Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Creación de una interfaz de movilidad del iRobot Roomba 600 mediante una Raspberry Pi

Autor: Pablo José Domínguez Camacho
Tutor: Sergio Luis Toral Marin
Co-tutor: Daniel Gutiérrez Reina

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Creación de una interfaz de movilidad del iRobot Roomba 600 mediante una Raspberry Pi

Autor:
Pablo José Domínguez Camacho

Tutor:
Sergio Luis Toral Marin
Profesor Catedrático

Co-tutor:
Daniel Gutiérrez Reina
Profesor Contratado

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado: Creación de una interfaz de movilidad del iRobot Roomba 600 mediante una Raspberry Pi

Autor: Pablo José Domínguez Camacho

Tutor: Sergio Luis Toral Marin

Co-tutor: Daniel Gutiérrez Reina

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis amigos

Agradecimientos

Estas palabras nacen de muchos horas de pensamiento y no ha sido una tarea fácil ya que con ellas doy por finalizada una etapa bonita y especial en la que he aprendido mucho y he crecido mucho como persona pero no hubiese sido posible sin la ayuda de todas aquellas que me han ido acompañando a lo largo de estos años.

Me gustaría agradecer en primer lugar a todos y cada uno de los profesores que he tenido durante toda mi etapa educativa desde educación infantil hasta estos días porque gracias a ellos ha sido posible que este escribiendo estas líneas.

A mis padres por confiar tanto en mi, por tener la capacidad de aguantarme en momentos que ni yo mismo me aguantaba, por darme tanto cariño en los momentos que mas lo necesitaba y por ser siempre los primeros en estar ahí para darme un empujon para seguir creyendo. Vuestro carácter luchador me ha llevado a estar donde estoy.

A mis hermanas por transmitirme la alegría constante que me impulsaba a seguir creyendo y a seguir creciendo, sois parte de mis logros porque sin ustedes nada hubiese sido posible.

A mi familia, mis abuelas, mis tios y "mi familia puntaumbrieña", sois gran parte de esto, todos y cada uno de ustedes me habéis ayudado tanto que quizás ni os lo imaginais, pero con estas líneas quiero dejar constancia de ello, gracias de verdad.

A mis amigos, la familia que se elige, porque siempre habéis sacado un momento para mi y habéis estado al pie del cañon, me habéis regalado tantos momentos de felicidad que no los podría ni contar, todos y cada uno de ustedes sois especiales. Gracias Carlos, Javi, David Orta, Ale, David Rodriguez, Burgui, Julian, Coca, Manuel, Juan, Isaac, Leire, Alejandra y Pozu, ustedes también sois parte de esto.

A mis compañeros, Ignacio, Boa, Ana y Samuel, por ser apoyos constantes, hacer que las clases, practicas y exámenes se enfrentaran con mayor tranquilidad y que estos 4 años hayan sido una etapa mas bonita y especial aún.

A mis compañeros de piso, Patricio, Daniel, Paco, Alfonso, Fran y Jose Mari, por enseñarme a vivir alejado de mi familia y hacerme sentir como en casa, por tantas charlas especiales, por todas las noches de futbol con sus piques correspondiente... y por lo verdaderamente importante hacerme feliz.

A Rocio por aparecer en un momento tan importante, acompañarme y ayudarme tanto, y por darme tanta luz en tiempos de oscuridad, tu has hecho que vea la luz al final del túnel. Gracias de corazón.

Y, por último, con estas líneas me gustaría hacer mención a todas las personas que por una razón u otra ya sea buena o mala me han aportado para ser hoy la persona que soy.

A todos y cada uno de ustedes, simplemente gracias, os estare eternamente agradecido.

Pablo José Domínguez Camacho

Aljaraque, 2020

Resumen

En los últimos años, los datos se han convertido en el elementro principal para impulsar la calidad y eficiencia de cualquier sistema de comunicación. La toma de estos datos en ocasiones es sencilla, pero pueden incidir diversos factores que hagan de esta una tarea de una dificultad considerable, por ello surgen los sistemas de adquisición de datos.

En este trabajo se desarrolla un sistema de adquisición de datos a partir de un robot de limpieza concretamente el iRobot Create 2 Roomba 600 que junto con una Raspberry Pi conseguiremos desarrollar un programa que nos permita tener su control y obtener datos de nuestro entorno. Para su desarrollo hemos realizado un estudio de los diferentes robots disponibles, continuando con un estudio de los diferentes elementos que componen nuestro sistema, tras ello hemos planteado y comprobado su funcionamiento y, para terminar, hemos redactado unas líneas con las que hemos concluido el trabajo y hemos propuestos distintas mejoras a desarrollar en un futuro.

Abstract

In recent years, data has become the main element in boosting the quality and efficiency of any communication system. The collection of this data is sometimes simple, but various factors can make this a very difficult task, which is why data acquisition systems have emerged.

In this project a data acquisition system from a cleaning robot is developped, specifically the robot used is the iRobot Create 2 Roomba 600, which together with a Raspberry Pi we will manage to develop a program that will allow us to have control over it and obtain data from our environment. For its development we have made a study of the different available robots, continuing with a study of the different elements that make up our system, after that we have proposed and checked its operation and, finally, we have written some lines with which we have concluded the work and have proposed different improvements to be developed in the future.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGOS

1 INTRODUCCIÓN

You never fail until you stop trying

Albert Einstein

1.1. Motivación

Este proyecto surge de la idea de transformar un dispositivo que se está ganando un hueco en el hogar de muchas personas, como es un robot de limpieza, en un sistema de recolección de datos.

La necesidad de obtener datos de nuestro entorno para conocer mejor lo que nos rodea se ha convertido en un problema en determinadas situaciones dado que nos podemos encontrar con escenarios marcados por ser entornos de muy difícil acceso para los humanos o en algunos casos imposible ya sea por razones de lejanía como puede ser la exploración de la superficie de un planeta o de peligrosidad como puede ser zonas altamente radiactivas, entre otros factores. Esto hace que cobren una gran importancia los sistemas recolectores de datos.

El elemento sobre el que se crea estos sistemas es un robot, la rama encargada de su estudio se ha desarrollado y evolucionado a lo largo de los últimos años, de tal forma que actualmente juega un papel importante en la sociedad. El objetivo al que intentan llegar es la creación de robots autónomos e inteligentes capaces de adoptar comportamientos, razonar y actuar como seres vivos jugando un papel importante la inteligencia artificial lo que supondría un gran avance para el empleo de estos en sistemas de adquisición de datos.

En un futuro próximo, los hogares y los puestos de trabajo estarán formados por sistemas automatizados y de robótica que a día de hoy se están haciendo un hueco en la sociedad, estos ademas de prestar servicios en la industria haciendo el trabajo desarrollado por personas serán capaces también de prestar servicios a ellas. Con ello se puede apreciar el impacto que la robótica está obteniendo en la sociedad, por lo que la realización de un proyecto en el que se emplee un dispositivo de está rama lo hace aun mas interesante.

El desarrollo del sistema de recolección de datos trata muchos de los temas que durante la carrera se han estudiado de manera teórica y que en ocasiones se han podido llevar a la práctica, pero de una forma no tan compleja ni tan compacta ya que se trataban los aspectos por separado. Por ello abordar un proyecto que incluya muchos de los temas tratados en el área de las telecomunicaciones tomando como parte de él un dispositivo doméstico suponía un reto para mí.

La idea era interesante por dos vertientes, por un lado, por aplicar los conceptos estudiados y asimilados durante el grado y por otro, por profundizar en el estudio de sistemas de recolección de datos que juegan un papel importante en los sistemas de telecomunicaciones de hoy en día. Estas dos vertientes se podrían considerar la motivación principal del proyecto.

1.2. Objetivos

La finalidad de este proyecto es desarrollar de un sistema de adquisición de datos, del cual podamos tener un control completo, es decir, tengamos el poder de decidir sus movimientos y además los datos de interés a obtener.

Para llevar a cabo lo anteriormente mencionado es necesario establecer unos objetivos específicos que serán los siguientes:

- Estudiar el robot de limpieza para su posterior control.
- Preparar los componentes del sistema.
- Establecer comunicaciones entre los diferentes dispositivos que forman el sistema.
- Desarrollar un cliente/servidor usando como lenguaje de programación Python para el manejo del robot y sus sensores.
- Desarrollar una tarea específica para el robot.

Una vez conseguido los objetivos establecidos habremos alcanzado el objetivo principal del proyecto.

2 ESTADO DEL ARTE

En este capítulo introduciremos algunos robots existentes haciendo una diferenciación entre robots programables los cuales se caracterizan por tener sus propios mecanismos de control, en el que hablaremos en último lugar de un robot que sirve de nexo de unión con los robots controlados por Raspberry Pi con los que finalizaremos el capítulo ya que este presenta la alternativa de poder ser controlado además por una Raspberry Pi.

2.1. Robots programables

2.1.1 JETBOT

Es un robot de código abierto basado en el NVIDIA Jetson Nano, un ordenador de placa reducida o simple board computer (SBC), el cual es mucho más potente que la Raspberry Pi y no presenta un consumo mayor que esta. Este ordenador se caracteriza por tener una GPU Nvidia Maxwell de 128 núcleos, una CPU ARM Cortex-A57 MPCore de 4 núcleos, 4GB de RAM y como puertos los siguientes: HDMI, 4 USB 3.0, USB 2.0 Micro-B, GPIO, I2C, I2S, SPI y UART. Con respecto al software, cabe destacar que es compatible con los frameworks de inteligencia artificial (IA) más populares en la actualidad como son TensorFlow, PyTorch y Caffe entre otros.

Este robot se compone de una estructura sobre la que se instala la placa NVIDIA Jetson Nano, una batería para un funcionamiento autónomo, una cámara, tres ruedas y controladores. El principal uso para el que se ha desarrollado este robot es para, mediante el uso de inteligencia artificial, perseguir objetos, reconocerlos y esquivarlos.

Para acceder al control de este robot es necesario el uso de un teclado, un ratón y una pantalla para poder conectarlo a una red Wi-Fi y usarlo de forma remota. La imagen del sistema operativo incluye un servidor Jupyter en ejecución por lo que el acceso a él es sencillo.

Figura 2-1. Jetbot

2.1.2 ROBOMASTER S1

Se trata de un robot educativo avanzado de DJI, una empresa china de tecnología, líder mundial en la fabricación de vehículos aéreos no tripulados.

Este robot posee un controlador inteligente el cual es posible programar, cámara, sensor de sonido, seis puertos PWM, un puerto SBus, sensores de infrarrojos, sensor detector de impacto, lanzador, estabilizador y batería inteligente.

Emplea la inteligencia artificial para reconocer gestos, sonidos y otros robots, por lo que se puede programar para que además de reconocer lo anterior reconozca y siga marcadores, personas y líneas. La programación de este robot es posible realizarla tanto en Scratch como en Python y se accede a él mediante la aplicación Robomaster desarrollada por el fabricante, la conexión se realiza vía Wi-Fi.

Figura 2-2. Robomaster S1

2.1.3 Ziwo

Se trata de un robot programable de carácter educativo de la marca BQ dedicada al diseño, venta y distribución de smartphones, tabletas, impresoras 3D y kits de robótica.

Este robot ha sido desarrollado con la finalidad de que los niños se inicien en el mundo de la tecnología y la programación, esta basado en un diseño y hardware con licencia Creative Commons – Attribution - Share Alike lo que permite personalizarlo con piezas compatibles u obtenidas mediante una impresora 3D.

Posee una matriz de 30 LEDs con la siguiente distribución 5x6, que se encuentran en la boca del robot, también cuenta con sensores ultrasónicos, los cuales son usados para evitar obstáculos, sensores de sonido para reconocer y reaccionar ante distintos sonidos, y con una placa controladora desarrollada por la propia marca que es compatible con Arduino, aumentando las posibilidades de uso de otros sensores.

Para la programación de este robot se emplea Bitbloq un lenguaje visual basado en la programación mediante bloques creado por la marca y con bastante similitud a Scratch, aunque la posibilidad de emplear una placa de Arduino le habre las puertas a otros lenguajes de programación.

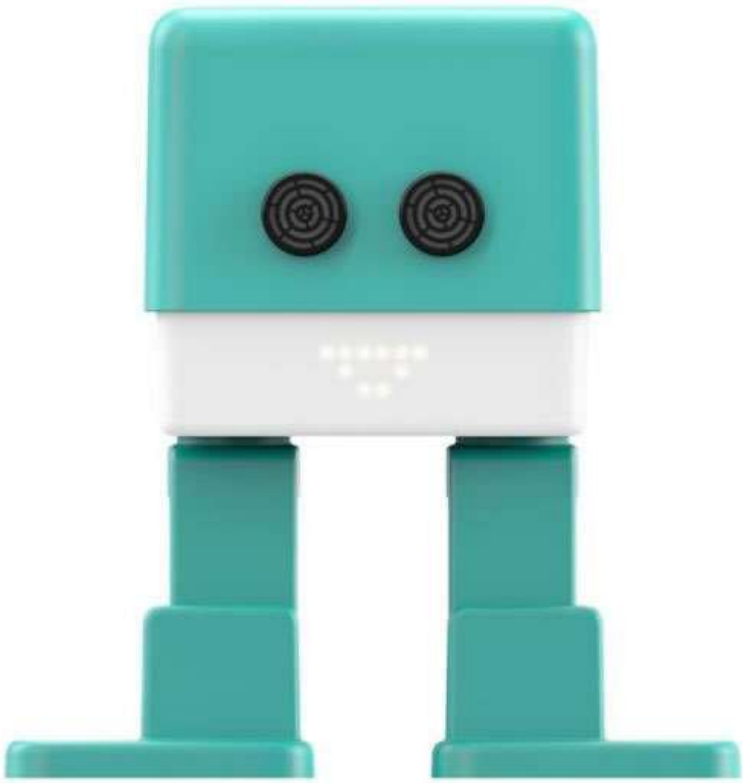

Figura 2-3. Ziwo

2.1.4 RVR

Se trata de un robot programable con finalidad educativa de la empresa Sphero, dedicada a la fabricación de robots programables y herramientas educativas.

Este robot ha sido desarrollado para poder moverse por cualquier tipo de superficie y cuenta con un gran número de sensores incorporados entre los que destacan sensores de luz ambiental, de color, de infrarrojos, acelerómetro, magnetómetro y giróscopo. Además, posee un puerto de expansión UART de cuatro pines que lo hace compatible con Arduino, placa micro:bit o Raspberry Pi.

Para la programación de dicho robot es posible recurrir a varios lenguajes de programación como puede ser Java o Python o también es posible hacer uso de la aplicación Edu, desarrollada por la propia empresa.

