Universidad Zaragoza

Mejora de un robot móvil para la implementación de algoritmos de control distribuidos

Improvement of a mobile robot for using with distributed control algorithms

Autor: Josué García Ramo
774767

Directores: Cristian Mahulea y Carlos Gracia

2022

Contents

1 Introducción

1.1 Resumen

Hoy en día, resulta de lo más común ver, trabajar o hablar de robots. Un robot es una entidad artificial que puede ser digital o física. El uso de la palabra robot nació en 1920 cuando el dramaturgo checo Karel Čapek quiso denominar a unos humanos artificiales orgánicos construidos con el fin de aligerar la carga de trabajo del resto de personas. La palabra *robota* significa literalmente trabajo o labor y figuradamente "trabajo duro" en checo y muchas lenguas eslavas[11]. No parte esta palabra de un origen equivocado pues, efectivamente, los robots están diseñados tanto para rebajar la cantidad de trabajo de las personas como para atenuar la dureza del mismo. Hoy en día podemos encontrar robots en cada ámbito de la vida cotidiana: transporte, salud, cocina, recreación y ocio, etc.

Con la llegada de los robots, el ser humano ha podido dedicarse a resolver otros problemas presentes en a sociedad, no solo de carácter tecnológico sino también medioambiental y social. Invertir y trabajar en el desarrollo de la robótica es trabajar por la innovación, la sostenibilidad y la igualdad entre aquellos que no pueden disfrutar de los beneficios de la robótica y la tecnología, haciéndolas cada día más asequibles y trabajando en herramientas de educación en la materia que todo el que quiera acceder a este privilegio tecnológico lo tenga al alcance de su mano.

Figure 1: Objetivos de Desarrollo sostenible involucrados en el trabajo

Este TFG quiere participar de este acercamiento a la robótica. Este trabajo se lleva a cabo dentro del Departamento de Informática e Ingeniería de sistemas de la Universidad de Zaragoza para dar lugar a un espacio donde poder investigar y trabajar en mejorar el estado del arte de la materia en cuestión.

Este trabajo pretende ser una continuación del Trabajo de Fin de Master: *Diseño e implementación de un algoritmo que evite colisiones en un sistema multi-robot utilizando el Modified Banker's Algorithm* por Jose Benigno García Barreto [1].

El resultado final del mencionado trabajo ofrece una plataforma (fig. 2) con tres robots móviles (fig. 3) cuya posición se estima a partir de las imágenes que capta una cámara (fig. 4) posicionada pocos metros sobre la plataforma. A partir de las tres posiciones estimadas un ordenador central decide las rutas óptimas, realiza un seguimiento de eventos para evitar colisiones según el *Modified Banker's Algorithm* y se comunica con los robots mediante unos módulos radio frecuencia. Aparte, estos robots son capaces de realizar un control proporcional de su posición.

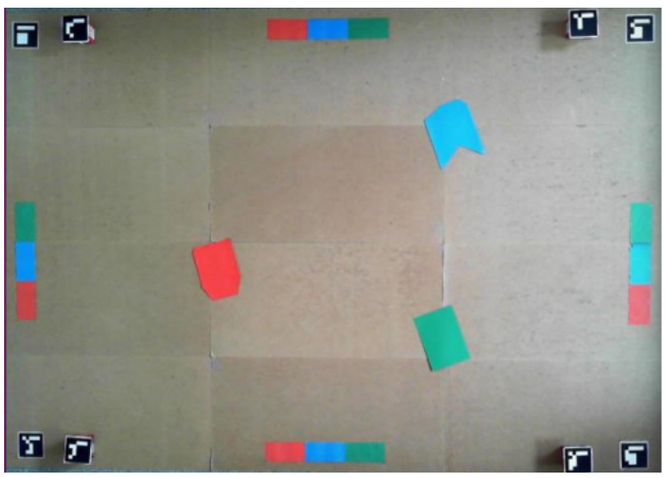

Figure 2: Plataforma sobre la que se desplazan los robots

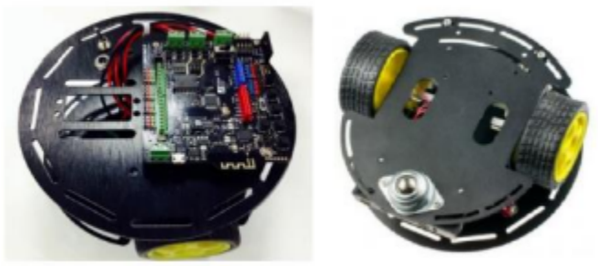

Figure 3: Robots móviles

Figure 4: Sistema de visión de la plataforma: Microsoft LifeCam Studio 1080p HD

Un sistema de control distribuido se caracteriza por tener varios controladores que se dedican a diferentes procesos específicos que en conjunto establecen un sistema de control para toda una planta o, en nuestro caso, toda una plataforma. Este documento trata de recopilar el trabajo realizado para la mejora de estos robots móviles con el fin de darles más autonomía, haciendo posible la implementación de algoritmos de control distribuidos.

Esta continuación introduce distintas mejoras a los robots móviles con tal de sean capaces de recoger más información del entorno en el que se encuentran. Esto requiere tanto la incorporación de sensores como la ampliación de la capacidad de cálculo de los robots. Aparte, también se requiere de la implementación de sistemas de comunicación entre robots y de los robots con el ordenador .

A continuación se detallan el objetivo y alcance del proyecto, así como la metodología seguida para implementar las modificaciones precisas.

1.2 Objetivo del trabajo

Actualmente, la plataforma cuenta, además de con los robots móviles, con un espacio de trabajo por el que se desplazan los robots móviles y una cámara como sistema de visión para localizar los obstáculos en el espacio de trabajo y detectar la posición de los robots[1].

El principal objetivo de esta producción consiste en mejorar los robots móviles ya existentes para que sean capaces de recabar más información del entorno. Por tanto será necesario tanto dotar a los robots de sensores para recoger esta información, como aumentar la capacidad de cálculo del robot. También se pretende cambiar el medio de comunicación entre robots y el ordenador central. Actualmente se comunican por antenas de radiofrecuencia y el objetivo es que puedan comunicarse vía Wifi.

Para recoger información visual de la plataforma y así por ejemplo, evitar la intrusión de los robots en las regiones consideradas como obstáculos, se va a colocar una cámara en cada uno de ellos inclinada sobre la perpendicular al suelo.

De igual manera se implementará un sensor de distancia por ultrasonidos para detectar la presencia de obstáculos con cierta altura como los otros robots, ya que, debido a la no idealidad de las condiciones circundantes, debemos evitar que dos robots colapsen entre ellos si sus rutas coincidieran en algún punto mientras aún no han calculado otro camino alternativo.

Asimismo, con tal de recabar la mayor información posible acerca del movimiento inercial del robot, será implementada una Unidad de Medida Inercial (IMU) que consiste en una placa con un integrado que ofrece información acerca de la aceleración, la orientación y las fuerzas gravitacionales del robot actuando a la vez como giroscopio y acelerómetro.

Para la implementación de todos estos sensores es necesario aumentar la capacidad de cálculo de los robots. Con esa finalidad se contará también con una placa Raspberry Pi 4B capaz de llevar a cabo la recopilación y tratamiento de todos los datos provenientes de los sensores.

Para la comunicación vía Wifi de los robots y el ordenador central se implementará un software diseñado para aplicaciones de robótica: ROS2.

1.3 Alcance del trabajo

La implementación de un sistema de control distribuido en estos robots móviles engloba varias tareas. Primero, la selección de los componentes físicos y seguidamente, la estructuración y ensamblaje de los mismos.

En cuanto al software, se ha programado este control distribuido para el algoritmo ya existente. Para ello es necesario primero configurar la comunicación entre la placa Raspberry Pi 4B y los sensores. La comunicación física se ha llevado a cabo sobre una placa virgen de soldadura. A continuación se elaboraría el programa de control distribuido en el que la información que

recogen los sensores es comunicada al ordenador central que establece las rutas y las manda de vuelta a cada robot.

Por último es necesario configurar la placa Raspberry Pi 4B para que se comunique con otra placa base que presenta un microcontrolador Arduino Leonardo a fin de comunicarle a esta última la acción que es necesario aplicar a los motores en función de la información recabada. La placa Arduino en cuestión es una "DFRobot RoMeo A11 In One COntroller V2.2" la cual presenta un controlador de un motor DC de dos vías con el que resulta muy cómodo el proceso de aplicación de acciones en los motores.

Debido a la gran magnitud del proyecto, este TFG centra en crear un prototipo del robot, con los nuevos elementos implementados y operativos. También se deja constancia de los programas diseñados para el funcionamiento del robot. De igual manera se dejará configurada la aplicación para que el trabajo que continúe este proyecto deba preocuparse únicamente de la programación de la mejora que se pretenda implementar gracias al espacio de trabajo creado en este TFG.

1.4 Fases de desarrollo

La sucesión de pasos necesarios para llevar a cabo la realización de este proyecto se ha resumido en los siguientes puntos:

- Formación en programación orientada a objetos y en los lenguajes de programación C++ y Python3.
- Formación en el proyecto, sus secciones y su funcionamiento.
- Conocimiento del Software y Hardware necesario para el uso de la plataforma Raspberry Pi.
- Formación en la herramienta ROS2.
- Búsqueda de los nuevos componentes físicos.
- Reestructuración de los componentes antiguos e implementación de los nuevos componentes.
- Planificación del mecanismo de control distribuido.
- Configuración de la comunicación entre el IMU y la placa Raspberry Pi 4B.
- Comprobación del funcionamiento de los distintos elementos implementados.
- Planificación de futuras acciones para finalizar el proyecto.
- Redacción de la memoria del proyecto.

2 Marco teórico

2.1 Comunicación SPI

El bus SPI, (*Serial Peripheral Interface*), es una interfaz de comunicación estándar que principalmente se usa para la comunicación entre circuitos inte-

grados de dispositivos electrónicos. Un dispositivo es denominado Maestro o Máster y el resto son llamados Slaves o Esclavos.

