Universidad de Granada

ESTUDIOS DE INGENIERÍA
DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

Plataforma de gestión de robots basados en Raspberry Pi

CURSO 2014/2015

Mark Rowsell

El tribunal constituido para la evaluación del proyecto PFC titulado:

Plataforma de gestión de robots basados en Raspberry Pi

Realizado por la alumna: **Mark Rowsell**
Y dirigido por el tutor: **Jorge Navarro Ortiz**

Ha resuelto asignarle la calificación de:

- ☐ SOBRESALIENTE (9 - 10 puntos)
- ☐ NOTABLE (7 - 8.9 puntos)
- ☐ APROBADO (5 - 6.9 puntos)
- ☐ SUSPENSO

Con la nota: puntos.

El Presidente:

El Secretario:

El Vocal:

Granada, a de de 2015

Universidad de Granada

ESTUDIOS DE INGENIERIA DE TELECOMUNICACIÓN

Plataforma de gestión de robots basados en Raspberry Pi

REALIZADO POR:
Mark Rowsell

DIRIGIDO POR:
Jorge Navarro Ortiz

DEPARTAMENTO:
Teoría de la Señal, Telemática y Comunicaciones.

Granada, a de de 2015.

Plataforma de gestión de robots basados en Raspberry Pi

Mark Rowsell

PALABRAS CLAVE:

Raspberry Pi, gestión de robots, conexión inalámbrica, HTML, PHP

RESUMEN:

Gracias a Raspberry Pi es posible crear robots autónomas de bajo coste y alimentados por pilas. Para sacar una mayor utilidad de los robots es necesario un programa de gestión capaz de comunicarse ellos, extrayendo información y mandando comandos.

La plataforma de gestión consiste en un interfaz web escrito en HTML y JavaScript al cual el usuario accede con un navegador web. Dentro se puede ver la posición de cualquier robot conectado a la red de la plataforma además de recibir video y controlar su movimiento.

Los robots consisten en miniordenadores Raspberry Pis conectados a una fuente de alimentación por pilas y con un sistema de movimiento con dos ruedas. Los accesorios conectados incluyen una cámara un chip GPS y una tarjeta Wi-Fi.

Robot Management Platform Based on Raspberry Pi

Mark Rowsell

KEYWORDS:

Raspberry Pi, Robot Management, Wireless Connection, HTML, PHP

ABSTRACT:

Thanks to the Raspberry Pi it is possible to create low cost, independent robots powered just using batteries. This democratises the use of robots. Due to the extended user base a simple type of management program is necessary in order to get maximum use out a small collection of robots. This management program would need to be able to communicate with the robots, extracting useful information from them and being able to send them commands.

The management platform created in this project consists of an HTML and JavaScript based web interface that the user interacts with through a browser. With this interface the user can see any robots position as well as receiving video feed and controlling the robots movement as long as they are both connected to the same network.

The robots created for this Project consist of various Raspberry Pi mini computers connected to battery based power supply and a two Wheel based movement system. The accessories connected to the robots include a camera, GPS location chop and a Wi-Fi communication card.

D. Jorge Navarro Ortiz, profesor del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada, como director del Proyecto Fin de Carrera de D. Mark Rowsell

Informa:

que el presente trabajo, titulado:

Plataforma de gestión de robots basados en Raspberry Pi

Ha sido realizado y redactado por la mencionada alumna bajo mí dirección, y con esta fecha se autoriza a su presentación.

Granada, a de de 2015

Fdo. Jorge Navarro Ortiz

Los abajo firmantes autorizan a que la presente copia de Proyecto Fin de Carrera se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a de de 2015

Fdo. Jorge Navarro Ortiz

Fdo. Mark Rowsell

Agradecimientos

Quiero agradecer a mi tutor Jorge Navarro Ortiz por su paciencia durante estos trece meses, por sus esfuerzos para dejar el proyecto de la mejor forma posible y su actitud siempre positiva ante cualquier contratiempo a lo largo del proyecto.

CONTENIDO

TABLA DE FIGURAS

TABLA DE TABLAS

GLOSARIO

IoT	Internet of Things
SSH	Secure SHell
HTML	Hyper Text Markup Language
JS	JavaScript
RMS	Robot Management System
IP	Internet Protocol
GPS	Global Positioning System
OS	Operating System
PHP	PHP: Hypertext Preprocessor
CSS	Cascading Style Sheets
LDO	Low Drop-Out
LiPo	Lithium-ion Polymer
USB	Universal Serial Bus
GPIO	General Purpose Input/Output
SD	Secure digital
CAT 5	Category 5 Cable

1 INTRODUCCIÓN

1.1 HISTORIAL

En los últimos años se está viendo el siguiente paso en la evolución de la tecnología. Érese una vez que los ordenadores ocupaban una sala entera y servían para realizar cálculos imposibles de realizar hasta esa fecha. Luego gracias a transistores de silicio se reducían de tamaño/precio y era posibles usarlos en empresas para realizar cuentas a gran escala. Tras otra reducción fueron introducidos en el hogar y llamados ordenadores personales.

La siguiente fase es donde nos encontramos ahora. Los ordenadores se han reducido de tamaño y precio de tal forma que se pueden incorporar en muchos objetos ordinarios. Como parte de esta evolución ha surgido Raspberry Pi, un ordenador pequeño y de bajo consumo. Es un dispositivo de bajo precio con el tamaño de una tarjeta de crédito. Agregando una batería y una tarjeta Wi-Fi se convierte en un ordenador totalmente móvil.

Gracias a la existencia de este tipo de mini-ordenadores, es posible la creación de robots autónomos de tamaño pequeño y sin tener que recurrir al mundo profesional de la robótica. Ahora que existe esta capacidad de crear una multitud de robots de bajo precio hay un elemento carente. Es la capacidad de manejar estos robots a distancia y de forma cómoda. No es posible enchufar un robot a un teclado, ratón y pantalla ya que se quiere mandarlo a hacer alguna actividad o ver su progreso.

Para resolver este problema se presenta este proyecto. Es una solución formada por un conjunto *hardware* y *software* para manejar varios robots pequeños en una red. El elemento fundamental reside en el servidor de gestión. Es la parte que se dedicará a hacer de intermediario entre el usuario final y los robots que éste quiere controlar.

Gracias a esta solución son posibles una variedad de sistemas usando robots. Por ejemplo, si se quiere una patrulla de cámaras móviles enviando *streams* de vídeo a un dispositivo móvil con esta solución es posible. Otro ejemplo de uso es el control remoto de un conjunto de robots y la visualización de su localización en un mapa. En el capítulo de estado del arte se encontrarán más ejemplos de uso reales.

1.2 MOTIVACIÓN

Con la tendencia de ordenadores cada vez más pequeños y accesibles junto con el aumento de robótica casera facilitado por Internet, surge la necesidad de soluciones para aprovechar estos aparatos inteligentes de forma sencilla.

La tecnología y soluciones software para conseguir la creación de esta solución ya existen y tienen comunidades muy activos. Dada una necesidad y una facilidad para su desarrollo usando tecnologías establecidos se ve una situación idílica para crear este tipo de plataforma.

Además una solución de este tipo sirve para aprovechar muchos conocimientos de muchas áreas de redes, electrónica, programación y protocolos.

1.3 OBJETIVOS

El objetivo principal de este proyecto es la creación de una plataforma para el control de robots a distancia. El programa tiene que cumplir unas expectativas de uso. Tiene que ser sobre todo sencillo dado su orientación hacia la robótica casera. Tiene que ser ampliable, permitiendo agregar funcionalidad adicional. También tiene que incluir una funcionalidad básica de serie que sirve tanto como ejemplo de la capacidad de la solución como para cumplir el fin de controlar robots a distancia

Para cumplir con la expectativa de sencillez ha de ser un programa que un usuario sin conocimiento previo puede usar. Para este fin se usara un diseño web basado en pestañas, botones y enlaces para aprovechar el conocimiento previo de páginas web que tiene un usuario medio. Además el diseño intentara minimizar la cantidad de información presentada al usuario a la vez, de allí el uso de pestañas, para no saturar al usuario dividiendo en cada pestaña cada funcionalidad.

El programa tiene que ser ampliable. Aprovechando de las pestañas una vez más se quiere conseguir hacer que la funcionalidad del programa sea algo modular. En cada pestaña se centra en una aplicación particular, por ejemplo, una pestaña para la cámara, otra para el mapa. Si el usuario quiere agregar grabación de temperatura solo hace falta agregar otra pestaña y otro programa en Python al Raspberry Pi.

Para ser útil el programa tiene que contener ciertas capacidades básicas. Para un robot lo más básico es quizás el movimiento. Para esto el programa incluye la capacidad de control el movimiento mediante botones además de localización GPS y mostrar video en tiempo real procedente del robot. Estas funcionalidades además sirven de ejemplo de cómo implementar funcionalidades adicionales al sistema.

2 ESTADO DEL ARTE

En este capítulo se verán los posibles usos de un sistema como este para conocidos sistemas de robots que existen. Además se hablara de las soluciones ya existentes de código abierto y código cerrado en el mercado realizando una comparación entre ellos y con la solución implementada en este proyecto.

2.1 POSIBLES USOS

Cada vez más hay robots autónomos que necesitan comunicarse y transmitir información además de recibir comandos gracias en parte a la aparición del llamado *Internet of Things* (IoT) y la miniaturización de la tecnología. Estos robots autónomos normalmente requieren de una aplicación de monitorización como lo que se quiere conseguir con este proyecto. Se listan a continuación un par de ejemplos interesantes en desarrollo activo de robots autónomos que utilizan software para la gestión de robots

El primer ejemplo se trata del uso de coches autónomos, que en la actualidad están siendo desarrollados en muchas empresas de automóviles como Volkswagen, o incluso empresas de tecnología como Google. Estos coches necesitan poder comunicarse con otros vehículos con el fin de evitar accidentes o compartir información útil además de consultar el tráfico y localizarse. Estos datos se pueden guardar en una base central para poder controlar el flujo de dicho tráfico, siendo así necesario un software centralizado capaz de gestionarlos.

Otro ejemplo de un *software* de gestión es para el uso de *drones* aireos y específicamente los *drones* de reparto en desarrollo por Amazon y Google. Son helicópteros pequeños y autónomos desarrollados para transportar cargas pequeñas. Lo que se quiere conseguir son envíos que llegan horas después de haberlos pedido en lugar de días. Como vuelan y son pequeños no tienen que preocuparse del tráfico ni los diseños de carreteras. Además de no necesitar conductor se ahorran el coste que normalmente se intenta ahorra combinando varios repartos en una sesión. Otro elemento que permite un ahorro de tiempo por parte de los drones. Estos *drones* requieren de muchos sensores para navegación y poder saber dónde se encuentra el *drone* en cada momento con el fin de guiarlos y para poder avisar al receptor del paquete del tiempo de llegada. De esta forma, requieren de una solución centralizada para su control.

2.2 SOLUCIONES ALTERNATIVAS

En la fase de especificación de requisitos se buscaron soluciones existentes simulares a la solución del proyecto. Existen muchas soluciones para robots específicos y propietarios sobre todo para uso industrial, pero las que se incluyen aquí intentan ser agnósticas respecto al robot utilizado o funcionar en toda una gama de robots modificables.

Microsoft Robotics Developer Studio

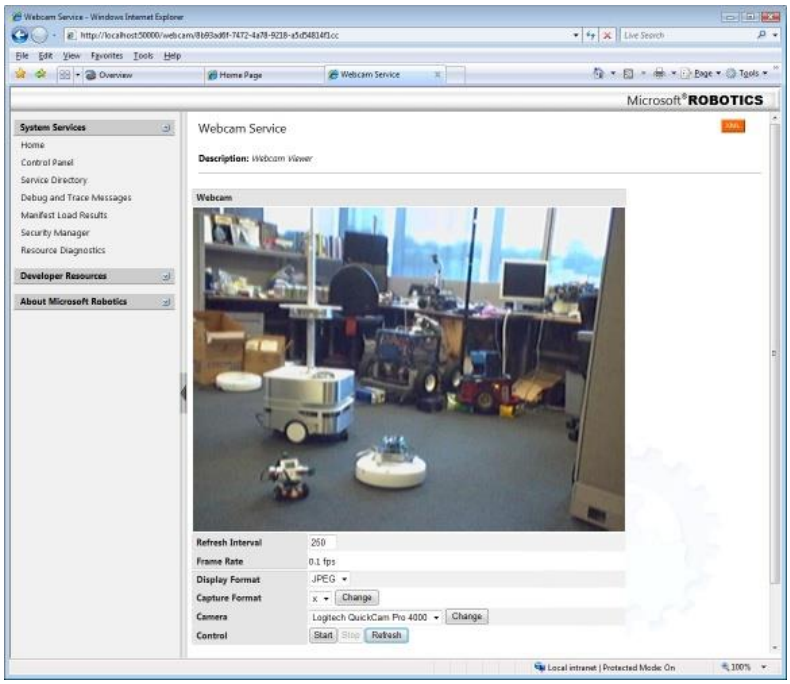

Figura 2.1. Microsoft Robotics Developer Studio

Esta aplicación creada por Microsoft y llamado *Robotics Developer Studio* [1] o RDS sirve para todo tipo de usuarios aunque se orienta para uso personal o para investigación. Contiene la mayoría de los requisitos principales expuestos en el apartado de especificación de requisitos pero utilizando tecnologías *software* desarrolladas y pertenecientes a Microsoft.

