Robot móvil conectado como plataforma de aprendizaje

José Carlos Maciá Mora
Grado en Ingeniería Informática
Sistemas Empotrados

Consultor: **Jordi Bécares Ferrés**
Profesor responsable de la asignatura: **Pere Tuset Peiró**

Junio de 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	Robot móvil conectado como plataforma de aprendizaje
Nombre del autor:	José Carlos Maciá Mora
Nombre del consultor/a:	Jordi Bécares Ferrés
Nombre del PRA:	Pere Tuset Peiró
Fecha de entrega	08/06/19
Titulación:	Grado en Ingeniería Informática
Área del Trabajo Final:	Sistemas Empotrados
Idioma del trabajo:	Castellano
Palabras clave:	Robot, Raspberry, MSP432P401R

Resumen del Trabajo (máximo 250 palabras): Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.

La finalidad de este trabajo es el desarrollo de un robot móvil, el cual puede ser usado, por alumnos de secundaria o ciclos formativos de formación profesional, como elemento de motivación para el aprendizaje de la programación de este tipo de dispositivos.

El trabajo se enmarca dentro del área de sistemas empotrados. Para el desarrollo del proyecto se ha seguido una metodología estructurada. Este documento incluye las distintas fases de análisis, diseño y realización de tareas para llevar acabo el proyecto.

El robot dispone de una cámara, motores para el desplazamiento, servos para el movimiento de la cámara y sensores de ultrasonidos. El alumno puede acceder al control de estos periféricos mediante una librería desarrollada para este fin.

El sistema está basado en la placa SimpleLink MSP432P401R LaunchPad de Texas Instruments y un mini ordenador Raspberry Pi 3. El primero corre el sistema operativo TI RTOS y el segundo corre el sistema operativo Raspbian.

Los alumnos se conectan al robot por medio de un navegador. Pueden subir los programas para su ejecución, o bien manejar el robot directamente, controlar la cámara etc. El lenguaje escogido para que los alumnos creen los programas es Python. Alumnos de mayor nivel pueden acceder mediante el protocolo SSH. Podrán estudiar, proponer cambios o añadir funciones al sistema.

El resultado ha sido un vehículo móvil plenamente funcional para los propósitos que se marcaron al inicio. El proyecto es viable tanto desde el punto de vista económico como desde el punto de vista técnico.

Abstract (in English, 250 words or less):

The purpose of this project is the development of a mobile robot, which can be used by secondary or vocational education and training students. Thus, its use may be considered as an element of motivation to learn how to program this kind of devices.

This project focuses on the area of embedded systems. A structured methodology has been followed in order to develop it. This document includes the different phases of analysis, design and task performance to carry out the project.

The robot comprises a camera, some movement motors, some camera movement servos and some ultrasonic sensors. The student can control these peripherals thanks to a specially developed library.

The system is based on the Simplelink MSP432P401R board by Texas Instruments and a Raspberry Pi 3 minicomputer. The former runs on the TI-RTOS operating system and the latter on Raspbian.

Students connect to the robot through a browser. They can upload programs for their execution, operate the robot directly, control the camera, and so on. Python is the programming language chosen for students to create their programs. Higher level students can access through the SSH protocol. They may study, propose changes or add some functions to the system.

Taking into account the objectives set out in its origin, the result has been a fully functional mobile vehicle. The project is technically and economically feasible.

Índice de contenido

Índice de ilustraciones

Índice de tablas

1 Introducción

1.1 Contexto y justificación del trabajo.

El siguiente documento es la memoria que sintetiza todo el trabajo realizado en el TFG del Grado en Ingeniería Informática de la Universitad Oberta de Catalunya. El Trabajo final de grado es una asignatura pensada para realizar un trabajo de síntesis de los conocimientos adquiridos en otras asignaturas. Requiere que se pongan en práctica conjuntamente en un trabajo concreto.

Este trabajo te enmarca en el área de sistemas empotrados. Además de conceptos básicos de diseño de software, otras asignaturas relacionadas con los sistemas operativos, estructura de computadores o redes de computadores son claves en este área.

Por un lado se se pensó en un trabajo en el que poder aplicar todo ese tipo de conocimientos. Por otro lado, se ha intentado que sea una aplicación que se pueda utilizar en un escenario concreto.

En los últimos años se está hablando mucho de la introducción de la programación en las aulas. Materias como Tecnología o Informática ya la incluyen como parte de sus contenidos en los planes de estudios.

Los que hemos aprendido a programar sabemos que los inicios son muy duros. Además los primeros programas no suelen ser muy motivadores. Uno se limita a hacer cálculos sencillos y mostrar los resultados en la salida estándar. Para esto surgen iniciativas como Scratch[1], un lenguaje de programación que hace más atractivo iniciarse en el mundo de la programación. Cuenta con un vistoso entorno de programación en el que se pueden crear historias interactivas, juegos y animaciones.

Por otro lado la robótica recreacional y educativa está implantándose cada vez más en las aulas.

Que tal aprender a programar con un robot?
Para mí, un elemento clave en el aprendizaje es la motivación. Así que la idea es crear un robot móvil con el que alumno pueda iniciarse de forma amena en el mundo de la programación de este tipo de dispositivos.

1.2 Descripción del trabajo.

Se pretende diseñar y construir un vehículo robot que sirva de plataforma de aprendizaje. Los alumnos pueden escribir código en Python y subirlo al robot para probarlo. El robot es capaz de desplazarse de forma autónoma. Tiene cuatro ruedas, cada una de ellas impulsada por un motor DC. También tiene instalados sensores de ultrasonidos para detectar obstáculos. Por último, porta una cámara con la que puede tomar imágenes o vídeo. Esta última está articulada con dos servos que le van a permitir elegir el encuadre. El sistema está conectado en todo momento a Internet. A través de una página web se pueden subir los programas al robot, o bien enviar comandos directamente.

El alumno puede acceder al control de los distintos periféricos mediante una librería desarrollada para este fin.

1 https://scratch.mit.edu

El sistema está basado en la placa SimpleLink MSP432P401R LaunchPad[2] de Texas Instruments y un mini ordenador Raspberry Pi 3[3]. El primero corre el sistema operativo TI RTOS[4] y el segundo corre el sistema operativo Raspbian[5].

Los alumnos se conectan al robot por medio de un navegador. Pueden subir los programas para su ejecución, o bien manejar el robot directamente, controlar la cámara etc. El lenguaje escogido para que los alumnos creen los programas es Python. Alumnos de mayor nivel pueden acceder mediante el protocolo SSH. Podrán estudiar, proponer cambios o añadir funciones al sistema.

El resultado ha sido un vehículo móvil plenamente funcional para los propósitos que se marcaron al inicio. El proyecto es viable tanto desde el punto de vista económico como desde el punto de vista técnico.

1.3 Objetivos del TFC

1.3.1 Objetivos principales

- Detección de obstáculos y medida de distancias por medio de sensores de ultrasonidos.
- Desplazamiento del vehículo por medio de Motores DC.
- Movimiento de la cámara. Control con servos PWM[6].
- Cuantificar el desplazamiento del vehículo por medio de un encoder incremental.
- Comunicación Raspberry – MSP432P401R mediante UART[7].
- Librería API para el control de los comandos del MSP432P401R mediante UART.
    - Comandos motores avance.
    - Comandos servos encuadre cámara.
    - Comandos lectura sensores ultrasonidos.
- Servidor WEB y página de acceso al robot.
    - Lectura de sensores.
    - Control de los motores de movimiento.
    - Control de los servos de la cámara.
    - Obtención de imágenes desde la cámara.
    - Ejecución de programas subidos por el usuario a la plataforma.
- Conexión wifi MSP432P401R.
- Encendido/apagado de la Raspberry a través del MSP432P401R.

2 http://www.ti.com/tool/MSP-EXP432P401R
3 https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
4 http://www.ti.com/tool/TI-RTOS-MCU
5 https://www.raspberrypi.org/documentation/raspbian/
6 Pulse Width Modulation
7 Stands for Universal Asynchronous Receiver/Transmitter

1.3.2 Objetivos Secundarios

- Gestión en remoto de los programas: subir, bajar, borrar, ver fichero de salida.
- Retransmisión de vídeo en *streaming*[8].

1.4 Enfoque y método seguido.

Como cerebro del sistema se ha elegido la Raspberry Pi 3, ya que esta ejecuta un sistema operativo de propósito general con muchas herramientas y software libre. Esto hace que el sistema desarrollado se pueda reutilizar más adelante para un sin fin de proyectos. A diferencia de un sistema embebido, no hay que re-programar la placa para cambiar o añadir funcionalidades. El mismo sistema operativo es un sistema dinámico que permite cambiar o ampliar las funciones.

