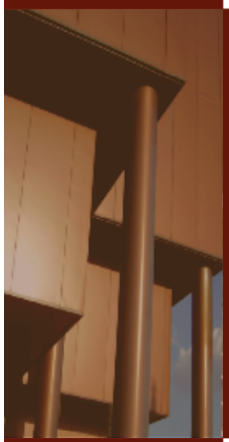
Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación de control de movimientos y recopilación de datos de contexto para un robot móvil.

Autor: Juan Cruz Muñoz
Tutor: Daniel Gutiérrez Reina



Dep. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

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

Aplicación de control de movimientos y recopilación de datos de contexto para un robot móvil.

Autor:
Juan Cruz Muñoz

Tutor:
Daniel Gutiérrez
Profesor titular

Departamento de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Carrera: Aplicación de control de movimientos y recopilación de datos de contexto para un robot móvil.

Autor: Juan Cruz Muñoz

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, 2017

El Secretario del Tribunal

A mi familia y amigos

A mis profesores

Resumen

En la actualidad la electrónica está en completo auge. Las posibilidades que ofrece el correcto uso de ciertos microcontroladores y sus módulos abre las puertas al desarrollo de cualquier sistema que se pueda imaginar.

A modo de puesta en práctica de los conocimientos adquiridos a lo largo de la carrera y con la intención de reunirlos todos bajo un mismo proyecto, se propone llevar a cabo un robot de reconocimiento de entorno controlado remotamente. Aunque este proyecto tenga un fin méramente educativo, no deja de ser un desarrollo completo que no queda lejos de cualquier robot comercial. Consistirá en un vehículo robot basado Raspberry Pi y Python mediante el cual se tomarán distintos datos del entorno a través varios periféricos.

A lo largo del proyecto se estudiarán los aspectos teóricos necesarios, se diseñará y se realizará el montaje del sistema, se programará el código necesario para el microcontrolador Raspberry Pi y se realizarán una serie de tests que demuestren su correcto funcionamiento.

Abstract

At present, electronics are booming. The possibilities offered by the correct use of certain microcontrollers and the modules that work with it opens the doors to the development of any system that can be imagined.

As a way of getting to practice the knowledge acquired throughout the career and with the intention of bringing them together under the same project, it is proposed to develop an environment data recognition robot drived using a remote control. Although this project has a very educational purpose, it is still a complete development that is not far away of being a commercial robot. The proyect will be about a robot based on Raspberry Pi and Python through which multiple data will be taken from the enviroment using some peripherals.

Throughout the project, the necessary theoretical aspects will be studied, the assembly of the system will be designed and carried out, the code required for the Raspberry Pi microcontroller will be programmed and a series of tests will be carried out to demonstrate its correct operation.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 OBJETIVOS Y MOTIVACIÓN DEL PROYECTO

Este es un proyecto cuya finalidad es conseguir un prototipo de vehículo robot de recopilación de datos de contexto. Es decir, un robot de reconocimiento del entorno.

Los vehículos robots nos permiten la adquisición de datos en entornos de imposible o difícil acceso humano, bien por razones de lejanía, de coste, de peligrosidad o incluso la suma de ellos. Aunque este proyecto tenga un fin méramente educativo, no cabe duda de que con los medios necesarios se podría extrapolar la idea de este proyecto y convertirlo en un proyecto real para ejecutar alguna de las numerosas aplicaciones posibles, desde apoyo de misiones de exploración planetarias, investigación de materiales peligrosos o en zonas peligrosas (radiación, temperaturas extremas..)

Hablando del ámbito educativo, todo sistema de telecomunicaciones se divide en parte física, o hardware, y parte software que se encargará de controlar dicho hardware. A lo largo de la carrera se aprenden varios lenguajes de programación, cada uno de ellos orientado a distintos fines o aplicaciones y estamos en constante contacto con distintos tipos de hardware. En cursos superiores, sobre todo en la mención de Sistemas Electónicos, trabajamos con distintos microprocesadores y microcontroladores programables y cada uno de ellos hará uso de un lenguaje de programación distinto.

Con la intención de encauzar estas ramas de conocimiento en la electrónica surgió la idea de desarrollar un vehiculo robot basado en el tan conocido ordenador de placa Raspberry Pi, que nació con el objetivo de estimular la enseñanza de la ciencias de computación, y para poner en práctica el conocimiento de programación software haremos uso del lenguaje de programación Python, en auge actualmente, para la programación de la Rasbperry Pi y el uso de distintos sensores.

Otro aspecto fundamental del proyecto será el desarrollo de la aplicación para el Sistema Operativo Android que actuará como control remoto del robot. Para este fin usaremos App Inventor, un entorno de desarrollo de aplicaciones creado conjuntamente por el MIT (Massachusetts Institute of Technology) y Google con el que pretenden fomentar el aprendizaje de programación de dichas aplicaciones.

Sin duda alguna, este proyecto engloba varios ámbitos que todo ingeniero de telecomunicaciones debe dominar, desde diseño hardware y software hasta comunicaciones inalámbricas y todo ello haciendo uso de entornos y herramientas pensadas para la educación y , ésta quizá sea la principal motivación del proyecto.

El proyecto abordará los siguientes aspectos que podríamos calificar de objetivos.

- El montaje del vehículo robot comercial y motores.
- El diseño del hardware para el correcto conexionado de la placa a los motores, controladores y sensores. Esto podría incluir el diseño de algún circuito extra o el uso de una placa de pruebas para poder alojar correctamente todo lo nombrado.
- El diseño de una aplicación móvil para el SO Android que hará la función de mando del vehículo
- El diseño software completo usando el lenguaje Python sobre Raspberry Pi para que sea teledirigido desde el móvil Android y para que obtenga datos del entorno mediante el uso de sensores de presión y temperatura, GPS y cámara.
- El diseño de una base de datos MySQL en la que se almacenarán los datos recogidos por el robot.

Además de la revisión y documentación del estado del arte de todos estos aspectos y las explicaciones que sean necesarias para la correcta interpretación de éste.

2 INTRODUCCIÓN A LOS VEHICULOS ROBOTS

"Agente activo artificial cuyo ambiente es el mundo físico"

Russel y Norvig

2.1 Importancia de la robótica

Con el fin de eliminar la necesidad de un operario en trabajos no solo complejos o repetitivos, sino peligrosos, que añadan un riesgo para la integridad física de la persona o incluso el trabajo en entornos donde no es posible la vida humana, surge la necesidad de sustituir a la persona por la máquina. A lo largo de la historia, el hombre siempre ha tendido a utilizar todos los recursos disponibles de su alrededor, incluyendo el propio entorno. Explorar un entorno no conocido, o simplemente navegar a través de él una vez se ha realizado este reconocimiento previo. Este afán por conocer y estudiar el medio que nos rodea es una de las causas fundamentales de la necesidad de la robótica y en la que enfocaremos este proyecto.

La complejidad del análisis del entorno cada vez es mayor debido a las exigencias y el nivel de detalle exigido en las medidas de los datos de contexto, y esto hace que exista una creciente complejidad en las tareas a realizar. Paralelamente, las herramientas en las que se apoya el hombre para llevar a cabo estas tareas también se han vuelto más sofisticadas y eficientes con el avance, cada vez más acelerado, de la tecnología. El resultado es el desarrollo de herramientas complejas capaces de realizar trabajos de forma más precisa, rápida y eficiente que una persona. Es decir, los robots.

Existe un amplio desconocimiento de lo que significa robótica y de sus alcances. Hay muchos tipos de robots y son muchos los campos de aplicación de éstos, desde la oceanografía, la industria, la medicina, el hogar y las armas. La robótica es una ciencia mixta que abarca muchos campos del saber tales como mecánica, electrónica, informática, servomecanismos, ergonomía e inteligencia artificial, entre otras.

En el contexto de este proyecto tenemos que destacar la importancia de la electrónica como parte de la robótica. La electrónica es la ciencia aplicada fundamental de la tecnología moderna, todo equipo que necesite un control rápido y efectivo hace uso de la tecnología electrónica. Con los componentes electrónicos se elaboran los circuitos de control y comunicación de los robots, a través de la circuitería electrónica se realizan también las funciones de detección de las señales que se emplean para manipular y controlar los servomecanismos y los sensores de los robots así como la comunicación de datos e instrucciones de control y operación.

2.2 Historia de los robots móviles

El desarrollo de los robots o vehículos móviles responde a la necesidad de extender el propio campo de aplicación de la robótica, el cual estaba restringido en sus inicios al alcance de la estructura mecánica anclada en uno de sus extremos, lo que podríamos entender por un brazo robot. De esta forma también se consigue

abordar otra de las cuestiones de la robótica, que es incrementar la autonomía del robot, limitando todo lo posible la intervención humana.

Los robots móviles tienen como antecesores los dispositivos electromecánicos, creados desde los años 30's con el objetivo de desarrollar funciones inteligentes tales como por ejemplo solucionar laberintos. En los años 60's comienzan a desarrollarse vehículos autónomos en la industria, siendo estos guiados por cables bajo el suelo o mediante sensores ópticos para seguir líneas trazadas en las plantas. Posteriormente, en los años 70's, se vuelve a trabajar en robots móviles de mayor autonomía, aunque el desarrollo tecnológico aun no era suficiente para lograr la navegación autónoma de una forma eficiente. En los años 80's, el incremento sobrecogedor de la capacidad computacional, acompañado del desarrollo de nuevos sensores, mecanismos y sistemas de control, permitieron aumentar esta deseada autonomía.

Algunos robots que han destacado a lo largo de la historia son:

- 1890, Tesla: vehículos radio controlados
- 1940s, Wiener: dispositivo antiaéreo
- 1950: Tortuga electrónica Walter que reaccionaba ante la presencia de obstáculos, subía pendientes y se autoredireccionaba a la estación de carga cuando era necesario
- 1966, S.R.I: Shakey (primer robot móvil con IA)
- 1960s, G.E., Quadruped (primeros robots de patas)
- 1973, Stanford: Cart
- 1975, Francia: Hilare I
- 1980, C.M.U.: Rover
- 1985, Auge de robots en universidades y compañías hasta nuestro día

2.3 Tipos de vehículos robots terrestres

En cuanto a la clasificación de los robots no hay un acuerdo entre autores. No se puede concluir cuantos y cuales son los tipos de robots existentes. Pueden ser clasificados según el medio en el que se mueven (terrestres, acuáticos, aéreos, híbridos…), según su finalidad (domésticos, militares, educacionales, espaciales…), según su autonomía (teleoperados, semi-automáticos, autómaticos…), según su tamaño (robots, micro-robots, nano-robots…), etc. Como este proyecto consiste en el desarrollo de un vehículo robot terrestre, vamos a enfocar la clasificación en éstos, comentando brevemente los sistemas de locomoción más comunes.

2.3.1 Diseño con ruedas

Los vehículos con ruedas son la solución más simple para conseguir la movilidad en terrenos medianamente duros y libres de obstáculos, alcanzando velocidades relativamente altas. Emplean diferentes tipos de locomoción mediante ruedas que les confieren características y propiedades diferentes respecto a la eficiencia energética, dimensiones, cargas y maniobrabilidad.

Como limitación más significativa hay mencionar el deslizamiento en la impulsión, que dependiendo de las características del terreno será más o menos grave y nos permitirá un movilidad más o menos eficiente y velocidades más o menos altas. Por otra parte, dotar de sistemas de estabilidad para adaptarse a la configuración del terreno no es una tarea barata, lo que limita de forma importante los entornos aceptables para un robot de este tipo.

2.3.1.1 Tipo coche o Ackerman

Es el utilizado en vehículos de cuatro ruedas convencionales y es el tipo de vehículo de nuestro proyecto. El sistema se basa en dos ruedas traseras tractoras que se montan de forma paralela en el chasis principal del vehículo, mientras que las ruedas delanteras son de direccionamiento, y se solo se utilizan para seguir la trayectoria.

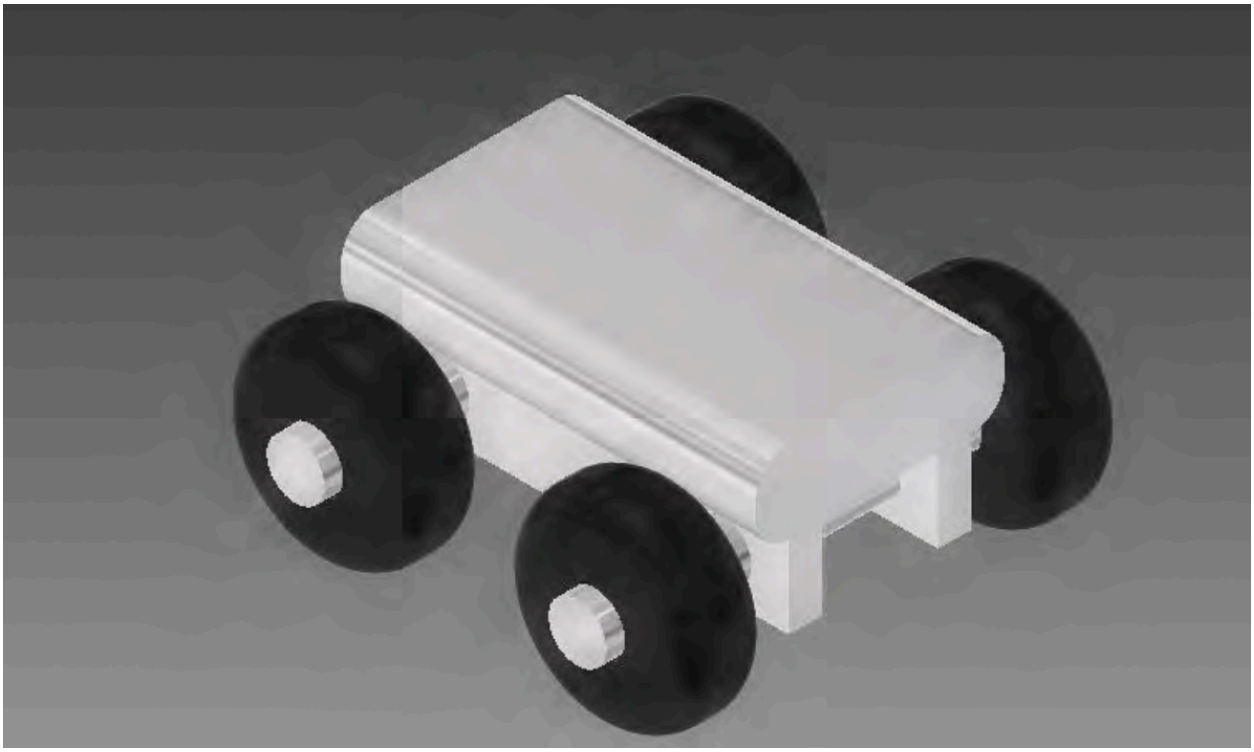


Figura 2-1. Diseño tipo coche o Ackerman

2.3.1.2 Triciclo

Esta configuración se basa en una rueda delantera que sirve tanto para la tracción como para el direccionamiento. El eje trasero, en cambio, consiste en dos ruedas pasivas que se mueven libremente. La maniobrabilidad es mayor que en la configuración de coche debido a la existencia de una sola rueda de direccionamiento, pero a su vez esto causa que se puedan presentar problemas de estabilidad en terrenos difíciles dado que el centro de gravedad tiende a desplazarse cuando el vehículo se desplaza por una pendiente, provocando una pérdida de tracción, o incluso el volcado. Ésta quizás sea la principal desventaja de este sistema de locomoción.

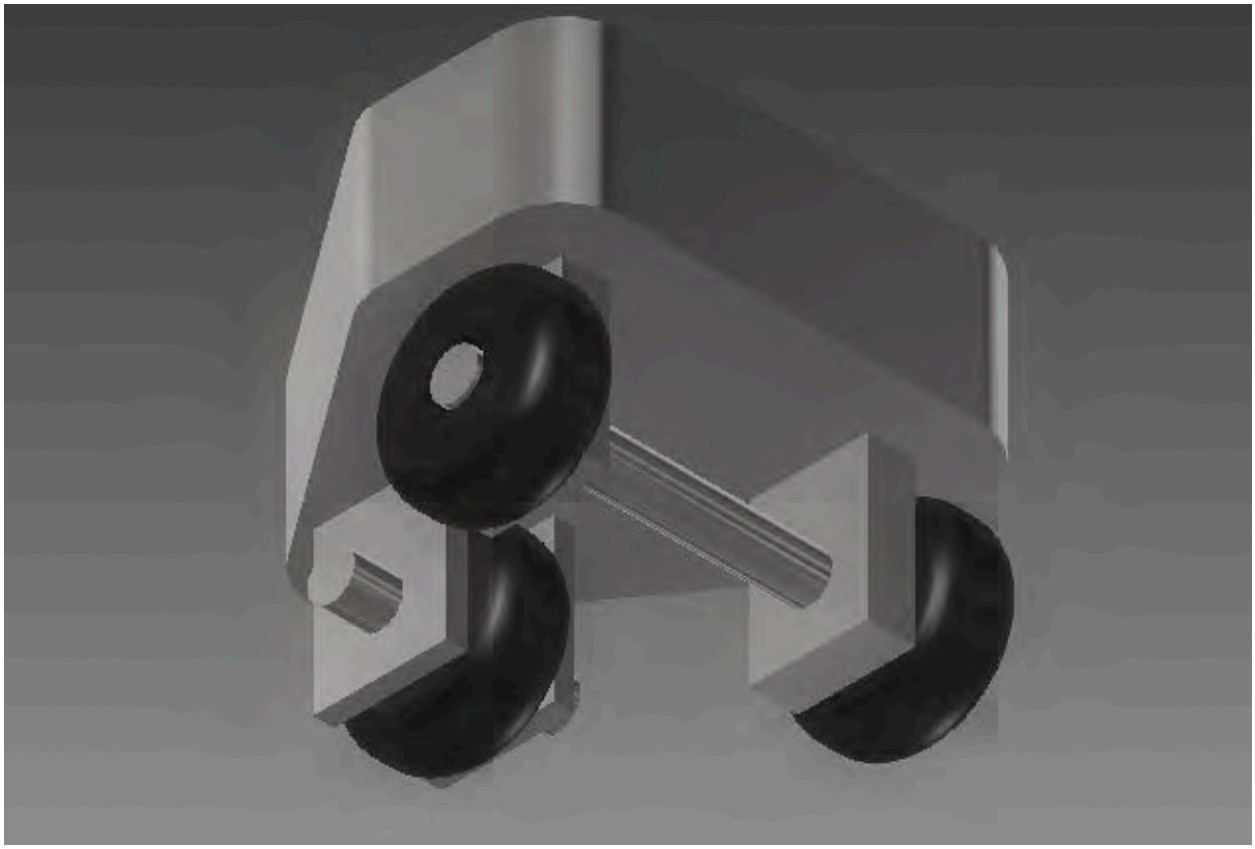


Figura 2-2. Diseño tipo triciclo

2.3.1.3 Diferencial

Tanto desde el punto de vista de la programación como de la construcción, el diseño diferencial es uno de los menos complicados. El robot puede ir recto, trazar curvas y girar sobre sí mismo. Es decir, puede hacer movimientos de traslación y también de rotación o spin. Este tipo de direccionamiento viene dado por la diferencia de velocidades de las ruedas laterales. Dos ruedas montadas en un único eje son independientemente propulsadas y controladas, proporcionando ambas tracción y direccionamiento.

Un problema importante es cómo resolver el equilibrio del robot. Hay que buscarle un apoyo adicional a las dos ruedas ya existentes. Esto se consigue mediante una o dos ruedas de apoyo añadidas en un diseño triangular o romboidal. El diseño triangular puede no ser suficiente dependiendo de la distribución de pesos del robot, y el romboidal puede provocar inadaptación al terreno si este es irregular, lo que puede exigir alguna clase de suspensión.

Otra consideración a hacer en este diseño es cómo conseguir que el robot se mueva recto. Para que el robot se mueva en línea recta sus ruedas tienen que girar a la misma velocidad y esto no siempre ocurre. Por ejemplo cuando los motores encuentran diferentes resistencias del suelo las velocidades de los motores varían y el robot girará incluso aún cuando se haya ajustado inicialmente para que vaya recto. Esto quiere decir que la velocidad debe ser controlada dinámicamente si queremos máxima fiabilidad de movimiento, lo que incrementaría el coste y la complejidad.

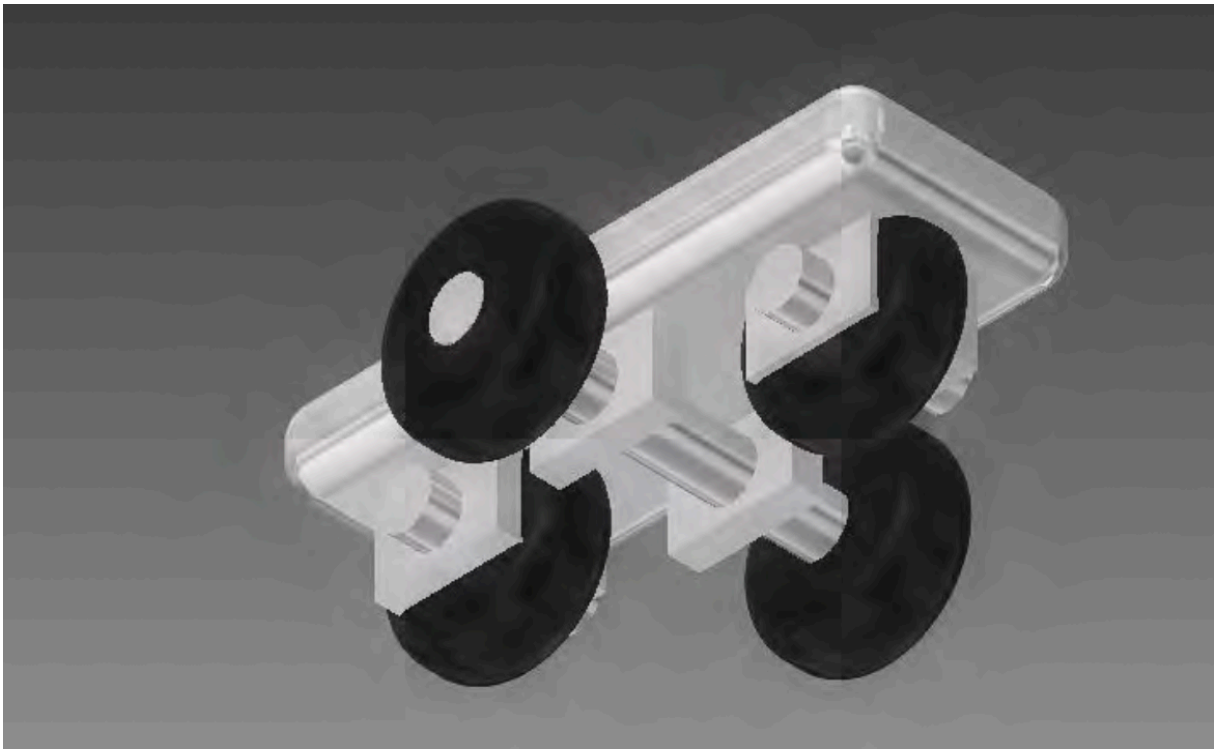


Figura 2-3. Diseño tipo diferencial

2.3.1.4 Skid steer

Se disponen varias ruedas en cada lado del vehículo que actúan de forma simultánea. El movimiento es el resultado de combinar las velocidades de las ruedas de la izquierda con las de la derecha. Estos robots se han se han utilizado en aplicaciones mineras y en misiones de exploración espaciales no tripuladas debido a la robustez del movimiento. Es un diseño orientado a terrenos difíciles en los que no se precisa tanto maniobrabilidad como fiabilidad en el avance del vehículo.

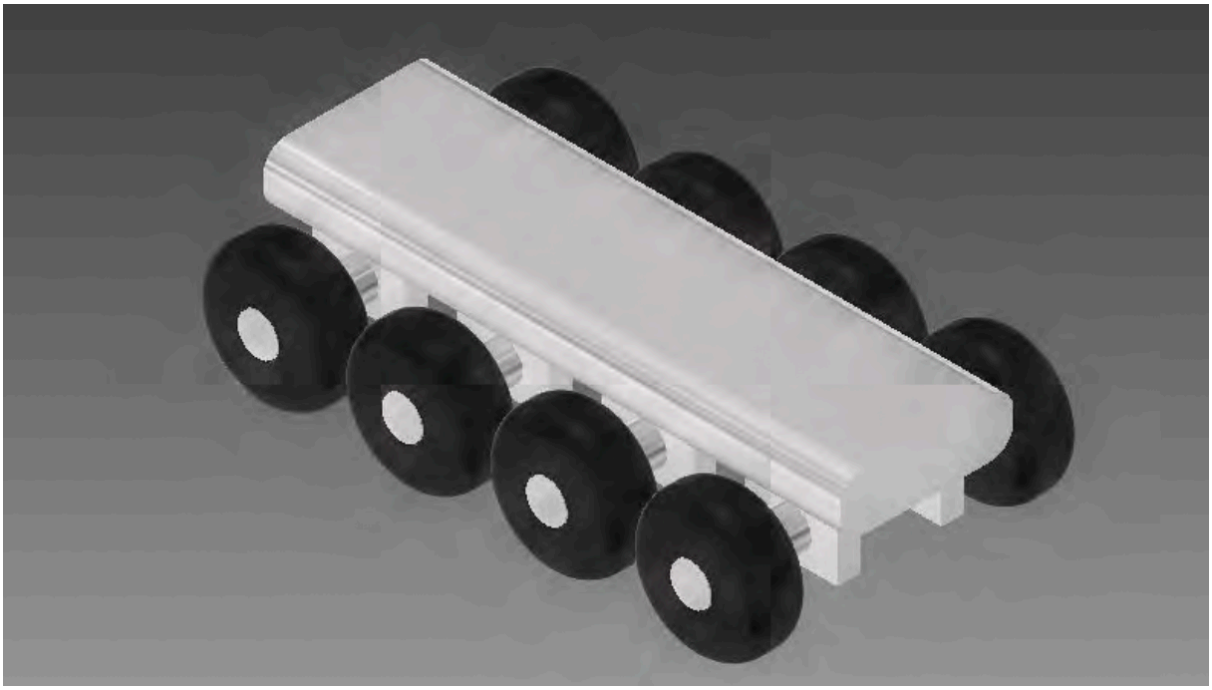


Figura 2-4. Diseño tipo Skid Steer

2.3.1.5 Síncrono

En este diseño todas las ruedas, generalmente tres, son tanto de dirección como motrices. Las ruedas están enclavadas de tal forma que siempre apuntan en la misma dirección y para cambiar de dirección el robot gira simultáneamente todas sus ruedas alrededor de un eje vertical, de modo que la dirección del robot cambia, pero su chasis sigue apuntando en la misma dirección que tenía. El diseño síncrono supera muchas de las dificultades que plantean el diseño diferencial, en triciclo y de coche, pero a costa de una mayor complejidad mecánica.