En todas las alternativas buscadas se mostrarán las características principales de la solución en una tabla para facilitar su comparación. En este caso los principales datos se incluyen en la Tabla 2.1.

Conexión	Varios, incluyendo Bluetooth y Wi-Fi entre otros
Interfaz	Web usando Internet Explorer
Enviar comandos	Sí y cualquier tipo de información
Tiempo Real	Sí
Modificable	Sí, con un lenguaje de programación visual.
Mantenido	No desde 2012
Destaca	El uso de .NETFramework. Compatible con Kinect
Licencia	EULA
Precio	Gratuito
Inconveniente principal	Compatibilidad con Sistemas Operativos limitada
Ventaja principal	Muy extensible con el sistema DSS

Tabla 2.1. Características principales de RDS

RDS es un software muy robusto y capaz de adaptar a casi cualquier robot incluyendo por ejemplo Lego NXT y el sensor Kinect. Sin embargo cabe destacar que la solución de Microsoft se limita a sistemas con Windows debido al uso de tecnologías propietarias y que no parece en desarrollo activo. Además, a no ser código abierto, si se encuentra algún error o limitación con esta solución no será posible resolverla.

The Robot Operating System

Figura 2.2. Robot Operating System

Robot Operating System [2] es un sistema para escribir *software* para robots, incluyendo interfaces web que se comunican a través de PHP. En el mundo de desarrollo de robots es la solución dominante. *Robot Management System* (RMS) es la parte más interesante en nuestro caso, ya que permite la implementación de interfaces web capaces de manejar a distancia múltiples robots. En la Tabla 2.2 se ven sus características principales.

Conexión	Cualquiera
Interfaz	Web, se crean entornos únicos a cada solución
Envío de comandos	Sí
Tiempo Real	Sí
Modificable	Todo
Mantenido	En desarrollo activo
Destaca	Código abierto
Licencia	BSD
Precio	Gratuito
Inconveniente Principal	Complejidad
Ventaja principal	Altamente adaptable

Tabla 2.2. Características principales de ROS

Entre todas las alternativas es la solución más robusta y más adaptable. Requiere muchas horas de uso para familiarizarse con el sistema, pero gracias a ello tiene muy pocas limitaciones. Entre sus pocas limitaciones cabe destacar que sólo se puede instalar la versión estable sin compilación previa en Ubuntu, pero hay soluciones no estables para sistemas ARM y

el sistema OS X, y el código está disponible para sistemas Linux. De todas las soluciones vistas es la más interesante desde el punto de vista de este proyecto.

MRPT

Figura 2.3. Mobile Robot Programming Toolkit

Mobile Robot Programming Kit [3]es la única solución entre las expuestas sin interfaz web. En su lugar se programa el interfaz en C++ usando en parte sus librerías o bien usando una aplicación creada por otro usuario. Sus características principales están en la Tabla 2.3.

Conexión	Cualquiera compatible con C++
Interfaz	Aplicaciones escritas en C++
Envío de comandos	Sí
Tiempo Real	Sí
Modificable	Todo
Mantenido	En desarrollo activo
Destaca	El único con un alto de compatibilidad de Sistemas Operativos
Licencia	BSD
Precio	Gratuito
Inconveniente Principal	No incluye interfaz web
Ventaja principal	Ligera en comparación con *Robot Operating System*

Tabla 2.3. Características principales de MRPT

Un inconveniente es el suporte que tiene, ya que los sensores se limitan a unos concretos, si bien su número es elevado. Como la tabla indica, todo se realiza con C++ por lo que es esencial su uso para poder implementar una solución. Sin embargo, es una solución actualizada y de código abierto.

3 ESPECIFICACIÓN DE REQUISITOS

La especificación de requisitos consiste en poner sobre papel exactamente qué va conseguir la solución y sobre todo, qué es lo que se busca que haga. Se ha dividido la especificación de requisitos en requisitos funcionales y requisitos no funcionales con el fin de facilitar la lectora.

3.1 REQUISITOS FUNCIONALES

Como el proyecto consiste en una parte de hardware y una parte de software se hablara de los requisitos de cada uno en apartados distintos empezando primero por el parte de *software*.

Servidor de gestión de robots

A continuación se listarán los requisitos funcionales que debe tener la aplicación que permite al usuario manejar los robots a distancia.

Alto accesibilidad: *La aplicación debería poder usarse en una variedad de dispositivos incluyendo dispositivos móviles.*

Conectar con varios robots: Este programa tiene que poder manejar varios robots a la vez y permitir cambiar de uno a otro en cualquier momento.

Recibir información: Los robots tienen que tener una manera de mandar información obtenida de sus sensores al programa en tiempo real a una tasa aceptable.

Mandar comandos: Tiene que poder controlar los robots mediante comandos de todo tipo. Requiere un cierto nivel de acceso al hardware de cada robot.

Seguridad: No se debería poder acceder a los robots o la información proveniente de los mismos por parte de usuarios no autorizados.

Poder modificarse: Para aumentar su utilidad el programa debe ser fácil de modificar con el fin de agregar funcionalidades no previstas inicialmente.

Robot

Los robots deben tener la capacidad de cumplir una lista de requisitos funcionales para poder usarse con el servidor de gestión, con el fin de probar el mismo durante su desarrollo:

Comunicación inalámbrica: Tiene que poder conectarse sin usar cables al servidor.

Procesamiento en tiempo real: Para poder utilizar información de una gama de sensores y para recibir comandos en tiempo real el robot tiene que poder procesar esta información en tiempo real. Además, la unidad debería tener la capacidad de procesamiento para adaptarse a información recibida de sus sensores sin depender del servidor central.

Sensores: Tiene que tener una compatibilidad con un rango amplio de sensores de precio ajustado.

Bajo coste: Dado que el presupuesto es limitado, no se puede usar cualquier *hardware* sino que hay que buscar componentes que cumplan sin superar un determinado coste.

Movilidad: Para poder probar un sistema de localización por GPS y un sistema de control de movimiento, hay que incluir un sistema de movimiento en el robot y por lo tanto no puede superar un peso ni tamaño que dificulten dicho movimiento.

3.2 REQUISITOS NO FUNCIONALES

Al igual que en los requisitos funcionales se dividirá este apartado en su parte hardware y su parte software.

Servidor de gestión de robots

Éste utilizará ciertos lenguajes y soluciones específicas elegidas por las cualidades que tienen. Las razones para elegir estas soluciones se explicaran a continuación.

Interfaz web: La solución tendrá una página web como interfaz hacia el usuario. Con esto se permite una interfaz que no dependa de limitaciones como sistemas operativos particulares o hardware especifico. Con una interfaz web es posible tener la aplicación funcionando en un dispositivo móvil con Android o un PC con Windows sin que resulte un problema.

Comunicación con SSH: Usando SSH se puede tener una comunicación entre los robots y la aplicación de control sobre una red normal usando un canal seguro.

Pestañas: Se utilizaran pestañas para dividir el interfaz, permitiéndose así concentrarse en la parte más importante según cada momento. Por ejemplo, si el usuario está interesado en ver dónde se encuentra el robot, con pulsar la pestaña adecuada será posible ver esa información sin que se mezcle con otra información que en ese momento no es relevante para el usuario.

Lista de robots: En todo momento debe ser visible una lista de los robots conectados al servidor. Así, pulsando encima de su identificación se puede pasar entre ellos.

Ejecutar archivos: Para hacer que el programa sea fácilmente ampliable, se debe incluir la capacidad de mandar a ejecutar diferentes tipos de programas en el robot, como por ejemplo un *script* o un programa en Python.

Robot

En este proyecto se han creado robots a modo de ejemplo para comprobar la funcionalidad del servidor. Estos robots deben de tener también ciertas características que se indican a continuación:

Raspberry Pi: En su núcleo el robot tiene que tener un mini ordenador capaz de dirigirlo. Usar un Raspberry Pi para este fin permite crear un robot con un ordenador alimentado por pilas y con un gran suporte para dispositivos además de una amplia red de recursos de aprendizaje que

simplifican mucho el proceso de creación de un robot. Este mini ordenador tiene un precio muy bajo y funciona con Linux.

Python: Dada la elección de Raspberry Pi para el ordenador de a bordo, se elige Python como lenguaje de los programas escritos para hacer funcionar el robot ya que es el lenguaje con mayor soporte en este sistema. Hay numerosas librerías y documentación en este lenguaje para Raspberry Pi.

Cámara: El robot debe incluir una cámara para poder probar que el servidor puede visualizar las imágenes mandadas por el robot. Para ello se utilizará un servicio de *streaming* desde el robot al servidor.

GPS: La aplicación tiene que monitorizar la posición de los robots en todo momento. Para conseguir este fin el robot debe tener un método de localización. Se ha optado por un GPS al ser el servicio de localización dominante.

4 PLANIFICACIÓN Y ESTIMACIÓN DE COSTES:

En este capítulo se explicará la división de tiempo que se ha hecho para realizar el proyecto, dividiendo el proyecto en varias fases y asignando cada fase un periodo de tiempo según las tareas a realizar. Después se verán los costes asociados con la realización del proyecto tanto en materiales como en recursos humanos.

4.1 PLANIFICACIÓN

El tiempo de desarrollo del proyecto se divide en varias fases. Una fase de investigación, una de especificación de los requisitos, uno de diseño, uno de implementación, una evaluación y una de documentación. En la Figura 4.1 se ve un diagrama de Gantt mostrando las distintas fases en el calendario. Algunas fases se solapan ya que en ese momento se trabaja en ambas fases a la vez.

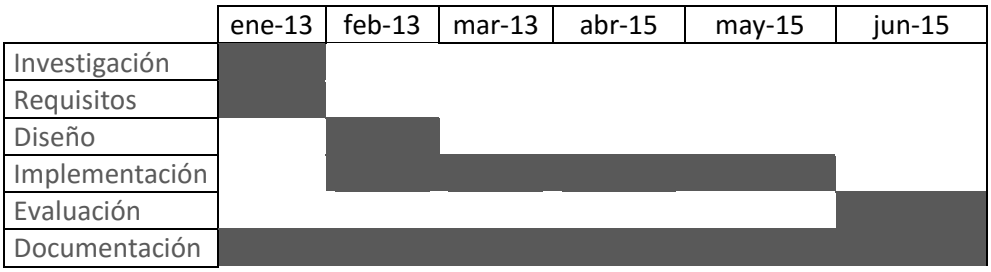

Figura 4.1. Diagrama de Gantt

Investigación

La primera fase consiste en buscar información sobre el uso potencial de la solución y sobre todo se miran a alternativas existentes del mercado. En esta fase se analizarán no sólo sus capacidades y su interfaz, sino también las tecnologías que utilizan. Esta fase tenía fijada una duración de dos meses.

Especificación de Requisitos

En esta fase se investigarán las opciones para la creación de robots basados en Raspberry Pi y se comprobarán las limitaciones de la plataforma, por ejemplo el presupuesto necesario y el nivel de dificultad de las distintas opciones de implementación del robot (sensores, actuadores, placas de interfaz, etcétera). Los elementos que se analizarán serán tres: el sistema operativo utilizado en cada robot, el lenguaje de programación a utilizar para la creación del *software* de monitorización y del *software* de control, y los componentes que se usarán para construir los robots.

Diseño

Una vez especificados los objetivos a cumplir y las limitaciones impuestas en la fase anterior, hay que realizar un diseño de la solución que pueda alcanzar dichos objetivos en el tiempo apropiado con los recursos disponibles.

Implementación

Es la fase más larga y el núcleo de la tarea. En esta fase se intenta seguir el diseño y cumplir con los requisitos en el periodo de ocho meses asignado. Ya que el diseño no puede prever los retos que surgen en la fase de implementación también hay que modificar el diseño para poder enfrentar a estos imprevistos. En esta fase se construirá un robot de prueba con los componentes elegidos o, si fuese necesario, mediante un programa de simulación. Además se creará el programa de monitorización para el ordenador con la capacidad de comunicarse con el robot, incluyendo el almacenamiento en una base de datos de la información recibida y el envío de comandos al robot tanto directamente del usuario como debidos a algún evento. Esta fase permitirá así crear un prototipo con la funcionalidad básica. Posteriormente se realizará el diseño del interfaz de usuario para el servidor de gestión de los robots, que se ejecutará en un PC. Por último, se dotará a la aplicación de la funcionalidad necesaria para recibir datos del GPS del robot y mostrar video de un *stream* del robot.