El protocolo SPI es síncrono, envía la información bit a bit a pulso de reloj. Es decir, cada vez que el reloj del Maestro manda un pulso el mismo maestro envía un bit. La sincronización y la transmisión de datos se realiza por medio de 4 señales:

- SCLK (Clock): Es el reloj que marca la sincronización. Con cada pulso, se lee o se envía un bit.
- MOSI (Master Output Slave Input): Salida de bits del Maestro y entrada de bits al Esclavo.
- MISO (Master Input Slave Output): Salida de bits del Esclavo y entrada al Maestro.
- SS/Select: Sirve para seleccionar un Esclavo, o para que el Master comunique al Esclavo que se active. A veces este canal no se utiliza, dejando solo 3 señales.

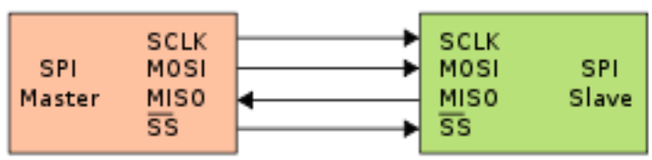

Figure 5: Esquema del funcionamiento del protocolo SPI

En la imagen inmediatamente superior se observa el flujo de información entre el Maestro y un Esclavo. En cuanto a la aplicación de este flujo de información en el hardware, el MISO del Maestro debe conectarse al MOSI del Esclavo y, análogamente. el MOSI del Maestro al MISO del Esclavo. Es la existencia de estas dos vías de comunicación la que dota a este protocolo de la capacidad de comunicación bidireccional. Mientras el Maestro manda información al Esclavo por un canal, el Esclavo puede enviar datos por el otro canal al Maestro.

El maestro es el que se encarga de controlar y generar el reloj. Las dos características del reloj son la polaridad (CPOL) y la fase (CPHA). Estas controlan el borde del reloj activo, desde donde se cronometra el dispositivo secundario en relación con los datos. Fijar la polaridad a cero establece el reloj en inactivo a una lógica 0. Por otro lado, establecer la polaridad en uno presenta el reloj inactivo en la lógica 1. Fijar la fase a cero establece los datos en el reloj en el borde principal, mientras que fijarla a uno establece los datos en el reloj en el borde de rastreo. En la siguiente imagen se muestra un esquema acerca de los modos del reloj en función de su polaridad y su fase[9].

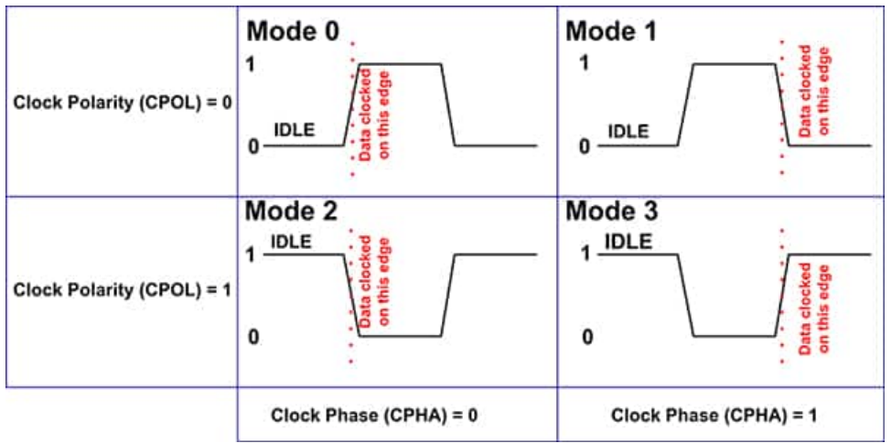

Figure 6: Esquema de los distintos modos del reloj[9]

La opción más frecuente es el Modo 1. Aún así, depende del fabricante establecer en qué modo trabaja el reloj de fábrica. Sin embargo, el programador puede acceder libremente a los otros tres modos.

El uso de este protocolo ofrece varias ventajas sobre otras opciones como conexión I2C o SMBus. Algunas de esas ventajas son: capacidad de comunicación bidireccional, mayor velocidad de transmisión de datos o la flexibilidad que ofrece al poder tener un control absoluto sobre los bits transmitidos pues se puede elegir el tamaño de la trama de bits, su significado y su misión.

3 Metodología de diseño

Al enfrentarse por primera vez a la decisión acerca de cómo proceder para lograr la implementación del mencionado sistema de control distribuido se tuvo que realizar un estudio del mercado de robótica para decantarse por una de los dos siguientes opciones: Adquirir nuevos robots capaces de recabar información de su entorno o adaptar los robots existentes para ello mediante la implementación de sensores y de una placa con un microcontrolador capaz de coordinar la información proveniente de ellos.

3.1 Entre la adquisición o la adaptación

Al buscar un robot adquirible para esta tarea, se tuvo en cuenta que fuera compatible con Raspberry Pi y ROS, dos sistemas operativos de robótica que se consideran clave para el desarrollo de este proyecto y que se describirán en esta misma memoria más adelante. Uno de los sitios web más adecuados para la búsqueda era Robotics College[1], una página especializada en la venta de robots con fines académicos. La opción más acorde a las demandas del proyecto era el Yahboom Transbot.

Este robot cumple con todas las especificaciones: cuenta con una cámara HD, una Raspberry Pi 4B de 4GB y es compatible con ROS. El precio de mercado de esta versión del robot es de 929$.

[1]https://category.yahboom.net/collections/r-college

Figure 7: Yahboom Transbot

Esta opción, aunque muy cómoda, resulta demasiado costosa a la vez que no deja lugar a adaptaciones futuras con la misma facilidad que lo haría el tener un diseño propio. Se decidió, por tanto, estudiar la viabilidad de adaptar el robot a estas nuevas necesidades. La primera gran desventaja de esta opción son los grandes tiempos de entrega de componentes electrónicos debido a la ruptura de stock que muchas empresas están experimentando actualmente y que podían resultar en una prolongación considerable del TFG. Seguidamente, se siguió estudiando dicha opción paso a paso, elemento a elemento, seleccionando los componentes más convenientes para cada una de las tareas del control.

La adquisición de una placa Raspberry Pi era imprescindible. Se precisó la necesidad de una Raspberry Pi 3B+ de 4GB para para poder soportar el sistema operativo ROS. Pero después de varias investigaciones, se decidió que el uso de ROS2 era más adecuado para la tarea y además facilita la implementación de futuras mejoras. Para soportar el nuevo sistema ROS2 se decidió entonces adquirir la siguiente versión de la compañía, la Raspberry Pi 4B de 2GB, que dependiendo del distribuidor podía costar entre 70 y 120 €.

Para la detección de elementos externos en la aplicación se concluyó que serían necesarios tanto un sensor de distancia como una cámara que reconozca las distintas regiones de la plataforma. Un sensor de distancia común en el mercado no cuesta más de 4 € y el precio de un módulo de cámara adecuado para la tarea con su soporte varía entre los 25 y los 40 € dependiendo del distribuidor.

Para el control del movimiento y la posición del robot, se pensó que sería muy apropiado cambiar los motores que presentaban los autómatas para sustituirlos por motores paso a paso. Los motores paso a paso pueden girar una determinada cantidad de grados dependiendo de la entrada proporcionada y de igual manera pueden proporcionar con precisión la cantidad de vueltas que han dado para calcular el espacio recorrido por el robot desde el punto de inicio. Al estudiar el mercado de dichos motores, se concluyó que un motor

capaz de mover una rueda del robot supondría un coste de más de 20 € que, en conjunto con su análogo, acercarían el precio a los 50 €. Además, para el uso de estos dispositivos haría falta la instalación de sendos reductores con salida en ángulo recto, lo que aumentaría el precio del conjunto y además necesitaría de un espacio no disponible en el interior del robot.

Debido a la inviabilidad ofrecida por la opción de implementar motores paso a paso se tomó la decisión de investigar si una Unidad de Medida Inercial podría realizar la tarea de llevar a cabo la recopilación de información para el control de la posición a la vez que encajara en los planes de ensamblaje del robot. Resultó que las opciones viables rondaban el presupuesto de 18 a 30 €. Por ello se decidió seguir adelante con la idea de adquirir un IMU.

Realizando un resumen de los elementos adquiridos con sus precios teniendo en cuenta los gastos reales de adquisición el resultado obtenido es el siguiente:

- Placa Raspberry Pi 4B de 2GB (77,5 €) con una batería portátil (18,94 €), una tarjeta de memoria micro SD de 32GB (9,27 €) y un kit de refrigeración (1,31 €)
- Placa adaptador del IMU junto con el microcontrolador (18,02 €).
- Modulo de la cámara Raspberry Pi (26,71 €) y un soporte para el mismo (2,38 €).
- Materiales de ensamblaje tales como una placa virgen de 1 cara para soldar las conexiones por cable (2,3 €), separadores (6 €) y tuercas y tornillos (1 €).
- El sensor de ultrasonidos tiene un valor de unos 2 € que no computa en el resultado final ya que había ya sensores disponibles para la realización del proyecto.

El precio total de adquisición de las piezas asciende a 163,43 €. Un valor del orden de 5 veces menor al propuesto por la anterior opción acerca de la compra del robot. Por ese motivo y por la facilidad de implementación de futuras mejoras, se acordó llevar adelante la adaptación de los robots.

4 La plataforma y sus elementos

La aplicación consta de tres grandes elementos: El espacio de trabajo, los tres robots y el sistema de visión. Estos elementos se hallan explicados anteriormente y además se detallan minuciosamente en la memora referente al anterior proyecto [1].

Aquí, sin embargo, se describirán los principales componentes implementados a la plataforma ya existente: la tarjeta Raspberry Pi 4B, el sensor de distancia, la unidad de medida inercial (IMU) y el módulo con cámara con su soporte. Las principales especificaciones técnicas de cada uno de los elementos se encuentran en detalle en el Anexo I. Para mayor inmersión en las características técnicas de cada uno de ellos, quedan referenciadas las hojas de datos de cada producto en su correspondiente subsección del Anexo I.