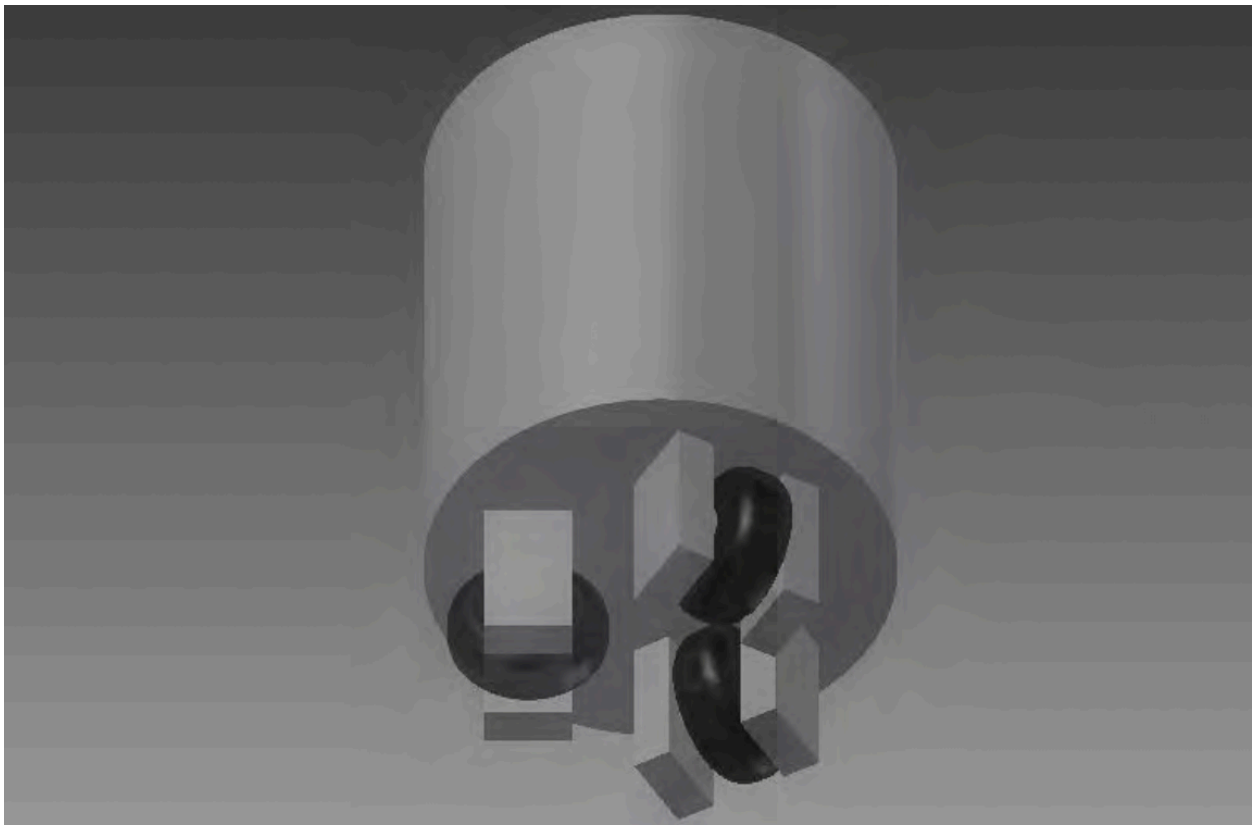


Figura 2-5. Diseño síncrono

2.3.1.6 Omnidireccional

Este sistema de tracción de basa en la utilización de tres ruedas directrices y motrices. Esta configuración tiene tres grados de libertad, por lo que puede realizar cualquier movimiento, posicionarse en cualquier posición y en cualquier orientación. No presenta limitaciones en cuanto a movimiento pero sí presenta la limitación de un precio prohibitivo.

Figura 2-6. Diseño omnidireccional

2.3.2 Diseño con pistas de deslizamiento

Son vehículos tipo oruga en los que tanto la impulsión como el direccionamiento se consiguen mediante pistas de deslizamiento. Pueden considerarse funcionalmente análogos al skid steer. La locomoción mediante pistas de deslizamiento es útil en navegación "campo a través" o en terrenos irregulares, en los cuales presenta un buen rendimiento. Las principales diferencias con el diseño skid steer son que la impulsión está menos limitada por el deslizamiento y la resistencia al desgaste es mayor pero el deslizamiento en los giros es más grande.

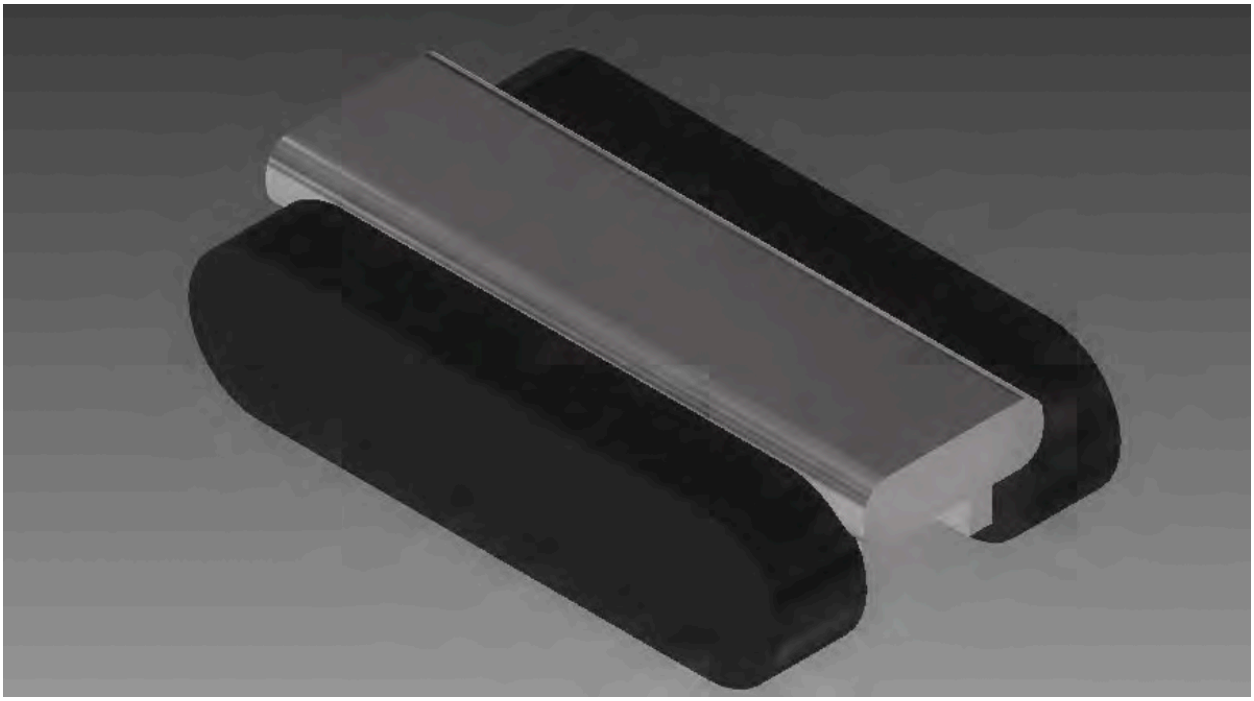


Figura 2-7. Diseño con pistas de deslizamiento

2.3.3 Diseño con patas

La motivación de este tipo de robot es imitar las distinas formas de desplazamiento de las que la naturaleza ha dotado a los animales, incluidos los humanos. Al dotar de movimiento con patas a un robot, debemos tener en cuenta su posición y velocidad, pero también debemos asegurar que el robot permanezca en equilibrio y no se caiga, usando solamente el movimiento en las articulaciones mediante motores.

En general, los sistemas que emplean patas son bastante complejos, por ejemplo, los robots bípedos y el problema que la estabilidad supone. Sin embargo, hay muchas configuraciones cuya complejidad es más asequible. Un sistema de patas tipo insecto es la solución básica dado que este problema se soluciona aumentando el número de patas. Se construye empleando sólo parejas de servos y dotándolos de una programación como la que sigue.

Para dar un paso un servo abre la pata, alejándola del cuerpo, para salvar un obstáculo si lo hubiese, luego el otro servo de la pareja gira para que la pata se mueva adelante. El primer servo, después, baja la pata hasta que esta toque el suelo, finalmente el segundo servo gira hacia atrás empujando el cuerpo del robot adelante. El movimiento coordinado de las patas permite al robot moverse adelante, atrás y girar. Existen multiples variantes dependiendo del numero de patas aunque todas tienen el mismo principio de funcionamiento mencionado.

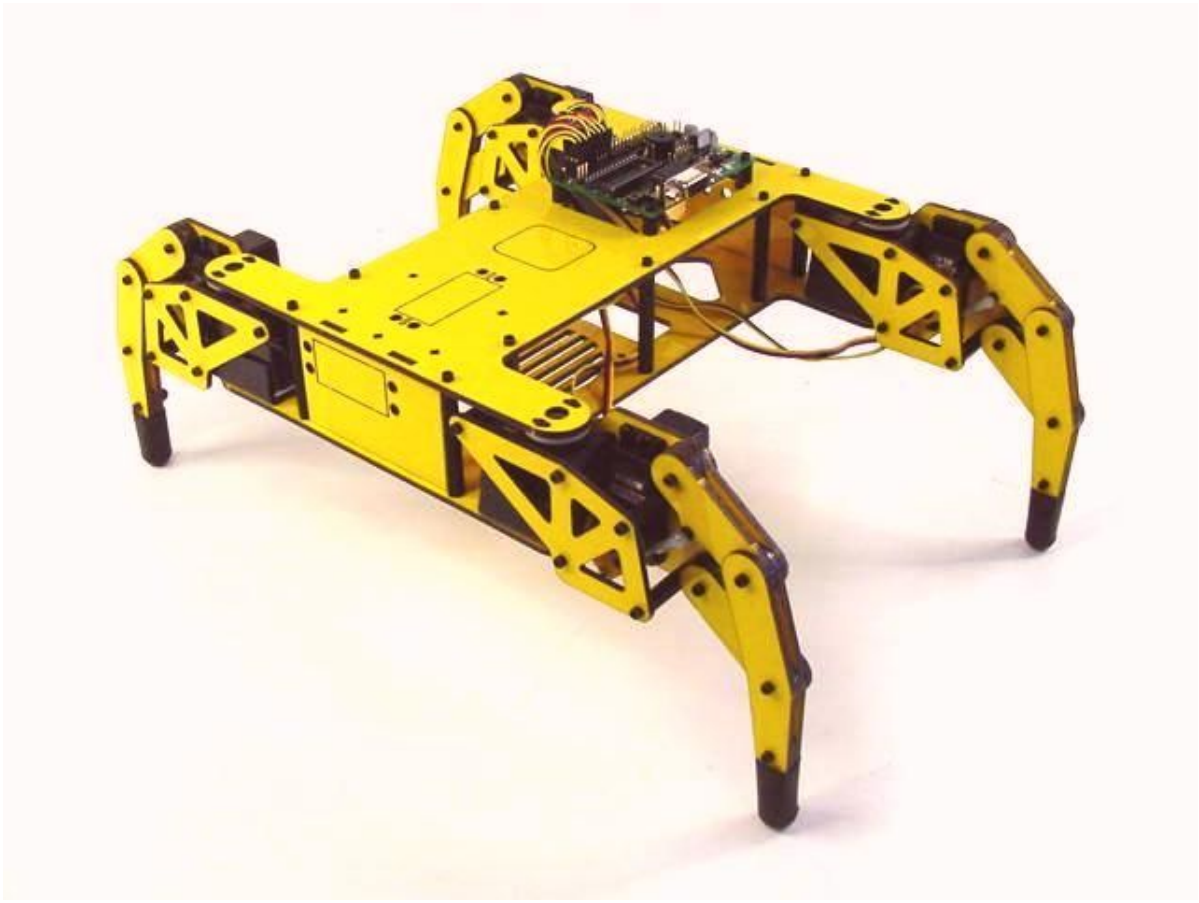


Figura 2-8. Diseño con patas o tipo insecto

2.4 Proyectos relacionados

Como se ha ido comentando a lo largo del capítulo, las aplicaciones y usos que se han dado y se darán a los robots son infinitas. Por eso mismo, no podemos abordar un capítulo en el que describamos todas las aplicaciones de los robots móviles. Lo que sí haremos será una recopilación de algunos proyectos relacionados

con el nuestro en algún sentido, ya sea que hagan uso de Raspberry Pi, que hayan sido desarrollados para la educación o que se controlen remotamente, entre otras razones. Este apartado hará las veces de Estado del Arte, ya que como hemos explicado, es muy difícil obtener una lista exhaustiva del estado actual de los vehículos robots para reconocimiento de entorno.

2.4.1 Vehículo Robot con Lego Technic, Minsdstorm y Raspberry Pi

Este proyecto desarrollado por José R. Sosa, consiste en adaptar Raspberry Pi a un vehículo LEGO motorizado con Mindstorm y controlado gracias a una aplicación Python. Hace uso de un módulo Wifi USB conectado a la Raspberry Pi para el acceso remoto vía SSH al control del vehículo. Una carcasa protectora será la encargada de la integración de Raspberry Pi con el vehículo Lego. Dispone, además, de una batería recargable para la alimentación de corriente de Raspberry Pi.

Este proyecto comparte con el nuestro el uso de Rapsberry Pi, una aplicación Python para su control y la forma de alimentar la Raspberry Pi mediante una batería externa, además del diseño tipo coche. La principal diferencia es que hace uso de SSH para las comunicaciones, mientras que en nuestro caso haremos uso de Bluetooth.

Figura 2-9. Proyecto relacionado 1

2.4.2 PiBot-B

Este proyecto desarrollado por Thomas Schoch consiste en adaptar la Raspberry Pi con un kit de chasis Zumo de Pololu. El PiBot-B se controla gracias a una aplicación de iPhone personalizada que se comunica a través de WiFi con Raspberry Pi. Un programa Python envía señales a un controlador de motor que impulsa los dos motores en el chasis Zumo. La aplicación para el iPhone muestra el vídeo de la cámara web y las teclas encargadas de los movimientos de PiBot-B. También dispone de un Sense HAT con matriz de LED integrada de 8 × 8 que indica su estado.

También es interesante nombrar este proyecto debido a su gran similitud con el nuestro. Además de la Raspberry Pi , hace uso de un chasis prefabricado y comercial, de una cámara, del SenseHat de Adafruit, y lo más importante, de una aplicación móvil que envía las direcciones de movimiento al robot. La principal diferencia es que hace uso de una cámara USB para PC y no del módulo de cámara de Raspberry. Ademñas podemos apreciar que hace uso de un diseño mediante pistas de deslizamiento.

Figura 2-10. Proyecto relacionado 2

2.4.3 Wifi Car “JJ”

Este proyecto llamado Wifi Car "JJ", fue patrocinado por VanLeeuwen Opensource Consultoría en la campaña de Kickstarter creada por Dexterindustries. También cuenta con 82 piezas LEGO y se compone principalmente por una placa BrickPi unida a una Raspberry que se controla vía Wifi gracias a una aplicación creada con el programa LEGO digital designer. Podemos apreciar un diseño tipo triciclo. Otro ejemplo más con características similares al nuestro y digno de mencionar.

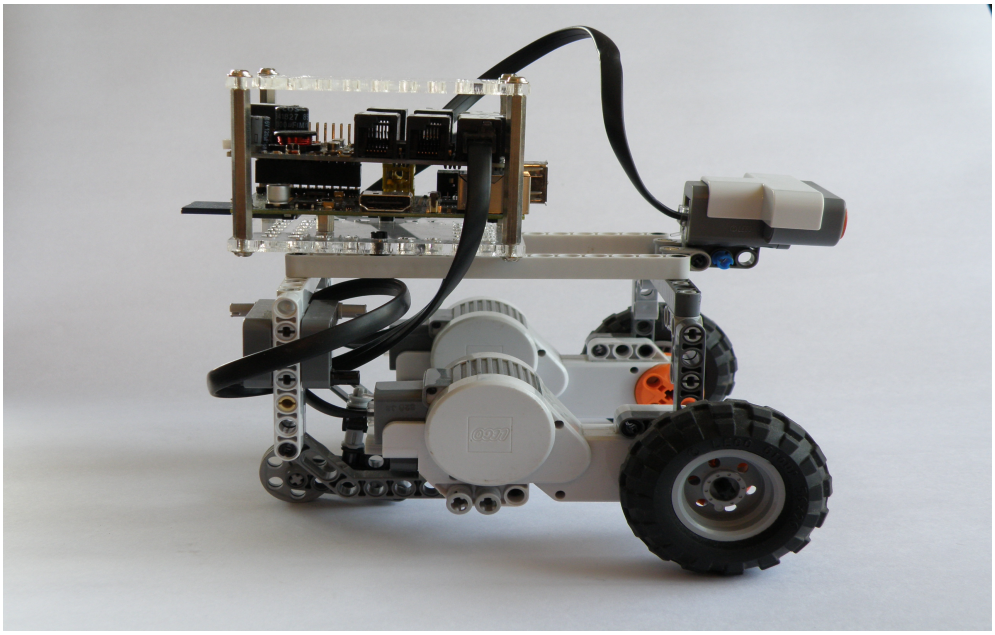


Figura 2-11. Proyecto relacionado 3

3 DESCRIPCIÓN DEL HARDWARE

En este capítulo describiremos el hardware utilizado en el proyecto. Hablaremos del sistema Raspberry Pi, de los sensores, controladores, módulos y periféricos, de sus fabricantes, de sus librerías de uso, de la forma en la que los conectaremos a la placa y de cualquier otro detalle que se considere importante para el entendimiento del sistema completo. Consistirá en un chasis comercial al que acoplaremos el sistema microprocesador Raspberry Pi y al que conectaremos los módulos necesarios para el movimiento de los motores y la recopilación de datos del entorno.

3.1 Parte mecánica

La parte mecánica del proyecto consiste en un vehículo tipo coche de la marca DFROBOT, modelo Pirate-4WD Mobile Platform. Éste es uno de los muchos chasis comerciales disponibles en el mercado de la electrónica para este fin. La decisión de adquirir este y no otro se basa en la facilidad de montaje. Al no ser el montaje del coche una tarea de ingeniería propiamente dicha, no se ha buscado un chasis que cumpla con ninguna especificación concreta sino la sencillez de montaje y uso. Esto sumado al bajo coste, hacen de este modelo una buena elección para este proyecto. El contenido de la caja incluye una guía de monteje muy útil e intuitiva. En la Figura 3-1 podemos ver una de las páginas del proceso de montaje a modo de ejemplo.

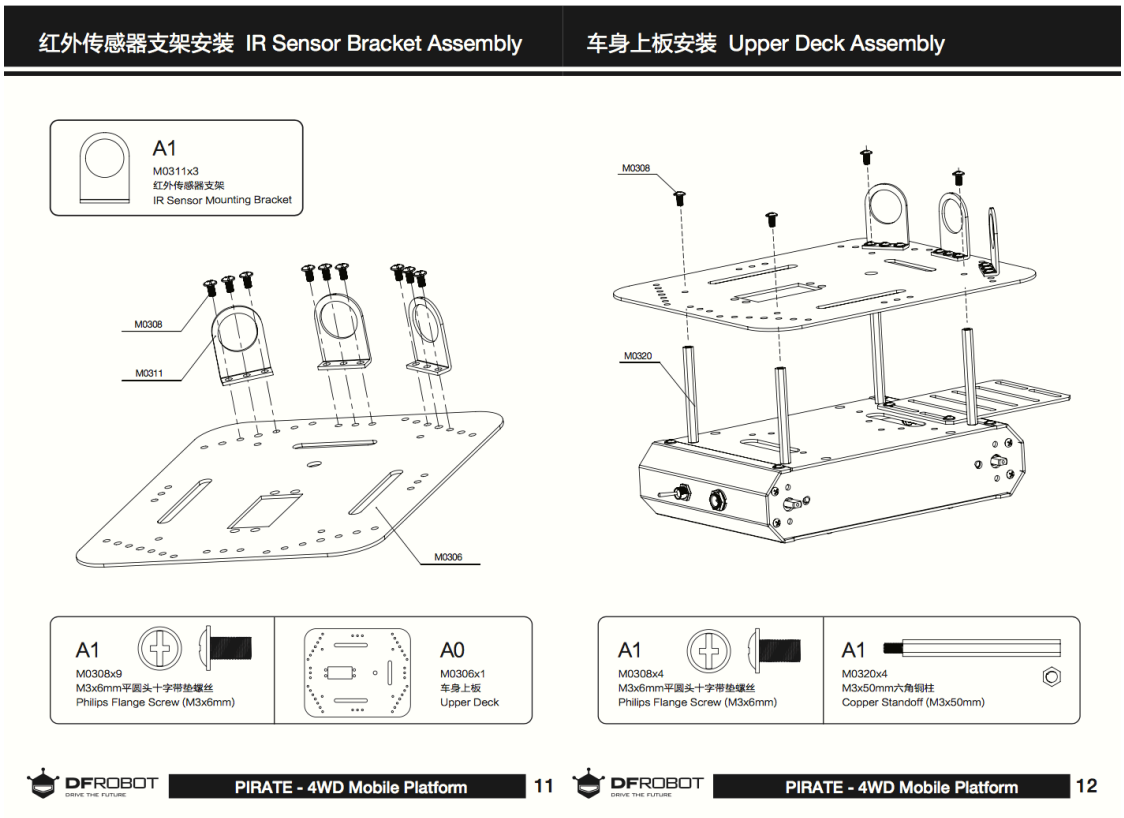


Figura 3-1. Ejemplo de página ded manual de montaje del chasis

Además, como se puede ver en la Figura 3-2 , este vehículo también incluye motores de corriente continua que se acoplarán a cada rueda para transmitir el moviento. A través de un controlador de motores, que veremos más adelante, se conectarán a la alimentación y se llevará a cabo la gestión del movimiento mediante el módulo PWM de Raspberry Pi.

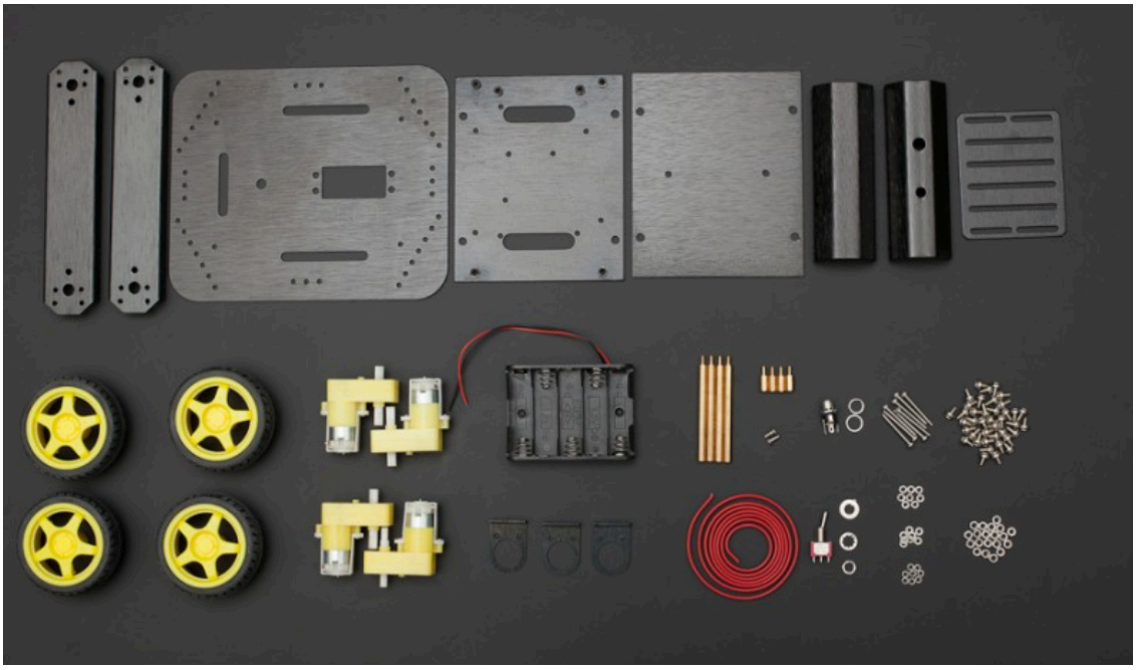


Figura 3-2. Contenido original de la caja del vehículo

Como se ha mencionado, el montaje completo del vehículo es tarea sencilla y de poca duración, por ello no daremos más importancia a este apartado más que mostrar el resultado final tras el montaje. En [5] podemos ver toda la documentación oficial y, entre ella, el manual utilizado para el montaje completo.

Figura 3-3. Resultado final del montaje del vehículo

3.2 Parte Electrónica

Si en algo debe destacar este proyecto, es en lo que refiere a la electrónica. Esta parte del diseño es bastante variable, y por ello, personal. En el mercado existen infinidad de productos con los que podríamos cumplir los mismos objetivos. Un ejemplo de ello es el microcontrolador. Existen múltiples opciones de ellos en el mercado capaces de soportar el presente proyecto. Nuestro objetivo es que la plataforma sea de código abierto, para así facilitar el desarrollo del trabajo. Entre estas plataformas encontramos, entre otras, Raspberry Pi, BeagleBone, Sharks Cove, Waspmote o Arduino.

Todavía más sorprendente es la diversidad de módulos que podemos conectar a estos sistemas. Entre estos módulos encontramos sensores de todo tipo, cámaras, módulos GPS, controladores, módulos de comunicaciones, etc. Es decir, todo lo que se nos pueda ocurrir, existe. A lo largo de este apartado describiremos los módulos que usaremos en el proyecto.