Se ha elegido la placa SimpleLink MSP432P401R LaunchPad de Texas Instruments para el control de los motores, servos y sensores. Por un lado porque forma parte del material de la asignatura, y por otro lado su sistema operativo de tiempo real y los periféricos que integra, *timers,* PWM, ADC, etc, lo hacen idóneo para este tipo de tareas. Así, me parece buena idea que un sistema embebido haga las veces de interface entre esos periféricos y la Raspberry comentada anteriormente.

Para la parte electro-mecánica del robot podría haber partido de uno de los muchos kits disponibles en el mercado. Pero para mí era un reto personal hacer todo el diseño desde cero.

Con todo esto, creo que el sistema desarrollado, aparte de cumplir con los objetivos esenciales del sistema, es un sistema muy versátil y escalable, que sirve de base para otros muchos proyectos.

8 Streaming es la distribución digital de contenido multimedia a través de una red de computadoras

1.5 Planificación del trabajo.

Inicialmente se hizo un plan de trabajo. En este se definió el trabajo concreto que se llevaría a cabo y cuales serían los objetivos principales y secundarios . Por último se hizo una descomposición de tareas y fechas clave.

La planificación se dividió en tres fases que coinciden con las fechas clave.

En la **fase 1** se trabajan los objetivos relacionados con el hardware; sensores de ultrasonidos, motores, servos, etc.

En la **fase 2** se trabaja la comunicación entre la Raspberry y el MSP432P401R. Esto incluye la definición de un protocolo serie y una librería que da acceso a los distintos comandos del robot.

En la **fase 3** se desarrolla todo lo relacionado con el acceso vía web. Se implementa la aplicación web con diferentes scripts para manejar el robot.

A continuación la tabla 1 muestra los objetivos ordenados por fases.

Fase	Nº	Objetivos Principales por Prioridad	Estado
Fase 1 Inicio:19/03/2019 Final:16/04/2019	1	Control de los sensores de ultrasonidos	Finalizado
	2	Control de los Motores DC PWM	Finalizado
	3	Control de los servos PWM. Movimiento de la cámara.	Finalizado
	4	Control del encoder contador de pasos	Finalizado
Fase 2 Inicio:16/04/2019 Final:14/05/2019	7	Comunicación Raspberry – MSP432P401R mediante UART.	Finalizado
	8	Librería API para el control de los comandos del MSP432P401R mediante UART	Finalizado
Fase 3 Inicio:14/05/2019 Final:01/06/2019	9	Servidor WEB y página de acceso al robot.	Finalizado
	5	Conexión wifi MSP432P401R	Pendiente
	6	Encendido/apagado de la Raspberry	Pendiente
	10	Gestión en remoto de los programas a ejecutar	Finalizado
	11	Retransmisión de vídeo en streaming	Finalizado

Tabla 1: Objetivos ordenados por fases

En la tabla 2 se muestra la planificación de las distintas tareas de cada una de las fases del proyecto.

Fase		Descripción Tarea	Inicio	Final		Estado
Fase 1 Inicio:20/03/2019 Final:16/04/2019	1	Montaje en el chasis de los diferentes componentes del sistema. Conexión de sensores y actuadores con los diferentes módulos y sistemas de control. Montaje del sistema de alimentación.	20/03/19	24/03/19	5	Finalizado
	2	Programación en el MSP432P401R del control de los sensores de ultrasonidos	25/03/19	01/04/19	8	Finalizado
	3	Programación en el MSP432P401R del control de los motores DC,	02/04/19	09/04/19	7	Finalizado
	4	Programación en el MSP432P401R del control de los servos de la cámara	10/04/19	14/04/19	4	Finalizado
	5	Programación en el MSP432P401R de sensor encoder cuenta pasos	15/04/19	16/04/19	2	Finalizado
Fase 2 Inicio:17/04/2019 Final:14/05/2019	6	Instalación y configuración del sistema operativo en la Raspberry. Conexión wifi. Interprete de Python. Conexión SSH. Configuraciones varias	17/04/19	22/04/19	6	Finalizado
	7	Configuración del hardware y software de comunicación Raspberry – MSP432P401R mediante UART. Especificación del formato de los mensajes.	23/04/19	28/04/19	6	Finalizado
	8	Pruebas comunicaciones	29/04/19	02/05/19	4	Finalizado
	9	Programación de la API. Lectura de sensores	03/05/19	06/05/19	4	Finalizado
	10	Programación de la API. Ordenes de desplazamiento. Motores DC	07/05/19	11/05/19	5	Finalizado
	11	Programación de la API. Ordenes de movimiento de la cámara	12/05/19	14/05/19	3	Finalizado
Fase 3 Inicio:15/05/2019 Final:01/06/2019	12	Instalación de Apache + PHP en la Raspberry.	15/05/19	15/05/19	1	Finalizado
	13	Prueba de scripts. Acceso al servidor. Ejecución de comandos API.	16/05/19	16/05/19	1	
	14	Diseño de la web HTML y programación de scripts php Contenido: -lectura de sensores -control de los motores movimiento -control de los servos cámara -obtención de imágenes desde la cámara	17/05/19	22/05/19	6	Finalizado
	15	Conexión módulo wifi al MSP432P401R Programación de un thread para manejar esta conexión	23/05/19	24/05/19	2	Pendiente
	16	Implementar el sistema de encendido y apagado de la Raspberry: Programación MSP432P401R. Recibir orden desde Internet Programación Interrupción Raspberry para gestionar el apagado	25/05/19	29/05/2019	5	Pendiente
	17	Estudio del consumo	30/05/19	01/06/19	3	Pendiente
18		Memoria	02/06/19	08/06/19	7	Finalizado
19		Presentación	09/06/19	15/06/19	7	Pendiente

Tabla 2: Tareas

1.5.1 Diagrama de Gantt

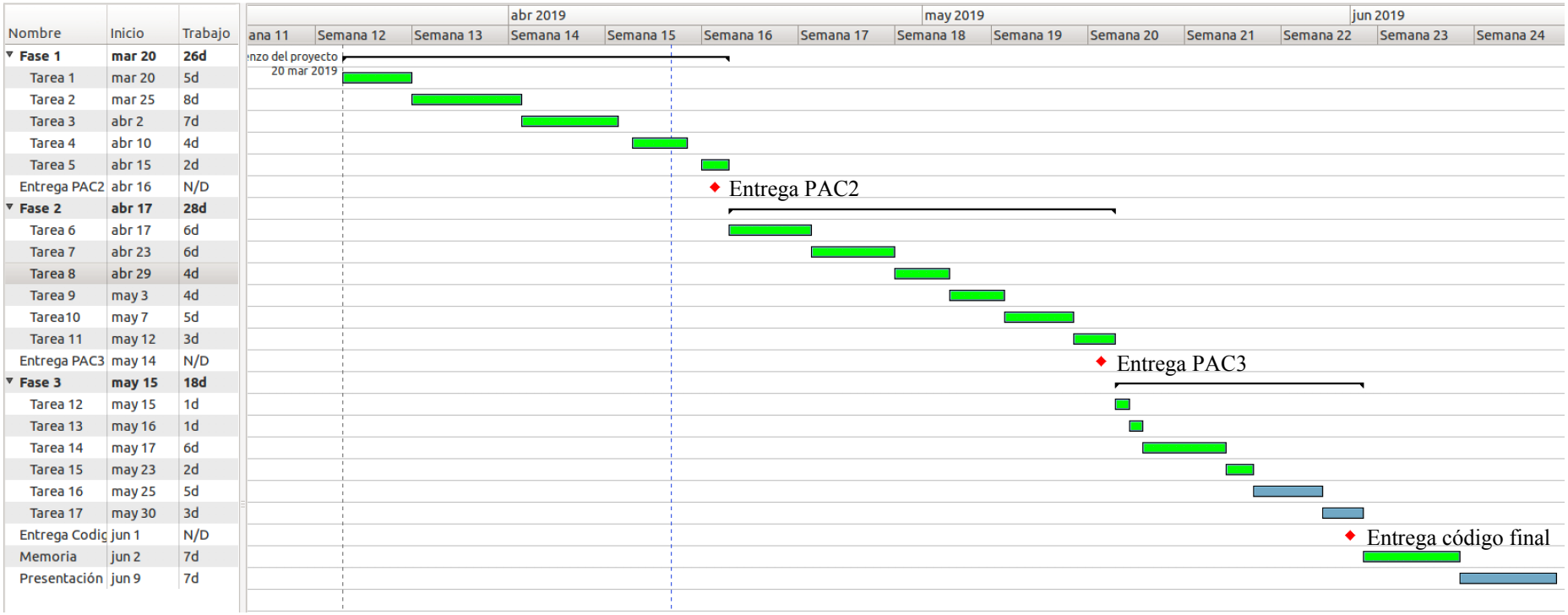


Ilustración 1: Diagrama de Gantt

1.5.2 Desviación de la planificación

Para las fases 1 y 2 se siguieron las tareas planificadas en los tiempos programados. Sin embargo, de la fase 3 no se han acabado las tareas 15,16, y 17. La tarea 13 y 14 relacionadas con la aplicación web necesitaron muchas más horas de las previstas. Se prefirió seguir trabajando en la aplicación web, pues es una de las partes más importantes del proyecto.

