^2}\right) = -m(g) \cdot \frac{V^2\left(\frac{m^2}{s^4}\right)}{R\ (m)}$$

$$+\circlearrowleft \sum M_0 = 0 \rightarrow -\mu \cdot N\left(g \cdot \frac{m}{s^2}\right) R(m) - F_{PAR}\,(N) \cdot R(m) = -I_z(g \cdot m^2) \cdot \alpha\left(\frac{m}{s^2}\right)$$

Con la Ecuación 7, se calcula el momento de inercia del circulo.

Ecuación 7. Momento de inercia de un círculo.

$$I_Z = \frac{1}{2}\, m \cdot V^2 = \frac{1}{2} \cdot\ 2070\ (g) \cdot 0{,}5^{\,2}\left(\frac{m}{s}\right) = 258{,}75\ \left(g \cdot \frac{m^2}{s^2}\right)$$

Se despeja la ecuación de $+\uparrow \sum F_y = 0$ para el cálculo de N:

$$N\left(g \cdot \frac{m}{s^2}\right) = m(g) \cdot \left(-\frac{V^2\left(\frac{m^2}{s^4}\right)}{R\ (m)} + g\left(\frac{m}{s^2}\right)\right) = 2070\ (g) \cdot \left(-\frac{0{,}5^{\,2}\left(\frac{m^2}{s^4}\right)}{0{,}08\ (m)} + 9{,}8\left(\frac{m}{s^2}\right)\right)$$

$$N\left(g \cdot \frac{m}{s^2}\right) = \ 13.817{,}25\ g \cdot \frac{m}{s^2}$$

A continuación, se crea un sistema de dos ecuaciones con dos incógnitas ($\overset{+}{\rightarrow} \sum F_x = 0$ y $+\circlearrowleft \sum M_0 = 0$) para obtener μ y α

$$\left.\begin{array}{l} +\mu \cdot N\left(g \cdot \frac{m}{s^2}\right) + m\ (g)\ \cdot \alpha\ \left(\frac{m}{s^2}\right) \cdot R\ (m) = F_{PAR}\,(N) \\ -\mu \cdot N\left(g \cdot \frac{m}{s^2}\right) R(m) + I_z(g \cdot \frac{m^2}{s^2}) \cdot \alpha\left(\frac{m}{s^2}\right) = F_{PAR}\,(N) \cdot R(m) \end{array}\right\}$$

$$\left.\begin{array}{l} +\mu \cdot 13.817{,}25\left(g \cdot \frac{m}{s^2}\right) + \ 165{,}6\ (g \cdot m)\ \cdot \alpha\ \left(\frac{m}{s^2}\right) = 5{,}63\ (N) \\ -\mu \cdot 1.105{,}38\left(g \cdot \frac{m^2}{s^2}\right) + \ 258{,}75(g \cdot m^2) \cdot \alpha\left(\frac{m}{s^2}\right) \ = \ 0{,}4504\ (N \cdot m) \end{array}\right]$$

Los resultados que se obtienen son:

$$\boldsymbol{\mu \ = \ 3{,}67 \cdot \ 10^{-4}}$$

$$\boldsymbol{\alpha \ = \ 3{,}31 \cdot \ 10^{-3}}$$

El valor del rozamiento es muy pequeño pero suficiente para no provocar el deslizamiento entre las orugas y el suelo.

ANEXO III. AUTONOMÍA DEL AGENTE AUTÓNOMO

El robot autónomo tiene la alimentación dividida de dos formas. Una está destinada al funcionamiento de los motores con las pilas recargables y la otra a la electrónica del sistema con una batería de litio.

Para el cálculo de la autonomía se utiliza un sensor de corriente lineal basado en el efecto Hall [82] que determina el consumo total de los componentes conectados a cada fuente de alimentación. El sensor seleccionado es el ACS714 -30 A + 30 A (Figura 49), que calcula la intensidad de corriente que recorre un componente. Tiene una baja resistencia de aproximadamente 1,2 mΩ y un aislamiento térmico de 2,1 kV RMS. Puede trabajar en rangos de entre -30 A y + 30 A, y genera una tensión analógica proporcional de 66 mV/A centrada en un voltaje de 2,5 V con un error de ± 1,5 %.

Figura 49. Sensor de corriente lineal Polulu [82].

Como el agente autónomo está en movimiento y se necesitan recoger los datos para poder estudiarlos, no se puede tener conectado el robot terrestre al ordenador todo el tiempo. Por eso se decide usar una placa de conexión de tarjeta microSD (Figura 50) [83] que se conecta a la placa Arduino y que acepta tarjetas microSD [84] para poder ir guardando los datos.

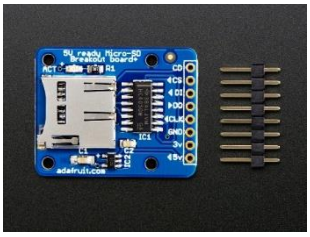


Figura 50. Placa de conexión de tarjeta microSD, Mouser [84].

Además, se ha realizado un cálculo de manera teórica para comprobar las mediciones. Las pruebas se han hecho en vacío (con el autómata rodando en el aire sin someterlo a rozamiento) y con el robot terrestre en funcionamiento sobre el suelo.

Para comenzar se calcula la capacidad total de las alimentaciones:

- Se tienen tres pilas de Ión-Litio [63] de 3,7 V y 12800 mAh cada una, colocadas en serie, teniendo un total de 11,1 V y 38400 mAh.
- Una batería de Litio [64] de 5V y una capacidad de 10800 mAh.

a) Pilas recargables Ión-Litio

Se hace una primera prueba en vacío durante aproximadamente 5 minutos para comprobar que valores de intensidad recoge el sensor, la gráfica obtenida es la siguiente:

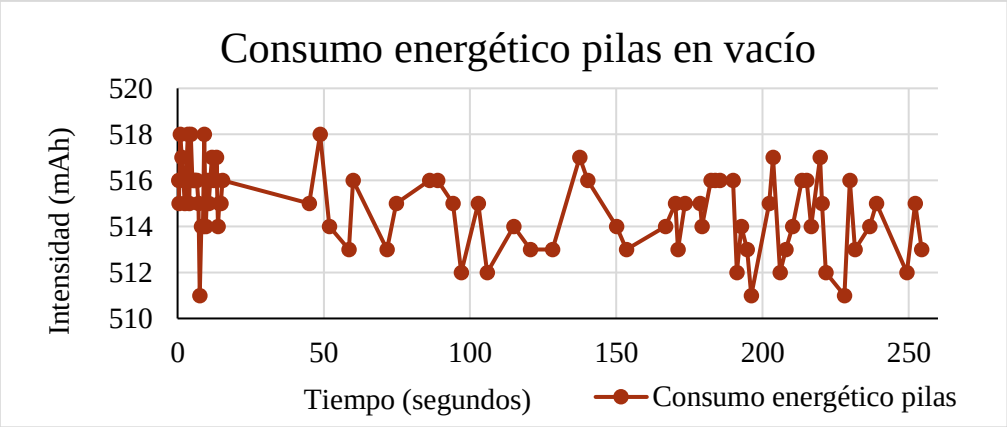


Gráfico 5. Consumo energético de las pilas recargables en vacío. Elaboración propia

Los valores entre los que se encuentra la intensidad de uso están en 511 mAh y 518 mAh.

Ahora se repite la prueba, pero con el agente autónomo en movimiento sobre superficies de diferente rugosidad y con peso adicional. Se estudian los resultados:

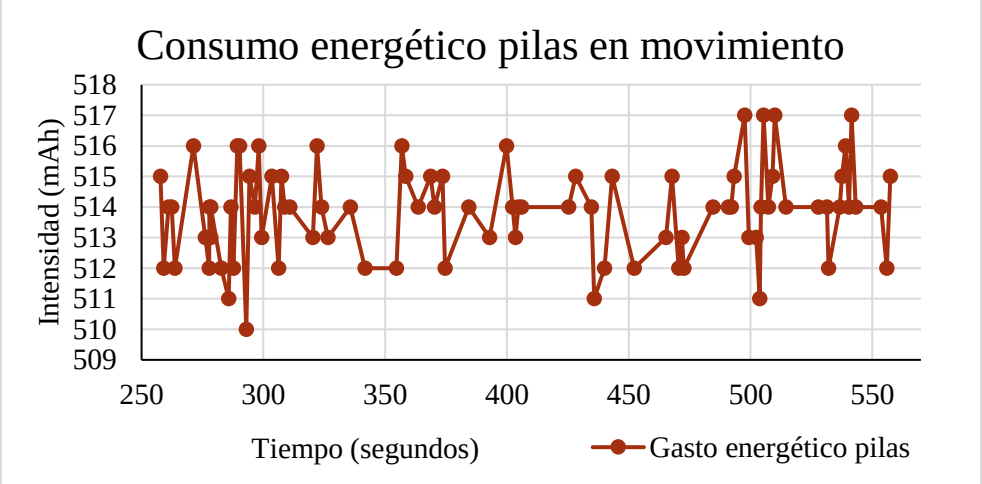


Gráfico 6. Consumo energético de las pilas recargables en movimiento. Elaboración propia

Al igual que antes, los valores de intensidad oscilan entre 510 mAh y 517 mAh, independientemente del terreno por el que circule el agente autónomo y del peso de este.

La razón por la cual ocurre esto es porque el controlador de motor regula la intensidad de los motores, por ello, siempre va a aportar la intensidad óptima para el movimiento, sin que se vea afectado por los factores externos. Otra de las ventajas es que la carga de las pilas no influye

en el comportamiento del autómata, es decir, no se van a obtener funcionamientos distintos al inicio de carga de las pilas o al final de su vida, ya que esto también está regulado. En el momento en el que las pilas se gasten y no puedan aportar la intensidad que requieran los motores, todo el sistema se parará. En este caso, el peso del robot terrestre es adecuado para que no suponga un cambio de comportamiento en los movimientos del agente autónomo.

Por último, se va a comprobar si estos datos obtenidos de manera experimental se aproximan a los cálculos teóricos. Se realiza una tabla con la intensidad necesaria para llevar a cabo el funcionamiento de los componentes conectados a las pilas, es la siguiente:

Elemento	Consumo (mAh)	Unidades	Consumo total (mAh)
Arduino UNO Wifi Rev2	45	1	45
Controlador de Motor	20	1	20
Motores	220	2	440
TOTAL	-	-	**505**

Tabla 12. Consumo teórico de las pilas recargables. Elaboración propia

Como se comprueba, la intensidad teórica es cercana a la que dan los datos recogidos por el sensor, lo que nos hace concluir que la información obtenida es correcta. Por lo tanto, si se divide la capacidad total de las pilas recargables entre el consumo de los componentes se obtiene:

Ecuación 8. Cálculo de autonomía de las pilas recargables

$$Autonomia\ (horas) = \frac{38400\ mAh}{505\ mAh} = 76{,}03\ horas \approx \mathbf{76\ horas\ y\ 11\ segundos}$$

También se va a calcular la resistencia teórica del circuito según la ley de Ohm [85] en función del voltaje teórico y la intensidad media teórica.

Ecuación 9. Cálculo de la resistencia teórica de las pilas.

$$R(\Omega) = \frac{V(V)}{I(A)} = \frac{11{,}1\ V}{0{,}505\ A} = \mathbf{21{,}98}\ \Omega$$

b) Batería de Litio

La batería de litio abastece de corriente a los componentes electrónicos. Las pruebas se han realizado al igual que antes durante 5 minutos en vacío y en movimiento. Los resultados de las pruebas en vacío se ven en el Gráfico 7.

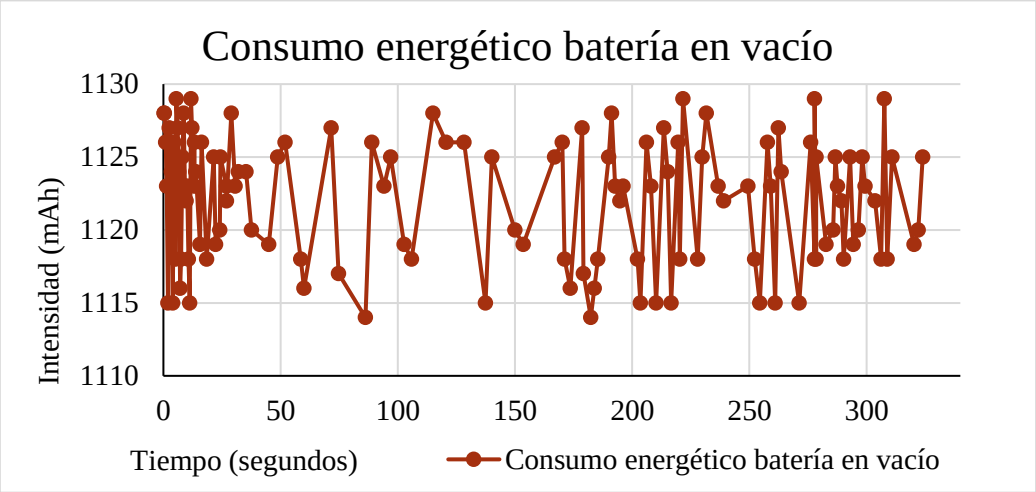


Gráfico 7. Consumo energético de la batería en vacío. Elaboración propia.

El rango de valores oscila entre 1115 mAh y 1130 mAh. Ahora se repite la prueba en movimiento (Gráfico 8).

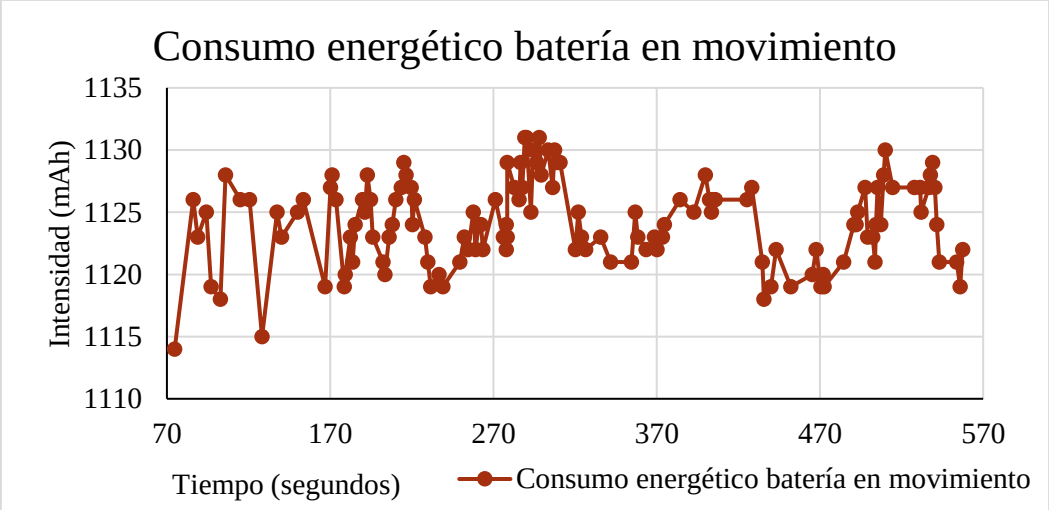


Gráfico 8. Consumo energético de la batería en movimiento. Elaboración propia

Los rangos de consumo se encuentran entre 1114 mAh y 132mAh. La razón por la cual los valores son muy parecidos es porque el peso del robot terrestre o el movimiento no afectan al consumo. También se realiza esta operación de manera teórica.

Elemento	Consumo (mAh)	Unidades	Consumo total (mAh)
Arduino UNO Wifi Rev2	45	1	45
Raspberry Pi 4	1000	1	1000
Sensor de temperatura	15	1	15
Sensor de calidad de aire	15	1	15
Baliza Pozyz	50	1	50
TOTAL	-	-	**1125**

Tabla 13. Consumo teórico de la batería. Elaboración propia

Para el cálculo de la autonomía se divide la capacidad total de la batería y el consumo calculado, obteniendo los resultados de la Ecuación 10:

Ecuación 10. Cálculo de autonomía de la batería

$$Autonomia\ (horas) = \frac{10800\ mAh}{1125\ mAh} = 9{,}6\ horas \approx \mathbf{9\ horas\ y\ 36\ minutos}$$

Por lo tanto, la alimentación limitante es la batería de litio ya que tiene menos autonomía que las pilas. Además, se ha creado un esquema (Figura 51) para ver la distribución de alimentación entre los componentes.

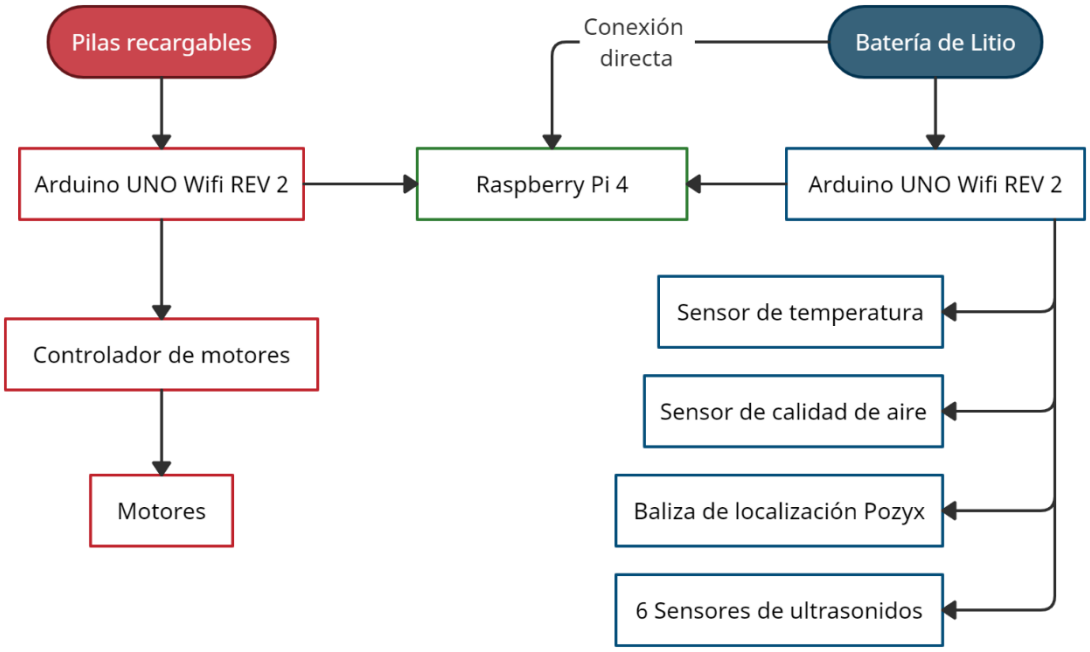


Figura 51. Esquema de funcionamiento de la alimentación. Elaboración propia.

Como se observa, aunque las dos placas de Arduino se conecten a la Raspberry Pi, para que esta funcione es necesario conectarla de manera directa a la batería. Por eso, en el cálculo de la autonomía de las pilas, la Raspberry Pi no está incluida porque la energía que le llega no procede de ellas.

ANEXO IV. PRESUPUESTO

En este capítulo se expone el presupuesto de los componentes y trabajadores involucrados en la intervención de las pruebas. En la Tabla 14, se detalla todo el material que se ha usado para el montaje y funcionamiento del agente autónomo, las unidades totales y el distribuidor. Todos los precios llevan incluido el Impuesto de Valor Añadido (IVA) correspondiente al país de compra, en este caso España.

Dispositivo	Precio/Ud	Unidades	Distribuidor	Precio Total
Chasis agente autónomo	63,53 €	1	Mouser	63,53 €
Motor		2	Mouser	
Interruptor ON/OFF	2,50 €	1	Amitrónica	2,50 €
Power bank	24,74 €	1	RS	24,74 €
Pilas recargables	5,37 €	4	Amitrónica	21,48 €
Controlador de motor	14,44 €	1	Mouser	14,44 €
Placa Arduino UNO Wifi Rev2	38,03 €	2	Mouser	76,06 €
Raspberry Pi 4	97,27 €	1	Mouser	97,27 €
Sensor temperatura y humedad	13,51 €	1	Mouser	13,51 €
Sensor calidad de aire	24,40 €	1	Mouser	24,40 €
Sensor ultrasonidos	10,93 €	6	Mouser	65,58 €
Cables	8,18 €	1	Mouser	8,18 €
Soporte para sensores	4,20 €	6	URJC	25,20 €
Separadores	5,51 €	1	Mouser	5,51 €
Soporte para pilas	6,19 €	1	Mouser	6,19 €
Tarjeta SD	7,05 €	1	Mouser	7,05 €
Adaptador tarjeta SD	6,34 €	1	Mouser	6,34 €
Sensor de energía	10,89 €	1	Mouser	10,89 €
Poxyz-Creator Kit Lite	847,00 €	1	Pozyx	847,00 €
Placas de metal separadoras	7,50 €	2	Mouser	15,00 €
Resistencias	3,01 €	2	Mouser	6,02 €
Total (sin IVA)				1.108,17 €
TOTAL (con IVA)				**1.340,89 €**

Tabla 14. Tabla de costes del agente autónomo. Elaboración propia

En la Tabla 15, se encuentran todos los materiales y personal involucrado en las pruebas en el CUS de Alcorcón ya que era necesario la supervisión de dos bomberos por el uso de humo frío y fuego.

Dispositivo	Precio/Ud u hora (€)	Unidades	Distribuidor	Horas	Precio Total
Tabla de madera	14,29 €	1	Leroy Merly	-	14,29 €
Pallet de madera	29,99 €	11	Leroy Merly	-	329,89 €
Máquina de humo	48,99 €	1	Amazon	-	48,99 €
Líquido máquina de humo	44,77 €	1	Siluj	-	44,77 €
Tubo PVC	32,77 €	2	Leroy Merly	-	65,54€
Valla de obra	60,00 €	18	Leroy Merly	-	1.080,00 €
Silla	7,00 €	3	Ikea	-	21,00 €
Sillón	52,00	4	Ikea	-	208,00 €
Nevera pequeña	99,00 €	1	Media Markt	-	99,00 €
Sartén	4,00 €	1	Ikea	-	4,00 €
Hornillo	19,99 €	1	Decathlon	-	19,99 €
Gasolina	1,40 €/litro	1	-	-	1,40 €
Bombero	100,00 €	2	-	2 h	400 €
TOTAL (con IVA)					**2.336,87 €**

Tabla 15. Tabla con el material y personal para la realización de pruebas. Elaboración propia

El precio total de todos los componentes para la realización del proyecto, el salario del personal implicado, materiales y elementos para llevar a cabo las pruebas es de 3.687,24 €.

ANEXO V. CÓDIGOS PARA EL FUNCIONAMIENTO DEL AGENTE AUTÓNOMO

Se ha creado un repositorio en Git-Hub [75] llamado "TFG-Noelia-Fernández-Talavera" donde se encuentran varias carpetas. Las dos carpetas que contienen los códigos para hacer funcionar los distintos modos del agente autónomo son "Códigos de Arduino" y "Códigos de Python para la Raspberry Pi".

