TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

“Robot con navegación autónoma y sistema de vigilancia”

Curso 2017/2018

Alumno/a:

Jonathan López Hernández

Director/es:

Juan Francisco Sanjuan Estrada

“No hay deber más necesario que el de dar las gracias” (Marco Tulio Cicerón)

En primer lugar, quiero agradecer a mi familia, compañeros, amigos y en especial a mi pareja por apoyarme desde que decidí emprender esta nueva etapa en mi vida que ha durado 5 años, ya que sin su apoyo no habría llegado hasta aquí. También quiero dar las gracias a todo el conjunto docente que ha contribuido en mi formación durante estos años, destacando mi gratitud por aquellos que me facilitaron compatibilizar su asignatura con mi horario de trabajo.

Por último, y no por ello menos importante dar las gracias a D. Juan Francisco Sanjuan, por todo su apoyo y ayuda para la realización de este TFG.

Gracias.

Índice

Índice de Tablas

Índice de Figuras

1. Introducción

1.1 Motivación

Hoy en día la domótica se está abriendo paso en nuestros hogares con el fin de hacer más cómodo el día a día e incluso más seguro, porque, aunque el ser humano todavía no se ha arraigado a las facilidades que ofrece la domótica en la vida cotidiana, es un hecho que en el futuro será indispensable en cualquier vivienda, puesto que actualmente las personas se están haciendo más dependientes de la Tecnología. Un ejemplo de ello se puede observar como en muchos de las casas de nueva construcción de hoy en día, están instalando persianas automáticas o instalando un sistema de calefacción controlado a distancia.

Pero, ¿qué es la domótica? Se podría definir como el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, además de aportar seguridad, confort y comunicación entre el usuario y el sistema.

Para poner en práctica esta tecnología se requiere de un sistema que recoja información del entorno que nos rodea, a través de sensores (entradas) y tras procesar los datos recopilados aplicando la lógica programada se emitirá la solución sobre los actuadores (salidas), de tal forma, que permitan llevar a cabo nuestro objetivo centrado en hacer la vida doméstica más fácil, cómoda y segura.

La domótica contribuye a mejorar la calidad de vida del usuario facilitando el ahorro energético, accesibilidad, comunicaciones y seguridad, estando está adaptada a cada vivienda y necesidad del propietario.

El presente Trabajo Fin de Grado (TFG) se centra en el ámbito de la seguridad de personas, animales y bienes, a través de la vigilancia autónoma de un robot móvil, así

como, la detección remota de incidencias y averías. De tal forma, que pueda ser utilizado fácilmente por las personas en su vida cotidiana.

Actualmente existen en el mercado modelos similares al prototipo que se ha implementado en el presente TFG, más centrados en la vigilancia a gran escala, como podrían ser vigilancia de complejos de naves industriales, parques, empresas etc. Un ejemplo de este tipo de robots son los "RAMSEE" como se puede ver en la figura 1, robots autónomos sin supervisión capaces de transmitir en tiempo real información recogida de su entorno como podría ser calor, humo, movimiento y gases. Este modelo puede llegar a medir metro y medio y pesar sobre unos 90 kg.

Fig. 1. Robot RAMSEE [1]

En este TFG se utilizará Raspberry Pi en conjunto con Arduino y a su vez estos estarán apoyados por varios sensores y dispositivos de diferente índole para poder construir un sistema domótico más barato.

Raspberry Pi, es un ordenador de placa única placa simple de bajo coste desarrollada en Reino Unido en febrero de 2012, con el objetivo de estimular la enseñanza en las aulas.

Arduino es una plataforma de hardware libre creada en 2005, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

Se ha escogido Raspberry Pi en conjunto con Arduino (figura 2) por su coste, ya que son unas plataformas relativamente baratas comparadas con otras de la competencia, además ambas son multiplataforma, tienen un entorno de programación simple y son de código abierto tanto en software como en hardware.

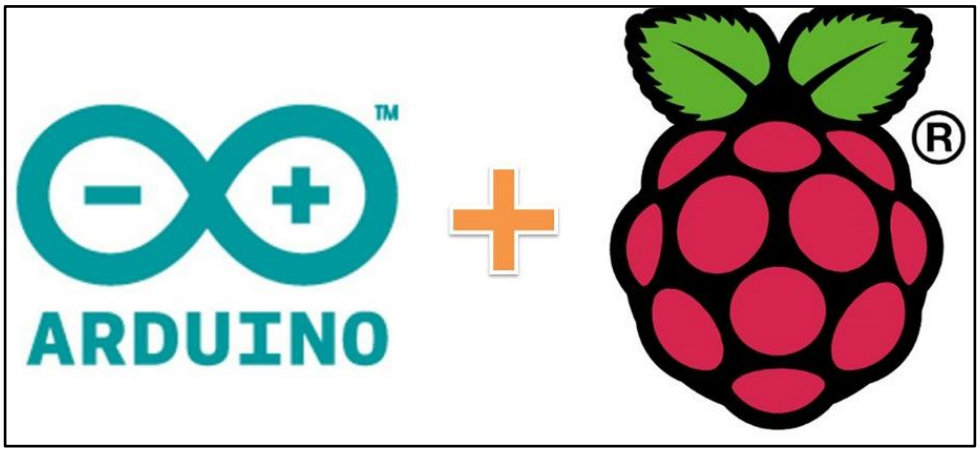

Fig. 2. Arduino más Raspberry Pi [2]

1.2 Objetivos

El principal objetivo que se ha propuesto para la realización de este TFG es la programación de un robot móvil que sea capaz de realizar diferentes funciones, de forma autónoma, como podría ser recorrer o vigilar una casa, habitación, local, etc., e incluso el usuario podría ver en tiempo real que está ocurriendo en el lugar donde se encuentre el robot, todo ello con el fin de hacer más cómodo y seguro nuestro hogar.

Para ello, se dispone de un Arduino robot con capacidad para moverse en todas direcciones, al cual se le integrará una plataforma Raspberry Phi junto con otros dispositivos, tales como sensores, actuadores y de comunicación. De tal forma que, entre otras capacidades, este TFG permita incorporar cierto nivel de inteligencia para el robot sea capaz de crear y memorizar la topología del habitáculo. También, se debe dotar al sistema de la lógica necesaria para que puedan comunicarse las placas con los distintos sensores para una adecuada interacción con el usuario.

A continuación, se describen las distintas funciones que se pretenden integrar en el robot:

- Creación de mapas y navegación autónoma: el robot será capaz de crear un mapa en 2D de forma autónoma evitando los obstáculos.

- Cámara retransmisión en tiempo real: estará provisto de una cámara que retransmitirá en tiempo real las imágenes capturadas.

- Estará dotado de movimiento libre: será capaz de moverse libremente a través del mapa anteriormente creado y evitando obstáculos, además este recorrido que ha realizado se almacenará y se mostrará en una imagen.

1.3 Requisitos

En cuanto a los requisitos del TFG, tanto funcionales como no funcionales, se va a realizar una breve descripción de ellos. Se mostrarán los requisitos funcionales en las tablas 1,2 3, 4 y 5 y los requisitos no funcionales en las tablas 6, 7, 8 y 9.

1.3.1 Requisitos funcionales

RF-1	**Gestión de creación de mapa**
Autor	Jonathan López Hernández
Descripción	El sistema será capaz de crear o borrar un mapa del hogar/local, el usuario elegirá si quiere borrar el mapa para crear uno nuevo.

Tabla 1. Requisito: Gestion de creación de mapa

RF-2	**Gestión de navegación autónoma**
Autor	Jonathan López Hernández
Descripción	El sistema será capaz de navegar autónomamente a través de un mapa creado anteriormente.

Tabla 2. Requisito: Gestión de autonoma

RF-3	Gestión de obstáculos
Autor	Jonathan López Hernández
Descripción	El sistema será capaz de evitar los obstáculos en modo navegación autónoma.

Tabla 3. Requisito: Gestión de obstáculos

RF-4	Gestión de navegación remota
Autor	Jonathan López Hernández
Descripción	El sistema tendrá la posibilidad de ser controlado de manera remota por el usuario.

Tabla 4. Requisito: Gestión de navegación remota

RF-5	Gestión de la transmisión de video
Autor	Jonathan López Hernández
Descripción	El sistema será capaz de enviar video en tiempo real al usuario, siempre a través de una conexión de Internet.

Tabla 5. Requisito: Gestión de transmisión de video

1.3.2 Requisitos no funcionales

RNF-1	**Entorno de explotación**
Autor	Jonathan López Hernández
Descripción	La interfaz de usuario del sistema podrá usarse en cualquier navegador web, sea un ordenador o un smartphone.

Tabla 6. Requisito: Entorno de explotación

RNF-2	**Estabilidad**
Autor	Jonathan López Hernández
Descripción	La aplicación debe funcionar de forma robusta y evitar, en la medida de los posible, fallos durante su ejecución que puedan ocasionar pérdida de datos o producir error en ellos.

Tabla 7. Requisito: Estabilidad

RNF-3	**Disponibilidad**
Autor	Jonathan López Hernández
Descripción	El sistema IoT podrá ser usado en cualquier momento. Esto se conseguirá gracias a una batería externa.

Tabla 8. Requisito: Disponibilidad

RNF-4	Usabilidad
Autor	Jonathan López Hernández
Descripción	El entorno web debe ser intuitiva para que el usuario no tenga dificultad para su utilización.

Tabla 9. Requisito: Usabilidad

1.4 Estado del arte

Actualmente existen diferentes medidas de seguridad a la hora de proteger un hogar, entre ellos se puede resaltar las cámaras y alarmas de seguridad. Una de las empresas más grandes y que abarcan más métodos de vigilancia sería “Securitas Direct”, esta cuenta con cámaras con sensor de movimiento, alarma con potentes sirenas, sensores en puertas y ventanas que detectan cualquier tipo de vibración, aperturas o golpes, sistema contra incendios, incluso inhibidores de frecuencia, todo ello conectado directamente a la central de la empresa, siendo incluso posible dar aviso a la policía. Todo ello debe de ser instalado por habitaciones por lo que aquellas donde no han sido instalados, no quedarán protegidas.

Respecto a nivel de dotar movimiento a un robot, se pueden encontrar pocos modelos que satisfacen todas las necesidades que se pueden presentar en un hogar, un ejemplo de ello es el Robot de Vigilancia “Riley” o RAMSEE.

El robot “Riley”, como se puede ver en la figura 3, posee 1 cámara de 5 megapíxeles con formato de Vídeo Full HD y visión nocturna, permitiendo al usuario conectarse en tiempo real a través del sistema WIFI de la vivienda, también cuenta con sensor de movimiento, de tal forma

Fig. 3. Robot Riley [3]

que, en caso de detectar algún movimiento inusual este mandará un aviso a través de la aplicación. Además, cuenta con 2 ruedas tipo tanque, que le facilita el desplazamiento por diferentes terrenos, “Riley” está programado para volver a su base de carga. La limitación más destacable a la hora de hablar de “Riley” sería la falta de movimiento autónomo, pues este es movido por el propio usuario que deberá de tener instalada la aplicación y mediante “joystick” será quien dirija su rumbo.

En la tabla 10 se muestra una comparativa del robot móvil que se desarrolla en este TFG junto con los robots que existen actualmente en el mercado mencionados anteriormente. Se puede observar que el robot Riley por precio es el más parecido al robot que se ha construido, pero en funciones está bastante limitado. A su vez, también se puede ver que el robot RAMSEE si tiene unas funciones similares a la del robot construido en este TFG, pero el costo es bastante más elevado, ya que se paga por suscripción mensual.

	Robot Arduino	**RAMSEE**	**Riley**
Coste	Robot Arduino: 149 € Raspberry PI 34.90 € Webcam 14.90 € Sensor HCSR04 2.90 € Batería Externa 14.90 €	RAMSEE más de 199€/ mes	Riley 199,90 €
Capacidad para crear mapas de lugar	Sí	Sí	No
Movimiento autónomo	Sí	Sí	No
Retrasmisión en tiempo real	Sí	Sí	Sí
Detección de movimiento	Sí	Sí	Sí
Controlador de temperatura y humedad	Sí	Sí	No
Aplicable en:	Viviendas o locales pequeños	Naves industriales	Viviendas

Tabla 10. Comparativa de robots

1.5 Tecnologías utilizadas en el TFG

Durante el desarrollo del Trabajo Fin de Grado se han hecho uso de varias herramientas y tecnologías para poder llevarlo a cabo. Se va a realizar una breve reseña de cada una de ellas, realizando una descripción más detallada en aquellas que han tenido un mayor peso en el desarrollo del presente TFG:

- JSON: es un acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos.

- Node-Red: se trata de una herramienta de código abierto con enfoque IoT creada por IBM Emerging Technology, la cual permite la creación de aplicaciones mediante protocolos como REST y MQQT además de ofrecer la integración de APIs de terceros como Facebook, Instagram, etc. [4].

- Python: se trata de un lenguaje de programación dinámico capaz de implementarse en diferentes plataformas, además permite crear no solo sitios sino aplicaciones en una gran variedad de sistemas operativos como pueden ser: iOS, Android, Windows o Mac [5].

- IDE Python: Entorno de programación que ha sido preparado como un programa de aplicación, y consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, orientado al lenguaje de programación Python.

- HTML: hace referencia al lenguaje usado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición del contenido de una página web, como texto, imágenes, videos y juegos, entre otros.

- Lenguaje C++: es un lenguaje de programación orientado a objetos, diseñado en 1979 por Bjarne Stroustrup. Con su creación lo que pretendía era extender el lenguaje de programación C y añadirle características para poder manipular objetos, por lo que C++ se puede considerar un lenguaje de alto nivel orientado a objetos.

- IDE Arduino: Entorno de diseño integrado, o entorno de desarrollo, es la plataforma que se utiliza tanto para realizar el desarrollo como para insertar el código a la placa, en este caso sería a la de Arduino [6].