3.2.1 Raspberry Pi

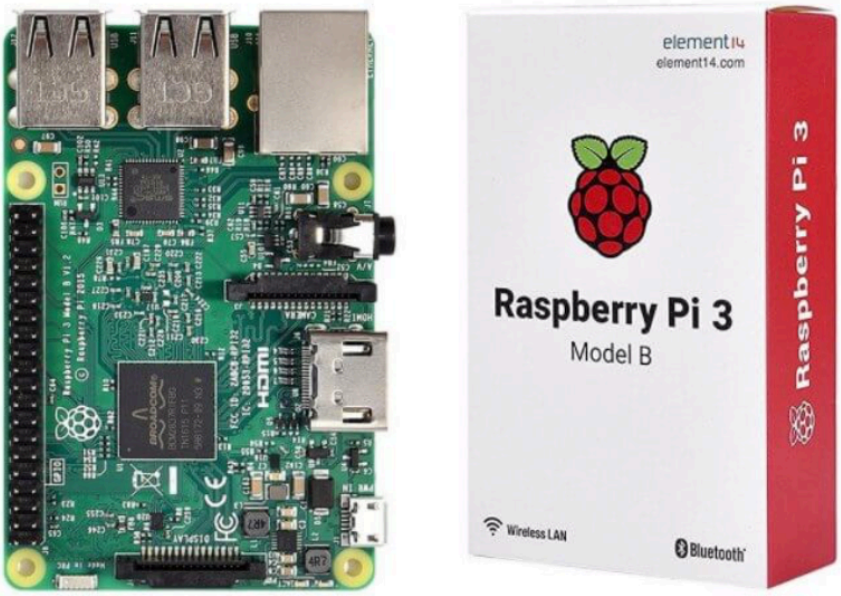


Figura 3-4. Raspberry Pi

Durante los años 80 proliferaron en los centros docentes microordenadores que permitían enseñar nociones de programación y cuyo bajo precio los hacía sencillos de reponer. En la década de los 90 la tendencia se interrumpió con los PC demasiado caros y los sistemas difíciles de reprogramar. Se agravó con la llegada de los smartphones y tablets, dispositivos completamente cerrados. Es entonces cuando surge la figura de Eben Upton, ex profesor de la Universidad de Cambridge, que idea y pone en marcha el proyecto Raspberry.

Raspberry Pi, conocido también como RPi, es un computador de placa reducida o simple board computer (SBC). Es decir, un pequeño ordenador del tamaño de una tarjeta de crédito. Fue desarrollado en 2006 por la Fundación Raspberry Pi con el objetivo de estimular la enseñanza de las ciencias informáticas en las escuelas de todo el mundo. Este ordenador es muy completo en cuanto a especificaciones y más si tenemos en cuenta que su precio ronda los 35€. No incluye cable de alimentación, carcasa de protección, ni dispositivo de almacenamiento, por ello, se utiliza una tarjeta de memoria SD externa. No obstante dispone de múltiples entradas (USB, HDMI, Ethernet, etc..) mediante las cuales podemos conectarle otros periféricos como es el caso del ratón y teclado, un monitor y si quisiéramos disponer de gran cantidades de memoria de almacenamiento, podríamos llegar a conectar un disco duro externo.

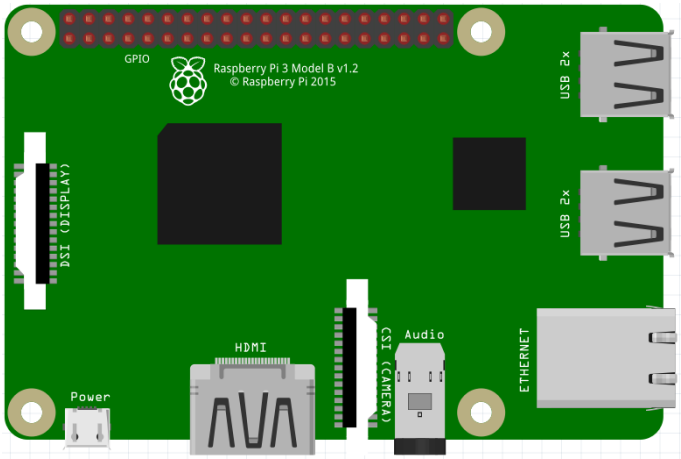


Figura 3-5. Esquema de Raspberry Pi 3

Cabe decir que tiene ciertas limitaciones que impiden considerarla un ordenador como podemos entender un PC de sobremesa o portátil, ya que carece de buena velocidad de respuesta tanto a la hora de cargar páginas web como a la hora de ejecutar programas que requieran un uso considerable de memoria RAM. Por el contrario, su tamaño y el uso el código abierto, hacen que sea una máquina con infinidad de posibilidades creativas. Gracias a la comunidad de desarrolladores con la que cuenta, cada vez más, se van compartiendo experiencias personales así como proyectos acabados para que cualquier usuario de este hardware pueda probar en sus manos. Sin ir más lejos. éste podría ser uno de ellos.

Existen varios modelos de Raspberry Pi que han ido actualizándose con los años, desde Raspberry Pi 1 Modelo A hasta Raspberry Pi 3 Modelo B. Para no extender este apartado demasiado, vamos a exponer una tabla comparativa de todos los modelos de los aspectos más interesantes a comparar.

Tabla 3–1 Comparativa de modelos Raspberry Pi según especificaciones técnicas

	Raspberry Pi 1 Modelo A	Raspberry Pi 1 Modelo B	Raspberry Pi 1 Modelo B+	Raspberry Pi 2 Modelo B	Raspberry Pi 3 Modelo B
SoC	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + Puerto USB)			Broadcom BCM2836 (CPU + GPU + DSP + SDRAM + Puerto USB	Broadcom BCM2836 (CPU + GPU + DSP + SDRAM + Puerto USB
CPU	ARM 1176JZF-S a 700 Mhz			ARM Cortex A7 quad-core 900 MHz	ARMv8 64-bit quad-core 1.2 GHz
Juego de Instruciones	RISC de 32 bits				
GPU	Broadcom VideoCore IV OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia), 1080p30 H.264/MPEG-4 AVC				
Memoria	256 MiB compartidos con	512 MiB (compartidos con la		1 GB (compartidos con GPU)	

(SDRAM)	GPU	GPU)		
Conectividad de red	Ninguna	10/100 Ethernet (RJ-45) via hub USB		10/100 Ethernet (RJ-45) vía hub USB, Wifi 802.11n, Bluetooth 4.1
Puertos USB	1	2	4	
Ranura para almacenamiento	SD/MMC		MicroSD	
Periféricos de bajo nivel	8 x GPIO, SPI, I²C, UART			17 x GPIO y un bus HAT ID
Consumo	500 mA (2.5 W)	700 mA (3.5W)	600 mA (3 W)	800 mA(4 W)
Dimensiones	85.60mm × 53.98mm			
Sistemas Operativos soportados	GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux, SUSE [68] Linux Enterprise Server for ARM.			

La fundación no indica expresamente si es hardware libre o con derechos de marca aunque prácticamente se da por hecho que es un producto con propiedad registrada pero de uso libre. De esa forma mantienen el control de la plataforma pero permitiendo su uso libre tanto a nivel educativo como particular. El software sí es open source, siendo su sistema operativo oficial una versión adaptada de Debian, denominada Raspbian, aunque como puede apreciarse en la Tabla 3–1, permite otros sistemas operativos, incluido una versión de Windows 10. La fundación promueve principalmente el aprendizaje del lenguaje de programación Python aunque otros lenguajes también soportados son Tiny BASIC, C, Perl y Ruby.

El modelo elegido para este proyecto ha sido Raspberry Pi 3 Modelo B, es decir, el más reciente. La decisión ha radicado en la incorporación de conexión WiFi y Bluetooth sin necesidad de añadir módulos USB para dichas funciones. Además, por supuesto, el hecho de que ser el modelo más reciente alargará la vida útil del proyecto.

3.2.2 Sense HAT

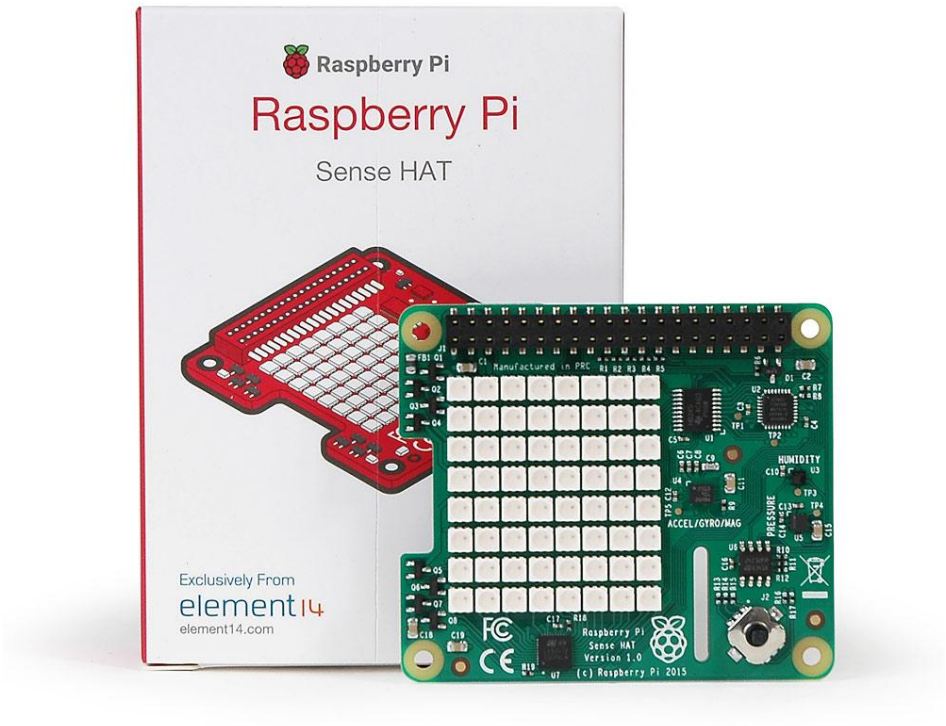


Figura 3-6. Sense HAT

Este dispositivo será el encargado de sensar la temperatura y humedad en nuestro proyecto. El Sense HAT (Hardware Attached on Tom), o sombrero, es un módulo adicional para Raspberry Pi. Fue creado especialmente para la misión Astro Pi, lanzado a la Estación Espacial Internacional en diciembre de 2015. Este dispositivo es verdaderamente sorprendente. Un complemento de bajo coste que permite recopilar múltiples medidas a través de un conjunto de sensores.

Cuenta con un sensor de presión barométrica con 24 bits de resolución, un sensor de temperatura con 16 bits de resolución y otro sensor de temperatura, combinado con un sensor de humedad, ambos con 16 bit de resolución, además de un sensor de humedad relativa. También cuenta con un sensor de aceleración, giroscopio y magnetómetro con nueve grados de libertad y 16 bit de resolución. Todo esto acompañado de una matriz de 8 × 8 RGB LED y un joystick de cinco botones. Los sensores pueden tomar muestras de forma periódica en una cola FIFO, pudiendo tomar valores en algunos de los sensores con una frecuencia máxima de 25 muestras por segundo. Por ejemplo, el acelerómetro, giróscopo y el medidor de campo magnético pueden tomar 952 muestras por segundo. Esto lo convierte en el recurso perfecto para llevar a cabo la gestión de estabilizar un dron aéreo, por ejemplo.

Como puede apreciarse en la Figura 3-7, el Sense HAT se conecta a la Raspberry Pi haciendo uso de los 40 pines GPIO. Todos sus sensores se conectan a través de I2C y la programación de la placa se hace mediante la interfaz SPI de la Raspberry, con lo que podemos reprogramarla o agregar firmware.

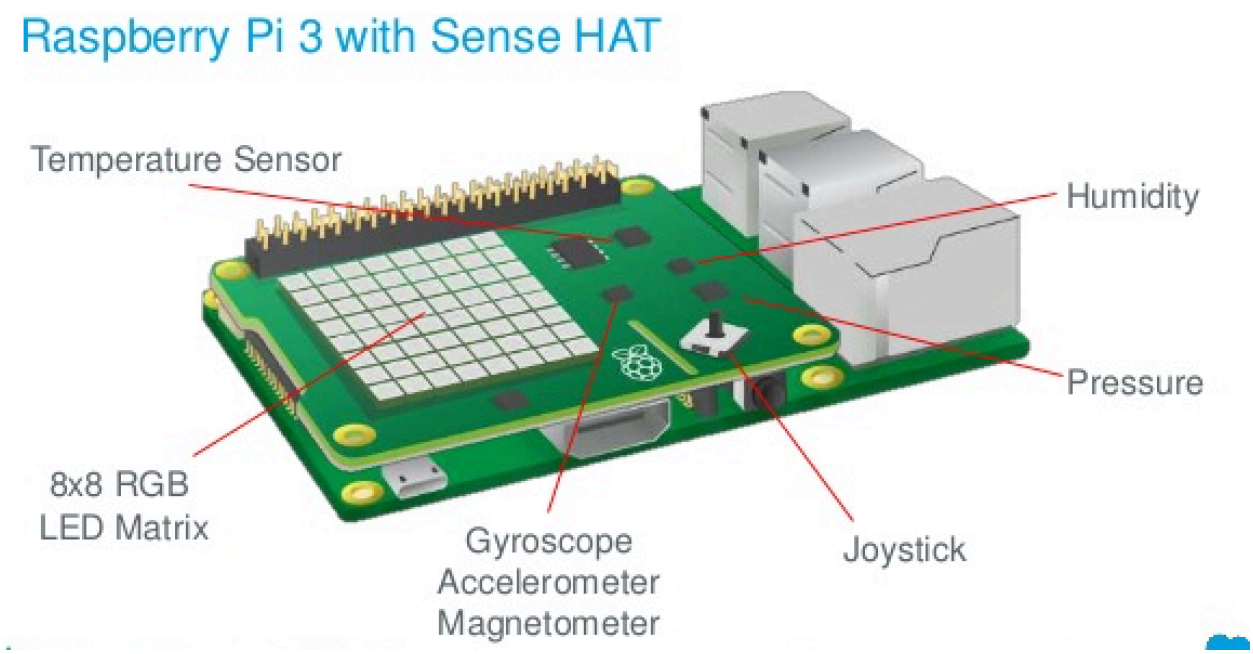


Figura 3-7. Sense HAT conectado a Raspberry Pi

Tabla 3–2 Especificaciones técnicas de Sense HAT

Módulo	Especificaciones
Giroscopio	Sensor de ratio angular: ±245/500/2000dps
Acelerómetro	Sensor de aceleración lineal: ±2/4/8/16 g
Magnetómetro	Sensor magnético: ±4/8/12/16 gauss
Barómetro	260 – 1260 hPa (la precisión depende de la temperatura y la presión, ±0.1 hPa en condiciones normales)
Sensor de temperaratura	Precisión en la temperatura de ± 2 °C en el rango de 0-65 °C
Sensor de humedad relativa	Precisión de ±4.5% en el rango de 20-80%rH , precisión de ± 0.5 °C en el rango de 15- 40 °C)
Matriz de LEDs	8x8 elementos con 15 bits de resolución
Joystick	5 botones, escaneado de 80 veces por segundo

En la Tabla 3–2 se muestran las especificaciones técnicas de cada sensor incluido según el datasheet oficial. El conjunto de todos estos sensores y actuadores convierte al Sense HAT en uno de los mejores complementos oficiales de Raspberry Pi. Esta integración es el motivo por el cual lo hemos elegido para el proyecto. Más en menos espacio. Aunque en nuestro proyecto sólo hacemos uso de los sensores de presión y humedad, el sense HAT nos ofrece la posibilidad de ampliar las funciones del robot en un futuro simplemente añadiendo la funcionalidad de uno u otro sensor en el código Python que desarrollaremos a lo largo del documento. Además, incluye una completa librería en Python denominada SenseHat, la cual hace aún más fácil el uso de uno u otro sensor.

3.2.3 Módulo de cámara V2

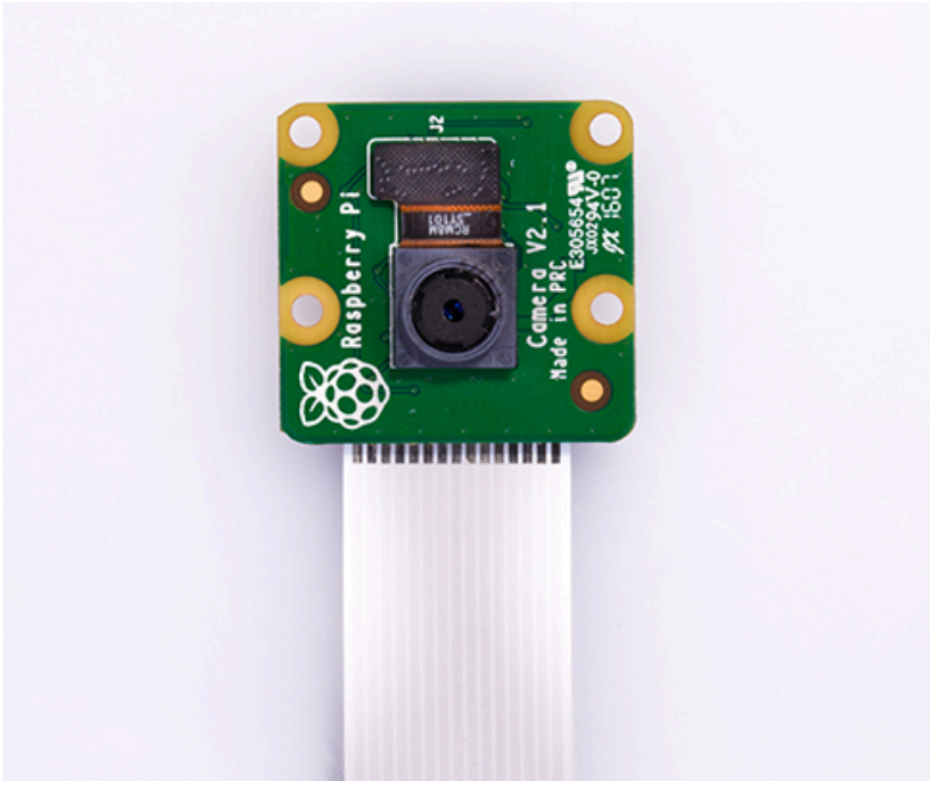


Figura 3-8. Módulo de cámara V2 para Raspberry Pi

Este módulo es la segunda generación del módulo de cámara diseñado específicamente para y por Raspberry. Consiste en una cámara con sensor de imagen de alta calidad IMX219 de 8 megapíxeles de Sony con una lente de enfoque fijo. Es capaz de tomar imágenes estáticas de 3280 x 2464 y es compatible con los formatos de video 1080p 30 fps, 720p a 60fps y 640x480p a 60/90 fps. Dado que no es materia de este proyecto el estudio de las especificaciones técnicas detalladas de la cámara. En [7] podemos ver el enlace a la documentación oficial.

Como puede apreciarse en la Figura 3-8, este módulo se conecta a la Raspberry Pi por medio de un cable de de cinta corta y un pequeño conector en la parte superior de la tarjeta. Utiliza la interfaz dedicada CSi diseñada especialmente para la conexión de cámaras y sobre la cual hablaremos en siguientes apartados.

Figura 3-9. Conexíon del módulo de cámara V2 a Raspberry Pi

La placa tiene un tamaño muy reducido, alrededor de 25 mm x 20 mm x 9 mm. También pesa poco más de 3g, por lo que resulta perfecto para nuestro proyecto donde el tamaño y el peso son importantes. Se conecta a Raspberry Pi por medio de un cable de cinta corta. Todos estos datos y más datos técnicos sobre las medidas los vemos en la Figura 3-10

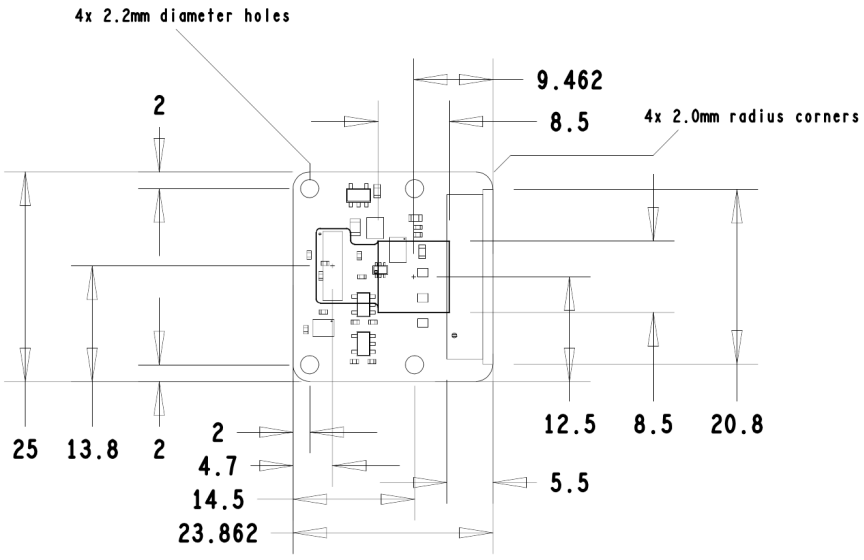


Figura 3-10. Medidas técnicas del módulo de cámara V2

La compañía no especifica que sea compatible con otros sistemas operativos, garantizando, en cambio, su compatibilidad con la ultima versión de Raspbian. Como la gran mayoría de los módulos para Raspberry, tiene asociado la librería Picamera para su correcto uso con Python. Esta librería resulta fácil de usar para principiantes pero, en cambio, ofrece multitud de opciones para usuarios avanzados desde time-lapse o cámara lenta hasta efectos visuales de todo tipo. Este módulo abre las puertas a múltiples aplicaciones desde detección de movimientos u obstáculos hasta cámara de seguridad aunque en este proyecto limitaremos su uso a capturas de imágenes del entorno.

3.2.4 Kit controlador de motores Pololu DRV8835

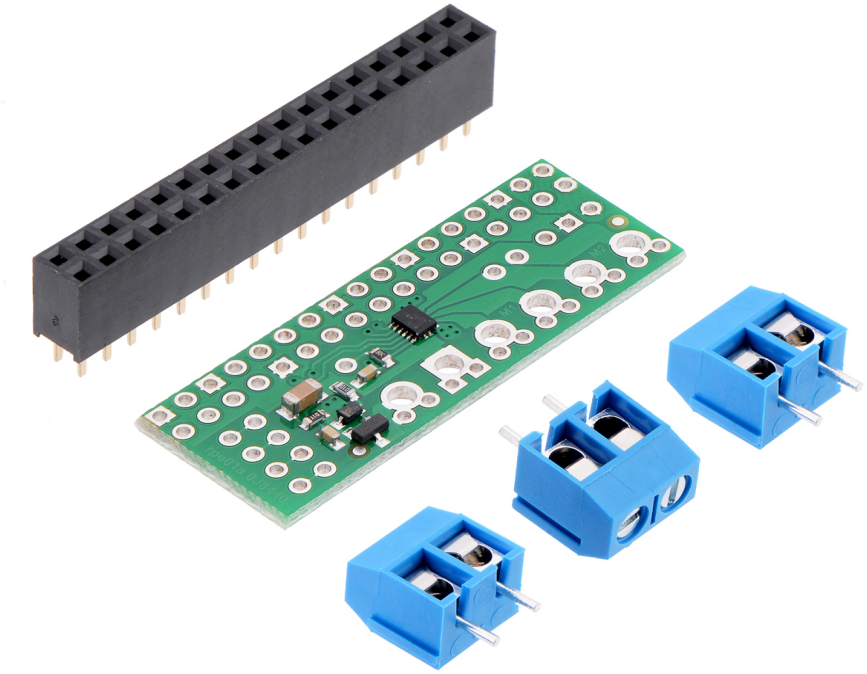


Figura 3-11. Contenido del kit controlador de motores DRV8835

El movimiento del vehículo robot es uno de los aspectos más importantes del proyecto. Como se explicará en el próximo capítulo, dispondremos de un control remoto para dirigir el robot. La otra parte encargada del movimiento será esta pequeña placa. Consiste en un kit controlador que ofrece una solución sencilla para controlar dos motores bidireccionales de corriente continua. Para ello, la placa hace uso del controlador DRV8835 en puente H dual de Texas Instruments a través de una librería Python

Como puede verse en la Figura 3-12 y en la Figura 3-13 se conecta directamente a la cabecera de pines haciendo uso de los pines GPIO 5 (pin 29) y 6 (pin 31) para la dirección de las ruedas y de los pines 12 (pin 32) y 13 (pin 33) para la salida PWM que controlará la velocidad.