Finalmente, con este robot, que incluye la posibilidad de ser controlado mediante una Raspberry Pi, damos pasos a la siguiente al siguiente epígrafe en el que hablaremos de robots que han sido fabricados para ser controlados mediante una Raspberry Pi.

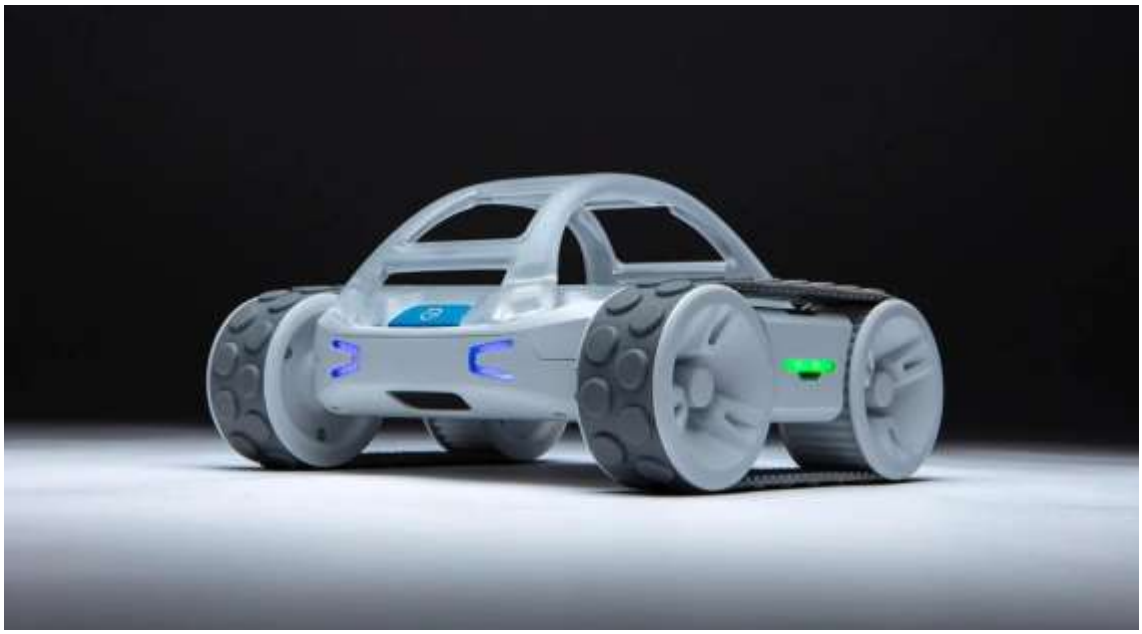

Figura 2-4. RVR

2.2. Robots controlados por Raspberry Pi

2.2.1 PiBot-B

Proyecto llevado a cabo por un alemán llamado Thomas Schoch consiste en un pequeño robot móvil controlado por la SBC Raspberry Pi. Los componentes por los que está formado dicho robot son una Raspberry Pi modelo B, una cámara web de la marca Logitech, un kit de chasis Zumo, una matriz de LED de Adafruit que indica el estado del robot, una batería externa para la alimentación de la Raspberry, un adaptador WLAN, dos motores para las ruedas y un controlador de motor.

Este robot está controlado mediante una aplicación para iPhone que se comunica a través de Wi-Fi con la Raspberry Pi en la que se ejecuta el servidor web lighttpd. En la aplicación se diferencian dos zonas, una zona inferior dedicada al control del movimiento del robot y otra superior en la que se muestra la imagen obtenida por la cámara web.

Los movimientos del robot han sido programados en Python usando la biblioteca WiringPi, la cual se usa para enviar señales al controlador del motor que se encarga de enviar las señales a los motores de las ruedas según el movimiento que se quiere realizar.

Por otra parte, Thomas Schoch también desarrollo otro proyecto con similares características llamado PiBot-A, en el cual el robot móvil estaba controlado por una Raspberry Pi modelo A. Este robot móvil se diferenciaba del anterior en que se movía por dos ruedas dispuestas de manera axial y se apoyaba tanto en la parte delantera como en la trasera en una bola deslizante lo que lleva a este modelo a moverse por superficies lisas, además contiene en la parte delantera una regleta a la que se le puede conectar una serie de sensores como puede ser un sensor de obstáculos o un sensor para el seguimiento de líneas. La mayor diferencia entre usar un modelo de Raspberry u otro reside en que el modelo B solo tiene una salida PWM lo que hace necesario el empleo de un conmutador PWM mientras que el modelo A tiene dos salidas PWM y puede emplearse una para cada rueda.

Figura 2-5. PiBot-B

Figura 2-6. PiBot-A

2.2.2 DarkPaw

Es un robot araña cuadrúpedo biónico de código abierto basado en Raspberry Pi desarrollado por Adeept, un equipo de servicio técnico de software y hardware dedicado a la aplicación de Internet y la última tecnología industrial al área de código abierto.

Este robot está compuesto por una estructura de metal que le aporta la forma de araña en la que se incluyen una Raspberry Pi, cuyos modelos compatibles son el modelo 2B, 2B+, 3B y 3B+, un módulo de sensor giróscopo, una cámara, faros LED, un módulo de LED RGB y un Adeept Robot HAT.

La programación del robot se lleva a cabo mediante el lenguaje de programación Python, con el que se consigue que el robot reconozca y siga a objetos y detecte movimientos gracias a la librería OpenCV. El control se lleva a cabo de forma remota mediante una interfaz gráfica de usuario (GUI) compatible con los sistemas operativos (SO) Windows y Linux, un servidor web o una aplicación para el teléfono móvil.

Figura 2-7. DarkPaw

2.2.3 PARSLEE

Este robot es un proyecto llevado a cabo por la científica de la NASA Jamie Molaro, el robot es una versión del rover Mars Curiosity, uno de los empleados por la NASA en sus misiones planetarias.

Surge del proyecto Rover de código abierto llevado a cabo por el laboratorio de investigación financiado por la NASA, Jet Propulsion Laboratory, y del enfoque adoptado en el trabajo de Jamie Molaro el cual es el estudio de superficies de cuerpos rocosos y helados sin presencia de aire como los cometas, asteroides y lunas que orbitan la Tierra, Júpiter y Saturno. Su nombre PARSLEE proviene de las siglas Planetary Analog Remote Sensor and Lil Electronic Explorer.

PARSLEE se compone de una estructura metálica con 6 ruedas que permiten el movimiento, una cabeza con una matriz de LED con la que se muestra su estado controlado por un arduino, un dispositivo portátil para la obtención de datos meteorológicos llamado Kestrel 5000, una cámara de la marca GoPro, controladores de motor ROBOCLAW, una batería que alimenta todos los componentes del robot y un sismómetro Raspberry Shake que está compuesto por un geófono, 3 acelerómetros y una Raspberry Pi.

El control del robot se lleva a cabo mediante un controlador de Xbox que se conecta a la Raspberry Pi, la cual actúa como el cerebro del robot. Todos los datos obtenidos por el sismómetro como por el Kestrel 5000 son enviados a un dispositivo portátil externo con una interfaz gráfica donde se analizan los resultados obtenidos. La conexión entre estos dispositivos se realiza mediante una red Wi-Fi.

Figura 2-8. PARSLEE

3 SITEMA DE ADQUISICIÓN DE DATOS

En este capítulo presentaremos el sistema propuesto y abordaremos los diferentes aspectos que lo conforman. Primero trataremos los aspectos hardware del sistema, tras ello comentaremos los aspectos software finalizando con el planteamiento de la conexión de los distintos componentes.

3.1. Hardware

3.1.1 iRobot Create 2

El iRobot Create 2 serie 600 es un robot de limpieza programable con finalidad educativa desarrollado por la empresa líder en robots de limpieza de hogar iRobot.

Figura 3-1. iRobot Create 2 Roomba 600

Este robot posee una interfaz abierta (OI) que es una interfaz de software para controlar y manipular sus comportamientos y leer sus sensores a través de una serie de comandos entre los que se incluyen modos de funcionamiento, comandos de actuador, comandos de limpieza y comandos de sensor que se envían mediante un cable serie al ordenador o microcontrolador que se le conecte.

El puerto serie al que se conecta el ordenador o microcontrolador presenta las siguientes características:

- Baudios: 11520 o 19200
- Bits de datos: 8
- Paridad: Ninguna
- Bits de parada: 1
- Control de flujo: Ninguno

Por defecto la Roomba se comunica a 115200 baudios por lo que si el microcontrolador usado no soporta estos baudios hay varias formas de cambiar este valor a 19200. La primera forma se consigue mediante la pulsación del botón de encendido durante varios segundos, una segunda forma se consigue mediante la clavija 5 del conector Mini-DIN que es la de cambio de velocidad o también es posible configurar este valor mediante los comandos de la OI.

Los modos de funcionamiento que posee son los siguientes:

- Modo apagado. Modo en el que se encuentra el robot inicialmente o tras un cambio de batería. en este modo la OI escucha a la velocidad predeterminada para obtener un comando de inicio, una vez recibido dicho comando se pasa a uno de los otros tres modos.
- Modo pasivo. Este modo se activa al enviar el comando iniciar o cualquiera de los comandos de limpieza. En él, se puede obtener datos de los sensores, pero no se puede configurar los parámetros de los comandos actuadores, para cambiar estos parámetros es necesario pasar al modo seguro o al modo completo. Además, en este modo la Roomba entra en modo ahorro de energía después de 5 minutos de inactividad para conservar la batería.
- Modo seguro. Modo que se activa al ejecutar el comando con dicho nombre. Este modo permite un control completo de la Roomba a excepción de los siguientes casos: detección de acantilado, detección de caída de rueda y cargador enchufado y encendido. En estos casos, todos relacionados con la seguridad, se detienen todos los motores y se pasa automáticamente al modo pasivo.
- Modo completo. Modo que se activa al enviar el comando completo. En este modo se da un control completo sobre la Roomba, es decir, sobre todos sus actuadores y sensores. Este modo apaga las características de seguridad anteriormente en el modo seguro, para poder volver al modo seguro es necesario enviar el comando seguro a la Roomba.

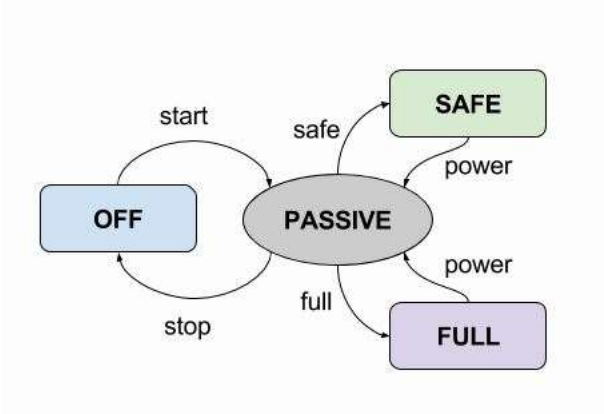

Figura 3-2. Diagrama de flujo de modos

Los comandos que se emplean para la Roomba son muy variados y están clasificados según el tipo de acción. A continuación, se muestran una serie de tablas en las que se menciona el comando, el número de secuencia en serie y una breve descripción. Para entrar en más detalle consultar: iRobot® Create® 2 Open Interface (OI) Specification based on the iRobot® Roomba® 600

Tabla 3-1. Comandos de inicio

Comando	Opcode	Descripción
Start	128	Comando que inicia la OI, siempre hay que enviar este comando antes de enviar cualquier otro.
Reset	7	Comando que reinicia el robot, tras su ejecución el robot pasa al modo apagado.
Stop	173	Detiene la OI. Todos los flujos se detienen y el robot deja de responder a los comandos.
Baud	129	Establece la tasa de baudios en bits por segundos a la que se envían los comandos y datos de la OI según el código de baudios enviado en el byte de datos.

Tabla 3-2. Comandos de modo

Comando	Opcode	Descripción
Safe	132	Cambia el modo de la Roomba al modo seguro que permite al usuario su control. Sin embargo, si se da una condición de seguridad se vuelve al modo pasivo.
Full	131	Cambia el modo actual al modo completo brindando total control sobre ella. Es posible usar este comando en los modos pasivo y seguro.

Tabla 3-3. Comandos de limpieza

Comando	Opcode	Descripción
Clean	135	Inicia el modo de limpieza predeterminado. Realiza la misma función que se consigue al presionar el botón de limpieza.
Max	136	Inicia el modo de limpieza Max que solo se detendrá si se agota la batería. Pausara un ciclo de limpieza si ya hay uno en curso.
Spot	134	Inicia el modo de limpieza puntual. Realiza la misma función que al pulsar el botón puntual del robot.
Seek Dock	143	Indica a la Roomba que se dirija a la base la próxima vez que se encuentre con ella. Misma función que el botón de la Roomba Dock y pausara un ciclo de limpieza si ya hay uno en progreso.
Power	133	Comando que apaga la Roomba. La OI puede estar en modo pasivo, seguro o completo para aceptar este comando.
Schedule	167	Envía a la Roomba una nueva programación. Para deshabilitar la limpieza programada enviar el comando con todos los 0.
Set Day/Time	168	Comando que configura el reloj de la Roomba.

Tabla 3-4. Comandos actuadores

Comando	Opcode	Descripción
Drive	137	Controla las ruedas motrices de la Roomba. Este comando envía la velocidad de las ruedas en milímetros por segundo y el radio en milímetros que girará. Velocidad positiva y radio positivo hace que gire a la izquierda mientras avanza mientras que si la velocidad es negativa retrocede y si el radio es negativo gira a la derecha.
Drive Direct	145	Permite controlar el movimiento hacia adelante y hacia atrás de las ruedas motrices. Este comando envía la velocidad de la rueda derecha y de la rueda izquierda por separados en milímetros por segundo. Para velocidades positivas avanza y para velocidades negativas retrocede.
Drive PWM	146	Permite controlar el movimiento puro hacia adelante y hacia atrás de las ruedas motrices independientemente. Este comando envía el PWM de la rueda derecha y el PWM de la rueda izquierda. Para PWM positivos la Roomba avanza y para PWM negativos retrocede.
Motors	138	Permite controlar el movimiento hacia adelante y hacia atrás tanto del cepillo principal como del cepillo lateral y aspirar de forma independiente. La velocidad del motor no se puede controlar con este comando por lo que cada vez que se use se ejecutara a velocidad máxima.
PWM Motors	144	Permite controlar la velocidad del cepillo principal, el cepillo lateral y de la Roomba. Las velocidades positivas hacer girar el motor en la dirección de limpieza, es decir, hacia dentro de la Roomba.