- Servidor XRDP: Protocolo de escritorio remoto, desarrollado por Microsoft, permite comunicar durante la ejecución de una aplicación un terminal (servidor) y otro terminal, de manera inalámbrica, para poder trabajar en este [7].

- OpenCV: es una biblioteca gratuita de visión artificial, originalmente desarrollada por Intel, multiplataforma con funcionamiento en Mac, Windows, Linux, etc. Se centra principalmente hacia el procesamiento de imagen en tiempo real [8].

- No-IP: El servicio de DNS dinámica permite identifica un ordenador en Internet, a través de un nombre sencillo, además de la utilidad de montar un servidor independientemente de tener una IP estática [9].

1.6 Metodología de Trabajo

En este punto se define la metodología de trabajo que se va a seguir durante la realización del TFG. Actualmente existen dos metodologías de trabajo muy extendidas, estas son la metodología tradicional y la nueva metodología ágil (“Agile”).

La metodología tradicional utiliza un ciclo en cascada que organiza el proyecto en varias etapas, hasta que no se termina una etapa de este ciclo no se avanza con la siguiente etapa.

Sin embargo, en la metodología ágil, todas las etapas pueden mezclarse y tiene más flexibilidad para adaptarse a condiciones cambiantes del sector o mercado, aprovechando dichos cambios para proporcionar ventaja competitiva. Es decir, el proyecto se “trocea” en pequeñas partes que tienen que completarse y entregarse en pocas semanas.

Los dos métodos son igual de válidos para la implementación de este TFG, pero se ha decidido usar una metodología tradicional, en el cual se estructura la realización del proyecto en varias etapas y que se irán ejecutando de manera secuencial, cuando finaliza una etapa se continua con la siguiente.

En este TFG se seguirá la siguiente estructura:

- Especificación de Requisitos.
- Análisis.
- Diseño.
- Desarrollo.
- Pruebas.
- Implantación y Mantenimiento.

Se ha descartado la metodología ágil ya que no se tiene la necesidad de ir entregando “partes” del proyecto y este se encuentra bien definido.

1.7 Planificación temporal

En este punto se va a describir la elaboración por fases de la realización del TFG, detallando en cada una de ellas como se va a llevar a cabo, además de mencionar la duración que tendrá el TFG en cada punto de su desarrollo, sumando un total de 300 horas divididas de la siguiente forma:

- **1º FASE: Análisis (50 horas):**
    - ➔ Elección del tema.
    - ➔ Definición del problema.
    - ➔ Formulación del problema.

En primer lugar, se escoge el tema del que tratará el TFG, en este caso, un robot autónomo de vigilancia doméstica. En esta fase, se buscará el problema existente, el cual se suplirá con el robot.

- **2º FASE: Construcción marco teórico (70 horas):**
    - ➔ Búsqueda de fuentes de información.
    - ➔ Diseño del sistema hardware.
    - ➔ Diseño del sistema software.

En esta fase se buscará información sobre cómo se diseña y crea el robot, estudiando de entre las posibilidades existentes la manera más eficaz y sencilla de cumplir con los objetivos propuestos.

- **3º FASE: Codificación y diseño (110 horas):**
    - ➔ Implementación sistema hardware.
    - ➔ Implementación sistema software.
    - ➔ Creación front-end.

A continuación, se implementará todo el hardware y software del proyecto, se unirán todas las piezas necesarias para su correcto funcionamiento, además de darle la programación necesaria para que realice las tareas deseadas.

- **4º FASE: Pruebas (40 horas):**
    - ➔ Pruebas.
    - ➔ Evaluación de resultados.
    - ➔ Depuración.

Parte crucial del TFG, se pondrá a prueba el robot, comprobando que todo aquello que se ha pensado y diseñado pueda ponerse en práctica sin ningún tipo de complicación.

- **5º FASE: Finalización (30 horas):**
  - ➔ Documentación del TFG.
  - ➔ Presentación del TFG.
  - ➔ Defensa del TFG.

Última fase, documentar todos los pasos necesarios para la creación del robot, una vez realizado esto, se pasará a presentar el TFG y a defenderlo.

Para entender mejor la planificación temporal del proyecto, y mostrar los hitos de una manera más detallada, se usará la figura 4.

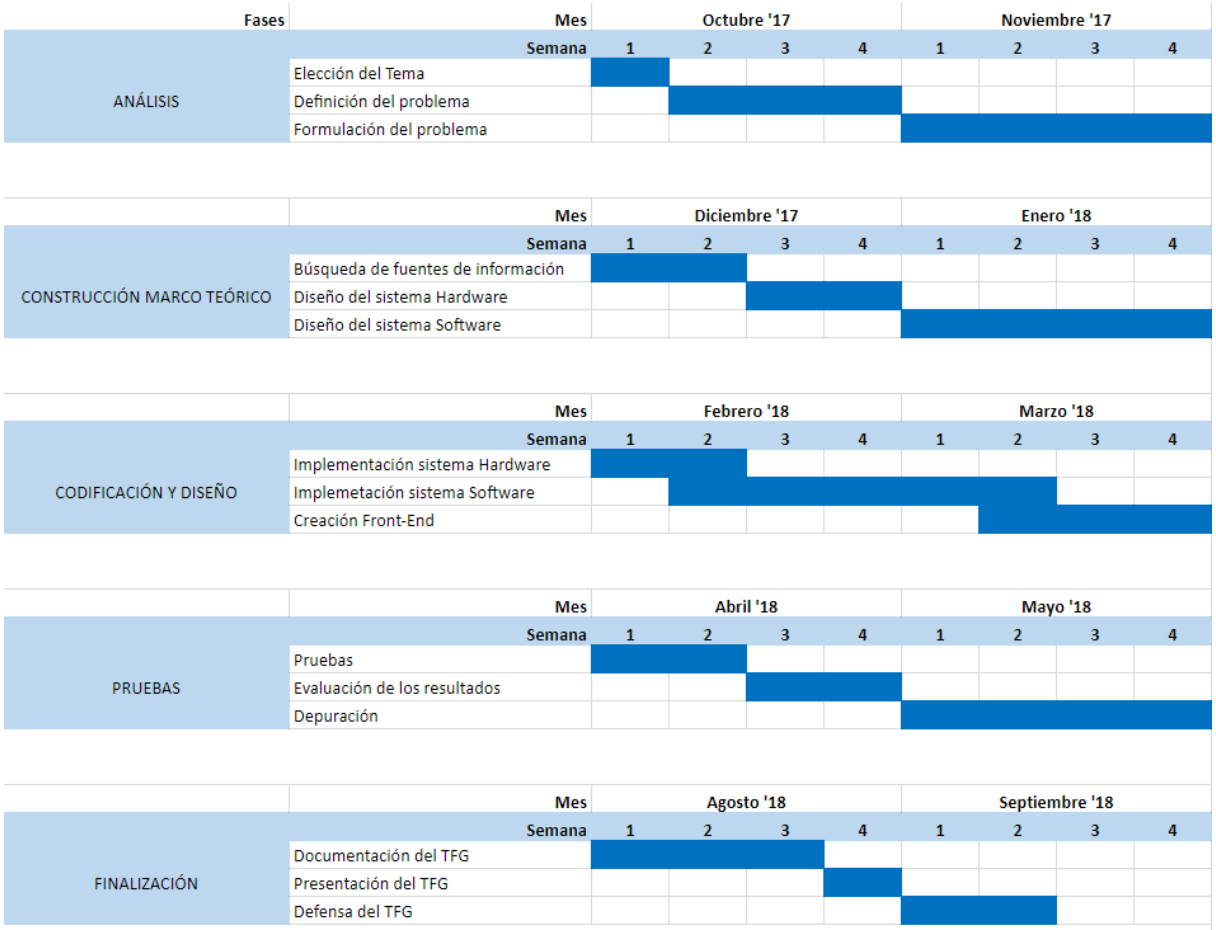

Fig. 4. Planificación temporal

2. Internet of Things (IoT)

2.1 Sistema IoT

2.1.1 Introducción

Internet of Things (IoT), como se le conoce en español “Internet de las cosas”, es un concepto que se refiere a la interconexión digital de objetos cotidianos a internet. Si los objetos de la vida cotidiana tuvieran incorporadas etiquetas de radio, podrían ser identificados y gestionados por otros equipos.

El robot a desarrollar requiere de un componente hardware que permita conectarse a una API para poder visualizar los recorridos, mapas creados y visión de video en tiempo real. Tras realizar un estudio, se tomó la decisión sobre que plataforma hardware usar. Finalmente, tras la búsqueda, se ha decantado por usar Arduino y Raspberry, ya que son plataformas relativamente asequibles en costo y porque tienen una gran comunidad existente.

La elección de estas plataformas fue clara y sencilla, dado que una de las grandes características necesarias era poder conectarse a Internet a través de estas plataformas. Además, Raspberry, al poderse considerar prácticamente un ordenador, permite implementar la aplicación en Python para el desarrollo del código necesario.

2.1.2 Raspberry Pi

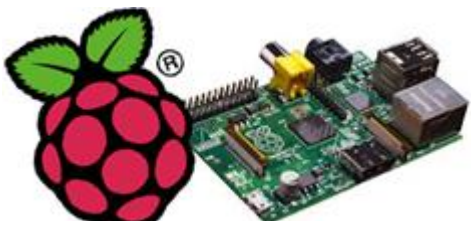

Fig. 5. Raspberry Pi [10]

Raspberry Pi (figura 5) es una placa base del tamaño de una tarjeta de crédito donde se encuentran diversos componentes integrados. No cuenta con disco duro o unidad de estado sólido, ya que lleva integrado un adaptador de una tarjeta

SD para almacenar todos los datos. Tampoco se puede encontrar fuente de alimentación en la placa ya que la poca potencia requerida por el dispositivo permite alimentarse a través de USB.

En el año 2006 nace Raspberry Pi con el objetivo de ayudar a los alumnos de ciencias de la computación de la Universidad de Cambridge. Sus primeros diseños se basaban en el microcontrolador Atmel Atmega644.

En mayo de 2009, Eben Upton, y otros miembros de la universidad fundaron Raspberry Pi, una asociación sin ánimo de lucro cuyo principal objetivo fue la creación de ordenadores portátiles y baratos con los que los alumnos pudiesen trabajar y aprender el funcionamiento básico de un ordenador. Upton fue el encargado de llevar a cabo la arquitectura de software y hardware. El primer prototipo ARM fue montado en una placa muy pequeña, del tamaño de un pendrive, y disponía de un puerto USB y un puerto HDMI.

Fig. 6. Raspberry Pi 1 [10]

El 29 de febrero de 2012 se llevó a cabo el primer lanzamiento del modelo A, en la figura 6 se puede observar la primera Raspberry. Contaba con una memoria de 256MB (inicialmente iba a ser de 128), era un modelo muy básico que contaba con salida de audio y video, además de un puerto USB.

En 2014 salió a la venta Raspberry Pi 2 (B). El motivo de su salida fue complementar al modelo A. Esta nueva placa contaba con dos puertos USB, una entrada de Ethernet y una memoria RAM de 512 MB.

A finales del año 2015 la fundación Raspberry lanzó una nueva placa aún más pequeña llamada Raspberry Pi Zero, con precio de 5€ en las tiendas oficiales de Raspberry. Sus características son similares a las del modelo A pero con una RAM a 512Mb y 1 GHz de velocidad de procesador.

Por último, a principios del 2016, la fundación Raspberry sacó el modelo Raspberry Pi 3, en el que se aumenta la potencia de la placa, y destacaba por disponer de tecnología wifi/bluetooth integrada.

En este TFG se va a usar la Raspberry Pi 3 principalmente por tres factores:

- La potencia del procesador: Una de las funciones del robot móvil que se va a desarrollar, es la capacidad de transmitir video en directo, y se necesita una gran capacidad de cálculo para poder llevar a cabo esta función.

- Tecnología wifi: Este modelo cuenta con la tecnología wifi incorporada y esta característica es imprescindible para la realización del TFG.

- Precio: El precio actualmente de esta Raspberry es de 34.90 €, es un precio muy similar al ofrecido por modelos anteriores, consiguiendo por precio similar características superiores y mayor número de conexiones que modelos anteriores.

2.1.3 Arduino

Arduino es un una plataforma de hardware libre que se basa en un microcontrolador principalmente Atmel AVR, montado en una PCB con los elementos esenciales para su funcionamiento, pensada para proyectos multidisciplinares y un entorno de desarrollo muy sencillo [6].

Además del microcontrolador, Arduino, en casi todas sus versiones, cuenta con entradas/salidas, puerto USB encargado de cargar la programación y alimentarla, y otros elementos, montados en una placa con conectores de fácil conexión y acceso. Dispone también de un entorno de desarrollo (IDE) libre y su propio lenguaje de programación simplificado. Esto le permite acercar el mundo de la electrónica a personas con menos conocimientos sobre microcontroladores o menos recursos económicos. Esto es debido a que la programación de las placas Arduino se realiza

directamente desde su IDE, con un simple cable USB, sin necesidad de utilizar programadores específicos.

Esta ventaja es debida a un bootloader, que viene precargado en el microcontrolador, y permite la carga de programas, sin usar un programador específico.

El IDE de Arduino es multiplataforma, por lo que es compatible con múltiples sistemas operativos, como Windows, Linux o Mac. Esta libertad, amplía su comunidad y facilidad su uso, lo que, sin duda, son algunas de las características en las que se fundamenta el éxito de Arduino.

El primer prototipo de Arduino fue fabricado en el instituto IVRAE (Italia) por Massimo Banzi. En un inicio, se basó en una placa donde se conectaba un microcontrolador simple junto con resistencias de voltaje, en la que únicamente podían conectarse sensores simples como leds y otras resistencias. Todavía no contaba con lenguaje de programación para manipularla.

Unos años más tarde, Hernando Barragán se integró al equipo, Hernando era un estudiante de tesis de la Universidad de Colombia, y al conocer el proyecto, contribuyó al desarrollo de un entorno de programación del procesador llamado “Wiring”, en colaboración con un compañero de Banzi, David Mellis.