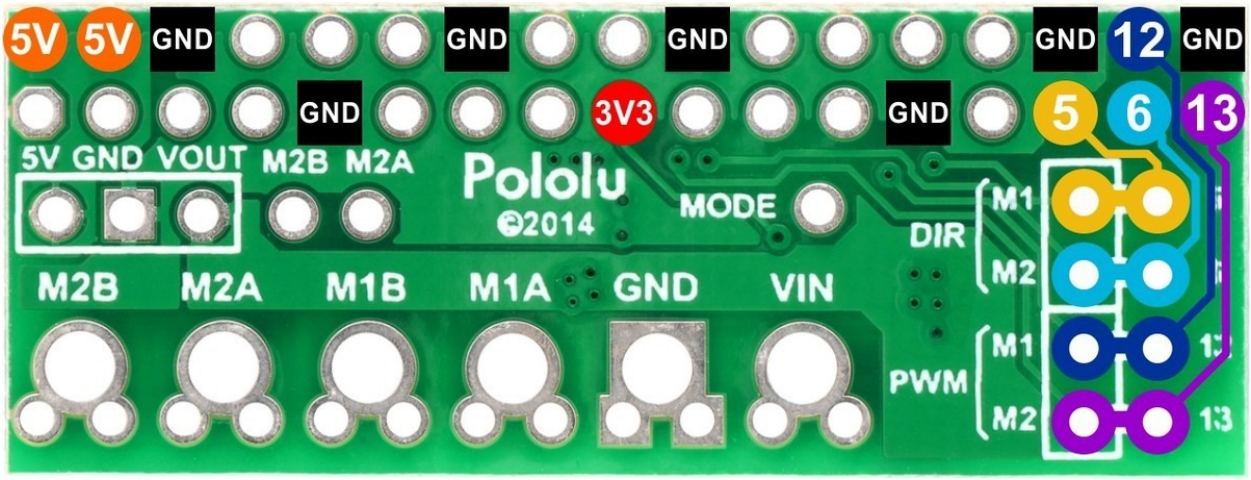


Figura 3-12. Placa del controlador DRV8835

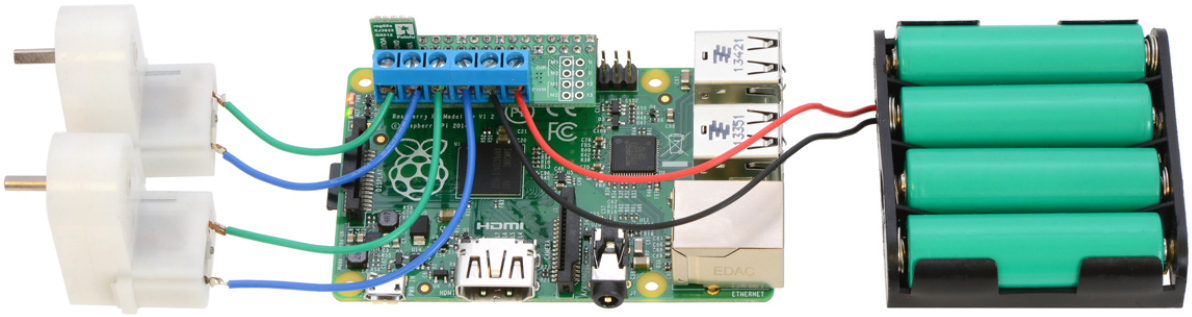


Figura 3-13. Conexionado completo del controlador DRV8835

Puede funcionar desde 2 V a 11 V, por lo que es una gran opción de control de motores de baja tensión. La placa puede entregar hasta 1,2A (1,5A en pico) por motor, o hasta 2.4A (3 en pico) a un único motor cuando está configurado con dos canales conectados en paralelo y el PWM puede operar hasta 250 kHz. Como en apartados anteriores, no vamos a entrar en describir al detalle las especificaciones de este componente. En [8] dispnemos de la documentación oficial donde pueden encontrarse las características técnicas, instrucciones de uso y esquemáticos.

3.2.5 Módulo Microstack GPS

Otra de las funciones del vehículo robot es la de obtener la posición geográfica. Este módulo será el encargado de llevar a cabo esta tarea ya que agrega funciones de posicionamiento GPS a todas las plataformas compatibles con el marco Microstack, entre ellas, Raspberry Pi

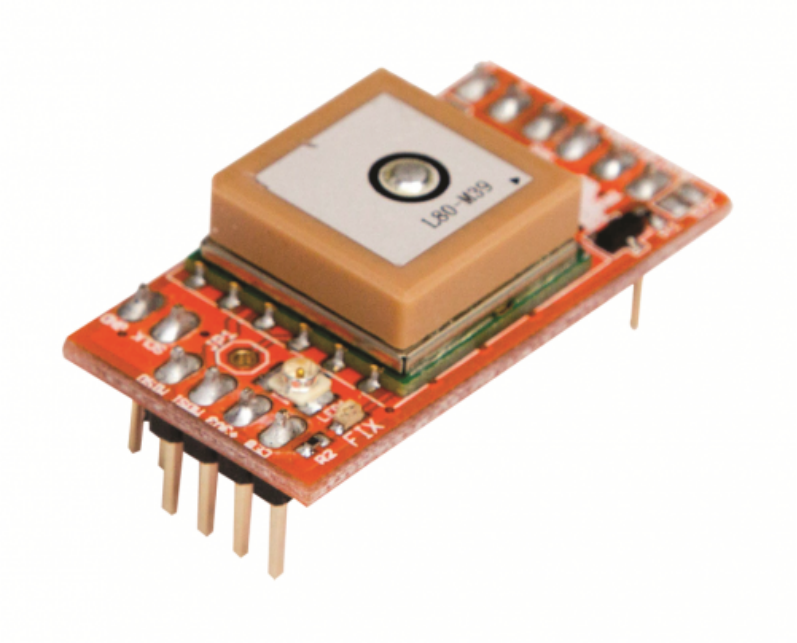


Figura 3-14. Módulo Microstack GPS

Cuenta con el módulo ultra delgado POT (Patch on Top) de L80 con una antena incorporada, el cual puede apreciarse en la Figura 3-14. Combina AGPS (GPS avanzado) llamado EASY (Embedded Assist System) y la tecnología AlwaysLocate para conseguir un buen rendimiento que cumple plenamente con el estándar industrial. La tecnología EASY asegura que puede calcular y predecir órbitas automáticamente usando los datos de efemérides (hasta 3 días) almacenados en la memoria flash interna, por lo que L80 puede fijar la posición rápidamente incluso con niveles de señal bajos debido a interiores, todo ello con bajo consumo de energía. Con la tecnología AlwaysLocate puede adaptar el tiempo de encendido y apagado para conseguir un equilibrio entre la precisión de posicionamiento y el consumo de energía según las condiciones ambientales y de movimiento.

En la Figura 3-15 vemos la correspondencia de pines. Todos son prescindibles excepto los pines de alimentación y los pines MTXSRX y MRXSTX, encargados de transmisión y recepción de datos.

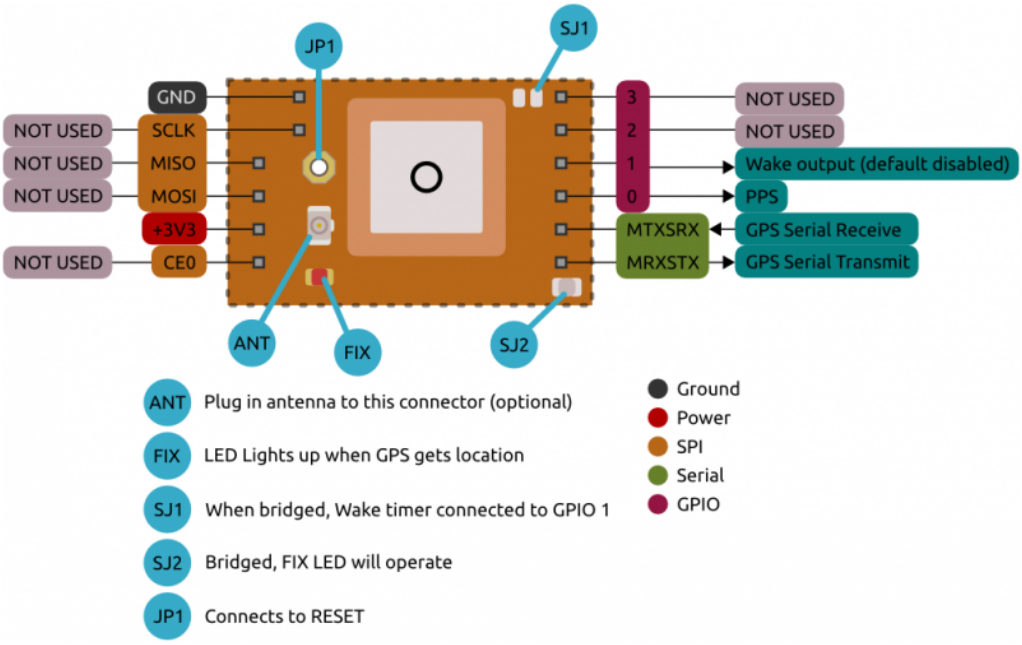


Figura 3-15. Pinout del módulo Microstack GPS

Aunque no vamos a entrar en detalles técnicos, en este caso sí vamos a mencionar algunas características importantes.

- Baja potencia (25 mA en modo de adquisición, 20 mA en seguimiento, hasta 3mA con el modo AlwaysLocate)
- Tiempo de fix muy corto
- Además de la antena incorporada, permite la conexión de una antena externa
- Señal de pulso por segundo (PPS) para sincronización de precisión
- Receptor de alta sensibilidad AntiJamming
- Períodos automáticos programables de despertar / dormir
- Modo AlwaysLocate para el reposo automático durante la inactividad para ahorrar energía
- Se puede utilizar con Raspberry Pi mediante un adaptador
- Se puede utilizar directamente en placas y prototipos con diseños personalizados

Cabe mencionar que este dispositivo, aún habiendo cumplido finalmente su objetivo, ha sido motivo de muchos errores y sus consecuentes horas de trabajo. El principal problema, y quiero destacar la gravedad de éste, ha sido una falta de actualización en la documentación [9] y en las librerías de uso por parte de Microstack que ha causado que en mitad del proyecto nos viésemos obligados a cambiar de Python 2.7 a Python 3. Otro problema ha sido trabajar sobre la programación del GPS estando en interiores, ya que si el dispositivo no dispone de cobertura lanza excepciones en el programa y no permite continuar la ejecución.

3.2.6 Esquema de conexionado

En este apartado veremos cómo se conecta cada bloque, a través de qué pines y mediante qué interfaces. De modo orientativo, vemos en la Figura 3-16 un diagrama de bloques del sistema en general. En el diagrama podemos observar como el sistema Raspberry ejerce de núcleo microprocesar y a él conectamos el resto de módulos mediante las distintas interfaces.

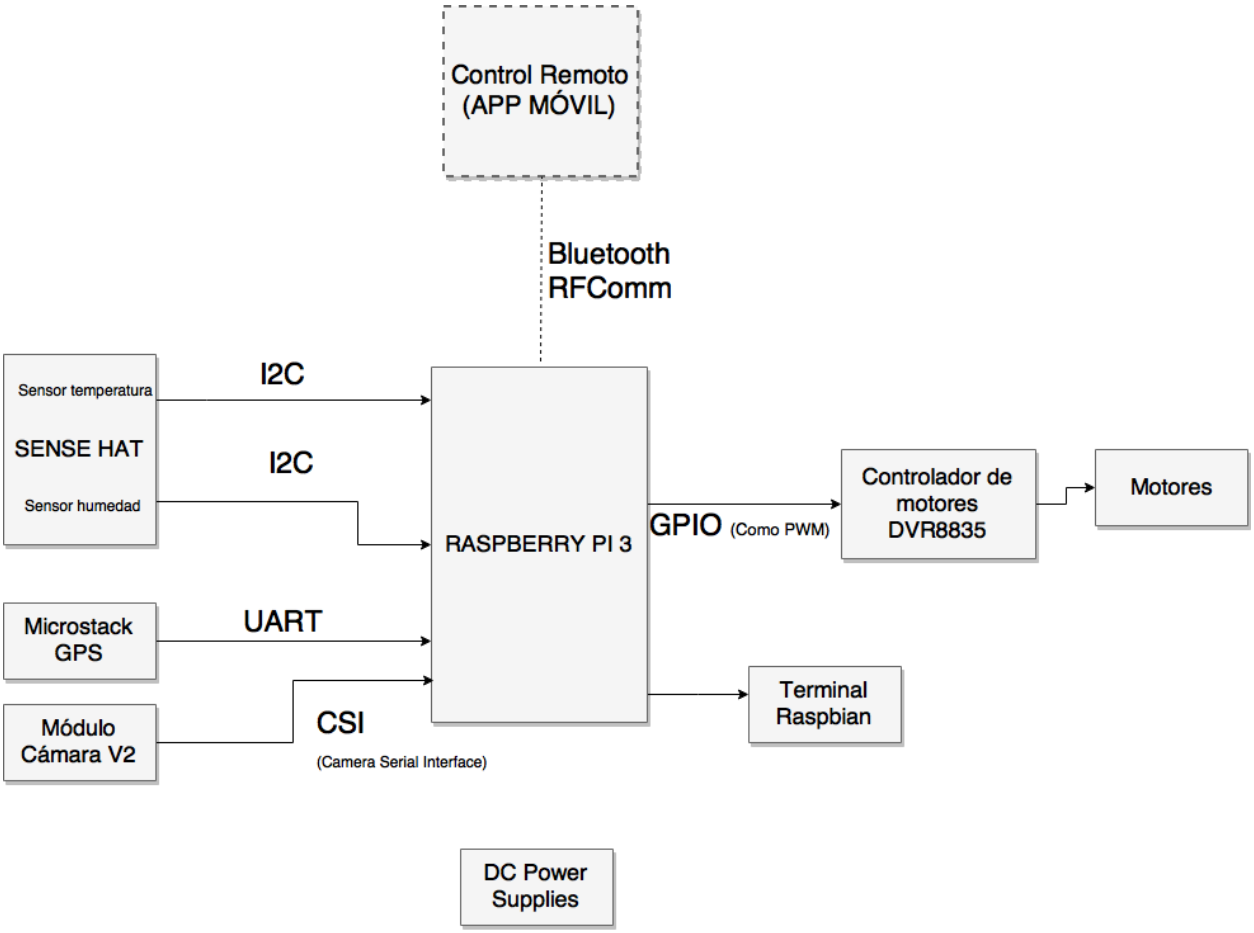


Figura 3-16. Diagrama de bloques del sistema completo

A continuación, en la Tabla 3–3 explicamos brevemente cada una de estas interfaces.

Tabla 3–3 Descripción de interfaces usadas en las conexiones

Interfaz	Descripción
I2C	I2C es un acrónimo de “Inter-Integrated Circuit”. Estándar que facilita la comunicación entre microcontroladores, memorias y otros dispositivos con cierto nivel de "inteligencia", sólo requiere de dos líneas de señal. La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca los pulsos de reloj (SCL) y la otra se utiliza para intercambiar datos(SDA).
UART	UART es un acrónimo de “Universal Asynchronous Receiver-Transmitter”, Transmisor-Receptor Asíncrono Universal. Es un dispositivo que controla los puertos y dispositivos serie Las funciones principales de UART son manejar las interrupciones de los dispositivos conectados al puerto serie y convertir los datos paralelos transmitidos al bus de sistema a datos en formato serie, para que puedan ser transmitidos a través de los puertos y viceversa.
CSI	CSI es un acrónimo de “Camera Serial Interface” En español, Interfaz Serie para Cámaras. Es una especificación de la Mobile Industry Processor Interface Alliance (MIPI Alliance) que define la interfaz entre una cámara digital y un procesador anfitrión y proporciona una conexión eléctrica en bus entre los dos dispositivos.
GPIO	GPIO es un acrónimo de “General Purpose Input/Output”. En español Entrada/Salida de Propósito General. Es un pin genérico en un chip, cuyo comportamiento (incluyendo si es un pin de entrada o salida) se puede controlar por el usuario en tiempo de ejecución. Los pines GPIO no tienen ningún propósito especial definido, y no se utilizan de forma predeterminada.

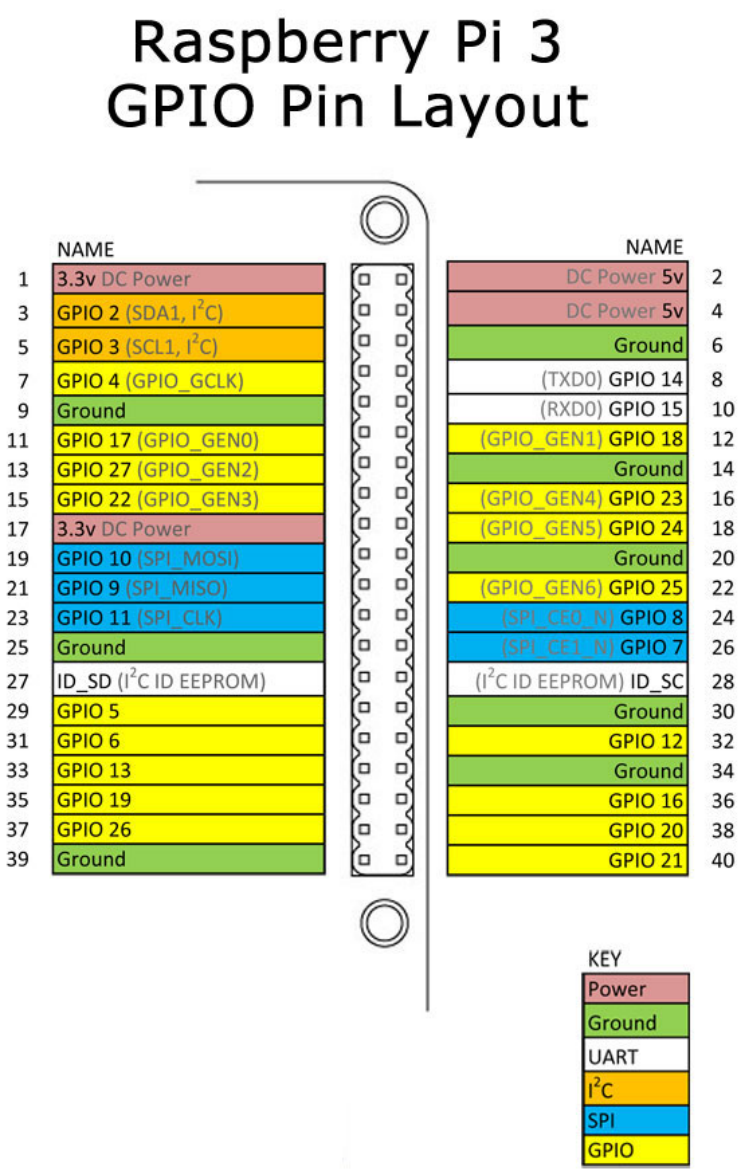


Figura 3-17. Descripción de pines GPIO de Raspberry Pi

En la Figura 3-17 podemos observar la descripción de cada pin del sistema Raspberry Pi 3 y a qué interfaz corresponde dependiendo del color. Cada uno de ellos tiene un propósito específico y por lo tanto habrá que conectar cada módulo donde corresponda. A excepción del módulo de cámara que dispone de una entrada propia para su conexión, el resto de módulos usarán algún pin de la cabecera. En la Tabla 3–4 detallaremos los pines que necesitamos para cada uno de ellos obtenidos a través de la documentación oficial de cada módulo.

Tabla 3–4 Relación de pines usados por cada módulo

Módulo	Pines ocupados	Pines usados	Propósito de los pines
Sense HAT	Todos los pines (1 a 40)	2, 6, 3, 5	Alimentación 5v GND I2C SDA

			I2C SCL
Controlador de motores	Del 1 al 34	2, 6, 29, 31, 32, 33	Alimentación 5v GND GPIO 5,6,12,13(Propósito General usado para PWM)
Módulo GPS	1, 6, 8, 10	1, 6, 8, 10	Alimentación 3.3v GND UART TX UART RX

Uno de los principales problemas del sistema es el hecho de que varios de los módulos usados ocupan pines que no necesitan realmente, pero han sido diseñados así. La mayoría de ellos se conecta al sistema Raspberry Pi mediante la cabecera de pines y solo el Sense HAT la ocupa entera. Por lo tanto, para llevar a cabo el conexionado completo del sistema tendremos que hacer uso de cables que conecten solo los pines que son realmente utilizados por cada módulo. A partir de la Tabla 3–4 obtenemos la Figura 3-18. Corresponde a un diagrama esquemático obtenido mediante un programa llamado Fritzing. Este programa es open-source y muy útil para este tipo de situaciones. En [10] tenemos acceso a la web oficial con toda la información a este programa.

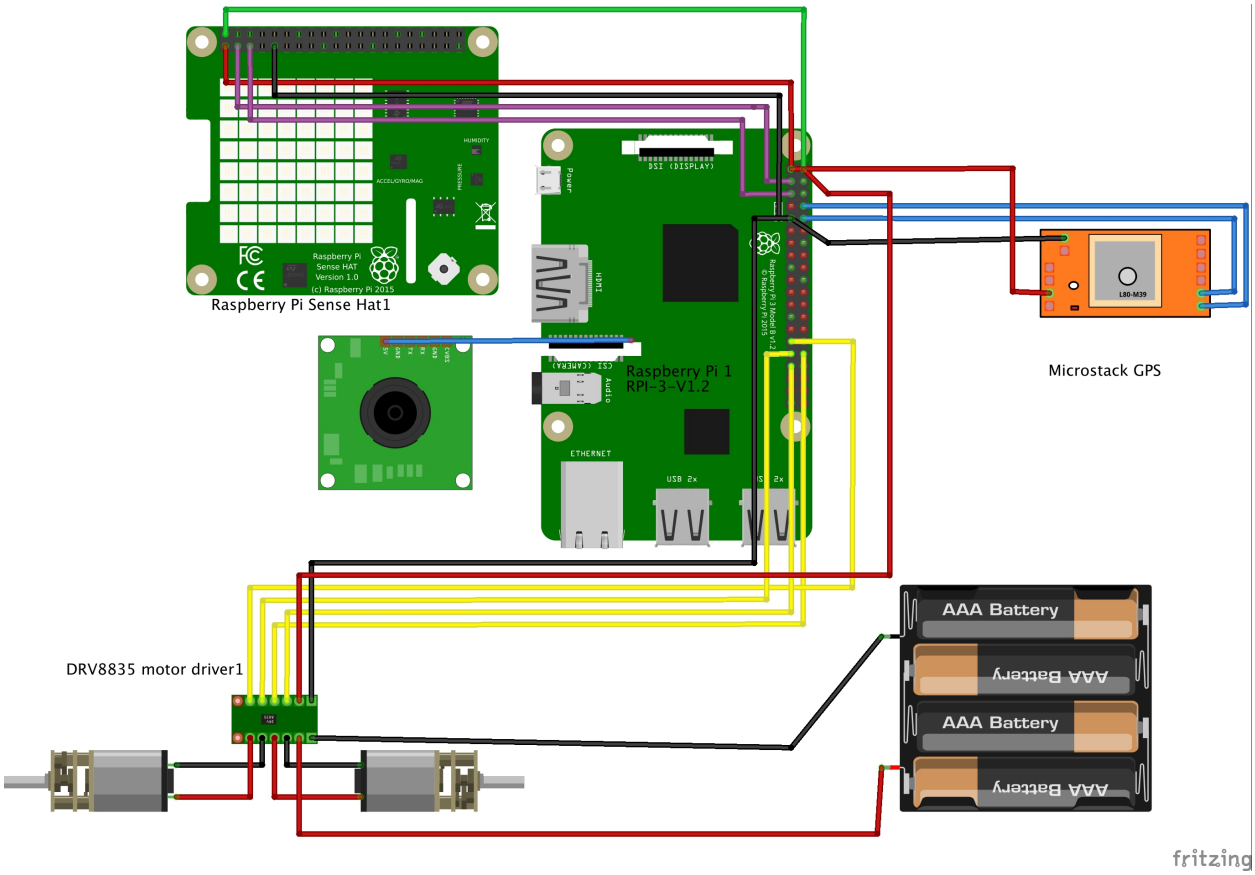


Figura 3-18. Esquemático del sistema completo

4 SISTEMA DE CONTROL REMOTO

Nuestro sistema de control remoto consistirá en una aplicación Android sencilla que hará uso del Bluetooth para enviar las órdenes de control a la Raspberry Pi emulando una comunicación serie mediante el protocolo RFCOMM. Este es el punto más personal del proyecto, ya que hemos desarrollado la app Android desde cero además de elegir la forma en la que se comunicarán entre sí cliente y servidor dado el amplísimo abanico de opciones disponibles en cuanto a tecnologías de comunicaciones inalámbricas (LAN, Bluetooth, NFC, SSH, etc.) y protocolos usados en estas tecnologías.

4.1 Protocolos de comunicación

4.1.1 Bluetooth

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz. Fue diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en transceptores de bajo costo. Del mismo modo que WiFi, Bluetooth utiliza la técnica FHSS (Frecuency Hopping Spread Spectrum, en español Espectro ensanchado por saltos de frecuencia). Esta técnica consiste en dividir la banda de frecuencia de desde los 2.402 a los 2.480 GHz en 79 canales de frecuencia o saltos de 1 MHz de ancho cada uno y transmitir la señal utilizando una secuencia de canales conocida por emisor y receptor. La capacidad de dar hasta 1600 saltos de frecuencia por segundo dota al estándar Bluetooth de gran robustez y seguridad ya que evita interferencias con otras señales radio.

Se basa en el modo de operación maestro/esclavo. El término "piconet" se utiliza para hacer referencia a la red formada por un dispositivo y todos los dispositivos que se encuentran dentro de su rango. Pueden coexistir hasta 10 piconets dentro de un solo área de cobertura. Un dispositivo maestro se puede conectar simultáneamente con hasta 7 dispositivos esclavos activos. Esto podría llegar a ser muy útil en nuestro proyecto. Por ejemplo si quisiéramos desplegar varios robots en cierta zona y controlarlos todos desde el mismo control remoto . En realidad, en un momento determinado, el dispositivo maestro sólo puede conectarse con un solo esclavo al mismo tiempo. Por lo tanto, rápidamente cambia de esclavos para que parezca que se está conectando simultáneamente con todos los dispositivos esclavos. Bluetooth permite que dos piconets puedan conectarse entre sí para formar una red más amplia, denominada "scatternet", al utilizar ciertos dispositivos que actúan como puente entre las dos piconets. Aplicado al ejemplo expuesto anteriormente, podríamos dividir cierta zona en zonas mas pequeñas o con distintas características y cada una de ellas tendría un despliegue de robots controlados por un “piconet”, y todos ellos conectados entre sí formando dicho “scatternet”.

El uso de Bluetooth es adecuado en una situación de dos o más dispositivos en un área reducida (o hasta 100 metros si hacemos uso de la versión de largo alcance) sin grandes necesidades de ancho de banda. Bluetooth además ofrece perfiles de servicio más detallados como por ejemplo un perfil para la emulación de línea serie que será el que usemos en este proyecto como veremos adelante. Por esto y por la notable diferencia de consumo respecto a otras tecnologías inalámbricas es por lo que hemos escogido Bluetooth como base de las comunicaciones entre el móvil y el robot.