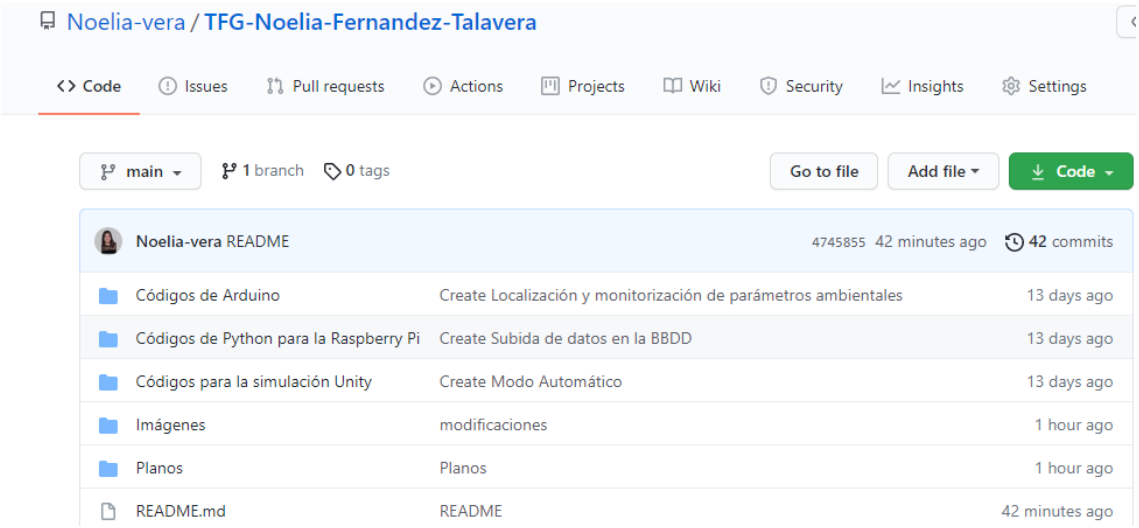


Figura 52. Repositorio TFG-Noelia-Fernandez-Talavera. Elaboración propia

ANEXO VI. SIMULACIÓN DEL AGENTE AUTÓNOMO

a) Entorno AutoCAD

Para explicar el proceso de diseño del robot terrestre, primero se va a presentar el entorno y funciones generales de AutoCAD [68]. El espacio de trabajo se encuentra definido por un eje de coordenadas (Figura 53). En la zona de arriba, se ven una serie de pestañas que permiten el diseño en modo alambre, sólido, superficial o malla, que se irá eligiendo en función de cómo se quiera visualizar un componente. Además, se encuentran otras funciones que permiten dotar de detalle los diseños como uniones, agujeros, aristas, empalmes de radios o cortes.

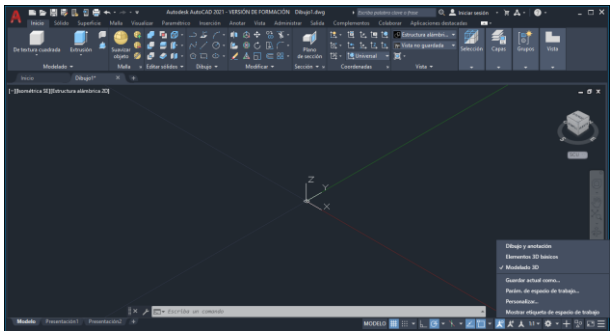


Figura 53. Entorno de AutoCAD. Elaboración propia

En la barra de abajo existen unos modos para definir la forma de diseño. Para nuestro caso, se seleccionará el modelado en 3D. Aquí también se puede configurar el escalado de las piezas a crear, las unidades en las que se van a trabajar, el plano de isometría en que el usuario prefiere trabajar e incluso la visualización del entorno de trabajo.

Por último, en la zona de la derecha se encuentran los comandos de visualización de ventana que indican la dirección *N, S, E* y *O* para saber la perspectiva desde la que se está mirando. Además, están los comandos para girar la vista de la pantalla y moverse por ella.

Existen muchas formas a la hora de afrontar el diseño de un elemento. Aquí se han combinado diferentes sistemas según la necesidad y dificultad de la pieza. Sin embargo, lo más importante es la organización del entorno ya que, como en este caso, al final del diseño se contará con un espacio lleno de muchos componentes que hay que tener identificados.

Por eso, lo primero es crear un conjunto de capas (Figura 54), siguiendo los niveles de ensamblaje explicado en el punto "4.4.3. Estructura final del agente autónomo*"*. Las capas son espacios de trabajo que se superponen y combinan entre ellas. Cada una tiene unas características distintas definidas por el usuario como color de capa, grosor de línea, etc. Una de las ventajas es que se pueden activar o desactivar, de modo que, si se crea un objeto en una capa, se puede dejar de visualizar momentáneamente, pero recuperarlo reactivando la capa cuando sea necesario.

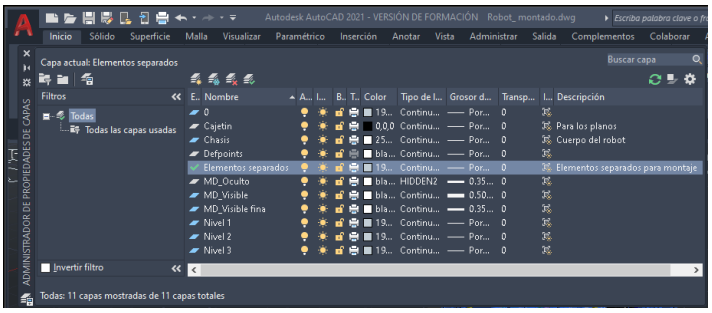


Figura 54. Capas del proyecto. Elaboración propia

Existen algunas capas creadas por defecto como "*Defpoints"*, pero nosotros hemos diseñado las capas del chasis correspondiente al nivel 0, los niveles, 1, 2 y 3, una capa con una copia de todos los elementos creados y otra con el cajetín que se colocará en los planos. También se crean de manera automática las capas "*MD_Oculto"*, "*MD_visible"*y "*MD_Visible fina"* en la elaboración de los planos de las piezas. Aprovechando que el agente autónomo está desmontado, se toman las medidas de todos los elementos.

b) Chasis

El proceso que se ha usado para la recreación de la estructura metálica del autómata es el diseño alámbrico, ya que la forma que presenta es complicada. Para ello se debe seleccionar el comando "línea*"* y se nos pedirá un punto de referencia y una longitud (Figura 55).

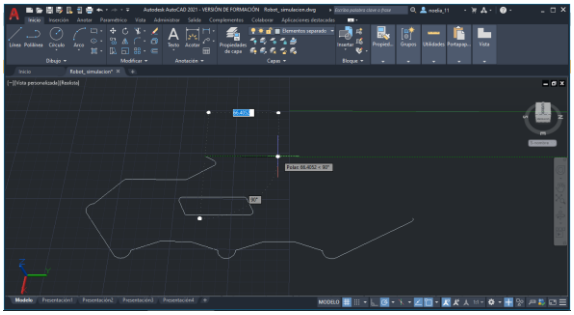


Figura 55. Diseño lineal del chasis Elaboración propia

De esta forma se pueden ir concatenando líneas que posteriormente se unirán para definir un entorno cerrado con el comando "Juntar*"* (Figura 56).

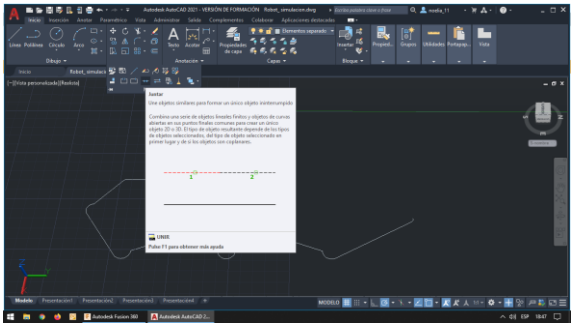


Figura 56. Comando para unir líneas. Elaboración propia

Una vez hecho esto, se procede a definir el espacio cerrado como superficie, seleccionando la pestaña "Superficie" y eligiendo el comando de "Superficie plana". Aquí hay que pasarle el contorno cerrado lineal creado anteriormente (Figura 57).

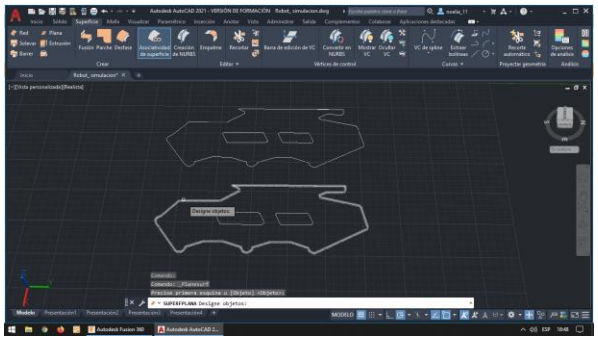


Figura 57. Comando para crear superficies. Elaboración propia

El resultado es el de la Figura 58, donde se distingue en color rosa la superficie creada.

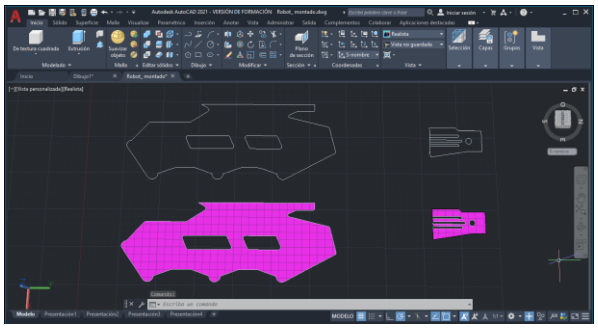


Figura 58. Superficies creadas dentro de un entorno cerrado. Elaboración propia

De esta forma se van elaborando todas las partes del chasis. Lo siguiente es darles espesor a las piezas y esto se hace con el comando "extruir" (Figura 59) donde se solicita la superficie creada anteriormente y un espesor.

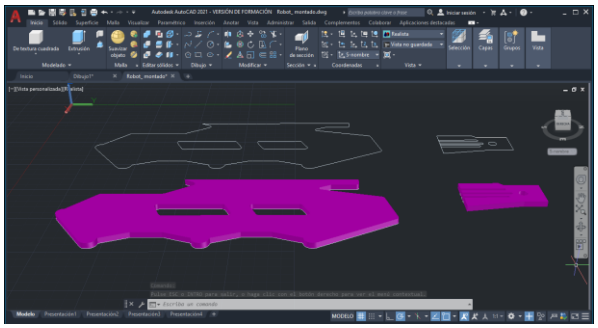


Figura 59. Extrusión de superficies. Elaboración propia

También hay piezas que presentan simetría, así que se hará uso de este comando creando solo la mitad de la pieza, de esta manera se trabaja de un modo más eficiente (Figura 60).

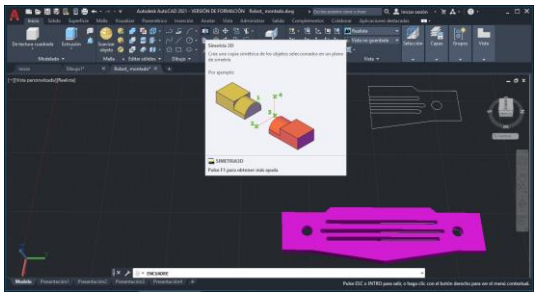


Figura 60. Creación de simetrías. Elaboración propia

Existen elementos ya creados con formas geométricas sencillas que facilitan el trabajo ya que solo se tienen que indicar las dimensiones. Estas formas son pirámides, cubos, cuñas, esferas, toroides y cilindros (Figura 62), donde únicamente es necesario señalar radio o diámetro requerido y altura en cuestión. Así se crean las ruedas del robot terrestre como se ve en la Figura 61.

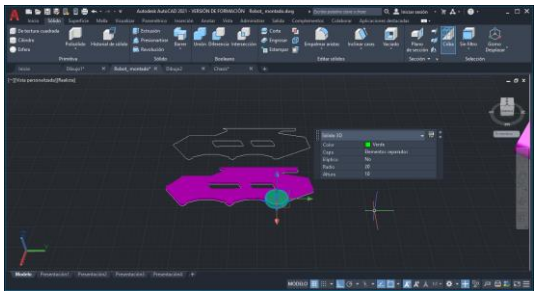


Figura 61. Creación de elementos geométricos. Elaboración propia

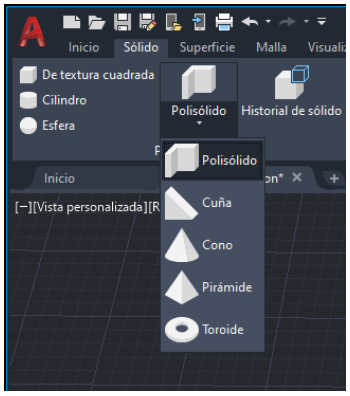


Figura 62. Posibles sólidos predefinidos. Elaboración propia

Una vez se crean las piezas, se procede al montaje de todos los elementos, pudiendo cambiar su color para que se asemejen al color real (Figura 63).

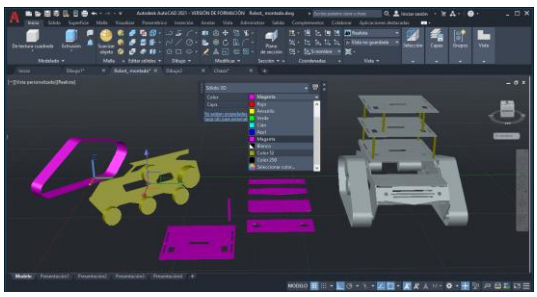


Figura 63. Despiece del chasis del autómata y chasis montado. Elaboración propia

El diseño final del chasis es el que se ve en la Figura 64. Como ya se ha explicado antes, aunque parezca un único componente, sus piezas están divididas en tres capas para que se puedan montar los elementos electrónicos más adelante.

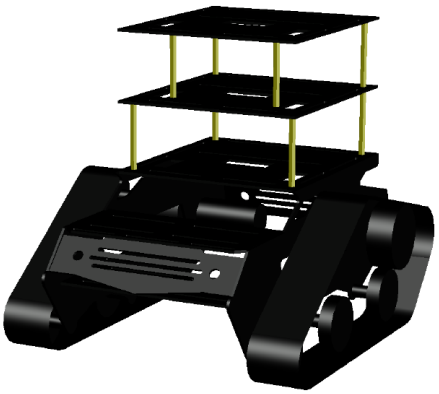


Figura 64. Diseño final del chasis del agente autónomo. Elaboración propia

c) Sensor de temperatura y sensor de calidad de aire

En la creación de estos sensores, se ha usado el mismo diseño base para ambos, ya que físicamente tienen las mismas dimensiones y solo se diferencian en la disposición de los microcomponentes dentro de la placa de soldado. A diferencia del chasis, como las geometrías que presentan son muy básicas, se pueden incluir directamente elementos sólidos prediseñados por AutoCAD sin necesidad de crear la estructura alámbrica. Se comienza con un rectángulo al que se le cambia el color. Con el comando "empalmar aristas*"* de la Figura 65, se pueden redondear las esquinas obteniendo el radio que se necesita. También se pueden hacer los huecos de los tornillos creando cilindros con el radio del hueco y con el comando "diferencia*"*.

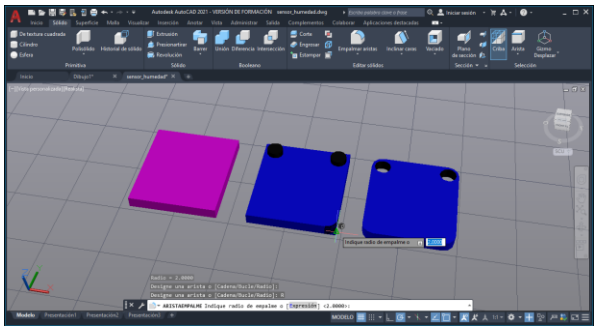


Figura 65. Proceso de creación de los sensores. Elaboración propia

Lo siguiente es crear con formas geométricas básicas los pequeños componentes de los sensores. En este caso se han usado cilindros de un reducido diámetro, cubos y cuñas. Y para finalizar con el diseño, se tendrán que incluir letras en los sensores. Con el comando "Texto*"* se pueden introducir pequeños cajetines a los que se les puede cambiar el tamaño de letra, grosor y de línea tal y como se ve en la Figura 66.

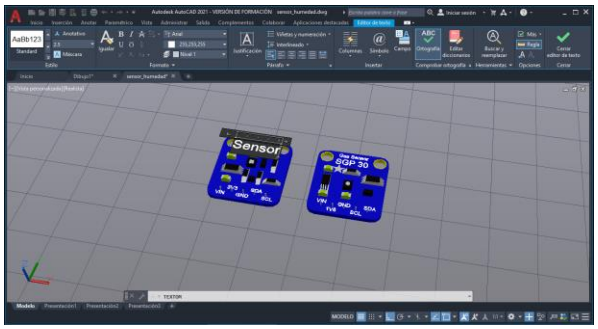
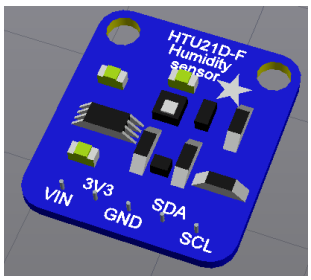


Figura 66. Inserción de texto en los diseños. Elaboración propia

En la Figura 67 se muestra el diseño final de los sensores medioambientales.



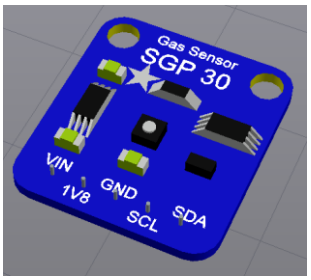


Figura 67. Diseño final de los sensores medioambientales. Elaboración propia

d) Sensor de ultrasonidos y soporte

Los sensores de ultrasonidos que se usan en este proyecto necesitaban de un soporte para ser acoplados en el robot terrestre. El soporte se ha diseñado respetando las características físicas del sensor, pero añadiendo agujeros para tornillos que sean accesibles en la instalación, y esto se ha conseguido por impresión 3D gracias a la Impresora Ultimaker 3 [77] que se encuentra en el departamento de Matemática Aplicada, Ciencia e Ingeniería de los Materiales y Tecnología Electrónica. El diseño tiene una tolerancia de 1 mm para asegurar el buen acople de los tornillos. Además, se ha aprovechado este diseño para incluirlo en la simulación.

El sensor se ha creado con cubos y cilindros, elaborando los agujeros de los tornillos con el comando "diferencia" que se ha visto anteriormente. También se han diseñado tornillos con un cilindro y un cono, y tuercas con una superficie hexagonal que luego ha sido extruida (Figura 68).

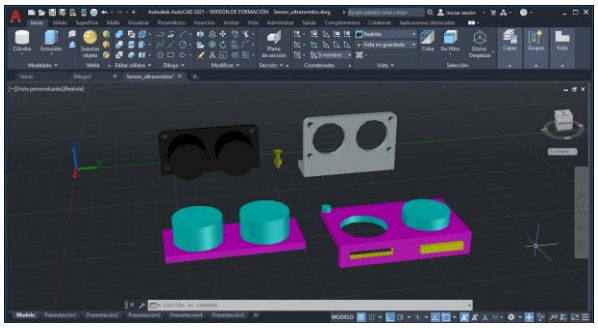


Figura 68. Proceso de creación del sensor de ultrasonidos y soporte. Elaboración propia

Como el soporte tiene las mismas dimensiones que el sensor, solo se ha ampliado el área del prisma base para darle unos centímetros de altura y se han creado unos empalmes entre la parte vertical y la base, para que las vibraciones del movimiento no modifiquen la lectura del sensor. Además, hay unos huecos para los tornillos en soporte base. Estos son alargados para poder acoplar el tornillo en la posición que mejor se adapte del chasis del autómata y evitar hacer más agujeros de lo necesario. El resultado final es el que se ve en la Figura 69.

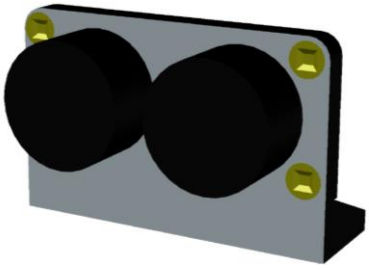


Figura 69. Diseño final del sensor de ultrasonidos y soporte. Elaboración propia

e) Proto placa

Para este componente se ha hecho uso del comando "simetría" tanto en la dirección *x* como *y*. Es el más sencillo de diseñar ya que solo se compone de un prisma rectangular y pirámides invertidas para crear los huecos de los agujeros donde conectar los pines (Figura 70).

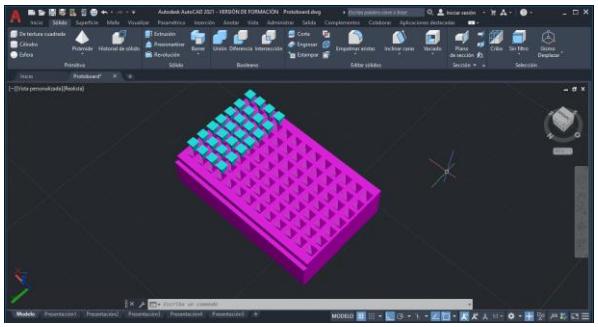


Figura 70. Proceso de creación de la proto placa de conexiones. Elaboración propia

Al finalizar los agujeros solo hay que cambiarlo de color y el resultado del componente es el que se muestra en la Figura 71.

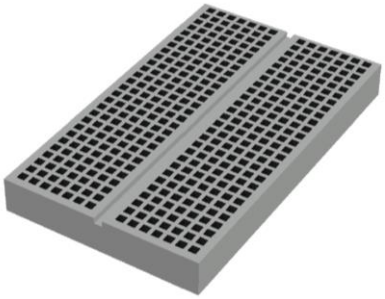


Figura 71. Diseño final de la proto placa de conexiones. Elaboración propia

f) Controlador de motor

Este componente empieza a tener más dificultad por su cantidad de detalles. Sin embargo, en la Figura 72, se ve el proceso de división en elementos geométricos más sencillos. Lo primero es crear una base rectangular. La parte vertical, son una combinación de prismas de diferentess dimensiones con un cono en el medio haciendo de cabeza de tornillo. Los elementos soldados en la parte de la unión en la base son cuñas de pequeñas dimensiones. Lo siguiente es crear los enganches donde irán atornillados los cables de los motores (en verde en la Figura 72). Al igual que antes son cubos a los que se les ha creado ese espacio central con cubos más pequeños y el comando "diferencia".

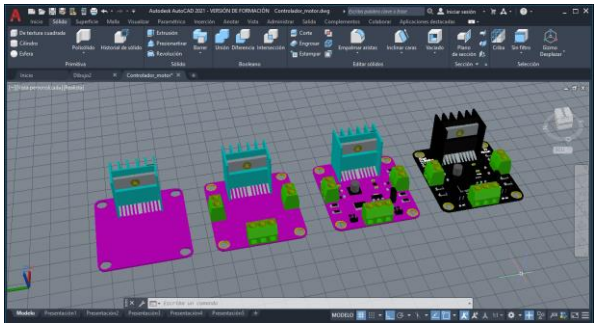


Figura 72. Proceso de creación por partes del controlador de motor. Elaboración propia

Para los pequeños componentes soldados se han reutilizado los diseños que se crearon para los sensores de temperatura y calidad de aire, además de la creación de algún elemento nuevo. Finalmente, se cambian los colores para que se asemejen al componente real y se añaden las letras con el comando de "Texto". El diseño final es el siguiente (Figura 73):

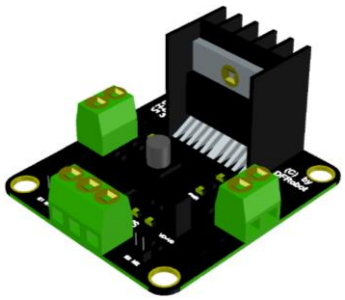


Figura 73. Diseño final del controlador de motor. Elaboración propia

g) Placa Arduino Wifi Rev 2

Este es uno de los componentes con mayor complejidad. A pesar de que existen muchas plataformas de diseño en la web donde personas han creado multitud de componentes electrónicos, la opción de descargar uno de estos diseños no era viable ya que la compatibilidad para poder insertarlo posteriormente en el programa de simulación del entorno no permitía hacer las modificaciones que el diseño exigía, así que se optó por crearlo de igual manera que los anteriores componentes.