Evaluación

En el caso de este proyecto se puede entender la fase de evaluación como una extensión de la fase de implementación ya que es una solución de *software* y *hardware* en conjunto. Así, la fase de evaluación consiste en comprobar los varios sistemas creados durante la fase de implementación, analizando que la solución cumpla los requisitos puestos en la fase de especificación.

Documentación

La fase de documentación se realiza a lo largo del desarrollo del proyecto. Durante cada una de las demás fases se puede escribir componentes del guion. En la fase de investigación se recompilan fuentes de información. En la fase de especificación de requisitos se escribe la información relatando la comparativa entre las varias opciones disponibles con sus ventajas e inconvenientes. En la fase de diseño se crean los esquemáticos tanto para el *software* como el desplegué de los componentes. En la última fase, la de evaluación fue posible escribir las conclusiones e introducciones junto con los detalles restantes.

4.2 COSTES

Aquí se incluyen los costes incurridos durante el proyecto, tanto de personal como de equipos necesarios para la realización del proyecto. Así pues, se divide este capítulo en costes en recursos humanos y costes en material.

4.2.1 Recursos Humanos

El proyecto se realizó durante el periodo de cinco meses. La hora de trabajo se computa a 20€ por la realización del proyecto y 40€ por la realización de consultas. Las consultas se realizan con el tutor del proyecto. El tiempo por la realización del proyecto se calcula contando solo días laborales con una jornada de ocho horas al día. El tiempo de consulta son 2 horas al mes. Sabiendo esto en la *Tabla 4.1* se presentan los costes en recursos humanos asociados a cada fase del proyecto, junto con el coste a por todo el proyecto.

Fase	**Duración (meses)**	**Coste Realización(€)**
Investigación	0,5	880
Requisitos	0,5	880
Diseño	1	1760
Implementación	3	5280
Evaluación	1	1760
	Total (€)	10560

Tabla 4.1. El coste de recursos humanos

4.2.2 Componentes y equipamiento

A lo largo del proyecto ha sido necesario comprar múltiples componentes. En el apartado de implementación se explicara el proceso de elección de los componentes comprobados y los alternativos que había. Los componentes listados en la Tabla 4.2 incluyen componentes usados o gastados en el proceso de desarrollo además de los elementos usados en el diseño final.

Elemento	**Coste Elemento (€)**	**Cantidad**	**Cose Conjunto (€)**
Raspberry Pi	32	3	96
L293D	3,02	3	9,06
Boost Buck Converter	4.8	1	4,80
LM317LZ	0.54	4	2,16
100:1 Micro Gearmotor	4,71	4	18,85
Rueda x 2	5.95	4	11,90
Ball Caster - 3/4"	3,82	2	7,62
Camera Board	23,52	1	23,52
Cable RJ45	0,57	1m	0,57
Adafruit Ultimate GPS	30,88	1	30,88
Bread Board	5,74	1	5,74
Placa BQ	2,89	2	5,78
Carcasa	3,42	1	3,42
Carcasa pilas	1,91	2	3,82
Pilas x 8	4	1	4
Total	228,12€		

Tabla 4.2. Lista de los componentes y sus precios

Además de los componentes usados en los robots hay que considerar el equipamiento usado durante el desarrollo del proyecto. En la Tabla 4.3 se resumen los costes asociados al material y equipamiento usado incluyendo software. El precio del ordenador personal y del portátil de demostración están pensados con un año de uso sobre dos años de uso total. Artículos de librería incluyen materiales de pequeño coste como puede ser soldadura, pegamento etc.

Recurso	**Coste (€)**
Ordenador personal (1 año de 2)	850
Portátil de demostración (1 año de 2)	400
Equipo de soldadura	25
Artículos de librería	20
Sistema operativo	100
Software de ofimática	83
Total	1478

Tabla 4.3. Costes en material/equipamiento

5 DISEÑO

En este capítulo se explicara el diseño de la solución. En primer lugar se explicara el diseño general y luego se procederá a explicar el diseño de cada uno de los dos partes de la solución, el servidor de gestión y los robots.

5.1 ARQUITECTURA GENERAL

A nivel general la solución implementara un sistema de servidor-cliente, siendo los robots los clientes y el servidor de gestión el servidor. En el esquema de la Figura 5.1 se describe la relación entre el servidor y los robots además de sus características principales.

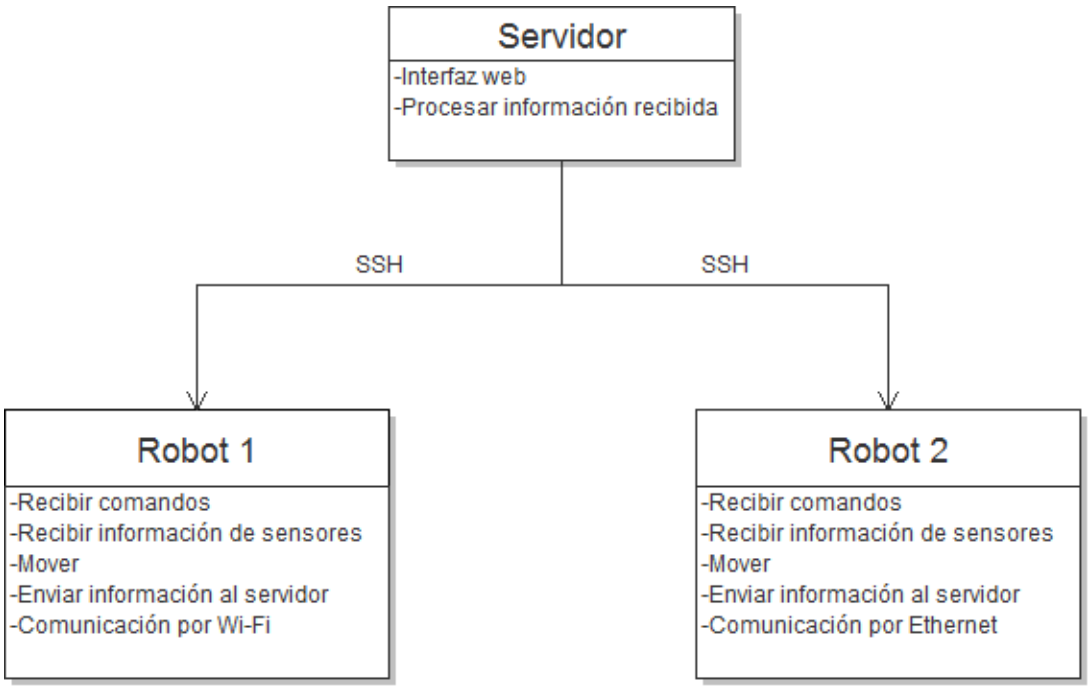

Figura 5.1. Esquema de la relación servidor-robot

5.2 SERVIDOR DE GESTIÓN

El servidor de gestión consiste en un servidor PHP con un interfaz escrito en *Hyper Text Markup Language* (HTML) y *JavaScript* (JS). En el esquema de la Figura 5.2 se ve la relación de los elementos principales para dar una idea general de su funcionamiento.

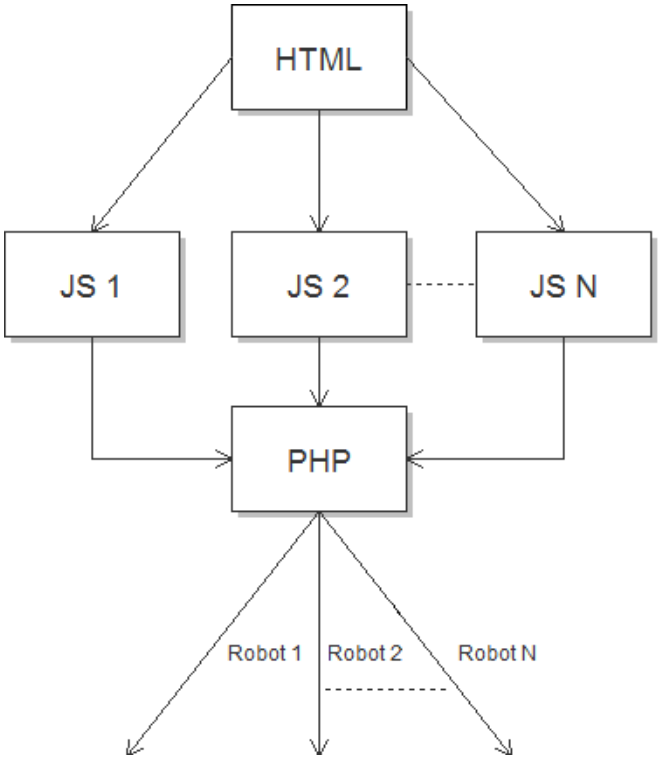

Figura 5.2. Esquema de la relación de los elementos del servidor

La interacción entre el servidor y los robots se consigue cuando se llama a un *script* en JS que se encarga de pedir la ejecución de un programa particular pasando el identificador del programa al script PHP encargado de conectar a los robots y enviarles la información necesaria. Uno de los *scripts* se encarga de obtener la información de acceso al robot para que a la hora de conectar a un robot particular el script PHP saber a qué robot conectarse. Para cada acción que el usuario quiere que haga el robot se cree un archivo JS para esa acción con los datos necesarios y solo hay que llamar al mismo programa PHP para realizar ese comando.

El interfaz de usuario se implementa usando una librería de JS, de código abierto, llamado jQuery. jQuery consiste en dos partes: una parte para la realización de páginas *Asynchronous JavaScript And XML* (AJAX), y otra parte para el diseño del interfaz de usuario llamado jQueryUI. Estas librarías se utilizan de forma extensiva para aplicaciones en *Internet* y es ideal para lo que queremos conseguir.

El acceso a la interfaz del gestor lleva un control de usuario para limitar el acceso. En la página principal se puede agregar y quitar robots del programa. Hay disponible un método de control que permite la selección del robot que mueve.

En el interfaz de usuario debería hay una sección para ver la posición del robot en un mapa. También hay una sección para poder visualizar video enviado desde el robot en tiempo real.

El programa está pensado para ampliarse así se podrá agregar secciones nuevos al interfaz de forma sencilla para poder usar otros sensores o simplemente agregar funcionalidad nueva.

La comunicación se realiza a través de *Secure Shell* (SSH) sabiendo de antemano la dirección IP de cada uno de los robots. SSH es un protocolo para acceder a maquinas remotas en una red de forma segura. Da un nivel de acceso lo suficiente alto como para poder ejecutar programas remotamente en los robots.

5.3 ROBOT

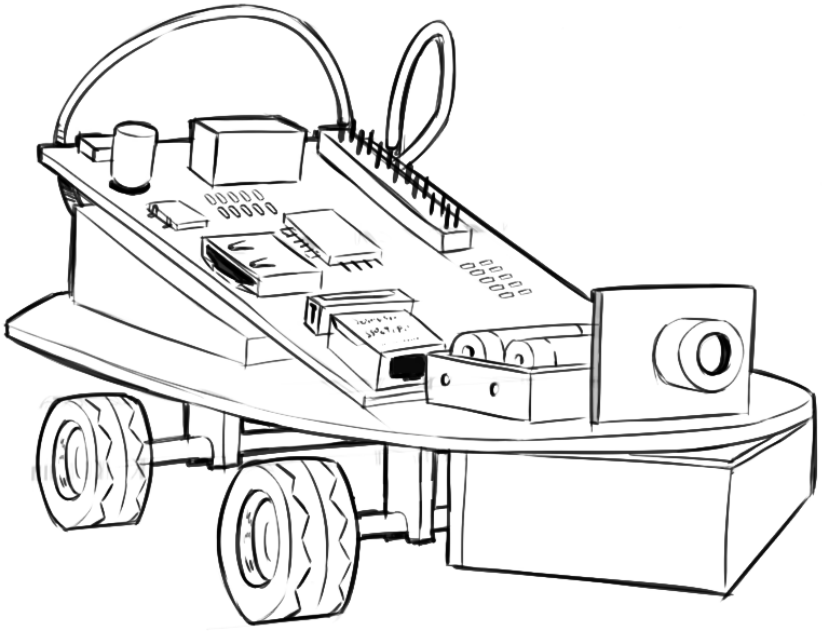

Figura 5.3. Diseño inicial del robot

En el capítulo de planificación se dijo que se había elegido la *Raspberry Pi* como el ordenador para controlar el robot. Veamos las características de este ordenador y los componentes necesarios para convertirlo en un robot como de la Figura 5.3.