4.1.2 El protocolo RFCOMM de Bluetooth

RFComm es la abreviatura del término inglés Radio Frequency Communication (Comunicación por radio frecuencia). El protocolo RFCOMM es un conjunto simple de protocolos de transporte, que se apoya sobre el protocolo L2CAP y que proporciona sesenta conexiones simultáneas para dispositivos bluetooth emulando puertos serie RS-232. El protocolo está basado en el estándar ETSI TS 07.10. Este protocolo, como se ha dicho, es a menudo denominado “emulador de puertos serie” ya que el puerto serie de Bluetooth está basado

en este protocolo y por lo tanto, las aplicaciones que usan comunicaciones por puerto serie pueden ser fácilmente trasladadas al uso de este protocolo.

Un camino de comunicación involucra siempre a dos aplicaciones que se ejecutan en dos dispositivos distintos (los extremos de la comunicación) que en nuestro caso serán la aplicación móvil y el robot. Entre ellos existirá un segmento que los comunica. RFCOMM pretende cubrir aquellas aplicaciones que utilizan los puertos serie de las máquinas donde se ejecutan. El segmento de comunicación será un enlace Bluetooth desde un dispositivo al otro (conexión directa).

Debido a la sencillez de uso y manejo de una comunicación por puertos serie, que no es mas que una interfaz de comunicaciones de datos digitales donde la información es transmitida bit a bit y debido también a que nuestros requerimientos se adecúan perfectamente a este tipo de comunicación puesto que lo único que vamos a transmitir entre mando y robot serán caracteres y palabras que, correctamente interpretadas por el robot provocarán el comportamiento de éste, se tomó la decisión de usar este protocolo en el proyecto. Veremos más de este protocolo además de su implementación real en los siguientes apartados.

4.2 Control Remoto

Ahora que ya sabemos en qué protocolos nos basaremos para implementar la comunicación entre el mando y el robot, vamos a explicar la solución adoptada en un lado de la comunicación. Es decir, qué órdenes de control usará el control remoto para ordenar un comportamiento al robot. Podriamos haber adoptado múltiples opciones para dar solución a este apartado. Por simplicidad, el principio de la comunicación entre el mando y el robot consistirá en un intercambio de palabras o cadenas de caracteres ya que el manejo de éstas es bien conocido en programación, además de ser la forma más intuitiva debido a que es forma natural de comunicación humana. Por lo tenato, la aplicación móvil que hará de control remoto transmitirá una palabra o cadena de caracteres distinta en función del botón pulsado por el usuario. Será función del robot interpretar las cadenas recibidas y asociarlas a uno u otro comportamiento.

4.2.1 Órdenes de control de movimientos

Las cadenas transmitidas orientadas al movimiento del robot serán:

Tabla 4–1 Cadenas de caracteres enviadas para el control de los motores

Cadena de texto enviada	Comportamiento
“avanza”	El robot moverá ambas ruedas hacia delante.
“retrocede”	El robot moverá ambas ruedas hacia atrás.
“para”	El robot parará ambas ruedas.
“izq”	El robot moverá la rueda izquierda consiguiendo un giro en la dirección.
“der”	El robot moverá la rueda derecha consiguiendo un giro en la dirección.

4.2.2 Órdenes de control para recopilación de datos

Lo mismo ocurre con las órdenes para controlar los sensores. Las cadenas orientadas a la captación de datos del entorno serán.

Tabla 4–2 Cadenas de caracteres enviadas para la captación de datos

Cadena de texto enviada	Comportamiento
"foto"	El robot moverá capturará una instantánea.
"tem"	El robot medirá la temperatura ambiente en grados centígrados.
"hum"	El robot medirá el porcentaje de humedad relativa del ambiente.
"pos"	El robot obtendrá su posición GPS.
"bd"	El robot almacenará todos los datos obtenidos anteriormente en un tabla en una base de datos.

En próximos apartados haremos hincapié en el uso de los sensores que hacen posible estos comportamientos.

4.2.3 Aplicación para el control remoto

Como se ha comentado en la introducción a este apartado, el control remoto consistirá en una aplicación Android desarrollada usando la herramienta App Inventor. App Inventor parte de una idea conjunta del Instituto Tecnológico de Massachusetts y de un equipo de Google Education. Se trata de una herramienta web de desarrollo para iniciarse en el mundo de la programación. Con él pueden hacerse aplicaciones muy simples, y también muy elaboradas, que se ejecutarán en los dispositivos móviles con sistema operativo Android.

En este entorno el usuario puede, de forma visual, muy intuitiva y a partir de un conjunto de herramientas básicas, ir enlazando una serie de bloques para crear la aplicación como si de un puzzle se tratase. Consiste en un lenguaje orientado a objetos. La diferencia con cualquier otro entorno de desarrollo radica en que con App Inventor no hay que escribir el código línea a línea sino que existen bloques (piezas de puzzle) que tendremos que ir uniendo. Las aplicaciones creadas con App Inventor están limitadas por su simplicidad, aunque permiten cubrir un gran número de necesidades básicas en un dispositivo móvil y más que suficientes para nuestro proyecto, que no se centra en la aplicación en sí, si no en la funcionalidad de ésta.

No vamos a hacer una guía o tutorial de cómo hacer uso de esta herramienta puesto que podemos encontrar todo tipo de documentación mucho mas detallada y completa que cualquiera que yo pudiera escribir aquí. En este documento haremos referencia [13] a un turorial obtenido de internet que ha resultado ser muy útil en mi inicio en esta herramienta y por consiguiente, en el desarrollo de la aplicación. El sistema es gratuito y se puede descargar fácilmente de la web. Además dispone de una aplicación web que resulta ser incluso mejor opción.

Sí vamos a hacer una introducción básica a los dos bloques o apartados principales de la herramienta para, al menos, poder apoyarnos en ella al describir la aplicación desarrollada para el proyecto. Comenzamos por el "Diseñador". Observamos en la Figura 4-1. A la izquierda están los objetos (botones, imágenes, dibujos, etc.) que vamos a usar para diseñar la pantalla de nuestra. Equivale a la paleta de un programa de pintura. La pantalla del centro es el Visor, representa la pantalla del móvil y sirve para diseñar el aspecto de la aplicación. A la derecha del visor tenemos los componentes que hemos ido añadiendo al Visor desde la Paleta y más a la

derecha un menú de propiedades de dichos componentes.

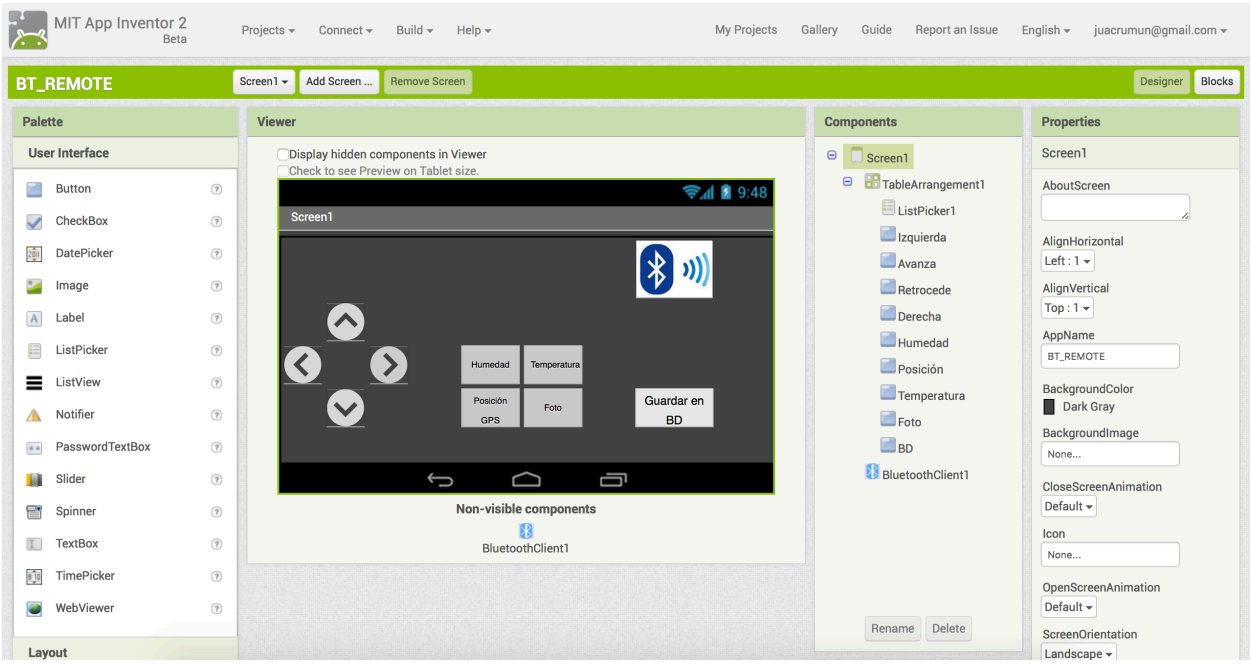


Figura 4-1. Diseñador de la herramienta App Inventor

Como podemos ver en la Figura 4-1, para nuestra aplicación hemos usado:

- Un elemento "TableArrangement" que servirá para ubicar los elementos visualmente con la disposición de celdas de una matriz.
- Varios botones, cuyas funciones serán dirigir el robot, que este tome las distintas medidas del entorno y que lo guarde dichos datos en una base de datos.
- Un "ListPicker" que no es mas que un bloque para seleccionar un elemento de una lista y que será usado para seleccionar y conectarnos a un dispositivo Bluetooth de entre todos los disponibles.
- Un cliente Bluetooth. Este bloque es de los llamados "No visibles" puesto que no aparecerán en el Visor.

Como hemos dicho, no vamos a entrar en más detalles, pero por supuesto hay infinidad de opciones y propiedades configurables para cada componente.

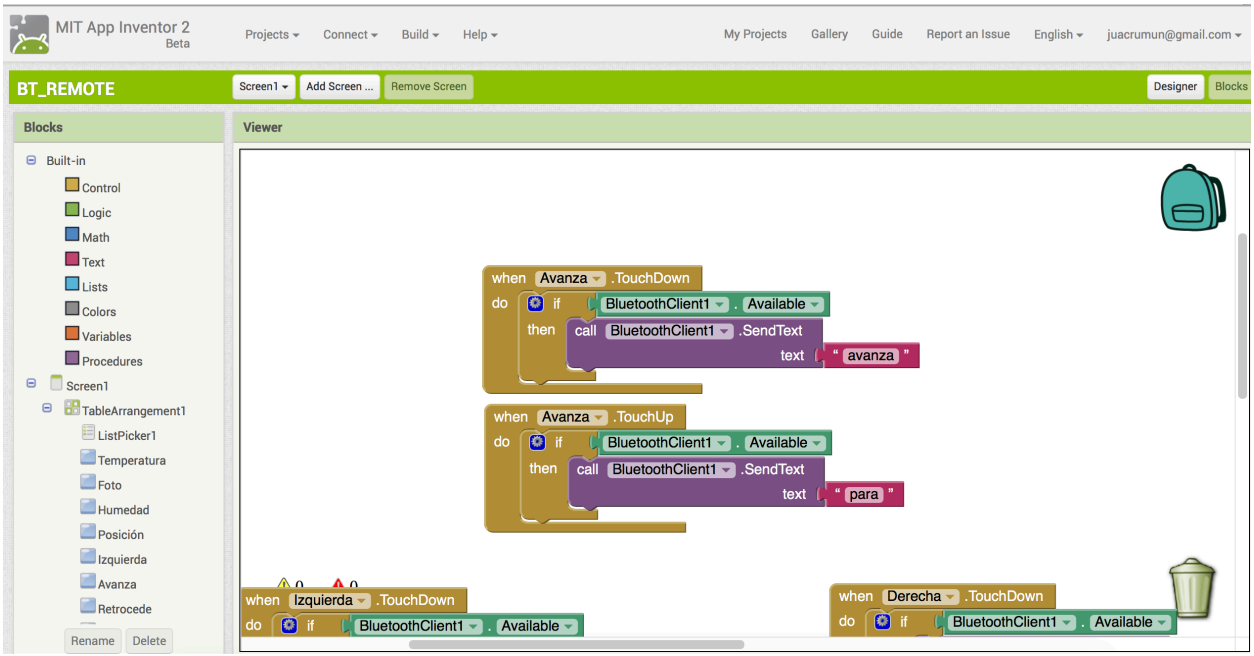


Figura 4-2. Bloques de los botones de dirección en la herramienta App Inventor

El siguiente apartado es “Bloques”. Aquí es donde se programa el comportamiento de cada uno de los componentes añadidos en el Diseñador. Como hemos dicho anteriormente, este proceso de programación consistirá en ir uniendo piezas con distintas funcionalidades. Como vemos en la Figura 4-2, a la izquierda podemos seleccionar distintos tipos de bloques o piezas, desde bloques equivalentes a las típicas sentencias de programación (sentencias de control como if-else, sentencias lógicas, variables, etc.) hasta bloques con funcionalidades personalizas para cada tipo de componente (por ejemplo, qué hace un botón cuando se presiona) . Los bloques que vayamos seleccionando irán apareciendo en el espacio, inicialmente en blanco, de la derecha. La programación de cada componente se hará uniendo estas piezas, dontándolas de sentido al complementarse unas con otras.

Para los botones de dirección usaremos los bloques “TouchDown” y “TouchUp”. Estos bloques indicarán el comportamiento de la aplicación cuando pulsemos y despulsemos los botones respectivamente. Es decir, transmitirán las cadenas de caracteres antes mencionadas. Podemos observar en la Figura 4-2 que el nombre del botón es “avanza”, por lo que puede intuirse que este es el botón que hará que el robot se desplace hacia delante mediante la transmisión de dicha palabra. Este diseño se repite para los cuatro botones de dirección, cuya única diferencia será el texto transmitido por el cliente Bluetooth.

Cabe destacar el componente “BluetoothClient”, el cual se encargará de usar el modulo Bluetooth del móvil para transmitir un texto u otro dependiendo del botón pulsado. Es decir, este componente será el que implemente la comunicación física. Acoplado a éste podemos observar el uso del componente de color verde “BluetoothClient.Available”, que como hemos mencionado antes, es un bloque con funcionalidad personalizada para el componente “BluetoothClient”. Este bloque se encargará de comprobar que hay un dispositivo Bluetooth emparejado y conectado correctamente. El bloque morado es “BluetoothClient.SendText” y es uno de los muchos bloques de procedimientos que pueden añadirse a ciertos componentes, como es en este caso al cliente Bluetooth. Este procedimiento será el encargado de transimitir las cadenas de caracteres. Como podemos ver a la izquierda de la Figura 4-2, estos bloques color cereza sirven para manejar multiples opciones de manejo de textos.

Los botones utilizados para recoger datos del entorno y almacenarlos también tienen una programación similar a los botones de dirección. Las diferencias serán que no nos preocuparemos por lo que ocurra cuando despulsemos el botón, ya que la medida se hará cuando se pulse y que los textos transmitidos harán referencia a la funcionalidad de los botones y, por ende, a los sensores asociados a estos botones.

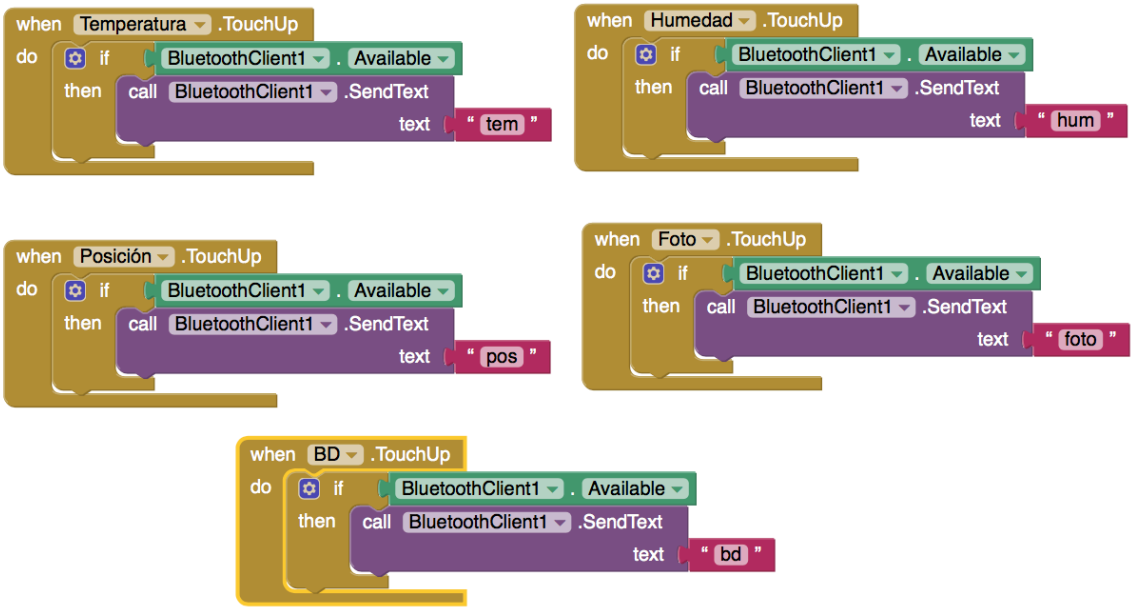


Figura 4-3. Bloques de los botones de recogida de datos en la herramienta App Inventor

Por último, vemos en la Figura 4-4 la programación del antes mencionado "ListPicker". Hemos usado los bloques "BeforePicking" y "AfterPicking" que servirán para decirle a la aplicación qué debe hacer antes y después de elegir una de las opciones deplegadas por el ListPicker. Antes de elegir una opción, el ListPicker nos mostrará las direcciones y los nombres de los posibles dispositivos Bluetooth detectados y después de elegir una opción, el cliente Bluetooth se conectará al dispositivo con nombre y dirección indicado. Observando la Figura 4-4, podemos comprobar lo explicado. No es necesario tener conocimientos avanzados en programación para intuir qué está ocurriendo, debido a la sencillez de uso de esta herramienta.

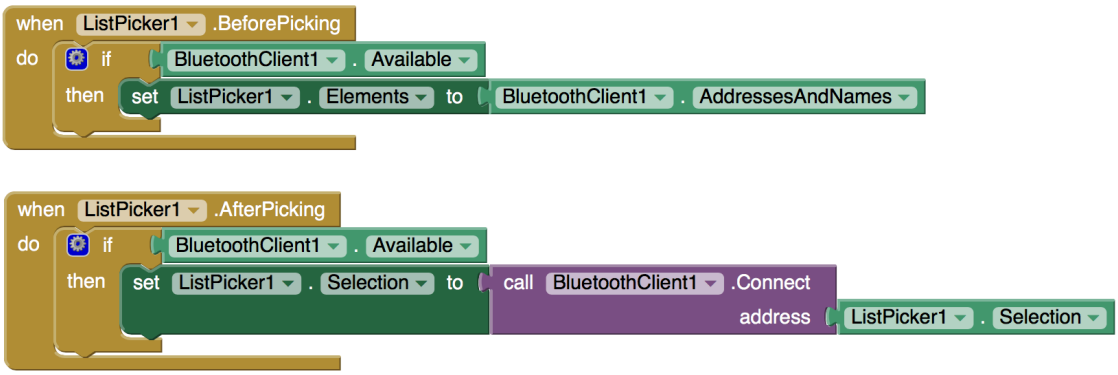


Figura 4-4. Bloques del elemento ListPicker en la herramienta App Inventor

Aunque en este documento se refleja el resultado, como toda tarea de programación ha requerido mucho tiempo para darla por completada a base de prueba y error. El último paso es compilar la aplicación e instalarla en el dispositivo móvil. Con esto ya tenemos una aplicación completamente funcional que actuará como control remoto de nuestro robot.

5 SISTEMA DE RECOPILACIÓN DE DATOS DE CONTEXTO

En este capítulo vamos a profundizar en cómo el robot será capaz de recoger datos del entorno. Se hará una breve descripción del entorno de trabajo para Python en el sistema operativo Raspbian. Hablaremos del código implementado para que el sistema reconozca la orden recibida del control remoto y actúe en consecuencia. Trataremos las librerías, funciones, métodos y variables necesarias para que cada módulo conectado a nuestro sistema Raspberry Pi devuelva un dato del entorno y lo almacene en una base de datos MySQL, además de cualquier otro detalle que se considere importante para el entendimiento del sistema completo.

5.1 Sistema Operativo Raspbian y lenguaje Python

Raspbian se trata de una distribución libre del sistema operativo GNU/Linux basado en Debian adaptada y optimizada para Raspberry Pi. Aunque existen muchas otras distribuciones para Raspberry Pi, Raspbian sigue siendo la más completa, estable y que mejor aprovecha el rendimiento del dispositivo. Por ello, los desarrolladores de esta distribución siguen actualizando su sistema para poder sacar el mayor provecho del hardware. La distribución usa LXDE como escritorio. Además contiene herramientas de desarrollo como un IDLE para el lenguaje de programación Python y diferentes aplicaciones destinadas a la educación como Wolfram o Mathematica, entre otras muchas cosas. Destaca también el menú "raspi-config" que permite configurar el sistema operativo sin tener que modificar archivos de configuración manualmente. Entre sus funciones, permite expandir la partición root para que ocupe toda la memoria, configurar el teclado, etc.

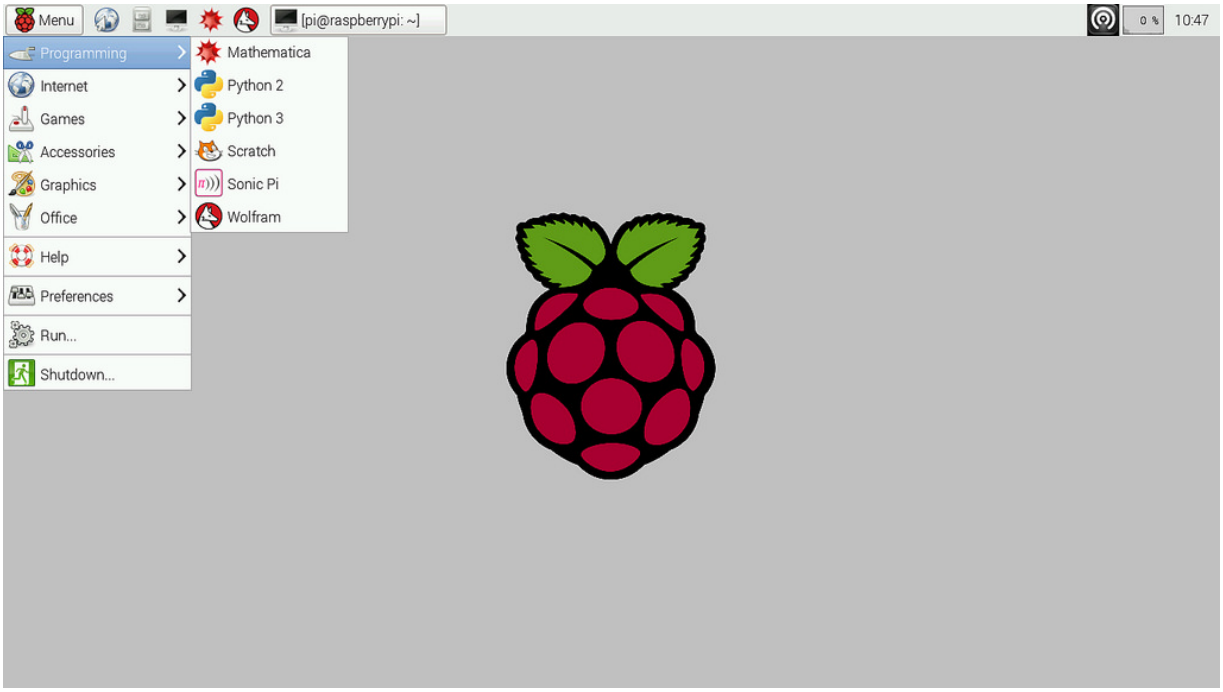


Figura 5-1. Vista de escritorio de Raspbian Jessie

En nuestro sistema Raspberry tendremos instalada la última versión de Raspbian llamada “Jessie”. En la Figura 5-1 podemos ver el escritorio dicha versión y el menú desplegado con algunas de sus aplicaciones de programación. La principal diferencia de esta nueva distribución respecto a las anteriores es que el sistema operativo carga por defecto la interfaz gráfica y el escritorio en vez de cargar el terminal y tener que cargar la interfaz gráfica manualmente. De esta forma los usuarios menos experimentados podrán empezar a utilizar el sistema a través de la interfaz gráfica de Raspbian de forma más intuitiva sin tener que ejecutar manualmente ningún comando en el terminal. Esta opción se puede cambiar, pudiendo elegir que el sistema sólo cargue el terminal, ahorrando tiempo y recursos. Otros cambios menores de esta versión es que se han retocado algunos ajustes del escritorio, así como la apariencia de los menús que ahora tienen un aspecto más moderno. Estos cambios de apariencia también acarrean mejoras de rendimiento. Ahora, aplicaciones como LibreOffice funcionan mucho mejor. Se han incluido también nuevas opciones de configuración para el sistema operativo, especialmente la GUI o interfaz gráfica para raspi-config, que permite configurar la mayor parte de las preferencias del sistema operativo sin necesidad de acceder a la versión de terminal del mismo.

No vamos a entrar en más detalles en cuanto al sistema operativo. Al fin y al cabo estamos acostumbrados al uso de ellos a diario y resulta ser una tarea sencilla e intuitiva. No será este la excepción.

Figura 5-2. Icono lenguaje Python

En cuanto a Python, es un lenguaje de programación interpretado. Se escribe el código e internamente se traduce y ejecuta sobre la marcha. La diferencia con algunos lenguajes, como C y Java radica en que éstos últimos se compilan. Es decir, se escribe el programa, se genera un archivo de compilación escrito en código de máquina y se ejecuta el archivo generado. Este lenguaje hace hincapié en una sintaxis que favorezca un un código legible mediante indentación. Es decir, no necesita abrir y cerrar llaves para separar bloques. Es un lengunaje de programación multiparadigma, ya que soporta programación orientada a objetos, programación imperativa y programación funcional. Además usa tipado dinámico y es multiplataforma. Otra característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Como ocurre con cualquier otro lenguaje de programación, la forma más común de escribir un programa consiste en el desarrollo de un archivo de texto plano con cierta extensión, en el caso de Python es “”.py”, en el que se recogen todas las órdenes que queremos ejecutar incluyendo el uso de los elementos de la siguiente lista:

- Comentarios
- Variables
- Tipos de datos
- Listas y Tuplas
- Diccionarios
- Conjuntos
- Funciones
- Clases
- Sentencias condicionales
- Bucles

- Módulos

Existen dos versiones: Python 2 y Python 3. Aunque Python 3 es el futuro, todavía se enseña y se usa Python 2. No podemos hacer una comparación entre ellos debido a la gran cantidad de diferencias que existen entre ellos. Quedará refernciado un enlace [15] a un artículo muy útil en el que se explican las principales diferencias entre ellos. Para este proyecto se escogió originalmente Python 2.7 debido a la estandarización de esta versión en el actualidad. Sin embargo, como se explicó en el apartado del módulo GPS, debido a la desactualización de las librerías de este módulo nos vimos obligados a cambiar a Python 3.