Por ello, se crean las figuras que son más sencillas y grandes con prismas y cilindros. Los pines analógicos y digitales estan creados con prismas y pirámides invertidas para hacer el agujero, tal y como se ha visto en la proto placa. Lo siguiente es crear los pequeños componentes soldados, los cuales algunos son reutilizados de diseños anteriores (Figura 74):

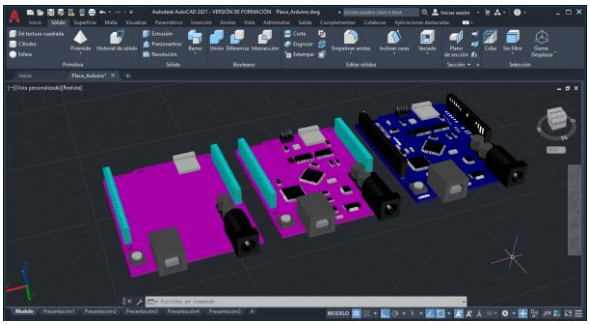


Figura 74. Proceso de creación por partes de la Placa Arduino UNO. Elaboración propia

Por último, se le cambia el color a la placa y se introducen los respectivos textos. Cabe destacar que hay algún componente tan pequeño que se ha dejado fuera del diseño puesto que no iba a modificar la funcionalidad de este. El resultado es el que se ve en la Figura 75.

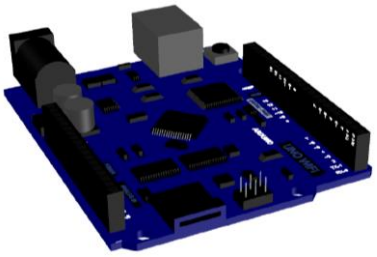


Figura 75. Diseño final de la placa Arduino UNO. Elaboración propia

h) Raspberry Pi 4

A pesar de su complejidad, se han aprovechado muchos de los elementos de la placa Arduino para conseguir recrear esta placa. Se inicia con una base y los elementos grandes más representativos, los conectores. Después se incluye el procesador de la placa y el resto de los conectores pequeños para la alimentación. Por último, se insertan los chips soldados, se le cambia el color y se introducen los textos de código de la placa. El proceso es el de la Figura 76:

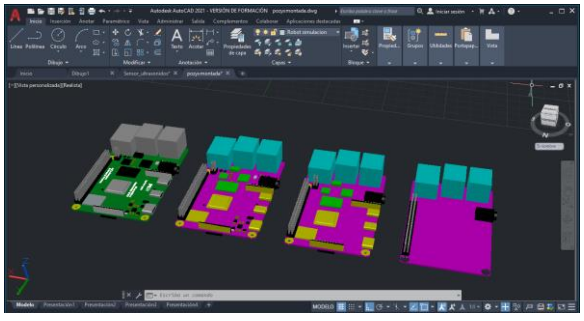


Figura 76. Proceso de creación por partes de la Raspberry Pi. Elaboración propia

Finalmente, se ve cual es el resultado del diseño en la Figura 77. Algo a destacar es que el color del texto se ve modificado al convertir el archivo ya que en el programa AutoCAD las letras son blancas pero una vez convertido a PDF se vuelven negras.

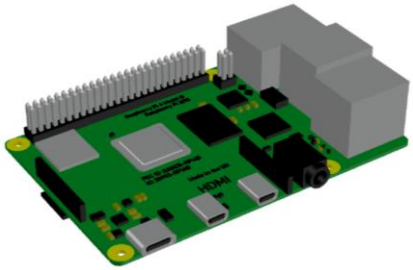


Figura 77. Diseño final de la Raspberry Pi 4. Elaboración propia

i) Baliza Pozyx

Esta placa se encuentra dentro de unas carcasas de plástico negro y son las balizas que se posicionan en la pared para hacer la localización por UWB. Sin embargo, la baliza móvil se conecta a la placa Arduino, por lo que su diseño es muy parecido este. Se crea una base y se reutilizan elementos de conexión ya creados (Figura 78). Se van insertando detalles y colores hasta llegar al diseño final.

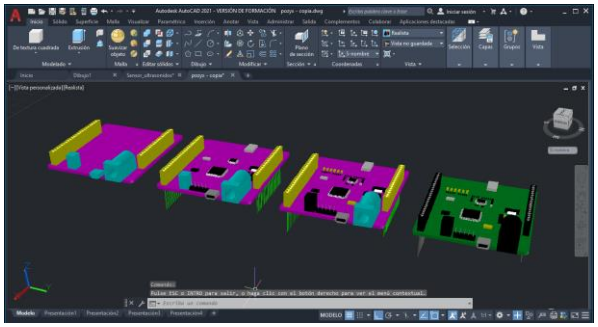


Figura 78. Proceso de creación por partes de la baliza Pozyx. Elaboración propia

El resultado es el que se ve a continuación en la Figura 79.

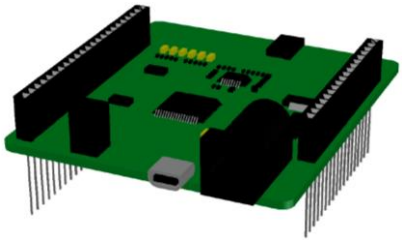


Figura 79. Diseño final de la baliza Pozyx. Elaboración propia

Es importante la toma de medidas, pues en este caso el componente tiene que encajar en otro y si las medidas no son correctas habrá que empezar el diseño desde el principio ya que, aunque se puedan modificar medidas de elementos anteriormente creados, las posiciones no son

las mismas y pueden moverse respecto a su situación real. Vamos a ver en la Figura 81 cómo quedaría la baliza Pozyx acoplada en la placa Arduino.

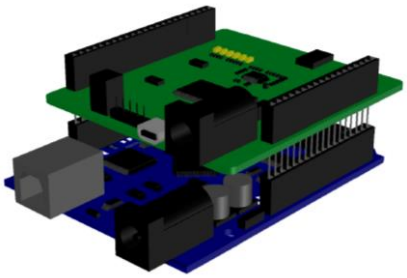


Figura 80. Ensamblado de la baliza Pozyx y la placa Arduino. Elaboración propia

ANEXO VI.I DISEÑO FINAL DEL AGENTE AUTÓNOMO

Lo último que hay que hacer una vez diseñados todos los componentes es ir uniéndolos de acuerdo con el montaje real. En este momento es útil tener el diseño dividido en capas para organizar los objetos de forma ordenada. El ensamblado del primer nivel coincidiendo con el diseño del punto "4.4.3 Estructura final del agente autónomo", es el que se ve en la Figura 81. Recordemos que se encuentra una placa Raspberry Pi, un controlador de motor y una placa Arduino Uno.

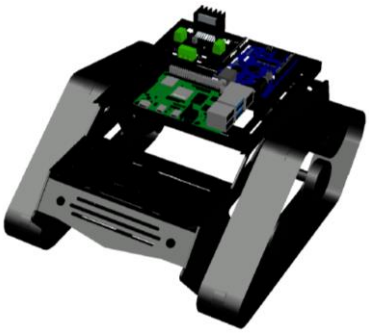


Figura 81.Diseño final del Nivel 0 y 1 del autómata. Elaboración propia

En el siguiente nivel estarán los sensores de ultrasonidos, la proto placa con los sensores de temperatura y calidad de aire, y la placa Arduino Uno con la baliza Pozyx montada encima como se muestra en la Figura 82.

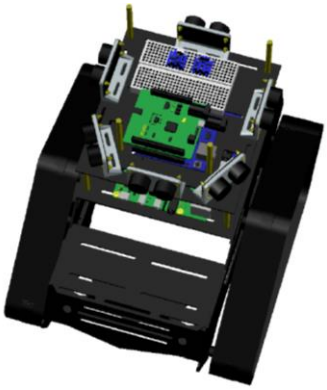


Figura 82. Diseño final del Nivel 2 del autómata. Elaboración propia

Por último, la forma que tendría todo el diseño entero sería a que se ve en la Figura 83.

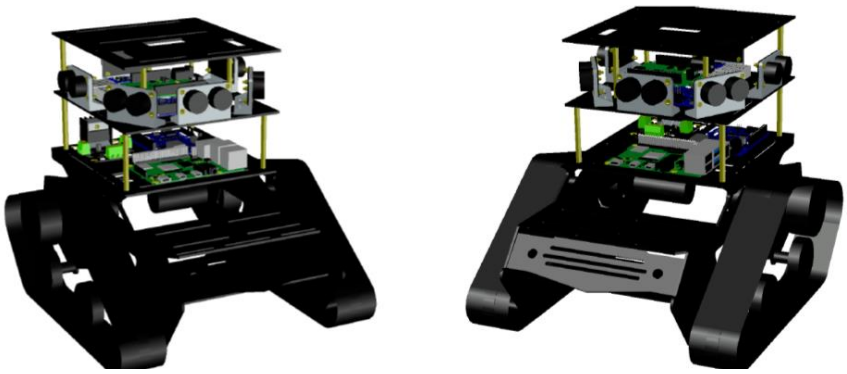


Figura 83. Diseño final del montaje del autómata. Elaboración propia

ANEXO VI.II CREACIÓN DE PLANOS

Como ya se ha dicho antes, con AutoCAD también se pueden crear los planos de los componentes. Lo primero es diseñar un cajetín con la información principal: autor de los diseños, unidades, fecha, escala y nombre del plano. También se pueden incluir de manera opcional otro tipo de información como el proyecto al que pertenece o logo de la entidad responsable del plano. Otro elemento que se debe indicar es el tipo de sistema que se utiliza para referenciar el plano, en nuestro caso se usa el sistema europeo que consiste en representar el alzado, planta y perfil situados en una zona específica del plano, se ve en el siguiente esquema de la Figura 84:

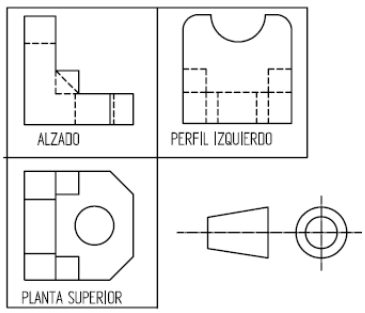


Figura 84. Posicionamiento de las vistas en el sistema europeo. Elaboración propia

De manera opcional se puede introducir la vista en isométrico de la pieza en la parte inferior derecha, este va a ser nuestro caso.

Para empezar a crear el plano se selecciona una pestaña nueva dentro del proyecto, que se encuentra abajo a la izquierda (Figura 85). Aparece una página en blanco donde se pueden insertar múltiples tipos de vista. Dentro de la opción "vista" se despliega la vista "base" que a su vez contiene dos opciones, la que nos interesa es "a partir del espacio de modelado". Lo que esto quiere decir es que dentro de la hoja en blanco va a aparecer todo lo que se encuentra en el espacio de trabajo dentro de la pestaña de "modelo". Esto es importante porque si a parte del diseño de la pieza hay algo más, también saldrá reflejado.

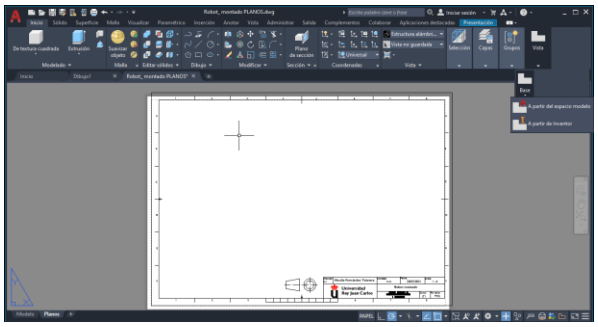


Figura 85. Entorno de AutoCAD para la creación de planos. Elaboración propia

Una vez seleccionado la forma de vista que se desea solo hay que desplazarse a la parte del plano donde se debe situar la planta del objeto (arriba a la izquierda). Una vez ubicada la planta habrá que mover el ratón en diferentes direcciones dentro del plano para que vayan apareciendo el resto de las vistas (Figura 86).

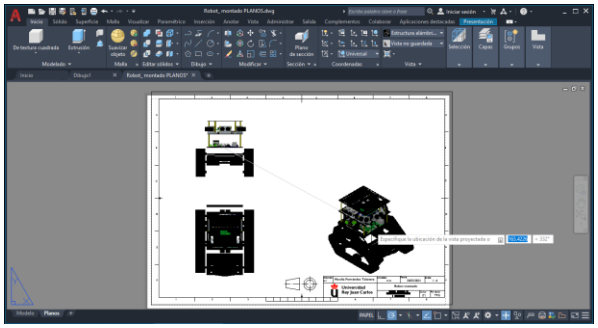


Figura 86. Creación de las vistas de un plano. Elaboración propia

Hay muchas opciones para seleccionar cual se quiere que sea la planta de la pieza, modificar la escala o la visualización del plano entre las opciones “líneas visibles”, “líneas visibles y ocultas”, “sombreado con líneas visibles” y “sombreado con líneas visibles y ocultas”.

Por último, solo queda exportarlo al formato que se necesite. Los planos y las dimensiones de todos los diseños se encuentran en el “Anexo X. Planos” para más ampliación.

ANEXO VI.III COMPATIBILIDAD DEL DISEÑO CON UNITY

El diseño del robot terrestre de AutoCAD tiene que ser compatible con el programa elegido para hacer el entorno de simulación. Sin embargo, también hay que entender de qué formas se van a exportar todos los elementos que se han creado. Unity acepta muchos tipos de formatos de modelado 3D, algunos de ellos son .fbx, .dae, .3ds, .dxf, .obj y .skp. Se hicieron pruebas exportando en la mayoría de formatos y el resultado que se obtenía era parecido, pero el seleccionado es el .fbx, ya que una vez el diseño se encuentra dentro del entorno Unity se pueden modificar los colores y texturas del mismo.

Como se ha visto, el autómata está compuesto de muchos elementos geométricos sencillos con unas características de color determinado. Unity solo entiende lo que son los elementos geométricos, mientras que los textos los elimina al no tener ese comando dentro de sus opciones. Sin embargo, una vez exportado el diseño a Unity se pierden todas las características de color, por lo que hay que volver a definir los colores de los componentes. Desmontar el agente autónomo en Unity para insertar colores a los elementos sería inviable ya que existen más de 500 piezas a las que dar color. Es por esta razón que desde AutoCAD se decide agrupar los elementos por colores. Existe el comando "Unir*"* que une todos los elementos que se necesiten convirtiéndolos en un bloque. Así que se crea el mismo número de uniones que de colores hay, con los colores más repetidos dentro del robot terrestre: azul, verde, negro, gris y amarillo. Esto hace que en Unity se coloreen los componentes de una forma más rápida.

Aunque el diseño en Unity no tiene el mismo visionado que AutoCAD, el resultado es muy realista (Figura 87) y se puede comprobar cómo se diferencian todos los detalles.

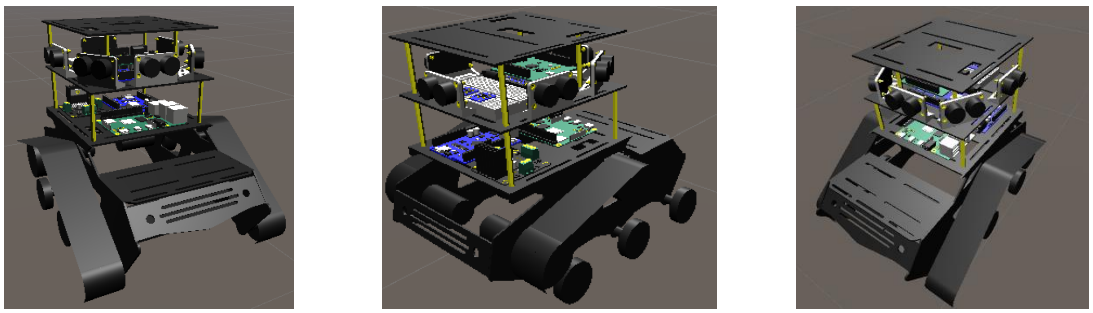


Figura 87. Diseño del agente autónomo importado a Unity. Elaboración propia

En el repositorio en Git-Hub [75] llamado "TFG-Noelia-Fernández-Talavera" se encuentra una carpeta con imágenes de todos los diseños de los componentes hechos en AutoCAD y en Unity. Además, también están los planos y las imágenes del robot terrestre real.

ANEXO VII. SIMULACIÓN DEL ENTORNO

a) Entorno Unity

Se va a explicar el entorno de Unity [71] para comprender cuál ha sido la forma de crear la simulación. La vista se compone de múltiples ventanas que se pueden mover para tener diferentes distribuciones en función de la necesidad del proyecto (Figura 88).

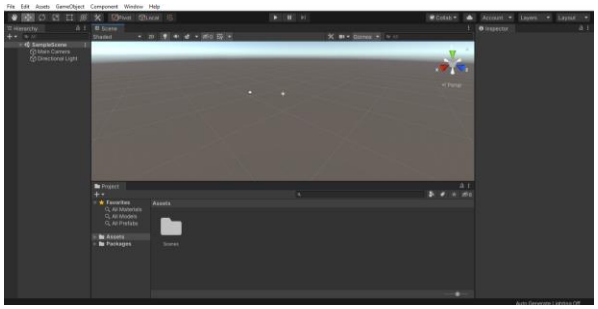


Figura 88. Entorno de Unity. Elaboración propia

La vista de la pestaña de proyecto (Figura 89) es un conjunto de carpetas y ficheros que componen la simulación, algunos ejemplos son scripts, texturas, modelos o sonidos, aunque se pueden añadir tantas carpetas como se necesiten.

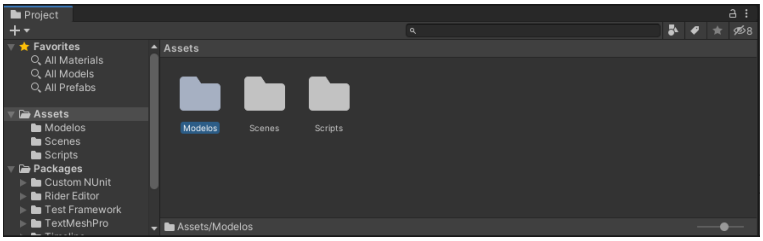


Figura 89. Ventana de proyecto en Unity. Elaboración propia

La ventana "*hiterarchy"* o también llamada jerarquía (Figura 90), es donde se van a ver todos los objetos incluidos en la escena, entre ellos algunos que aparecen por defecto como la cámara o la luz.

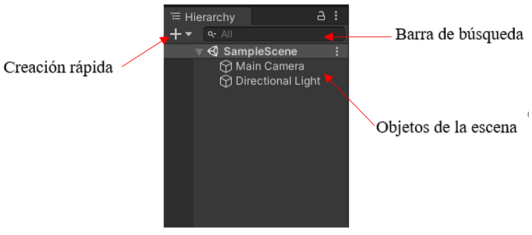


Figura 90. Ventana de jerarquía en Unity. Elaboración propia

La ventana de "*inspector*" (Figura 91) que se encuentra a la derecha de la pantalla y es la que va a recoger todas las propiedades del objeto, pudiendo modificar tamaños, posiciones, rotación con el componente "*transform"*, comportamiento, aspecto del objeto en cuestión con el

componente "*box collider"* o "*mesh renderer"*, etc. En esta ventana se van a poder añadir más características a los modelos.

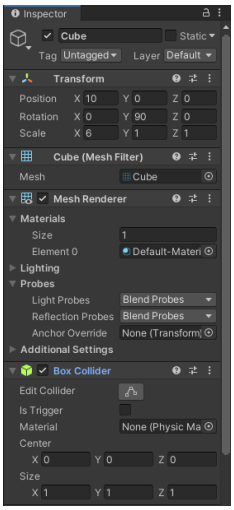


Figura 91. Ventana de Inspector en Unity. Elaboración propia

La vista de la escena (Figura 92) va a permitir ver el aspecto real del entorno que se vaya a crear, indicando si se quiere ver en 2D (vista superior) o 3D (perspectiva). Además, se puede modificar la luz con la que ver la escena, activar sonidos, insertar líneas de horizonte, niebla u otros efectos y, por último, se encuentra el *gizmo*, con los ejes de coordenadas.

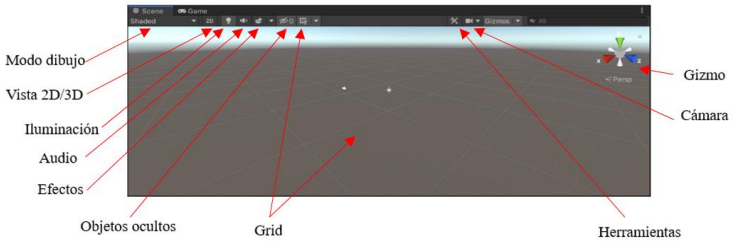


Figura 92. Ventana de Escena en Unity. Elaboración propia

Existe también una vista juego (Figura 93), que es la ventana donde se puede comprobar cómo se verá la simulación una vez iniciada. Esto varía en función de las posiciones de las cámaras de la escena. Se puede modificar la relación de aspecto, el zoom, visualizar la simulación, pausarla o verla paso a paso.

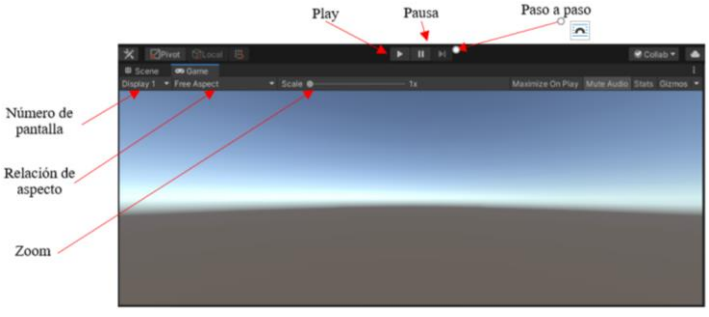


Figura 93. Ventana de Juego en Unity. Elaboración propia

Y por último se encuentra la barra de herramientas (Figura 94) con los controles para navegar por la escena, moverse, girar y hacer transformaciones en los objetos, compartir el proyecto o modificar la disposición del editor.