LEDs	139	Controla los LEDs, pudiendo confirar el color y la intensidad.
Buttons	165	Comando que permite presionar los botones de la Roomba. Los botones se soltarán automáticamente después de 1/6 de segundo.
Digit LEDs ASCII	164	Permite controlar los 4 displays de 7 segmentos que se hallan en la Roomba.
Song	140	Permite especificar hasta cuatro canciones en la OI que se puede reproducir en un momento posterior. Cada canción es asociada con un numero de canción.
Play	141	Selecciona una canción para reproducir de las canciones agregadas usando en comando Song.

Tabla 3-5. Comandos de entrada

Comando	Opcode	Descripción
Sensors	142	Solicita al OI que le envíe un paquete de bytes de datos del sensor. Hay 58 paquetes de datos de sensores diferentes. Cada uno proporciona un valor de un sensor especifico.
Query List	149	Permite solicitar una lista de paquetes de sensores. El robot devuelve los paquetes en el orden especificado.
Stream	148	Inicia una secuencia de paquetes de datos. La lista de paquetes solicitados se envía cada 15 ms, que es la tasa que la Roomba usa para actualizar los datos.
Pause/Resume Stream	150	Permite detener y reiniciar el flujo de datos sin borrar la lista de paquetes solicitados.

Los datos de los sensores se obtienen una vez obtenido los paquetes de datos mediante uno de los comandos mencionados en la Tabla 3-5. Para acceder a estos datos se utilizan una serie de comandos cada uno para obtener los datos de un sensor concreto. A continuación, se muestra en una tabla los sensores mas significativos. Para obtener mayor información y conocer el resto de comandos consultar iRobot® Create® 2 Open Interface (OI) Specification based on the iRobot® Roomba® 600

Tabla 3-6. Comandos obtención de datos sensores

Comando	Packet ID	Descripción
Bumps and Wheel Drops	7	Devuelve el estado del parachoques y los sensores de caída de las ruedas.
Wall	8	Devuelve el estado de sensor de pared.
Cliff Left	9	Devuelve el estado del sensor de desnivel izquierdo.
Cliff Right	12	Devuelve el estado del sensor de desnivel izquierdo.
Dirt Detect	15	Devuelve el nivel del sensor de detección de suciedad.
Distance	19	Devuelve la distancia recorrida por la Roomba en milímetros desde la última vez que se solicitó.
Angle	20	Devuelve el ángulo girado por la Roomba en grados desde la última vez que se solicitó.
Light Bumper	45	Devuelve el estado de los sensores del parachoques ligero.

Por último, se muestra una vista superior y una vista inferior donde se pueden apreciar el robot y la ubicación de los diferentes sensores y elementos que lo componen.

Figura 3-3. Vista superior del iRobot Create 2

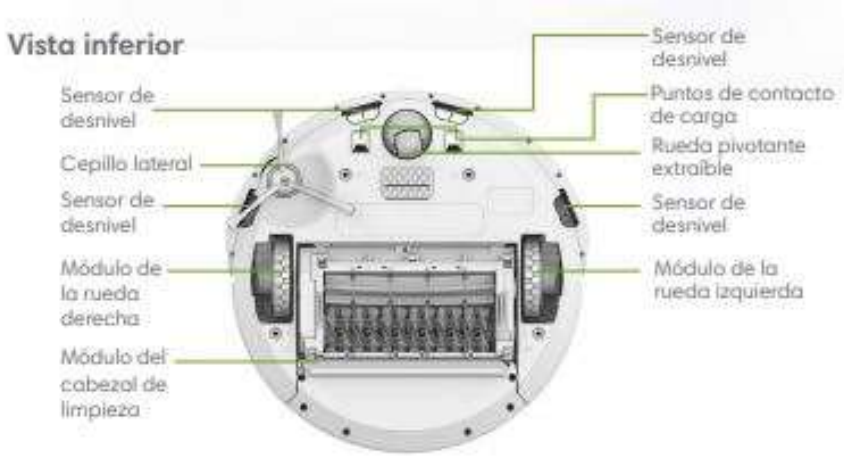

Figura 3-4. Vista inferior del iRobot Create 2

3.1.2 Raspberry Pi

La Raspberry Pi es una placa del tipo SBC de bajo coste, fue desarrollada en el Reino Unido por parte de la Fundación Raspberry Pi de la Universidad de Cambridge con el fin de fomentar el aprendizaje de la informática en la educación que hasta el año 2012 no se comenzó a comercializar.

La Raspberry Pi 3 modelo B es el elegido para el sistema de adquisición de datos, el modelo 3B es el primer modelo de la tercera generación. Este correrá Raspbian un sistema operativo (SO) gratuito basado en Debian una distribución de Linux. Este SO esta optimizado para el hardware de Raspberry Pi.

A continuación, se muestra una imagen donde se puede identificar sus elementos principales.

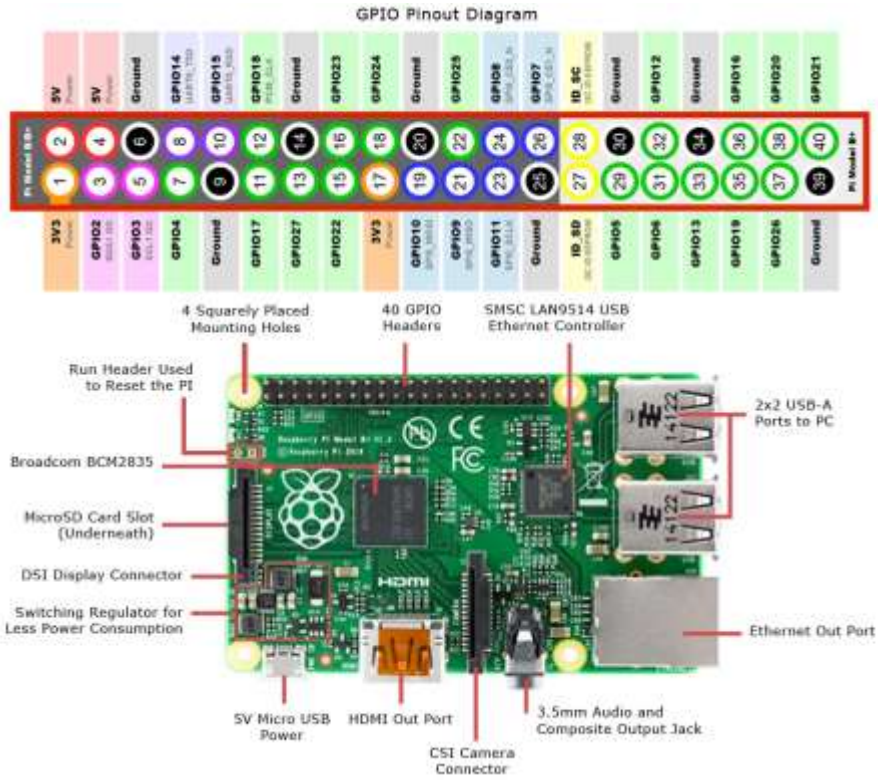

Figura 3-5. Diagrama de elementos Raspberry Pi 3B

Las características principales de esta tarjeta son las siguientes:

- SoC: Broadcom BCM2837
- CPU: 4× ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Red: 10/100 Ethernet, 2.4GHz 802.11n wireless
- Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
- Almacenamiento: microSD
- GPIO: 40-pin header, populated
- Interfases externas: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

3.1.3 Batería externa

Para la alimentación de la Raspberry Pi 3B se ha usado una batería externa. Para su elección se ha tenido en cuenta que la entrada de alimentación de la Raspberry Pi es 5V/2.5A por lo que la batería debería aportar 5V en tensión y al menos 2.5A de corriente.

Por ello, la batería externa usada es la Pilot X7 de POWERADD en su versión mejorada. Las características de esta batería son las siguientes:

- Dimensiones: 15.2 x 8 x 2 cm
- Peso: 399.16 gramos
- Batería: 2 pilas de polímero de litio.
- Capacidad: 20000 mAh
- Ciclo de vida > 500
- Entrada: 5V/2ª
- Salidas: 2xUSB 5V/3.4A
- Carga rápida

Figura 3-6. Batería externa Pilot X7

3.2. Software

3.2.1 Python

Es un lenguaje de programación creado por el europeo Guido Van Rossum. Este lenguaje es interpretado por tanto los errores se dan en tiempo de ejecución.

Python es un lenguaje muy popular y esto se debe a que es de propósito general, es decir, con él se puede crear cualquier tipo de programa; es multiplataforma ya que es compatible con cualquier sistema operativo siempre que exista un intérprete programado para él; como se mencionó anteriormente es interpretado; además, es interactivo debido a que dispone de un intérprete por línea de comandos en el que se puede introducir sentencias; es orientado a objetos lo que ofrece en muchas ocasiones una reutilización del código; posee una gran variedad de funciones y librerías y cabe destacar la sintaxis clara que posee.

Por lo comentado anteriormente es el lenguaje en el que se va a realizar la programación del código, la versión empleada es la v3.7.3 que ya viene instalada en la imagen del SO cargada en la Raspberry.

3.2.2 Librería pycreate2

Es una librería de Python para controlar el iRobot Create2. Esta librería fue usada en la clase de introducción a la robótica impartida en la academia de la fuerza aérea de los EEUU.

Pycreate2 desarrolla en lenguaje de programación Python los comandos del iRobot Create2 anteriormente comentados, haciendo que el desarrollo de código empleando estos comandos sea sencillo. Esto corrobora varias de las características de Python que mencionamos como la variedad de librerías, ser de propósito general, ser multiplataforma, etc.

Para la instalación de esta librería es necesario tener instalado el pip, un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.

En primer lugar, para instalar el pip será necesario ejecutar el siguiente comando:

```
sudo apt-get install python3-pip
```

En segundo lugar, se instala la librería ejecutando lo siguiente:

```
pip install pycreate2
```

Una vez se ha realizado lo anterior ya es posible hacer uso de dicha librería. Aunque se puede ir más allá y participar en el desarrollo de la librería realizando lo siguiente:

```
git clone https://github.com/walchko/pycreate2
cd pycreate2
poetry install
poetry run pytest -v
```

3.2.3 Cliente/Servidor

La arquitectura cliente/servidor es una estructura que tiene dos partes bien diferenciadas. Por un lado, se encuentra la parte del servidor el cual es el encargado de recibir peticiones y responder a ellas. Por otro, está la parte del cliente que es el encargado de demandar servicios y recibir una respuesta a esa demanda.

Tras este primer análisis de una arquitectura cliente/servidor se ve que se asemeja a la forma que tiene nuestro sistema de adquisición de datos siendo esto un impulso para llevar a cabo un cliente/servidor que responda a una arquitectura de dos capas que se describe como aquella en la que el cliente solicita recursos y el servidor responde directamente a la solicitud con sus propios recursos.

Para que sea posible una comunicación entre ambas partes es necesario un canal de comunicación bidireccional y es en este punto en el que destaca el papel de los sockets que representar los extremos del canal de comunicación. Los sockets se pueden comunicar dentro de un mismo proceso, entre procesos dentro de la misma maquina o entre procesos de máquinas diferentes. Como comentamos anteriormente el lenguaje utilizado es Python por lo que se hace uso de la librería socket de este lenguaje de programación.

El cliente de nuestro sistema de adquisición de datos responde a una estructura sencilla en la que se comienza importando las librerías necesarias, como son la librería de sockets para poder establecer un canal de comunicación y la librería sys.

Tras ello se declara el host y el puerto al que se desea conectar y se realiza la conexión y su comprobación, para ello se emplean las funciones socket() y connect(). Con la función sockect() se crea el socket, a esta funcion se le pasa la familia de direcciones a usar, en este caso la familia de direcciones IPv4 y el tipo de protocolo de transmición, para este caso el protocolo TCP, y con la función connect() se realiza la conexión con el servidor, a esta función se le pasa el host y el puerto declarados. Una vez realizado lo anterior se espera la recepción de un mensaje del servidor que confirme que la conexión se ha establecido con éxito, para la recepción de este mensaje se emplea la función recv().

Seguido se interactúa con el servidor mediante un bucle while sin condición de entrada en el que se envía un comando y se analiza la respuesta. En primer lugar, se pide la introducción de un comando para su envio al servidor y posteriormente se recibe la respuesta del servidor la cual es analizada. Se contemplan tres posibilidades, que el mensaje recibido por parte del servidor sea “Comando incorrecto”, lo que lleva a solicitar un nuevo comado, que sea “Cerrando conexión...” condición de salida del bucle que lleva al cierre de la conexión porque el cliente ha realizado dicha petición o que el comando introducido sea correcto mostrándose por pantalla la respuesta por parte del servidor tras la ejecución de dicho comando, tras este mensaje se vuelve a pedir que se introduzca otro comando.

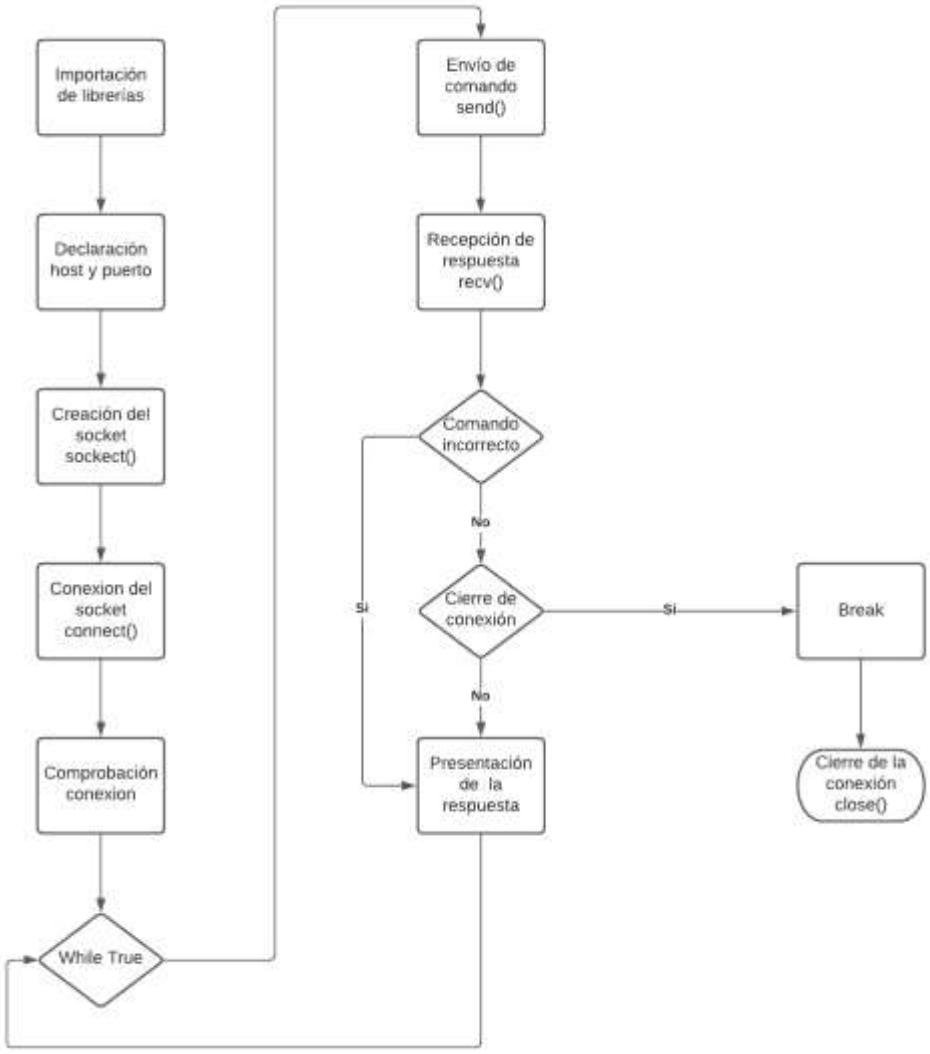

Figura 3-7. Diagrama de flujo del cliente

En cambio, el servidor del sistema responde a una estructura más elaborada, al igual que el cliente se comienza importando las librerías, pero teniendo en cuenta que el servidor interactúa con la Roomba para obtener los datos o ejecutar las ordenes enviadas por el cliente es necesario importar la librería pycreate2, además de la librería time para controlar algunos tiempos necesarios en la ejecución de acciones por parte de la Roomba.