Más tarde, se integró el estudiante español David Cuartielles, experto en circuitos y computadoras, quien junto a Banzi mejoró la interfaz de hardware de la placa, agregando microcontroladores necesarios para dar soporte y memoria al lenguaje de programación para así poder manipular la plataforma.

Después, Tom Igoe, un estudiante de tesis estadounidense se interesó en el proyecto y visitó las instalaciones del Instituto. Al regresar a Estados Unidos, recibió un e-mail de Massimo Banzi en el que invitaba a Igoe a participar a mejorar Arduino. Igoe contribuyó a mejorar la placa haciéndola más potente y agregando puertos USB para conectarla a un ordenador. Igoe sugirió a Banzi la distribución de este proyecto a nivel mundial [11].

En este TFG se va a usar un modelo específico de Arduino, concretamente el dispositivo Arduino Robot, en el punto siguiente se detallan las características de este dispositivo.

2.1.3.1 Arduino Robot

Arduino Robot es el primer Arduino oficial sobre ruedas como se puede apreciar en la figura 7. El robot tiene dos procesadores, uno en cada una de sus dos placas.

Fig. 7. Arduino Robot [12]

La placa del motor controla los motores, y la placa de control lee los sensores y decide cómo operar. Cada una de las placas es una placa Arduino completa programable usando el IDE de Arduino. Tanto las placas motor como las placas de control están basadas en ATmega32u4.

El Robot tiene muchos de sus pines asignados a sensores y actuadores incorporados. Ambos procesadores tienen comunicación USB incorporada, eliminando la necesidad de un procesador secundario.

Como siempre con Arduino, todos los elementos de la plataforma (hardware, software y documentación) están disponibles de forma gratuita y de código abierto, por lo que ya sabes “cómo está hecho” y puedes usar su diseño como punto de partida para tus propios robots.

El Arduino Robot es el resultado del esfuerzo colectivo de un equipo internacional que analiza cómo hacer más divertido aprender ciencia [12].

Se ha elegido este Arduino Robot porque cuenta con una pequeña batería, un servo motor y unas ruedas que junto a varias librerías que tiene ya predefinidas le permite:

- Seguir una línea trazada en el suelo.
- Moverse de forma autónoma y aleatoria.

Aprovechando estas características y funciones, en el TFG se ha usado este motor y estas librerías ya instaladas para poder desplazar el robot con cierto grado de inteligencia (por ejemplo, memorizando el recorrido), pero se encontró con la limitación de que no evitaba obstáculos al carecer de sensores para ello, además este Arduino Robot no cuenta con wifi que es indispensable para realizar el TFG. Por estas limitaciones, es por lo que se ha decidido complementar este robot con una Raspberry Pi y unos sensores de ultrasonido.

2.1.4 Node-Red

Node-RED es una herramienta que posibilita interconectar diferentes dispositivos de hardware distintos, API's y servicios en línea usando una interfaz sencilla mediante soluciones software.

Node-RED nos proporciona un editor visual de flujos basado en el navegador que permite de manera sencilla enlazar los flujos de datos usando nodos. Los flujos se pueden desplegar en tiempo de ejecución, mediante peticiones o al realizar un determinado evento.

Se pueden crear todo tipo de funciones en JavaScript que amplían las funcionalidades de los nodos y los flujos. La biblioteca nos permite almacenar las funciones que se desarrollan en los nodos para poder utilizarlos cuando sea necesario.

A principios de 2013 surgió Node-Red como un proyecto paralelo de Nick O'Leary y Dave Conway-Jones del grupo de Servicios de Tecnología Emergentes de IBM.
Lo que comenzó como un concepto de prueba para visualizar y manipular asignaciones entre temas MQTT, rápidamente se convirtió en una herramienta más general que podía extenderse fácilmente en cualquier dirección [13].

2.2 Configuración básica

2.2.1 Configurando Raspberry Pi y entorno de desarrollo

Como primer paso es necesario configurar la Raspberry y el entorno que se usará para desarrollar el código necesario. Desde la página oficial de Raspberry (https://www.raspberrypi.org/downloads/) se puede descargar el sistema operativo Raspbian y siguiendo las indicaciones que muestra la misma página web, mediante una tarjeta MicroSD se puede instalar el sistema [14].

Con el sistema operativo ya instalado se pasa a configurar el software para poder acceder mediante escritorio remoto a la Raspberry, para ello se ha instalado el servidor XRDP con el cual se podrán conectar ordenadores con Windows de una forma rápida y sencilla, una vez instalado el servidor habrá que asignar una IP estática al dispositivo para poder acceder a él.

Para poder implementar todo el software que se utilizará para controlar el robot, se usará Node-Red y Python.

- Se usará Python porque es libre, de código abierto, fácil de aprender y porque Raspbian ya cuenta con la instalación de Python y no se necesitará de ninguna instalación ni configuración extra.
- De Node-Red ya se ha comentado anteriormente su uso y además existe la ventaja de que Raspbian ya viene instalado también con el Node-Red por lo que no hará falta ninguna instalación ni configuración extra.

Por último, se instalará No-IP para poder acceder al dispositivo desde fuera de nuestra red del hogar.
Lo primero que se debe hacer es asignarle una ip estática al dispositivo para que siempre tenga la misma IP y esta no cambie.

Con el siguiente comando se accede al siguiente archivo para configurarlo:

$ sudo nano /etc/network/interfaces

Y se configura como se muestra en la figura 8.

```
  GNU nano 2.7.4          Fichero: /etc/network/interfaces

# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto eth0
iface eth0 inet static
        address 192.168.0.108
    netmask 255.255.255.0
    gateway 192.168.0.1
    dns-nameservers 192.168.0.1

allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.1.107
    netmask 255.255.255.0
    gateway 192.168.1.1
   # dns-nameservers 192.168.1.107
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Fig. 8. Configuración Interfaces

Adicionalmente se instalará la herramienta No-IP para poder acceder al robot desde fuera de la red local.

En el centro de software de la Raspberry se puede descargar No-IP, se accede a su página oficial (https://www.noip.com) y hay que registrarse con un nuevo usuario (figura 9).

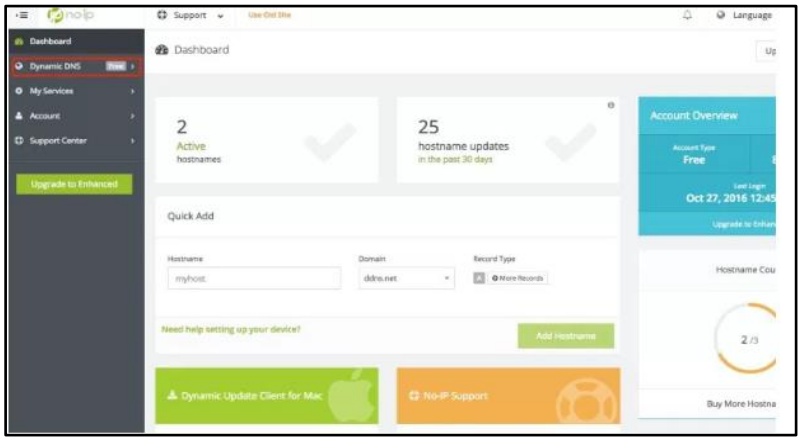

Fig. 9. Dashboard No-IP [9]

Una vez registrados y con No-IP instalado, sólo queda abrir los puertos del router al que conecta, en este caso se abre el puerto 1880 porque es el puerto que utiliza Node-Red.

Una vez que se ha creado y configurado la cuenta, vamos a trabajar en la Raspberry Pi. La idea es que periódicamente, la Raspberry detecte la dirección IP pública a la que está conectado el router, y la envíe al servicio No-IP para actualizar el dominio virtual. Así se garantiza que el dominio virtual creado en No-IP tenga siempre asignada nuestra dirección IP externa de nuestra red. Por último, se configura los servidores virtuales en nuestro Router como podemos ver en la figura 10.

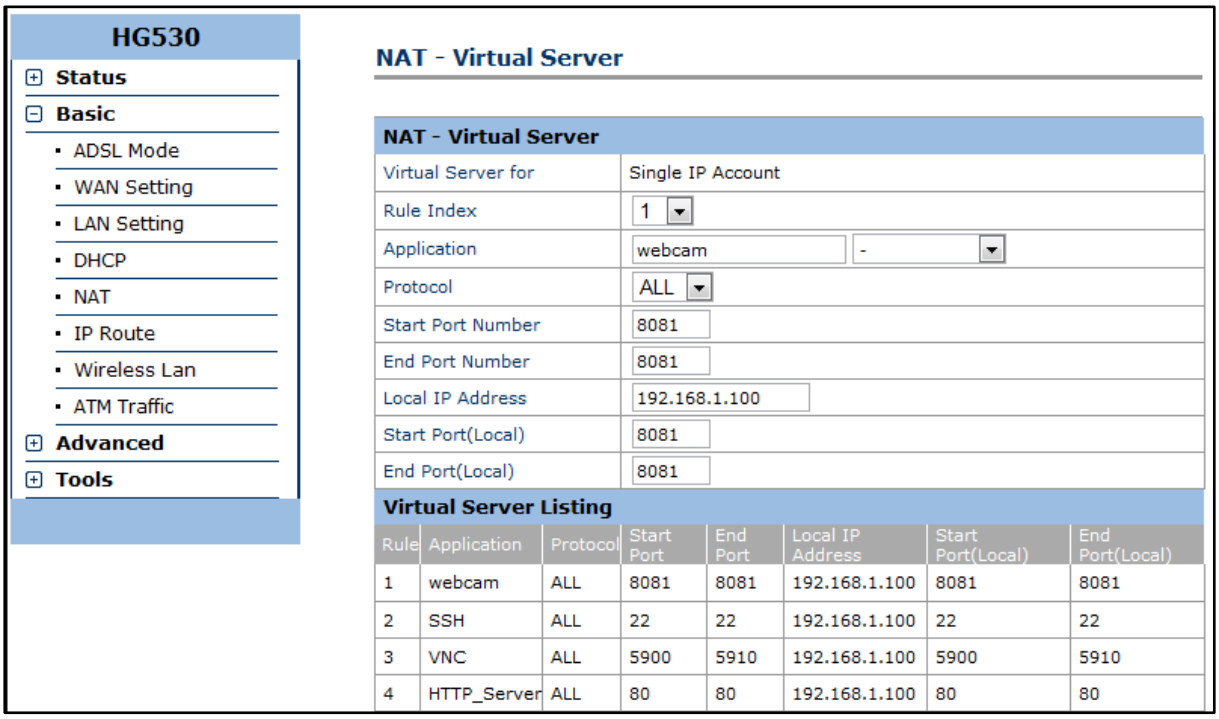

Fig. 10. Configuración Router

2.2.2 Integrando Node-Red y Raspberry

Existen dos versiones posibles de como instalar Node-RED:

- Modo Standalone: Se ejecuta como un proceso NodeJS que es independiente de todos los demás procesos.

- Modo Embebido: Donde forma parte de una aplicación mayor, de forma que es responsabilidad de esta aplicación controlar el ciclo de vida del propio Node-RED.

En el caso de este TFG se usará la versión Standalone ya que no forma parte de una aplicación mayor y es el propio Node-Red el encargado de controlar la ejecución. Además, estará configurado de forma que se ejecute automáticamente cada vez que se encienda la Raspberry Pi.

Por otro lado, se debe comentar que Node-Red trabaja con motores de flujos enfocados al Internet of Things con el que se pueden realizar flujos de servicios en el que se pueden integrar APIs de terceros como Facebook o Twitter, o utilizando protocolos estándares como MQTT o REST.

Además, Node-Red funciona con un editor de flujos, este flujo es una interfaz en HTML, por lo que ofrece la posibilidad de acceder desde cualquier navegador web, ya sea un ordenador o de un smartphone.

Para trabajar con Node-Red, se debe definir un flujo que será el que ofrezca un servicio, en el cual se van añadiendo nodos y conectándolos entre sí, en la figura 11se puede apreciar un flujo a modo de ejemplo.

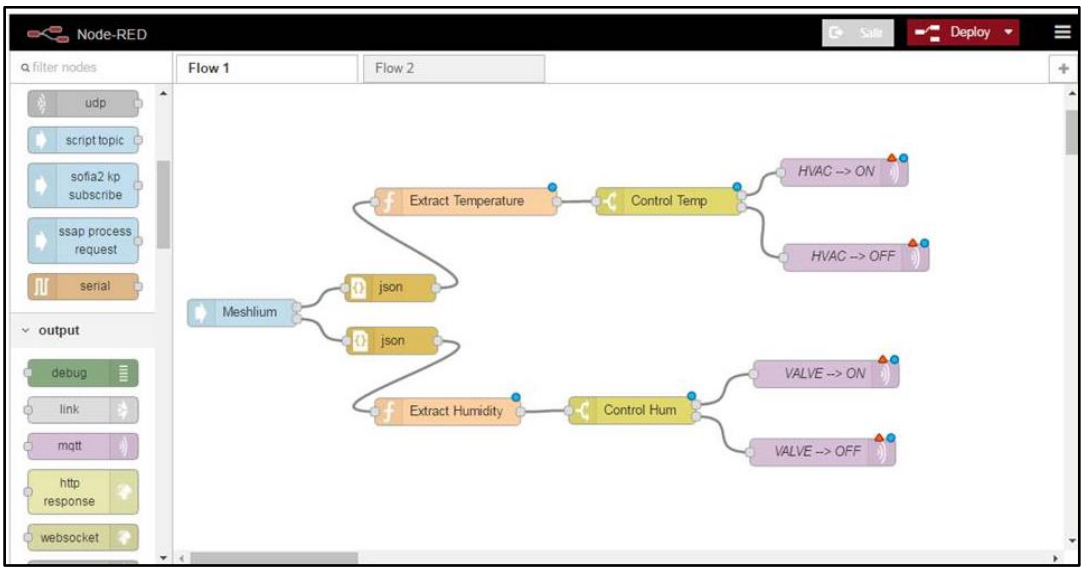

Fig. 11. Flujo Node-Red [4]

Una vez se tiene abierto el editor en el navegador web, este se divide en el editor (donde se añaden todos los nodos y conexiones) y una paleta de nodos (donde aparecen todos los nodos disponibles).

Por último, comentar que el acceso a Node-Red consta con una seguridad mediante usuario y contraseña. En nuestra carpeta de instalación de Node-Red se tiene el archivo Settings.js y sólo quedaría modificarlo con el código mostrado en la figura 12.