Las tareas 15 y 16 iban destinadas a dotar al Msp432 de una segunda conexión wifi, de forma que pudiese recibir algunas ordenes básicas directamente a través de internet. Una de ellas sería controlar el encendido, apagado o reinicio de la Raspberry, comprobar su estado etc. El objetivo era hibernar el robot llevándolo a un estado de bajo consumo , desactivando la Raspberry y cualquier otro dispositivo de mayor consumo. Mientras la Raspberry puede llegar a consumir 500 mA, el Msp432 puede consumir menos de 10 mA. Lo cual puede aumentar mucho la autonomía de las baterías del robot.

En cualquier caso no se ha podido explorar esta vía.

1.6 Recursos empleados.

1.6.1 Herramientas Software

- **Paquete de ofimática OpenOffice:** Usado para escribir este trabajo y confección de los esquemas.
- **Pixelmator.** Retoques y montajes fotográficos.
- **Entorno de desarrollo CSS de Texas Instruments:** Para programar la placa MSP-EXP432P401R.
- **Sistema Operativo TI-RTOS:** Instalado en la placa MSP-EXP432P401R.
- **Librerías TI-Drivers:** Se han usado para desarrollar la aplicación en el MSP-EXP432P401R.
- **Sistema operativo Raspbian:** Sistema operativo GNU/Linux derivado de Debian instalado en la Raspberry.
- **Web Server Apache:** Software que implementa un servidor web. Instalado en la Raspberry.
- **Open SSH:** Software que implementa el protocolo SSH. Usado para la conexión en remoto con la Rasp Berry. De esta forma no necesitamos ningún periferico de entrada y salida (teclado, pantalla) para interactuar con la Raspberry.
- **Interprete de Python:** Usado para ejecutar los scripts de Python.
- **Editor de código Atom:** Lo he usado sobre todo en el desarrollo de la aplicación web.
- **Compilador GCC:** Para compilar ficheros C o C++
- **Editor de textos Joe:** Para editar ficheros de configuración, scripts, etc, desde la línea de comandos.
- **Rsync:** Lo he usado para sincronizar ficheros entre el Macbook y la Raspberry.

1.6.2 Herramientas hardware

- Soldador.
- Multímetro digital..
- Dremel.
- Destornillador, pinzas, etc.

1.6.3 Material usado para construir el robot.

- 1 Raspberry Pi 3
- 1 SimpleLink MSP432P401R LaunchPad
- Advanced Emulation Kit for SimpleLinkTM Wi-Fi® CC31xx Boosterpack
- 1 Tarjeta SD (para el sistema operativo Raspbian)
- 3 Sensores ultrasonidos HC-SR04
- 4 Motores DC
- 2 mini Servos sg90
- 1 Cámara para Raspberry
- 1 Chasis del robot
- 1 Módulo relé
- 2 Módulo opto-acopladores
- 2 Módulo *driver* potencia motores
- 1 Convertidor de nivel BSS138
- 1 Soportes sensor ultrasonidos
- 1 Soporte articulado cámara
- 1 caja para baterías
- 1 Power Bank
- 6 batería de litio recargable 18650 3,7 v 3400 mah
- Varios: cables, conectores, tornillos etc

1.7 Productos obtenidos.

El resultado es un vehículo robot funcional, capaz de realizar la mayoría de funciones que se han planteado en los objetivos iniciales:

- Interactuar con él desde internet
- Subir archivos de programa y ejecutarlos.
- Ver vídeo en *streaming*.
- Tomar fotografías.

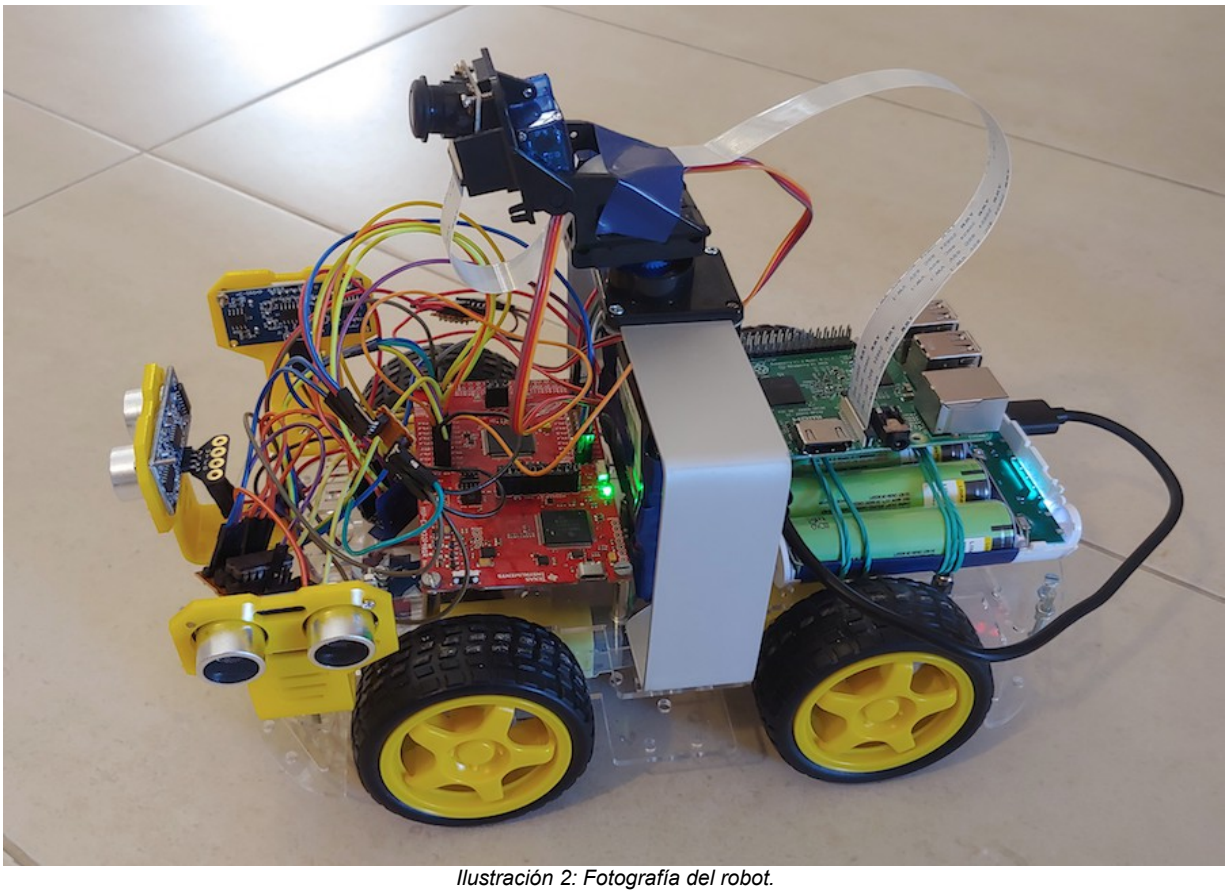


Ilustración 2: Fotografía del robot.

1.8 Breve descripción de otros capítulos.

El **capitulo 2** esta dedicado a los antecedentes y estado del arte. En él se discute las opciones similares al proyecto que se puedan encontrar en el mercado.

En el **capítulo 3** se hace una descripción de cómo funciona el sistema. Describiendo cómo es el robot y como se comporta. También cómo un usuario interacciona con el sistema. Este capítulo da al lector una idea de todo lo que puede hacer el sistema.

El **capítulo 4** describe cómo y qué dispositivo hace cada función. Está dividido en tres partes. La primera de ellas está dedicada al protocolo que usan la Raspberry el MSP432P401R para comunicarse por el puerto serie. La segunda parte está dedicada al hardware y en ella se describen todas las conexiones y se presentan los esquemas de todos los circuitos. La tercera parte está dedicada al software. Aquí se describen todos los módulos del sistema. A su vez este apartado está dividido en dos partes; una dedicada a los módulos que se ejecutan en el MSP432P401R y otra para los módulos que se ejecutan en la Raspberry.

En el **capítulo 5** se estudia la viabilidad técnica.