Después, se declara el host y el puerto al que conectarse, se crea la conexión con la Roomba, en la que es necesario indicar el puerto serie, en este proyecto el puerto serie se encuentra en el fichero '/dev/ttyUSB0', se arranca y se inicia en el modo seguro.

Una vez realizado esto se crea el socket, se realiza la conexión con el cliente y se comprueba que la conexión se ha establecido. En el lado del servidor existen cosas similares a las comentadas con el cliente en este aspecto como la creación del socket pero también existen diferencias, las cuales se comentan a continuación. En el caso del servidor es este el que acepta y crea las conexiones por lo que entra en juego la función bind() a la que se le pasa como parámetros el host y el puerto con el que se consigue asociar el socket a una interfaz de red y numero de puerto, otra función con un papel importante es listen() con la cual es servidor espera las peticiones de conexión por parte de los clientes. Tras la ejecución de lo anterior y una vez recibida la petición de conexión mediante la funcion accept() se acepta y establece la conexión con el cliente.

Con la conexión establecida comienza la interacción entre cliente y servidor, en la que el servidor recibe un dato, lo analiza y envía su respuesta, sobre las que se hablará en capítulos posteriores. Finalmente, la conexión termina cuando el servidor recibe por parte del cliente una petición de cierre.

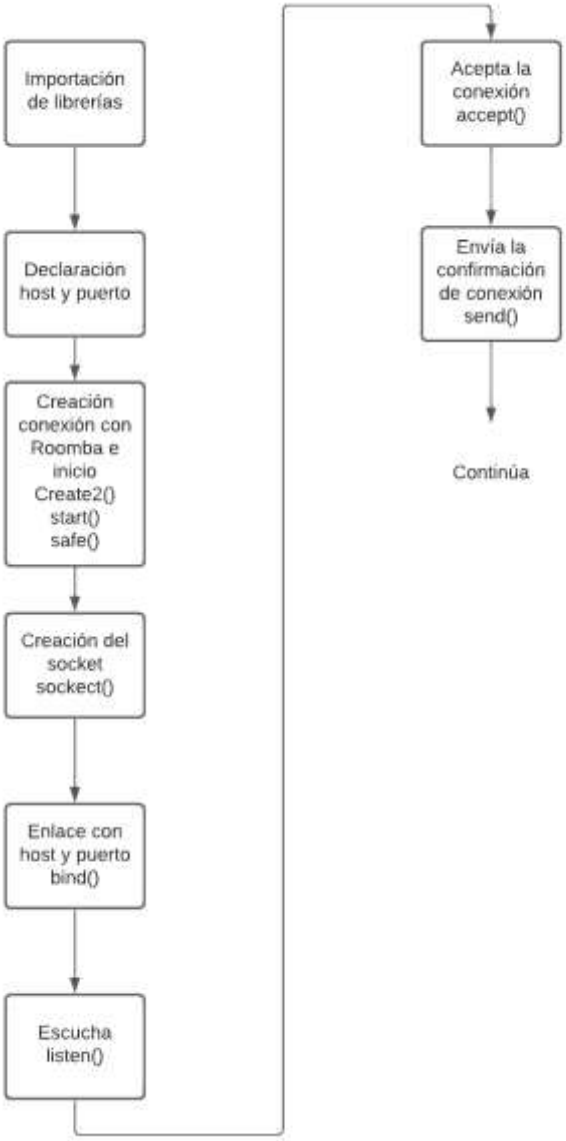

Figura 3-8. Diagrama de flujo servidor primera parte

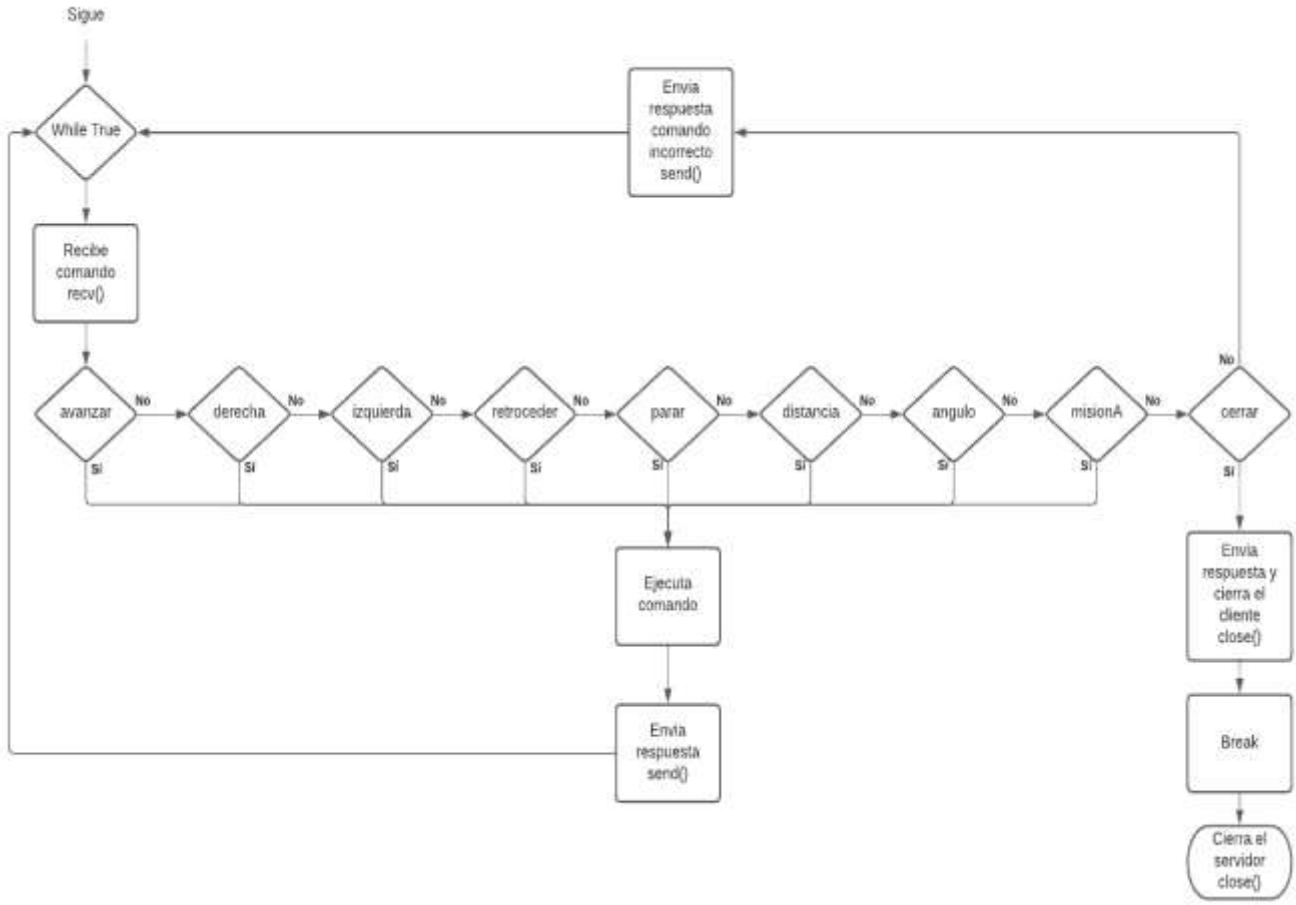

Figura 3-9. Diagrama de flujo servidor segunda parte

3.2.4 Red

Una red es un conjunto de nodos y software conectados entre sí por medio de un dispositivo físico o inalámbrico. En nuestro caso, el software lo acabamos de mencionar anteriormente y se trata de un cliente/servidor y los nodos que forman nuestra red los mencionamos a continuación. Por un lado, tenemos el ordenador en el que se ejecuta el cliente y donde recopilaremos la información y llevaremos el control del iRobot Create 2. Por otro lado, tenemos el nodo formado por el conjunto Raspberry Pi y el iRobot Create 2, el servidor se ejecutará en la Raspberry Pi pasándole y extrayendo la información necesaria del iRobot Create 2 a través del puerto serie.

Una vez conocidos los nodos que forman la red se elige la vía de conexión de ambos nodos. Dado que se pretende que el control del sistema sea remoto la vía elegida para la conexión entre ambos nodos es vía Wi-Fi. Esto permite que el módulo Raspberry Pi + iRobot Create 2 se mueva con total libertad sin las limitaciones que impondrían los dispositivos físicos de conexión con el ordenador. Para que la conexión entre ambos nodos sea satisfactoria estos deben estar conectados a la misma red Wi-Fi.

Figura 3-10. Red

4 RESULTATOS EXPERIMENTALES

En este capítulo llevaremos a cabo la puesta en marcha del sistema y la comprobación del correcto funcionamiento. En primer lugar, comentaremos como se ha realizado el conexionado de todos los elementos y los programas usados para ello. En segundo lugar, hablaremos del sistema de control manual realizado, su funcionamiento y su respuesta. Por último, expondremos las misiones programadas para el sistema, su ejecución y resultado.

4.1 Puesta a punto de los experimentos

Antes de poder realizar cualquier prueba de nuestro sistema es necesario que todo el conjunto esté conectado y preparado para su funcionamiento, por ello vamos a detallar paso a paso como hemos realizado el conexionado del sistema y que programas nos han sido útiles para conseguirlo.

En primer lugar, necesitamos una conexión entre la Raspberry Pi y el ordenador para que podamos visualizar el sistema operativo que se ejecuta en la Raspberry Pi. Para ello necesitamos tener instalado VNC (Virtual Network Connection) que es una aplicación de acceso remoto que permite tener disponible la interfaz gráfica del dispositivo al que estamos conectado. La instalación de esta aplicación es sencilla y se puede consultar en Raspberry Pi VNC

Para poder acceder con VNC a la interfaz gráfica de la Raspberry Pi necesitamos realizar la conexión entre ambos dispositivos. Para ello hemos necesitado en la instalación de la imagen del sistema operativo en la memoria SD incluir un fichero en el directorio de arranque sin extensión con el nombre de SSH, con esto conseguimos que al arrancar la Raspberry Pi se active este servicio ya que por defecto no está activado. Con este servicio podemos realizar conexiones encriptadas.

Una vez activado el servicio SSH podemos acceder a la Raspberry Pi mediante Putty que es un software que nos permite realizar conexiones de distintos tipos, entre ellas la conexión SSH, entre dos sistemas interconectados, su principal inconveniente es que el manejo de la Raspberry Pi se lleva a cabo con un terminal y no a través de una interfaz gráfica. Para poder acceder mediante VNC es necesario habilitarlo por ello la primera vez tenemos que acceder a ella mediante la aplicación Putty.

La conexión entre la Raspberry Pi y el ordenador para poder acceder mediante Putty se realiza mediante un cable ethernet, realizada esta conexión abrimos la aplicación que presenta la forma mostrada en la Figura 4-1, marcamos el tipo de conexión SSH, rellenamos el campo host con raspberrypi.local, este es el nombre por defecto al que responde dentro de una red, y abrimos la conexión.

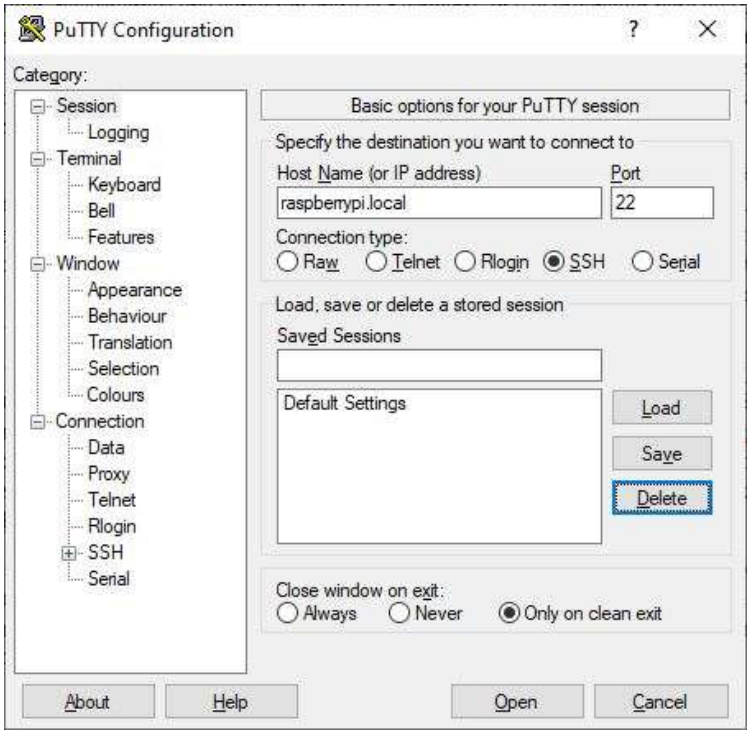

Figura 4-1. Configuración del Putty

Al abrir la conexión nos pide el usuario y contraseña, que por defecto es pi y raspberry. Tras ello estamos dentro de la Raspberry Pi y ejecutamos el comando que se muestra en la Figura 4-2 para acceder a la configuración de la Raspberry Pi.