5.2 Software desarrollado y módulos embarcados

Este apartado, sin lugar a dudas, engloba la mayor parte de este proyecto. A modo de resumen vamos a mencionar las tareas ya realizadas hasta este punto para luego, tras este apartado, dar por concluida la explicación del proyecto completo. Hasta ahora se ha realizado el análisis de los elementos físicos o hardware que compondrán el vehículo y su correcto conexionado, el análisis de cómo se ha implementado una aplicación móvil para controlar remotamente el vehículo robot, y en el principio de este capítulo se ha introducido el contexto sobre el que se programará el código que se desarrolla a continuación y que irá implementado en el sistema Raspberry a modo de "cerebro" del robot. Este código se configurará para que se ejecute automáticamente al iniciar el sistema y se ejecutará en segundo plano o "modo demonio". Este apartado se compondrá de un subapartado en el que explicaremos, primero, la estructura general del código, seguido de otros subapartados en los que se detallarán las librerías importadas, variables, métodos y funciones usadas y estructura del código de cada módulo presente en el sistema.

5.2.1 Estructura general del código

Este código no se compondrá de varios fichero en el que se desarrollen distintas funciones, ya que pensamos que dadas las características del lenguaje Python que hacen de él un lenguaje muy legible, desarollaríamos todo el código en el mismo archivo.

Al principio importamos las librerías necesarias para cada módulo y creamos las variables e instancias necesarias. Esto se desarrollará más detenidamente para cada módulo en su correspondiente subapartado. A continuación creamos las conexiones con el control remoto y la base de datos MySQL para empezar a recibir datos. Entrará en un buble infinito en el que dependiendo del dato recibido, el robot tendrá uno u otro comportamiento. Solo se saldrá del bucle en caso de que algún módulo lance una excepción o error, o se pulse la combinación de interrupción por teclado (ctrl+c). Si esto ocurre, se cierran las conexiones y finaliza la ejecución. Podemos comprobar esta explicación con el diagrama de flujo de la estructura de dicho en la Figura 5-3.

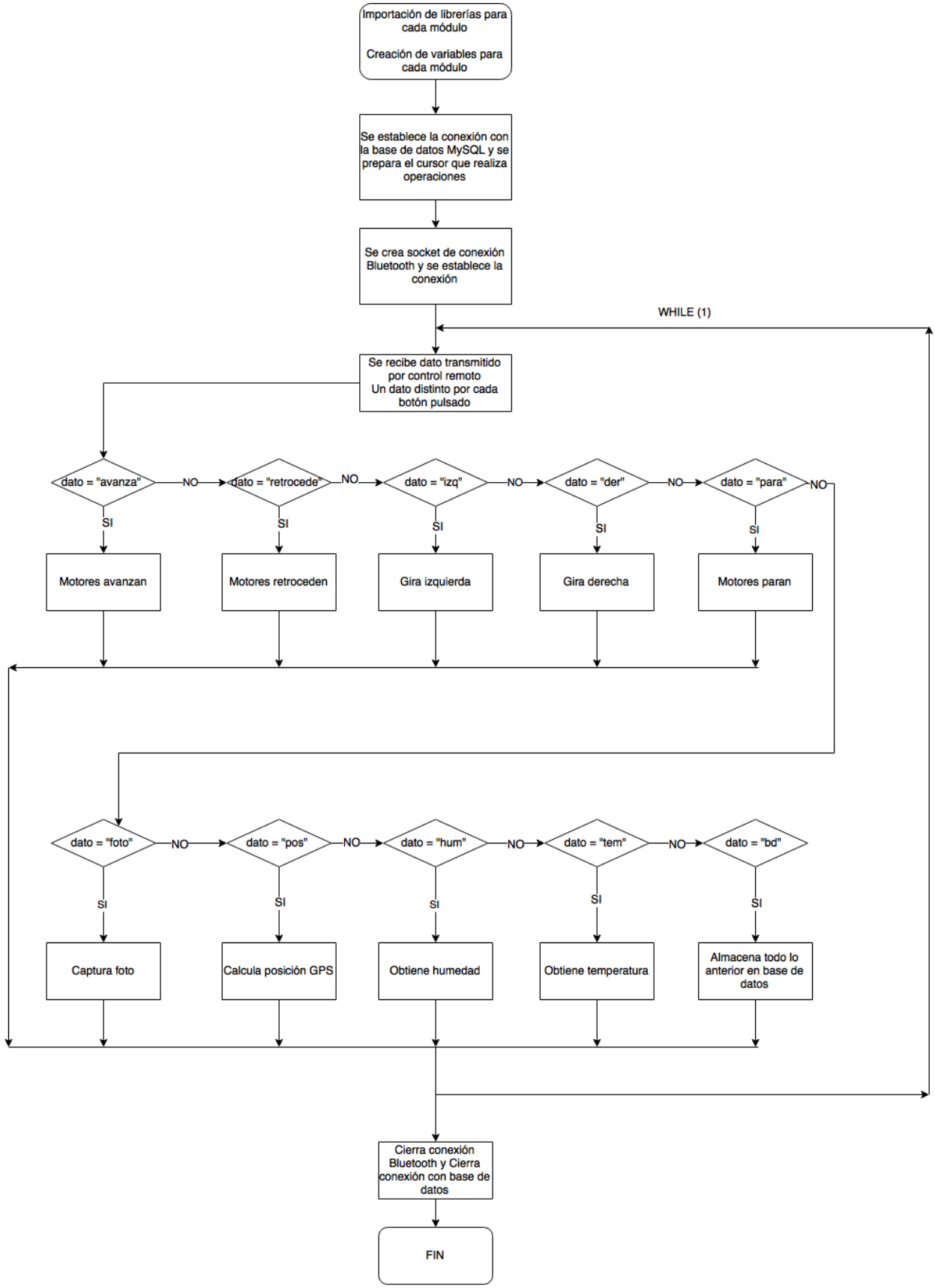


Figura 5-3. Diagrama de flujo general del código

5.2.2 Manejo de comunicación Bluetooth

Para dicha crear dicha comunicación, será necesario instalar el paquete “PyBluez”. Esto se hace desde el terminal escribiendo el comando “sudo apt-get install python-bluez”. Este paquete nos dará acceso a la librería “bluetooth” que contendrá los métodos y funciones para manejar una comunición mediante el protocolo RFComm. En la Figura 5-4 vemos un sencillo diagrama de flujo en el que se explica el funcionamiento de dicha conexón. En color azul se describirán las líneas de código más relevantes.

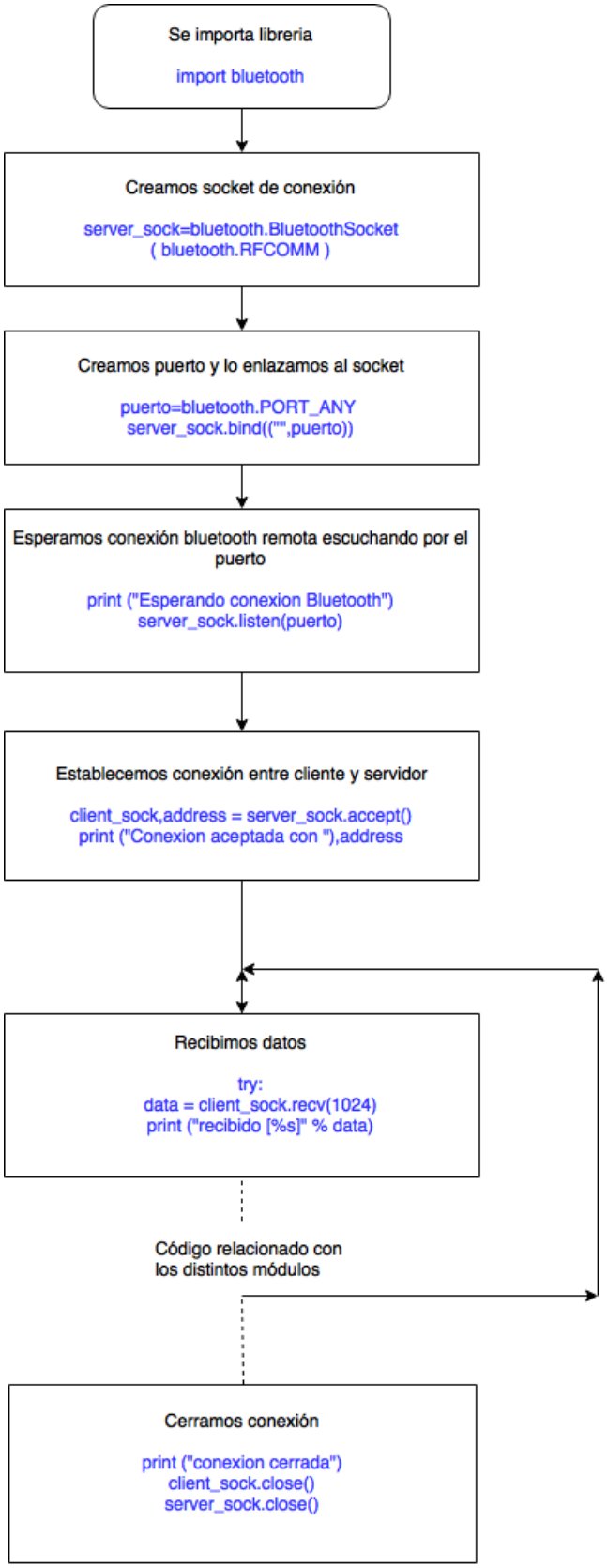


Figura 5-4. Diagrama de flujo de la comunicación Bluetooth

5.2.3 Control de motores

Dependiendo del botón de dirección pulsado en el mando los motores de las ruedas deberán tener uno u otro comportamiento.

- Avanzar: Los dos motores girarán en sentido normal a máxima velocidad.
- Retrocededer: Los dos motores girarán en sentido inverso a máxima velocidad.
- Girar izquierda: El motor izquierdo retrocederá a media velocidad y el motor derecho avanzará a media velocidad.
- Girar derecha: El motor derecho retrocederá a media velocidad y el motor izquierdo avanzará a media velocidad.
- Parar: Ninguno de los dos motores se moverá.

Para el uso del controlador de motores DRV8835, la empresa facilita un paquete descargable mediante la plataforma Github. Podemos ver toda la información rerferente a estos paquetes en [8]. Para instalar este paquete seguimos los pasos explicados en el enlace.

Introducimos en el terminal los comandos “sudo apt-get install python-dev python-pip” y “sudo pip install wiringpi”. Con esto instalamos el famoso paquete WiringPi. Finalmente para descargar e instalar la librería pololu_drv8835_rpi, hacemos “git clone https://github.com/pololu/drv8835-motor-driver-rpi.git” , “cd drv8835-motor-driver-rpi” y “sudo python setup.py install”. Con esto ya tendremos acceso a la librería. Esta librería utiliza PWM ultrasónico de 20 kHz para accionar los motores. Las velocidades del motor se representan como números entre -480 y 480, ambos inclusive. Una velocidad de 0 corresponde al motor parado, las velocidades positivas corresponden a la corriente que fluye de M1A / M2A a M1B / M2B, mientras que las velocidades negativas corresponden a la corriente que fluye en la otra dirección. Como en todos los apartados, podemos ver un diagrama de flujo de la parte del código correspondiente a la gestión de los motores en la Figura 5-5.

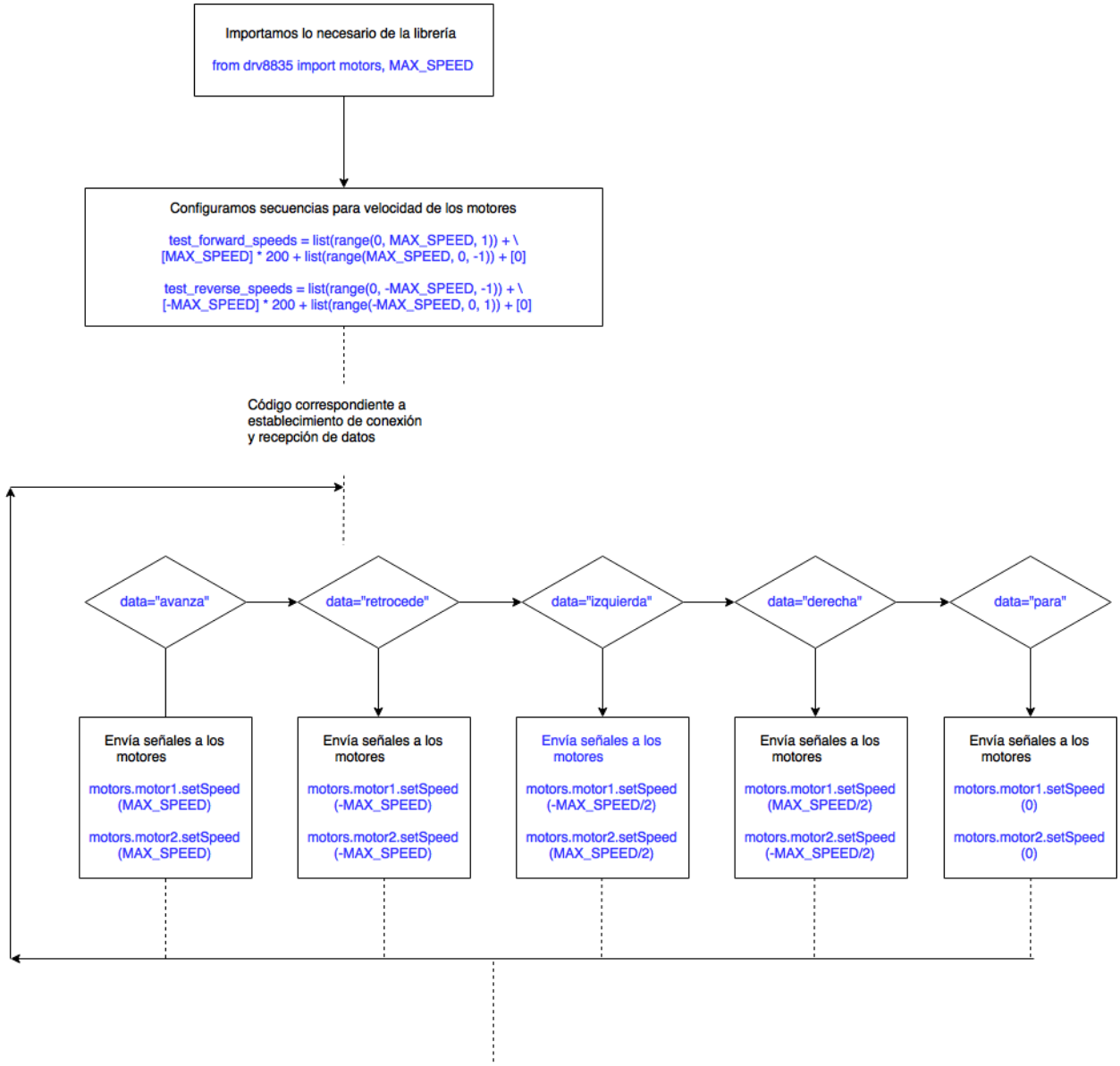


Figura 5-5. Diagrama de flujo de la gestión de los motores

5.2.4 Captura de foto

Otra de las funciones del robot será la de tomar una instántanea haciendo uso del módulo de cámara. Al ser este módulo oficial para Raspberry Pi, en la página oficial [7] ofrecen toda la documentación respecto a la librería python-picam. Para usar la cámara, primero de todo tenemos que habilitarla en el menú raspi-config. Hacemos "sudo raspi-config", navegamos por el menú y habilitamos la cámara. Es necesario reiniciar. Esto solo es necesario hacerlo una vez. El siguiente paso es instalar la librería mencionada. Para ello hacemos "sudo apt-get install python-picamera". Tras esto tendremos acceso a métodos, clases, etc. de dicho módulo. En la Figura 5-6 vemos el diagrama de flujo del uso de la cámara. Cabe mencionar que en este proyecto hacemos un uso muy ligero de la cámara para lo que en realidad permite su librería.

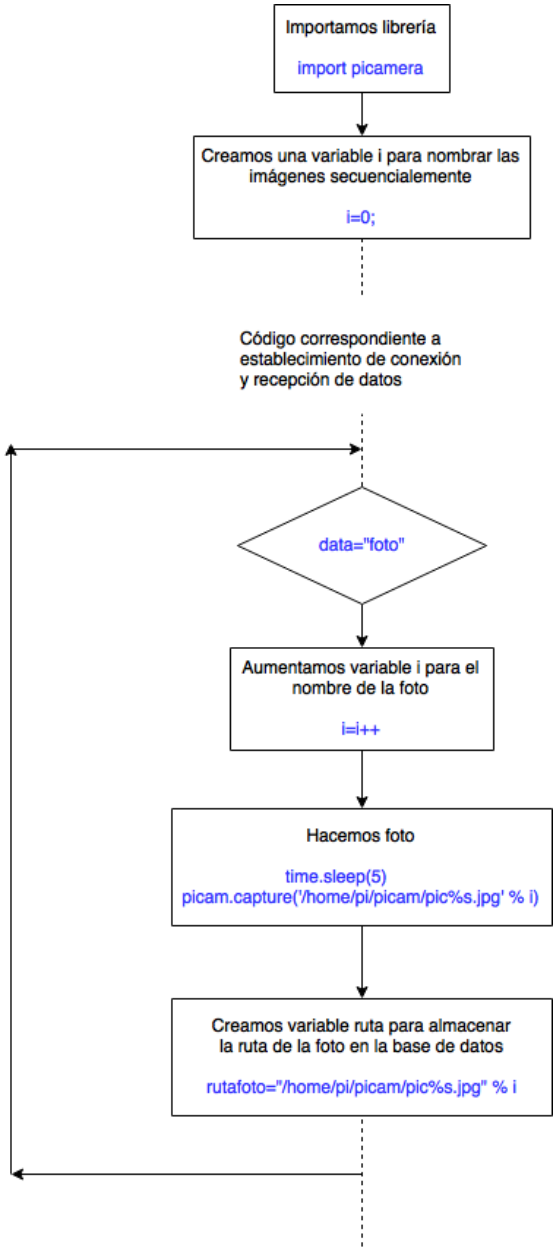


Figura 5-6. Diagrama de flujo de la gestión del módulo de cámara V2

5.2.5 Posición GPS

Como hemos mencionado varias veces a lo largo de este documento, el módulo GPS ha sido causante de múltiples problemas. Por ello, aunque permite obtener una infinidad de datos útiles como pueden ser, la altitud respecto al nivel del mar, el número de satélites, la hora o la fecha, entre otros, hemos limitado su uso al mínimo para evitar más problemas con el uso de los distintos métodos del protocolo NMEA. Incluso la documentación oficial [9] para la instalación de la librería estaba obsoleta. Según la guía oficial, para instalar el paquete necesario habría que hacer "sudo apt-get install python3-microstacknode". Y para instalar herramientas oficiales para el uso de GPS en Raspberry, habría que hacer "sudo apt-get install gpsd gpsd--clients python gps". Nada de esto. La primera orden lanza un error al instalar los paquetes debido a errores en el código de algunos códigos necesarios para usar el GPS. Esta situación nos ha llevado a tener que cambiar a Python3 ya que usando éste no se producían estos errores de código y conseguimos instalar el paquete.

Otro de los problemas ha sido que el hecho de Raspberry Pi 3 tenga Bluetooth incorporado. Para utilizar correctamente el Bluetooth, el /dev/ttyAMA0 ha sido "robado" de la cabecera GPIO. Desafortunadamente

/dev/ttyAMA0 era un puerto serie hardware (uart) de alto rendimiento y por ello se ha modificado para ser usado por Bluetooth. Aquí entra en juego un segundo puerto serie que suele conocerse como "mini uart" en /dev/ttyS0. Este puerto será ahora el encargado de las comunicaciones serie en GPIO. En resumen la situación de puertos queda así.

Tabla 5–1 Relación de puertos serie en Raspberry Pi

Puerto	Función
/dev/ttyAMA0	Bluetooth
/dev/tty/S0	Puerto Serie GPIO

El problema viene cuando todos los códigos usados por el GPS dados por la librería hacen referencia al puerto /dev/ttyAMA0. Por lo tanto no solo hemos tenido que escribir nuestro código, sino que hemos tenido que modificar parte de los códigos dados por la librería y sustituir el puerto /dev/ttyAMA0 por /dev/ttyS0. La razón de que este error haya requerido tanto tiempo no es la dificultad intrínseca que tiene sino la poca información que exite en internet respecto a este tema. Cabe destacar el hilo de un foro [16] que ha sido clave para la corrección de esta situación.

El último problema relacionado con este módulo ha sido la dificultad de trabajar con él debido a la falta de cobertura en interiores. Incluso adquirimos una antena externa para mejorar la cobertura que resultó ser un fracaso. No ha habido solución a este problema más que aprovechar los momentos con cobertura, y como se ha dicho antes, hacer el uso más sencillo posible del GPS. Justo lo necesario para obtener la latitud y la longitud. En la Figura 5-7 se describe el diagrama de flujo de la captura de datos de latitud y longitud mediante el método get_gpgga() del protocolo NMEA.

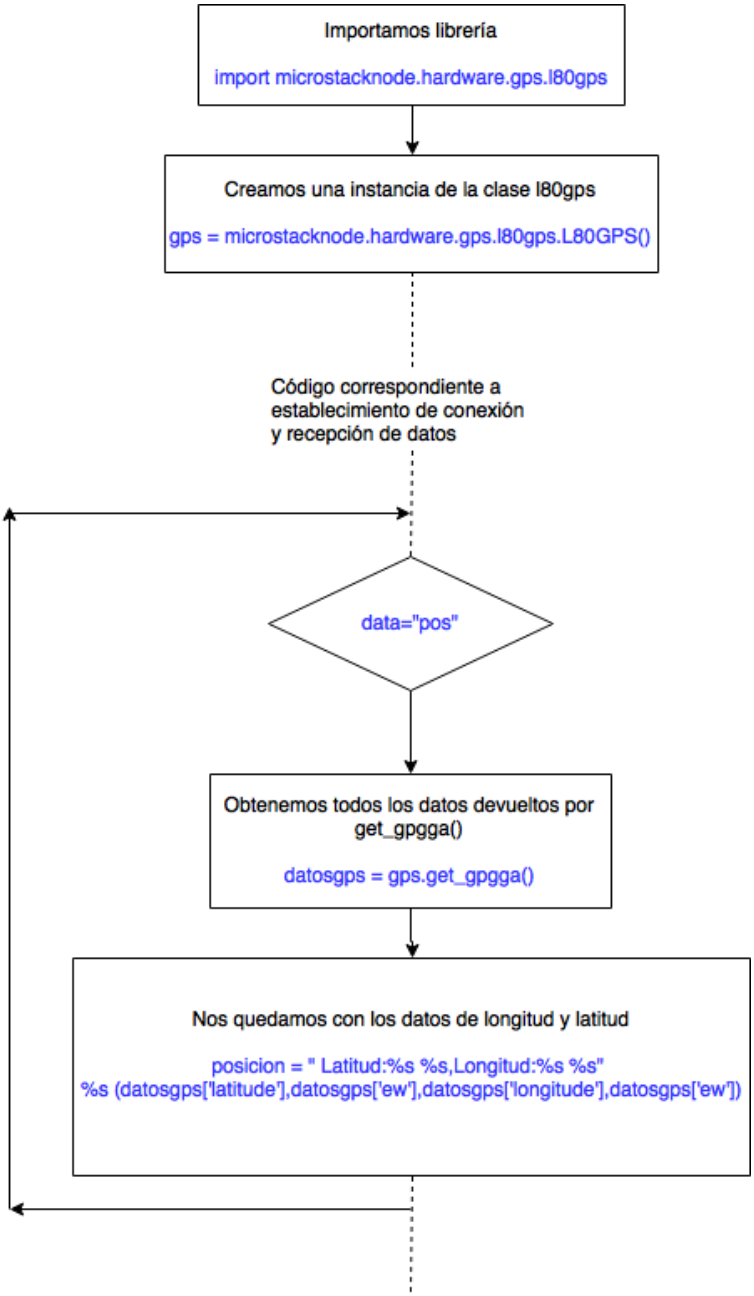


Figura 5-7. Diagrama de flujo de la gestión del módulo GPS

5.2.6 Temperatura y humedad

Trabajar con el dispositivo Sense HAT ha sido una maravilla. Es un dispositivo con muchísimas prestaciones, pero no por ello difícil de usar. Una pena que sólo lo hayamos usado para adquirir temperatura y humedad. Como pasa con el modulo de cámara, este es un accesorio oficial de Raspberry Pi y por ello cuenta con buena documentación en la página web oficial [17] de la fundación.

Introducimos "sudo apt-get install sense-hat" en el terminal para la instalación la librería de uso. Con esto, tendremos acceso a todos los periféricos del Sense HAT, aunque como ya hemos dicho, solo usaremos los sensores de humdad y temperatura. La obtención de estos datos ha sido sencilla y no ha generado ningún problema. Además de estar todo perfectamente guiado en la documentación oficial. En la Figura 5-8vemos el diagrama de flujo para la obtención de estos datos. Hemos diseñado un solo diagramada de flujo debido a que

los dos datos provienen del mismo módulo aunque tengamos que pulsar botones distintos en el control remoto. Lo mismo ocurría con la gestión de los motores.