En el **capítulo 6** se hace una valoración económica.

En el **capítulo 7** se exponen las conclusiones del trabajo.

2 Antecedentes

2.1 Estado del arte.

Tradicionalmente el uso de la robótica ha estado ligado a la industria. Se trataba de sistemas muy caros y para los que se necesitaban grandes conocimientos técnicos.

En los últimos años la robótica ha entrado en el hogar, es fácil encontrar dispositivos robóticos para realizar tareas en la cocina, para la limpieza de la casa, juguetes, etc. Todo esto propiciado por el avance tecnológico y el abaratamiento de los dispositivos.

Surge también el concepto de robótica recreacional y educativa. Se pueden encontrar en la red multitud de proyectos lúdicos relacionados con la robótica. Arduino y Raspberry son las plataformas más usadas en la comunidad *maker*[9].

La introducción de la robótica en las aulas permite trabajar materias como ciencias, tecnología o matemáticas. Pueden parecer juguetes, pero realmente se encuentran un sin fin de actividades educativas que proponer a los alumnos mientras aprenden divirtiéndose.

2.2 Estudio de mercado.

El producto desarrollado esta enfocado al campo de la educación.

En la actualidad existen multitud de dispositivos enfocados al uso en las aulas. Desde educación infantil hasta la educación superior, se pueden encontrar dispositivos que se adaptan a las necesidades de cada edad. Bee bot y Lego WeDo 2.0 por poner unos ejemplos.

Muchos de ellos se venden en forma de kit para que los alumnos ensamblen sus componentes.

En cuanto la forma de programar hay un poco de todo. Los hay que sólo permiten ciertas variaciones preestablecidas en su configuración . Otros se pueden programar con un lenguaje propio. Algunos se pueden programar con Scratch.

Quizá lo más parecido a este proyecto es el Lego Mindstorm EV3[10] MicroPython[11] programing language. Esta plataforma se puede programar con MicroPython[12], una versión de Python específica para microcontroladores.

Aun así, hay una diferencia fundamental. Nuestro sistema ejecuta un sistema operativo de propósito general, abierto y de software libre. Este echo lo hace mucho más potente en cuanto recursos software.

9 La comunidad maker promueve la idea que todo el mundo es capaz de desarrollar cualquier tarea en vez de contratar a un especialista para realizarla.
10 https://www.lego.com/en-us/mindstorms/products/mindstorms-ev3-31313
11 https://education.lego.com/en-us/support/mindstorms-ev3/python-for-ev3
12 https://micropython.org/

3 Descripción funcional

3.1 Robot móvil conectado como plataforma de aprendizaje

Como comentamos en el primer capitulo la finalidad de este proyecto consiste en proporcionar una plataforma de fácil acceso para la introducción a la programación.

La finalidad de este trabajo es el desarrollo de un robot móvil, el cual puede ser usado, por alumnos de secundaria o ciclos formativos de formación profesional, como elemento de motivación para el aprendizaje de la programación de este tipo de dispositivos.

El robot dispone de una cámara, motores para el desplazamiento, servos para el movimiento de la cámara y sensores de ultrasonidos. El alumno puede acceder al control de estos periféricos mediante una librería desarrollada para este fin.

Los alumnos se conectan al robot por medio de un navegador. Pueden subir los programas para su ejecución, o bien manejar el robot directamente, controlar la cámara etc. El lenguaje escogido para que los alumnos creen los programas es Python. Alumnos de mayor nivel pueden acceder mediante el protocolo SSH. Podrán estudiar, proponer cambios o añadir funciones al sistema.

El resultado ha sido un vehículo móvil plenamente funcional para los propósitos que se marcaron al inicio. El proyecto es viable tanto desde el punto de vista económico como desde el punto de vista técnico.

3.1.1 Diagrama general de los elementos que componen el sistema.

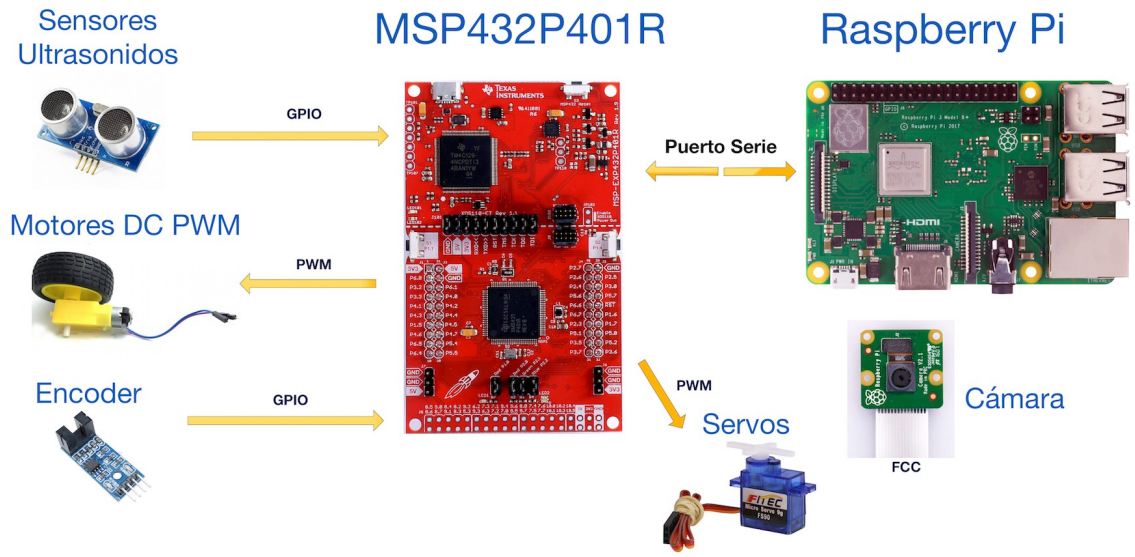


Ilustración 3: Dispositivos integrados en el sistema

Como se aprecia en la imagen, se cuenta con dos sistemas computacionales; una Raspberry, y una placa MSP432P401R.

La placa MSP432P401R ejecuta el sistema operativo TI RTOS. Se trata de un sistema operativo de tiempo real, idóneo para el control de los motores, servos y sensores de ultrasonidos.

A placa la placa MSP432P401R se conectan los siguientes periféricos:

- **Sensores de ultrasonidos**: Permiten detectar obstáculos y medir distancias. Se han instalado tres sensores de este tipo con distinta orientación.
- **Motores DC:** Proporcionan la tracción al robot para desplazarse, disponen de mecanismo de reductora. Hay cuatro, uno por cada rueda.
- **Servo motores:** Por medio de una articulación mueven el encuadre de la cámara. Hay dos, uno para el movimiento sobre el eje horizontal y otro para el eje vertical.
- **Encoder incremental:** Está acoplado a una de las ruedas y permite controlar la distancia que debe avanzar el robot en cada caso.

La Raspberry dota al robot de una mayor capacidad de computo y comunicación. Ejecuta el sistema operativo Raspbian, una distribución derivada del sistema Debian. Se trata de un sistema operativo de propósito general GNU/Linux. Esto proporciona un gran potencial en cuanto a la variedad de software que puede ejecutar.

La conexión a la red se realiza mediante la Wifi que tiene integrada la Raspberry. Mientras la comunicación entre las dos placas se realiza con mediante un puerto serie.

3.1.2 Descripción funcional

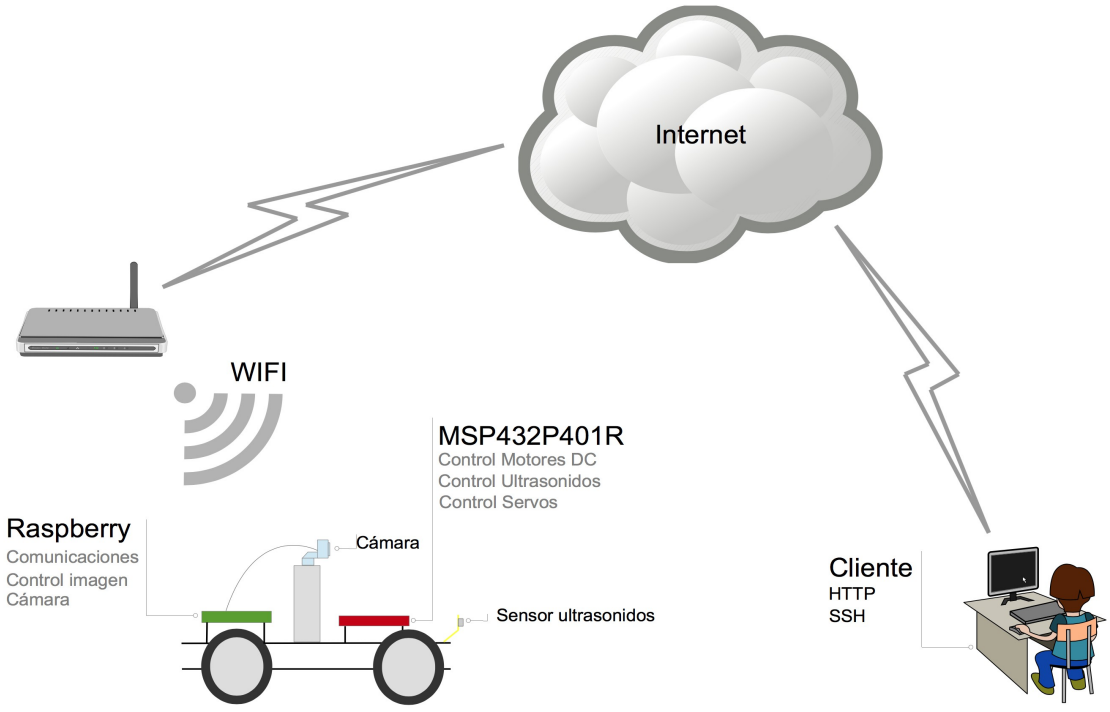


Ilustración 4: Diagrama funcional del sistema.