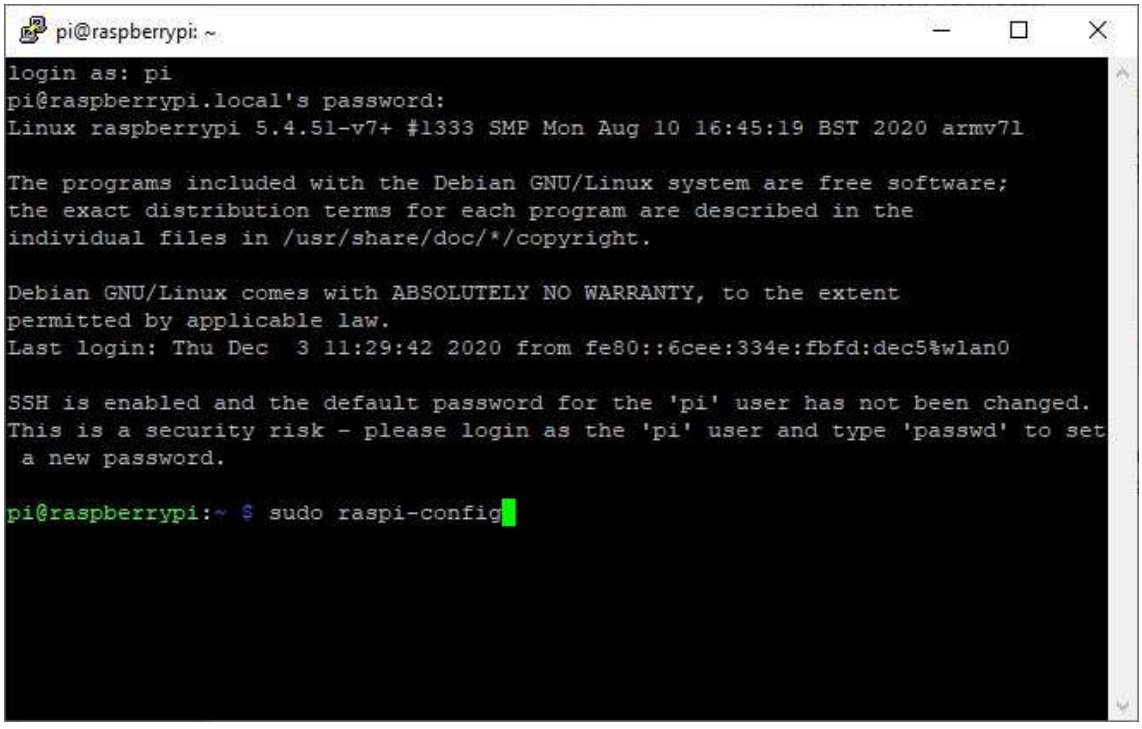

Figura 4-2. Establecimiento de la conexión y comando de configuración

Tras ejecutar el comando se nos abre el menú de configuración de la Raspberry Pi desde donde habilitaremos la aplicación VNC. Para ello marcamos las siguientes opciones Interfacing Options -> VNC -> Sí y con esto habremos habilitado el servidor VNC. Las siguientes imágenes muestran los pasos seguidos.

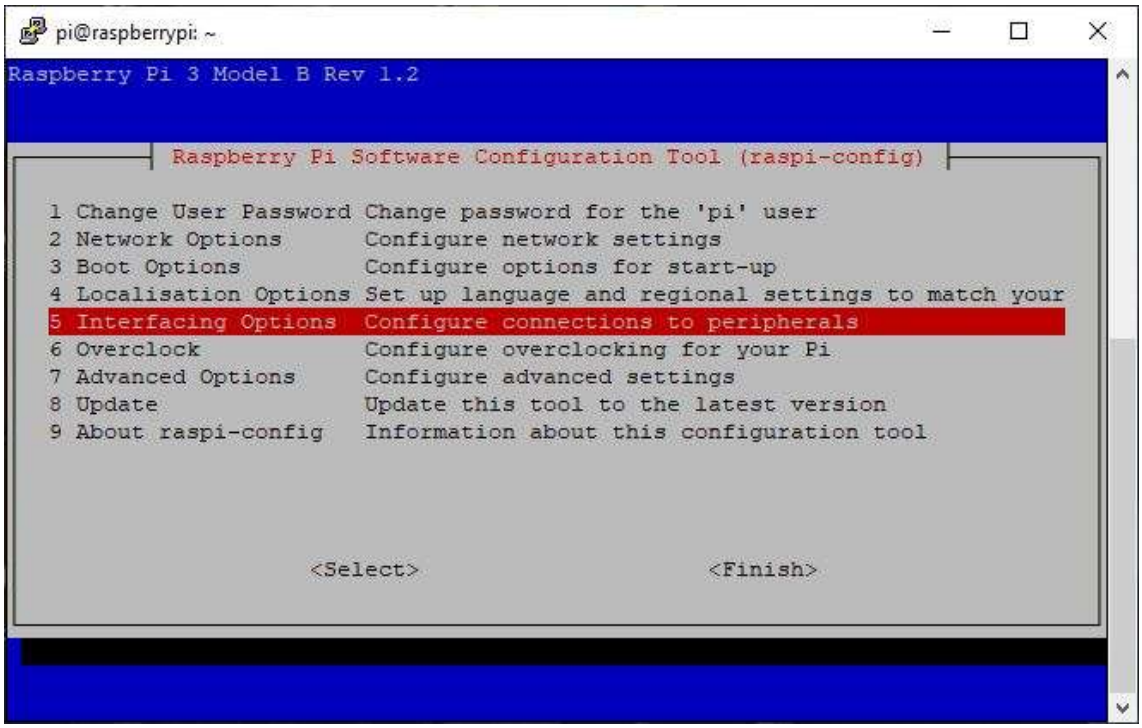

Figura 4-3. Menú de configuración

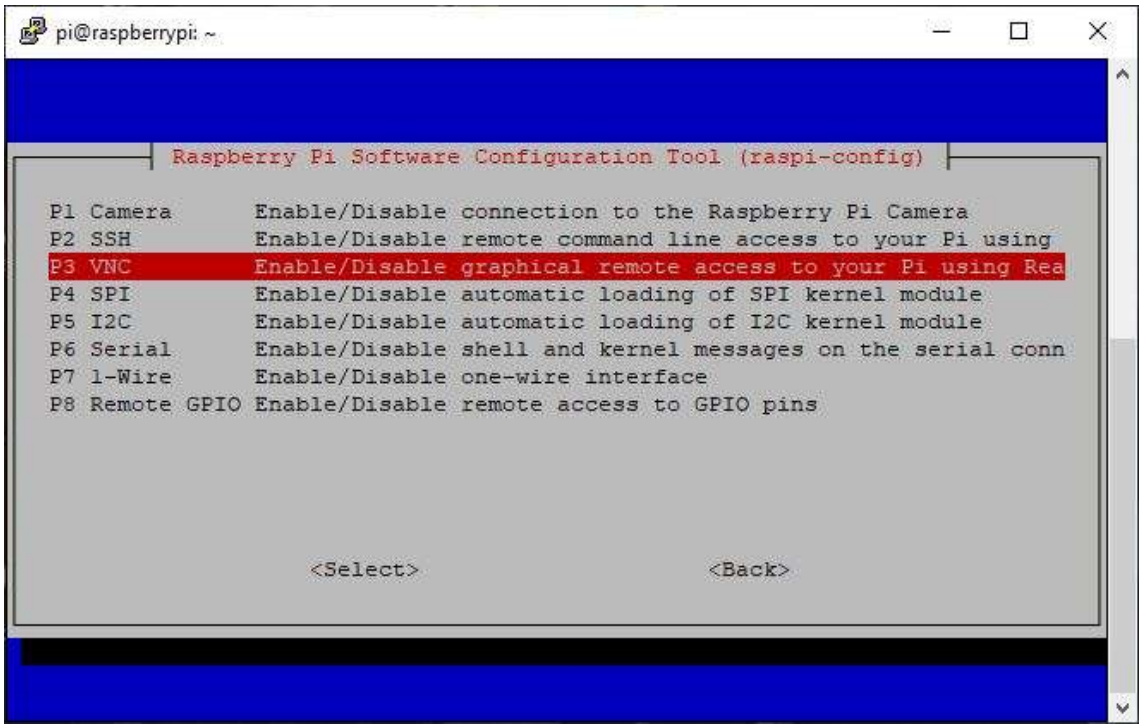

Figura 4-4. Opciones de interfaz

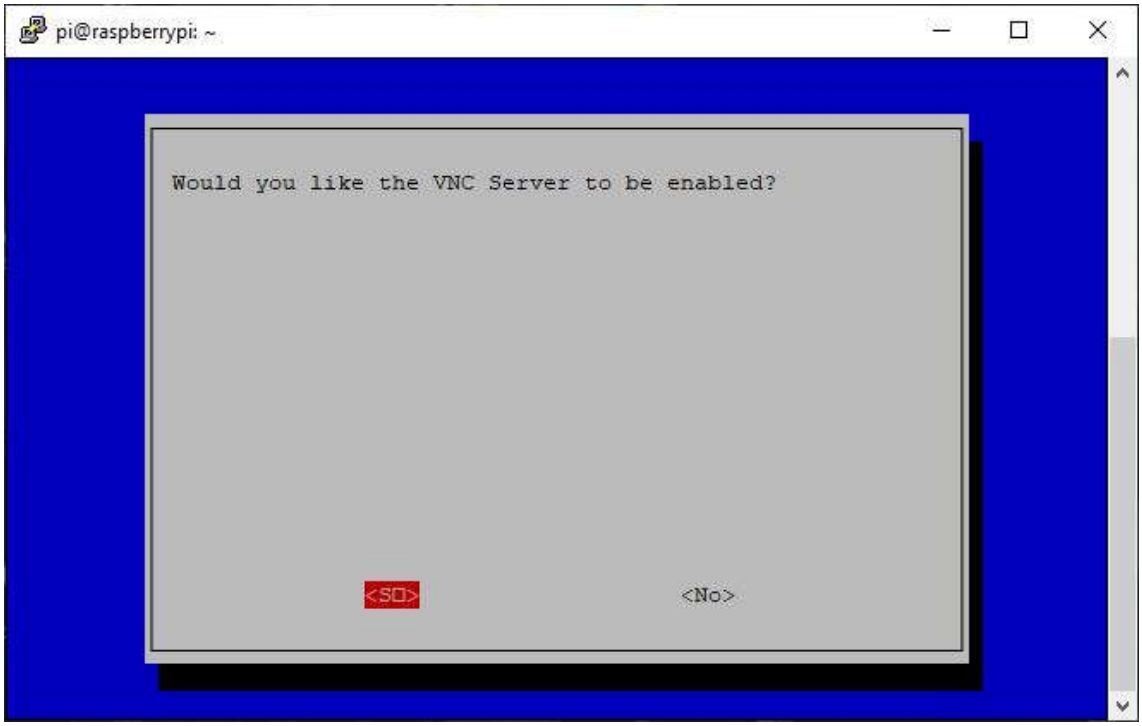

Figura 4-5. Confirmación de habilitación de VCN

Una vez hemos activado el servidor VNC ya es posible acceder de manera gráfica a la Raspberry Pi. Para ello abrimos la aplicación en el ordenador de VNC, que debemos haber instalado previamente, e introducimos el nombre por defecto usado en el Putty en la barra de direcciones de servidores de VNC como se muestra en la Figura 4-6 y accedemos, nos vuelve a pedir el usuario y la contraseña como se aprecia en la Figura 4-7 y finalmente accedemos a la Raspberry Pi de forma gráfica.