```
adminAuth: {
    type: "credentials",
    users: [{
        username: "admin",
        password: "$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6tl8sJogENOMcxWV9DN.",
        permissions: "*"
    }]
}
```

Fig. 12. Acceso securizado

De esta forma, cada vez que se quiera acceder a Node-Red nos pedirá usuario y contraseña previamente configurados, tal y como se muestra en la figura 13.

Fig. 13. Acceso por credenciales

2.2.3 Configurando Arduino y entorno de desarrollo

Una vez realizada la instalación y configuración básica de Raspberry, se continuará con la instalación del IDE de Arduino en un PC.

Para poder instalar el IDE de Arduino, hay que dirigirse a la página oficial y descargar la última versión del software (http://arduino.cc/en/Main/Software), se podrá elegir tanto la versión de Windows installer o la versión portable del IDE. En el caso de este TFG se usará la versión de Arduino 1.8.6.

Una vez instalado, con las versiones más recientes del IDE de Arduino no es necesario instalar los drivers en Windows porque ya vienen integrados en el IDE y con las firmas necesarias [15].

Una vez instalado el software de Arduino, en la figura 14 se puede ver como es el entorno de programación de Arduino.

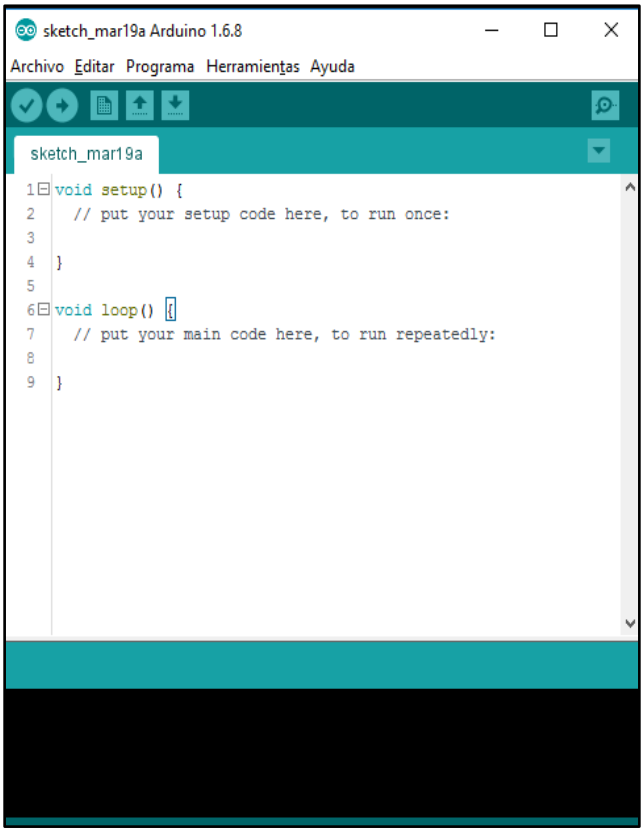

Fig. 14. Arduino IDE

2.2.4 Hardware

Para realizar el montaje, primero se detallarán los distintos componentes que conforman el robot móvil utilizado.
Como ya se comentó anteriormente, el robot diseñado está compuesto por un robot Arduino y una Raspberry Pi pero, además, se ha integrado una webcam y sensores ultrasonido, de tal forma que se procede a detallar.

2.2.4.1 Webcam

La Webcam que se va a usar es el modelo “Logitech QuickCam Chat”, es un modelo bastante antiguo, pero sirve para el uso que requiere el TFG. Las características con las que cuenta relevantes para nuestro TFG son las siguientes:

- Tiene un FOV (field of view) de 46º, que es más que suficiente para poder tener un buen campo de visión.
- Cuenta con una resolución máxima de 640*480 pixeles.
- A la hora de grabar video es capaz de grabar hasta en 30 fps en resolución 320*240 pixeles y 15 fps en 640*480 pixeles.

Además, se debe indicar que para el uso que se le da en el TFG, podría funcionar cualquier otra cámara que sea compatible con Linux.

2.2.4.2 Sensores

Para poder realizar un recorrido por la casa evitando los obstáculos, se van a utilizar dos sensores de ultrasonido HC SR04 como se pueden ver en la figura 15, de tal forma, que estos dispositivos permitan medir distancias. Su funcionamiento se basa en el envío de un pulso de alta frecuencia, no audible por el ser humano. Este pulso rebota en los objetos cercanos y es reflejado hacia el sensor, que dispone de un micrófono adecuado para esa frecuencia.

Midiendo el tiempo entre pulsos y conociendo la velocidad del sonido, podemos estimar la distancia del objeto contra la superficie sobre la que rebotó el impulso de ultrasonidos.

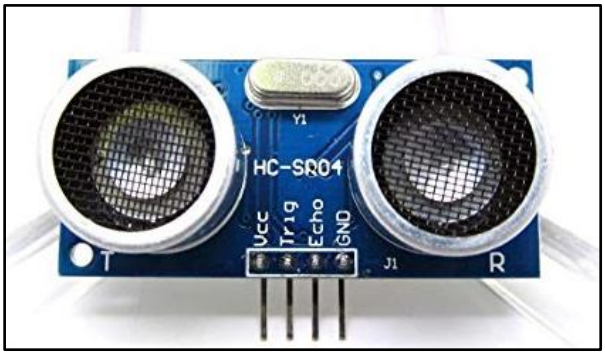

Fig. 15. Sensor HC SR04

2.2.4.3 Hardware Adicional

Además de estos componentes se usarán otros componentes que son más generales como:

- Cables Jumper: serán los cables necesarios para conectar la Raspberry, Arduino y los sensores.

- Protoboard: es una placa (figura 16) que posee unos orificios conectados eléctricamente entre sí siguiendo un patrón horizontal o vertical. Los emplearemos como base para interconectar los componentes.

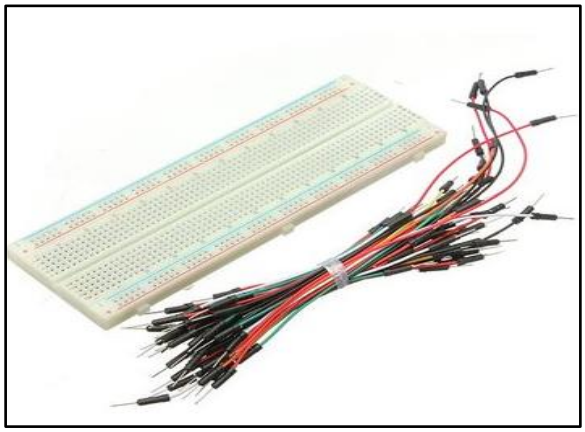

Fig. 16. Protoboard

- Resistencias: Se utilizarán dos resistencias de 1000 ohmios y otras dos resistencias de 2000 ohmios, necesarias para el correcto funcionamiento de los sensores de ultrasonido.

- Batería Externa: Necesaria para la alimentación del robot, esta será de 10 Amperios.

2.2.5 Montaje del dispositivo

2.2.5.1 Conectando Raspberry con sensores de ultrasonido

Una vez detallado el Hardware, se va a proceder a describir el montaje de este, la Raspberry será el centro neurálgico del robot, por lo que tanto el Arduino como los sensores van conectados a la Raspberry [16].

Primero se va a mostrar la conexión de los sensores HC SR04 con la Raspberry a través de la protoboard. Como se puede observar en la imagen, cada sensor debe ir conectado a un pin de 5v, otro de GND y otros 2 GPIOS de la Raspberry. En la figura 17 se puede apreciar de forma más detallada y clara el montaje.

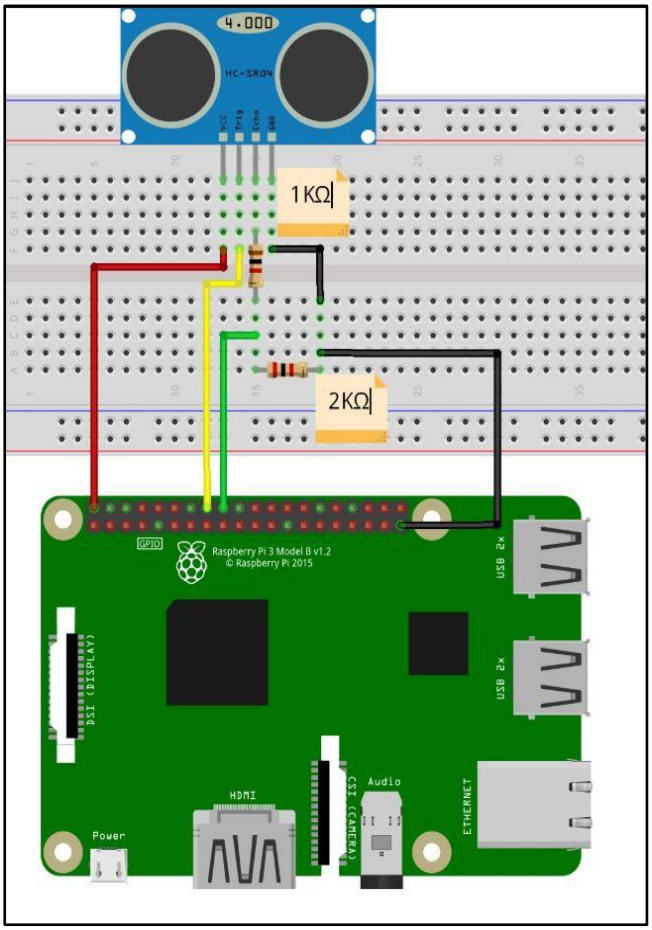

Fig. 17. Raspberry y sensor HC SR04

Con ambos pines se mide el tiempo entre pulsos y sabiendo la velocidad del sonido podremos medir la distancia al objeto a través de la fórmula mostrada en la figura 18.

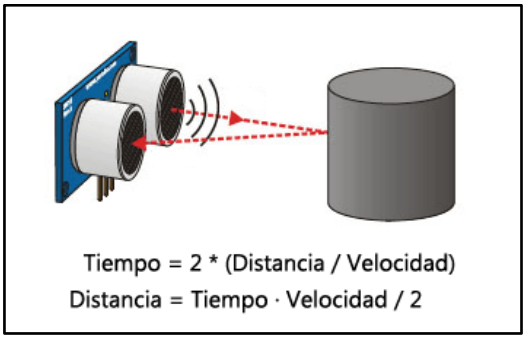

Fig. 18. Fórmula Sensor HC SR04

En la figura 19 se puede observar cómo queda el montaje de los sensores con la protoboard y con la Raspberry.

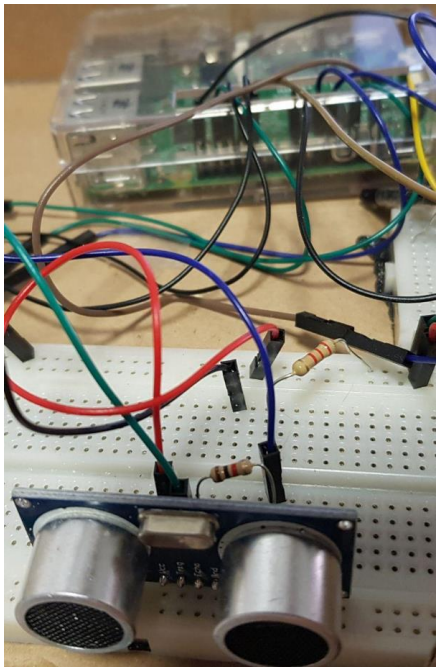

Fig. 19. Montaje Raspberry y sensor HC SR04

2.2.5.1 Conectando Raspberry con Arduino

En este punto se describe cómo se conecta la Raspberry con Arduino, esto se hará mediante cuatro pines GPIO que conectan directamente los dos dispositivos como se puede ver en la figura 20.

Tres GPIOS se usarán para controlar la dirección del Robot Arduino y con el cuarto GPIO se controlará si el robot está en navegación autónoma o en modo creación de mapa [17].

En el siguiente punto se detalla cómo se realizan estos movimientos.

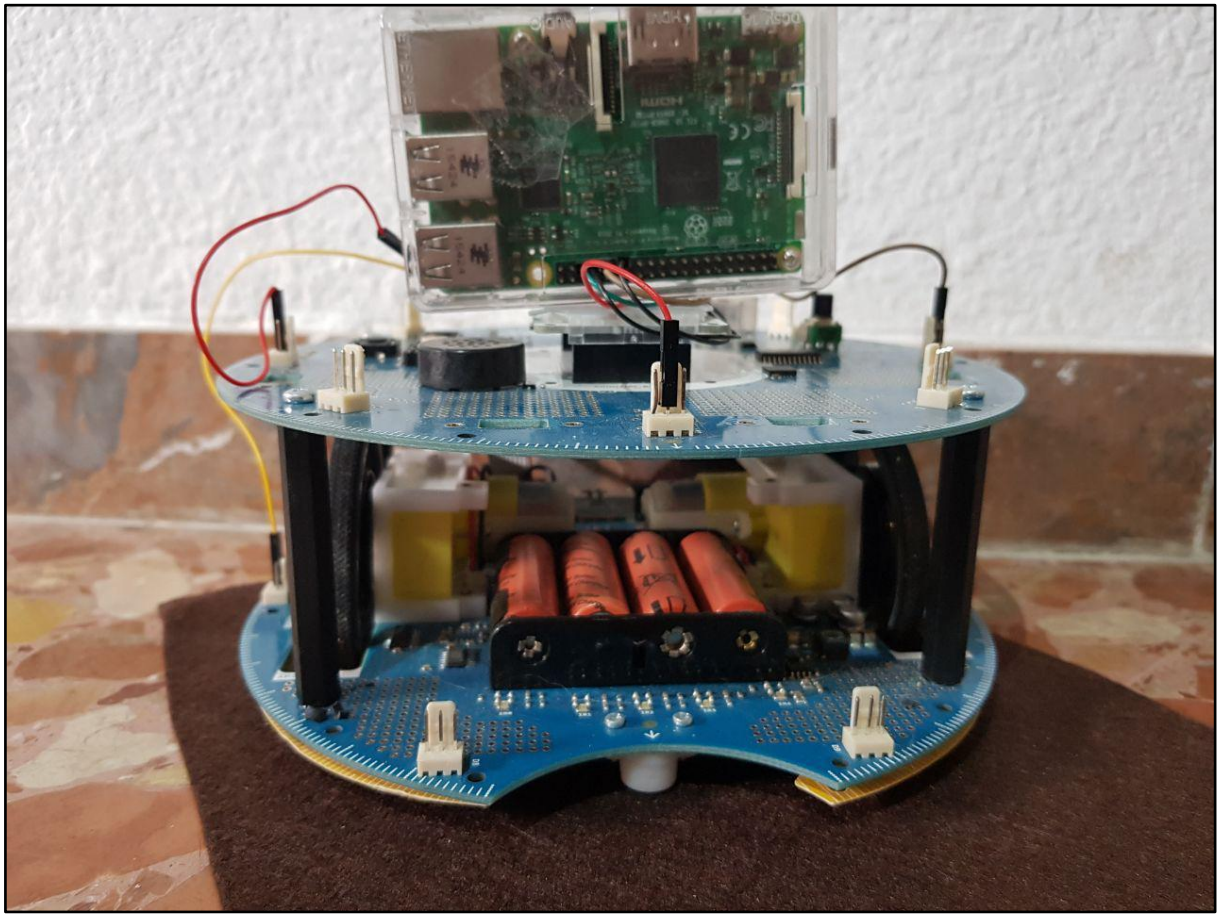

Fig. 20. Raspberry y Arduino Robot

Finalmente, con todo el hardware montado (figura 21), se ha creado una cubierta superior, como se puede ver en la figura 22, que permita proteger todos los componentes para evitar cualquier tipo de problema. Y el montaje final de la parte hardware del robot quedaría así:

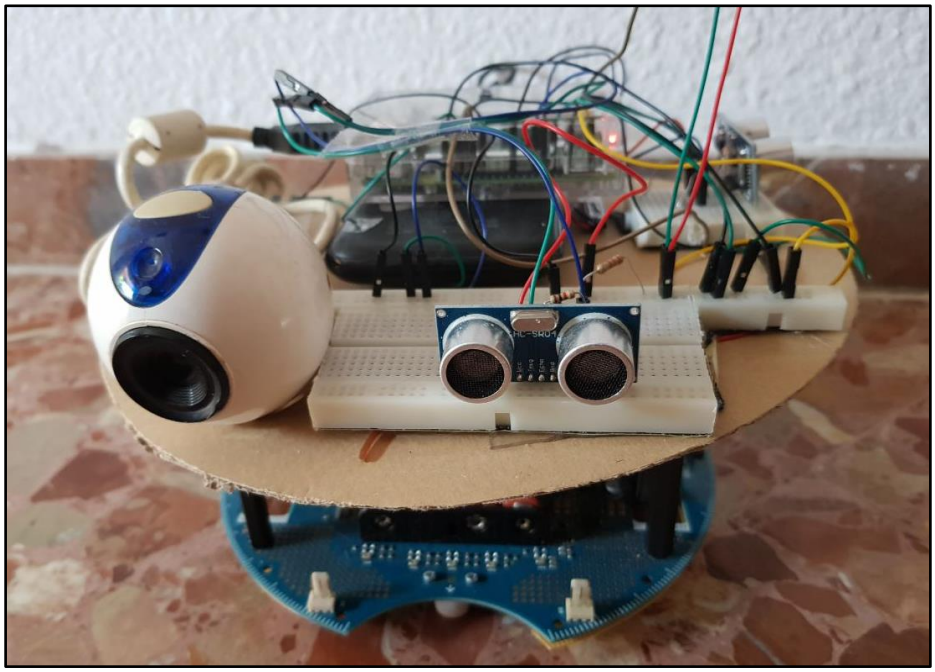

Fig. 21. Montaje del Robot

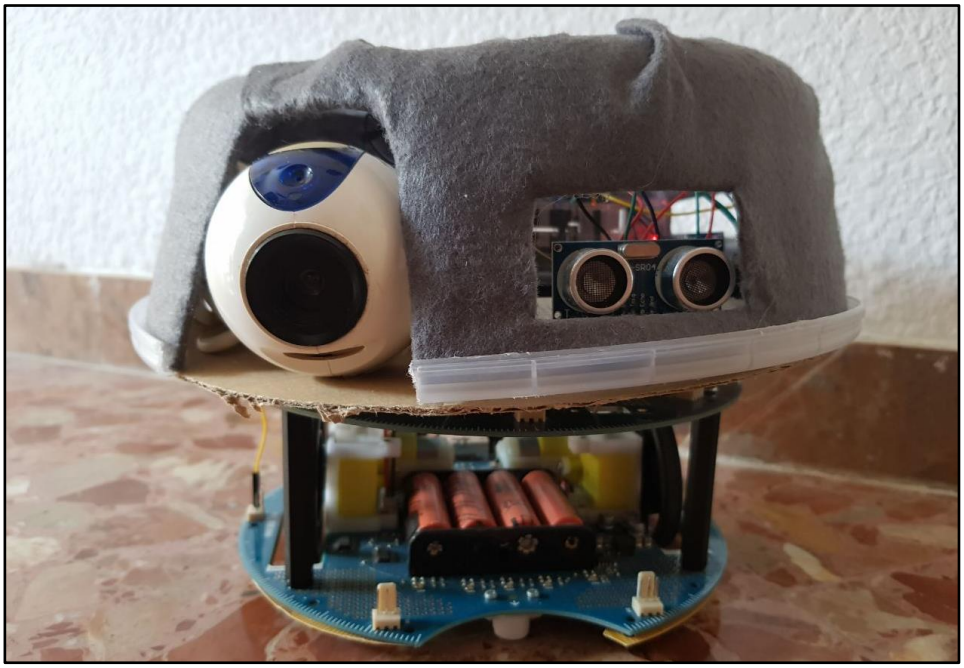

Fig. 22. Montaje robot completo

2.2.6 Control de Raspberry Pi sobre Arduino

La Raspberry será la encargada de indicarle al Arduino Robot cómo se tiene que mover, es decir, le informará permanentemente si tiene que hacer algún movimiento hacia adelante, giro a la izquierda, pararse, etc. A medida que se vaya profundizando se estudiarán la lógica que utiliza la Raspberry para llevar a cabo estos movimientos, pero este apartado se centrará exclusivamente en cómo se mueve el Arduino Robot una vez ha recibido la señal del movimiento por parte de la Raspberry.

Lo primero que debe hacer el Arduino es incluir la librería `ArduinoRobot.h`, esta librería tiene varias funciones para poder hacer todo tipo de movimientos con el Arduino Robot.

Arduino estará constantemente leyendo los pines M0, M2 y M4 situados donde se nos indica en la figura 23, estos pines están conectados directamente a la Raspberry a los pines GPIOS 19, 13, 6. En el momento que se activa el pin correspondiente, llamará a la función correspondiente (izquierda, derecha o hacia adelante) para realizar el movimiento indicado por la Raspberry, en la figura 23 se puede ver el código de la inclusión de la librería, la inicialización del Arduino y lectura de GPIOs.

TFG §