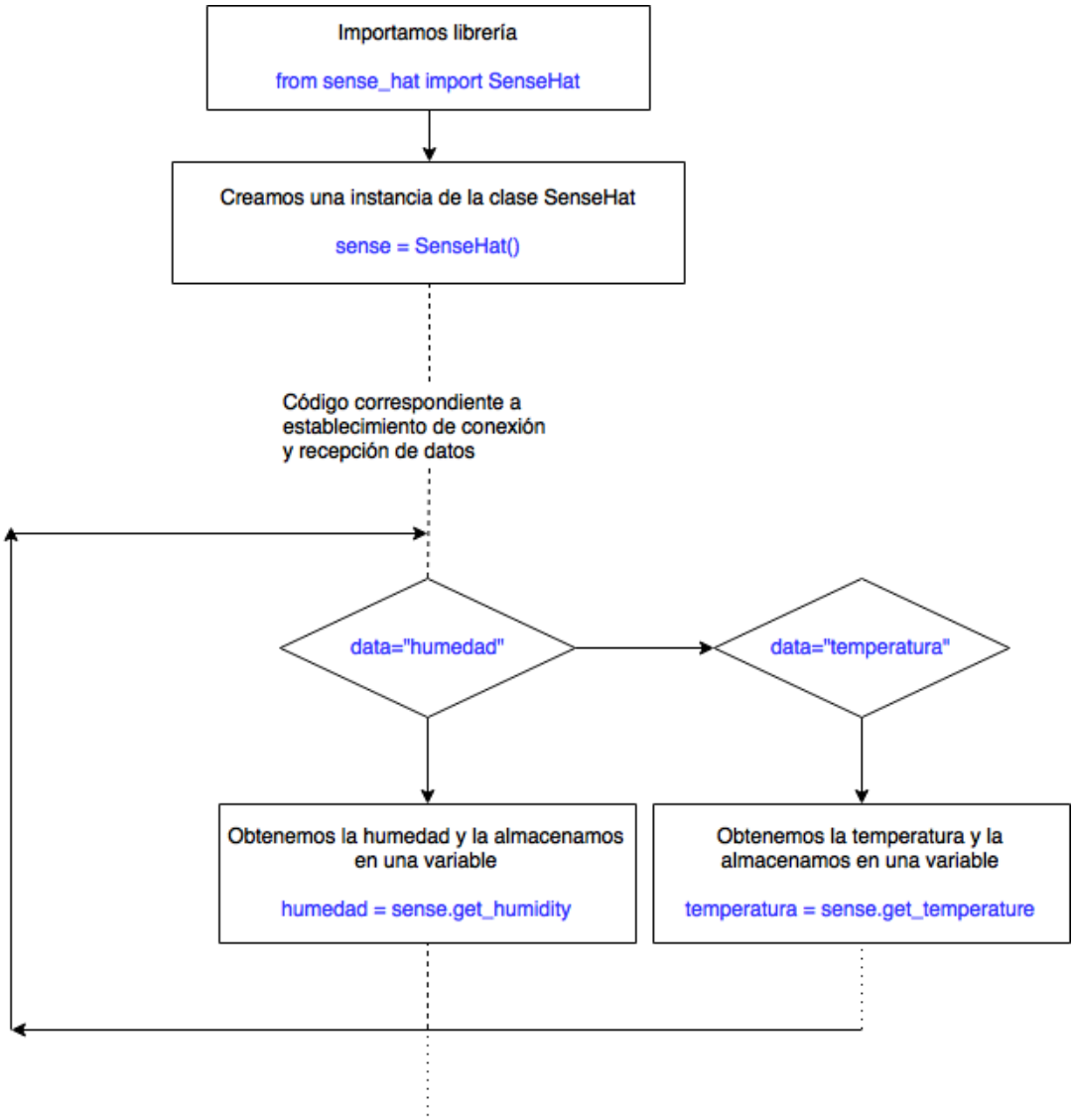


Figura 5-8. Diagrama de flujo de la gestión del Sense HAT

5.2.7 Almacenamiento de datos en base de datos MySQL

Otro de los objetivos descritos en el primer capítulo de este documento era el objetivo de almacenar en una base de datos todos los datos obtenidos por el robot. Por cuestiones de estandarización decidimos usar MySQL. Al principio creamos la base de datos usando MySQLdb para Python 2. Dado que nos vimos obligados a usar Python 3, cambiamos a la interfaz a PyMySQL, que implementa la API v2.0 de la base de datos de Python y contiene una libreria de cliente MySQL para Python. El objetivo de PyMySQL es ser un sustituto de MySQLdb para Python3.

El primer paso para trabajar con esta base de datos es instalar el servidor MySQL en el sistema haciendo “sudo apt-get install mysql-server”. Tras esto, aparecerá una ventana en la que nos pedirá que introduzcamos una contraseña para el usuario “root”. Para este proyecto se ha dejado sin contraseña. Tras esto instalamos el cliente MySQL haciendo “sudo apt-get install mysql-client php5-mysql”

El segundo paso es crear una base de datos. Nos conectamos a nuestra instancia local MySQL haciendo “mysql -uroot -hlocalhost –p”. Se nos abrirá un entorno de uso de MySQL. Aquí tenemos que crear un usuario y una base de datos.

- “CREATE DATABASE dbrobot;” : Esto creará una base de datos llamada dbrobot.
- “CREATE USER 'juacrumun'@'localhost' IDENTIFIED BY "001134”;: Esto creará un usuario llamado juacrumun con contraseña “001134”.
- “GRANT ALL PRIVILEGES ON juacrumun.* TO ‘juacrumun`@'localhost';” : Esto dará permisos al usuario.

Una vez creada la base de datos tenemos que instalar la librería de uso MySQL para Python. Se hace como “pip install pymysql”. Con esto ya podremos trabajar con una base de datos desde un código Python. A partir de este momento tenemos completa disponibilidad sobre esta herramienta. Para este proyecto, hemos elegido almacenar los datos en una tabla en la que cada fila será un conjunto de todos los datos recogidos. La Figura 5-9 muestra el diagrma de flujo del código que crea una tabla a la que hemos llamado “CONTEXTO”. Este código solo hace falta ejecutarlo una vez para crear la tabla.

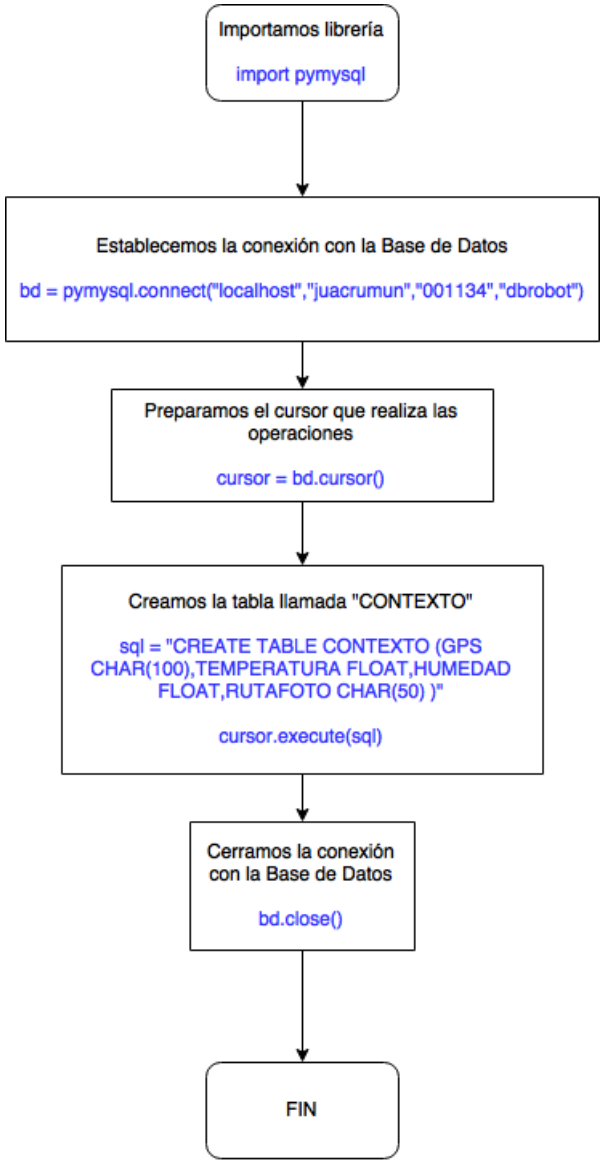


Figura 5-9. Diagrama de flujo para la creación de la tabla en la base de datos

El uso de la base de datos consistirá en crear una nueva entrada en la tabla para almacenar los datos cada vez que pulsemos el botón “Guardar en BD” del control remoto. Si el dato recibido es “bd” y conseguimos conectarnos correctamente, se crea la fila y se guardan los datos. Se ejecutará el código de la Figura 5-10.

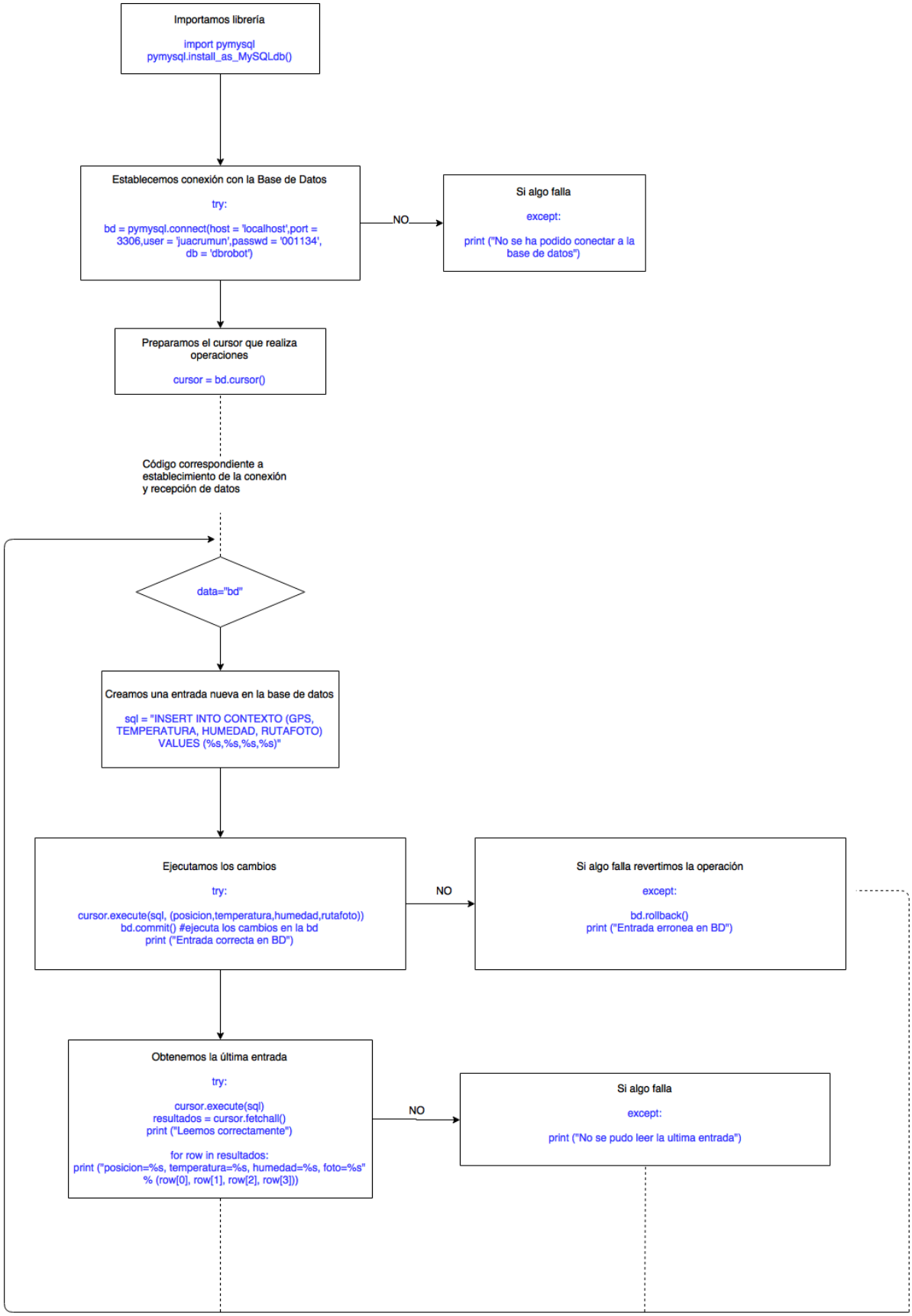


Figura 5-10. Diagrama de flujo de la gestión de la base de datos

Con esto quedan cumplidos todos los objetivos propuestos para este proyecto.

6 RESULTADOS EXPERIMENTALES

En este capítulo se comentarán las principales pruebas realizadas para comprobar el correcto funcionamiento de cada módulo. Para que funciones el software correctamente, el hardware debe estar conectado correctamente, por ello no se han realizado más que pruebas del software del sistema. En un proyecto como este la mejor prueba posible es la comprobación por inspección, es decir, ver si funciona. Dado que hay que dejar constancia por escrito, hemos recurrido a este tipo de pruebas.

6.1 Funcionamiento del control remoto

Para comprobar que la app diseñada funciona correctamente, la única comprobación posible es comprobar si se conecta correctamente y se reciben en la Raspberry Pi las cadenas de caracteres correspondientes. La comprobación consistirá en la salida dada por la función print con el dato recibido en el terminal. En la Figura 6-1 podemos apreciar un ejemplo de recepción de la cadena "hum", correspondiente a la orden de sensado de humedad.

```
pi@raspberrypi:~/Desktop $ sudo python3 finalv3pruebas.py
Esperando conexion Bluetooth

Conexion Bluetooth creada con ('AC:EE:9E:48:87:A5', 1)

El dato recibido es b'hum'
```

Figura 6-1. Comprobación de funcionamiento de control remoto

6.2 Funcionamiento del Sense HAT

Dado que mediante el Sense HAT tomamos las medidas de humedad y temperatura, la prueba consistirá en solicitar mediante el control remoto estas dos medidas en la misma ejecución del programa. De nuevo, la comprobación se hará mediante la función print con los valores obtenidos. Podemos ver los resultados en la Figura 6-2.

```
pi@raspberrypi:~/Desktop $ sudo python3 finalv3pruebas.py
Esperando conexion Bluetooth

Conexion Bluetooth creada con ('AC:EE:9E:48:87:A5', 1)

El dato recibido es b'hum'

Humedad: 45.245033264160156 %rH

El dato recibido es b'tem'

Temperatura: 29.556385040283203 C
```

Figura 6-2. Comprobación de funcionamiento del Sense HAT

6.3 Funcionamiento del modulo de cámara

Las única prueba que puede hacerse para comprobar si se hace una foto es comprobar si la foto aparece en el directorio en el que se pretende almacenar. Dado que no podemos dejar constancia por escrito de ello, la prueba consistirá en la salida de la función print con el nombre de la foto creada. Dada la dificultad para almacenar una foto en una base de datos, tomamos la opción de almacernar la ruta a la foto. Para ello creamos una variable que aumentaba con cada foto tomada. Esta variable es la parte final del nombre de la foto. Con esto evitamos almacenar todas las fotos con el mismo nombre y evitamos que se reemplacen entre ellas. Para comprobar esto, hacemos varias fotos en la misma ejecución obteniendo la salida de la Figura 6-3.

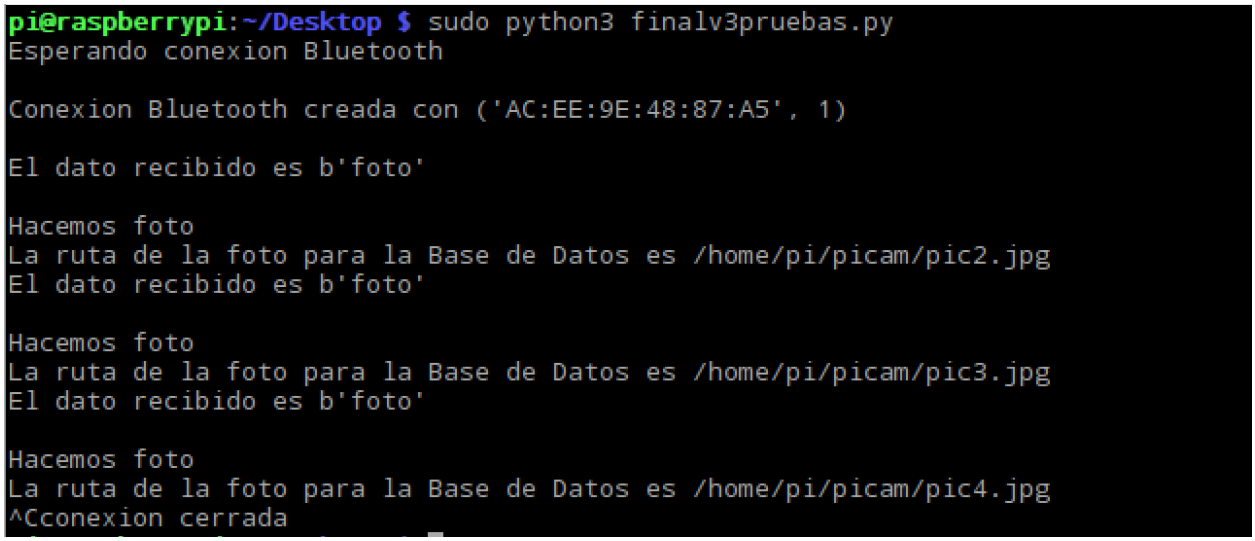


Figura 6-3. Comprobación de funcionamiento de la cámara

6.4 Funcionamiento de los motores

En este caso ocurre lo mismo que con la cámara. La única comprobación posible es observar si los motores se mueven al pulsar los botones. No hay ninguna comprobación del funcionamiento de los motores que pueda dejarse aquí por escrito más que ver la salida en el terminal del movimiento de los motores dependiendo del botón de dirección pulsado. En este caso, se pulsarán todos los botones de dirección en la misma ejecución. Obtenemos la salida dada en la Figura 6-4

```
pi@raspberrypi:~/Desktop $ sudo python3 finalv3pruebas.py
Esperando conexion Bluetooth
Conexion Bluetooth creada con ('AC:EE:9E:48:87:A5', 1)
El dato recibido es b'avanza'
Motores avanzan
El dato recibido es b'para'
Motores paran
El dato recibido es b'izq'
Gira izquierda
El dato recibido es b'para'
Motores paran
El dato recibido es b'retrocede'
Motores retroceden
El dato recibido es b'para'
Motores paran
El dato recibido es b'der'
Gira derecha
El dato recibido es b'para'
Motores paran
```

Figura 6-4. Comprobación de funcionamiento de los motores

6.5 Funcionamiento del GPS

Como se ha explicado en capítulos anteriores, el módulo GPS ha sido el que más dificultades ha dado, entre otras cosas, debido a la ausencia de recepción en interiores y a que los códigos de la librería proporcionada por el fabricante lanzaban excepciones constantes . Debido a estos problemas, resultaba bastante repeptitivo el hecho de tener que lanzar el programa y conectarse al control remoto cada vez que se probaba el GPS y se cerraba automáticamente por las excepciones lanzadas. Por ello, a lo largo del desarrollo del proyecto hemos hecho uso de la orden “cgps –s” en el terminal, la cual devuelve todos los datos recibidos por el GPS. Esta fue la primera prueba de funcionamiento del módulo GPS. Puede verse en la Figura 6-5.

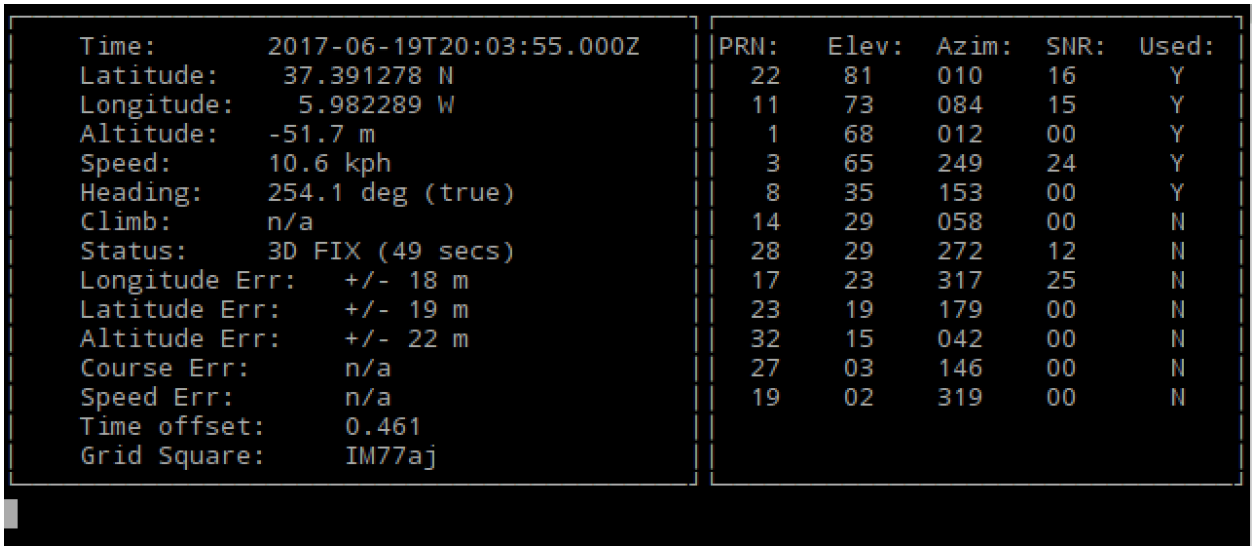


Figura 6-5. Comprobación de funcionamiento del GPS usando “cgps -s”

El siguiente paso tras comprobar el funcionamiento del módulo era hacerlo funcionar con el código implementado. Tras solucionar los errores correspondientes, la única prueba posible fue ver la salida de los valores de latitud y longitud obtenidos por nuestro código de la lista de valores devueltos por el método get_gpgga() . La Figura 6-6 lo muestra.

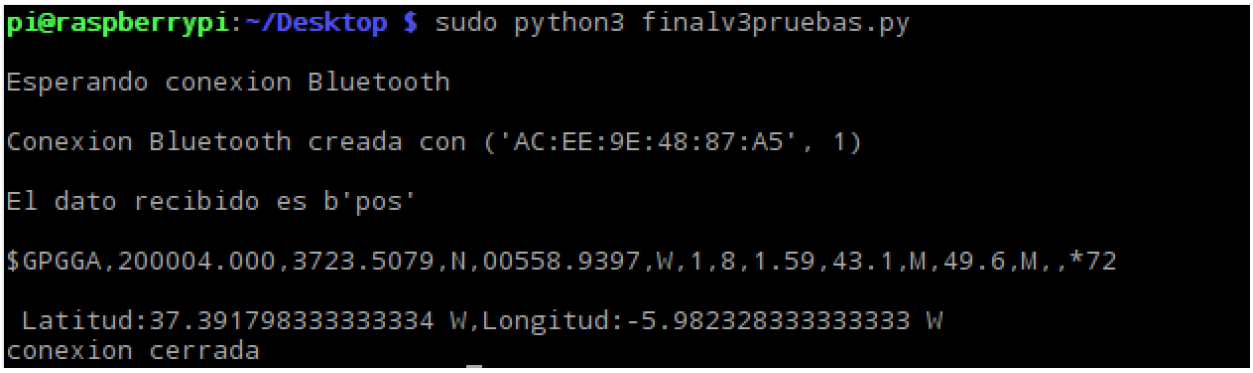


Figura 6-6. Comprobación de funcionamiento del GPS

6.6 Funcionamiento global y almacenamiento en la base de datos

La última prueba, además de comprobar el funcionamiento de la base de datos MySQL, servirá a modo de resumen de todas las pruebas anteriores. Consistirá en tomar valores de temperatura y humedad, capturar una foto, obtener la posición GPS y almacenar todo en una nueva fila de la base de datos. Para comprobar que la base de datos almacena una entrada tras otra, se muestran todas las filas almacenadas mientras se hacía la prueba. Cabe mencionar que en el momento de la prueba el módulo GPS se encontraba sin señal. En este caso se almacena en la base de datos la última posición registrada. Todo esto puede verse en la Figura 6-7.

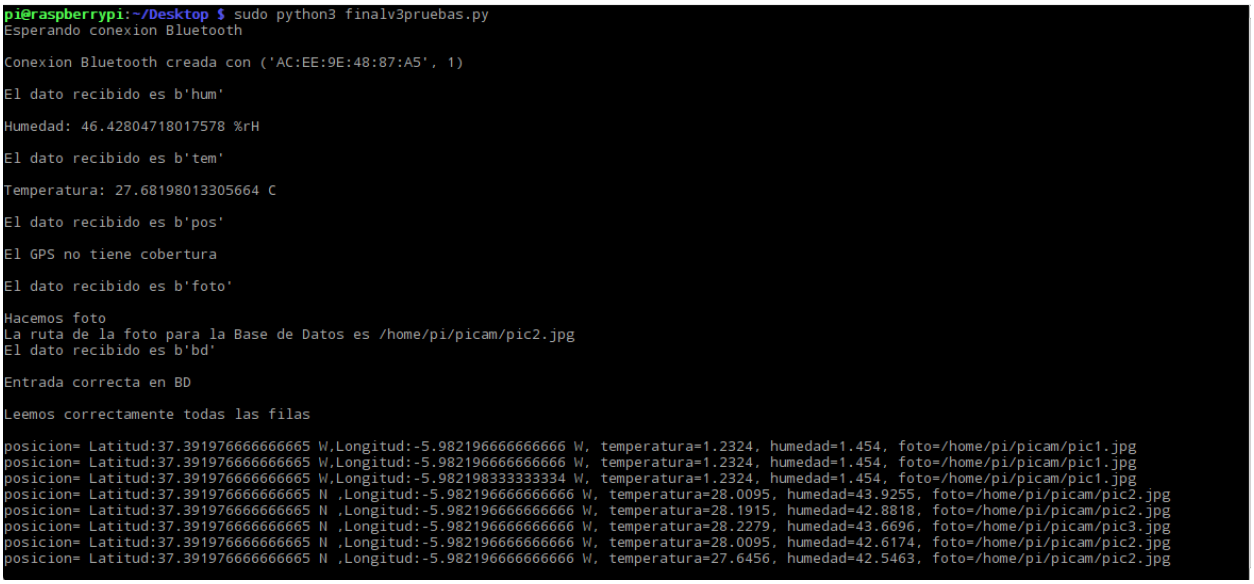


Figura 6-7. Comprobación funcionamiento de la base de datos.