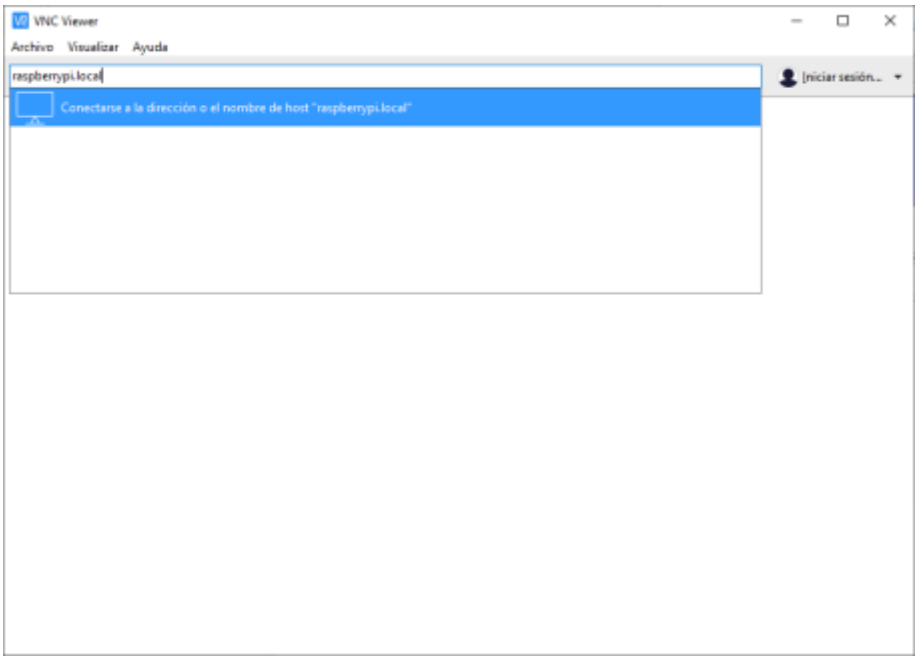

Figura 4-6. Introducción del servidor en aplicación VNC del portátil

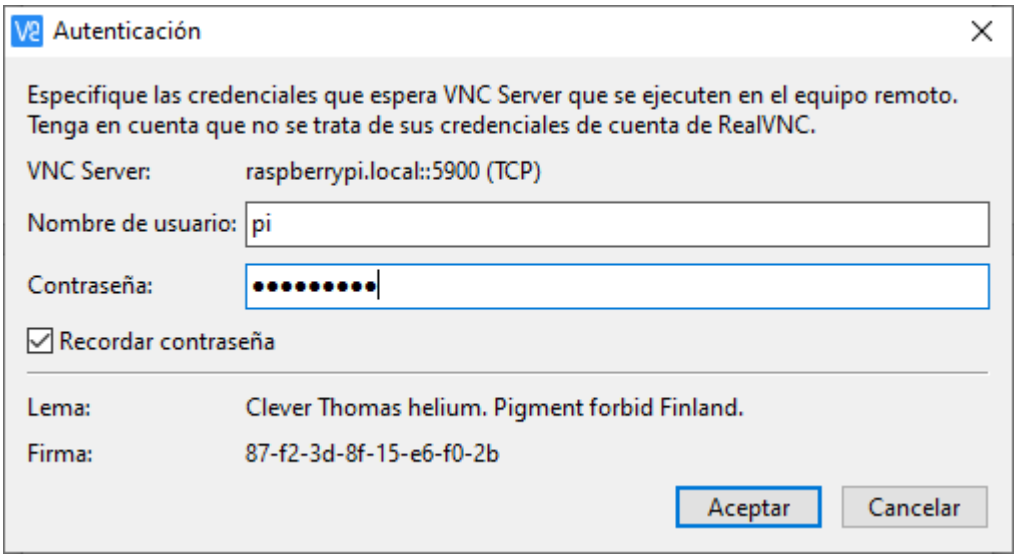

Figura 4-7. Autenticación VCN

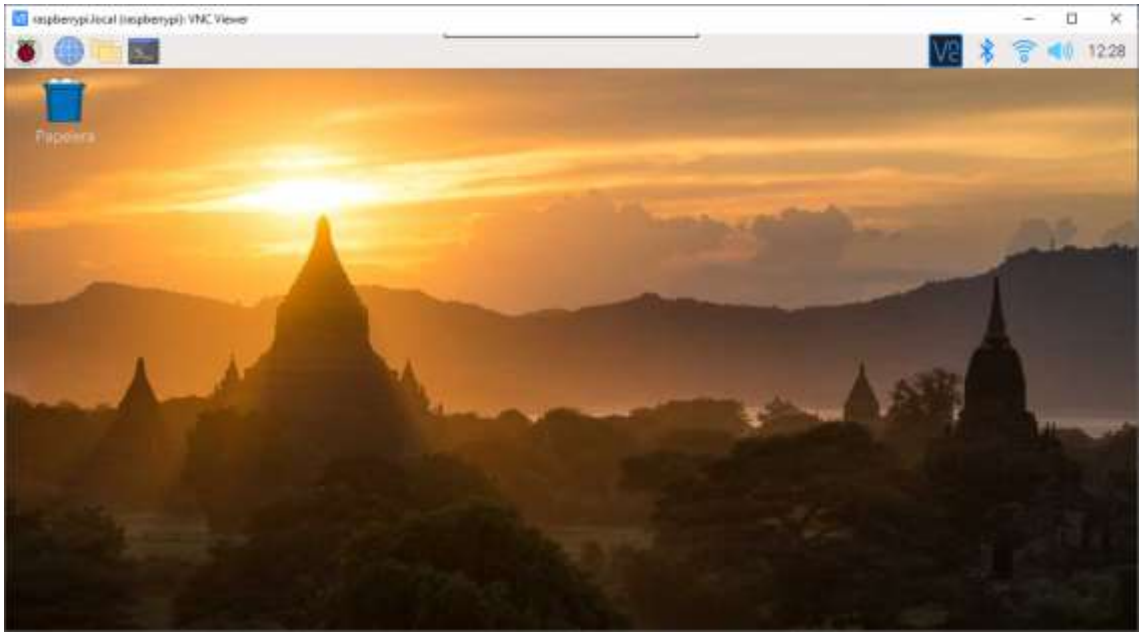

Figura 4-8. Escritorio Raspberry Pi

Para conseguir la conexión de manera remota y de esta manera prescindir del cable ethernet, llegados a este punto configuramos la conexión Wi-Fi de la Raspberry Pi y obtenemos su dirección IP, abriendo un terminal y ejecutando el comando ifconfig como se muestra en la Figura 4-9. Con esta dirección IP se abre otra ventana VNC como se realizó anteriormente y se establece la conexión, tras ello el sistema queda conectado de forma remota. Para las futuras conexiones solo habría que acceder a través de VNC a la Raspberry Pi mediante raspberrypi.local y realizar esto último que acabamos de comentar.

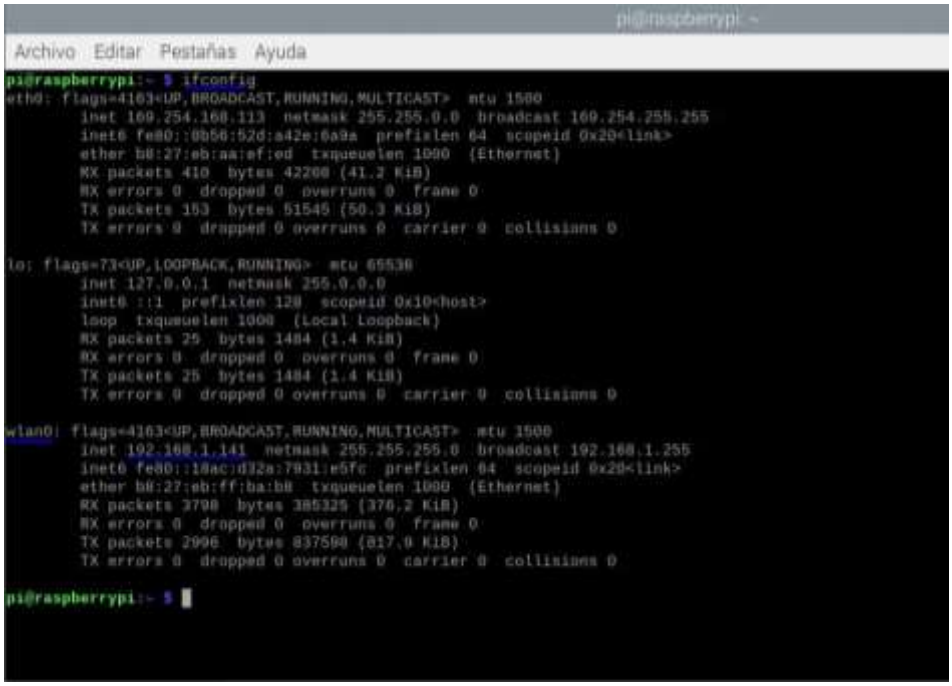

Figura 4-9. Obtención de la dirección IP

4.2. Sistema manual

El sistema manual consiste en un control del iRobot Create 2 mediante nuestro ordenador a través de la conexión cliente/servidor. Por esta conexión enviamos una serie de ordenes con las que conseguiremos manejarlo. La implementación de estas órdenes se ha realizado en el servidor, que se ejecuta en la Raspberry Pi, configurando de manera predeterminada los comandos de control del iRobot Create 2 de la librería pycreate2.

Las ordenes implementadas son las siguientes:

- avanzar. Al enviar esta orden conseguimos que el iRobot Create 2 se mueva hacia adelante.
- retroceder. Con esta orden conseguimos la acción inversa al anterior, es decir, el iRobot Create 2 se mueve hacia atrás.
- parar. Esta orden detiene al iRobot Create 2 en la posición actual.
- derecha. Orden desarrollada para que el iRobot Create 2 realice un giro en sentido horario sobre su posición.
- izquierda. Orden inversa a la anterior mediante la que se realiza un giro en sentido anti horario sobre su posición.
- angulo. Orden que pide al iRobot Create 2 que devuelva el ángulo total girado en grados por el robot desde la última vez que se le pidió.
- distancia. Orden que pide al iRobot Create 2 que le proporcione la distancia total recorrida en milímetros desde la última vez que se le pidió.

A continuación, comprobaremos el funcionamiento y observaremos la respuesta de cada una de estas órdenes. La comprobación de las ordenes avanzar y retroceder la haremos conjuntamente con la orden distancia y parar ya que esta es la única manera de observar que la orden funciona, lo mismo pasa con las ordenes derecha e izquierda que tienen que ser comprobadas conjuntamente con la orden angulo y parar.

En la Figura 4-10 mostramos una imagen de la ejecución de la orden avanzar, para enseñar que realmente el comando funciona realizamos como comentamos anteriormente una ejecución conjunta con las ordenes parar para detener la roomba y poder tomar una medida de la distancia recorrida con la orden distancia comprobando que el valor mostrado es positivo sinónimo de un movimiento hacia adelante.

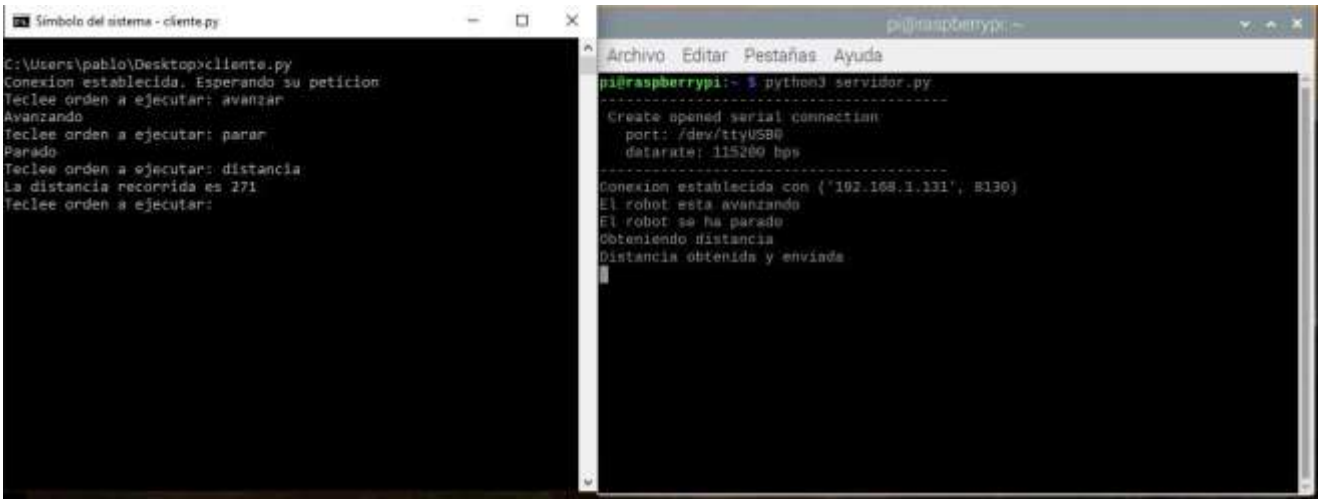

Figura 4-10. Funcionamiento orden avanzar

En la siguiente imagen, Figura 4-11., comprobamos el funcionamiento de la orden retroceder viendo que obtenemos una distancia negativa lo cual significa que el movimiento ha sido contrario al anterior como queríamos que ocurriese.

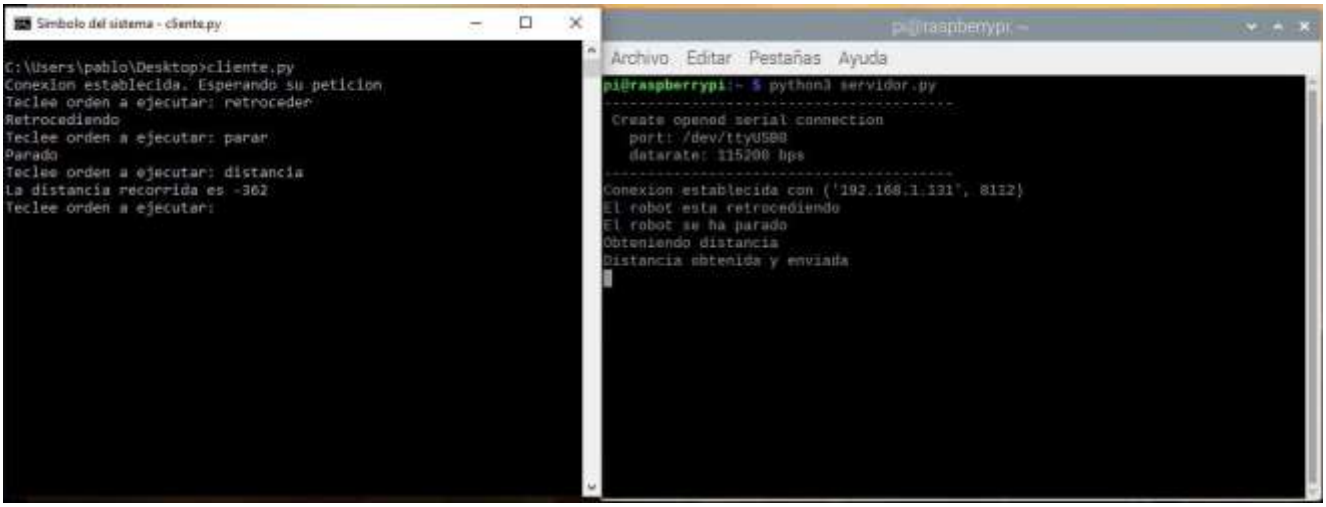

Figura 4-11. Funcionamiento orden retroceder

Tras lo anterior, comprobamos los giros. En concreto, el giro a la derecha para poder mostrar que se ha realizado un giro a la derecha tenemos que tomar la medida del angulo, teniendo en cuenta que el sensor angulo del iRobot Create 2 devuelve valores positivos para el sentido contrario a las agujas del reloj y valores negativos en el caso contrario. Como podemos comprobar el giro a la derecha es un giro en sentido de las agujas del reloj por lo que obtenemos un valor negativo en este caso.

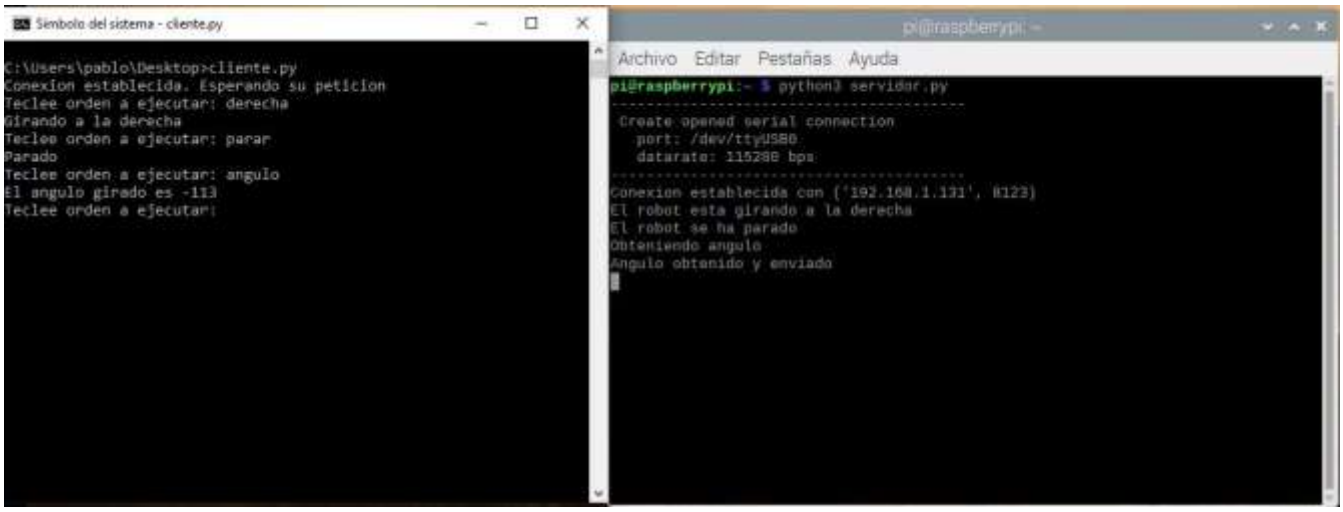

Figura 4-12. Funcionamiento orden derecha

Por último, comprobamos que el giro a la izquierda funciona correctamente. Con su ejecución tenemos que obtener un valor positivo ya que el giro se produce en sentido contrario a las agujas del reloj. Apreciamos en la ejecución su correcto funcionamiento al obtener un valor positivo.