La Raspberry Pi necesita una fuente de alimentación de 5V y desde 800mA hasta 1,5A. Para conseguir un robot independiente haca falta buscar una fuente de energía pequeña y ligera. Además, si el robot va a ser móvil necesitara algún mecanismo que le permite de moverse. En el capítulo de implementación se discutirá dicho método.

En cuanto a software, la Raspberry Pi necesita un sistema operativo (OS) para funcionar. En este caso se ha elegido a Raspbian, una distribución de Linux basado en Debian adaptada para ir sobre la arquitectura ARM 7 que utiliza Raspberry Pi. Este sistema operativo se arranca al encender el robot y en el proceso de arranque se ejecutan los servicios necesarios para conectar la Raspberry Pi a la red y al servidor de gestión. También tiene la capacidad de ejecutar scripts y programas escritos en Python.

Para accionar los motores del robot, la Raspberry Pi cuenta con un sistema de entrada y salida llamado GPIO. Con esto se puede mandar las señales apropiados al controlador de los motores para controlar tanto la dirección como la velocidad del robot.

Para poder probar el funcionamiento de *streaming* hay que instalar una cámara al robot. La cámara tiene sus propias librerías necesarios para su funcionamiento que se instalan antes de ponerlo en marcha. En cuanto al modelo de cámara se verá más en el apartado de implementación, pero Rapsberry Pi ofrece muchas opciones para esto gracias a la incorporación de puertos USB y además de entradas propias. De forma similar a la cámara, también sería interesante usar un chip GPS con el fin de probar el funcionar del posicionamiento en un mapa. Por suerte ya hay una tarjeta con GPS desarrollada específicamente para Raspberry Pi, que se conecta a través del GPIO.

6 IMPLEMENTACIÓN

En este capítulo se verá cómo se ha implementado el diseño del capítulo anterior. Al igual que en ese capítulo, éste se divide en tres apartados, uno con una visión general, otro para la parte correspondiente al servidor de gestión y otro para los robots de prueba.

6.1 VISIÓN GENERAL

Como la solución consiste de dos partes, se verá por un lado el servidor de gestión y por el otro lado los robots que dependen de servidor, aunque antes de meternos con la implementación de cada parte hay que tener una visión general del sistema. En la Figura 6.1 se ve un esquema de todo el sistema.

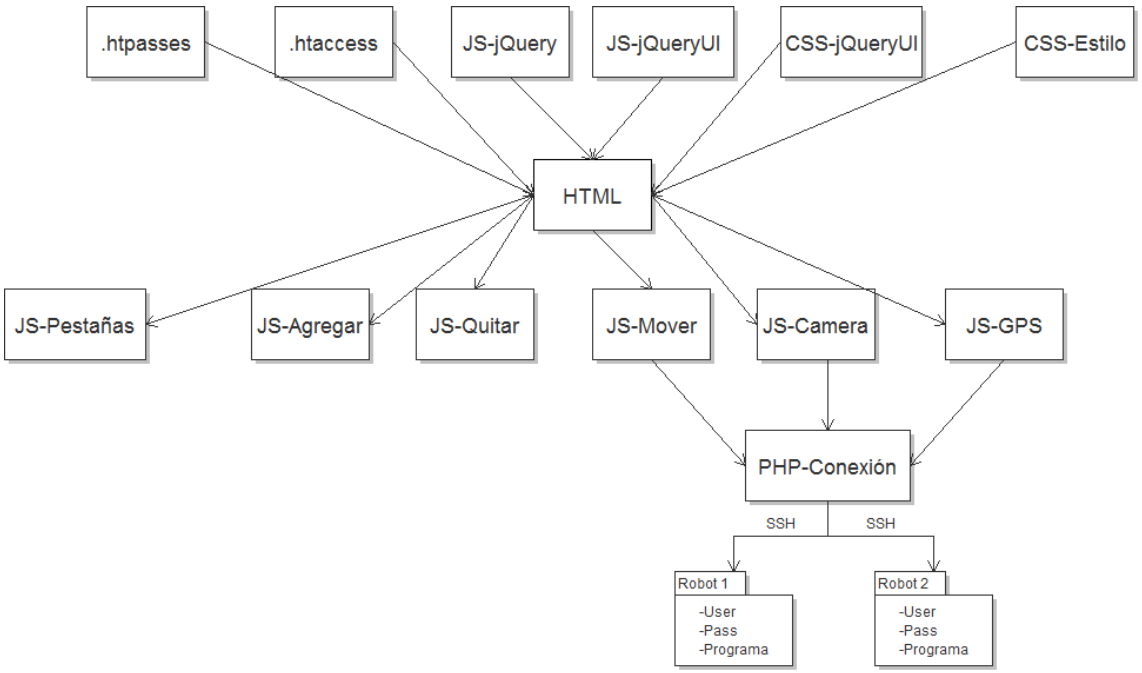

Figura 6.1. Esquema de la implementación general

En medio del esquema se ve un cuadro denominado HTML. Es una representación de la página *web* con el interfaz de usuario, por encima del cuadro HTML están todos los elementos de *software* que se incorporan con la página para hacerlo funcionar. Por debajo se ven los elementos activos de la página. Éstas son las partes que posibilitan el control de los robots. Algunos de estos elementos apuntan hacia PHP-Conexión, que es el programa encargado de comunicarse con los robots. Se ve por debajo dos robots, Robot 1 y Robot 2, que se conectan con el programa mediante SSH.

6.2 SERVIDOR DE GESTIÓN

Este apartado se centrará en la parte correspondiente al servidor de gestión. Volviendo a la Figura 5.1, esta parte corresponde a todos los bloques menos los dos Robots. Es la parte central de la solución. A continuación se tratará componente por componente. Primero se explicará la página principal y el acceso, para después seguirse con la incorporación de jQuery, seguido por el funcionamiento de los *scripts* JS y por último se explicará el programa PHP.

Antes hay que explicar cómo se ha conseguido desarrollar una aplicación web en un ordenador. Para poder trabajar de forma cómodo en la aplicación se ha instalado XAMPP (1.8.3) [4] XAMPP es un entorno de desarrollo que permite tener páginas webs en el ordenador como si estuviera instalada en el servidor. Así se puede modificar los archivos de la página sin tener que volver a subirlos a un servidor para probar su funcionamiento. Para activar XAMPP hay que iniciar el servidor Apache e ir a la página, que, a estar guardado de forma local se encuentra en http://localhost/proyecto.html. En la Figura 6.2 se ve el panel de control de XAMPP.

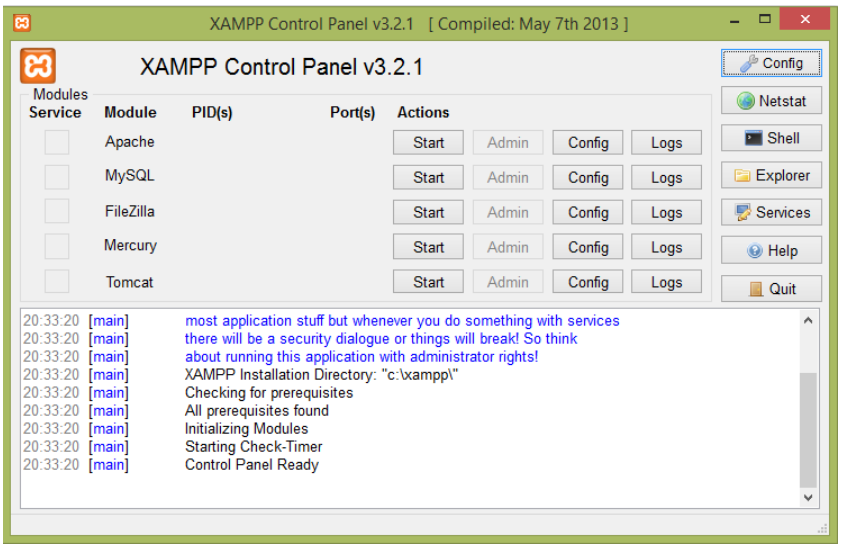

Figura 6.2. Panel de control de XAMPP

6.2.1 Página principal

La página principal consiste en un archivo HTML que se accede después de pasar por un control de acceso gestionado por un archivo .htaccess. Antes de entrar, solicita el nombre de usuario y su contraseña. En la Figura 6.3 se ve la ventana correspondiente a dicha solicitud.

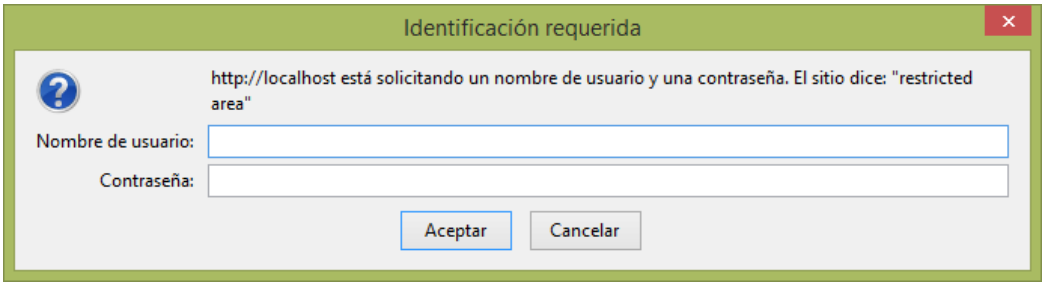

Figura 6.3. Ventana de control de acceso

Para configurar el control de acceso se cree en el directorio a proteger un archivo .htaccess y se configura con *htpasswd*. Puede encontrar más información en el manual de uso en el Anexo A.

En la página principal incluida en la Figura 6.4 se ve por la izquierda una lista vacía donde aparecerán los robots cuando estén conectados. Por debajo se ven los botones *Conectar* y *Desconectar*, que sirven para agregar y quitar los robots. Para agregar un robot también hace falta poner su dirección IP en la caja encima del botón conectar. Debajo de los botones para conectar el robot, se encuentran los botones para controlarlo. Los botones *Adelante* y *Atrás* permiten mover el robot adelante y atrás mientras que los botones *Izquierda* y *Derecha* giran el robot 90º en la correspondiente dirección.

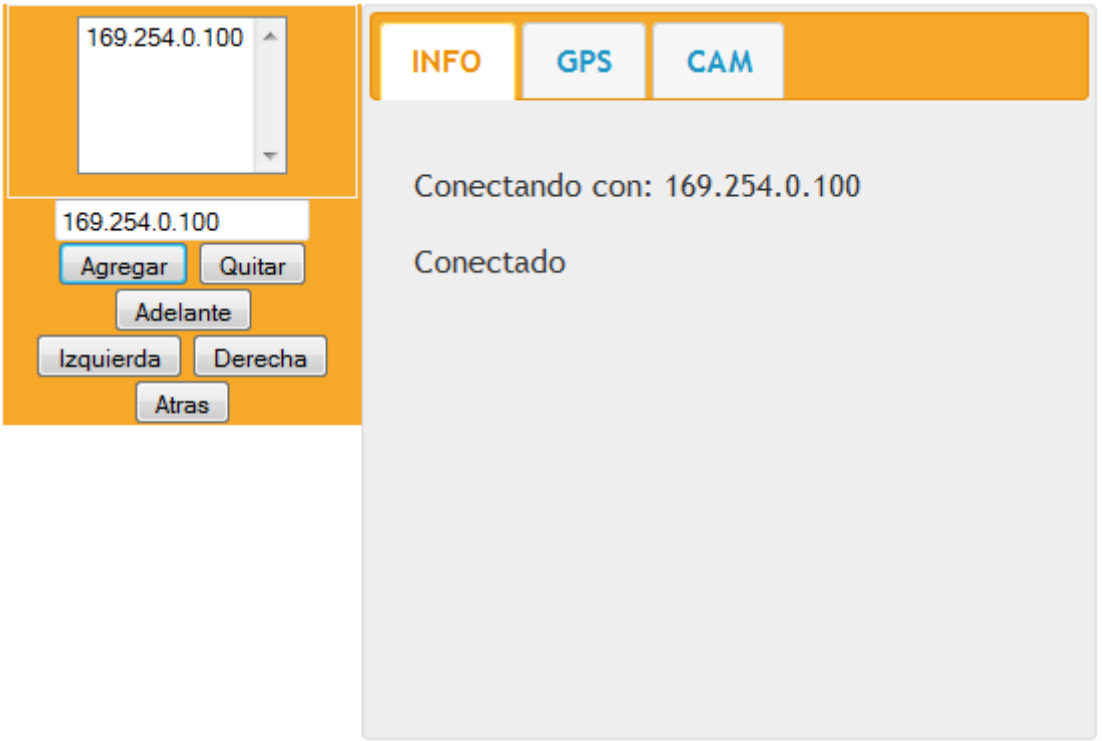

Figura 6.4. Página principal del servidor

Por la derecha se encuentran las pestañas. La pestaña que se ve en la Figura 6.4 es la pestaña de información donde el programa mostrará información relevante. Si, por ejemplo, ha acorrido un error a la hora de conectarse al robot, se mostrará el mensaje de error en esta pestaña.