```
//incluir libreria
#include <ArduinoRobot.h>

//declaracion variables
int l, r, t;

void setup() {
  // inicializamos arduino
  Robot.begin();
  Robot.beginSD();
  Serial.begin(9600);
}

void loop() {
  // lectura de pins
  l = Robot.digitalRead(M4);
  r = Robot.digitalRead(M0);
  t = Robot.digitalRead(M2);

  //leemos que pin está en estado HIGH
  if(r!=0 or t!=0 or l!=0){
    else{
        //giro a la izquierda
        if(l == 1 ) {
          leftt();
          delay(400);
          //topp();
        }
        //giro hacia adelante
         if(t == 1 ) {
            topp();
         }
         //giro a la derecha
         if(r == 1 ) {
            rightt();
            delay(400);
            //topp();
         }
    }
```

Fig. 23. Código Arduino para los movimientos

Según se aprecia en el código de la figura 23, una vez que la lectura de un pin está en estado HIGH, se llama a la función correspondiente: girar izquierda, girar derecha o avanzar.

Por otro lado, como se puede observar en la figura 24, el código fuente de las funciones de giro es muy sencillo ya que se utiliza la librería `ArduinoRobot.h`, para hacer cualquier movimiento por lo que solo basta con llamar a la función `motorsWrite()` para que realice los movimientos necesarios con el motor.

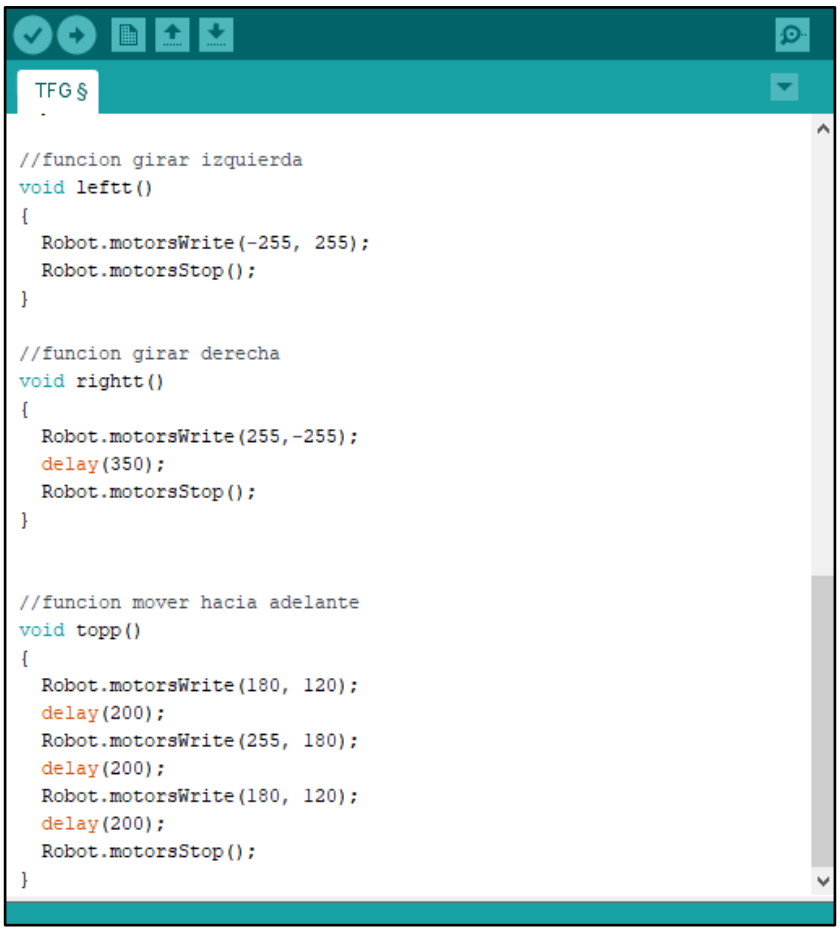

Fig. 24. Funciones para controlar movimiento

3. Robot con navegación autónoma

3.1 Fundamentos

En este punto se va a comentar las funciones principales del robot construido que permita un mayor nivel de inteligencia, de tal forma, que sea capaz de realizar una navegación autónoma:

1. Creación del mapa de forma autónoma: Uno de los objetivos de la creación de este robot es recorrer una habitación, casa, local, etc. de forma autónoma, creando un mapa de esta.
2. Recorrido sobre el mapa creado: Una vez está el mapa creado, existirá una función para poder navegar de forma autónoma y aleatoria a través de este mapa, evitando los obstáculos que se pueda encontrar por el camino.
3. Retransmisión en video: Otra de las funciones principales será poder retransmitir en vivo las imágenes que vaya captando el robot, a modo de cámara de vigilancia.

3.2 Desarrollo software

Todo el desarrollo del software se ha realizado principalmente con Python y Node-Red (JavaScript). En Python, se han utilizado las siguientes librerías externas para poder controlar de forma fácil y sencilla los GPIO de la Raspberry y demás funcionalidades del robot:

- RPi.GPIO: Librería necesaria para controlar los GPIOS de la Raspberry.
- time: Librería necesaria para usar intervalos de tiempo.
- csv: Librería necesaria para poder importar y exportar un archivo csv.
- Image: Librería necesaria para poder importar y exportar una imagen.
- ImageDraw: Librería necesaria para poder modificar una imagen.

Todo el software implementado se complementa con Node-Red, de tal forma, que se ha implementado una interfaz para que el usuario pueda interactuar de forma sencilla con los distintos modos de funcionamiento del robot, así como visualizar la imagen capturada por el robot en tiempo real. Principalmente como módulo externo se ha usado OpenCV para poder visualizar las imágenes de la webcam.

3.2.1 Creación del mapa de forma autónoma

Como idea general, para poder crear un mapa de una vivienda/local se sigue la siguiente lógica:

- El mapa es una matriz de 512x512, de tal forma, que se valida si hay un obstáculo/pared cuando está a una distancia inferior a 30 cm. Cada movimiento hacia adelante del robot es de aproximadamente 15 cm.
- El robot comienza en un punto fijo, en el que tenga una pared (u obstáculo) situada a la izquierda. Este punto debe ser fijo y se tomará como referencia.
- Una vez en movimiento, siempre intenta girar a la izquierda, si no puede girar a la izquierda (significa que tiene un obstáculo) sigue hacia adelante, si tampoco puede seguir hacia adelante (significa que tenemos obstáculo a la izquierda y enfrente) realiza un giro a la derecha.
- Cada vez que realiza un movimiento vuelve a comenzar con la misma lógica descrita anteriormente, primero izquierda, luego enfrente y por último derecha.
- Al finalizar todos los movimientos (cada movimiento se ha ido guardando en una matriz), se crea una imagen/mapa y se guardan los movimientos en un archivo.csv.

Una vez vista la idea general de este programa, se describirá el funcionamiento en detalle.

3.2.1.1 Programa para crear un mapa

Para crear el mapa, el dispositivo se apoyará sobre un programa creado en Python llamado `crearCaminoMapaPins.py`, A continuación, se describen las principales funciones del código:

1. Lo primero que hará será importar las librerías y clases necesarias para el correcto funcionamiento y se crea una matriz de 512x512 inicializada con un "9" cada elemento de la matriz. Se asigna a un elemento de la matriz el número "1" que será nuestro inicio de partida (un punto más o menos central). En la figura 25 se puede ver el código explicado.

crearMapapins.py *