Un usuario accede al robot desde internet. Puede establecer una conexión HTTP a través de un navegador para usar la aplicación web, o bien conectarse mediante un cliente SSH.

En el primer caso el usuario interactúa con la aplicación web. Esta permite realizar las siguientes acciones:

- Manejar el desplazamiento del robot.
- Mover el encuadre de la cámara.
- Actualizar la lectura de los sensores de ultrasonidos periódicamente.
- Tomar fotografiás con la cámara.
- Ver video de la cámara en *streaming.*
- Gestionar los programas: Subir, bajar, borrar.
- Ejecutar los programas subidos.
- Ver la salida generada por los programas en un archivo *output*.

En el segundo caso el usuario tiene acceso a todo el sistema operativo. Las posibilidades son mayores, ya que se tiene al alcance todas las herramientas y software que proporciona el sistema operativo Raspbian. Se requiere que el usuario tenga cierto nivel sobre sistemas operativos GNU/Linux. Dispondrá de la librería desarrollada que da acceso a todas las funciones del robot, pero además puede introducir nuevas funcionalidades en el sistema.

3.1.3 Bloques funcionales.

En la siguiente ilustración se muestran los bloques funcionales del sistema.

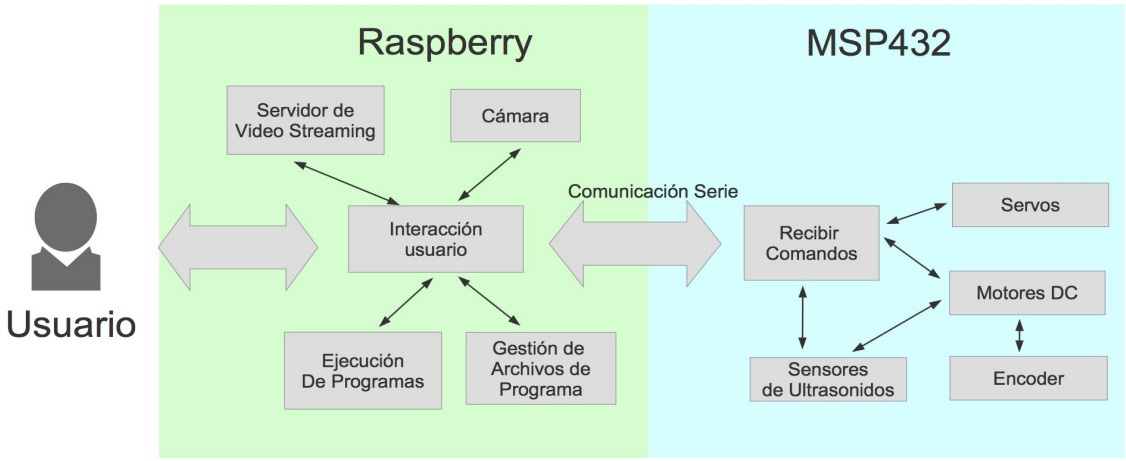


Ilustración 5: Bloques funcionales.

La Raspberry se encarga ejecutar los siguientes módulos:

- **Interacción usuario:** A partir de la interacción del usuario con el sistema se van distribuyendo las tareas entre los distintos módulos. Si el módulo implicado se encuentra en el MSP432P401R, se emplea el puerto serie para retransmitir los comandos.

- **Servidor de video streaming:** se ha implementado un servicio en el sistema para distribuir video en streaming. Este módulo se encarga del control de este servicio. Se puede ver el video a través del navegador.

- **Cámara:** Este módulo controla la cámara. Se pueden tomar fotografiás, estas se almacenan en el robot. Desde la página web se tiene un enlace que da acceso al directorio donde se almacenan las fotografiás.

- **Gestión de archivos de programa:** este módulo se va a ocupar de almacenar los programas subidos por el usuario, bajarlos o borrarlos.

- **Ejecución de programas:** se encarga de la ejecución de los programas subidos por el usuario. Cuando se ejecuta un programa se genera un fichero de salida asociado, el usuario puede consultar la salida de ese programa. Si el programa se ejecuta con normalidad el fichero contendrá la salida estándar generada por el programa. Si se produce un error de ejecución el archivo contendrá los mensajes de error arrojados por el interprete de Python.

El MSP432P401R se va a encargar de ejecutar los siguientes módulos.

- **Recibir comandos:** este módulo se encarga de la comunicación con la Raspberry y distribuir las ordenes recibidas a los demás módulos para ser materializadas.

- **Motores DC**: controla los motores DC. A partir de los parámetros que contiene el comando acciona las señales necesarias para llevar a cabo las diferentes acciones.

  Los motores pueden rodar hacia delante o hacia atrás.

  Para obtener un giro a la derecha, las ruedas del lado izquierdo ruedan hacia delante, mientras que las del lado derecho ruedan hacia atrás.

  Se pueden seleccionar 5 velocidades. 1 es la velocidad mínima y 5 es la máxima.

- **Encoder:** lleva la cuenta de los pasos que avanza el disco del encoder. Este disco está situado en una de las ruedas. Para iniciar la cuenta se pone un contador a cero y se le indica cual es el final de la cuenta. Cuando se alcanza el final se detiene el avance.

- **Ultrasonidos:** se encarga del control de los sensores de ultrasonidos. Esta lectura se produce cada 250ms. El valor de cada lectura es almacenado. Otros módulos pueden consultar estos valores en cualquier momento.

  Si el valor de una lectura sobrepasa un cierto límite de seguridad definido en el programa, este módulo alertará al módulo de control de motores DC para que, en el caso de que el robot se esté moviendo hacia delante, pare los motores y evitar que el robot choque contra un obstáculo.

- **Servos:** Controla la posición de los servos que mueven la cámara. Un servo se usa para el movimiento horizontal y otro para el vertical. Los servos tienen un recorrido de 180º. Se ha dividido este recorrido en nueve posiciones. La posición 5 indica 0º. La posición 1 indica -90º, y la posición 9 indica 90º.

3.1.4 Cómo es la red.

El sistema se conecta a la red mediante la Wifi que integra la Raspberry. El protocolo usado es el 802.11b. También podemos conectar un cable ethernet a la Raspberry. El sistema operativo Raspbian incluye toda la pila de protocolos TCP/IP. En la capa de aplicación encontramos el protocolo HTTP y SSH.

Si queremos conectarnos desde un punto en Internet fuera de nuestra red local podemos redirigir los puertos en el router a la dirección del robot.

Por otro lado la comunicación entre la Raspberry y el MSP432P401R se realiza a través de un puerto serie. En este caso se ha implementado un protocolo de enlace para llevar a cabo el proceso de comunicación.

3.1.5 Casos de uso.

A continuación se muestran un caso de uso de las distintas opciones que debe tener la aplicación web. Se ha dividido el esquema para una mejor presentación. Todas las acciones las realizaría un usuario interaccionando con la aplicación web.