Las otras dos pestañas implementadas son para mostrar la posición del robot en *Google Maps* y para ver el vídeo capturado desde la cámara de Raspberry Pi. La pestaña del GPS enseña la posición actual del robot en un mapa gracias a la incorporación de un GPS en el robot. Para que se pueda seleccionar, hay que tener el robot con GPS seleccionado en la lista. Si no hay ningún robot seleccionado o el robot seleccionado no tiene GPS, el mapa se verá en el lugar por defecto. En la Figura 6.5 se ve el mapa con la posición del robot.

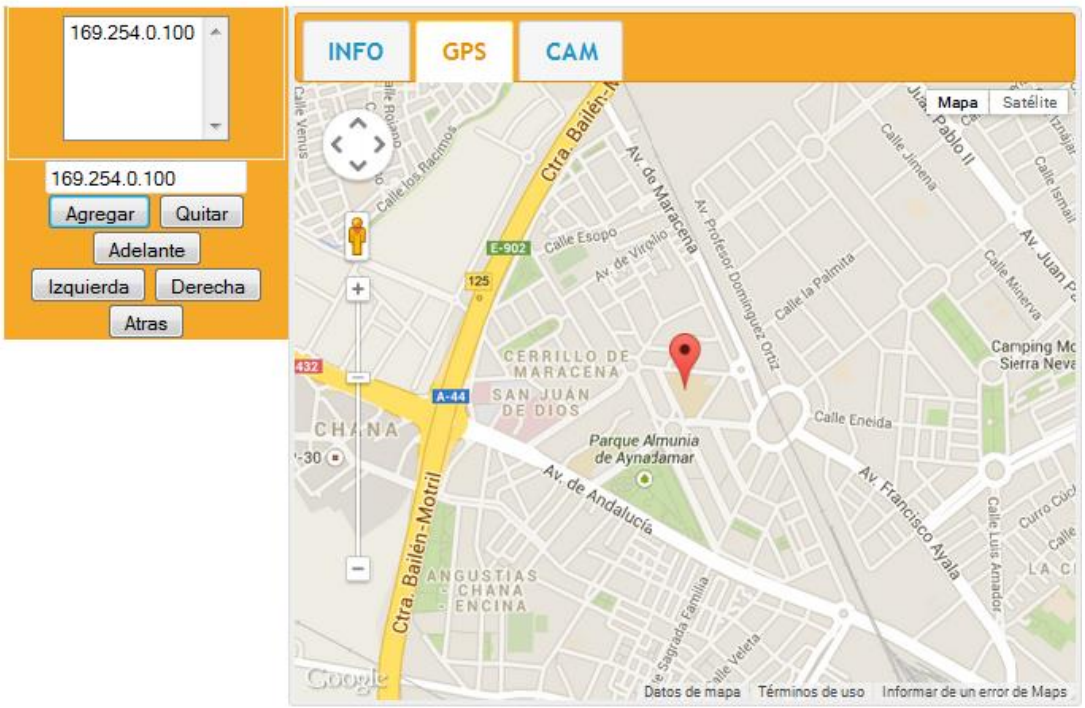

Figura 6.5. Pestaña del mapa

La tercera pestaña contiene el video recibido desde el robot. Al igual que en el caso del GPS hace falta tener seleccionado un robot con una cámara. Para activar el video hay que pulsar el botón Cámara. Tras unos segundos se recibirá un video con una frecuencia de refresco baja. Se ve cómo queda dicha pestaña en la Figura 6.6.

Figura 6.6. Pestaña de la cámara

6.2.2 JavaScript

Para obtener la funcionalidad descrita en el apartado anterior se ha utilizado *JavaScript* y la libraría jQuery. Para conseguir que una página tenga la funcionalidad proporcionada por jQuery sólo hay que incluir los archivos JS en la página y llamar a las funciones queridas desde otro archivo JS incorporado en la página. Por ejemplo, en el listado 6.1 y el listado 6.2 se ven las líneas en el archivo HTML y en el archivo *JavaScript* que permiten la incorporación de pestañas en la página web.

Listado 6.1: proyecto.html

```
<SCRIPT SRC="JQUERY-1.10.2.JS"></SCRIPT>
<SCRIPT SRC="JQUERY-UI-1.10.4.JS"></SCRIPT>
<SCRIPT TYPE="TEXT/JAVASCRIPT" SRC="TABS.JS"></SCRIPT>
```

Listado 6.2: tabs.js

```
$(FUNCTION(){
        $( "#TABS" ).TABS();
});
```

Para dar formato a las páginas hay que usar *Cascading Style Sheets* (CSS). Sin ellos no es posible usar pestañas ni dar el formato correcto a la página. Al igual que con JavaScript hay que incluir los CSS con la página HTML, pero en lugar de llamar las funciones se crean referencias

que permiten al navegador recuperar el estilo apropiado para esa sección. En el listado 6.3 y 6.4 se ve el *id* dentro del archivo HTML y el estilo general asociado a ese *id* del archivo estilos.css.

Listado 6.3: proyecto.html

```
<DIV ID="MAP_CANVAS"></DIV>
```

Listado 6.4: estilos.css

```
#MAP_CANVAS {
    WIDTH: 500PX;
    HEIGHT: 400PX;
}
```

6.2.3 PHP

Para conseguir conectar la página web a un robot se utiliza un programa PHP funcionando en el servidor. Al llamar a este programa, éste se conecta al robot pasado como argumento, a través de SSH. Dar al programa PHP la capacidad de usar el protocolo SSH requiere una librería específica para ello. La librería utilizada se llama *phpseclib* y permite el uso de SSH2 y SFTP dentro de PHP. Para usarla, solamente hay que incluir la librería y después llamar a los comandos queridos. En el listado 6.5 se ve cómo se incluye la librería y cómo se utiliza para hacer *login* y mostrar el directorio actual del robot.

Listado 6.5: conexion.php

```
//INCLUIR LAS LIBERIAS PARA SSH
SET_INCLUDE_PATH(GET_INCLUDE_PATH() . '\PHPSECLIB');
INCLUDE('NET/SSH2.PHP');
$SSH = NEW NET_SSH2($TEXTO);
IF (!$SSH->LOGIN('PI', 'RASPBERRY')) {
        EXIT('LOGIN FALLADO');
}
ECHO '<P>CONECTADO</P>';
ECHO $SSH->EXEC('PWD'); //MOSTRAR DIRECTORIO
```

6.3 ROBOT

La implementación del robot se divide en tres apartados, la elección de componentes, el montaje del robot y el *software* a bordo.

6.3.1 Elección de componentes

En la fase de planificación se hizo una investigación donde se decidió que componentes usar. En los siguientes apartados se ve una comparación entre las varias opciones existentes para cada elemento del *hardware* del robot. Primero se mostrará la elección de la fuente de alimentación, después la del chip GPS y la tarjeta de red y por último la del resto de los componentes.

Alimentación

Para la batería del dispositivo hay varias opciones entre ellas: baterías externas para móviles, adaptar baterías de los teléfonos móviles o usar pilas AA con un porta-pilas. En la Tabla 6.1 se va a comparar ciertas especificaciones que interesan a la hora de montar un coche remoto. Sobre todo se consideran el precio y la complejidad para montarlo, pero también entran características como peso y tamaño.

	Batería externa	**Pilas**	**Batería móviles**
Precio (€)	10-40	1-4 adaptador 4 por 12 pilas	14-40
Complejidad	Medio	Bajo	Alto
Peso (Gramos)	77-170	23 por 4 ó 6	40-50
Tamaño (mm)	9 x 32,5 x 21	35 x 5 x 35	2 x 60 x 40
Capacidad (mAh)	2000 a 8000 a 5V	1800 a 6V	1650 a 7,4V
Tiempo de carga	180 min	Comprar nuevas	90 min
Ventajas únicas	Opción más limpia en principio	Da 6V	Aprovechar móviles antiguos
Dudas	No es seguro que los motores de 6V funcionen con alimentación de 5V.	Hay que sacar a 5V también para Raspberry Pi	Hay que bajar a dos tensiones distintas

Tabla 6.1. Fuentes de energía posibles

Las baterías externas pensadas para los móviles serían una buena alimentación para Raspberry Pi, ya que vienen ya preparado para usar micro-usb y funciona a 5V. Sin embargo además de la Raspberry Pi hay que alimentar los motores que manejan las ruedas con una potencia suficientemente alta para soportar el peso del coche. Los motores elegidos operan a 6V, lo cual supondría aumentar el voltaje de salida con todo el aumento de complejidad y reducción de eficiencia que supondría. Así que lo que era en principio la opción más fácil y limpia se convierte en la opción más compleja. Además ninguna batería externa soporta más de 1A de salida y como el conjunto Raspberry Pi más los motores requiere en teoría hasta 2,2A (suponiendo que Raspberry funciona a su máximo rendimiento y las ruedas del coche están bloqueadas), no son suficiente.

También se pueden usar baterías de móviles. Juntando a dos en serie se consiguen 7,2V. En cuanto a precio y capacidad es la peor opción pero si se disponen de dos móviles viejos con baterías de la misma capacidad, entonces se puede convertir en una opción bastante barata. Eso sí, entre los adaptadores de voltaje, los cables y las baterías no queda muy elegante la

solución. Si se mira a un adaptador de voltaje como el LM7805 [5] se ve que con 7,2 voltios no se puede conseguir por lo que habría que usar un regulador *Low-dropout* (LDO) [6].

La tercera opción es usar pilas AA (o AAA, pero éstas tienen menor capacidad y peor relación precio/capacidad). Usar pilas supone una desventaja en cuanto a la relación peso/capacidad y, según el modelo de batería externa que se elija tampoco es la opción con la mejor relación capacidad/precio. Aun así es la mejor opción entre los tres por las siguientes razones. Es barata. Si se compran pilas desechables el conjunto puede salir por debajo de los 10€. Es variable. Si hace falta que el robot dure más tiempo en operación se puede comprar pilas de mayor duración. Pero la razón de peso para elegir la opción de las pilas es su voltaje. Funcionan a 6V (poniendo 4 pilas en serie), lo ideal para los motores. Además bajar la tensión a 5V para Raspberry Pi se consigue con relativa facilidad con un regulador de tensión de bajo precio [7].

Eligiendo las pilas quedan todavía dos dudas. Si elegir pilas recargables o desechables y si usar 4 pilas, 5 pilas o 6 pilas. Si se eligen pilas desechables con 4 habrá 6 voltios y sólo hay que usar un regulador de tensión (o diodos) para bajarlo a 5V. Esta opción es barata en cuanto a su coste inicial pero haciendo cálculos de consumo, un conjunto de 4 pilas durara 90-120mins según el uso. Habrá que comprar varios conjuntos de 4 pilas. Aun así considerando que las pilas *alkaline* valen 8€ por un pack de 20 sigue siendo la opción más barata si se incluye el precio de un cargador para las otras opciones.

Dicho esto, es la opción con mayor impacto ambiental así que hay que mirar a las alternativas. La más obvia es usar pilas recargables que dan el mismo voltaje. Si se usan 4 dan 4,8V en lugar de 6V, lo cual no es suficiente para los motores. Hay que usar 5 pilas. El único inconveniente (aparte del hecho que no se pueden comprar packs de 5 pilas) es tener que comprar un contenedor de 5 pilas que es más difícil de encontrar y potencialmente incrementa el coste, ya que hay que comprarlo en una tienda aparte (aunque se podría adaptar uno de 6 pilas con un cable también). Usar 5 supone que no se pueden recargar un conjunto a la vez para poder seguir usando el coche ya que los cargadores dan suporte a 4 o 8 pilas y las pilas vienen en packs de 4 y 8.

Con 6 pilas potencialmente se puede aumentar la duración de uso. El problema viene con bajar la tensión a las tensiones deseadas de 5V y 6V. Con 6 pilas AA desechables se consigue un voltaje de 9V y con recargables se consigue 7,2V. Ninguno de estos voltajes son apropiados para un adaptador de tensión como el LM7805 ya que la diferencia de tensión en ambos casos es demasiado poco para garantizar un buen funcionamiento según su hoja de características. Hace falta un voltaje de más bien 12V. Se puede bajar a las tensiones adecuadas utilizando un regulador LDO (*low-dropout*) tal como el NCP1117-D [8] que vale $14 para un lote de 30 [9]. Esta opción requiere un chip más sofisticado pero en cuanto a precio y facilidad el resultado es el mismo si se puede conseguir.