```
import RPi.GPIO as GPIO
import time
import signal
import sys
import os
import csv
import collections
from PIL import Image
from PIL import ImageDraw

matriz = []
for i in range(512):
    matriz.append([0]*512)

for i in range(512):
    for j in range(512):
        matriz[i][j]=9

matriz [a][b] = 1
matriz [a][b+10] = 1
```

Fig. 25. Código librerías Python

2. A continuación, se inicializan todos los pines que usaremos de la Raspberry, tanto pines de entrada (GPIOS 24 y 25) como pines de salida (GPIOS 18, 23, 13, 19 y 6).

3. Se disponen de dos sensores HC SR04 para medir las distancias con algún obstáculo, un sensor en el lado izquierdo del robot y otro en la parte delantera del robot, por lo que el código mostrado en la figura 26 permite medir las distancias de cada sensor en ambas direcciones (izquierda y adelante).

crearMapapins.py *

```
startTime = time.time()
stopTime = time.time()

# save start time
while 0 == GPIO.input(pinEcho):
    startTime = time.time()
# save time of arrival
while 1 == GPIO.input(pinEcho):
    stopTime = time.time()
# time difference between start and arrival
TimeElapsed = stopTime - startTime
# calculate distance
distance = (TimeElapsed * 34300) / 2
```

Fig. 26. Código medir distancia

4. Una vez conociendo las distancias de cada sensor, con el siguiente algoritmo descrito en la figura 27, se controla la dirección a la que se tiene que mover el Arduino Robot y se llama a las funciones necesarias para ir creando el mapa.

```
if distanceL < 30:
    if distance > 30:
        top()
        GPIO.output(pinTop, 1)
        time.sleep(1)
        GPIO.output(pinTop, 0)
    else:
        rigth()
        GPIO.output(pinRight, 1)
        time.sleep(1)
        GPIO.output(pinRight, 0)
else:
    left()
    GPIO.output(pinLeft, 1)
    time.sleep(1)
    GPIO.output(pinLeft, 0)
```

Fig. 27. Codigo para definir dirección movimiento

Si el sensor de la izquierda tiene un obstáculo a más de 30 cm, el robot gira a la izquierda enviando la al pin 23. Sin embargo, si el sensor de la izquierda tiene un obstáculo a menos de 30 cm, se comprueba el sensor delantero, si este tiene la distancia a un obstáculo a más de 30 cm, se moverá hacia adelante enviando la señal a su pin 13. Sin embargo, si la distancia es inferior a 30 cm de un obstáculo, entonces gira a la derecha enviando la señal de HIGH al pin 06.

5. Una vez conocido el movimiento que debe realizar el robot en cada situación, se llama a las funciones para ir creando el mapa, como se puede observar en la figura 28. Como no se dispone de ningún tipo de hardware adicional que informe de la dirección hacia donde está mirando el robot, se ha creado una variable para ir almacenando permanentemente hacia qué dirección apunta el robot.

   En cada función (movimiento enfrente, derecha o izquierda) dependiendo del movimiento anterior, se van restando y/o sumando filas o columnas de la matriz

y se guarda la dirección del movimiento, en la matriz se va almacenando todas las posiciones por las que pasa el robot.

```
def top():
    global b
    global a
    global mensaje
    global matriz
    global pos
    if(pos == "n"):
        b = b-10
        pos = "n"
    elif(pos == "s"):
        b = b-10
        pos = "s"
    elif(pos == "e"):
        a = a+10
        pos = "e"
    elif(pos == "o"):
        a = a-10
        pos = "o"
    matriz [a][b] = 1
```

```
def rigth():
    global b
    global a
    global mensaje
    global matriz
    global pos
    if(pos == "n"):
        a = a+10
        pos = "e"
    elif(pos == "s"):
        a = a-10
        pos = "o"
    elif(pos == "e"):
        b = b+10
        pos = "s"
    elif(pos == "o"):
        b = b-10
        pos = "n"
    matriz [a][b] = 1
```

```
def left():
    global b
    global a
    global mensaje
    global matriz
    global pos
    if(pos == "n"):
        a = a-10
        pos = "o"
    elif(pos == "s"):
        a = a+10
        pos = "e"
    elif(pos == "e"):
        b = b-10
        pos = "n"
    elif(pos == "o"):
        b = b+10
        pos = "s"
    matriz [a][b] = 1
```

Fig. 28. Código verificar dirección movimiento

3.2.1.2 Crear csv e imagen del mapa

Una vez se ha rellenado la matriz con los movimientos almacenados, mediante la función `draw.line()` se creará una imagen del mapa como el que se puede ver en la figura 29 y se almacena en memoria.

Por último, se crea el archivo “matriz.csv” con los movimientos que se han almacenado, en la figura 30 se puede ver el código usado, este archivo será necesario para poder recorrer el mapa, en el siguiente punto se hará uso de él.

```
draw.line(matriz, width=dotSize, fill="red")
img.save('/home/pi/Desktop/mapa.png')
```

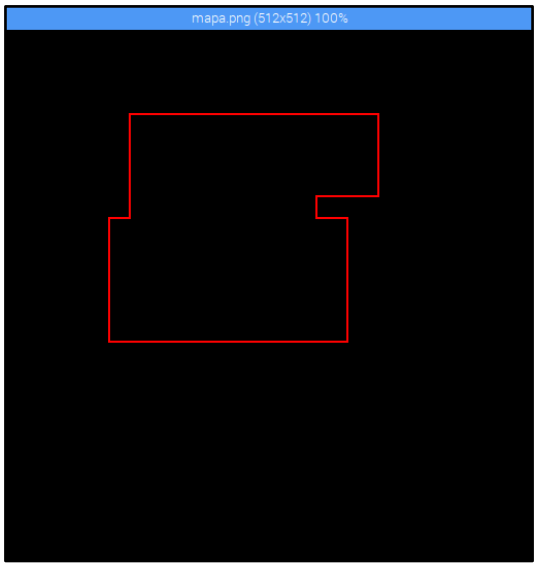

Fig. 29.Código para ver el mapa creado

```
with open('/home/pi/Desktop/matriz.csv', 'w', newline='', encoding='utf-8') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerows(matriz)
```

Fig. 30. Crear archivo CSV

3.2.2 Recorrido sobre el mapa evitando obstáculos

Como idea general el programa diseñado para que el robot realice un recorrido de forma autónoma, realiza lo siguiente:

- Carga el mapa creado anteriormente (archivo csv) y realiza recorridos a través de él.
- Puede evitar los obstáculos que se encuentra en su camino.
- Dibuja en el mapa cargado anteriormente el recorrido realizado.

Una vez vista la idea general de este programa, se describirá el funcionamiento en detalle.

3.2.2.1 Programa para moverse aleatoriamente

Para realizar un recorrido sobre un mapa creado anteriormente, el robot se apoyará sobre un programa creado en Python llamado `caminoAleatorio.py`, A continuación, se van a detallar las principales funciones que realiza el código:

1. Lo primero que hará será importar las librerías y clases necesarias como se puede apreciar en la figura 31.

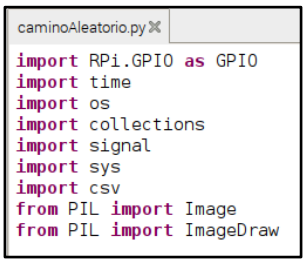

Fig. 31. Importar librerias Python

2. A continuación, se inicializan todos los pines que usaremos de la Raspberry, tanto pines de entrada (GPIOS 24 y 25) como pines de salida (GPIOS 18, 23, 13, 19 y 6).

3. Se carga el archivo "`matriz.csv`" con los datos del mapa creado anteriormente. En la figura 32 se puede apreciar el código necesario.

```
with open('/home/pi/Desktop/matriz.csv') as csvfile:
    reader = csv.reader(csvfile)
    for row in reader:
        matriz.append(row)
```

Fig. 32. Cargar archivo CSV

4. Se disponen de dos sensores HC SR04 para medir las distancias con algún obstáculo, un sensor en el lado izquierdo del robot y otro en la parte delantera del robot. Se miden las distancias de cada sensor con el código descrito en la figura 33.

crearMapapins.py *

```
    startTime = time.time()
    stopTime = time.time()

    # save start time
    while 0 == GPIO.input(pinEcho):
        startTime = time.time()
    # save time of arrival
    while 1 == GPIO.input(pinEcho):
        stopTime = time.time()
    # time difference between start and arrival
    TimeElapsed = stopTime - startTime
    # calculate distance
    distance = (TimeElapsed * 34300) / 2
```

Fig. 33. Medir distancias sensor HC SR04

5. Con el algoritmo mostrado en la figura 34 se escoge el camino a seguir, siempre intenta ir hacia adelante, si no puede seguir hacia adelante, realiza un giro aleatorio a izquierda o derecha.

   La principal diferencia con el algoritmo anterior es que, para poder realizar un movimiento, comprueba la distancia a un obstáculo y comprueba que no está en el borde del mapa, es decir, comprueba que en la posición de la matriz que se encuentra, sea distinto de “1”.

```
if distance > 30 and matriz [a][b] != "1":
    top()
    GPIO.output(pinTop, 1)
    time.sleep(1)
    GPIO.output(pinTop, 0)
else:
    num = int(randint(0, 1) )
    if num < 1 and matriz [a][b] != "1":
        left()
        GPIO.output(pinLeft, 1)
        time.sleep(1)
        GPIO.output(pinLeft, 0)
    else:
        rigth()
        GPIO.output(pinRight, 1)
        time.sleep(1)
        GPIO.output(pinRight, 0)
```

Fig. 34. Definir movimiento a realizar

6. Una vez se conoce el movimiento, se llama a las funciones para ir creando el recorrido sobre el mapa. El algoritmo que utiliza para ir creando el recorrido es el mismo algoritmo descrito en el apartado 3.2.2.1. La única diferencia radica en que va guardando la posición en la que se encuentra en una cadena (figura 35).

```
camino = camino +'('+ str(a) + ', ' + str(b) + '),'
```

Fig. 35. Crear archivo txt con recorrido

7. Para mostrar el recorrido, primero se abre la foto del mapa creado anteriormente, y sobre éste se dibuja el camino como se puede ver en la figura 36, por último, se guarda otra imagen con el nombre "`mapa.png`".

```
img = Image.open("/home/pi/Desktop/mapa.png")
draw = ImageDraw.Draw(img)
draw.line(camino, width=1, fill="blue")
img.save('/home/pi/Desktop/caminoAleatorio.png')
```

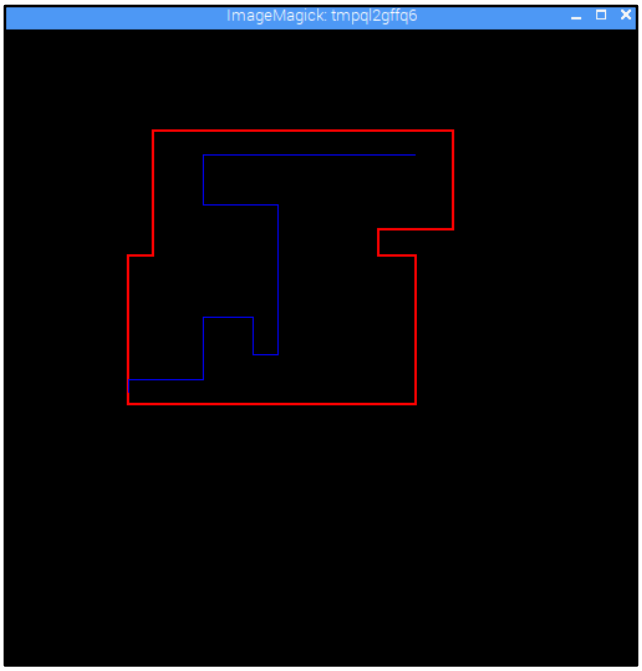

Fig. 36. Dibujar recorrido

8. Finalmente, se crea un archivo de texto `caminoAleatorio.txt` con el recorrido que se ha utilizado por si se le quiere dar alguna utilidad más adelante. En la figura 37 se detalla el código usado.