Así pues, tras el montaje y la implementación de las nuevas mejoras, el robot móvil debería funcionar de la siguiente manera. Los tres sensores recabarán información del entorno: la cámara aportará información visual, el IMU información inercial y el sensor de ultrasonidos, información de la presencia de obstáculos. Tras la recopilación de datos, la Raspberry Pi se encargará del cálculo de la acción que es necesaria aplicar, se la comunicará a la Arduino Romeo y esta accionará los motores.

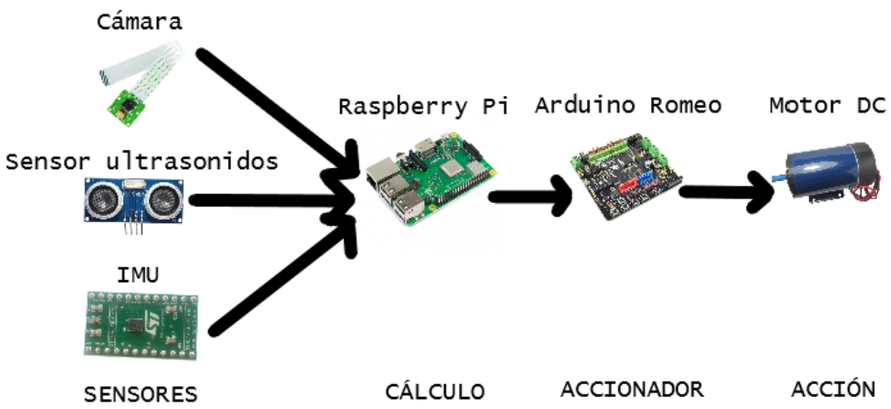

Figure 8: Esquema general de funcionamiento

4.1 Raspberry Pi 4B

La placa SBC Raspberry Pi 4B se diseñó con la intención de estimular el estudio y aprendizaje de la programación y acercar a más gente a el campo de la robótica, muchas veces concebido como demasiado complejo para acceder. Esta placa permite así, de una forma fácil y sencilla, programar el uso de los pines GPIO con opción a conexión serial, I2C y SPI.

Figure 9: Placa Raspberry Pi 4B

La tarjeta Raspberry Pi 4B tiene como componente principal al System on Chip Broadcom BCM2837 compuesto por un procesador ARM-Cortex A72 de cuatro núcleos de 64 bits a una velocidad de 1.5GHz, GPU Broadcom VideoCore IV y Memoria Ram de hasta 4GB, la adquirida presenta 2GB. Además

incluye 2 puertos USB 2.0 y 2 puertos USB 3.0, conexión HDMI, Wifi doble banda, Bluetooth 5.0, Ethernet y mucho más. Gracias a su procesador ARM-Cortex A72 soporta el rango completo de distribuciones ARM GNU/Linux, siendo Raspberry Pi la distribuidora más recomendada. [7].

4.2 Sensor ultrasónico de distancia

El sensor ultrasónico de distancia HC-SR04 proporciona de 2 a 400cm de medida con una precisión de hasta 3mm. Es altamente apropiado para nuestra aplicación debido a su alcance, a su fácil implementación y a su bajo precio.

Figure 10: Sensor ultrasónico HC SR04

Únicamente son requeridos cuatro pines para el control del sensor:

- Fuente de alimentación de 5V.
- Trigger, o gatillo en español, es el encargado de activar el sensor.
- Toma de tierra.
- Echo, eco en español, se conmuta a alto cuando el trigger es activado y devuelve un valor bajo cuando recibe la señal de ultrasonidos.

4.3 Módulo Raspberry Pi Camera V2

El módulo de cámara de la placa Raspberry Pi es de un canal y 8 Mpx. Su frecuencia de captura de imágenes máxima alcanza los 30 fotogramas por segundo. Esta placa de cámara de alta definición es compatible con la placa base que utilizamos y con la que comparte nombre. Debido a su fácil conexión, implementación y uso es la opción más apropiada y asequible para nuestra aplicación.

Figure 11: Raspberry Pi Camera Module V2.1

4.4 Unidad de medida inercial STEVAL-MKI159V1

La placa de adaptador STEVAL-MKI159V1 está diseñada para evaluar su módulo interno iNEMO Inertial LSM9DS1: acelerómetro 3D, giroscopio 3D y magnetómetro 3D. La placa, al conectarse a un zócalo DIL24 estándar, proporciona acceso a la configuración completa del módulo LSM9DS1. Conviene describir las características de dicho módulo.

El LSM9DS1 es un sistema en paquete (system-in-package) que presenta un sensor de aceleración lineal digital 3D, un sensor digital 3D, un sensor de velocidad angular y un sensor magnético digital 3D. El LSM9DS1 incluye un bus serie I2C y un bus serie SPI estándar, lo que nos facilita la comunicación con la placa Raspberry Pi 4B [14].

Figure 12: Unidad de medida inercial STEVAL-MKI159V1

5 Conexiones entre elementos

Una de las tareas más arduas de este proyecto es la conexión de todos los componentes ya mencionados mayormente debido al poco holgado espacio de trabajo que presenta el interior del robot.

Para realizar dicha tarea, es importante tener claro cómo se quiere tener conectados los componentes. Debido a la estructura del proyecto, tanto los tres sensores como la placa Arduino deben estar conectados a la Raspberry Pi. La conexión de la placa Arduino con los motores no es necesaria ya que dichos elementos se encontraban ya conectados, fruto de anteriores proyectos. A continuación se expone el *Pin Layout* de la placa Raspberry Pi 4B:

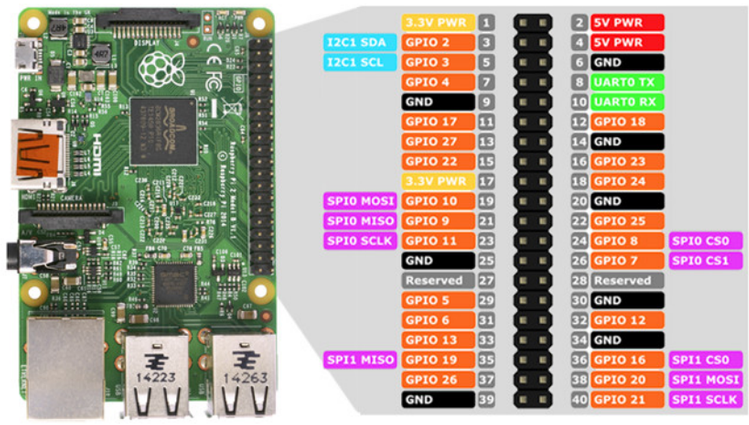

Figure 13: Pin Layout Raspberry Pi 4B[5]

En adelante se hará referencia a cada pin por su correspondiente número, situado en el centro del *Pin Layout* que se muestra en la figura anterior (Fig. 11).

5.1 Conexión del sensor ultrasónico

El sensor ultrasónico de distancia HC-SR04 solamente necesita de la conexión de 4 pines para su correcto funcionamiento. A continuación se muestra un esquema de la conexión ejecutada.

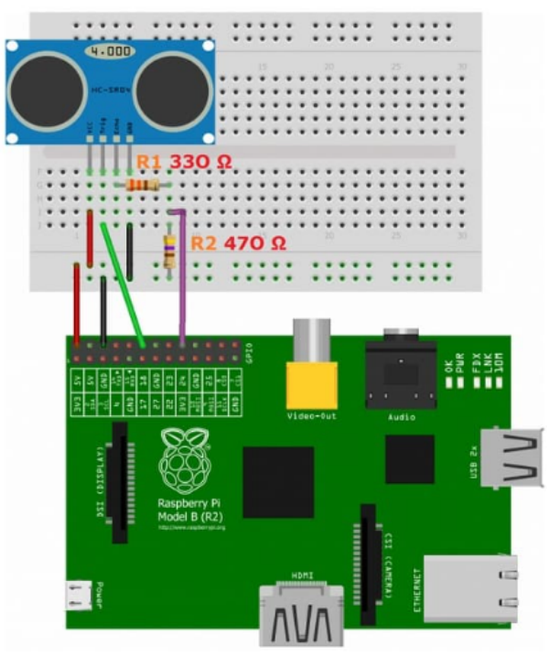

Figure 14: Esquema de la conexión del sensor HC-SR04 con la placa Raspberry Pi 4B[3]

Los cuatro pines del sensor se conectan de la siguiente manera:

- Para la alimentación se ha conectado el pin del sensor POWER al pin 2 de la Raspberry, el cual le proporciona un voltaje de 5V para s funcionamiento.

- El pin de toma de tierra GND del sensor se ha conectado al pin 14 de la Raspberry que también tiene función de toma de tierra.
- El pin del sensor TRIGGER se encuentra conectado al pin 13 de la Raspberry, el cual tiene función de entrada y salida digital.
- El pin del sensor ECHO se encuentra conectado al pin 11 de la Raspberry, el cual tiene función de entrada y salida digital. Se ha colocado una resistencia de 330 Ω entre el ECHO y el pin 11 así como otra resistencia de 470 Ω entre el pin 11 y el pin 14 (GND) para reducir el voltaje de 5V que devuelve el sensor a 3.3V que es el valor que toleran los pines de la Raspberry Pi.

5.2 Conexión del módulo de la cámara

Esta es la conexión más sencilla de todas. Al tratarse de un módulo distribuido por la misma compañía que fabrica la placa Raspberry Pi, esta viene con un puerto ya diseñado para su montaje.

Figure 15: Módulo de la cámara montado en el puerto de la Raspberry Pi[2]

A continuación, se editó el archivo interno de la Raspberry encargado de la comunicación con elementos externos situado en el directorio /boot y con el nombre config.txt para activar la comunicación con la cámara. Se debe añadir la línea de código *start_x=1*. Después de reiniciar el sistema, la cámara es reconocida por el sistema. Podemos observar en la siguiente imagen la prueba de ello. La cámara se encuentra en la última línea de código que devuelve el sistema, bajo el nombre /video0.