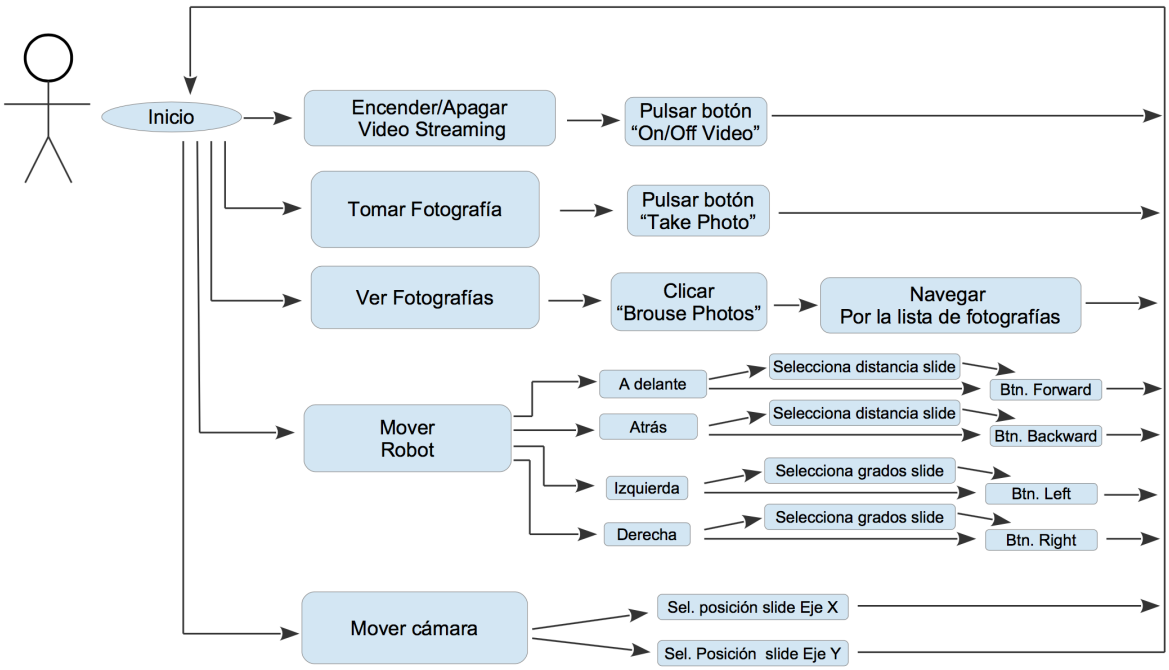


Ilustración 6: Caso de uso parte 1

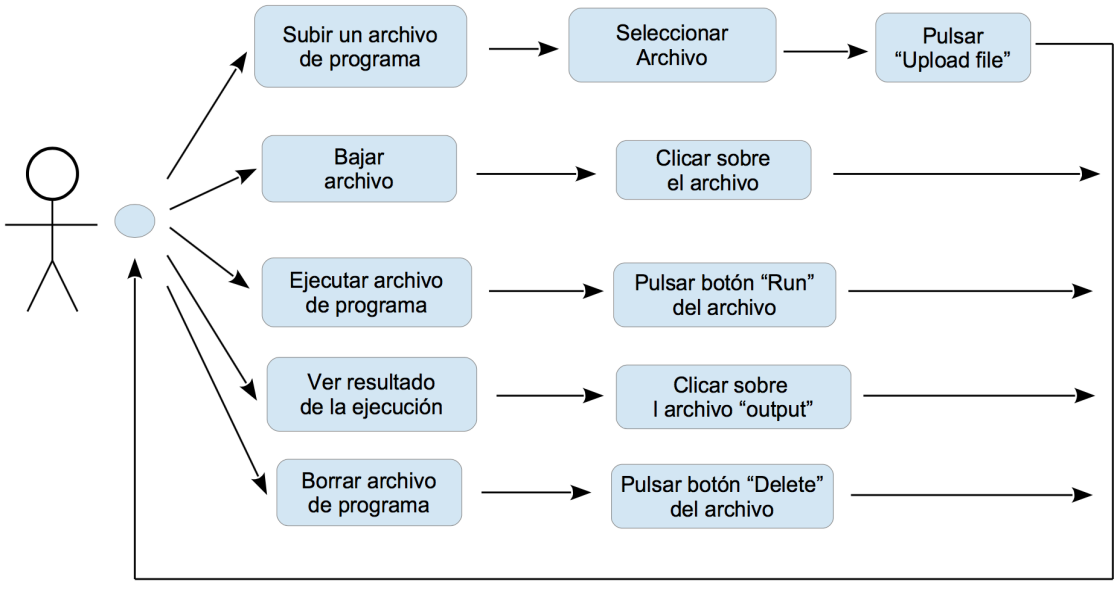


Ilustración 7: Caso de uso parte 2

3.1.6 Aplicación web.

La estructura de la aplicación web es muy sencilla, se tiene acceso a todas las funciones del robot desde una única página web.

Ilustración 8: Captura de pantalla aplicación web

En la imagen se distinguen tres zonas:

- **Zona Izquierda:** En ella se ubica la imagen de la cámara y sus controles asociados. Los controles permiten poner el marcha el vídeo en streaming y tomar fotografías.
- **Zona Central:** Controles para accionar los mecanismos de los motores y servos. Además se muestra una lectura actualizada de los sensores cada tres segundos.
- **Zona derecha:** Es la zona dedicada a la gestión de archivos de programa. Se compone de un formulario que permite subir archivos y una lista de los que se han subido. Con los archivos subidos se pueden efectuar tres acciones:
  - Bajar el archivo.
  - Borrar el archivo.
  - Ejecutar el programa.

La página tiene un diseño *responsive design* para adaptarse a distintos tamaños de dispositivos móviles.

4 Descripción detallada

4.1 protocolo de comunicación entre la Raspberry y el MSP432P401R.

Para la comunicación entre la Raspberry y el MSP432P401R se usa el puerto serie. Se ha implementado un protocolo de enlace sencillo, del tipo parada y espera.

4.1.1 Tipos de mensaje.

Tenemos tres tipos de mensajes:

Comandos simples: Los comandos simples envían una orden al MSP432P401R para su ejecución, sólo esperan una confirmación del tipo ACK o NACK. Son los mensajes relacionados con el manejo de los motores y servos. Ver ilustración 9 figura a.

Mensajes de confirmación: Confirman la recepción del los comandos, la respuesta es ACK si la recepción del comando ha sido correcta, y NACK si ha sido incorrecta.

Comandos de solicitud de información: Los comandos de tipo solicitud, además de recibir la confirmación, esperan cierta información como respuesta. Es el caso, por ejemplo, de solicitar la lectura de los sensores de ultrasonidos. En este caso se espera una respuesta con información de los sensores. Ver ilustración 9 figura c.

Si uno de los comandos recibe NACK en la confirmación, se procederá a repetir el envío del comando. Se reintentará por tres veces si el problema persiste.

En el siguiente diagrama se muestran unos ejemplos de cómo se produce el intercambio de mensajes.

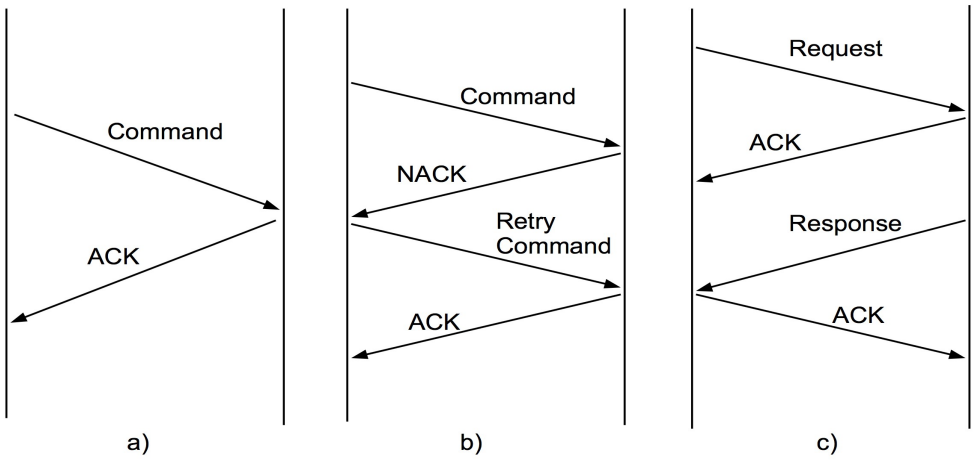


Ilustración 9: Intercambio de mensajes. Protocolo serie.

En el lado de la Raspberry las funciones que llevan a cabo el proceso de comunicación están implementadas en la librería robolib.py, se describe con más detalle en el siguiente apartado. El dispositivo usado es /dev/serial0.
En el lado del MSP432P401R estas funciones forman parte del módulo contrlUART.c. El dispositivo usado es el UART1.
La velocidad del puerto es de 115200 baudios[13].

13 Número de símbolos por segundo en un medio digital.

4.1.2 Algoritmos del protocolo de comunicación.

A continuación se expone mediante diagrama de flujo el comportamiento del protocolo de comunicación tanto en la parte de la Raspberry como en el lado de MSP432P401R.