```
file = open("/home/pi/Desktop/caminoAleatorio.txt", "w")
file.write(str(camino))
file.close()
```

Fig. 37. Crear archivo txt con recorrido

3.2.3 Retransmisión en video

Para poder realizar una transmisión en video, como se comentó anteriormente se usará OpenCV, Node-Red y una webcam, cuya instalación de cada una de estas ya se comentó en puntos anteriores.

En la figura 38 se puede observar el flujo creado en Node-Red para poder retransmitir video.

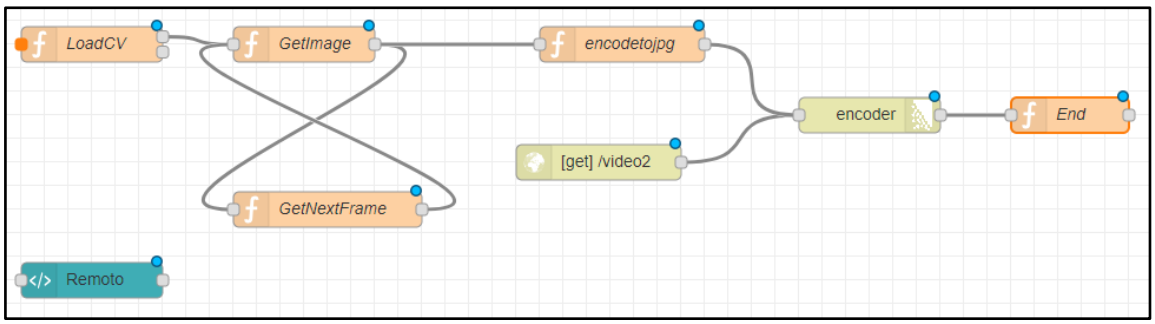

Fig. 38. Flujo Node-Red OpenCV

A modo general, se carga el módulo OpenCV, cada 100 ms se obtiene una imagen de la webcam, de tal forma, que se almacena la imagen en un buffer de imágenes. Una vez se ha cargado la imagen en el buffer, se procede a borrar esta imagen actual, por último, se retransmite el flujo continuo de imágenes sobre HTTP.

Aunque la cámara es capaz de retransmitir hasta 15 fps en resolución 640*480 pixeles, es decir, puede transmitir hasta 15 imágenes por segundo, en este TFG se ha decidido usar 100 milisegundos por cada imagen, que da un resultado de 10 imágenes por segundo. Las razones para escoger esta cifra son las siguientes:

- Si se utiliza más de 10 imágenes por segundo, se necesitaría una potencia de cálculo mayor por lo que se puede llevar a sobrecalentar la Raspberry. Además, a mayor número de fotos por segundo, hace falta una velocidad de subida de internet mayor.

- Si se utiliza menos de 10 imágenes por segundo la retransmisión en video se vería poco fluido, además de que se podrían perder información importante si se hiciera algún movimiento rápido, ya que la cámara no lo captaría.

Se va a explicar la función de cada nodo para poder entender el funcionamiento:

- LoadCV: Nodo encargado de encender y poner en funcionamiento la herramienta OpenCV, en la figura 39 se detalla el código usado.

```
var require = global.get('require');
var cv = null;
try{
    // global
    cv = require.main.require('opencv');
} catch (e) {
    // local
    cv = require('opencv');
}
var cvdesc = Object.keys(cv);
node.send([null, {payload:cvdesc}]);
flow.set('cv', cv);

node.send({payload:1});
node.send({payload:'next'});
```

Fig. 39. Nodo cargar OpenCV

- GetImage: Nodo encargado de obtener la imagen de la webcam, en las figuras 40 y 41 se detallan el código usado.

```
if (msg.payload === 0){
 var vid = flow.get('cvvid');
 if (vid){
 node.warn(util.inspect(vid));
 vid.release();
 flow.set('cvvid', null);
 delete vid;
 }
}


if (msg.payload === 1){
 try{
 flow.set('start', null);
 flow.set('count', null);
 flow.set('last_s', null);

 var cv = flow.get('cv');
 var timings = flow.get('timings') || {};
 timings.startup = {};
 timings.startup.start = Date.now();
 var vid = new cv.VideoCapture(0);
 timings.startup.end = Date.now();
 timings.startup.diff = timings.startup.end - timings.startup.start;
```

Fig. 40. Nodo obtener imagen Webcam

```
node.warn(vid);
flow.set('cvvid', vid);
} catch (e){
node.warn(e);
}
}

if (msg.payload === 'ack'){
var timings = flow.get('timings');
timings.imagecidiff = context.imageci - msg.imageci;
return;
}


if (msg.payload === 'next'){
var vid = flow.get('cvvid');

if (vid){
try{
//vid.grab(function(err, im)
{
//node.warn("grabbed " + util.inspect(err) + " "+util.inspect(im));
```

Fig. 41. Nodo obtener imagen Webcam 2

- GetNexFrame: Nodo encargado de obtener una nueva imagen cada 10 ms, en la figura 42 se puede ver el código usado.

```
setTimeout(function(){
 node.send({payload:'next'});
}, 100);
return;
```

Fig. 42. Nodo obtener nueva imagen

- Encodejpg: Nodo encargado de ir almacenando cada imagen a un buffer de imágenes, en la figura 43 se detalla el código usado.

```
var enable = flow.get('enableavg');

if (1){
    //node.warn(util.inspect(av));
    var d = msg.img.toBuffer(function(err, d){
        var newmsg = {
            payload: d
        };
        node.send(newmsg);
    });
    msg.img.release();
    delete msg.img;
}
```

Fig. 43. Nodo añadir nueva imagen

- Encoder: Nodo encargado de realizar la retransmisión del flujo continuo de imágenes sobre HTTP.

- [GET]/video2: Nodo para crear un "HTTP endpoint" que responda a las solicitudes GET procedentes de una página HTML. Con este nodo creamos una página HTML (llamada video2) donde se aloja el video.

- Remoto: Div HTML donde aloja el "HTML endpoint" creado anteriormente (video2), en la figura 44 se puede ver el código empleado.