Figure 16: Reconocimiento de la cámara por parte del sistema

5.3 Conexión del IMU

Como ya se ha comentado, la comunicación entre el IMU y la Raspberry Pi será vía SPI. También hay que conectarle un pin de alimentación y otro de toma de tierra. La conexión queda de la siguiente manera:

- El pin VDD del IMU se conecta al pin 17 de la Raspberry Pi que lo alimenta con un voltaje de 3.3V.
- El pin GND del IMU se conecta al pin 20 de la Raspberry Pi que lo conecta a la toma de tierra.
- El pin CS.AG del IMU se conecta al pin 29 de la Raspberry Pi que tiene función de pin digital.
- El pin CS.M del IMU se conecta al pin 31 de la Raspberry Pi que tiene función de pin digital.
- El pin SPC del IMU se conecta al pin 23 de la Raspberry Pi que tiene función de SCLK.
- El pin SDO del IMU se conecta al pin 21 de la Raspberry Pi que tiene función de MISO.
- El pin SDA del IMU se conecta al pin 19 de la Raspberry Pi que tiene función de MOSI.

A continuación se ofrece una visualización del *Pin Layout* de la placa STEVAL con los nombres de sus pines en el exterior del recuadro de la placa, siendo los nombres del interior del recuadro las referencias a los pines del microcontrolador iNEMO integrado.

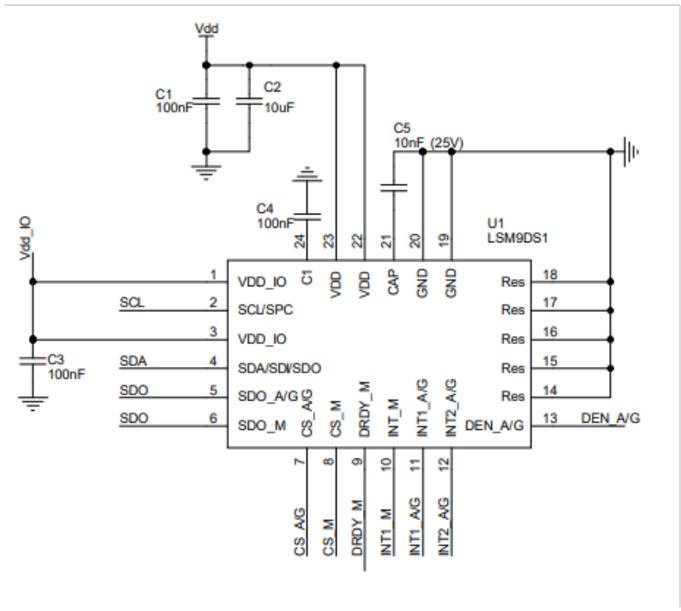

Figure 17: Pinout STEVAL-MKI159V1

5.4 Conexión de la placa Arduino

Para conectar la Raspberry Pi con la placa Arduino haremos uso del segundo bus SPI disponible en la placa Raspberry Pi. En cuanto a la placa Arduino, tiene un bus SPI también disponible con la peculiaridad de que solo usa 3 puertos, no hace uso de un canal SS.

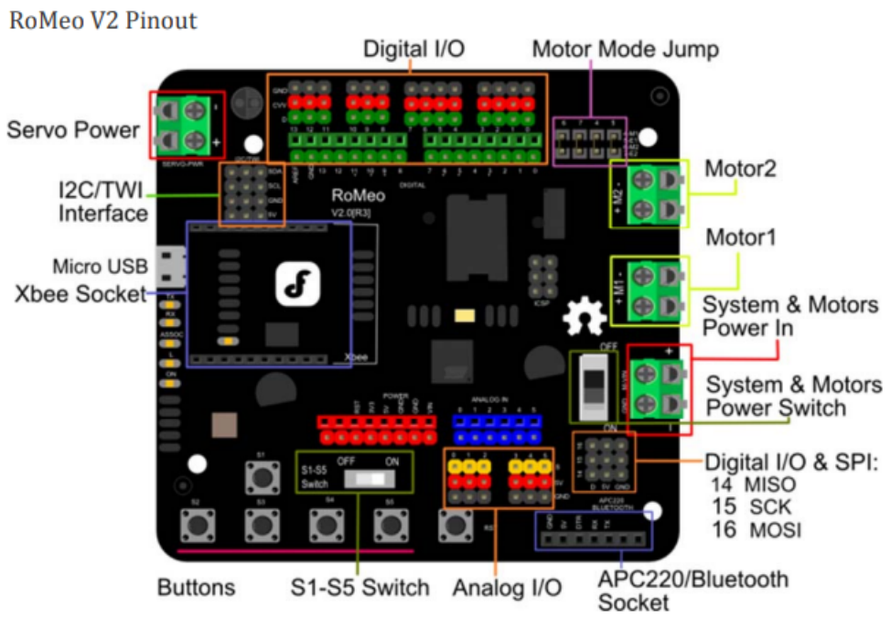

Figure 18: Pinout Arduino Romeo V2

Por tanto la conexión de los 3 pines necesarios para la comunicación SPI es la siguiente:

- El pin MISO de la Arduino se conecta al pin 35 de la Raspberry Pi con

función de MOSI.

- El pin SCK de la Arduino se conecta al pin 40 de la Raspberry Pi con función de SCLK.
- El pin MOSI de la Arduino se conecta al pin 38 de la Raspberry Pi con función de MISO.

La placa Raspberry Pi tiene el bus SPI0 activado de serie, para usar el segundo puerto SPI (SPI1) fue necesario volver a editar el archivo ya mencionado /boot/config.txt y esta vez añadir la línea de código *dtoverlay=spi1-3cs*. Después de reiniciar nuevamente, el sistema habilita tres nuevos canales para el bus SPI1 recién activado. Cuando el sistema los habilita, responde creando los archivos que se muestran en la siguiente imagen.

```
ubuntu@ubuntu:~$ ls /dev/spidev*
/dev/spidev0.0  /dev/spidev0.1  /dev/spidev1.0  /dev/spidev1.1  /dev/spidev1.2
ubuntu@ubuntu:~$
```

Figure 19: Archivos creados al habilitar el bus SPI1

5.5 Ensamblaje

Al trabajar sobre un robot ya montado, es difícil encontrar un equilibrio entre practicidad, adecuación y estética a la hora de ensamblar los nuevos componentes. Con el objetivo de mantener el robot lo más compacto posible, se han diseñado por CAD las distintas partes del robot para estudiar a fondo el espacio disponible. El resultado del diseño, propio del primer enfoque que se le dio a la disposición de los elementos, es el siguiente:

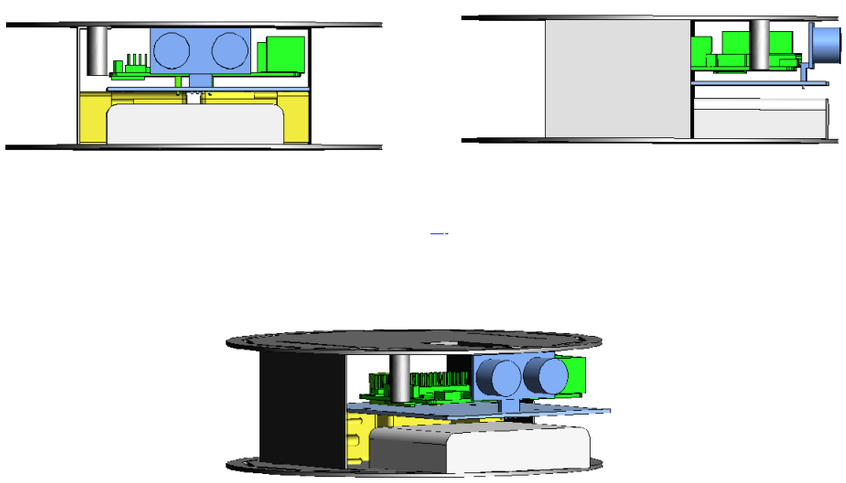

Figure 20: Imágenes primer diseño

Se puede observar el sensor de ultrasonidos en azul al frente del robot. La placa Raspberry Pi en el interior en verde junto con los motores en amarillo.

Además, el banco de pilas que alimenta la Arduino y los motores se encuentra representada en blanco, debajo del sensor de ultrasonidos. Podemos observar también que el espacio superior del robot, que solo ocupaba la Arduino, queda libre para el ensamblaje de la cámara y el IMU.

Además de intentar disponer del mismo espacio que presentaba el proyecto anterior, hay algunas pautas que se han tenido que tener en cuenta a la hora de diseñar la estructuración:

- Accesibilidad a las placas: tanto si que quiere modificar un programa en la placa Arduino como si se quisiera conectar la Raspberry por ejemplo a un monitor para comprobar el funcionamiento o probar nuevos programas, se necesita que ambas tengan sus puertos accesibles. Además la Raspberry debe tener la ranura de inserción de la tarjeta SD de igual manera accesible.
- El sensor de ultrasonidos y la cámara deben ir en la parte delantera del robot para que nada interfiera en sus mediciones. Además, como el soporte de la cámara es metálico, hay que procurar que no interfiera con la señal del sensor de ultrasonidos.
- El IMU tiene que posicionarse en el centro del robot o, mejor dicho, en el punto medio de la linea que une los ejes de rotación de ambas ruedas. Este centro no coincide exactamente con el centro geométrico de la plataforma. El hecho de que el integrado y el eje rotatorio coincidan facilita mucho la tarea de calcular el giro del robot, pues es el que proporciona el IMU con un pequeño factor de calibración.

Teniendo en cuenta todas las pautas anteriores, se llevó a cabo el ensamblaje de los sensores. El primer diseño tuvo que ser ligeramente alterado para adaptarse a las necesidades de espacio conducidas por las limitaciones debidas a el cableado y la accesibilidad a las placas. El resultado tras el ensamblaje con el que se empezó a realizar las pruebas de funcionamiento se muestra a continuación.A la izquierda, se muestra el robot antes de comenzar el proyecto. A la derecha, el prototipo al final del ensamblaje.