Como cuarta opción existen las baterías LiPo que dan 11.1V. Es un voltaje lo suficientemente alto para poder usar un regulador de tensión normal y fácil de conseguir. Para usar esta opción, habría que comprar un cargador aparte y para dar la opción de cambiar la batería cuando ésta se agota hay que comprar una de repuesto y unas cabeceras. Además no ofrecen mucha duración en el rango de precio y peso deseado debido a su relativa elevada tensión.

Viendo todos los opciones la opción preferente parece ser usar 4 pilas AA no recargables ya que la diferencia de precio con 5 pilas es mínima (no venden packs de 5 igualmente y el

contenedor tendría que ser uno de 4 también, la única diferencia es el regulador de voltaje extra que hay que comprar). Además da la capacidad de cambiar las pilas y seguir usando el coche con otras pilas de forma sencilla y sin un incremento de precio elevado.

Por ultimo decir que también se han mirado a las placas solares pero sus características eran peores que todas las demás opciones, sobre todo en precio y tamaño [10]. Su único uso posible seria cargar pilas recargables cuando el coche está en reposo (o utilizando otras pilas).

GPS y Wi-Fi

Para poder crear un coche capaz de localizarse es necesario que tenga un chip GPS. Por suerte Raspberry Pi ya dispone de un chip GPS compatible. Se llama Adafruit Ultimate GPS Breakout [11]. El chip que utiliza la mini placa se puede conseguir aparte pero la diferencia de precio tomando en cuenta el envío no supone un ahorro tomando en cuenta los componentes que hay que comprar aparte. Es un chip con una instalación fácil pero hay que comprar un cable adaptador USB [12] o fabricar uno para GPIO.

Para la tarjeta Wi-Fi hay que buscar una opción compacta y de bajo consumo. Hay una multitud de tarjetas Wi-Fi de estas características [13] y por lo tanto se puede conseguir una cualquiera a la hora de comprar otros componentes. Únicamente hay que ver si sus controladores soportan Debian (La distribución base de Raspbian).

Otros Componentes

Para el dispositivo habrá falta montar un sistema de motores con ruedas para mover el robot. Debido a su tamaño reducido con unos motores de 6V y 250mA serán suficientes para moverlo [14] Las ruedas valen entorno a los 5 euros cada par en varias páginas [15].

Para controlar los motores hace falta un chip adicional capaz de subir y bajar de velocidad además de girar el coche. El L293D es un chip barato [16] capaz de controlar de forma individual 2 ruedas [17]a varias velocidades y también da el coche la capacidad de girar cambiando el sentido de giro de una de las ruedas.

Este chip tiene la limitación de sólo funcionar con dos ruedas. Para manejar cuatro hay dos opciones: tener dos chips L293D o usar dos ruedas y una pelota en la otra punta para estabilizarlo. Debido al incremento de consumo que supondría poner dos L293D con cuatro motores y dado el bajo precio de una pelota de estabilidad se ha elegido la pelota como opción preferente [18].

Para dotar el coche de una cámara se puede comprar una que se conecta a un puerto particular de la Raspberry y promete un consumo de energía baja, lo que facilita su incorporación en un dispositivo móvil [19]. Para las pruebas mientras el coche se encuentra estacionado (conectado mediante una fuente de alimentación) se puede usar una *webcam* genérica conectado a uno de sus puertos USB.

6.3.2 Montaje del robot

Antes de proceder con el montaje del robot hay planear el diseño con un esquemático mostrando las interconexiones entre la fuente de alimentación, el GPIO de Raspberry Pi y el controlador *half-H* L293-D. En la figura 0 se ve el esquemático.

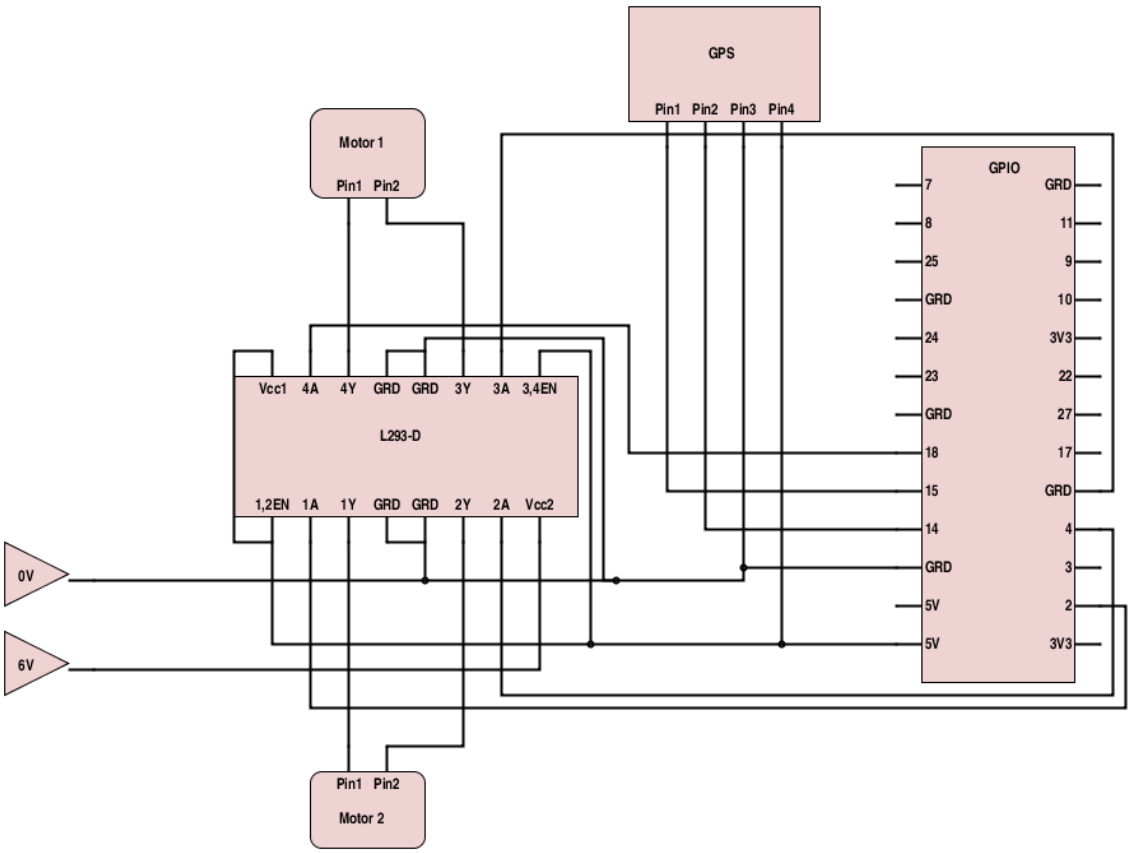

Figura 6.7. Esquemático

El robot se monta primero usando una placa de pruebas (*breadboard)* para probar el funcionamiento de cada componente antes de soldarlos. En la Figura 6.8 se ve esta placa en uso. Para conectar los componentes se usó un cable Ethernet Cat 5 cortado en trozos. Los cables Cat 5 son cables aptos para este tipo de trabajo ya que son cables de alta calidad por el precio y están hechos de cobre.

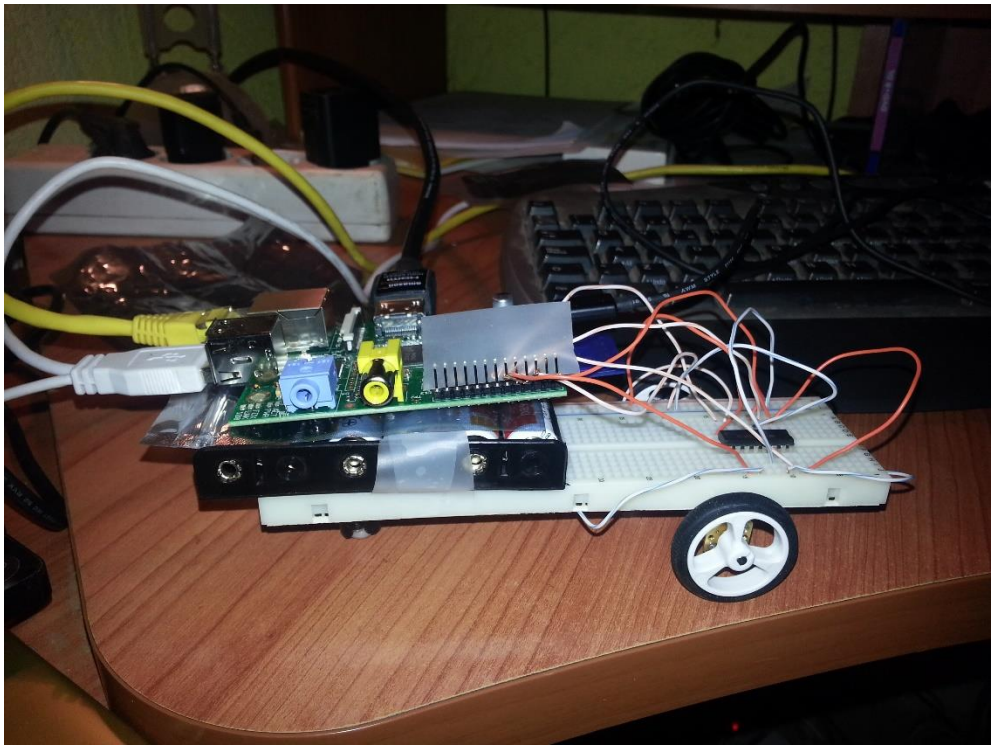

Figura 6.8. Breadboard

Con todos los componentes ya probados se monta un prototipo del robot con la placa de pruebas para ver si funciona como se ha planeado. En la Figura 6.9 se ve el prototipo ya montado.

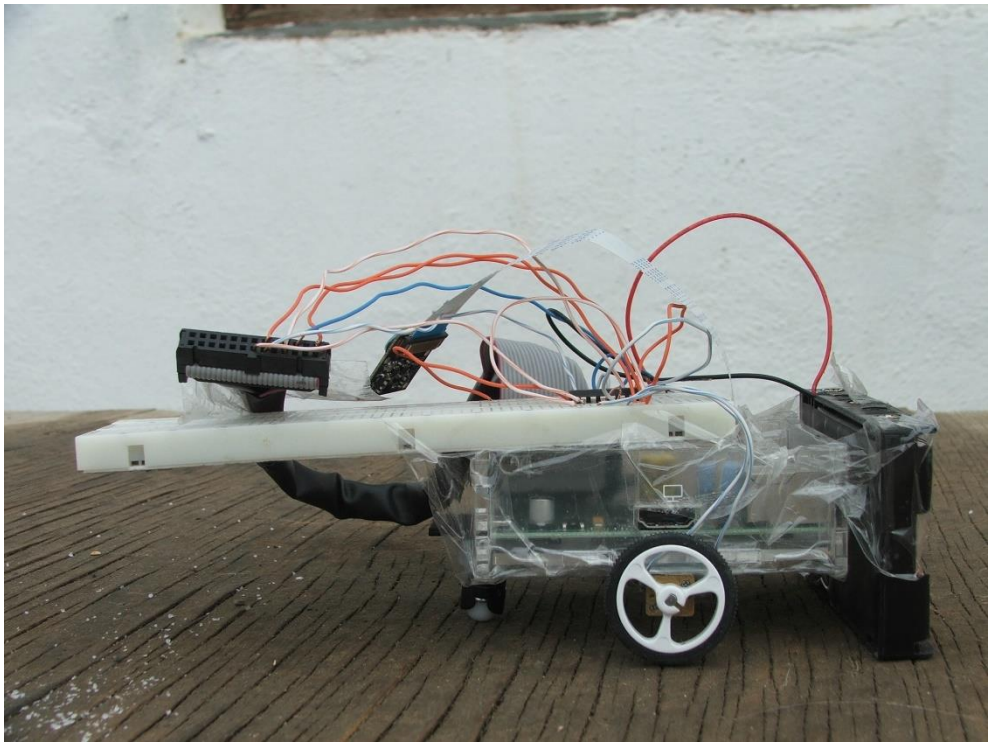

Figura 6.9. El prototipo del robot montado

Una vez realizado pruebas con el prototipo se cree la versión final reemplazando el breadboard por una placa soldada fijando los componentes de forma más segura con pegamento PCS.

6.3.3 Software

Como se dijo en el capítulo de diseño se ha programado el robot en el lenguaje Python, con la excepción de la operación de la cámara donde no hacía falta crear un programa propio sino que se ha usado un archivo *batch* para ejecutar un grupo de comandos. En el listado 6.6 se incluyen dichas líneas.