7 CONCLUSIONES Y POSIBLES MEJORAS

7.1 Conclusiones

Se ha conseguido diseñar, tal y como se propuso en un principio, un vehículo robot encargado de captar datos del entorno. Es capaz de medir temperatura y humedad, calcular su posición GPS, tomar una foto y almacenar todo en una base de datos. Todo esto a través de un control remoto.

Las conclusiones sacadas al realizar este proyecto son varias, destacando las numerosas posibilidades de uso que ofrece la plataforma Raspberry Pi, además de la gran cantidad de código y documentación relacionada existente en internet, que facilita en gran medida proyectos de este tipo. Aunque la principal conclusión es a nivel personal. A lo largo de la carrera se estudian muchos conceptos teóricos y, aunque sabes que tienen utilidad, no te das cuenta hasta el momento en que te ves involucrado en un proyecto tan práctico y motivante como este. La experiencia desarrollando el trabajo ha sido bastante satisfactoria ya que, en un principio, no tenía experiencia ninguna con la plataforma Raspberry Pi ni con el lenguaje Python y, a base de trabajo, constancia y organización he conseguido desarrollar el proyecto en su totalidad dándome cuenta de lo mucho que disfruto practicando con la electrónica y las miles de opciones que ofrece.

7.2 Posibles mejoras

Este proyecto sirve como base para futuros proyectos más completos ya que, tomando como eje el sistema creado, es bastante sencillo incluir nuevos sensores y actuadores. A continuación numeraremos una serie de mejoras más concretas que se han ido descubriendo a lo largo del trabajo y que, por cuestiones de tiempo o presupuesto, no se han llevado a cabo.

7.2.1 Diseño de PCB para adaptación de conexiones

Como se mencionó en la introducción de este documento, uno de los objetivos del proyecto era el diseño de un PCB con la finalidad de alojar todos los módulos descrito. Esto es necesario ya que la mayoría de ellos se conecta al sistema Raspberry Pi mediante los pines GPIO. Por lo tanto, esto es más un objetivo no cumplido que una futura mejora. Aún con el diseño completo, la falta de tiempo ha provocado que no se pueda llevar a cabo la fabricación de dicha placa PCB. Por ese motivo, se dejará como futura mejora.

Como se ha explicado anteriormente, algunos pines son requeridos por dos módulos a la vez. Esto se sebe a que el Sense HAT ocupa la totalidad de los pines. Por lo tanto, necesitaremos sacar copias para los pines usados por el controlador de motores y el módulo GPS y una copia completa de la cabecera GPIO para el Sense HAT. En teoría no deben existir cortocircuitos ya que el acceso al pin es gestionado mediante el software que se detallará más adelante. La solución adoptada consistirá en una placa de las mismas dimensiones que el Sense HAT y que irá ubicada entre la Raspberry Pi y el sense HAT. Es decir, el segundo nivel de una torre de tres niveles. Con esta configuración obtendremos como resultado un sistema Raspberry Pi al que conectaremos nuestro PCB personalizado ocupando toda la cabecera GPIO. A nuestro PCB conectaremos el módulo GPS y el controlador de motores a través de las copias de los pines necesarios obtenidos de la cabecera GPIO y el Sense HAT a la copia de la propia cabecera GPIO completa. Cabe mencionar que al diseño se añadieron varios pulsadores y LEDs para posibles implementaciones en el futuro. Para ello, hemos hecho uso del tan conocido software llamado EAGLE.

EAGLE es una aplicación perteneciente a la familia de los llamados programas CAD (Computer-Aided Design). Es decir, es un software dedicado al diseño de circuitos electrónicos en alguna de sus fases. En concreto, EAGLE permite, entre otras, las funciones de:

- Editor esquemático: Diseño electrónico para diagramas de circuitos. Las distintas partes de un circuito obtenidas de las librerías de cada frabicante pueden conectarse a través de los puertos.
- Editor de diseño PCB: Diseño de placa de circuito impreso (PCB). Permite convertir el esquemático a huellas reales de cada parte y el rutado automático para conectarlas basándose en las conexiones definidas en el esquema.

Aunque estas dos funciones forman el grueso del programa, EAGLE es un programa completísimo que permite muchísimas funciones que no se detallan aquí. Algunos ejemplos son, algoritmos de auto rutado, creación de huellas personalizadas, impresión en formato PDF del layout a tamaño real, etc…

En este documento no se detallará el proceso de diseño, ya que hacer un análisis completo de cada herramienta usada en el proceso es una tarea muy densa e innecesaria. Se referenciará un enlace [18] a un tutorial que ha resultado ser muy útil y que, sin duda, resultará mejor herramienta que cualquiera que pueda ser escrita aquí.

Podemos ver el resultado del diseño de la placa en su fase esqumático en la Figura 7-1 y en la fase PCB en la Figura 7-2.

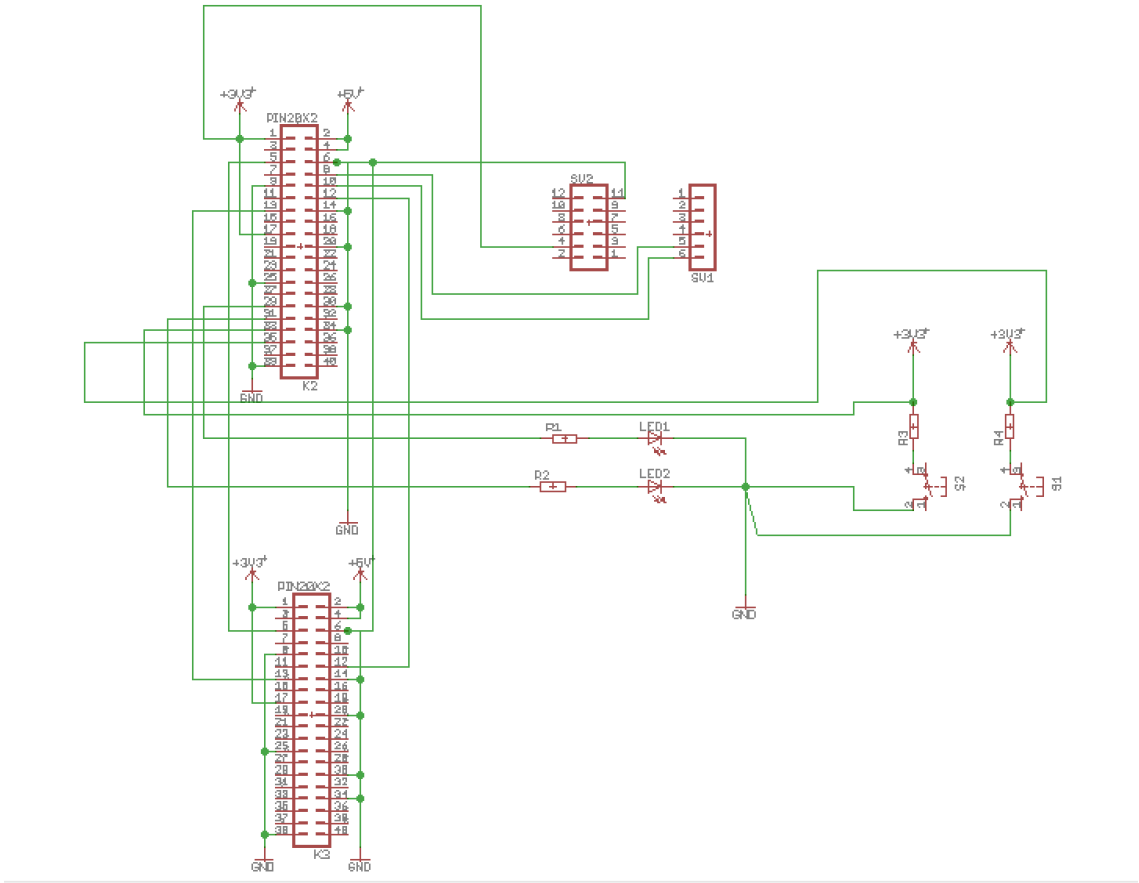


Figura 7-1. Esquemático de circuito de conexión de módulos

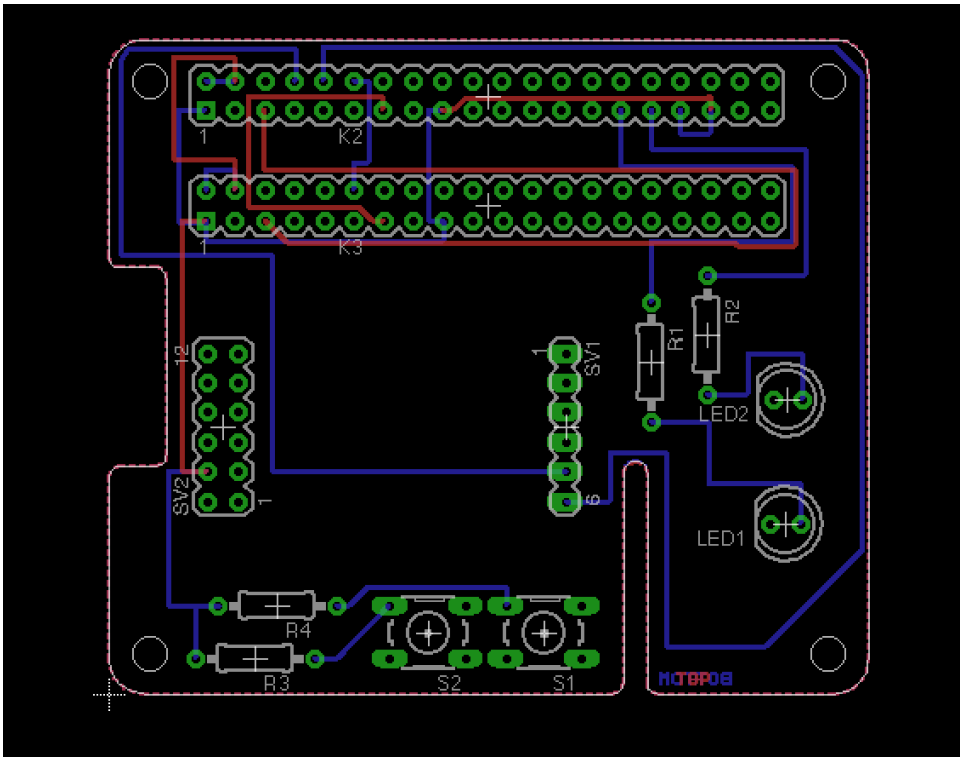


Figura 7-2. PCB de circuito de conexión de módulos

7.2.2 Sustitución de pilas AAA por batería recargable

El chasis adquirido para este proyecto trae de serie alojamiento para 4 pilas tipo AAA. Está claro que el futuro pasa por las baterías recargables y por ello sería una buena opción sustituir la alimentación. Además no es una modificación compleja ya que lo único que hay que hacer es adquirir la batería y conectarla a los motores en sustitución la las pilas.

7.2.3 Alimentación de Raspberry Pi mediante alimentación de motores

El fabricante del controlador de motores ofrece en la página web un dispositivo encargado de suministrar alimentación al sistema Raspberry Pi a través de la alimentación de los motores. Esto es, sin duda, una muy buena opción de cara a aligerar el peso del robot. Pasamos de necesitar una batería para la Raspberry y otra para los motores a necesitar una sola para los dos. El dispositivo puede verse en la siguiente

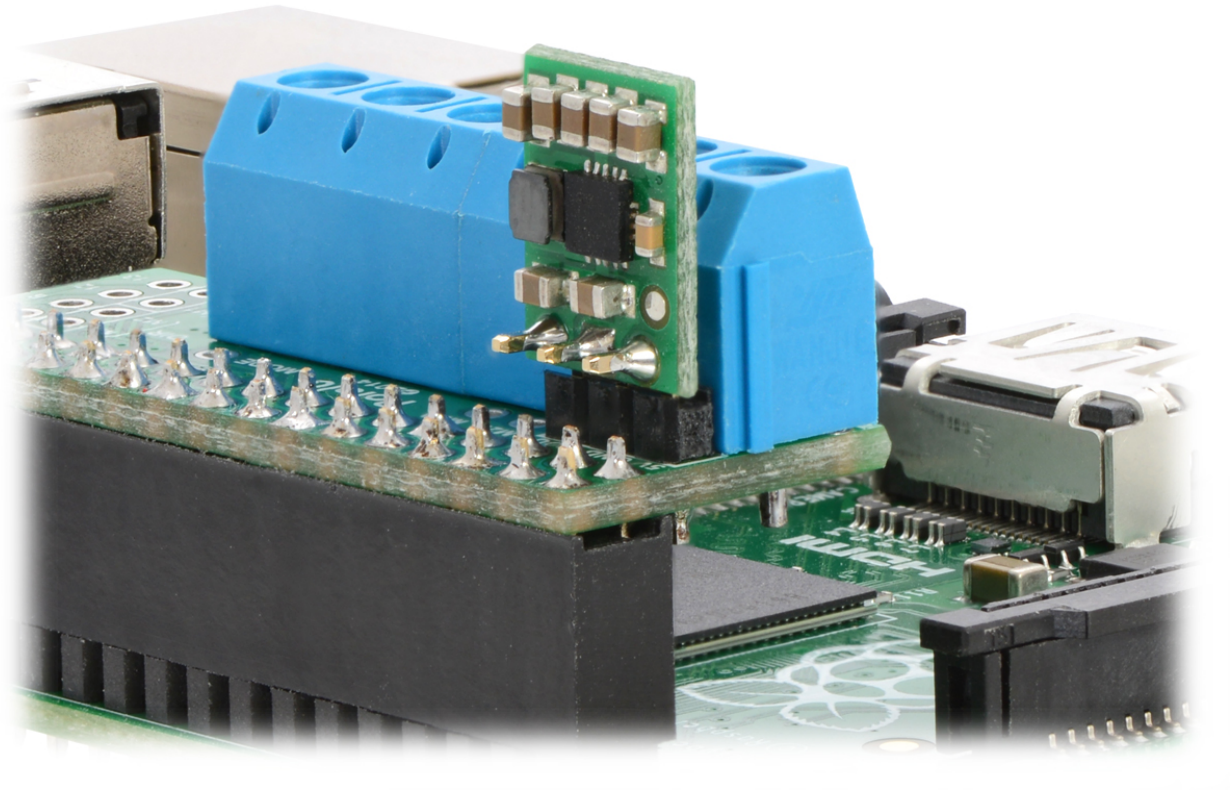


Figura 7-3. Circuito para alimentación de Raspberry a través de motores

7.2.4 Comunicación Bluetooth bidireccional

En este proyecto el control remoto envía cadenas de caracteres al robot, pero no a la inversa. Habría sido muy buena idea hacer que esta comunicación fuese bidireccional y , por ejemplo, representar los valores obtenidos por el robot en la pantalla del control remoto. Quizás esta sea la mejor de todas las posibles mejoras dado que dotaría al sistema de la independencia de un monitor conectado al sistema Raspberry Pi para visualizar los resultados obtenidos por cada módulo. En un principio se planteó la idea de llevar a cabo esta mejora, pero de nuevo la falta de tiempo dejó la idea en el aire.

ANEXOS

Scripts desarrollados

Script para creación de la tabla en la base de datos

```
#!/usr/bin/python3.4
# -*- coding: utf-8 -*-

import pymysql

#Establecemos la conexion con la BDD
bd = pymysql.connect("localhost","juacrumun","001134","dbrobot")

#Preparamos el cursor que realiza operaciones
cursor = bd.cursor()

#Creamos la tabla
sql = "CREATE TABLE CONTEXTO (GPS CHAR(100),TEMPERATURA FLOAT,HUMEDAD
FLOAT,RUTAFOTO CHAR(50) )"

cursor.execute(sql)


print ("TABLA CONTEXTO CREADA")

#Desconexion de la BDD
bd.close()
```

Script general

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""


#dvr8835
from __future__ import print_function
import time
##from drv8835 import motors, MAX_SPEED


#bluetooth
import bluetooth

#base de datos
import pymysql
pymysql.install_as_MySQLdb()

#camara
import picamera

#sense hat
from sense_hat import SenseHat

#gps
import microstacknode.hardware.gps.l80gps

# Configuramos secuencias para velocidad de motores

test_forward_speeds = list(range(0, MAX_SPEED, 1)) + \
  [MAX_SPEED] * 200 + list(range(MAX_SPEED, 0, -1)) + [0]

test_reverse_speeds = list(range(0, -MAX_SPEED, -1)) + \
  [-MAX_SPEED] * 200 + list(range(-MAX_SPEED, 0, 1)) + [0]



#instancia para gps
gps = microstacknode.hardware.gps.l80gps.L80GPS()

#Establecemos la conexion con la BDD
try:
    bd = pymysql.connect(host = 'localhost',port = 3306,user = 'juacrumun',pa
sswd = '001134',db = 'dbrobot')
except:
    print ("No se ha podido conectar a la base de datos")

#Preparamos el cursor que realiza operaciones
cursor = bd.cursor()


#Creamos socket
server_sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )

print ("Esperando conexion Bluetooth")
print ("")
puerto=bluetooth.PORT_ANY   #es lo mismo poner port=1
```

```
server_sock.bind(("",puerto)) #enlaza el socket al puerto
server_sock.listen(puerto)

client_sock,address = server_sock.accept()

print ("Conexion Bluetooth creada con {0}". format(address))
print ("")

i=1 #variable para ir actualizando el nombre de las fotos

while 1:
    try:
        data = client_sock.recv(1024)
        print ("El dato recibido es %s" % data)
        print ("")


        #motores
        if data == b'avanza':

            print("Motores avanzan")
            print ("")

            motors.motor1.setSpeed(MAX_SPEED)
            motors.motor2.setSpeed(MAX_SPEED)

        if data == b'retrocede':

            print("Motores retroceden")
            print ("")

            motors.motor1.setSpeed(-MAX_SPEED)
            motors.motor2.setSpeed(-MAX_SPEED)

        if data == b'izq':

            print("Gira izquierda")
            print ("")

            motors.motor1.setSpeed(-MAX_SPEED/2)
            motors.motor2.setSpeed(MAX_SPEED/2)

        if data == b'der':

            print("Gira derecha")
            print ("")

            motors.motor1.setSpeed(MAX_SPEED/2)
            motors.motor2.setSpeed(-MAX_SPEED/2)


        if data == b'para':
            print("Motores paran")
            print ("")
            motors.setSpeeds(0, 0)


        #foto
        if data == b'foto':
            try:

                i=i+1
```

```
                print ("Hacemos foto")

                with picamera.PiCamera() as picam:


                    time.sleep(5)
                    picam.capture('/home/pi/picam/pic%s.jpg' % i)

                    picam.close()

                rutafoto="/home/pi/picam/pic%s.jpg" % i
                print ("La ruta de la foto para la Base de Datos es %s" %
rutafoto)

            except:
                print("No se pudo hacer foto")

        #gps
        if data == b'pos':

            try:

                datosgps = gps.get_gpgga()

                posicion = " Latitud:%s %s,Longitud:%s
%s" % (datosgps['latitude'],datosgps['ns'],datosgps['longitude'],datosgps['ew
'])

                print (posicion)

            except:
                print("El GPS no tiene cobertura")
                print ("")
                posicion = " Latitud:37.391976666666665 N ,Longitud:-
5.982196666666666 W "


        #sensehat

        if data == b'hum':

            try:
                #variable para usar sensehat
                sense = SenseHat()


                humedad = sense.get_humidity()
                print ("Humedad: %s %%rH" % humedad)
                print ("")

            except:
                print ("No se pudo medir la humedad")
                print ("")

        if data == b'tem':

            try:
                #variable para usar sensehat
                sense = SenseHat()

                temperatura = sense.get_temperature()
                print ("Temperatura: %s C" % temperatura)
```

```
                print ("")

            except:
                print("No se pudo medir la temperatura")


        # Base de datos
        if data == b'bd':

            #creamos una entrada en la base de datos
            sql = "INSERT INTO CONTEXTO (GPS, TEMPERATURA, HUMEDAD, RUTAFOTO)
VALUES (%s,%s,%s,%s)"

            try:
                cursor.execute(sql, (posicion,temperatura,humedad,rutafoto))

                bd.commit() #ejecuta los cambios en la bd
                print ("Entrada correcta en BD")
                print ("")
            except:
                #Si hay algun error, revertimos la operacion
                bd.rollback()
                print ("Entrada erronea en BD")
                print ("")

            #obtenemos la ultima entrada
            sql = "SELECT * FROM CONTEXTO "
            try:
                print ("Leemos correctamente todas las filas")
                print ("")
                cursor.execute(sql)
                resultados = cursor.fetchall()


                for row in resultados:
                    print ("posicion=%s, temperatura=%s, humedad=%s,
foto=%s"  % (row[0], row[1], row[2], row[3]))

                print ("")
            except:
                print ("No se pudo leer la ultima entrada")
                print ("")


    except KeyboardInterrupt:
        break



print ("conexion cerrada")
client_sock.close()
server_sock.close()

bd.close()
```

REFERENCIAS

1. Electro-robotica.blogspot.com.es. (2017). Electro Robotica. [online] Available at: http://electro-robotica.blogspot.com.es [Accessed 20 Jun. 2017].

2. Areatecnologia.com. (2017). Tipos de Robots Clasificacion. [online] Available at: http://www.areatecnologia.com/electronica/tipos-de-robots.html [Accessed 17 May 2017].

3. Leon, J., Leon, J. and perfil, V. (2017). SISTEMAS DE LOCOMOCION DE ROBOTS. [online] Sistemasdeunrobot.blogspot.com.es. Available at: http://sistemasdeunrobot.blogspot.com.es [Accessed 17 May 2017].

4. Muchotrasto.com. (2017). Citar un sitio web - Cite This For Me. [online] Available at: http://www.muchotrasto.com/TiposDePlataformas.php [Accessed 17 May 2017].

5. Dfrobot.com. (2017). 4WD Mobile Platform (SKU:ROB0003) - DFRobot Electronic Product Wiki and Tutorial: Arduino and Robot Wiki-DFRobot.com. [online] Available at: https://www.dfrobot.com/wiki/index.php/4WD_Mobile_Platform_(SKU:ROB0003) [Accessed 10 Jun. 2017].

6. Doutel, F. (2017). Sense Hat, la placa que acompañará a la Raspberry Pi al espacio. [online] Xatakahome.com. Available at: https://www.xatakahome.com/trucos-y-bricolaje-smart/sense-hat-ahora-ya-puedes-comprar-la-placa-que-acompanara-a-la-raspberry-pi-al-espacio [Accessed 10 Jun. 2017].

7. Raspberrypi.org. (2017). Python picamera - Raspberry Pi Documentation. [online] Available at: https://www.raspberrypi.org/documentation/usage/camera/python/README.md [Accessed 10 Jun. 2017].

8. Pololu.com. (2017). Pololu DRV8835 Dual Motor Driver Kit for Raspberry Pi. [online] Available at: https://www.pololu.com/product/2753 [Accessed 16 Jun. 2017].

9. Microstack.org.uk. (2017). Microstack – Microstack GPS. [online] Available at: http://www.microstack.org.uk/products/microstack-gps/ [Accessed 11 Jun. 2017].

10. Fritzing.org. (2017). Fritzing. [online] Available at: http://fritzing.org/home/ [Accessed 14 Jun. 2017].

11. CCM. (2017). Cómo funciona Bluetooth. [online] Available at: http://es.ccm.net/contents/69-como-funciona-bluetooth [Accessed 13 Mar. 2017].

12. Es.wikipedia.org. (2017). Bluetooth. [online] Available at: https://es.wikipedia.org/wiki/Bluetooth [Accessed 13 Mar. 2017].

13. Robologs.net. (2017). Tutorial de Arduino, Bluetooth y Android #3 – Robot teledirigido con MIT inventor – robologs. [online] Available at: http://robologs.net/2015/12/26/tutorial-de-arduino-bluetooth-y-android-3-robot-teledirigido-con-mit-inventor/ [Accessed 15 Mar. 2017].

14. RedesZone. (2017). Raspbian, el sistema operativo para Raspberry Pi, se actualiza a Debian 8 "Jessie". [online] Available at: https://www.redeszone.net/2015/09/30/raspbian-el-sistema-operativo-para-raspberry-pi-se-actualiza-a-debian-8-jessie/ [Accessed 16 Jun. 2017].

15. Pythonmania.net. (2017). Las principales diferencias entre python 2 y 3 con ejemplos - Python Mania. [online] Available at: https://www.pythonmania.net/es/2016/02/29/las-principales-diferencias-entre-python-2-y-3-con-ejemplos/ [Accessed 16 Jun. 2017].

16. GPS, H. (2017). How to connect Rpi-3 and Microstack GPS. [online] Raspberrypi.stackexchange.com. Available at: https://raspberrypi.stackexchange.com/questions/50239/how-to-connect-rpi-3-and-microstack-gps/50245 [Accessed 15 May 2017].

17. Raspberrypi.org. (2017). Sense HAT - Raspberry Pi Documentation. [online] Available at: https://www.raspberrypi.org/documentation/hardware/sense-hat/ [Accessed 17 Jun. 2017].

18. Sparkfun.com. (2017). Beginning Embedded Electronics - 10 - SparkFun Electronics. [online] Available at: https://www.sparkfun.com/tutorials/110 [Accessed 15 Jun. 2017].

GLOSARIO

IDE: entorno de desarrollo integrado (Integrated Development Environment)

MIT: Instituto Tecnológico de Massachusetts (Massachusetss Institute of Technology)

PWM: modulación por ancho de pulsos (pulse-width modulation)

USB: bus serie universal (Universal Serial Bus)

PCB: placa de circuito impreso (Printed Circuit Board)

I2C: circuito interintegrado (Inter-Integrated Circuit)

SDA: línea de datos (Serial Data) SCL: línea de reloj (Serial Clock)

LED: diodo emisor de luz (light-emitting diode)

SSH: Intérprete de órdenes seguro (Secure SHell)

RPi: Abreviatura de Raspberry Pi

NMEA: especificación eléctrica y de datos entre aparatos electrónicos marinos y receptores GPS