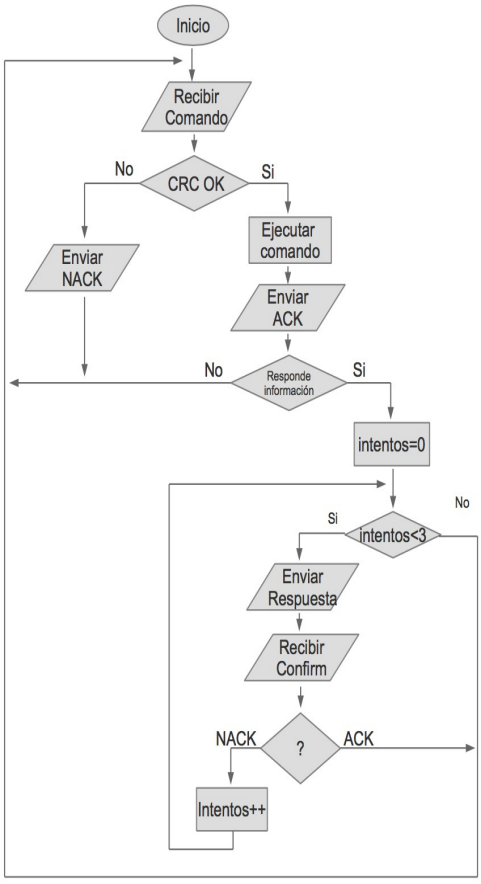


Ilustración 10: Diagrama de flujo del proceso de comunicación en el lado del MSP432P401R.

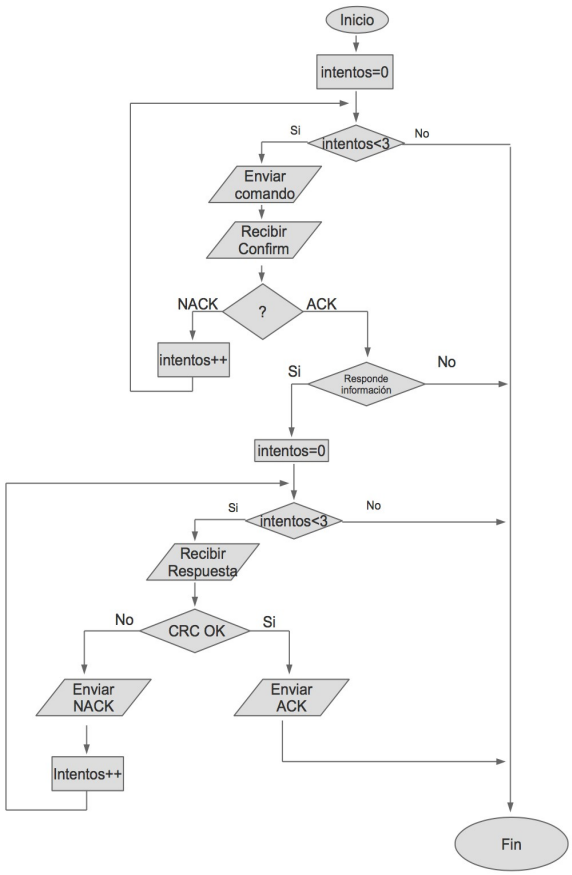


Ilustración 11: Diagrama de flujo del proceso de comunicación del lado de la Raspberry.

4.1.3 Formato de las tramas.

En general una trama esta formada por los siguientes campos:

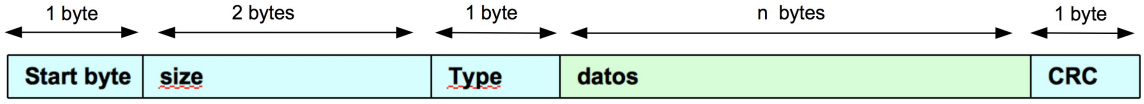


Ilustración 12: Formato de trama

- **Start byte**: es usado para indicar e inicio de una trama. Es útil para sincronizar el inicio de transmisión de una trama.
- **Size:** es el tamaño en bytes de toda la trama.
- **Type:** es el tipo de mensaje. Es útil para que la aplicación sepa el formato del campo datos.
- **CRC:** es un campo de verificación de trama, sirve para detectar la integridad de la trama recibida.
- **Datos:** son los parámetros que necesita cada tipo de comando.
- El campo datos tiene tamaños distinto dependiendo del tipo de mensaje. Son los datos que pasan a la capa de aplicación.

En la siguiente ilustración se especifican cada uno de los tipos de mensaje:

Tipo	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11
ACK	Start byte	Size(5)		Type (1)	CRC						
NACK	Start byte	Size(5)		Type (2)	CRC						
Comando Motores DC	Start byte	Size(11)		Type (5)	f/b	Speed Left	f/b	Speed Right	Steps		CRC
Comando Servos	Start byte	Size(7)		Type (6)	Pos X	Pos Y	CRC				
Lectura Sensores	Start byte	Size(5)		Type (7)	CRC						
Respuesta, Lectura Sensores	Start byte	Size(11)		Type (8)	Value S1		Value S2		Value S3		CRC

Ilustración 13: Formato de los distintos tipos de comandos.

Los datos que encapsula cada tipo de mensajes se describen en los apartados de los módulos correspondientes a la acción que desempeñan.

4.1.4 Cálculo del CRC.

El código de redundancia cíclica es es un código de detección de errores usado altamente en redes digitales. Esta basado en el cálculo del residuo en una división de polinomios.

En este caso se ha implementado un algoritmo con polinomio de grado 8. Por lo que el resultado de la operación es del tamaño de un byte.

El algoritmo usado algoritmo está optimizado para lograr mayor velocidad de cálculo teniendo precalculados los coeficientes de la operación XOR.

Se ha codificado en en lenguaje C. En el lado del MSP432P401R se ha integrado directamente en el proyecto CCS.

En el lado de la Raspberry se ha creado un *wrapper*, por medio de la librería *ctypes*. De esta forma se puede llamar a la función desde el código Python. Así la función que calcula el CRC también ha quedado integrada en la librería robolib.c.

El fichero *libcrc8.so* es la librería de enlace dinámico obtenida al compilar el fichero *crc8.c*. El *wrapper* se crea con el fichero *crc8.py*. Este último es el que se incluye en la librería *robolib.py*.

4.2 Descripción del hardware

El siguiente esquema muestra una visión general de los distintos módulos que componen el sistema.

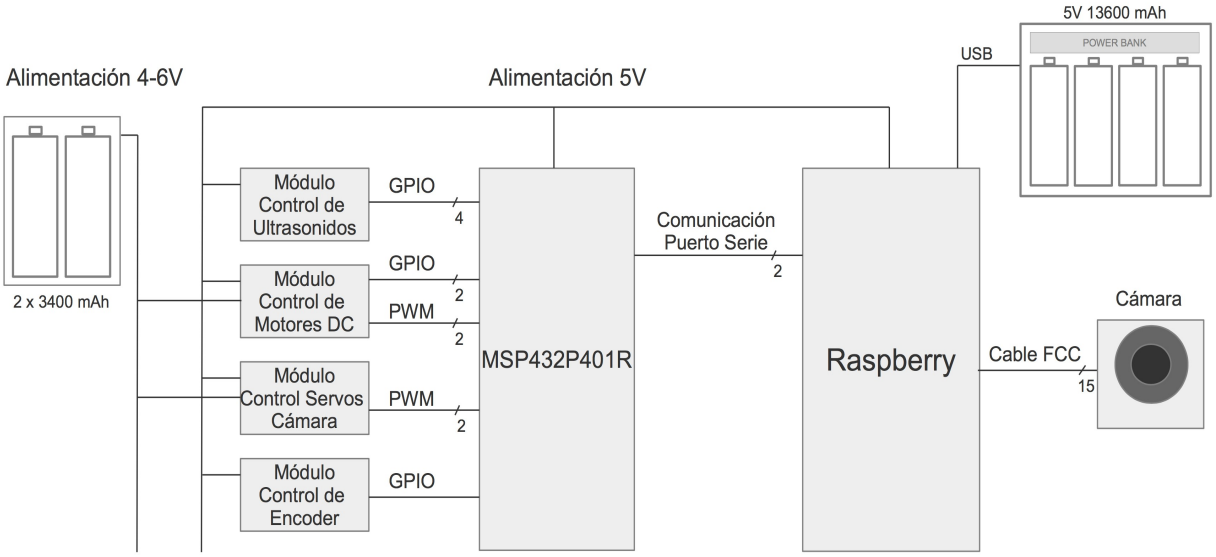


Ilustración 14: Esquema general del circuito eléctrico

4.2.1 Sistemas de alimentación

Los motores son elementos que pueden introducir ruido en los circuitos digitales, provocando errores e incluso el deterioro de los mismos. Además, consumen bastante energía. Cuando las baterías empiezan a estar agotadas el accionamiento de estos puede provocar una bajada de tensión y el mal funcionamiento de el MSP432P401R y la Raspberry.