Listado 6.6: camera.sh

```
MKDIR /TMP/STREAM
RASPISTILL --NOPREVIEW -VF -HF -W 320 -H 240 -Q 5 -O /TMP/STREAM/PIC.JPG -TL 100 -T 9999999 -TH 0:0:0
&>/DEV/NULL &
LD_LIBRARY_PATH=/USR/LOCAL/LIB MJPG_STREAMER -I "INPUT_FILE.SO -F /TMP/STREAM -N PIC.JPG" -O
"OUTPUT_HTTP.SO -W /USR/LOCAL/WWW"
```

Para ejecutar programas en Python primero hay que instalarlo. Por suerte es una tarea sencilla y si interesa ver el proceso se puede consultar el tutorial de uso en el apéndice A. Una vez que se puede ejecutar programas en Python hay que instalar las librerías apropiadas. Por ejemplo para conseguir mover las ruedas del robot hay que utilizar una librería llamado *WiringPi*

[20] para controlar el interfaz GPIO de la Raspberry. En el listado 6.7 se ve un programa que mueve el robot hacia delante en Python utilizando esta librería.

Listado 6.7: moverUp.py

```
IMPORT WIRINGPI2 AS WIRINGPI
WIRINGPI.WIRINGPISETUP(
WIRINGPI.PINMODE(7,1)
WIRINGPI.PINMODE(1,1)
WIRINGPI.DIGITALWRITE(7,1)
WIRINGPI.DIGITALWRITE(1,1)
```

La primera línea del código es para importar la librería mencionado anteriormente. La segunda línea hay que llamarlo cada vez que se quiere usar la librería. Las siguientes líneas convierten los pines 1 y 7 del GPIO en pines de salida de 3,3V. Las últimas dos líneas ponen esos pines a 3,3V. Cuando están esos pines encendidos se activan los motores en una dirección y sentido tal que el robot se mueve hacia delante.

7 PRUEBAS REALIZADAS

7.1 INTRODUCCIÓN

En este apartados se detallen los pasos seguidos para probar todos los elementos de la solución. En cada apartado se detalle la funcionalidad a comprobar y la mitología seguida para comprobar dicha funcionalidad.

7.2 MOVIMIENTO

Como el programa es capaz de enviar de comandos de movimiento a los robots hay que probar ese funcionamiento. Para comprobar el uso se ha creado dos robots con ruedas controladores por dos motores independientes. Uno de esos robots de prueba se ve en la Figura 7.1. Con el robot conectado a la misma red Wi-Fi que el programa de control y agregado a la aplicación, se pulsa en el botón Avanzar para hacer el robot mover hacia delante unos centímetros. El botón Retroceder hace el robot ir hacia atrás la misma distancia y los botones Izquierda y Derecha sirven para girar el robot 45º en sus respectivas direcciones.

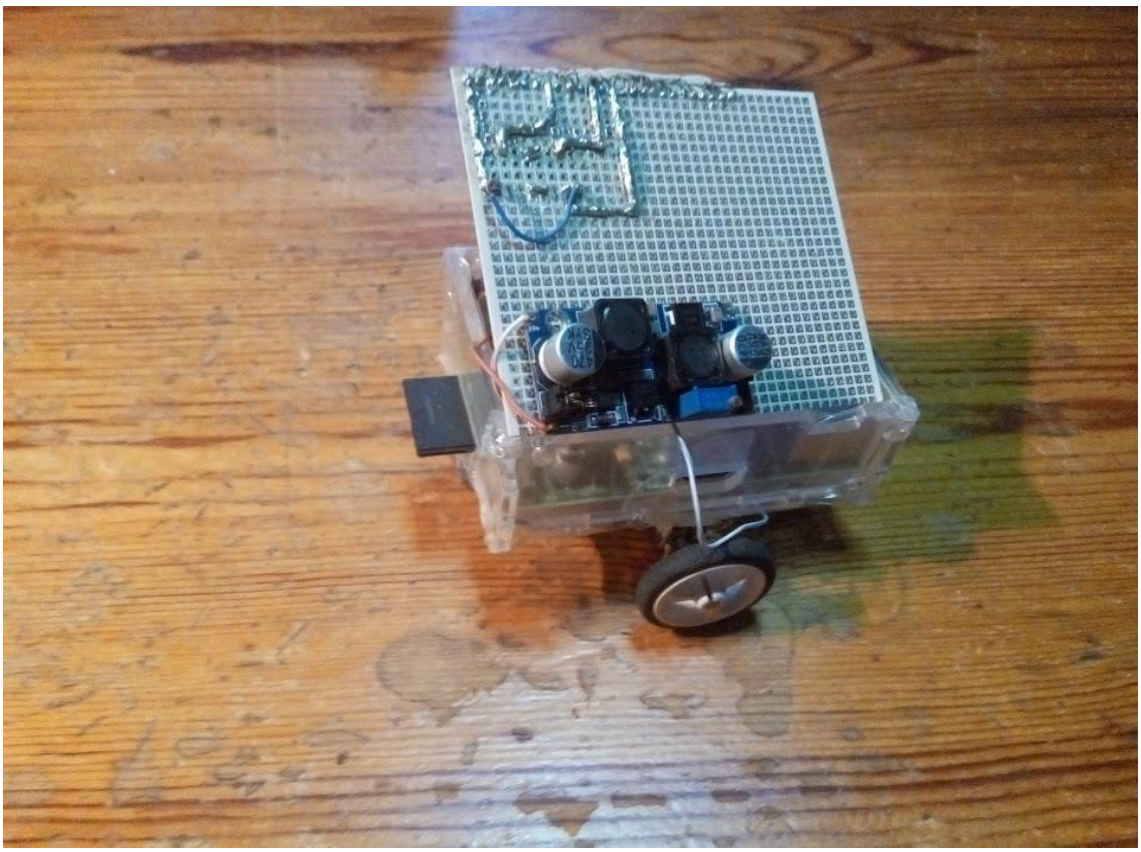

Figura 7.1. Robot con ruedas

7.3 MÚLTIPLES RASPBERRY PI

En total se usan tres Raspberry Pi para formar robots de prueba. Dos de ellos son la versión 1 modelo B y el último es el modelo 2. Se conectan los tres dispositivos a la misma red Wi-Fi que el ordenador con el programa de control usando direcciones IP fijas y conocidas puestas en el archivo de configuración dentro del OS de los Raspberry Pi. Con los tres IPs puestos se incluyen en la lista de robots dentro de la aplicación con el botón agregar.

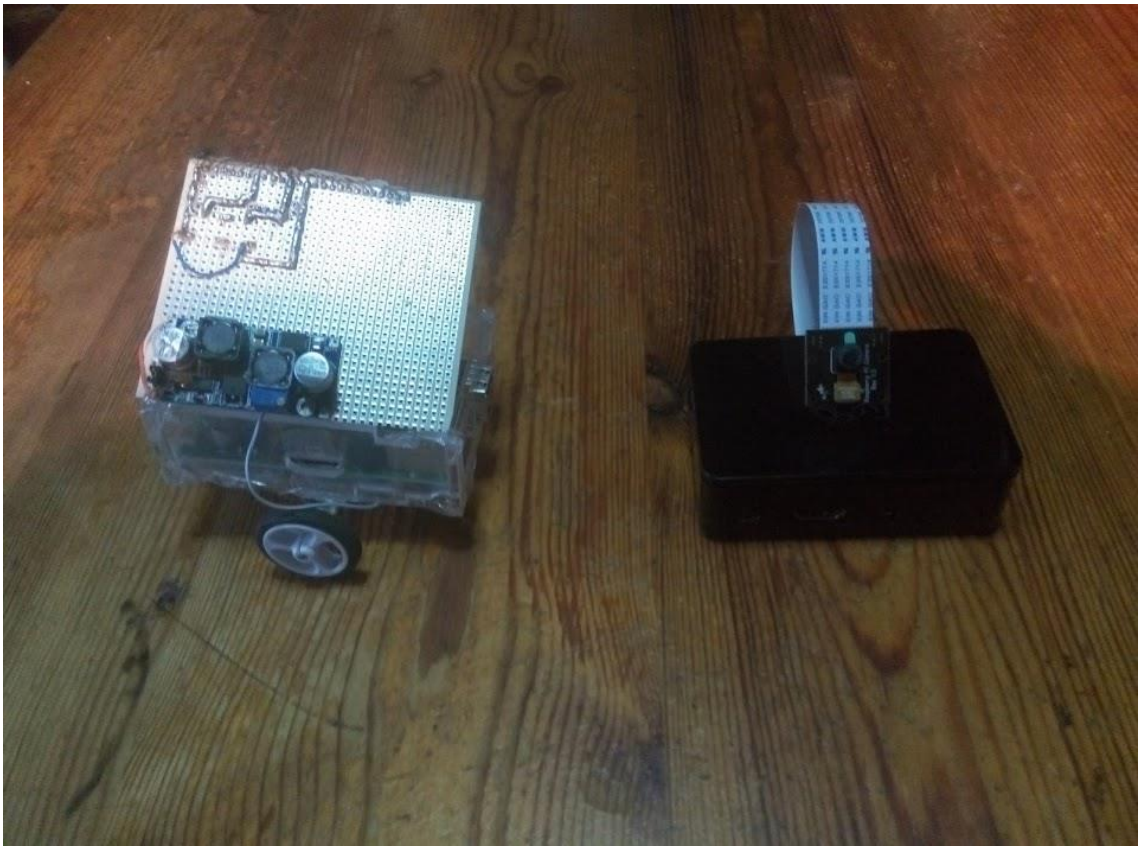

Figura 7.2. Los dos variantes de robots: fijo con cámara y móvil

Para usar cada robot se selecciona de la lista uno y este se puede controlar con los botones de movimiento si tiene ruedas o se puede saber su posicionamiento si tiene GPS. Cuando se quiere controlar otro robot se selecciona este de la lista y ahora todos los comandos se enviaran a este robot y los datos se recibían desde este.

7.4 GPS

El sistema aprovecha un módulo GPS para su uso en Google Maps. Por lo tanto uno de los robots está equipado para tal fin. Es un conjunto placa y chip creado específicamente para funcionar con el Raspberry Pi. Con esto el robot es capaz de enviar su posición a la aplicación y este lo usa para actualizar el Google Maps mostrando su posición centrada en la página. Se ve el robot con el módulo montado en la Figura 7.3.

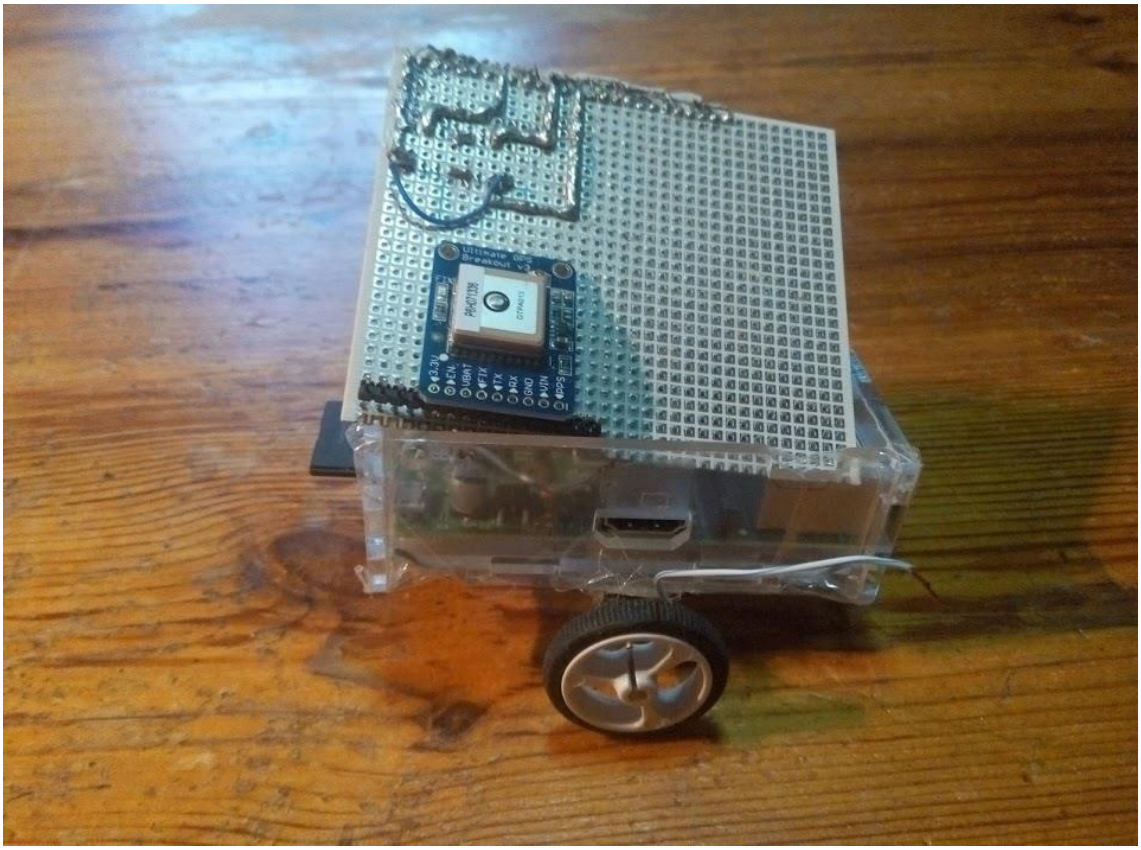

Figura 7.3. Robot con el sistema GPS puesto

7.5 STREAMING

Usando la cámara montada en una de los Raspberry Pi se puede comprobar las capacidades de streaming. Se graban las fotogramas en el robot y estas se van enviando a la aplicación atreves de la red Wi-Fi. Esta aplicación se encarga de mostrar lo que recibe del robot en cuanto le va llegando. La resolución elegida son 320x240 pixeles buscando conseguir la mayor fluidez posible. La cámara usada es el modelo Raspberry Pi Camera Module. Se puede ver el modulo conectado al Raspberry Pi 2 en la Figura 7.4.