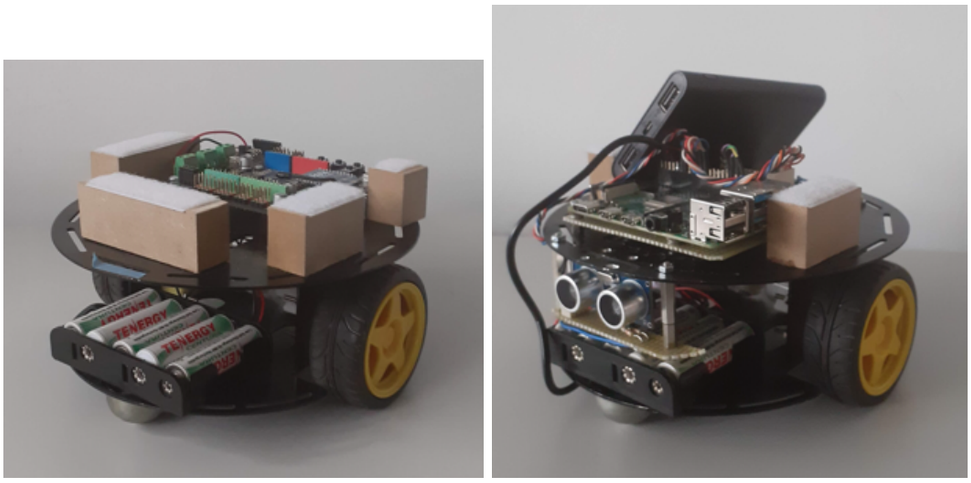

Figure 21: Comparación del robot antes y después del ensamblaje

La placa Arduino se encuentra en el interior del robot, mientras que la placa Raspberry Pi y el IMU están situados en la parte superior. El sensor de

ultrasonidos se encuentra al frente del robot. A falta de montar la cámara y de colocar la batería en el espacio preparado para ella en el interior del robot, este es el resultado del prototipo. Estos dos últimos elementos se encuentran descolocados para mayor facilidad de acceso del cableado necesario a la placa Raspberry Pi durante los ensayos.

6 Software

Una vez que el robot tiene montados los sensores, es importante planificar la comunicación entre los robots y los robots con el ordenador central. Para el estado de la plataforma actual y para futuros proyectos en la misma, se ha decidido trabajar con la herramienta ROS2.

6.1 ROS2

ROS 2 (Robot OperatingSystem 2) es un SDK (Software Development Kit) de código libre para aplicaciones en robótica. La finalidad de ROS2 es ofrecer una plataforma estándar a los desarrolladores de cualquier rama de la industria que les ofrezca un mismo entorno desde las fases de investigación y prototipos hasta desarrollo y producción[10]. Es la versión actualizada de la herramienta original ROS.

El uso de la herramienta ROS2 queda justificado por su gran potencial. La robótica requiere un conjunto de competencias muy diferentes y variadas que suelen estar fuera del alcance de una sola persona. Entre ellas están la gestión del hardware, la gestión de la memoria y los procesos, usar algoritmos de razonamiento selecto, la gestión de concurrencia, el paralelismo y la fusión de datos, así como el uso de la inteligencia artificial. Es aquí donde entra ROS y su comunidad, creando el entorno que nos permite conectar desarrollos de diferentes personas a lo largo del planeta. Estos desarrollos se llaman paquetes. La mayor ventaja de ROS es que puede ejecutar, sincronizar y comunicar de forma paralela estos paquetes, permitiendo que distintos paquetes puedan controlar un mismo robot.

Otra de las razones de su uso es que se trata de un sistema muy liviano que no necesita de muchos recursos y la mayoría de los productos orientados a la robótica son compatibles con ROS2.

Figure 22: Logo ROS2

ROS2 proporciona potentes herramientas para desarrolladores y se ha convertido en un estándar gracias a que es extremadamente modular,la comunicación entre nodos (o programas) es sublime y es compatible con código C++11, C++14, C++17, Python3.5 y Python3.8. Aparte, existen excelentes

herramientas de simulación compatibles con ROS2 que además sirven para controlar aplicaciones multi-robot como la nuestra.

El funcionamiento de la herramienta ROS2 se basa en la comunicación entre nodos. Los nodos son programas con una estructura determinada que realizan tareas independientes. La comunicación entre nodos se lleva a cabo a través de tópicos. Los tópicos son cadenas de información sobre las que un nodo puede escribir datos y otros nodos pueden leer esos datos. Al nodo que envía los datos se le denomina "publicador" y a los que están esperando a leer aquello que se envíe se les llama nodos "suscriptores".

La elección de ROS2 sobre ROS para este proyecto se debe a las nuevas características integradas en esta nueva versión, como la ausencia de un nodo "maestro" que tenga que ser responsable de todas las comunicaciones entre nodos. Otra gran ventaja que ofrece es que trabaja en tiempo real, queriendo esto decir que el sistema de procesamiento de información responderá a estímulos de entrada generados externamente en un período de tiempo y funcionamiento.Por tanto las respuestas correctas dependen no solo de los resultados lógicos sino también del tiempo en que son entregadas[4].

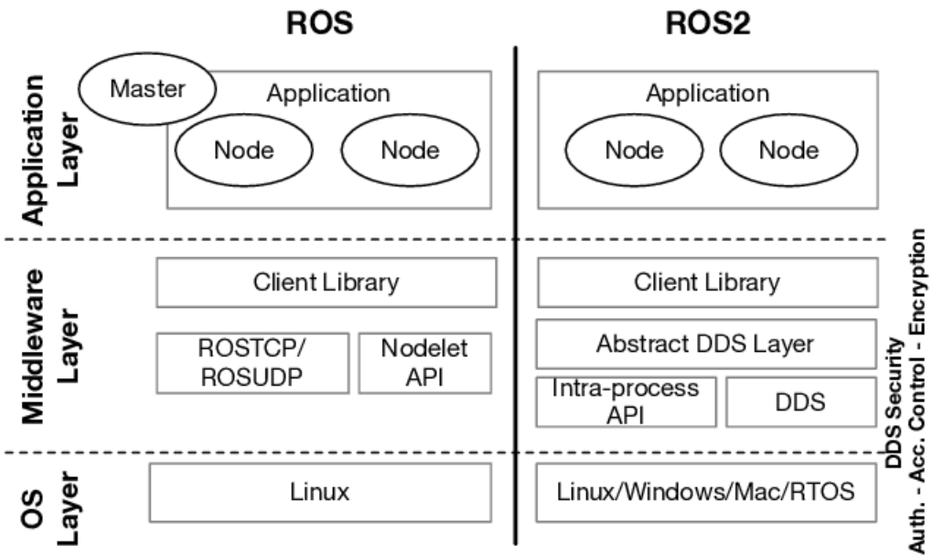

Figure 23: Comparación ROS/ROS2

En nuestro caso, para una comunicación básica entre el prototipo del robot y el ordenador, se pensó utilizar tanto un nodo publicador que establece el punto de destino y lo escribe en el tópico creado para ello como un nodo suscrito a este tópico que lee el punto objetivo y calcula las acciones necesarias para llegar. Estas acciones serían comunicadas a su vez a la placa Arduino para accionar los motores. A partir de esta idea se podrían diseñar más nodos para mejorar la plataforma una vez que este TFG haya dejado la herramienta ROS2 accesible e implementada en el sistema.

Con ese objetivo se instalará ROS2 en la placa Raspberry Pi y se diseñarán dos nodos, un publicador y un suscriptor, para comprobar la correcta implementación de la herramienta en el sistema.

6.2 Ubuntu en Raspberry Pi 4B

Para el funcionamiento correcto de esta herramienta han sido valoradas varias opciones y llevado a cabo muchas pruebas siendo el resultado de ellas que la opción más apropiada es el uso del sistema operativo Linux en la Raspberry Pi bajo la distribución Ubuntu 20.04 servidor, la cual permite hacer uso de ROS2 Foxy, la última versión de ROS2.

Esta distribución no consta de una interfaz de escritorio así que el desarrollo de los programas se ha llevado a cabo desde por medio de dos entornos externos, los cuales habilitan mejor la edición y el manejo de archivos de texto.

El primero de ellos es una máquina virtual administrada desde Oracle VM Virtual Box Administrator en la que se ha instalado la distribución Ubuntu 20.04 Desktop. Para poder editar y compilar los paquetes con mayor facilidad se ha hecho uso del editor de código Visual Studio Core. Para poder enviar los archivos a la Raspberry Pi desde la máquina virtual se ha creado un servidor FTP que permita el intercambio de archivos entre los dispositivos a través de una red local LAN.