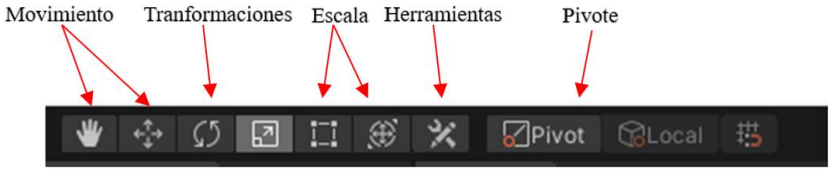


Figura 94. Barra de herramientas Unity. Elaboración propia

En Unity a los objetos se les llama *assets* y hay algunos que están incorporados de manera predefinida. Para incluirlos hay que clicar en la pestaña de "*GameObject"*, seleccionar "*3D Object"* y se desplegará una ventana (Figura 95) con una serie de sólidos como cilindros, cubos, esferas, etc., que se podrán insertar y modificar para crear elementos del entorno.

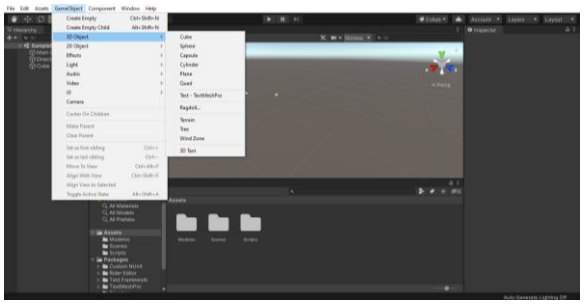


Figura 95. Creación de sólidos. Elaboración propia

También se pueden importar modelos externos ya que esta plataforma cuenta con un web (Figura 96) donde otros usuarios han creado y subido modelos para que otras personas puedan descargárselos y usarlos en sus simulaciones. Estos abarcan desde algo tan sencillo como puertas o coches hasta entornos complejos con mobiliario o personas. Sin embargo, los modelos más elaborados son de pago, por lo que, si se quieren conseguir buenas simulaciones, se deben crear modelos propios.

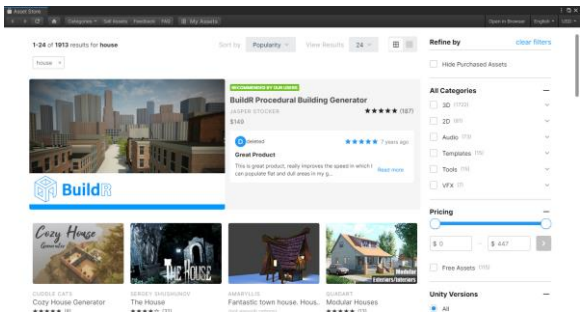


Figura 96. Ventana de la tienda de objetos. Elaboración propia

Ahora que ya estamos familiarizados con el entorno se comienza con el diseño de las plantas seleccionadas.

ANEXO VII. I PRIMERA PLANTA

Para empezar a diseñar la primera planta, se crea el suelo. En este caso, la estancia tiene dimensiones asimétricas, por lo que se va a dividir el suelo en dos partes para poder trabajar más cómodamente tal y como se ve en la Figura 97.

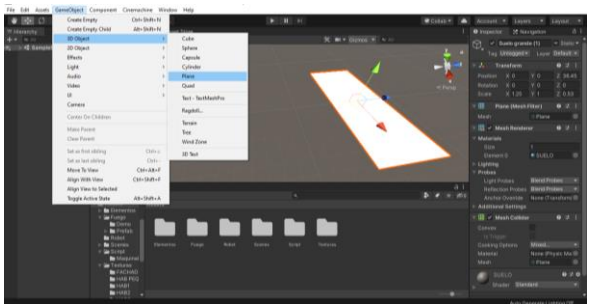


Figura 97. Creación del suelo. Elaboración propia

Lo siguiente es insertar los muros, para ello solo hay que ir creando cubos, modificando su tamaño para que se adecuen a las medidas de los planos. También se incluyen los pilares de carga, muros interiores de separación de estancias y dejar espacios para las ventanas y puertas (Figura 98).

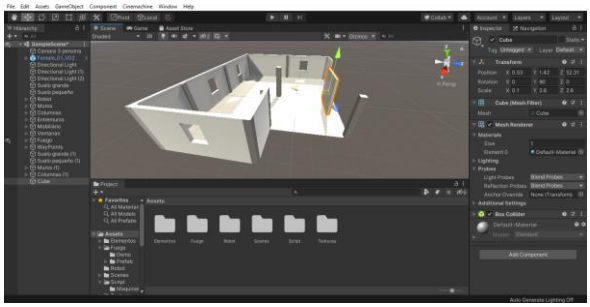


Figura 98. Creación de muros y columnas. Elaboración propia

La forma de posicionarlos puede ser imprecisa si se hace a ojo, pero se puede hacer con precisión ya que Unity entiende los objetos como la posición donde se encuentra el centroide de este. Tomando esto como referencia se pueden calcular las posiciones de todos los elementos en función de sus dimensiones. Lo siguiente que se hará será incluir las puertas y ventanas. Las puertas son metálicas con un diseño muy sencillo (Figura 99), por lo que se creará con un cubo y una cápsula como pomo.

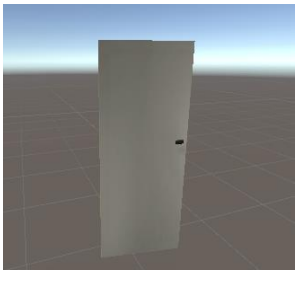


Figura 99. Diseño de una puerta. Elaboración propia

Sin embargo, las ventanas se han descargado de la web de Unity. Para ellos se busca en *Assets Store* el elemento y se descarga como en la Figura 100.

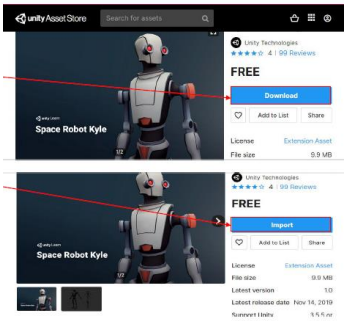


Figura 100. Búsqueda e importar objetos en la tienda de Unity. Elaboración propia

Una vez descargado se importa al proyecto (Figura 101), donde aparecerá una ventana con todas las características del objeto.

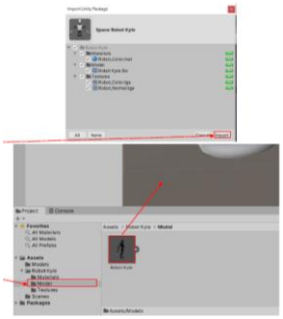


Figura 101. Importar e insertar objetos descargados a Unity. Elaboración propia

Finalmente, para poder incluirlo en la escena, se busca dentro de la ventana de proyecto y se arrastra hacia la escena. Aquí se modificarán las dimensiones para acoplarlo al hueco de los muros. De esta forma se cierra el espacio creado (Figura 102).

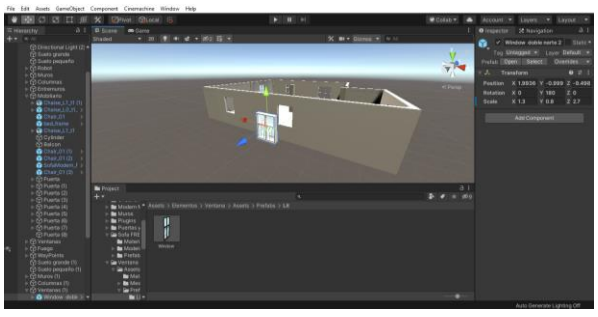


Figura 102. Colocación de una ventana. Elaboración propia

Una vez creado el espacio de simulación, hay que darle un aspecto más real. Para ello, nos desplazamos hasta la torre de bomberos y cogimos fotos de todas las texturas de paredes y suelos para poder importarlos al proyecto. En el diagrama de la Figura 103, se ve cómo se crea un material. La textura es la foto que se quiere importar y es diferente al material final. El material recoge la textura que se quiere y el modelado, y finalmente ese material se aplica a un modelo.

Figura 103. Proceso de creación de un material. Elaboración propia

En la ventana de proyecto habrá que crear una carpeta de texturas donde se importarán las imágenes mediante el comando "*import new asset"* (Figura 104). También se creará otra carpeta donde se guardan los materiales.

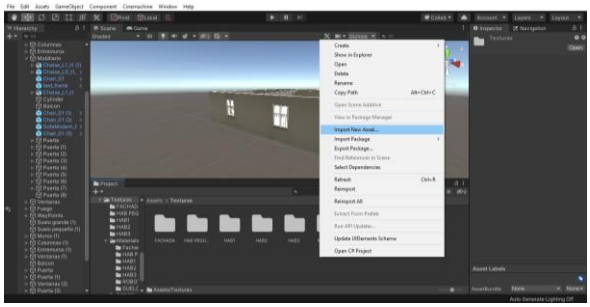


Figura 104. Importar una textura. Elaboración propia

Para crear un material hay que darle al botón derecho del ratón dentro de la ventana de proyecto, luego a la opción "*create"* y por último a "*material"*, como en la Figura 105.

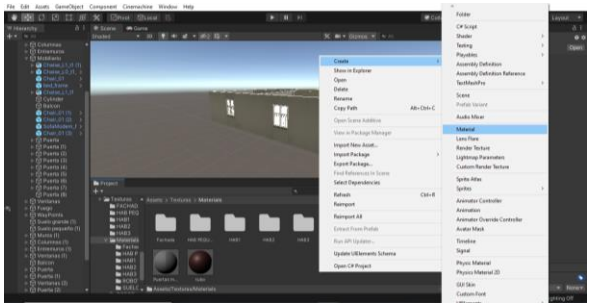


Figura 105. Crear un material. Elaboración propia

Una vez creado el material hay que seleccionarlo y asociarlo a una imagen de la carpeta de texturas creadas anteriormente. Para ello se selecciona el comando "*Albedo"* (Figura 106) y ahí se escoje la imagen que se quiera. Cuando se haya conseguido el material se puede modificar el grado de intensidad, la iluminación y demás características para amoldarlo a nuestro gusto.

Figura 106. Importar una textura en un material. Elaboración propia

Y ahora para poder insertar el material en un objeto este se arrastra al objeto determinado y automáticamente tomará sus características. De esta forma se le va dando textura a la fachada, suelo, paredes interiores y pilares de carga (Figura 107).

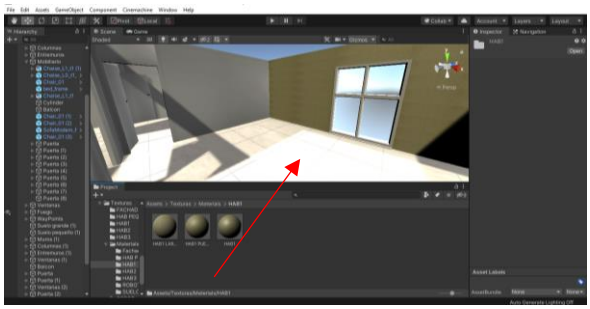


Figura 107. Creación de texturas en objetos. Elaboración propia

Para finalizar con el diseño de la primera planta, se incluye el mobiliario, personas y el fuego del mismo modo que se ha hecho con la ventana, se descarga e importa de la web de Unity. Es importante también la iluminación ya que los muros producen sombra en algunas direcciones y hace que la luz de la escena sea pobre, por eso se han incluido más puntos de luz en zonas estratégicas para que el escenario sea realista.

El resultado es el que se muestra a continuación:

Figura 108. Planta de la primera planta de la torre de bomberos. Elaboración propia

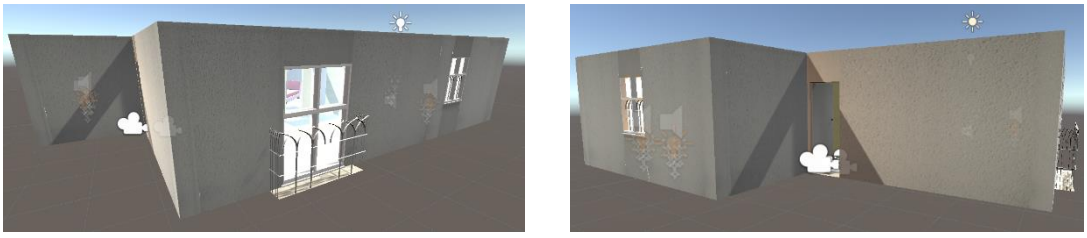


Figura 109. Fachada de la primera planta de la torre de bomberos. Elaboración propia

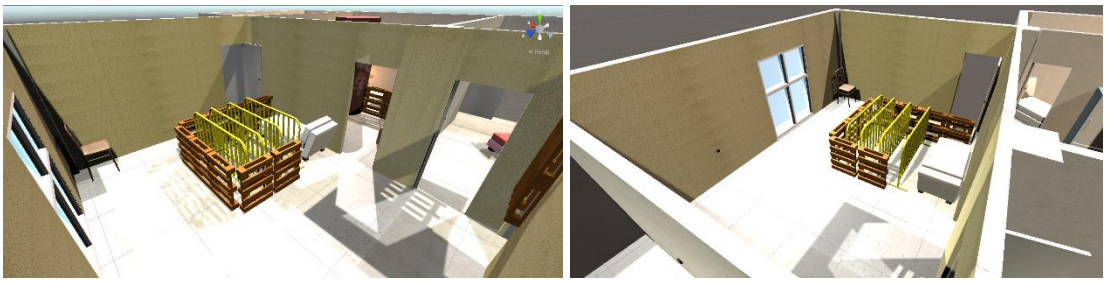


Figura 110. Interior de una de las habitaciones de la primera planta de la torre de bomberos. Elaboración propia

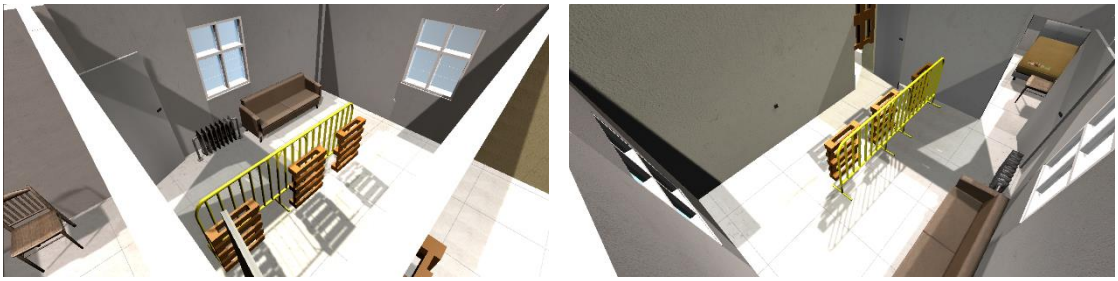


Figura 111. Interior de una de las habitaciones de la primera planta de la torre de bomberos. Elaboración propia

Figura 112. Interior de una de las habitaciones de la primera planta de la torre de bomberos. Elaboración propia

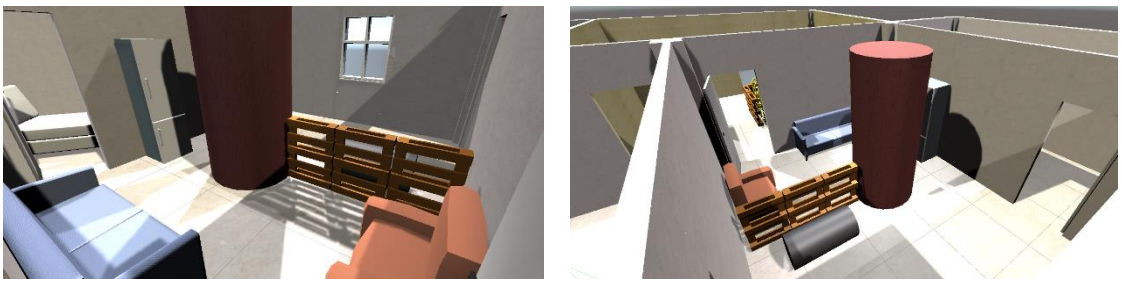


Figura 113. Interior de una de las habitaciones de la primera planta de la torre de bomberos. Elaboración propia

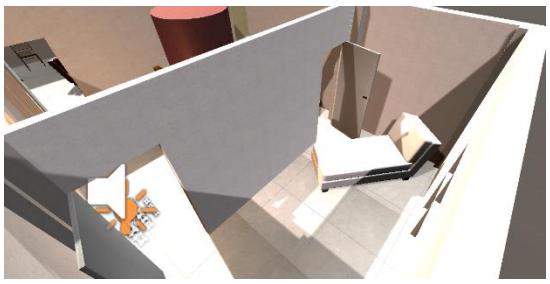


Figura 114. Interior de una de las habitaciones de la primera planta de la torre de bomberos. Elaboración propia

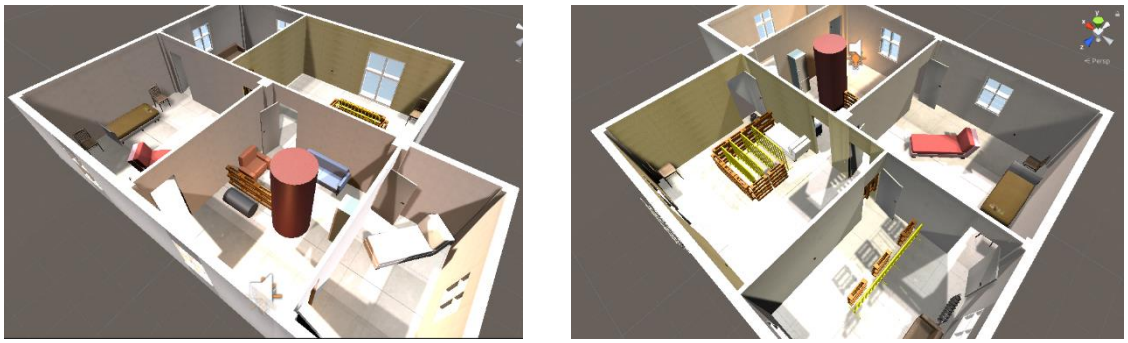


Figura 115. Vista de la primera planta de la torre de bomberos. Elaboración propia

ANEXO VII.II SÓTANO

La creación del sótano es más sencilla porque solo hay una habitación sin muros. El proceso será el mismo que se ha visto para la primera planta, incluyendo el suelo, cuatro muros y vigas. Algo destacable es que no hay ventanas así que el diseño es menos elaborado (Figura 116).

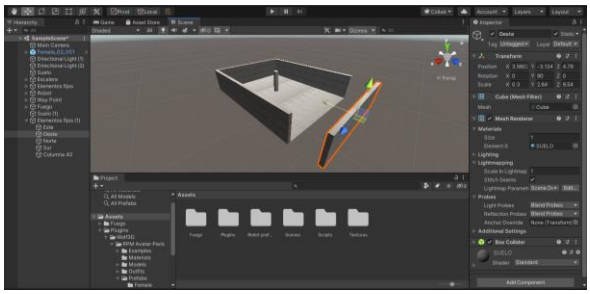


Figura 116. Creación del sótano. Elaboración propia

Además, también se insertan unas escaleras con cubos, un depósito con una geometría de cilindro, tuberías, pallets y obstáculos (Figura 117).

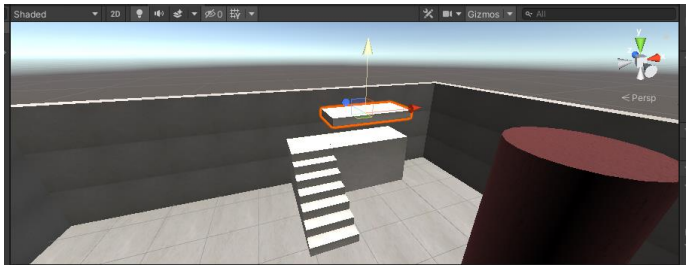


Figura 117. Creación de las escaleras del sótano. Elaboración propia.

El resultado final del sótano es el que se ve en las siguientes figuras.

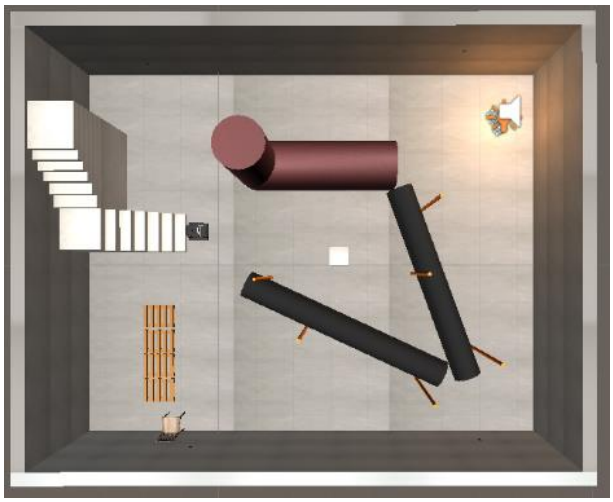


Figura 118. Planta del sótano de la torre de bomberos. Elaboración propia

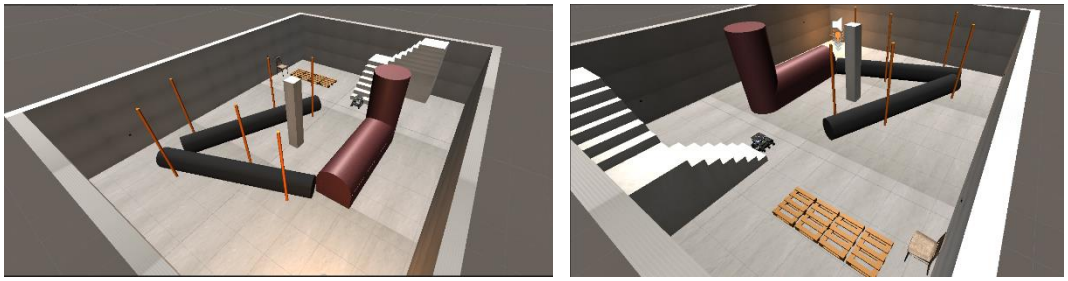


Figura 119. Vista en perspectiva del sótano de la torre de bomberos. Elaboración propia

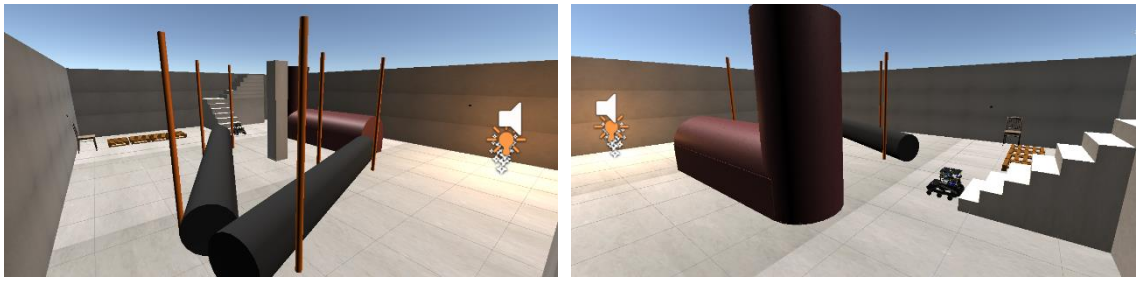


Figura 120. Vista en perspectiva del sótano de la torre de bomberos. Elaboración propia

ANEXO VI.III COMPORTAMIENTO DEL AGENTE AUTÓNOMO

Una vez que se tienen los escenarios creados, hay importar el agente autónomo diseñado en AutoCAD a Unity y hacerlo funcionar. Se inserta el archivo a la ventana de proyecto de Unity y para meterlo en la escena se arrastrará, pero no se puede tratar directamente ya que viene de un programa con sus propios ejes de coordenadas y características, las cuales no son iguales que las de este entorno. En nuestro caso, el robot terrestre está de lado (Figura 121), los ejes de coordenadas para moverlo no salen en el plano de la escena y presenta unas dimensiones diferentes a las reales.