```
<div style= "overflow-y: hidden; width=640; height=420">
    <img width="640" height="420" src="http://192.168.1.107:1880/video2?resolution=640x420\" />
</div>
```

Fig. 44. HTML endpoint

3.2.4 Node-Red: Herramienta para controlar el dispositivo

Una vez explicado todos los modos de uso del robot, queda por explicar cómo se agrupan en una sola herramienta de tal forma que un usuario pueda activarlos de una forma clara y sencilla. Con Node-Red se han creado principalmente tres flujos de trabajo para poder controlar el robot.

3.2.4.1 Activar modo creación del mapa forma autónoma

Este flujo consta de un botón llamado "Crear Mapa" que será el encargado de lanzar el programa `crearMapapins.py` (explicado anteriormente), que básicamente, solicita confirmación antes de borrar el mapa anterior. Posteriormente, lanza el programa para crear el mapa y termina con un mensaje de "mapa realizado". En la figura 45 se puede ver los nodos usados.

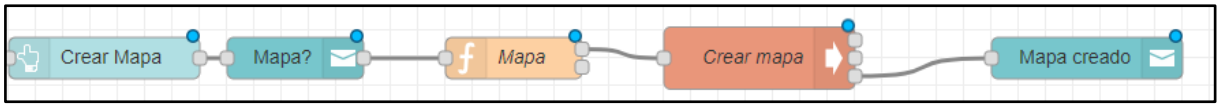

Fig. 45. Flujo crear mapa

Con los nodos mostrados en la figura 46, se podrá visualizar el mapa creado.

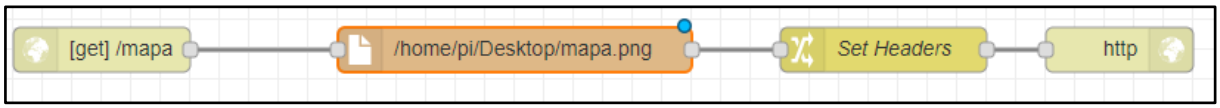

Fig. 46. Flujo visualizar mapa

Finalmente, en la figura 47 se muestra un ejemplo de aspecto que tiene la interfaz de usuario para poder interactuar con el robot.

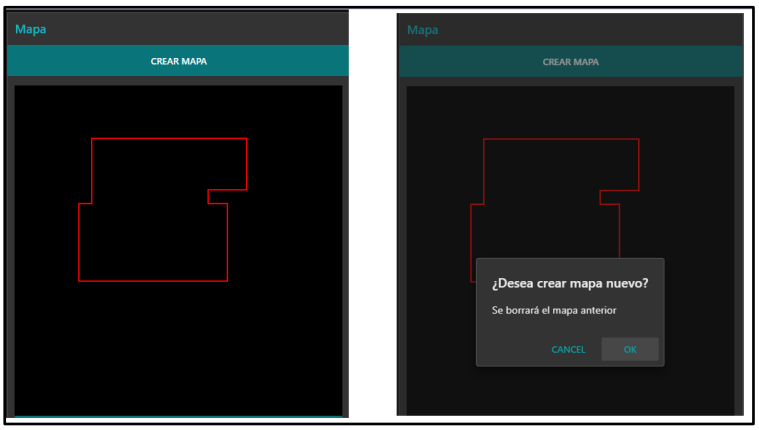

Fig. 47. Node-Red mapa creado

3.2.4.2 Activar modo recorrido sobre el mapa

En este flujo tenemos los botones de “Encender” y “Apagar”, de tal forma que, si se pulsa el botón de “Encender”, el flujo creado lo que hace es lanzar el programa `recorrerCamino.py` y comenzar con la retransmisión en directo y con el botón “Apagar” se interrumpe la ejecución.

Con los siguientes nodos mostrados en la figura 48, se lanza el programa `caminoAleatorio.py` y arranca el módulo OpenCV comentado en el apartado 3.2.3, haciendo que comience con la retransmisión en video comentada.

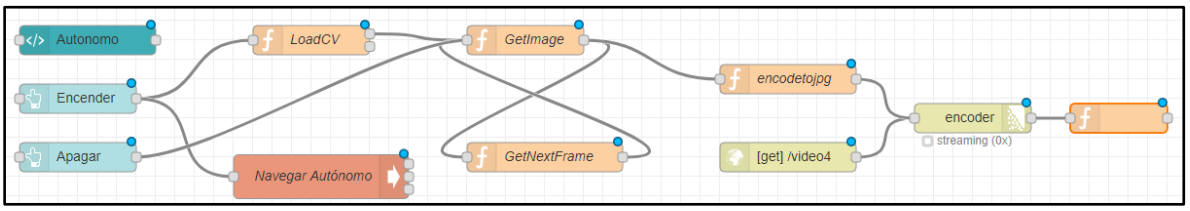

Fig. 48. Flujo OpenCV

Y con los nodos adicionales mostrados en la figura 49, se consigue ver la imagen creada anteriormente, visualizándose la imagen que se va creando al recorrer un camino.

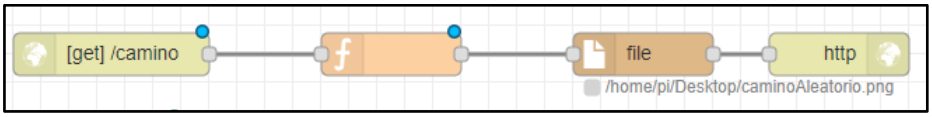

Fig. 49. Flujo ver camino

Finalmente, la figura 50 muestra el aspecto de la interfaz de usuario para poder interactuar con el robot.

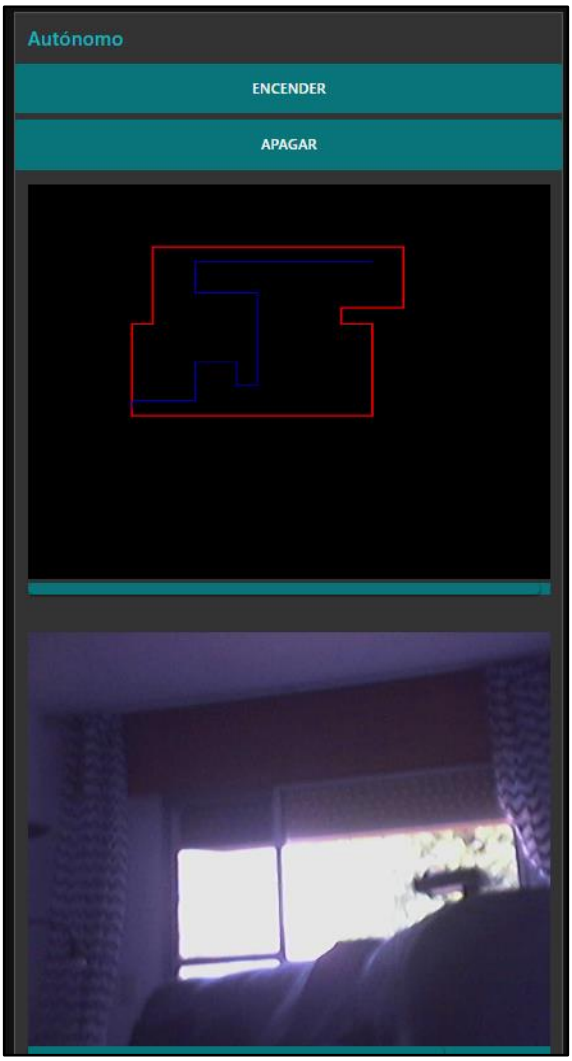

Fig. 50. Node-Red camino creado

3.2.4.3 Node-Red: Menú de navegación

A continuación, se describen los flujos utilizados en Node-Red para crear una API principal que permita al usuario interactuar con las distintas funcionalidades implementadas en el robot.

La figura 51 muestra el flujo, encargado de seleccionar los diferentes flujos, es decir, se diseña un menú para poder acceder a los distintos modos.

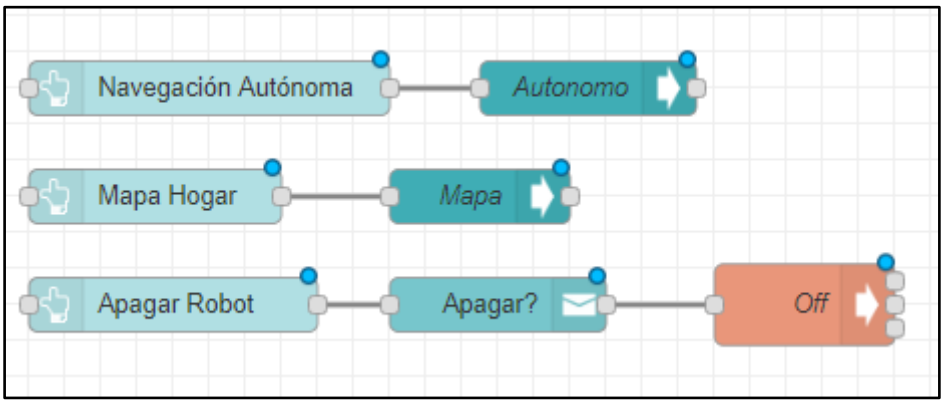

Fig. 51. Flujo menú

Todos estos nodos redirigen a los diferentes flujos implementados, el único que se va a destacar es el nodo “Apagar Robot”, con este nodo lo que conseguirá es que se lance el comando “sudo shutdown” para apagar la Raspberry.

Y la interfaz gráfica para poder hacer uso de este flujo se muestra en la figura 52.

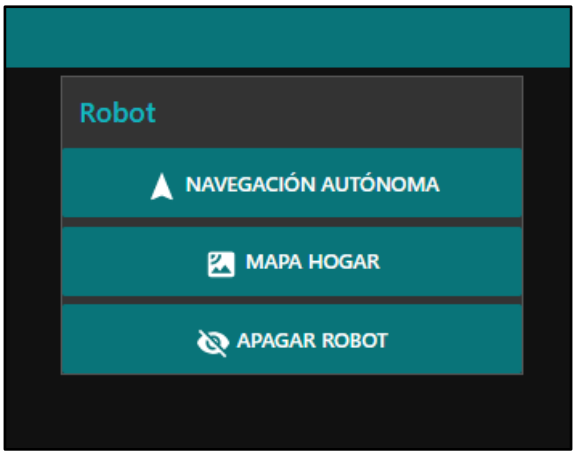

Fig. 52. Node-Red menú navegación

4. Resultados

4.1 Despliegue

El despliegue del robot implementado se hará sobre los mismos dispositivos ya configurados, por un lado, la Raspberry Pi y por otro lado el Robot Arduino.

Para probar su correcto funcionamiento, se probará sobre un ordenador con Windows 10 y sobre un smartphone con Android, en principio debe funcionar en cualquier otro dispositivo que tenga un navegador de Internet, ya que la interfaz de Node-Red se controla a través de un entorno web.

El despliegue se hará un hogar que cuenta con una conexión es 20Mb/2Mb de bajada y de subida respectivamente Al realizar el despliegue dentro de un hogar y tener la opción de conectarse desde fuera de la red local, existe el problema de que las conexiones de la mayoría de los hogares son IP's dinámicas. Este problema se soluciona con No-IP, instalado anteriormente para solucionar este tipo de problemas.

Para desplegarlo, bastaría con encender la Raspberry Pi y el dispositivo Arduino Robot, como anteriormente se ha comentado en el punto 2.2.2, Node-Red está configurado para que se auto ejecute a la hora de encender la Raspberry Pi, una vez arrancados los dos dispositivos, basta con entrar a la dirección web, que en este caso puede ser 192.168.1.107:1880/ui o la dirección www.raspitfg.com si estamos fuera de la red local.

4.2 Resultados obtenidos

Una vez que ya se ha montado todo el hardware y se ha implementado todo el software, se va a proceder a realizar unas pequeñas pruebas para comprobar su correcto funcionamiento y depurar los posibles errores.

Para verificar los resultados se ha creado un entorno de pruebas dentro de un domicilio.
Los pasos seguidos para probarlos han sido:

1. Entrar en la dirección web de Node-Red para poder hacer uso de su interfaz.
2. Una vez dentro la web, se procede a crear un mapa del recorrido de la vivienda, pasados unos minutos se puede apreciar cómo ha creado el mapa correctamente.
3. Una vez se tiene el mapa creado, se prueba la opción de recorrer el mapa de forma aleatoria y autónoma. Finalmente, se puede observar cómo a medida que se va realizando un recorrido, este se dibuja en en mapa creado anteriormente.
4. Adicionalmente se prueba a encender la cámara del robot móvil para asegurarnos que está transmitiendo el video correctamente.

Una vez se ha probado todo, se han visto varios inconvenientes a la hora de crear el mapa, si el hogar o local donde se vaya a desplegar tiene muchos obstáculos, el mapa dibujado sale un poco extraño porque tiene que realizar muchos giros el robot móvil, por lo que sería conveniente dejar el área lo más despejada posible.

Y otro de los inconvenientes que han surgido, es a la hora de transmitir el video en directo, esta retransmisión se realiza a través de la conexión wifi de la vivienda, si el robot móvil se aleja alrededor de 15-20 metros del router, el video empezaba a perder frames ya que bajaría la calidad de la conexión. En un hogar de tamaño medio, este problema no se suele dar porque el router suele estar en un sitio céntrico de la vivienda, pero si quiere usar en algún local habría que tener en cuenta la potencia del router para poder usar esta función del robot.

Por lo demás, creo que se puede dar por satisfecho las funciones que realiza el robot ya que se cumplen los objetivos fijados.

4.3 Publicación de resultados

Todo el código utilizado, así como resultados en forma de imágenes y archivos, se encuentran publicados para compartirse en GitHub como puede verse en la figura 53. Con el siguiente enlace se puede encontrar y acceder al proyecto:

https://github.com/JonathanLopezHernandez/robotMovil

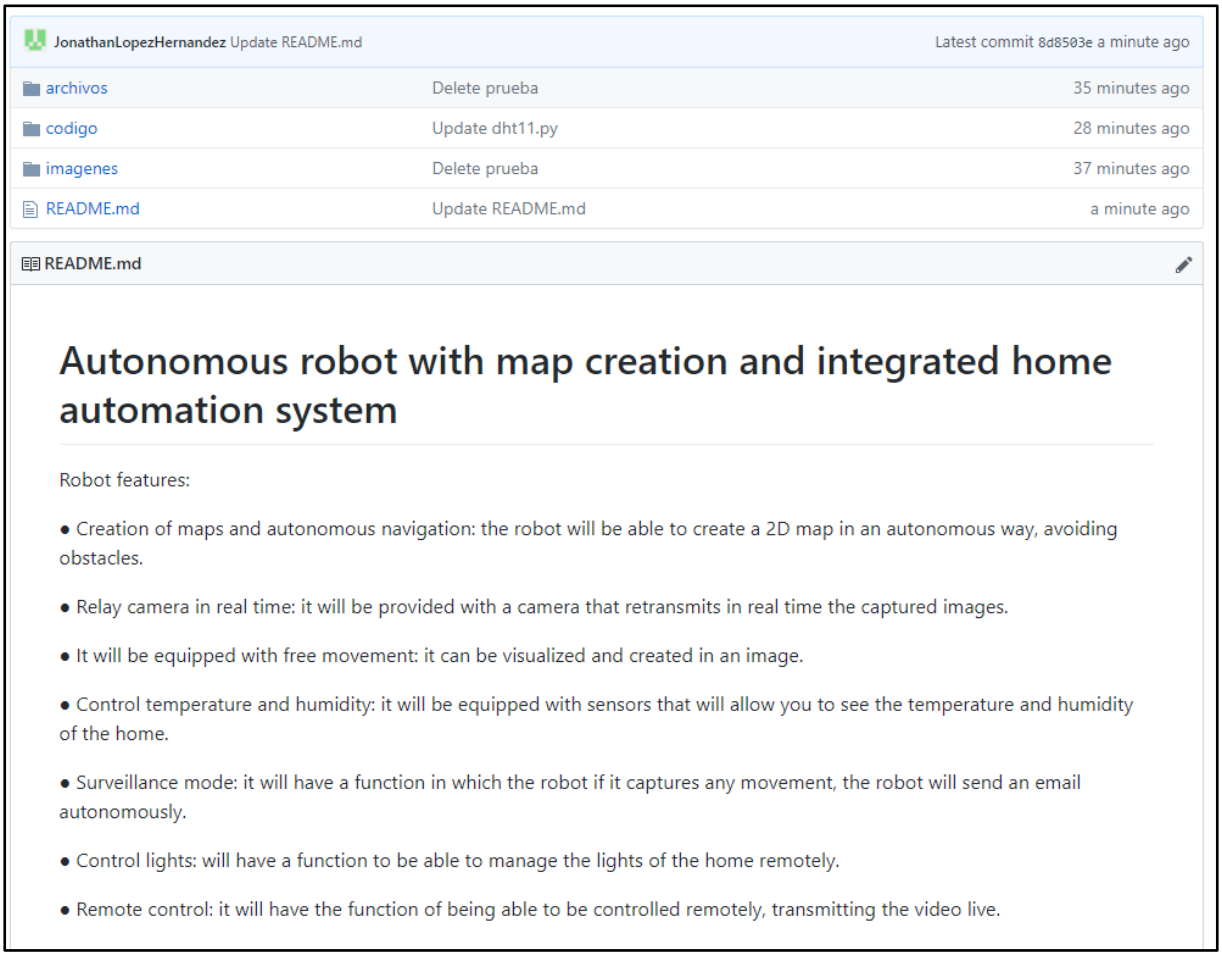

Fig. 53. Proyecto en GitHub

5. Conclusiones

Durante el desarrollo de este proyecto hemos adquirido conocimientos acerca en varios aspectos como pueden ser:

- Se ha usado un lenguaje de programación Python, este lenguaje de programación está en auge en los últimos años, por lo que me ha servido para formarme en un lenguaje nuevo.

- Se ha empleado un conjunto de herramientas que podríamos englobar dentro del Internet of Things (IoT), como puede ser Raspberry Pi, Arduino y Node-Red.

El principal objetivo era el de monitorear la seguridad de una vivienda o local, creando un recorrido y poder visualizar en tiempo real lo que está captando la cámara.
Este objetivo se ha cumplido y se ha conseguido tener una cámara de vigilancia autónoma en el que se moverá por todo el hogar.

Por último, hay que destacar que al realizar el despliegue en una vivienda con una conexión de internet de 20Mb/2Mb, si el usuario final se encuentra fuera de la red local, a la hora de retransmitir el video en directo en algunas ocasiones la señal se podría cortar unos segundos. Esto es debido a que la subida de esta conexión de internet no es muy alta (2 MB), como conclusión se obtiene que, si el usuario final está fuera de la red local, la conexión de internet de este hogar deberá tener al menos 3 MB o 4 MB de subida.

5.1 Trabajo Futuro

Una vez terminado el trabajo del Trabajo Fin de Grado han surgido una serie de ideas que serían interesantes poder implantarlas en el robot móvil creado es este TFG, las siguientes ideas que han surgido son:

- Implantar un módulo GPS y un módulo de conexión a internet para poder llevar a cabo todas estas funciones en exteriores.

- Crear un centro de demótica para poder controlar varios aspectos de una vivienda como luces, persianas, temperatura, etc. Pero para crear esto nos harían falta varios módulos de hardware adicionales al robot y en el hogar.

- Añadir reconocimiento facial al robot, para ello se requiere de una cámara de alta resolución, el uso de una Base de Datos y utilizar alguna herramienta que permita llevarlo a cabo como Bluemix. Esta herramienta ha sido creada por IBM y cuenta con varias funciones prediseñadas para poder realizar esta función. Como inconveniente hay que añadir que esta herramienta es de subscripción mensual.

Bibliografía

[1] William P. “Robot para vigilancia: Ramsee, un héroe de metro y medio”. 2016. Web: http://www.pcworldenespanol.com/2016/12/08/robot-vigilancia-ramsee-heroe-metro-medio/

[2] Albright. “Arduino vs Raspberry Pi: A Detailed Comparison”. 2016. Web: https://beebom.com/arduino-vs-raspberry-pi/

[3] Empresa Ipatrol. Robot Riley. 2018. Web: https://www.ipatrol.net/product/riley/

[4] Página oficial Node-Red. “Flow-based programming for the Internet of Things”. 2018. Web: https://nodered.org

[5] Página oficial Python. “Fundamentos de Python”. 2018. Web: https://www.learnpython.org/es/

[6] Página oficial de arduino. 2018. https://www.arduino.cc

[7] Página oficial. “Servidor XRDP”. 2018. Web: http://www.xrdp.org

[8] Página oficial OpenCV. “Open Source Computer Vision Library”. 2018. Web: https://opencv.org/

[9] Página oficial No-IP. No-IP: free dynamic DNS. 2018. Web: https://www.noip.com

[10] Página oficial Raspberry Pi. “Teach, Learn, and Make with Raspberry Pi”. Web: https://www.raspberrypi.org

[11] Arduino, Tecnología para todos. “Historia”. 2018. Web: https://arduinodhtics.weebly.com/historia.html

[12] Página oficial Arduino. “Arduino Robot”. 2018. Web: https://store.arduino.cc/arduino-robot

[13] Página oficial Node RED. Node RED Programming Guide, Programming the IoT. Web: http://noderedguide.com

[14] Mocq, François. Raspberry Pi 2: Utilice todo el potencial de su nanoordenador, ENI, Barcelona, 2016.

[15] Ruiz-Gutiérrez J.M. “Arduino: Manual de Programación”. 2007. Web: https://arduinobot.pbworks.com/f/Manual+Programacion+Arduino.pdf

[16] Francesc. “Sensor ultrasónico HC-SR04 para Raspberry Pi”. 2018. Web: http://fpaez.com/sensor-ultrasonico-hc-sr04-para-raspberry-pi/

[17] Reyes-Cortés F, Jaime Cid Monjaraz. “Arduino: aplicaciones en robótica, mecatrónica e ingenierías”.

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

El presente Trabajo Fin de Grado (TFG) se centra en el ámbito de la seguridad de personas, animales y bienes, a través de la vigilancia autónoma de un robot móvil, así como, la detección remota de incidencias y averías. De tal forma, que pueda ser utilizado por las personas en su vida cotidiana.

El principal objetivo que se ha propuesto para la realización de este TFG es la programación de un robot móvil que sea capaz de realizar diferentes funciones, de forma autónoma, como podría ser recorrer o vigilar una casa, habitación, local, etc., e incluso ver a tiempo real que está ocurriendo en el lugar donde se encuentre el robot. Todo ello con el fin de hacer más cómodo y seguro nuestro hogar.

The present TFG focuses on the field of security of people, animals and goods, through the autonomous monitoring of a mobile robot, as well as the remote detection of incidents and breakdowns. In such a way, that it can be used by people in their daily lives.

The main objective of this TFG is the programming of a mobile robot that is capable of performing different functions, such as touring or monitoring a house, room, etc., and even see in real time what is happening in the place. All this in order to make our home more comfortable and safe.