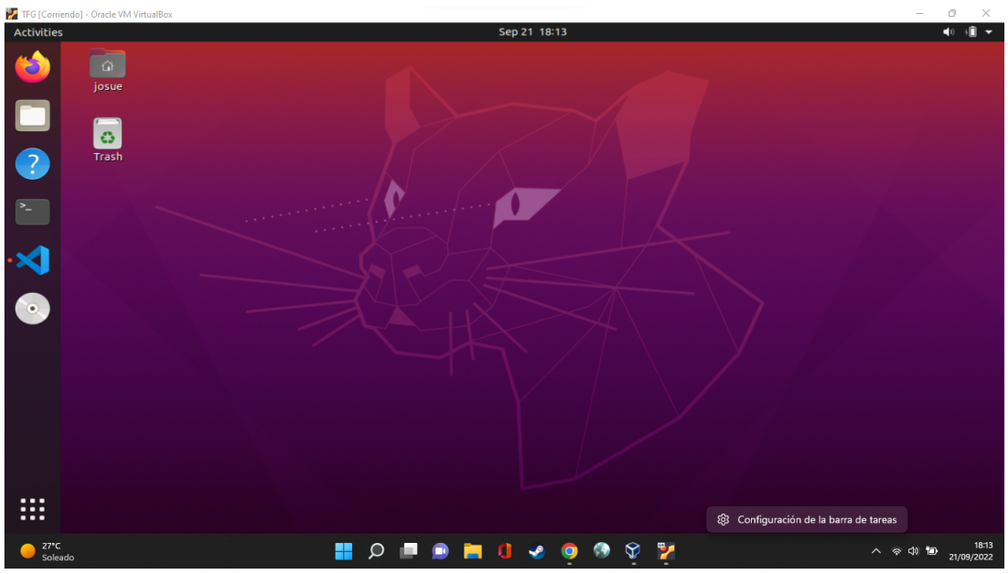

Figure 24: Visualización del entorno Ubuntu 20.04 desktop en la máquina virtual

Esta opción resultó consumir más recursos de los pensados y el dispositivo encargado de ejecutar la máquina virtual empezó perder fluidez durante la instalación de ROS2. Así que una vez creado el servidor FTP, se señaló la opción de acceder a la Raspberry Pi desde un ordenador personal con Windows 11 mediante SSH. Secure SHell (SSH) es el nombre de un protocolo y del programa que lo implementa cuya principal función es el acceso remoto a un servidor por medio de un canal seguro en el que toda la información está cifrada[13]. Para poder acceder mediante SSH al Ubuntu de la Raspberry Pi se necesita conectarse a la misma red. Por tanto, se conectó la Raspberry Pi a la red Wifi disponible en el laboratorio e igualmente se hizo con el ordenador personal.

Para conectarse a una red Wifi desde una terminal Ubuntu hace falta editar el archivo de redes presente en la carpeta /etc/netplan. Es necesario

cumplimentar el archivo con los datos de la red a la que se desea acceder. Al tratarse de un acceso a la red Wifi que no necesita contraseña sino que el servidor reconoce la dirección MAC del dispositivo y le asigna una IP fija, el archivo queda como se muestra a continuación.

```
network:
        wifis:
                wlan0:
                        access-points:
                                        "wiuz":{}
                        dhcp4: true
                version 2
```

Figure 25: Configuración de red para conectarse a red Wifi

Una vez conectados a la misma red Wifi, se pudo volver a acceder al servidor FTP creado para intercambiar archivos con la Raspberry Pi. Esta vez, al acceder al servidor FTP desde Windows 11, se utilizó la aplicación de escritorio Filezilla Client para llevar a cabo el intercambio de archivos desde una interfaz sencilla e intuitiva.

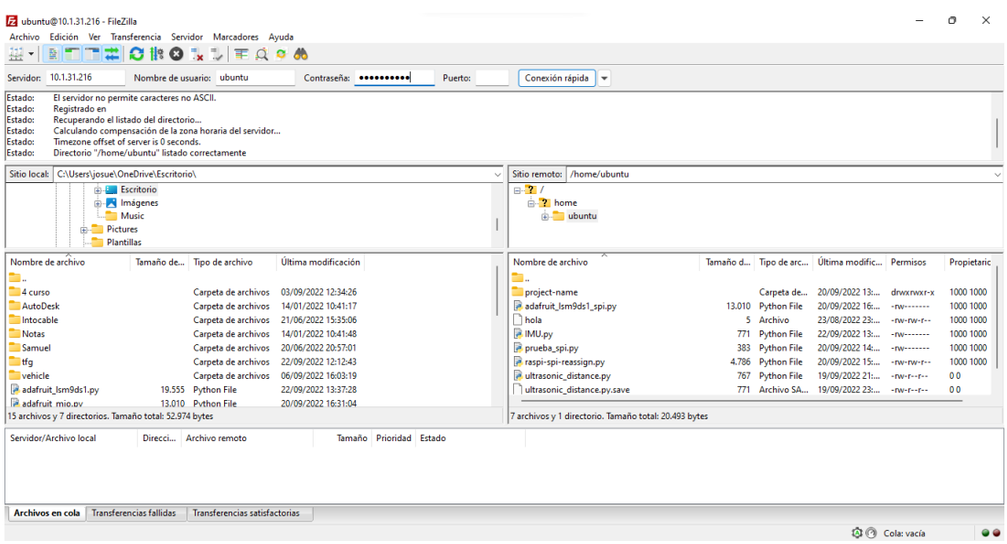

Figure 26: Interfaz de Filefilla Client conectada al servidor FTP

7 Ensayos de funcionamiento

7.1 Ensayo del sensor ultrasónico HC-SR04

Para el ensayo del sensor de ultrasonidos se ha utilizado un programa muy sencillo que muestra la distancia al obstáculo más cercano. Para ello simplemente activa el trigger durante 0.01 ms y recoge el tiempo que pasa hasta que el echo se activa. Multiplicando el tiempo resultante T por la velocidad aproximada del sonido c y dividiéndolo por 2 al considerar que es el tiempo de ida y vuelta hasta el obstáculo, obtenemos la distancia deseada D.

$$D = \frac{T * c}{2}$$

$$D = \frac{T * 34300}{2}$$

Para la programación de los pines de la Raspberry Pi se ha usado la librería *RPi.GPIO*. Esta librería provee un módulo Python para el control de los pines GPIO de una Raspberry Pi[12].

A continuación se muestran los resultados del ensayo de funcionamiento, en el que se ejecuta el programa con el robot mirando hacia una pared y después coloca un obstáculo a apenas unos centímetros para ver si el registro cambia.

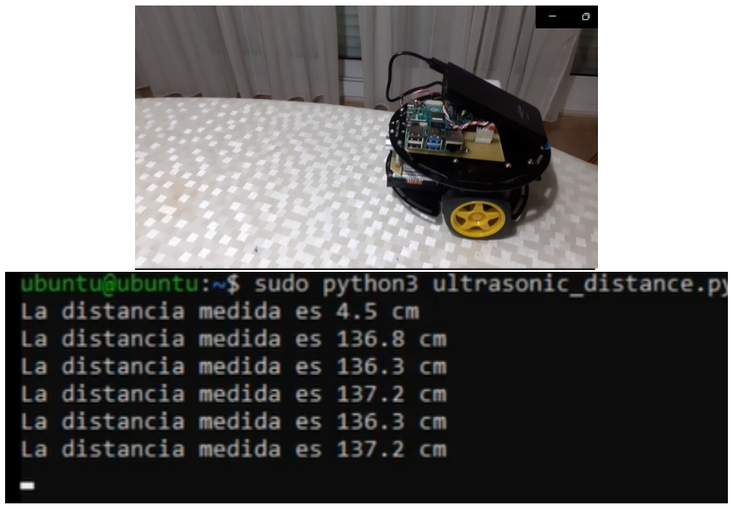

Figure 27: Respuesta del programa sin obstáculo

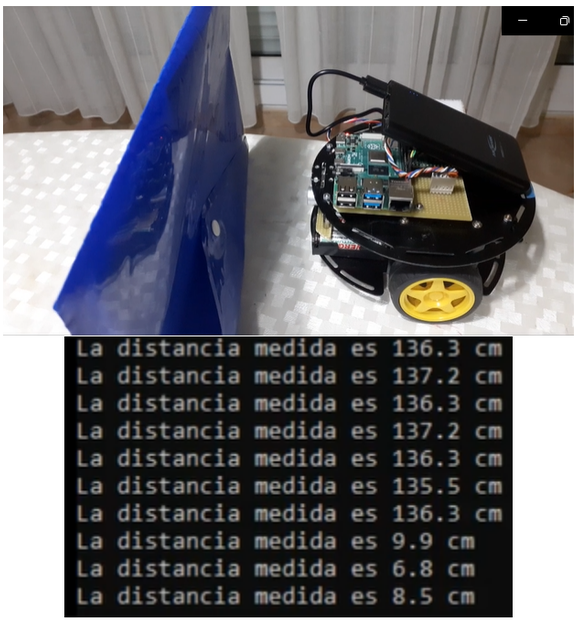

Figure 28: Respuesta del programa ante un obstáculo

Como podemos observar, el programa detecta la presencia de un obstáculo y así nos lo hace saber. Observamos también que la medición presenta una pequeña incertidumbre de aproximadamente ± 1 cm. Esta incertidumbre puede ser producida por distintos objetos del medio que puedan afectar a la percepción de los ultrasonidos por parte del sensor. Podemos considerar pues que el sensor de ultrasonidos queda válidamente implementado. Se ha realizado una documentación en vídeo del ensayo para una mejor percepción del mismo[2].

7.2 Ensayo de la Unidad de Medida Inercial

Para el ensayo de la Unidad de Medida Inercial se ha utilizado un programa que demanda al IMU toda la información que ofrece (aceleración linear, velocidad angular, densidad de flujo magnético y temperatura) y la muestra en pantalla. Se ha hecho uso de las librerías *Adafruit_Blinka_LSM9DS1* y *Adafruit_CircuitPython_LSM9DS1* para poder acceder a los datos del integrado sin tener que acceder directamente a los registros del mismo y además utilizar el bus SPI de manera más sencilla.