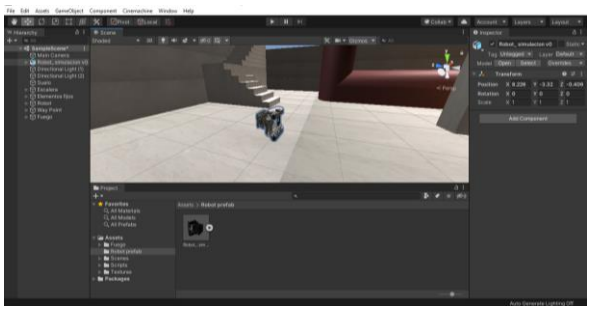


Figura 121. Inserción del agente autónomo en el entorno Unity. Elaboración propia

Para poder manipularlo, se va a crear un objeto vacío con el comando "*create empty"*. Esto es un elemento transparente que presenta las mismas características que un cubo o una esfera. Lo único que se tiene que hacer es arrastrar el robot terrestre a ese elemento vacío y de manera automática los ejes de coordenadas coincidirán con los del entorno. Además, también se podrá cambiar el color y con el factor de escala modificar el tamaño para que coincidan las dimensiones.

Lo que se hace ahora es añadir componentes en la ventana de "*inspector"* para que el autómata tenga un comportamiento parecido al real (Figura 122).

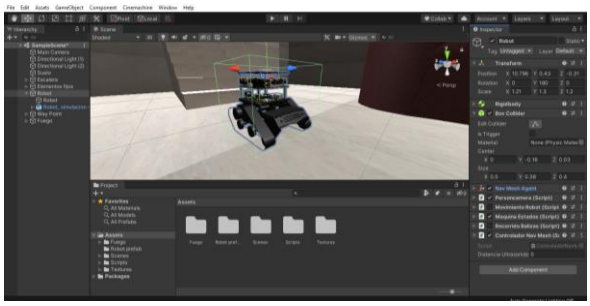


Figura 122. Adición de componentes al agente autónomo. Elaboración propia

Los componentes que se van a insertar para configurar el movimiento son los siguientes:

- Rigidbody: sirve para que el agente autónomo se comporte siguiendo las leyes de la física, consiguiendo un movimiento realista. Aquí se incluye el peso del robot terrestre, la resistencia al aire para cuando este se mueva, la resistencia que opondrá el autómata cuando haga giros, gravedad y que sea un objeto sólido sin posibilidad de que otros objetos lo atraviesen.
- Box collider: función que crea una geometría alrededor del objeto para delimitar las colisiones contra muros u otros objetos.
- Nav mesh agent: componente que se adjunta a un objeto que se vaya a mover y permite navegar libremente en un espacio definido por el usuario.
- Scripts: códigos con el comportamiento del agente autónomo.

Por último, solo queda programar el movimiento del autómata.

a) Modo manual

En este modo el agente autónomo se moverá en todas las direcciones con el teclado del ordenador. Además, se añadirá la posibilidad de controlar con el ratón el movimiento de la cámara 360º para que a la vez que se esté moviendo el autómata se pueda mirar lo que hay alrededor. Para comenzar se escriben unas funciones estándar de Unity para que el código funcione, y luego se declaran una serie de variables principales que contendrán el movimiento del robot terrestre, "*horizontal move y vertical move"*, la velocidad de movimiento "*move speed"*, la variable de movimiento de la cámara y otra serie de variables necesarias para que el autómata se mueva parecido a la realidad. Una de las ventajas de Unity es que la declaración de estas variables puede ser privada, por lo tanto, el valor que se programe será fijo en el momento de la ejecución de la simulación y el usuario no tendrá acceso a él, y la declaración pública, que permite modificar ciertos valores para que el usuario se configure la simulación a su gusto.

Este será el caso de la velocidad del agente autónomo (Figura 123), podrá ser modificada por el usuario y el movimiento de rotación de la cámara, ya que es algo muy peculiar en función de la persona que vaya a controlar la simulación.

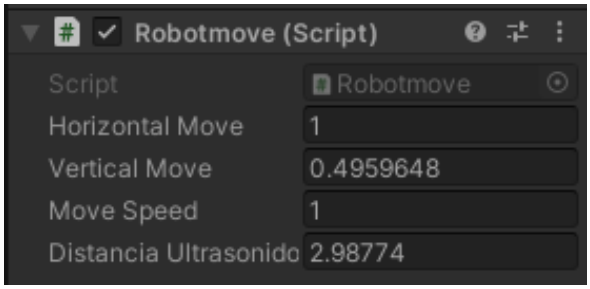


Figura 123. Variables públicas del movimiento del agente autónomo. Elaboración propia.

Lo siguiente es el bucle inicial que solo se ejecuta una vez. Aquí se inicializan las variables que quieren ser llamadas al principio del funcionamiento de la escena. A continuación, el bucle "*Update"* es donde se encuentra toda la programación del código y se ejecuta en cada *frame*. Aquí se indican las funciones de movimiento con las teclas que se corresponden, en nuestro caso existen dos posibilidades, en la siguiente tabla se ven cuáles son las teclas.

Movimiento hacia delante	↑	W
Movimiento hacia atrás	↓	S
Movimiento hacia derecha	→	D
Movimiento hacia izquierda	←	A

Tabla 16. Movimientos del agente autónomo simulado por teclado.

Lo que esto permite también es un movimiento diagonal presionando las teclas que se encuentren entre la dirección de movimiento de la diagonal.

Para continuar, se añade el movimiento de la cámara conjunta con el robot terrestre, mediante funciones de cuaterniones para que gire y la función de seguimiento para que la cámara siga constantemente al autómata. Por último, se fija la posición del eje vertical para que no se mueva en esa dirección, sino que solo la gravedad pueda influirle, y se añade una última función para que el movimiento sea fluido.

El código explicado en profundidad se encuentra en el "Anexo VIII. Códigos de Unity C#" y en el repositorio de GitHub [75] llamado "TFG -Noelia-Fernández-Talavera".

b) Modo autónomo

En este modo lo que se quiere es que el agente autónomo vaya haciendo un recorrido libre dentro del espacio, pero yendo por orden hacia unas balizas colocadas en la pared de la habitación. Para ello se necesita definir el entorno libre en el que se podrá mover el agente autónomo y eso se hace incluyendo el modo navegación en Unity con la pestaña "*window"*, "*IA"* y "*navegation"*, como en la Figura 124.

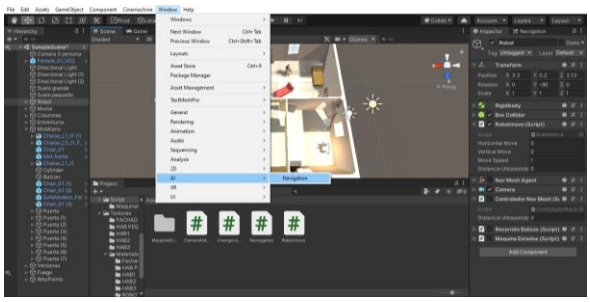


Figura 124. Navegación en Unity. Elaboración propia

Aparecerá una ventana en la derecha de la pantalla donde, en la pestaña "*Object"* se tendrán que seleccionar todos los elementos por los que no se quiere que el autómata pase o atraviese. Una vez hecho esto, en la pestaña "*bake"* se definen las dimensiones del robot terrestre y se le da al botón "*bake"* apareciendo en la escena un manto azul que define las zonas libres por donde se puede navegar sin chocarse con el mobiliario o atravesar paredes (Figura 125).

Figura 125. Configuración del espacio de navegación. Elaboración propia

Ahora se procede a programar el código, para ello se va a usar una máquina de estados donde los diferentes estados serán los puntos a los que se quiere que vaya el agente autónomo, se llamarán *Waypoints* (Figura 126) y serán unos cubos negros iguales que las balizas reales.

Figura 126. Waypoint dentro de la escena. Elaboración propia

Dentro de la máquina de estados se necesitan tres scripts:

- Script principal de la máquina de estados: define el estado actual y el estado de recorrido de las balizas, y de esta forma activará y desactivará los estados. Aunque en este proyecto el estado actual solo puede tomar la función del recorrido de las balizas, se ha hecho de este modo pensando en una posible ampliación en líneas futuras.
- Script de recorrido de las balizas: se inicializan los *Waypoints* indicando al autómata que vaya al primero y cuando llegue a esas coordenadas modifique el punto al que tiene que ir para que vaya recorriendo todas las balizas en orden, volviendo al punto de origen.
- Script de navegación: controla la forma en la que el robot terrestre se mueve por el espacio, indicando cuando ha llegado al punto de destino, paradas y movimiento fluido dentro del entorno.

Con esto se obtiene es el siguiente recorrido evitando el fuego (Figura 127). La figura de la izquierda es la primera planta del CUS y la de la derecha es el sótano.

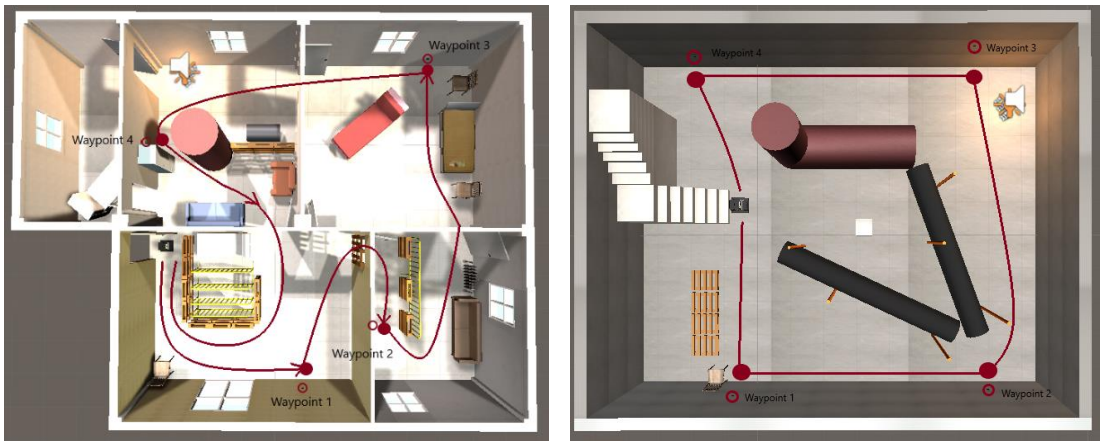


Figura 127. Recorrido de navegación en la simulación. Elaboración propia

En la simulación del recorrido de la primera planta la colocación de las balizas no está en una habitación tal y como se deben posicionar. Se colocan así solo para comprobar si todo está bien configurado y ver si se hace el recorrido de forma correcta. Sin embargo, en el sótano la posición de las balizas sí es correcta.

Los códigos se encuentran mejor explicados en el "Anexo VIII. Códigos de Unity C#" y en el repositorio GitHub [75] llamado "TFG -Noelia-Fernández-Talavera".

c) Modo evacuación

Este último modo es el más sencillo de todos ya que el agente autónomo se encuentra en un aparte del espacio y tiene que encontrar la salida. Para ello Unity tiene un modo en el que el autómata analiza todo el terreno de movimiento y encuentra la ruta más rápida y segura hacia el objetivo.

Esto se consigue con la función de navegación que se ha usado antes, y un código que le indica ir hacia la posición de un objeto llamado "*target"*. Este "*target"* se ha incluido como una persona a modo de bombero que se encontrará en la salida para auxiliar a la gente que salga del incendio. Lo mejor de esta función es que una vez el robot terrestre haya llegado a su destino, se puede volver a recolocar el "*target"* en cualquier parte del espacio y este recalculará el recorrido.

En la Figura 128, se ve el recorrido más rápido hacia la salida que el agente autónomo calcula de forma automática.

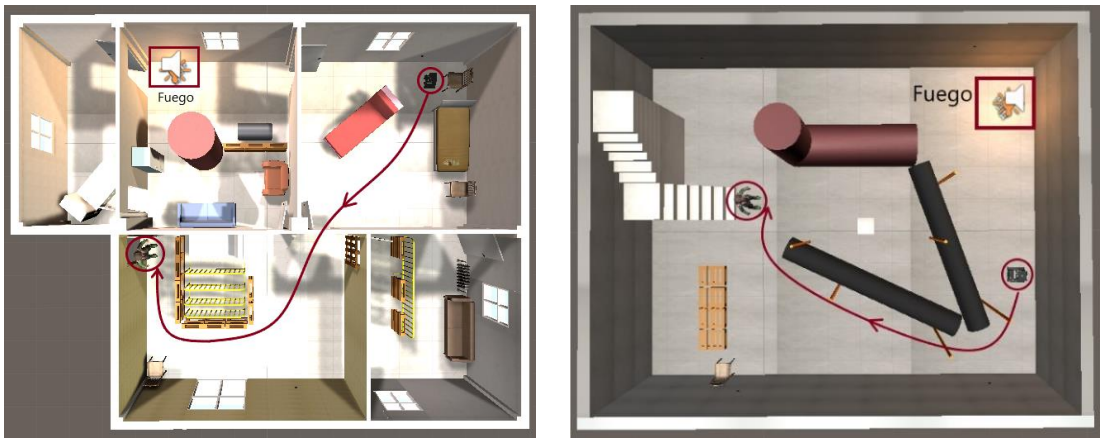


Figura 128. Modo evacuación en la simulación. Elaboración propia

El código se encuentra explicado en el "Anexo VIII. Códigos de Unity C#" y en el repositorio de GitHub [75] llamado "TFG -Noelia-Fernández-Talavera".

ANEXO VII.IV FUNCIONES ADICIONALES

Como la simulación tiene como finalidad que el comportamiento del agente autónomo sea igual en el entorno real y simulado, se han incluido una serie de elementos adicionales tanto físicos como programados para hacer que esto suceda:

- Rugosidad al suelo para que el movimiento del agente autónomo simulado tenga la misma dificultad que el real en el movimiento. Además, en la simulación se comprueba que la visión de la cámara del autómata no es lineal, sino que se mueve un poco debido a esa rugosidad.
- El peso del agente autónomo también se ha modificado para que pese lo mismo que el real con todos los componentes electrónicos.
- La resistencia que el agente autónomo sufre en los giros es un parámetro que se ha cambiado para hacerlo más realista.
- Unity es un entorno que no dispone de sensores, por lo que no es posible calcular la temperatura ambiente y la calidad de aire. Sin embargo, sí existe una función que es parecida a los sensores de ultrasonidos, pues calcula en tiempo real la distancia a todos los objetos que se encuentra el agente autónomo a su paso y los visualiza por pantalla. Esto se ha incluido en todos los modos de movimiento explicados en el punto anterior.

ANEXO VIII. CÓDIGOS DE UNITY C#

Se ha creado un repositorio en Git-Hub [75] llamado "TFG-Noelia-Fernández-Talavera" donde se encuentran varias carpetas La carpeta que contiene los códigos para hacer la simulación es "Códigos para la simulación Unity" (Figura 129).

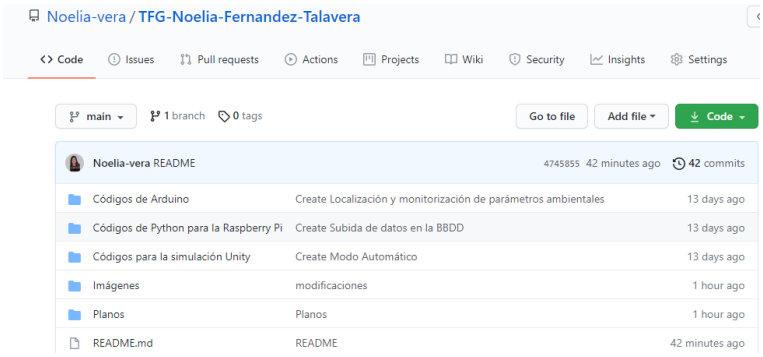


Figura 129. Repositorio TFG-Noelia-Fernández-Talavera. Elaboración propia.

Dentro de esta carpeta hay dos carpetas, una para el funcionamiento del robot terrestre en la primera planta y otra para el sótano (Figura 130). En ambas se encuentran los códigos para el funcionamiento de los modos manual, automático (máquina de estados) y evacuación (Figura 131).

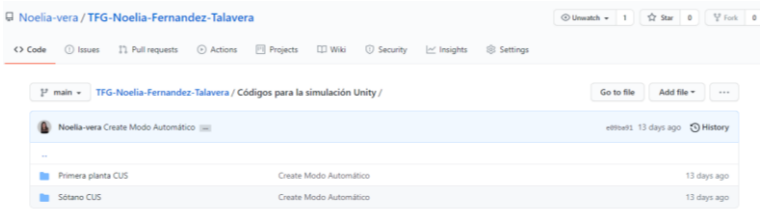


Figura 130. Repositorio TFG-Noelia-Fernández-Talavera. Elaboración propia.

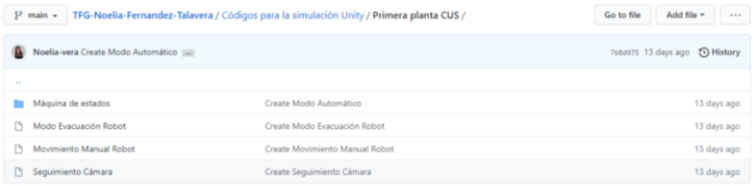


Figura 131. Repositorio TFG-Noelia-Fernández-Talavera (movimiento). Elaboración propia.

ANEXO IX. PROTOCOLO DE ACTUACIÓN EN INTERVENCIONES

Dado que el agente autónomo va a ser usado por los sistemas de emergencia es necesario establecer un protocolo de actuación para las intervenciones donde se explique los pasos a seguir para hacer funcionar el autómata.

Lo primero será poner en marcha los sistemas de alimentación, encendiendo el interruptor de las pilas que dan voltaje a los motores y conectando la batería a la Raspberry Pi 4. Seguidamente, hay que encender la conexión a internet para que todos los elementos se conecten a la misma red, en este caso usaremos la red del teléfono móvil.

Para acceder a la Raspberry Pi 4 que es la que controla el movimiento, se necesita un ordenador y se introduce en el buscador de "*Windows"* la palabra "PUTTY*"*, que es un emulador usado para las conexiones en remoto entre máquinas o servidores mediante Raw, Serial, Rlogni, SSH o Telnet. Aparecerá la ventana de la Figura 132.

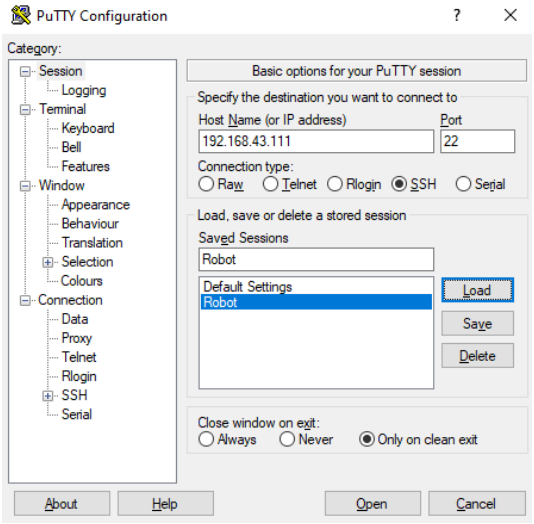


Figura 132. Ventana de PUTTY de acceso a la Raspberry Pi 4

Para poder acceder al servidor, es necesario introducir el *"Host Name"* que es la IP de conexión y el Puerto, en este caso 192.16843.111 y puerto 22. Una vez hecho esto se clica en "*Load"* y se cargará el servidor del sistema.

De forma paralela, se puede abrir la base de datos que se ha creado para comprobar que la subida de datos se está haciendo de manera correcta, y se puede abrir Grafana para visualizar los datos en tiempo real.

- Para abrir la base de datos se introduce en el buscador de internet: "*http://192.168.43.111/phpmyadmin/"*
- Para abrir Grafana se introduce en el buscador: *" http://192.168.43.111:3000/*"

Lo siguiente será indicarle al agente autónomo qué modo de movimiento se quiere implementar y para eso será necesario iniciar la Raspberry Pi 4. El usuario que hay que introducir es "pi" y la contraseña es "pitfg*"*.

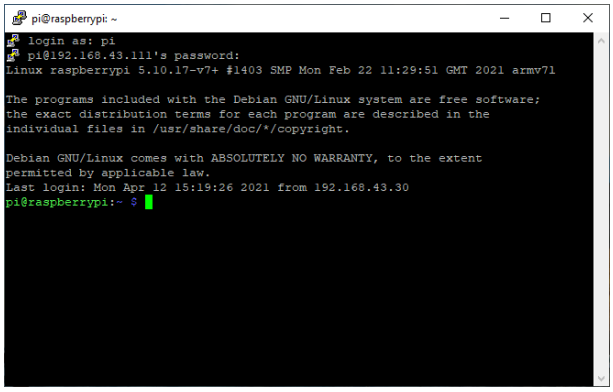


Figura 133. Iniciar sesión en la Raspberry Pi 4

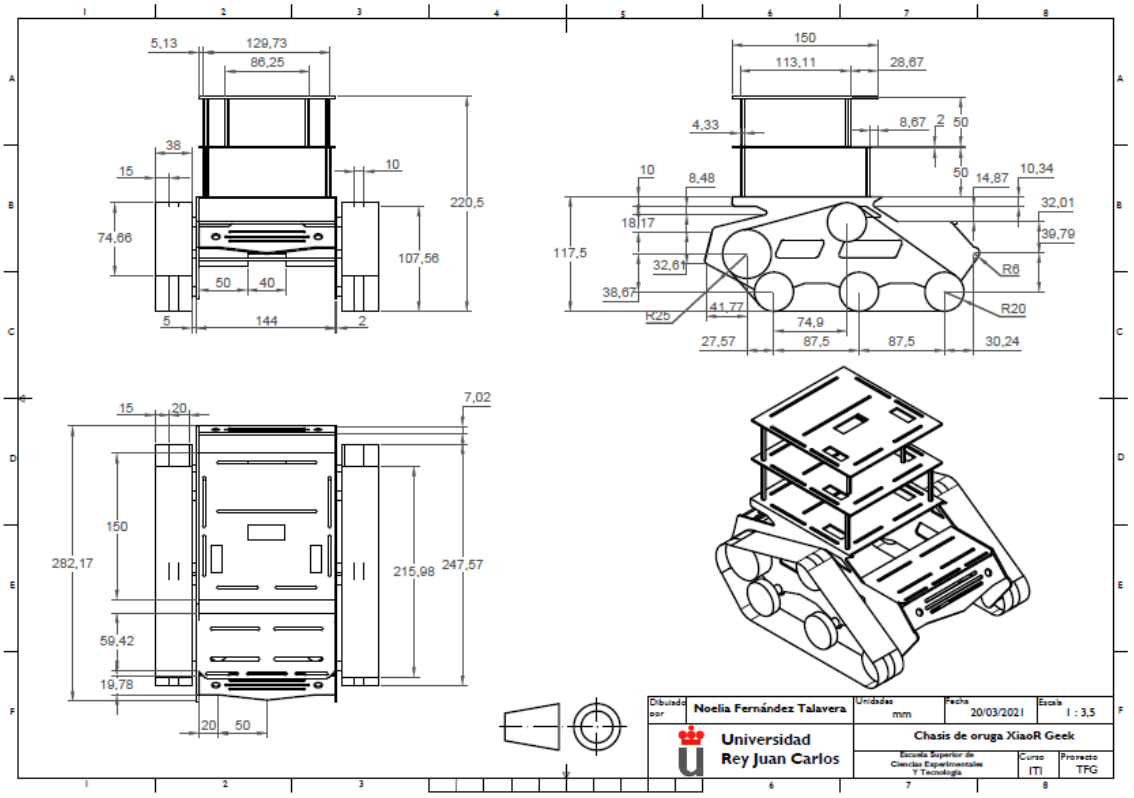
Por último, solo hay que seleccionar el modo de movimiento que se quiere ejecutar. Se escribe en la línea de comandos las siguientes funciones para que se ejecuten y el agente autónomo empezará a moverse.