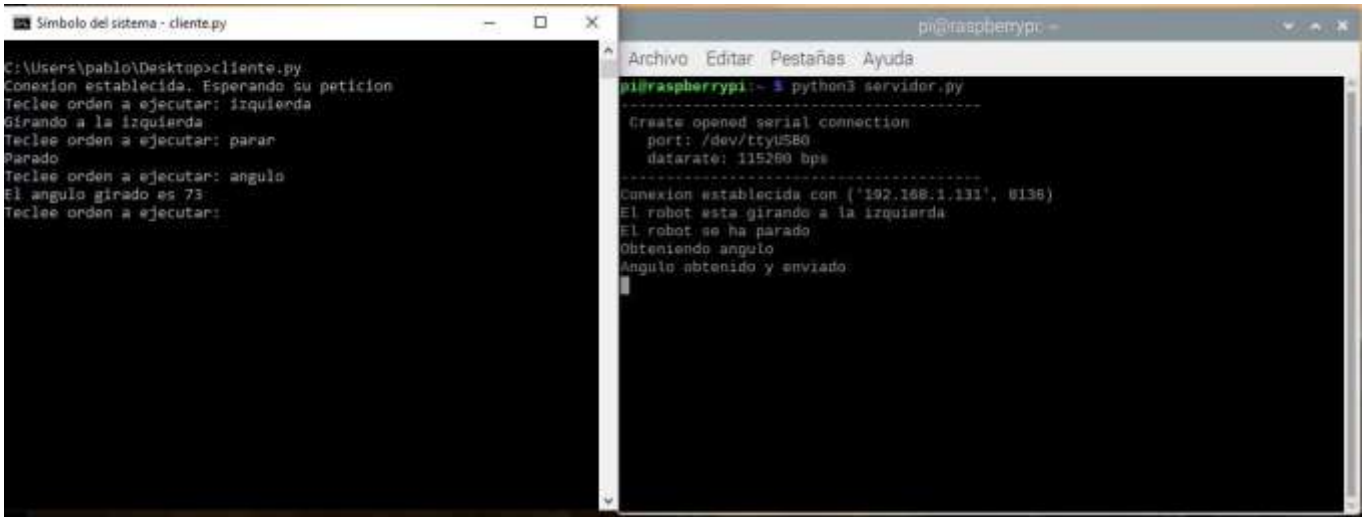

Figura 4-13. Funcionamiento orden izquierda

4.3. Misión

La misión desarrollada con este robot consiste en el seguimiento de una trayectoria, en la que se tiene una serie de puntos claves de medición en los que el iRobot Create 2 debe tomar unas medidas con sus sensores. Concretamente tomamos medidas de la suciedad en esos puntos haciendo uso del sensor dirty.

La trayectoria desarrollada en esta misión es un cuadrado en el que los puntos claves son sus cuatro vértices, esta trayectoria se ha elegido dado que los espacios en los que se ha realizado el proyecto y las mediciones se regian por este tipo de figura.

La forma de organizar la misión se puede decir que ha sido el desarrollo de tres bloques diferenciados:

- Avanzando. Bloque con el cuál establecemos las velocidades de las ruedas, en este caso a 200 cada una de ellas, el tiempo de sleep que es el tiempo durante el cual el iRobot Create 2 continua ejecuando el ultimo comando. Este bloque lo que representa es la arista del cuadrado.
- Girando. Bloque con el cual establecemos un giro de 90 grados, esto se ha conseguido pasándole como velocidad a cada rueda el valor de -100 para la rueda derecha y de 100 para la rueda izquierda, el parámetro clave de este bloque es el tiempo que se ejecuta la anterior opción el cual se ha establecido en 1.7 segundos que tras varias pruebas ha sido el tiempo que ha devuelto un valor de 90 grados. Este bloque representa los vértices del cuadrado
- Midiendo. Bloque con el cuál conseguimos el valor del sensor que nos interesa. Con este bloque establecemos la velocidad de las ruedas a 0 para que se mantenga parado, el tiempo no es relevante en este bloque. En el caso de que se ejecute este bloque hemos creado una canción para que, si se da que el punto es un punto limpio, habiendo considerado punto limpio para obtener una medida precisa como aquel en el que el sensor devuelva un valor menor que 30 siendo su máximo 255, se reproduzca dicha canción. Esta canción es la que marca el tiempo de ejecución de este bloque que si no es un punto limpio no reproduce nada y alerta con un mensaje en la pantalla del servidor.

Tras la ejecución de estos bloques habremos recorrido el trayecto planteado y con ello se le envía un mensaje al cliente comunicando que la misión se ha completado con éxito.

La única forma de mostrar el correcto funcionamiento de esta misión es mostrando una imagen de lo obtenido tanto en la parte del servidor como en la parte del cliente al ejecutar la misión. Por ello, en la Figura 4-14 se muestra lo conseguido con el programa desarrollado al seguir una trayectoria cuadrada en la cual se ha configurado uno de sus vértices, concremente el segundo, con una suciedad adecuada para que el sensor la detecte.

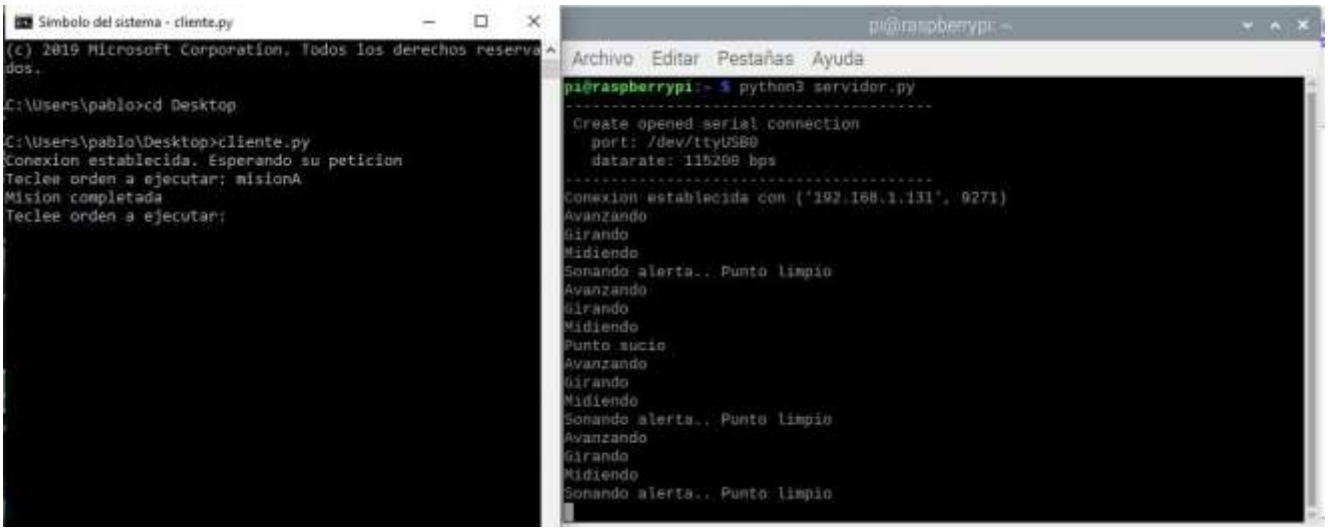

Figura 4-14. Funcionamiento de la misión

5 CONCLUSIÓN Y FUTUROS TRABAJOS

5.1. Conclusión

Se ha desarrollado un proyecto basado en un sistema de recolección de datos con el que hemos aprendido el gran abanico de posibilidades que abarca estos tipos de sistemas al realizar el estudio del estado del arte. Además, hemos conocido en profundidad el manejo y funcionamiento del iRobot Create 2, dispositivo tomado como base del proyecto.

La preparación de la Raspberry Pi para la realización del proyecto y el establecimiento de las comunicaciones de los dispositivos que forman el sistema, nos ha ayudado entender mejor esta placa y darnos cuenta de todas las posibilidades que ofrece trabajar con este tipo de dispositivos.

El desarrollo del cliente/servidor mediante el lenguaje de programación Python nos ha brindado la oportunidad de poner en práctica los conocimientos adquiridos durante nuestra formación, dejándonos constancia de que se trata de un lenguaje muy manejable y que se puede aplicar en muchos ámbitos.

Finalmente, con el cumplimiento del objetivo principal hemos conseguido darle vida a una primera idea que en un principio se nos presentó como algo difícil de alcanzar, pero con el tiempo fue tomando forma, a pesar de las complicaciones, hasta alcanzar este punto final. Con esto nos hemos involucrado en un proyecto que posee características similares a los proyectos a los que esperamos que se nos presenten en un futuro aportándonos un ápice de experiencia.

5.2. Futuros trabajos

Este proyecto sirve como punto de partida de proyectos más complejos o incluso se podría emplear siguiendo lo desarrollado e incluyendo algunas mejoras como las que proponemos a continuación.

Como primera futura mejora planteamos perfeccionar el cliente/servidor que ejecuta nuestro sistema de adquisición de datos haciéndolo más robusto y añadiéndole más opciones a las que se le puedan sacar partido ya que no solo se podría usar el cliente/servidor para el manejo del iRobot Create 2 sino que también podría emplearse para ejecutar programas en la Raspberry Pi que realice un tratamiento de los datos obtenidos antes de ser enviados al cliente.

La segunda futura mejora que se nos ocurre es introducir una interfaz gráfica que permita al usuario del sistema entenderlo y manejarlo más fácilmente, ya que se tiene la idea de que el sistema pueda ser empleado tanto por usuarios que posean los conocimientos necesarios como por aquellos que carezcan de estos y le sirva para poder adquirirlos.

Finalmente, nuestro proyecto se ha centrado en el desarrollo de un sistema de adquisición de datos, el cual pueda ser programado para realizar movimientos determinados o controlar dichos movimientos. Aprovechando que el sistema es capaz de seguir trayectorias programadas la última mejora propuesta es, además de leer los sensores del iRobot Create 2, hacer uso de la Raspberry Pi para obtener datos. Por ejemplo, programar una trayectoria concreta y a medida que se recorre la Raspberry Pi va accediendo a los puntos de accesos disponibles descargando y subiendo paquetes a la red con ello conseguiríamos analizar qué puntos de accesos tienen mayor o menos demanda y de esta manera configurar la red para que sea lo más óptima posible. Otro posible ejemplo podría ser programar una determinada trayectoria para obtener los niveles de radiación de una antena en determinados puntos de un espacio.

APÉNDICE

A.1. Errores y complicaciones

En esta sección comentaremos algunos fallos y complicaciones que nos hemos ido encontrando a lo largo del proyecto que nos han impedido tomar algunos caminos alternativos que teníamos en mente pero que no han sido posible desarrollarse.

Todo comenzó con una primera idea de llevar a cabo un seguidor de línea con algunas funciones extras, pero nos encontramos con el problema de que los sensores infrarrojos que usaríamos presentaban un inconveniente. Este sensor infrarrojo era usado para detectar desniveles y bloqueaba el robot para evitar caídas por escaleras u otro desnivel que produzcan defectos en él, la idea fue usar este sensor para que recorriese una línea negra que representaría este desnivel pero los materiales empleados para ello no conseguían que el sensor se activara, decidimos obtener la señal que regia ese sensor para establecer un escalon de decisión pero aun asi no se encontraba una diferencia que nos permitiese definir una región de decisión.

Una consideración a tener en cuenta es que el tipo de suelo sobre el que se desplace el robot hace que el comportamiento de los sensores cambie, encontrándose en numerosas ocasiones con valores que inducen al error. Los sensores en estos casos suelen devolver valores aleatorios que se puede comprobar tomando muestras sucesivas, el robot esta configurado para que tras la toma de una muestra su valor se reincie a 0, para que la siguiente muestra no se vea influenciada por la anterior, por lo que tras la toma de una segunda muestra por ejemplo del sensor distancia sin realizar ningun movimiento el valor esperado seria 0 y nos encontrábamos con valores que no aportaban ningun información y que cambiaban ejecución tras ejecución por tanto quedaba descartado que se tratase de un offset introducido por el robot.

Por último, comentaremos un fallo presente en el programa actual y que hemos observado que sigue un patrón, este fallo tiene que ver con la obtención de los sensores distancia y angulo usados en el sistema manual, los cuales, tras varias ejecuciones, se ha observado que este valor ronda las 6/7 veces, los sensores devuelven el valor de uno de sus extremos de su rango, la única solución encontrada para este problema es el reinicio del sistema.

Estos fallos y complicaciones mas destacadas han hecho que el proyecto tome la forma actual y se hayan descartado una serie de posibles alternativas muy interesantes a llevar a cabo con este robot, no se ha encontrado alguna referencia a estos fallos en concreto por lo que pueden tratarse de fallos concretos de este robot.