A continuación se muestran los resultados del ensayo de funcionamiento, en el que se ejecuta el programa con el robot quieto y después se pone en movimiento para ver si el registro cambia.

```
ubuntu@ubuntu:~$ sudo python3 IMU2.py
Acceleration (m/s^2): (0.292,-0.165,9.857)
Magnetometer (gauss): (0.511,-0.387,0.360)
Gyroscope (rad/sec): (4.993,5.003,-4.976)
Temperature: 27.562C
Acceleration (m/s^2): (0.292,-0.151,9.834)
Magnetometer (gauss): (0.512,-0.386,0.362)
Gyroscope (rad/sec): (0.017,0.269,0.013)
Temperature: 27.562C
Acceleration (m/s^2): (0.389,-0.181,9.875)
Magnetometer (gauss): (0.513,-0.394,0.361)
Gyroscope (rad/sec): (0.016,0.258,0.007)
Temperature: 27.562C
Acceleration (m/s^2): (0.303,-0.132,9.919)
Magnetometer (gauss): (0.517,-0.397,0.366)
Gyroscope (rad/sec): (0.018,0.258,0.011)
Temperature: 27.562C
```

```
Acceleration (m/s^2): (0.377,-0.036,9.859)
Magnetometer (gauss): (0.518,-0.398,0.353)
Gyroscope (rad/sec): (0.021,0.259,0.009)
Temperature: 27.562C
Acceleration (m/s^2): (1.236,1.837,10.868)
Magnetometer (gauss): (0.521,-0.387,0.342)
Gyroscope (rad/sec): (0.026,0.282,-0.012)
Temperature: 27.562C
Acceleration (m/s^2): (0.504,1.542,10.457)
Magnetometer (gauss): (0.522,-0.377,0.350)
Gyroscope (rad/sec): (-0.005,0.308,0.017)
Temperature: 27.625C
Acceleration (m/s^2): (0.342,-0.878,12.001)
Magnetometer (gauss): (0.528,-0.368,0.353)
Gyroscope (rad/sec): (0.032,0.226,0.049)
Temperature: 27.625C
```

Figure 29: Respuesta del programa ante el cambio de inercia. A la izquierda la respuesta en reposo y a la derecha la repuesta al movimiento y la vuelta al reposo

Como podemos observar, el programa detecta un cambio en la inercia del robot y así nos lo hace saber. Se observa que la aceleración lineal en el eje X cambia desde un valor próximo a 0.3 hasta unos 1.2 m/s^2. Un cambio similar puede apreciarse en cuanto a la aceleración similar en el eje Y. Nótese que cuando el robot se encuentra en reposo, los valores de aceleración a pesar de mantenerse aproximadamente constante, no toma valor nulo. Esto pone de manifiesto que para la implementación definitiva de este sensor será necesario calibrar las medidas para obtener resultados precisos. Se ha realizado una documentación en vídeo del ensayo para una mejor percepción del mismo[3].

Al considerarse la función básica de los robots moverse por un espacio bidimensional, se debe comprobar también que la velocidad angular en el eje

[2]URL del vídeo del ensayo del HC-SR04: https://youtu.be/ljIK3orUxcs
[3]URL del vídeo del ensayo del IMU: https://youtu.be/EE4UPbxxlQ0

Z cambie al girar. El resultado cuando el robot gira en el plano XY se muestra a continuación.

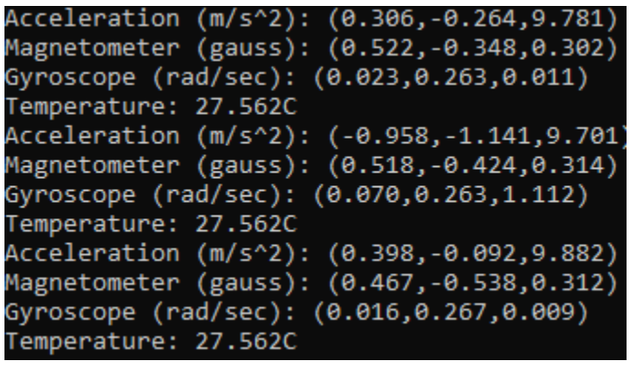

Figure 30: Respuesta del programa ante un giro en el plano XY

Se puede observar que la velocidad angular cambia de valor prácticamente nulo a 1.112 rad/s. Consideramos pues que a falta de calibrar las medidas. La Unidad de Medida Inercial está lista para su utilización.

7.3 Ensayo de la herramienta ROS2 en la Raspberry Pi

Después de la instalación de ROS2 en la Raspberry Pi, el primer paso para probar la herramienta es crear el primer paquete. Este paquete contendrá dos nodos, uno suscriptor y otro publicador. El ensayo consistirá en ejecutar cada nodo en una terminal distinta de la Raspberry Pi y comprobar si son capaces de comunicarse entre ellas. El nodo publicador estará programado para publicar cada segundo en un tópico la cadena de caracteres: "Ensayo de la herramienta ROS2 en la Raspberry Pi". Al mismo tiempo mostrará en pantalla el mensaje junto con el número de veces que el mensaje ha sido publicado.

Por otro lado, el nodo suscriptor al comprobar una nueva publicación en dicho tópico mostrará en pantalla la siguiente cadena de caracteres: "El mensaje recibido es: " y la publicación correspondiente. A continuación se muestra una captura de pantalla del resultado del ensayo.

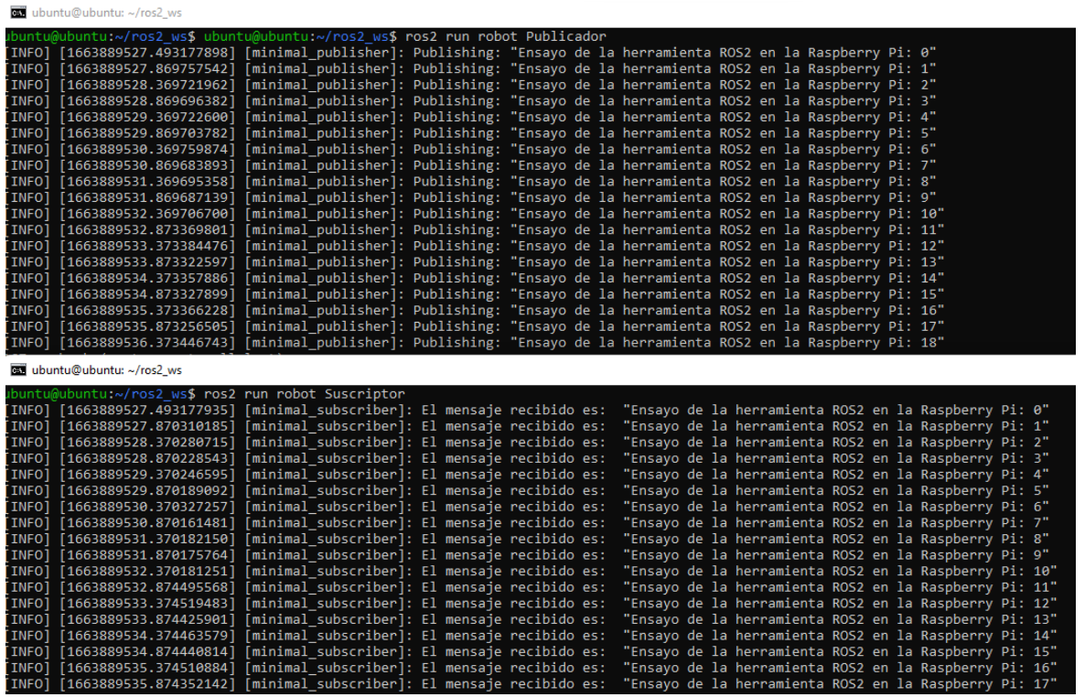

Figure 31: Comunicación entre dos terminales de la Raspberry Pi

Como se puede comprobar, las dos terminales se comunican correctamente. Pasamos pues a considerar que la herramienta se encuentra implementada en la placa Raspberry Pi. Se ha realizado una documentación en vídeo del ensayo para una mejor percepción del mismo[4].

8 Resultados y conclusiones

A continuación se exponen los resultados y, junto a ellos, las conclusiones llegadas al término del trabajo.

El ensamblaje del robot ha quedado compacto y, como después se ha comprobado, el montaje ha sido correcto. Esta tarea tomó bastante tiempo debido a los largos tiempos de entrega de dispositivos electrónicos. Además, se realizaron varios cambios en el diseño debido a la necesidad de implementar nuevos elementos que no aparecían en el diseño original como la batería para alimentar la Raspberry Pi al resultar imposible alimentarla desde la Arduino.

Los tres sensores han quedado correctamente implementados y, a falta de la calibración del IMU, están listos para su utilización en la plataforma. Uno de las tareas que más tiempo ha consumido ha sido averiguar cómo conectar el IMU a la placa Raspberry Pi y, sobretodo, como programar esta última para que leyera los datos. Esto fue debido a que era difícil de encontrar una librería capaz de realizar la tarea de leer los datos del microcontrolador presente, que trabajase con Python y que fuera posible ejecutarla en una Raspberry Pi. Como solución hubo que adaptar una librería diseñada para otra placa de adaptador del LSM9DS1 para poder trabajar. Como conclusión queda que antes de elegir un componente de este tipo, además de atenerse a las características del producto y el precio, sería conveniente también investigar si tiene librerías de fácil implementación.

[4]URL del vídeo del ensayo de ROS2: https://youtu.be/mLLrJZE0X4E

La placa Raspberry Pi 4B ha sido modificada para llevar a cabo las tareas necesarias. Ahora tiene dos puertos SPI activos y es capaz de reconocer un módulo de cámara. Aparte tiene instalada la herramienta ROS2 la cual le provee de un medio de comunicación vía red local tanto con los otros robots como con el ordenador central cuando implementen las mismas adaptaciones.

Durante la realización de este proyecto se han desarrollado programas en Python pensados para llevar a cabo el control proporcional de la posición del robot. Estos programas no se han podido implementar pero quedan expuestos en la siguiente sección *Líneas de acción futuras* junto a la estrategia de implementación.

9 Líneas de acción futuras

A continuación se exponen los posibles siguientes pasos que continuen la línea abierta por este TFG. Algunas de ellas fueron concebidas como parte de este proyecto pero se han descartado debido a la prolongación del TFG que conllevarían. El objetivo de las siguientes propuestas consiste en que el robot pueda llegar a un punto indicado por el ordenador central aplicando un control de posición.