- Modo manual: "*python control_motores.py"*
- Modo automático: "*python automático.py"*
- Modo evacuación: "*python evacuación"*

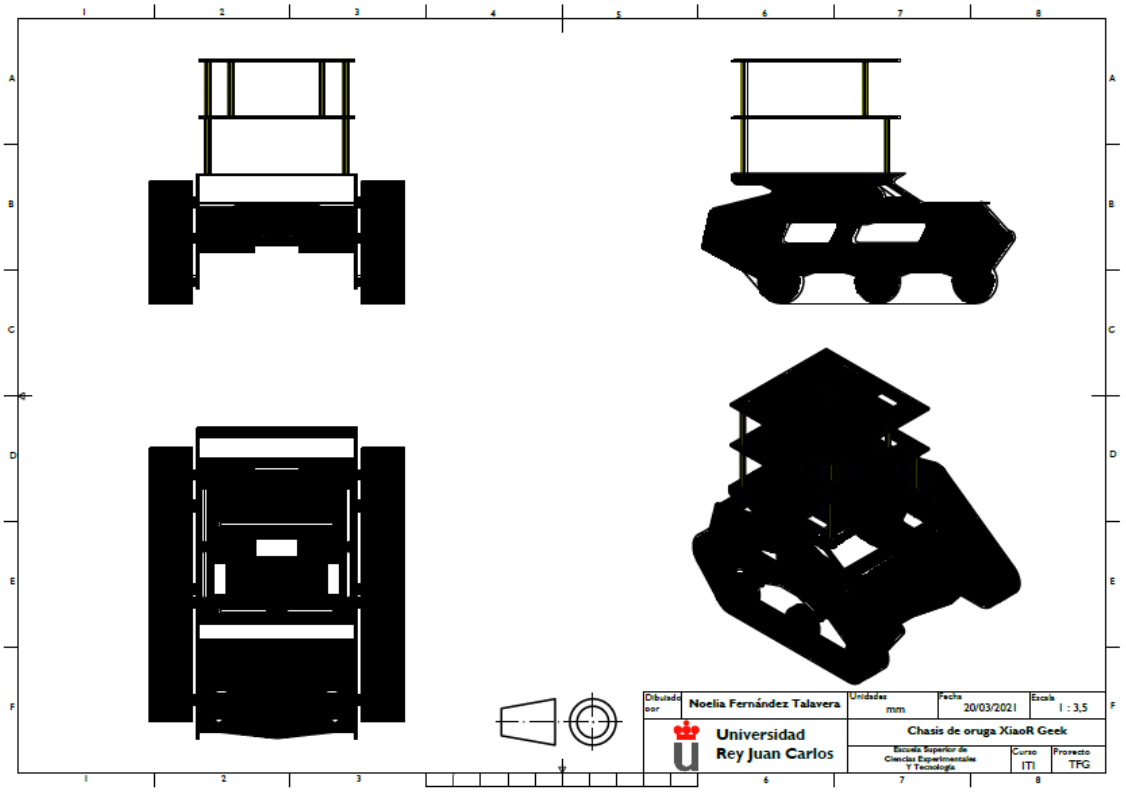
No es necesario iniciar el comando que ejecuta la base de datos porque está configurado para que se empiece a ejecutar en el momento en el que se inicia la Raspberry Pi 4.

ANEXO X. PLANOS

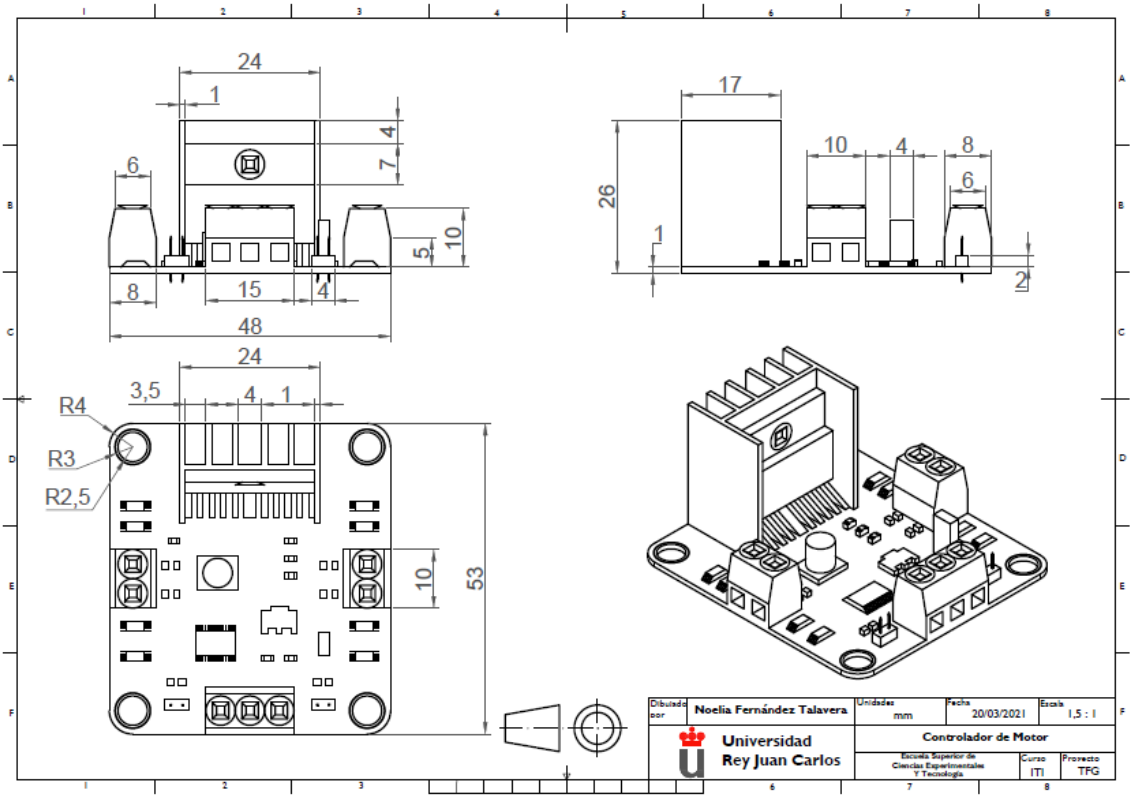
Plano 1. Cotas del chasis del agente autónomo. Elaboración propia



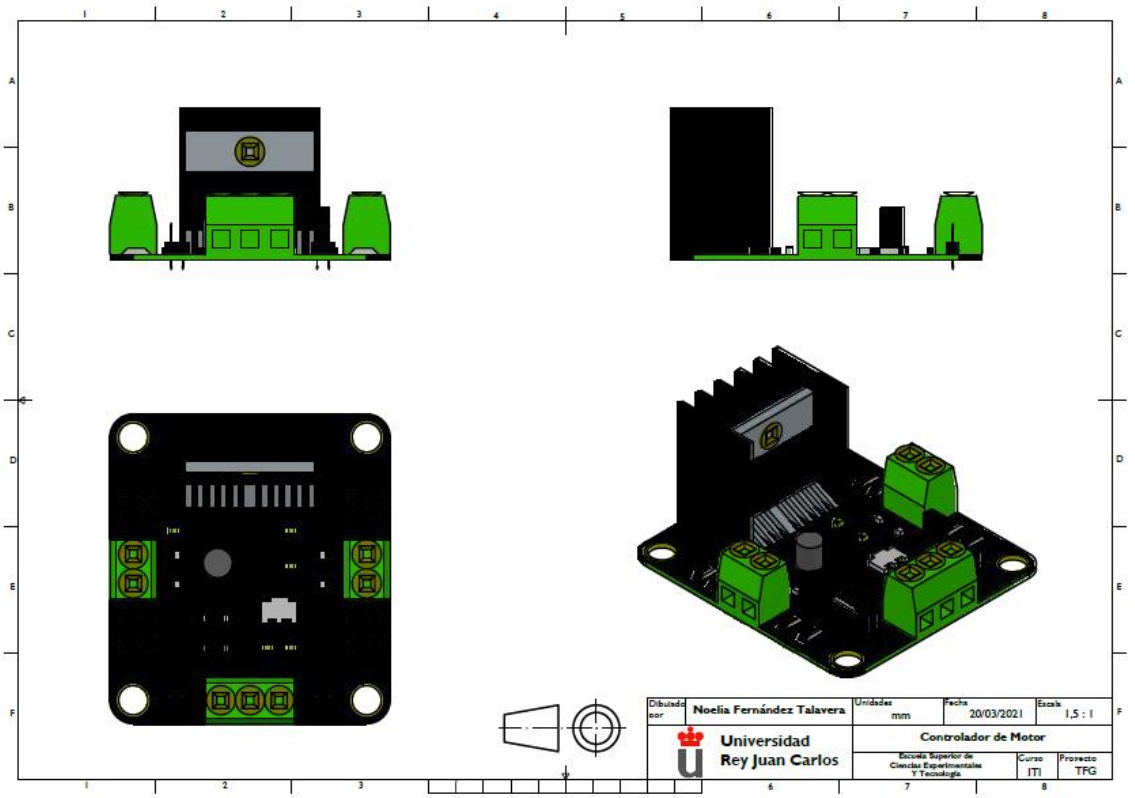
Plano 2. Vistas del diseño del agente autónomo. Elaboración propia



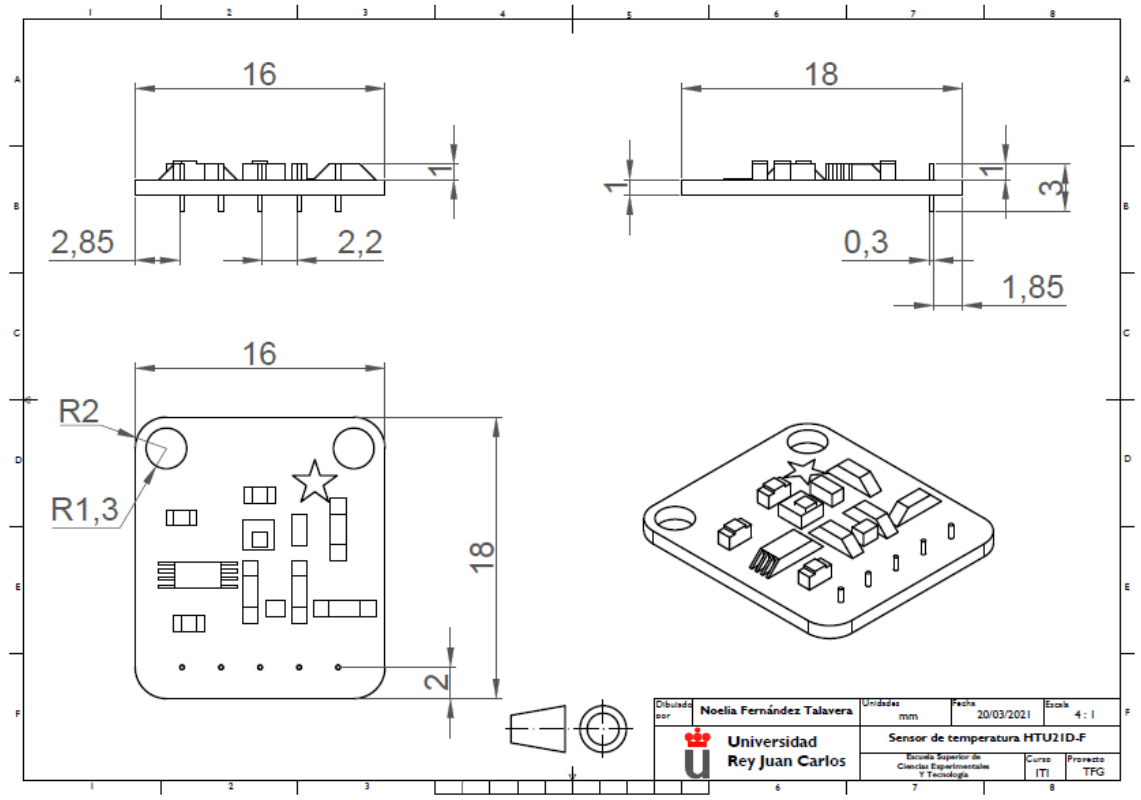
Plano 3. Cotas del controlador de motor. Elaboración propia



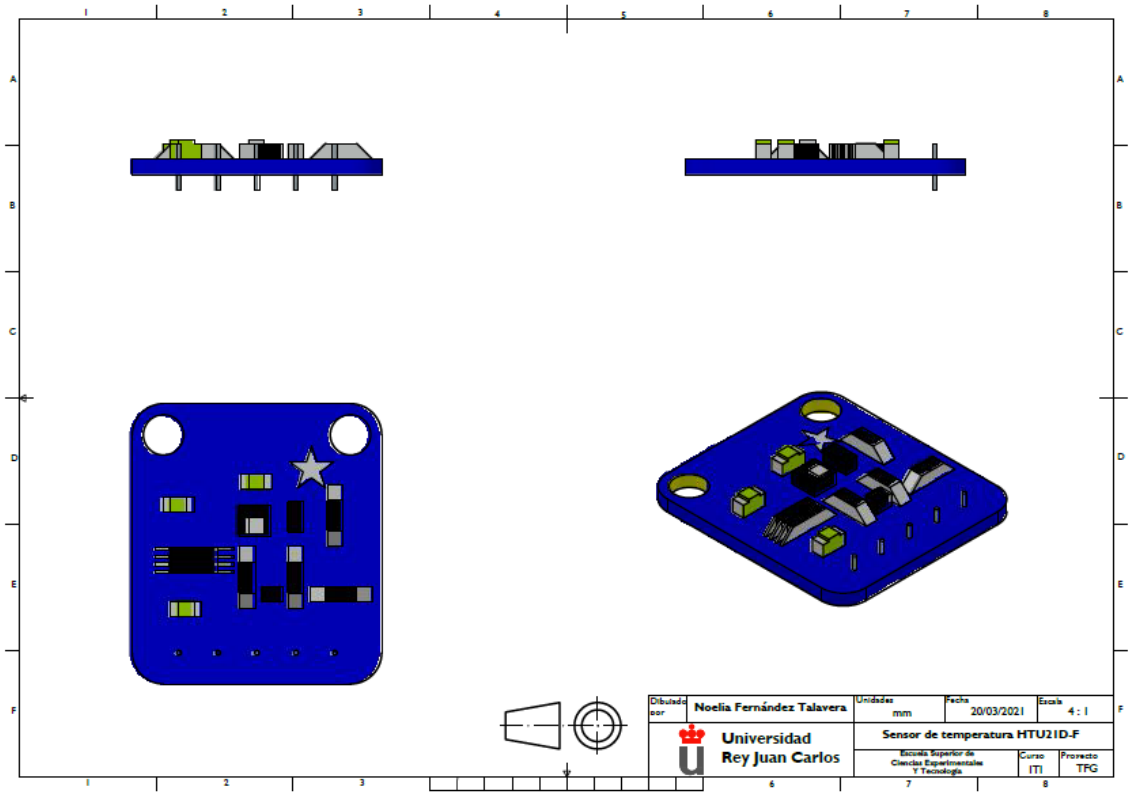
Plano 4. Vistas del diseño del controlador de motor. Elaboración propia



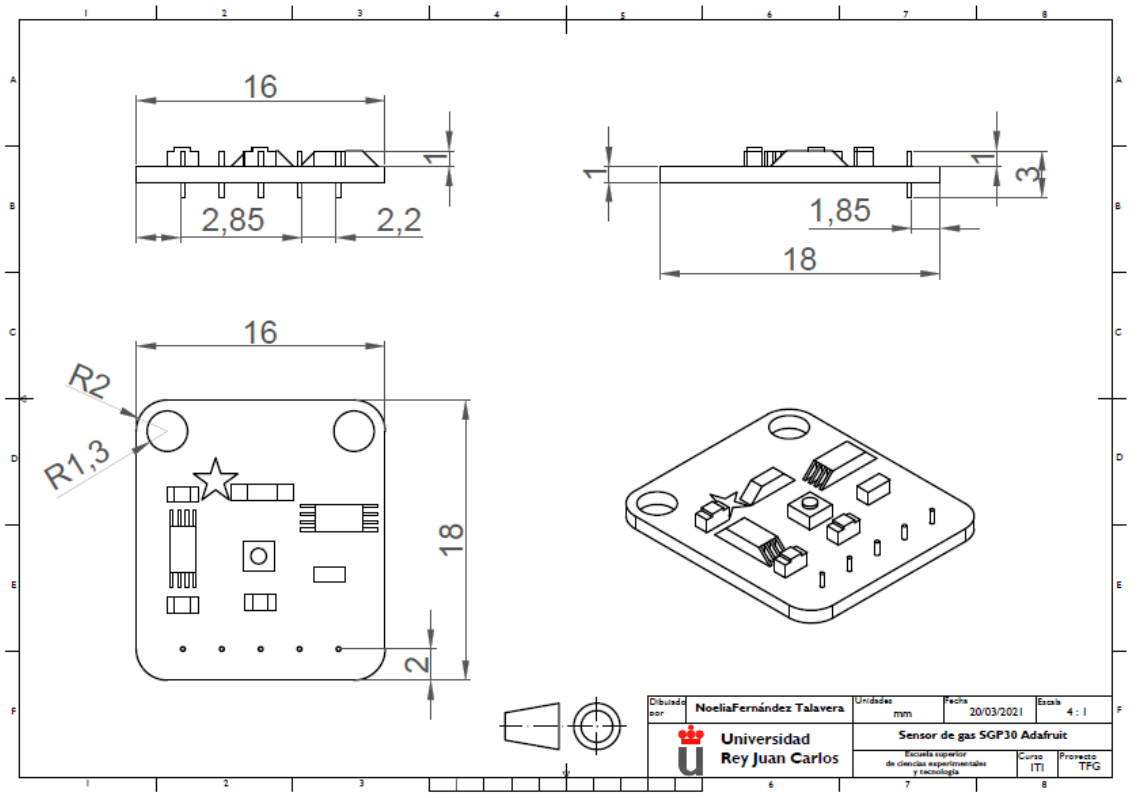
Plano 5. Cotas del sensor de temperatura. Elaboración propia



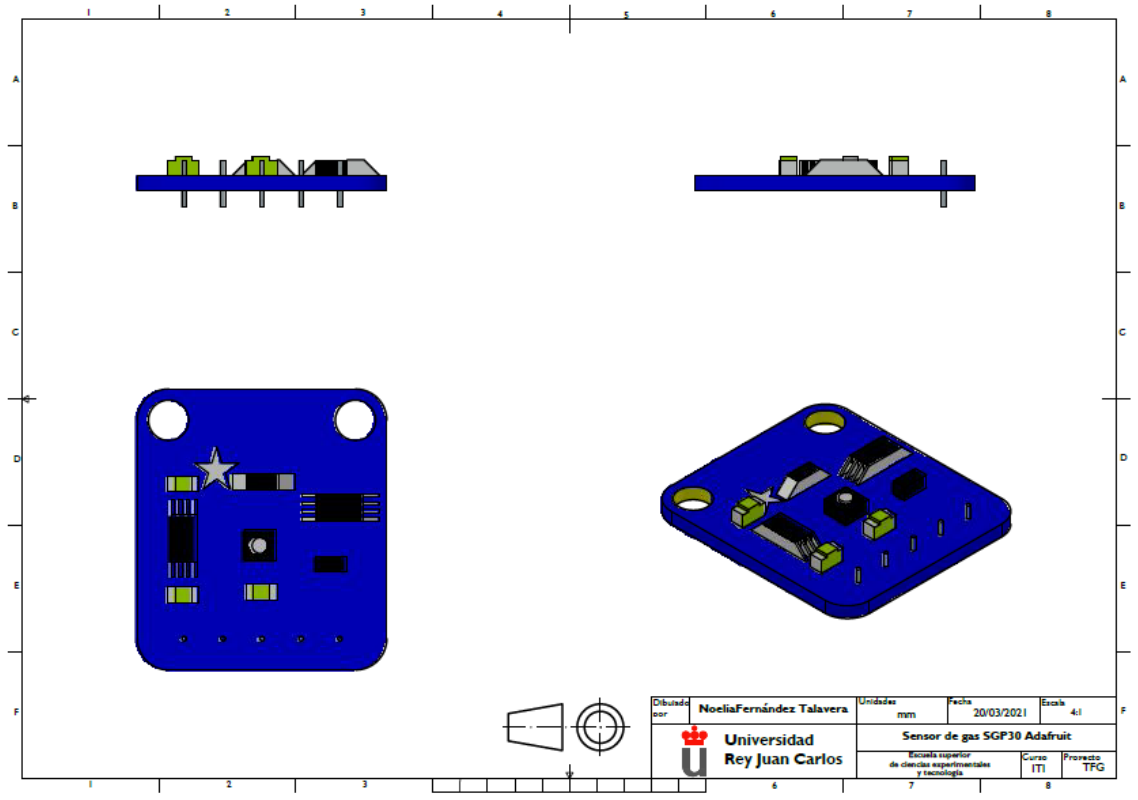
Plano 6. Vistas del diseño del sensor de temperatura. Elaboración propia