A.2. Código

Código A-1. Código cliente.py

```
#Socket
import socket
import sys

#Host y puerto a conectar
host = '192.168.1.139'
port = 65436

#Creamos el socket
clien_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
clien_sock.connect((host, port))

#Vemos si la conexion se ha establecido
conexion = clien_sock.recv(1024)
print(conexion.decode())

#Interactuamos con el servidor
try:
      while True:

            orden = input('Teclee orden a ejecutar: ')
            clien_sock.send(orden.encode())

            respuesta = clien_sock.recv(1024)
            respuesta = respuesta.decode()

            if respuesta == 'Comando incorrecto':
                  print(respuesta)
            elif respuesta == 'Cerrando conexion..':
                  break
            else:
                  print(respuesta)

#Cerramos conexion
finally:
      clien_sock.close()
      print('Conexion cerrada')
```

Código A-2. Código servidor.py

```
#Roomba
from pycreate2 import Create2
import time

#Socket
import socket
import sys

#Host y puerto a conectar
host = '192.168.1.139'
port = 65436

#Creacion conexion con la roomba
portRoomba = "/dev/ttyUSB0"
bot = Create2(portRoomba)

#Arranque de la roomba y activacion del modo seguro
bot.start()
bot.safe()

#Creacion del socket y espera de la conexion
serv_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serv_sock.bind((host, port))
serv_sock.listen()

#Comprobacion de conexion con el cliente
clien_sock, addr = serv_sock.accept()
print('Conexion establecida con {}'.format(addr))
clien_sock.send(b'Conexion establecida. Esperando su peticion')

#Interaccion con el cliente
while True:
      #Lectura de dato y decodificacion
    dato = clien_sock.recv(1024)
    dato = dato.decode()

    #Identificacion del comando y configuracion de parametros
      #Avance de la roomba
    if dato == 'avanzar':
        bot.drive_direct(100,100)
        clien_sock.send(b'Avanzando')
        print('El robot esta avanzando')

      #Giro derecha
    elif dato == 'derecha':
        bot.drive_direct(-100,100)
        clien_sock.send(b'Girando a la derecha')
        print('El robot esta girando a la derecha')

      #Giro izquierda
    elif dato == 'izquierda':
        bot.drive_direct(100,-100)
        clien_sock.send(b'Girando a la izquierda')
        print('El robot esta girando a la izquierda')

      #Retroceso de la roomba
```

```
    elif dato == 'retroceder':
        bot.drive_direct(-100,-100)
        clien_sock.send(b'Retrocediendo')
        print('El robot esta retrocediendo')

      #Parar roomba
    elif dato == 'parar':
        bot.drive_stop()
        clien_sock.send(b'Parado')
        print('El robot se ha parado')

      #Medicion de la distancia
    elif dato == 'distancia':
        sensores = bot.get_sensors()
        dist = str(sensores.distance)
        mensaje = 'La distancia recorrida es '+dist
        print('Obteniendo distancia')
        clien_sock.send(mensaje.encode())
        print('Distancia obtenida y enviada')

      #Medicion del angulo
    elif dato == 'angulo':
        sensores = bot.get_sensors()
        ang = str(sensores.angle)
        mensaje = 'El angulo girado es '+ang
        print('Obteniendo angulo')
        clien_sock.send(mensaje.encode())
        print('Angulo obtenido y enviado')

      #Ejecucion de la mision
    elif dato == 'misionA':
           #Configuracion de la cancion y tipo
        song = [67, 45, 67, 45, 67, 45, 67, 45, 67, 45, 67, 45, 67, 45,
67, 45]
        song_num = 0

           #Establecimiento de la trayectoria
        path = [
            [200, 200, 5, 'Avanzando'],
            [-100, 100, 1.7, 'Girando'],
            [0, 0, 1, 'Midiendo'],
            [200, 200, 5, 'Avanzando'],
            [-100, 100, 1.7, 'Girando'],
            [0, 0, 1, 'Midiendo'],
            [200, 200, 5, 'Avanzando'],
            [-100, 100, 1.7, 'Girando'],
            [0, 0, 1, 'Midiendo'],
            [200, 200, 5, 'Avanzando'],
            [-100, 100, 1.7, 'Girando'],
            [0, 0, 1, 'Midiendo']
            ]

           #Seguimiento de la trayectoria
        for lft, rgt, t, s in path:
                 #Comprobacion de la opcion de medicion
            if s == 'Midiendo':
                bot.drive_direct(lft,rgt)
                sensors=bot.get_sensors()
                print(s)
```

```
                    #Comprobacion del nivel de suciedad
                if sensors.dirt_detect < 30:
                    print('Sonando alerta.. Punto limpio')
                            #Creacion de la cancion y ejecucion
                    bot.createSong(song_num, song)
                    duracion = bot.playSong(song_num)
                    time.sleep(duracion)
                else:
                    print('Punto sucio')
            else:
                print(s)
                bot.drive_direct(lft,rgt)
                time.sleep(t)
            #Recorrido de la trayectoria completado
        mensaje = 'Mision completada'
        clien_sock.send(mensaje.encode())

      #Cierre de la sesion
    elif dato == 'cerrar':
        clien_sock.send(b'Cerrando conexion..')
        clien_sock.close()
        print('Conexion cerrada')
        break

    #Si no se da ninguna de las acciones el comando introducido no es
correcto
    else:
        clien_sock.send(b'Comando incorrecto')

#Cerramos conexion
serv_sock.close()
```

REFERENCIAS

[1] Maldonado Templos, Alejandro. La importancia de la robótica. *Milenio.* 10 Noviembre 2020. [Consulta 2 Nov 2020]. Disponible en: https://www.milenio.com/opinion/varios-autores/universidad-politecnica-de-tulancingo/la-importancia-de-la-robotica

[2] *Nano Jetbot kit - an educational ai robot based on NVIDIA Jetson nano.* Silicon HighWay LTD, 2020. (En linea). Disponible en: https://www.siliconhighwaydirect.co.uk/product-p/jetbot-kit.htm [Consulta 4 Nov 2020]

[3] *Nvidia lanza Jetson Nano, un Mini PC de 109 € para fans de la robótica y la inteligencia artificial.* Xakata, 2019. (En linea). Disponible en: https://www.xataka.com/inteligencia-artificial/nvidia-lanza-jetson-nano-mini-pc-109-eur-para-fans-robotica-inteligencia-artificial [Consulta 5 Nov 2020]

[4] *Especificaciones Robomaster S1.* DJI, 2020. (En línea). Disponible en: https://www.dji.com/es/robomaster-s1/specs [Consulta 4 Nov 2020]

[5] *Robomaster S1, análisis: el primer robot educativo de DJI es un arma para aprender a programar.* Xakata, 2020. (En linea). Disponible en: https://www.xataka.com/analisis/dji-robomaster-s1analisis-caracteristicas-precio-especificaciones [Consulta 6 Nov 2020]

[6] *Hands On With DJI's New RoboMaster S1 Battling Robot.* Extremetech, 2019. (En linea). Disponible en: https://www.extremetech.com/extreme/293394-hands-on-with-djis-new-robomaster-s1-battling-robot [Consulta 6 Nov 2020]

[7] *Zowi, el robot infantil de BQ.* Codigo21 Tecnologías creativas, (sin fecha). (En línea). Disponible en: https://codigo21.educacion.navarra.es/autoaprendizaje/zowi-el-robot-infantil-de-bq/ [Consulta 4 Nov 2020]

[8] Penalva, Javier. Probamos Zowi, un robot con cerebro Arduino que puede dar más de lo que aparenta. *Xakata.* 2 Noviembre 2019 [Consulta 5 Nov 2020]. Disponible en: https://www.xataka.com/analisis/probamos-zowi-un-robot-con-cerebro-arduino-que-puede-dar-mas-de-lo-que-aparenta

[9] *Sphero lanza su nuevo robot compatible con Raspberry Pi y Arduino, y lo financia en un solo día mediante crowdfunding.* Xakata, 2019. (En linea). Disponible en: https://www.xataka.com/inteligencia-artificial/sphero-logra-financiar-solo-dia-mediante-crowdfunding-su-nuevo-robot-compatible-periferico [Consulta 4 Nov 2020]

[10] *Sphero RVR es el robot con el que los niños aprenderán programación.* RedBull, 2019. (En linea). Disponible en: https://www.redbull.com/cl-es/sphero-rvr-robot-infantil-programacion [Consulta 5 Nov 2020]

[11] *Sphero RVR – The go anywhere, do anything programmable robot.* Kickstarter, 2019. (En linea). Disponible en: https://www.kickstarter.com/projects/sphero/sphero-rvr-the-go-anywhere-do-anything-programmabl [Consulta 5 Nov 2020]

[12] *Sphero RVR.* Sphero, (sin fecha). (En linea). Disponible en: https://sphero.com/products/rvr?_pos=1&_sid=8b609af18&_ss=r [Consulta 5 Nov 2020]

[13] *PiBot-B.* RobotShop Community, 2013. (En linea). Disponible en: https://www.robotshop.com/community/robots/show/pibot-b [Consulta 4 Nov 2020]

[14] *PiBot-B: mobile robot with a Raspberry Pi.* Pololu Robotics and Electronics, 2013. (En linea). Disponible en: https://www.pololu.com/blog/233/pibot-b-mobile-robot-with-a-raspberry-pi [Consulta 4 Nov 2020]

[15] *PiBot-B mobiler Roboter mit Raspberry Pi.* Thomas Schoch, 2014. (En linea). Disponible en: http://www.retas.de/thomas/raspberrypi/pibot-b/ [Consulta 5 Nov 2020]

[16] *PiBot-A mobiler Roboter mit Raspberry Pi A+.* Thomas Schoch, 2015. (En linea). Disponible en: http://www.retas.de/thomas/raspberrypi/pibot-b/ [Consulta 5 Nov 2020]

[17] *Adeept DarkPaw Bionic Quadruped Spider Robot Kit for Raspberry Pi STEM Crawling Robot.* Robotics Technology for Everyone, (sin fecha). (En linea). Disponible en: https://ozrobotics.com/shop/adeept-darkpaw-bionic-quadruped-spider-robot-kit-for-raspberry-pi-stem-crawling-robot/ [Consulta 7 Nov 2020]

[18] *DarkPaw Spider Robot Kit for RPi.* Adeept, (sin fecha). (En linea). Disponible en: https://www.adeept.com/learn/detail-38.html [Consulta 7 Nov 2020]

[19] *Adeept DarkPaw Bionic Quadruped Spider Robot Kit for Raspberry Pi 4/3 Model B+/B/2B, STEM Crawling Robot, OpenCV Tracking, Self-stabilizing.* Adeept, (sin fecha). (En linea). Disponible en: https://www.adeept.com/darkpaw_p0125.html [Consulta 7 Nov 2020]

[20] Molaro, Jamie. PARSLEE. *Data Arcana* [blog]. Sin fecha. [Consulta 7 Nov 2020]. Disponible en: https://dataarcanacom.wordpress.com/portfolio/parslee/

[21] Bate, Alex. NASA, Raspberry Pi and a mini rover. *Raspberry Pi Blog* [blog]. 10 Julio 2019 [Consulta 7 Nov 2020]. Disponible en: https://www.raspberrypi.org/blog/nasa-raspberry-pi-and-a-mini-rover/

[22] Santamaria, Pedro. iRobot Create 2, la "roomba" para makers y amantes de la robótica. *Xakata smart home.* 12 Diciembre 2014 [Consulta 10 Nov 2020]. Disponible en: https://www.xatakahome.com/electrodomesticos-innovadores/irobot-create-2-la-roomba-para-makers-y-amantes-de-la-robotica

[23] Ecoaula.es. iRobot® presenta el robot programable Create® 2 para inspirar a la próxima generación de líderes en robótica. *El Economista [periódico].* 10 Abril 2014 [Consulta 10 Nov 2020]. Disponible en: https://www.eleconomista.es/ecoaula/noticias/9060673/04/18/iRobot-presenta-el-robot-programable-Create-2-para-inspirar-a-la-proxima-generacion-de-lideres-en-robotica.html

[24] *iRobot® Create® 2 Open Interface (OI) Specification based on the iRobot® Roomba® 600* (En linea). [Consulta 2 Dic 2020] Disponible en: https://www.irobotweb.com/~/media/MainSite/PDFs/About/STEM/Create/iRobot_Roomba_600_Open_Interface_Spec.pdf

[25] Anónimo. RASPBERRY PI. *Historia de la informática* [blog]. 18 Diciembre 2013. [Consulta 12 Nov 2020]. Disponible en: https://histinf.blogs.upv.es/2013/12/18/raspberry-pi/

[26] *Raspberry Pi Pinout Diagram | Circuit Notes.* Jameco Electronics, (sin fecha). (En linea). Disponible en: https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html [Consulta 12 Nov 2020]

[27] *Datasheet RASPBERRY PI 3 MODEL B.* Element14 (En linea). [Consulta 12 Nov 2020]. Disponible en: https://www.terraelectronica.ru/pdf/show?pdf_file=%252Fds%252Fpdf%252FT%252FTechicRP3.pdf

[28] *User Manual Portable Charger Power Bank Pilot X720000 mAh.* POWERADD, (sin fecha). (En linea). Disponible en:

https://www.ipoweradd.com/Private/Files/20180829162717605.pdf [Consulta 12 Nov 2020]

[29] Alvarez, Miguel Angel. Qué es python. *desarrolloweb.com.* 19 Noviembre 2003. [Consulta 3 Dic 2020]. Disponible en: https://desarrolloweb.com/articulos/1325.php

[30] *Python.* Raspberry Pi, (sin fecha). (En línea). Disponible en: https://www.raspberrypi.org/documentation/usage/python/ [Consulta 3 Dic 2020]

[31] *Installing Library package in Raspberry Pi-Chapter 2.* Pantech, 2018. (En linea). Disponible en: https://www.pantechsolutions.net/blog/installing-library-packages-in-raspberry-pi/ [Consulta 19 Nov 2020]

[32] *pycreate2 0.8.0.* Kevin Walchko, 2017. (En linea). Disponible en: https://pypi.org/project/pycreate2/ [Consultado 3 Dic 2020]

[33] *Socket Programming in Python (Guide).* Nathan Jennings, (sin fecha). (En linea). Disponible en: https://realpython.com/python-sockets/ [Consultado 25 Oct 2020]

[34] Anonimo. PROGRAMACIÓN DE REDES EN PYTHON: SOCKETS. *Unipython* [blog]. 2019.[Consulta 23 Nov 2020]. Disponible en: https://unipython.com/programacion-de-redes-en-python-sockets/

[35] Schiaffarino, Andrés. Modelo cliente servidor. *infranetworking* [blog]. 12 Marzo 2019. [Consulta 23 Nov 2020]. Disponible en: https://blog.infranetworking.com/modelo-cliente-servidor/

[36] *Acceso remoto a un Raspberry Pi con SSH: cómo habilitarlo.* Digital Guide IONOS, 2019. (En linea). Disponible en: https://www.ionos.es/digitalguide/servidores/configuracion/activar-ssh-en-raspberry-pi/ [Consulta 3 Nov 2020]

[37] *IP Adress.* Raspberry Pi, (sin fecha). (En línea). Disponible en: https://www.raspberrypi.org/documentation/remote-access/ip-address.md [Consulta 3 Nov 2020]

GLOSARIO

CPU: Central Processing Unit
CSI: Camera Serial Interface

DSI: Display Serial Interface

GPIO: General Purpose Input/Output
GPU: Graphics Processing Unit
GUI: Graphic User Interface

HDMI: High Definition Multimedia Interface

I2C: Inter-Integrated Circuit
I2S: Integrated Interchip Sound

LED: Light Emitting Diode

OI: Open Interface

PWM: Pulse Width Modulation

RGB: Red Green Blue

SBC: Simple Board Computer
SO: Sistema Operativo
SPI: Serial Peripheral Interface
SSH: Secure SHell

TCP: Transmission Control Protocol

UART: Universal Asynchronous Receiver-Transmitter

VNC: Virtual Network Connection

WLAN: Wireless Local Area Network