El primer paso a partir de los resultados obtenidos sería crear un nodo publicador en el ordenador central igual que el diseñado para el ensayo llevado a cabo en este proyecto. Sería necesario por tanto cambiar el tipo de mensaje a, por ejemplo, un vector de dos dimensiones que representen las componentes X e Y relativas al robot del punto objetivo.

```
class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__('minimal_subscriber')
        self.subscription = self.create_subscription(
            Int32MultiArray,
            'Objetivo',
            self.listener_callback,
            10)
        self.subscription

    def listener_callback(self, msg):
        Objetivo = msg.data

def main(args=None):
    rclpy.init(args=args)

    minimal_subscriber = MinimalSubscriber()

    rclpy.spin(minimal_subscriber)

    minimal_subscriber.destroy_node()
    rclpy.shutdown()
```

Figure 32: Clase diseñada para el nodo publicador propuesto

El siguiente paso sería diseñar un nodo suscriptor en la Raspberry Pi que lea estas componentes y la función que lance al leer un cambio en el tópico sea el control proporcional de la posición. Este se llevaría a cabo a partir la información aportada, una vez calibrada debidamente, por el IMU.

```python
class ReadObjectiveNode(Node):

  def __init__(self):
    super().__init__('ReadObjective')
    self.subscription = self.create_subscription(
      Int32MultiArray,
      'Objetivo',
      self.ControlP,
      10)
    self.subscription
```

Figure 33: Clase diseñada para el nodo suscriptor propuesto

Para el control proporcional de la posición se inicializarían las variables a 0 ya que el objetivo es relativo al robot y por tanto toma su posición inicial como origen del sistema de referencia.

```python
def ControlP(self, msg):

  destino = msg.data
  xdes = destino[1]
  ydes = destino[2]
  theta = 0
  T=0
  xact=0
  yact=0
```

Figure 34: Inicialización de las variables

Xdes e Ydes hacen referencia a las componentes del punto de destino mientras que Xact e Yact lo hacen a la posición actual del robot. Theta es el ángulo de giro del robot sobre el plano XY relativo a la posición de origen del robot. T es el instante de tiempo de la medición del IMU.

A continuación se utilizan las ecuaciones teóricas para comprobar cual es el error de posición y calcular la acción necesaria para cada rueda a partir de las ecuaciones que definen el robot[1], siguiendo el modelo *Car-like robot*.

```python
while error>tolerancia:

  error = math.sqrt(pow(xdes - xact,2)+pow(ydes - yact,2))

  u_p = kp * error

  error_theta = math.atan2(ydes-yact,xdes-xact)

  diff_ang = round(error_theta-theta)

  if (diff_ang < 0): diff_ang = diff_ang + 2*math.pi

  alpha = kh*round(diff_ang)

  Uveld = u_p * (1 - math.tan(alpha) / 2)* conversion

  Uvels = u_p * (1 + math.tan(alpha) / 2)* conversion

  veld = Uveld * maxduty / U_max
  vels = Uvels * maxduty / U_max
```

Figure 35: Ecuaciones de cálculo de la velocidad de las ruedas

Para volver a calcular el nuevo error de posición, el robot necesita estimar su posición. Mediante el timer de la propia placa Raspberry medimos el tiempo entre mediciones del IMU y aplicamos las ecuaciones de Movimiento Rectilíneo Uniformemente Acelerado.

```
deltaT= time.perf_counter() - T
T = time.perf_counter()

accel_x, accel_y, accel_z = sensor.acceleration


xact = xact + u_p * deltaT * math.sin(theta) + 0.5* accel_x()*math.pow(deltaT,2)
yact = yact + u_p * deltaT * math.cos(theta) + 0.5* accel_y()*math.pow(deltaT,2)

gyro_x, gyro_y, gyro_z = sensor.gyro

theta = gyro_z() * deltaT - theta
```

Figure 36: Estimación de la posición del robot a partir de la información del IMU

Una vez que el robot sea capaz de llegar a un punto, el siguiente paso sería diseñar un programa que, a partir del programa ya creado, fuera capaz de seguir una sucesión de puntos. Una vez que completase el primer objetivo la siguiente orden sería que alcanzase el segundo, y así sucesivamente hasta que no queden puntos. Esto, en otras palabras, es seguir una ruta.

Una vez que el robot ya es capaz de seguir una ruta, ya puede volver a implementarse el algoritmo de distribución de rutas y de la prevención de colisiones que usaba la plataforma para los tres robots móviles.

References

[1] Jose Benigno García Barreto. *Diseño e implementación de un algoritmo que evite colisiones en un sistema multi-robot utilizando el Modified Banker's Algorithm. A 17/07/2021.* 2020. URL: https://zaguan.unizar.es/record/96262/files/TAZ-TFM-2020-1101.pdf.

[2] *Cómo Instalar una cámara en Raspberry Pi: Todo lo que necesitas saber. A 30/08/2021.* URL: https://descubrearduino.com/instalar-una-camara-en-raspberry-pi/.

[3] *Cómo usar un sensor de distancia con tu Raspberry Pi (sensor ultrasónico HC-SR04). A 30/08/2021.* URL: https://descubrearduino.com/hc-sr04-con-tu-raspberry-pi/.

[4] Marco Antonio Espinel Cangui. *Diseño de un sistema de control en tiempo real para el Kernel del Sistema Operativo utilizando Matlab-Simulink. A 23/07/2021.* URL: http://repositorio.espe.edu.ec/xmlui/handle/21000/6016.

[5] Vazquez Leandro Gastón. *Uso de pines GPIO en Python. A 20/07/2021.* URL: http://cdr.ing.unlp.edu.ar/files/presentaciones/013_Pines%20GPIO%20en%20Python.pdf.

[6] Elijah J. Morgan. *HCSR04 Ultrasonic Sensor. A 17/07/2021.* 2014. URL: https://pdf1.alldatasheet.com/datasheet-pdf/view/1132203/ETC2/HC-SR04.html.

[7] Naylampmechatronics. *RASPBERRY PI 4B.A 17/07/2021.* 2018. URL: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf.

[8] Raspberry Pi. *Raspberry Pi Camera Module. A 20/07/2021.* URL: https://docs.rs-online.com/3b9b/0900766b814db308.pdf.

[9] *Por qué y cómo usar la interfaz periférica serial para simplificar las conexiones entre distintos dispositivos. A 30/08/2021.* URL: https://www.digikey.es/es/articles/why-how-to-use-serial-peripheral-interface-simplify-connections-between-multiple-devices.

[10] QNV. *ROS2: hacia la nueva generación de robótica. A 20/07/2021.* URL: https://qnv.com/2019/11/20/ros2-hacia-la-nueva-generacion-de-robotica/#:~:text=ROS%202%20(Robot%20OperatingSystem%202,prototipos%20hasta%20desarrollo%20y%20producci%C3%B3n..

[11] *Robot. A 30/07/2021.* URL: https://es.wikipedia.org/wiki/Robot.

[12] *RPi.GPIO 0.7.1, a module to control Raspberry Pi GPIO channels. A 30/08/2021.* URL: https://pypi.org/project/RPi.GPIO/.

[13] *Secure Shell. A 30/07/2021.* URL: https://es.wikipedia.org/wiki/Secure_Shell.

[14] STMicroelectronics. *iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer. A 20/07/2021.* 2015. URL: https://docs.rs-online.com/3e61/0900766b814d7c2f.pdf.

10 Anexo I: Especificaciones técnicas de los elementos.

10.1 Especifiaciones técnicas Raspberry Pi 4B [7]

- SoC: Broadcom BCM2837B0 (CPU, GPU, DSP y SDRAM)
- CPU: 1.5GHz Quad-core ARM-Cortex A72
- GPU Broadcom VideoCore IV @ 400MHz/300MHz
- Memoria RAM: 1,2 o 4GB LPDDR4
- Conexiones:
  1. 2 x USB 2.0
  2. 2 x USB 3.0
  3. 1 x Salida audio mini jack 3.5 mm
  4. 1 x Salida vídeo/audio HDMI
  5. 1 x Micro USB (Alimentación)
  6. 1 x RJ45 10/100/1000 Ethernet (RJ-45) sobre USB 2.0 (máx.300 Mbps)
  7. 40 x pines macho GPIO (27 E/S, UART, I2C, SPI)
- Interfaz para cámara (CSI)
- Interfaz para Display Raw LCD (DSI)
- Slot para tarjeta Micro SD
- Wifi mejorado: Doble banda de 2.4GHz y 5GHz (IEEE 802.11.b/g/n/ac)
- Antena Wifi rediseñada para mejor rendimiento
- Bluetooth 5.0, BLE
- Alimentación: 5V/2.5 A (3.5 W) via micro-USB
- Dimensiones: 85.6 mm x 56.5 mm
- Peso: 45g
- Año de lanzamiento: 2019

10.2 Especifiaciones técnicas HC-SR04 [6]

- Suministro de energía: +5V DC.
- Corriente de reposo: ¡2mA.
- Corriente de trabajo: 15mA.
- Ángulo efectivo: ¡15º.
- Rango de distancia: 2-400 cm.
- Precisión: 0.3 cm.
- Ángulo de medida: 30º.

- Anchura del pulso del Trigger Input: 10uS.
- Dimensiones: 45mm x 20mm x 15mm.
- Peso: aprox. 10g.

10.3 Especifiaciones técnicas Raspberry Pi Camera V2 [8]

- Número de Canales: 1
- Interfaces de bus admitidas: CSI-2
- Ranuras de expansión: 3280 x 2464 píxeles
- Máxima frecuencia de captura de imagen: 30fps
- Dimensiones: 23.86mm x 25mm x 9mm
- Altura: 9mm
- Longitud: 23.86mm
- Temperatura Máxima de Funcionamiento: +60 °C
- Temperatura de Funcionamiento Mínima: -20 °C
- Ancho: 25mm

10.4 Especifiaciones técnicas iNEMO module LSM9DS1 [14]

- 3 canales de aceleración, 3 canales de ratio angular y 3 canales de campo magnético.
- Escala linear de la aceleración: ±2/±4/±8/±16g.
- Escala magnética de Gauss: ±4/±8/±12/±16.
- Escala del ratio angular: ±245/±500/±2000 dps.
- Salida de datos de 16 bits.
- Interfaces SPI y I2C.
- Suministro de voltaje analógico: de 1.9 V a 3.6 V.
- Sensor de temperatura incluido.
- Funciones de detección de movimiento y posición.
- Ahorro de energía inteligente.
- Distinción entre Click y Doble click