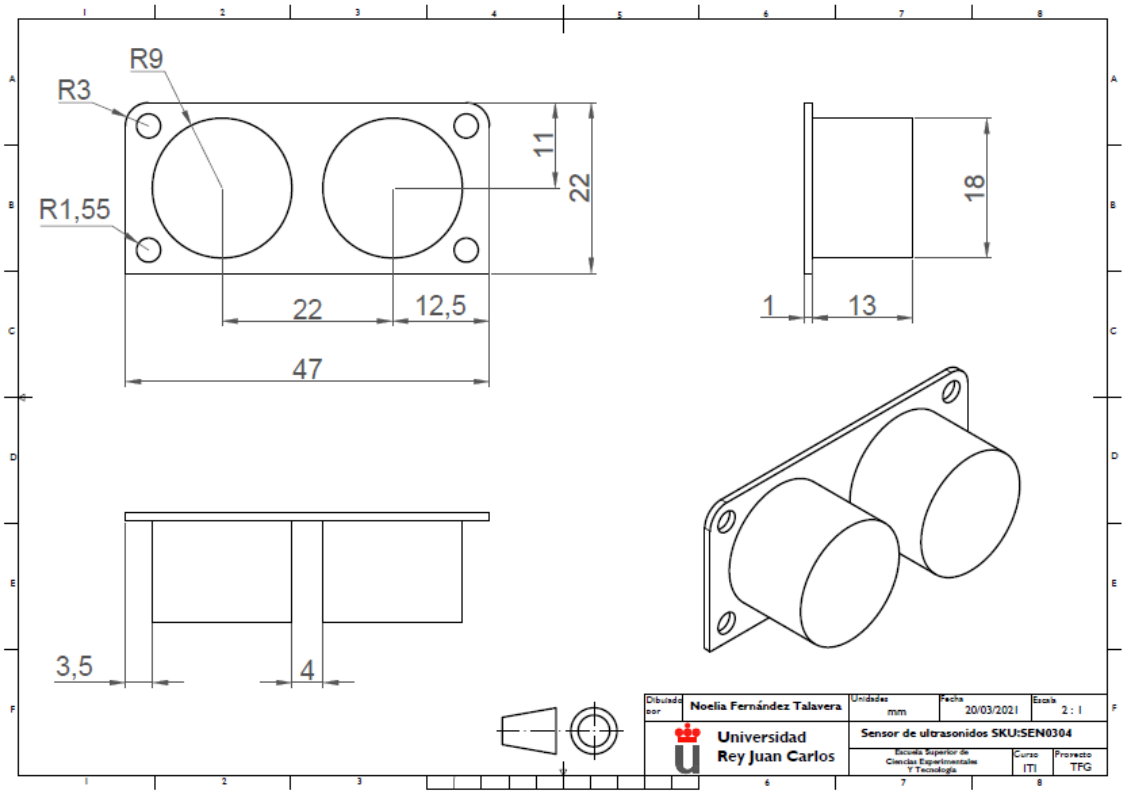
Plano 7. Cotas del sensor de calidad de aire. Elaboración propia



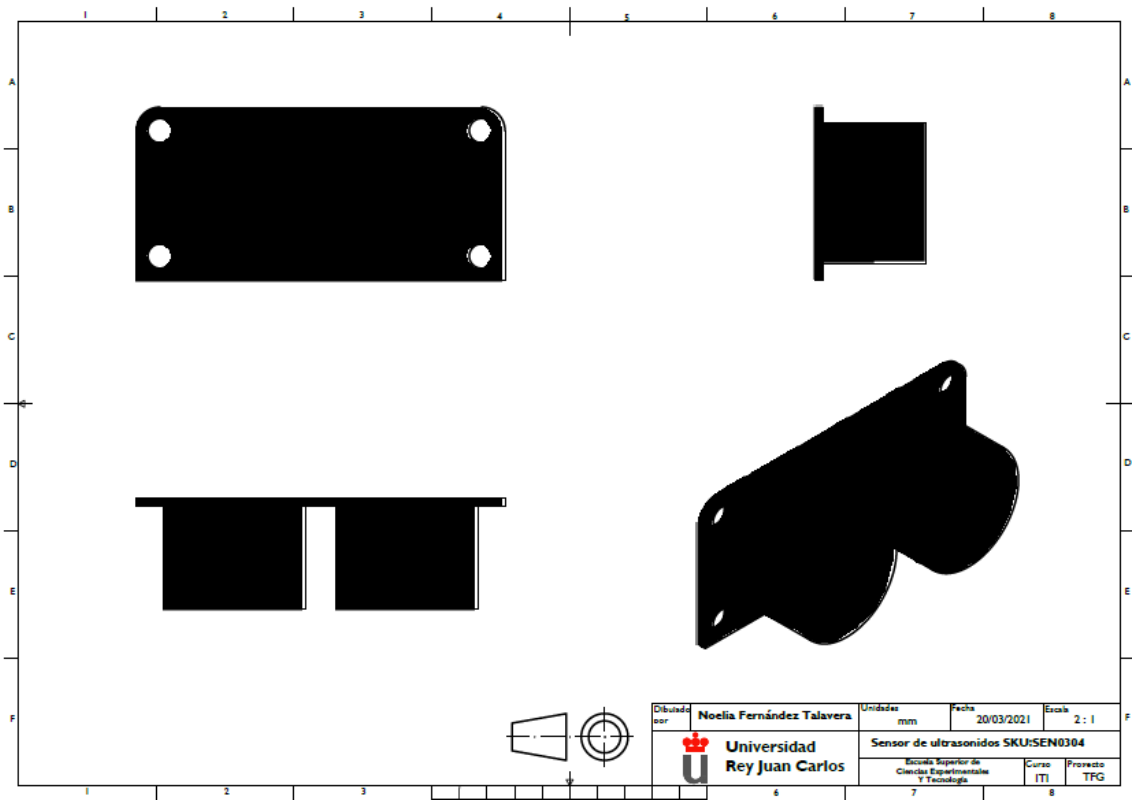
Plano 8. Vistas del diseño del sensor de calidad de aire. Elaboración propia



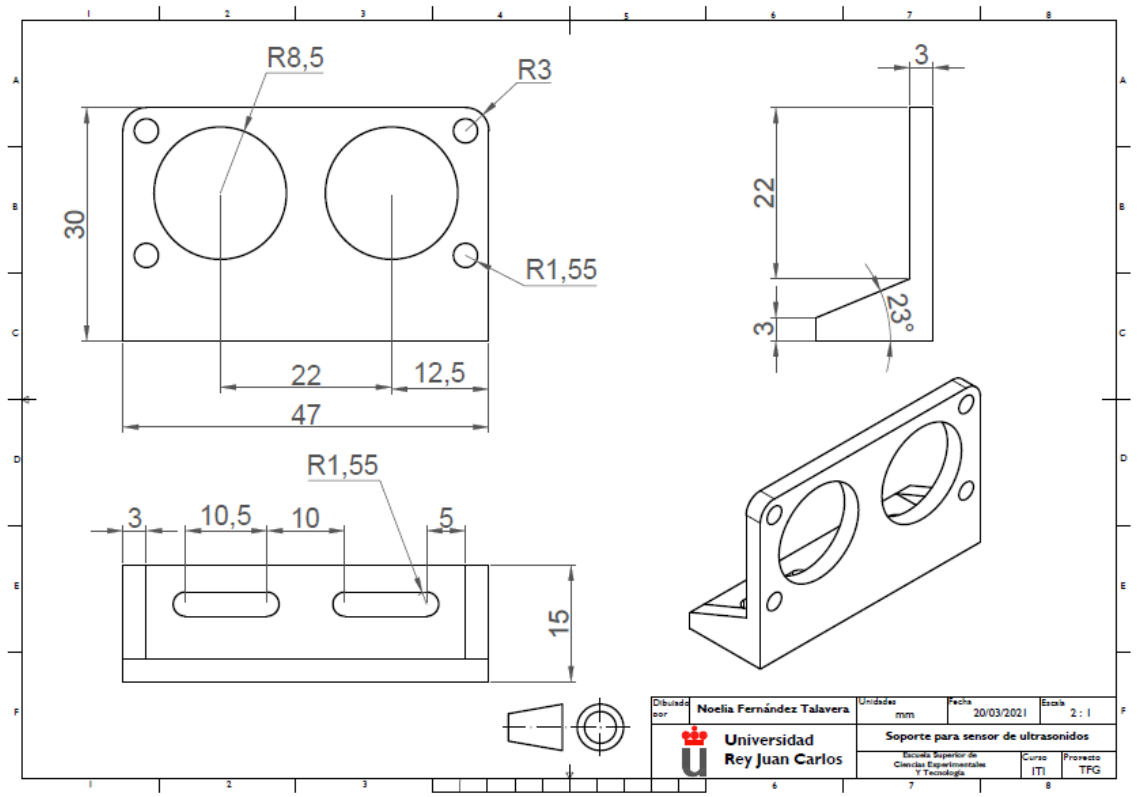
Plano 9. Cotas del sensor de ultrasonidos. Elaboración propia



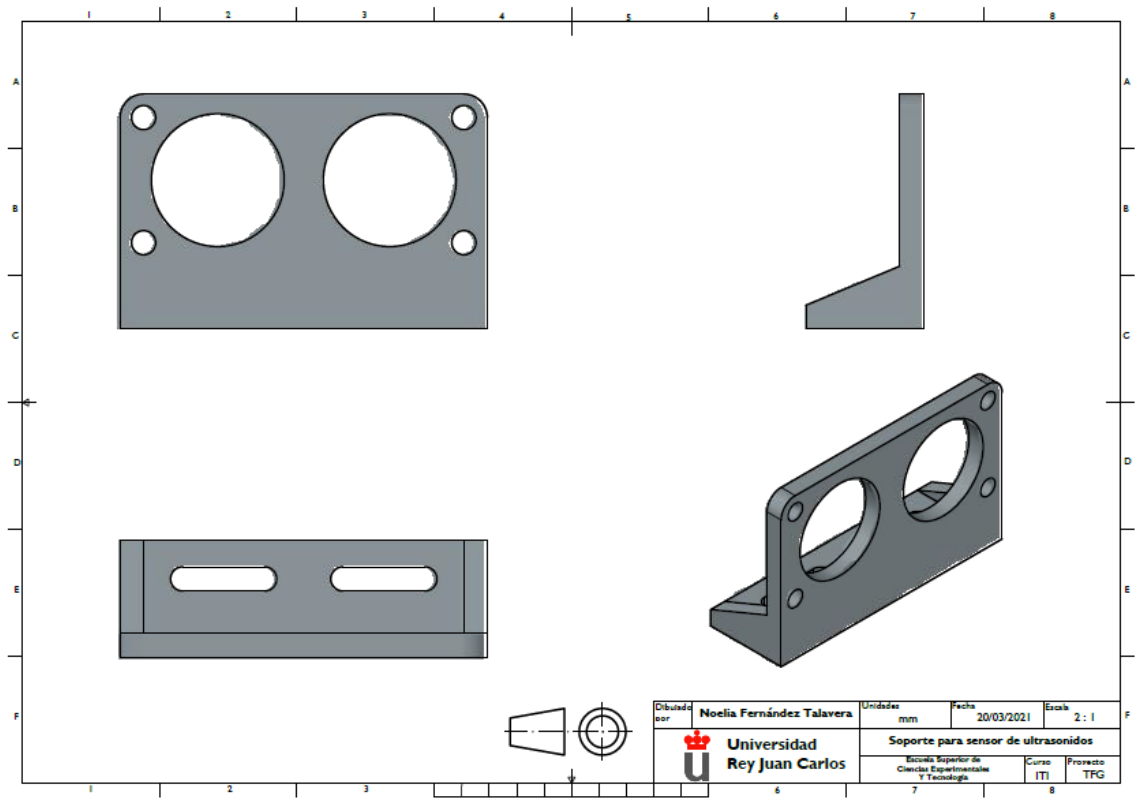
Plano 10. Vistas del sensor de ultrasonidos. Elaboración propia



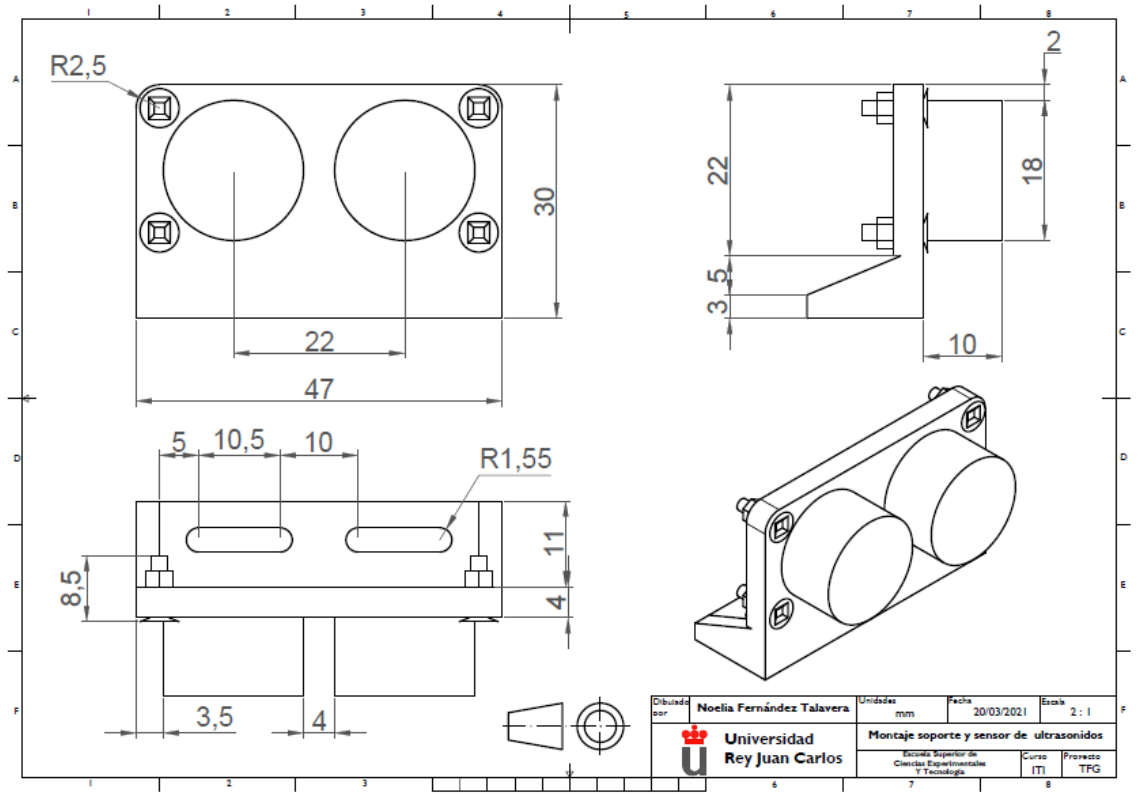
Plano 11. Cotas del soporte del sensor de ultrasonidos. Elaboración propia



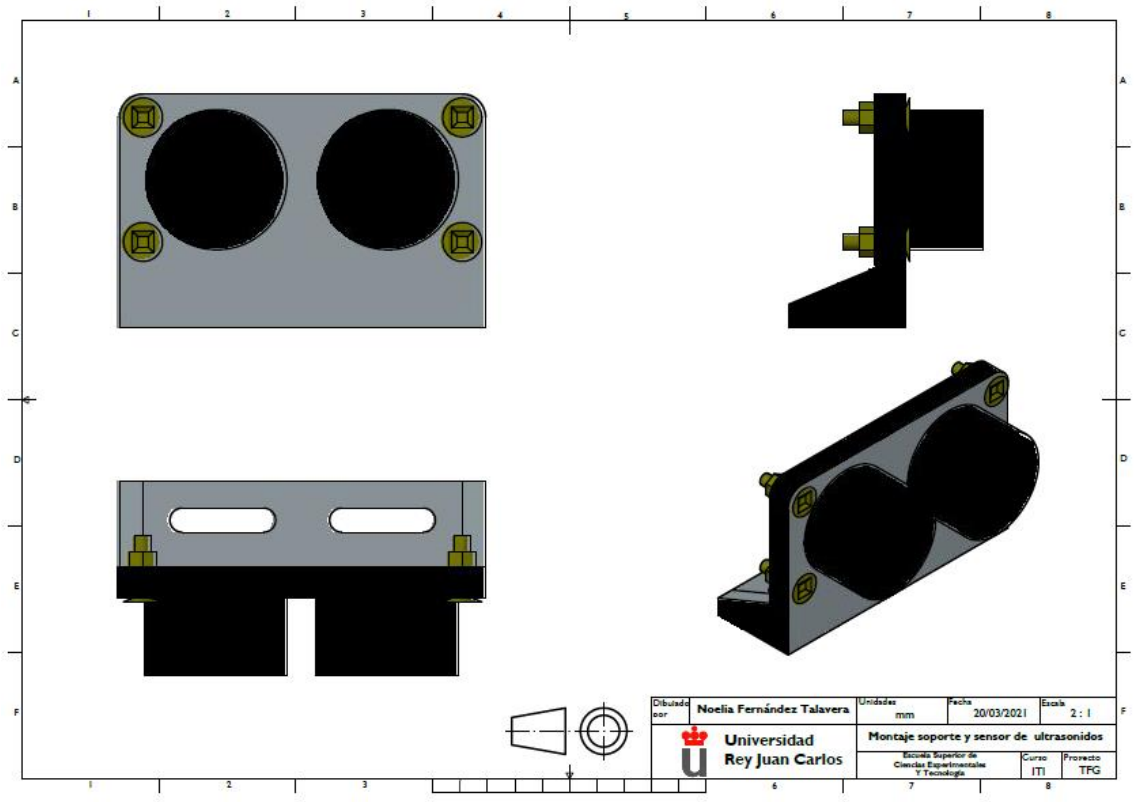
Plano 12. Vistas del diseño del soporte del sensor de ultrasonidos. Elaboración propia



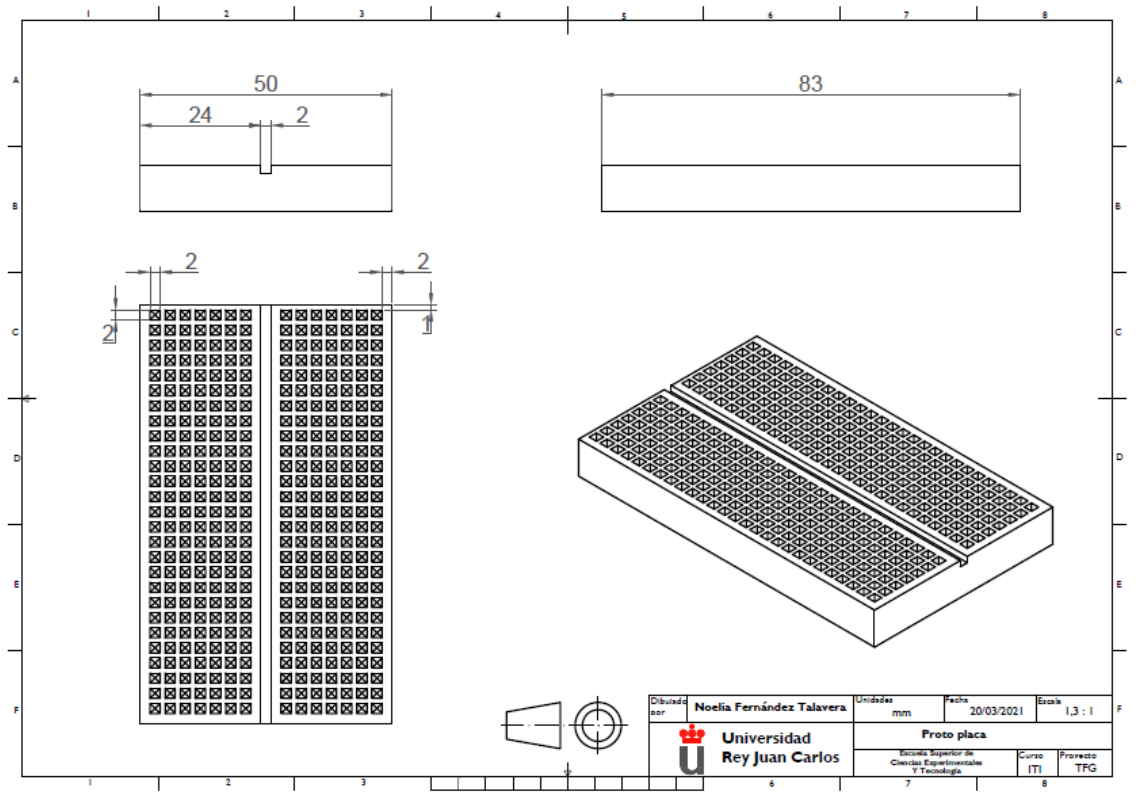
Plano 13. Cotas del montaje del sensor de ultrasonidos y el soporte. Elaboración propia



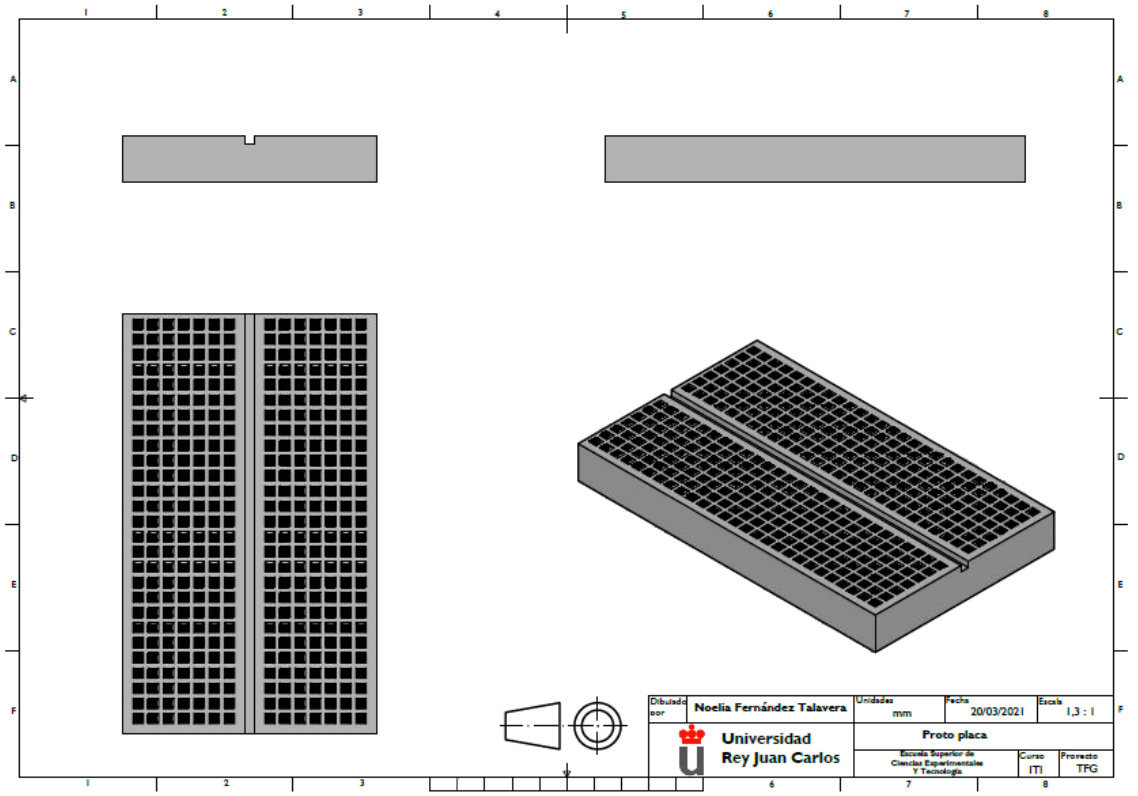
Plano 14 .Vistas del diseño del sensor de ultrasonidos y el soporte. Elaboración propia