Figura 7.4. Raspberry Pi con el módulo de cámara conectado

Los resultados son unas imágenes con una calidad suficiente que consigue sacar suficiente detalles incluso en condiciones pocas óptimas de luz como para poder dirigir un dispositivo robótico. En la Figura 7.5 se ve la calidad que recibe el programa atreves de la cámara del robot bajo la dicha situación luz poca optima, en una zona obscura y con un fuente de luz fuerte centrada un una parte de la imagen. Aun así se consiguen ver donde está situado el robot y todos los obstáculos potenciales.

Figura 7.5. Imagen recibida atreves de la cámara del robot

7.6 CONCLUSIONES

Se considera que estas pruebas cumplen con el objeto de comprobar las capacidades del software usando una mezcla de video y comandos mandados en tiempo real a uno o varios dispositivos a la vez. Se ha hecho hincapié en el uso de varios dispositivos y de la transmisión de datos en tiempo real, dos elementos esenciales al funcionamiento de un aplicación de las características que se quiere conseguir.

8 CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

Aquí se habla sobre las conclusiones del proyecto y posibles mejores y trabajos futuros realizable sobre este proyecto.

8.1 CONCLUSIONES

Los resultados más destacados son:

- El servidor se conecta a los robots de forma inalámbrica y a través de un canal seguro es capaz de mandar comandos aprovechando del protocolo SSH.
- Se conecta al servidor de gestión de forma segura con .htaccess
- El interfaz se escribió utilizando HTML y CSS de tal forma que es compatible con una variedad de *browsers* modernas.
- El interfaz está escrito de tal forma que permite la integración futuro de sensores adicionales a los robots.
- Un programa PHP permite la separación entre lo que maneja el usuario y la interacción con los robots.
- El robot consigue moverse de usando su propia alimentación.
- Los robots pueden incluir motores, cámaras y localización con GPS.
- Los robots pueden conectar a la red utilizando una tarjeta Wi-Fi o por cable Ethernet.

8.2 LÍNEAS DE TRABAJO FUTURAS

Hay muchas posibles líneas de trabajo abiertas. Aquí se listan las que han surgido a lo largo del desarrollo de la solución pero no estaban contempladas en el diseño inicial o no había tiempo para implementarlas.

- La solución incluye la capacidad de agregar sensores adicionales de una forma sencilla así que líneas futuras podrían incluir sensores adicionales. Estos incluyen sensores de infrarrojos o de ultrasónicos además de sensores de temperatura o de productos químicos [21]. Dado que la solución ya incluyen la habilidad de hacer *streaming* de video se puede incorporar audio agregando un micrófono al robot a través del puerto USB libre.
- Se puede mejorar el control de acceso al servidor de gestión. Actualmente utiliza *.htacess* pero es posible aprovechar el programa PHP para tener más funcionalidad y versatilidad que PHP ofrece [22].
- Además de ruedas hay múltiples maneras de mover un robot. Un método popular es utilizar un sistema de rotores para conseguir un robot aéreo.

ANEXO A: TUTORIAL DE USO

Para poder usar la solución hay que configurar un servidor apache para manejarla. Con XAMPP se puede usar la solución en un ordenador normal funcionando con Windows, Linux o OS X. Se instala XAMPP como cualquier aplicación. Luego en el panel de control se arranca el servidor Apache, asegurando de agregar los apropiados excepciones al cortafuegos del sistema. Para verifica que todo va con debería se abre un navegador para entrar en *localhost*.

Figura A.1. Pantalla splash de XAMPP

Una vez configurado el servidor Apache hay agregar una librería para permitir el uso SSH en programas PHP. Una implementación compatible con XAMPP es el phpseclib. Se descarga desde su página [23]. Después de descomprimir la carpeta se debe moverlo hasta el directorio php/pear dentro de donde se haya instalado XAMPP. Hay que renombrar la carpeta a phpseclib ya que es este nombre que se utiliza en la solución, así se puede usar versiones más actuales sin tener que modificar el código del programa PHP.

Los archivos que forman el servidor de gestión se pasan a la carpeta htdocs dentro de la carpeta xampp. Ya sólo queda agregar un usuario para poder acceder al servidor. Para el control de acceso hay que crear un archivo vacio llamado .htaccess dentro de la carpeta donde queremos establecer control de acceso. Después se utiliza el terminal/consola de comandos y se navega hasta el directorio apache\bin. Aquí se llama a la aplicación htpasswd para crear un usuario con contraseña. Si está utilizando la versión 2.4.4 de Apache hay un bug que cree la contraseña mal. Para corregir la contraseña hay que reescribirla para que coincide con el valor real. Esto se hace usando de nuevo htpasswd.exe. En la Figura A.2 se ven las dos llamadas a htpasswd que hay que realizar. La palabra *user* refiere al nombre del usuario y la palabra pass refiere a la contraseña. Siempre operamos dentro de la carpeta donde se instaló XAMPP.

Figura A.2. Creación de un usuario

Con todo esto hecho sólo queda conectar los robots a la red, encenderlos y entrar en la página https://localhost/proyecto.html.

ANEXO B: MANUAL DE MODIFICACIÓN

Para poder agregar funcionalidad adicional al servidor de gestión hay una manera muy clara de hacerlo. El programa funciona con pestañas que separan las varias funcionalidades. Si lo que se quiere es agregar un sensor a un robot y tener otra pestaña para ver la información del sensor o modificarlo se puede conseguir de forma sencilla.

Lo primero que hay que hacer es agregar una pestaña nueva a la página principal. Esto se hace añadiendo líneas a proyecto.html. En el listado B.1 se ve un ejemplo de esto suponiendo que la pestaña que se quiere agregar corresponde a un sensor de temperatura. Las líneas nuevas están subrayadas. También se aprovecha para incluir en la página el nuevo script JS necesario para el funcionamiento de la interacción entre la página y el servidor PHP.

Listado B.1: proyecto.html

```
(…)
<script type="text/javascript" src="mapa.js"></script>
<script type="text/javascript" src="tabs.js"></script>
<script type="text/javascript" src="camera.js"></script>
<script type="text/javascript" src="temperatura.js"></script>
(…)
<!-- Pestañas -->
        <div id="tabs" style="float:left;">
                <ul>
                <li><a href="#tabs-1">INFO</a></li>
                <li><a href="#map_canvas" id="maptab">GPS</a></li>
                <li><a href="#camera_tab">CAM</a></li>
<li><a href="#temperatura">TEMP</a></li>
                </ul>
                <div id="tabs-1"></div>
                <div id="map_canvas"></div>
                <div id="camera_tab">
                        <button id="Camera">Camera</button>
                </div>
<div id="temperatura">
<button id="TempButton">Medir Temperatura</button>
</div>
        </div>
```

Una vez que este creada la pestaña hay que crear el *script* incluido en la página. En este ejemplo el *script* se llama *temperatura.js*. Como regla general es una buena idea copiar el archivo *agregar.js* y usarlo de base para el *script* que quiere hacer. En el caso del ejemplo se modifican las líneas copiadas de *agregar.js* para sean como las líneas subrayadas del listado B.2, agregando también líneas nuevas. Estas líneas dicen al servidor PHP que tiene que lanzar el programa *temp.py* en el robot y la respuesta se pasa a la pestaña TEMP.

Listado B.2: temperatura.js

```
(…)
$('#TEMPERATURA').HTML(R); // MOSTRAR LA RESPUESTA DEL SERVIDOR EN EL DIV CON EL ID "TEMPERATURA"
(…)
VAR PARAMETROS = {
                        DIRIP: $('#VALOR').VAL(),
                        TAREA: 2,
                        COMANDO: "SUDO PYTHON TEMP.PY",
                };
(…)
$('#TEMPBUTTON').CLICK(PETICION);
```

ANEXO C: PROBLEMAS PRÁCTICOS

Durante el proceso de desarrollo surgieron problemas tanto con el lado software como el lado hardware.

SOFTWARE

Debido a una limitación de la implementación del servidor Apache utilizado y sobre todo a la librería *phpseclib* no era posible mantener una conexión persistente entre el servidor y el robot debido que se cierre la conexión a ejecutar un comando [24]. Para resolver este problema hay que modificar la librería phpseclib y encontrar una alternativa que no actúa de esta forma.

HARDWARE

Cuando se conectó dos dispositivos, un teclado y una tarjeta Wi-Fi a la primera placa Raspberry Pi el consumo que generaban era lo suficiente para quemar el chip controlador de los puertos USB. Dado que Raspberry Pi aprovecha de un canal USB para la comunicación por Ethernet también se dejó inútil ese puerto y había que comprar otra Raspberry Pi. Parece un problema bastante común del modelo B de Raspberry Pi ya que al mes de romperse el chip se sacó otro modelo con el nombre B+ capaz de proporcionar una alimentación mayor y con una protección sobre-corriente [25]. La segunda Raspberry Pi a ser un modelo con fecha de fabricación más nueva resultó ser incompatible con la tarjeta de memoria *Secure Digital* (SD) utilizado con el anterior. Para resolver el problema se tuvo que comprar una tarjeta de memoria SD nuevo [26].

REFERENCIAS

[1] *Robotics Developer Studio*, http://msdn.microsoft.com/en-us/library/bb648760.aspx
[2] Robotics Operating System, http://www.ros.org/
[3] Mobile Robot Programming Kit, http://www.mrpt.org
[4] XAMPP, https://www.apachefriends.org/es/index.html
[5] LM7805, http://www.fairchildsemi.com/ds/LM/LM7805.pdf
[6] Low-dropout Regulator, http://en.wikipedia.org/wiki/Low-dropout_regulator
[7] LM317, http://mabisat.com/reguladores-de-tension/1006-regulador-de-tension-lm317.html
[8] NCP1117-D, http://www.onsemi.com/pub_link/Collateral/NCP1117-D.PDF
[9] Lote de NCP1117-D, http://www.onsemi.com/pub_link/Collateral/NCP1117-D.PDF
[10] Placa solar, http://www.electan.com/energia-solar-c-300.html
[11] GPS, https://www.adafruit.com/products/746
[12] Guía de instalación del GPS, http://learn.adafruit.com/adafruit-ultimate-gps-on-the-raspberry-pi/introduction)
[13] Adaptador Wi-Fi, http://dx.com/es/p/ultra-mini-nano-usb-2-0-150mbps-802-11b-g-n-wifi-wlan-wireless-network-adapter-white-67532
[14] Motor, http://www.adafruit.com/products/711
[15] Rueda, http://www.bricogeek.com/shop/robotica/110-rueda-de-goma-32x7mm.html
[16] Precio L293D, http://www.adafruit.com/products/807
[17] L293D, http://www.datasheetcatalog.com/datasheets_pdf/L/2/9/3/L293D.shtml
[18] Bola, http://www.skpang.co.uk/catalog/pololu-ball-caster-with-38-metal-ball-p-464.html
[19] Cámara para Raspberry Pi, http://www.raspberrypi.org/camera
[20] Wiring Pi, http://wiringpi.com/
[21] Sensores disponibles, https://www.modmypi.com/raspberry-pi-hacking/sensors
[22] Seguridad PHP, http://php.net/manual/en/features.http-auth.php
[23] Phpseclib, http://phpseclib.sourceforge.net/
[24] Limitación de phpseclib, http://stackoverflow.com/questions/15361748/how-to-serialize-phpseclib
[25] Raspberry Pi B+, http://www.raspberrypi.org/products/model-b-plus/
[26] Compatibilidad tarjetas SD, http://www.raspberrypi.org/forums/viewtopic.php?f=6&t=5652