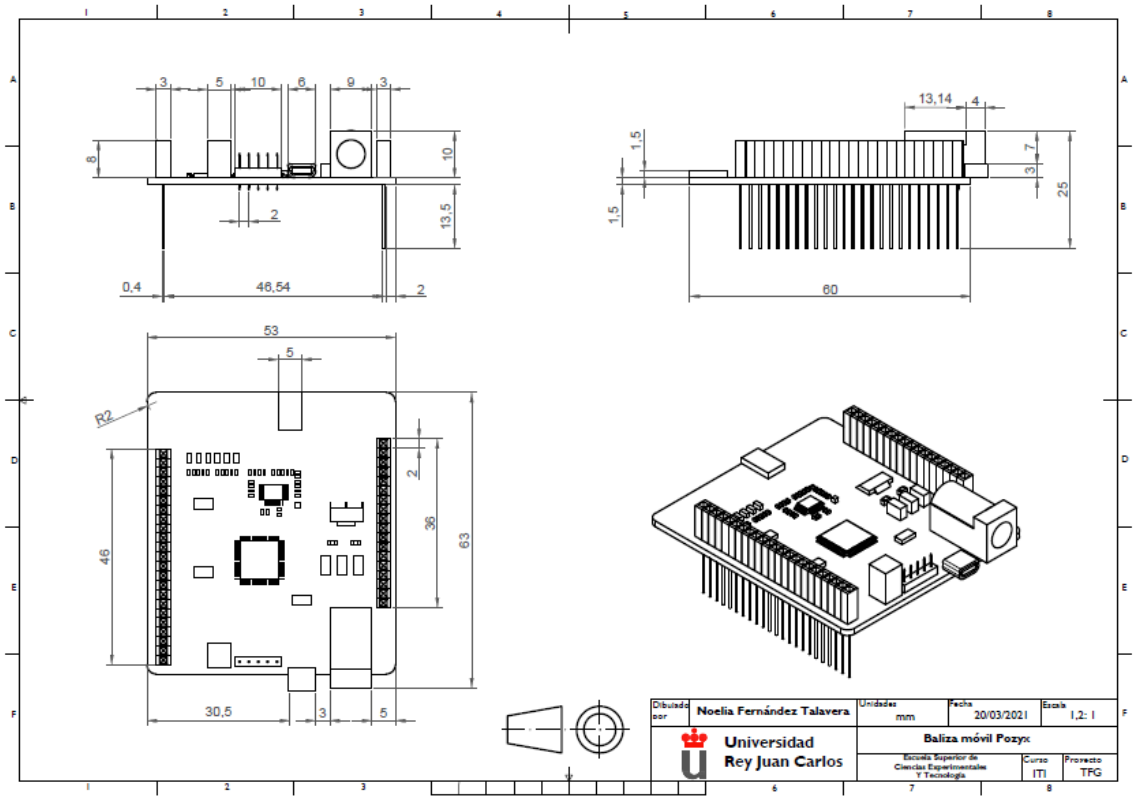
Plano 15. Cotas de la proto placa. Elaboración propia



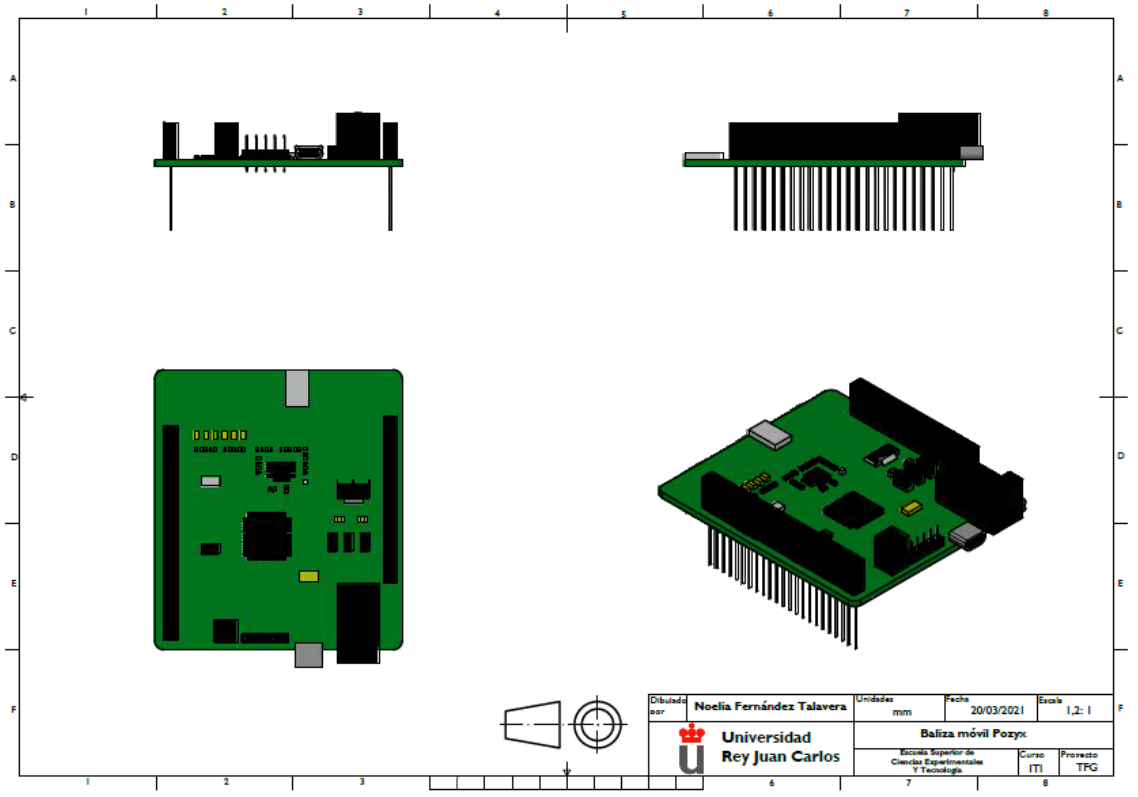
Plano 16. Vistas del diseño de la proto placa. Elaboración propia



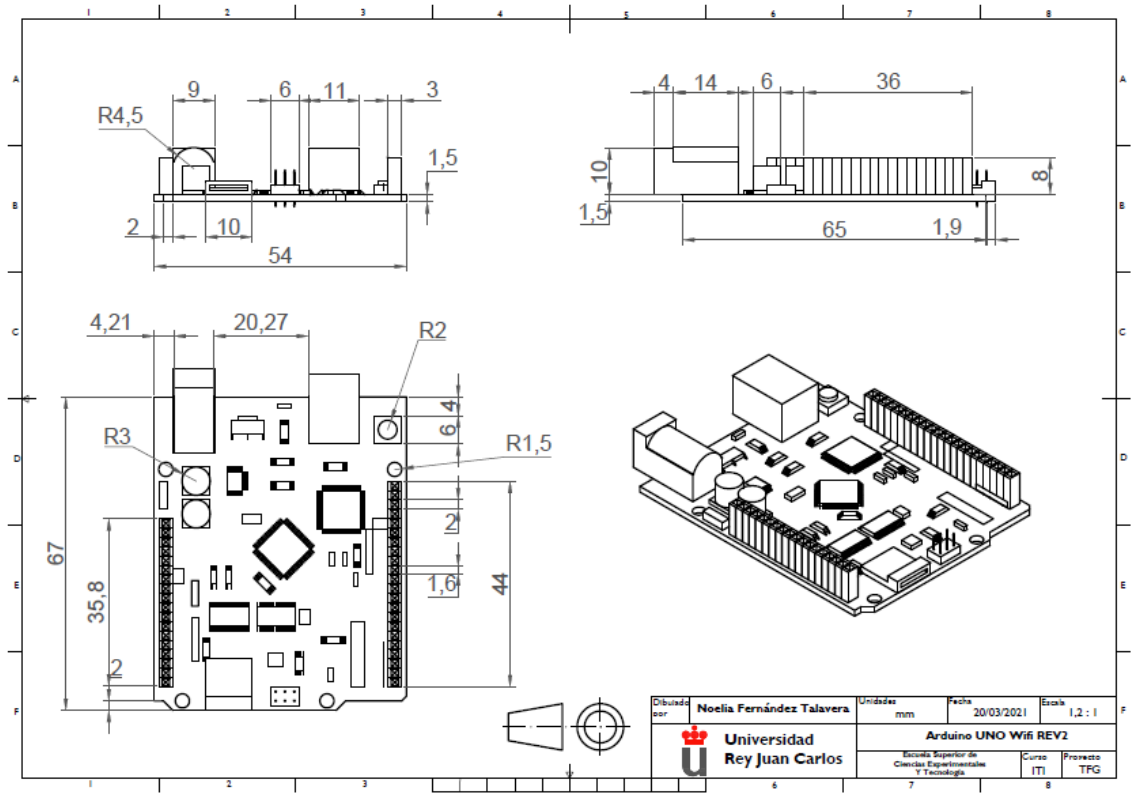
Plano 17. Cotas de la baliza Pozyx. Elaboración propia



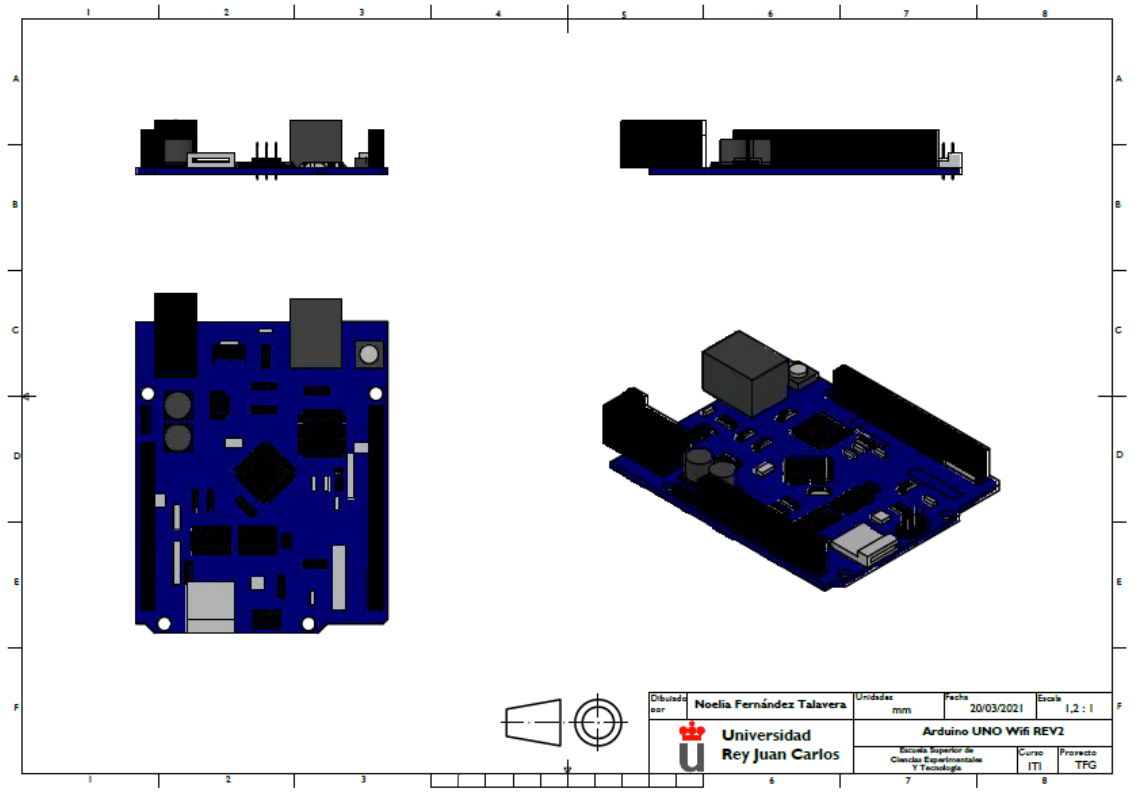
Plano 18. Vistas del diseño de la baliza Pozyx. Elaboración propia



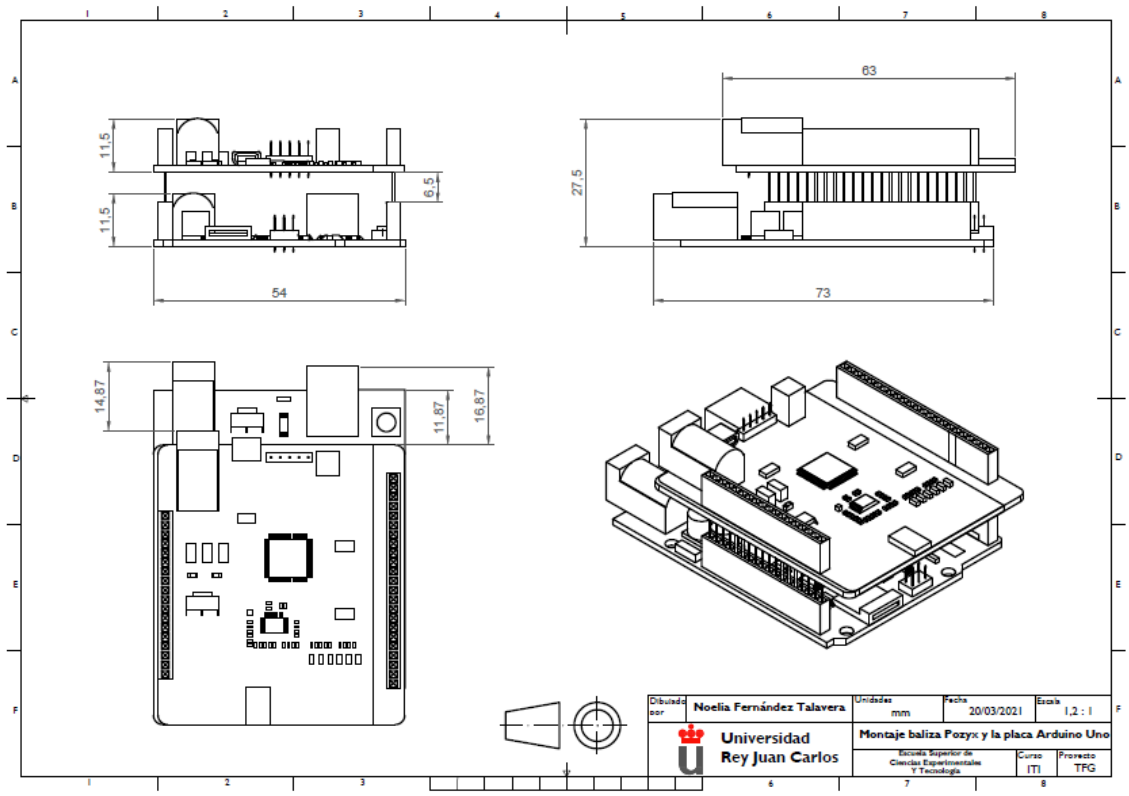
Plano 19. Cotas de la placa Arduino UNO Wifi REV2. Elaboración propia



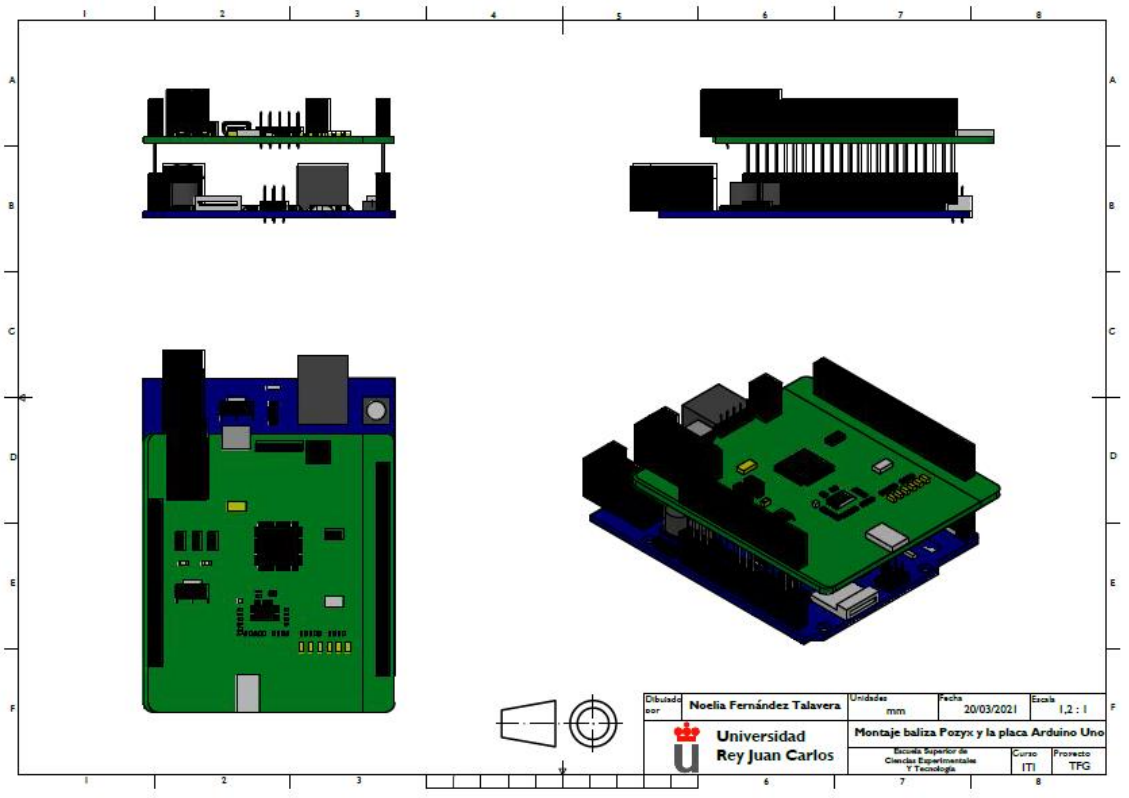
Plano 20. Vistas del diseño de la placa Arduino UNO Wifi REV2. Elaboración propia



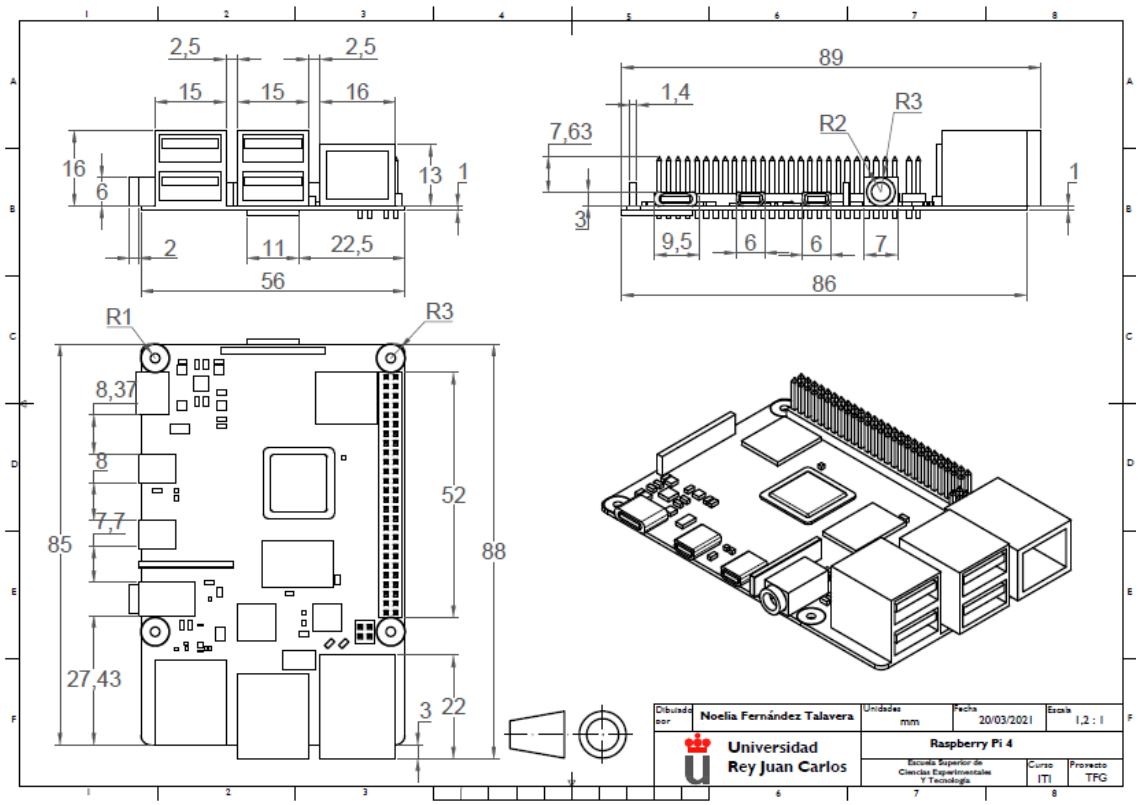
Plano 21. Cotas del montaje de la baliza Pozyx y la placa Arduino UNO Wifi REV 2. Elaboración propia



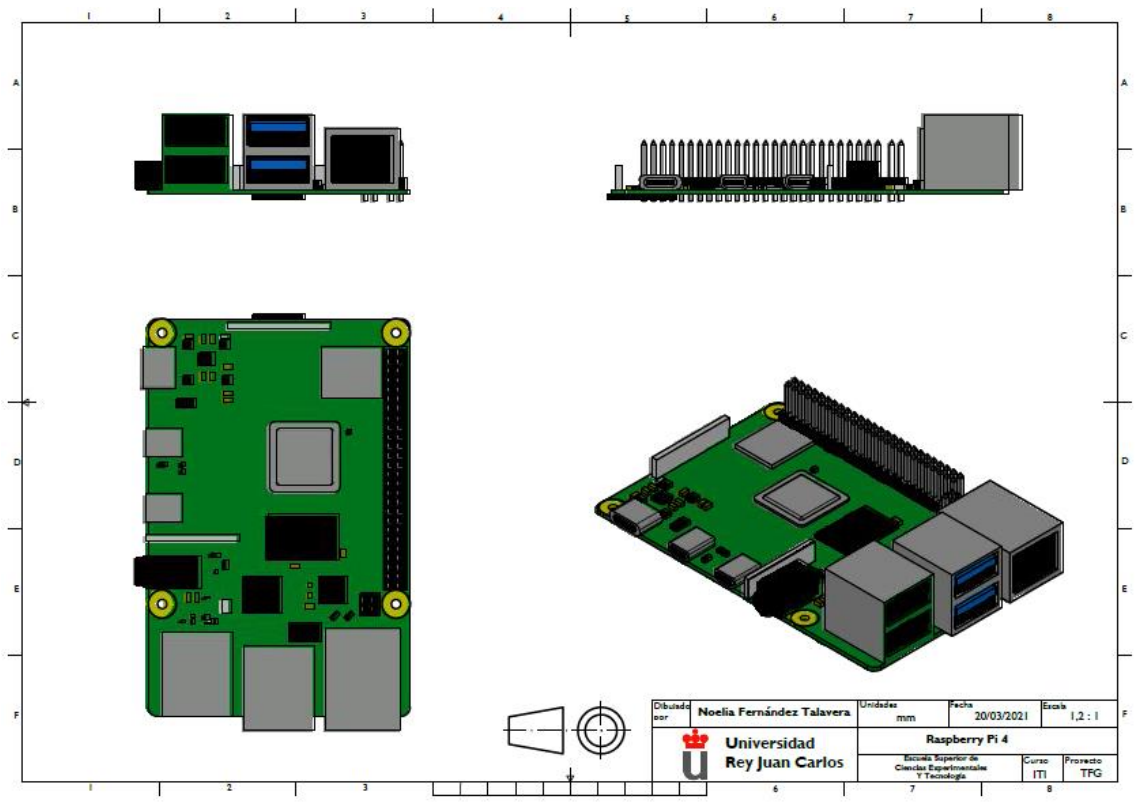
Plano 22.Vistas del diseño del montaje de la baliza Pozyx y la placa ArduinoUNO Wifi REV 2. Elaboración propia



Plano 23. Cotas de la Raspberry Pi 4. Elaboración propia



Plano 24. Vistas del diseño de la Raspberry Pi 4. Elaboración propia



Plano 25. Torre de bomberos de Alcorcón.