Por otro lado, aunque el consumo de los motores elegidos para este proyecto no es excesivamente alto, el separar esta alimentación puede hacer más fácil la sustitución de los motores por otros más potentes en el futuro, contribuyendo a la escalabilidad del sistema.

Por todo esto se ha optado por una alimentación separada para los motores DC y servos.

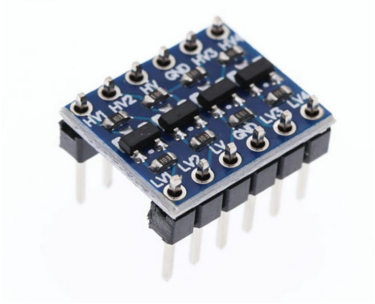
En la ilustración 14 podemos ver los dos sistemas de alimentación. Por un lado tenemos un circuito de 5V. Esta tensión es proporcionada por el *power bank*, el cual se conecta a la Raspberry mediante un cable USB. A su vez la Raspberry proporciona alimentación de 5V al MSP432P401R por uno de los pines dedicados a este fin. Algunas partes de los módulos son conectadas también a este circuito de 5V. Esto último se verá con más detalle en la descripción de cada módulo.

El Power Bank es como los que se usan para alimentar teléfonos móviles o tablets. Contiene 4 pilas formato 18650 y un circuito para regular el voltaje y la carga de las mismas. Proporciona una autonomía ente 8 y 12 horas dependiendo del uso.

Por otro lado contamos con un circuito que proporciona 4V a los módulos que manejan los motores DC y los servos. Se usan dos baterías en formato 18650 en paralelo. También se podría alimentar a 6 voltios con cuatro pilas AA en serie.

4.2.2 Niveles de voltaje

El MSP432P401R y la raspbery PI son muy similares en cuanto a los niveles de voltaje que manejan. Las dos placas se alimentan a 5 voltios. Sin embargo los pines de los puertos de entrada/salida trabajan a 3.3 voltios, tanto si están configurados como entrada o como salida. Lo más común es encontrar dispositivos que trabajan a 5 voltios a precios económicos (aunque cada día hay más dispositivos capaces de trabajar con a 3.3V). Esto hace que a veces tengamos que adaptar esas señales de 3.3v a 5v y viceversa. Para conseguirlo se pueden emplear unos módulos convertidores de nivel. Estos son bidireccionales.



Estos módulos estos son pequeños y económicos.

4.2.3 Tabla de puertos MSP432P401R

La siguiente tabla muestra la asignación de puertos en la placa MSP432P401R.

Puerto	Board.h name	Descripción	Configuración
Sensores ultrasonidos			
P6.7	Board_CAPTURE1	Echo. Todos sensores	Entrada
P5.1	Board_GPIO_SU1TRIG	Sensor 1 Trigger	Salida
P3.5	Board_GPIO_SU2TRIG	Sensor 2 Trigger	Salida
P3.7	Board_GPIO_SU3TRIG	Sensor 3 Trigger	Salida
Control Motores DC			
P2.6	Board_PWM3	PWM motores Izquierda	Salida
P5.2	Board_GPIO_TRACTION_1	Adelante/Atrás Izquierda	Salida
P2.7	Board_PWM2	PWM motores Derecha	Salida
P3.6	Board_GPIO_TRACTION_2	Adelante/Atrás Derecha	Salida
Control servos cámara			
P2.4	Board_PWM4	Cámara PWM Servo1 eje X	Salida
P2.5	Board_PWM5	Cámara PWM Servo2 eje Y	Salida
UART1			
P3.2	-	RX	Entrada
P3.3	-	TX	Salida

Tabla 3: Asignación de puertos del MSP432P401R

4.2.4 Esquema eléctrico de los sensores de ultrasonidos.

Los sensores de ultrasonidos son dispositivos que se usan para medir distancias. Son muy baratos y sencillos de usar.

Los sensores de ultrasonidos envían una onda acústica que rebota en los obstáculos que se encuentren en su camino. Midiendo el tiempo entre el envío y la recepción de la onda rebotada se puede determinar la distancia que hay hasta el obstáculo.

Una limitación de este tipo de sensores se produce cuando las ondas sonoras rebotan en una superficie oblicua. Como se aprecia en la ilustración 15, la onda rebotada no vuelve a la posición donde se encuentra el receptor.

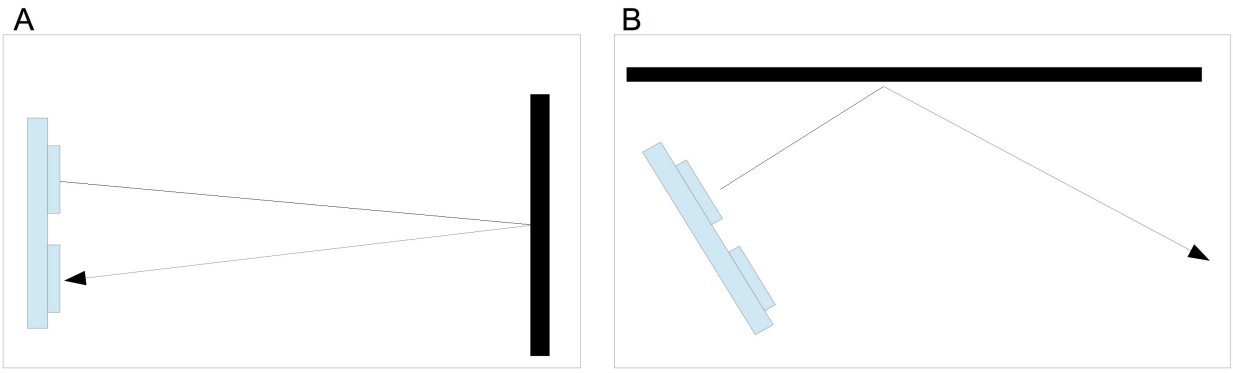


Ilustración 15: Problema al rebotar la onda sonora sobre una superficie oblicua.

El módulo de ultrasonidos se basa en el sensor HC-SR04. Se trata de un sensor muy usado en la comunidad *maker*[14]. El rango de medición teórico va 2 cm a 400 cm con una resolución de 0.3 cm.

Ilustración 16: Módulo

Este módulo trabaja con dos señales; Trigger y Echo. El *Trigger* dispara el pulso sonoro, y el Echo devuelve un pulso proporcional al intervalo de tiempo medido entre la onda enviada y la recibida.

14 La comunidad maker promueve la idea que todo el mundo es capaz de desarrollar cualquier tarea en vez de contratar a un especialista para realizarla.

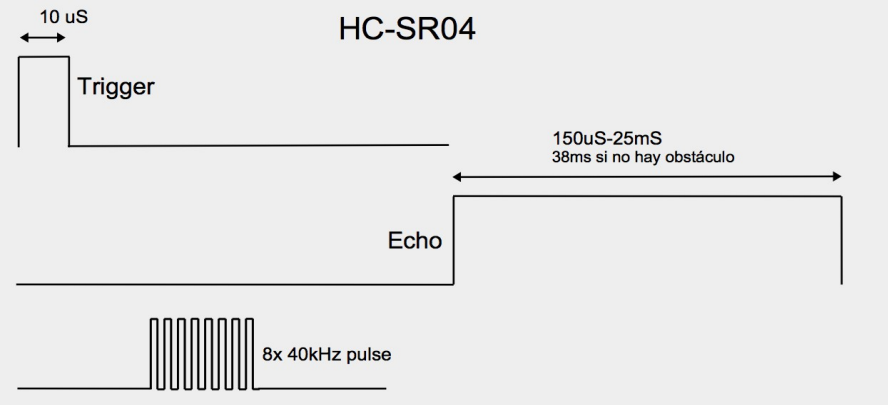


Ilustración 17: Diagrama señales del sensor HC-SR04.

La señal de Trigger sirve para disparar la emisión de la onda sonora. Se debe activar durante al menos 10us.

El sistema dispone de tres sensores de este tipo, por tanto se necesitan tres puertos GPIO configurados como salida para controlar las señales Trigger de cada uno de ellos.

Los sensores no pueden actuar al mismo tiempo, puesto que las señales de cada uno de ellos podría interferir con los demás. Así pues, para medir el ancho del pulso de la la señal Echo se utiliza un sólo pin de entrada en el MSP432P401R. La misma entrada será usada todos ellos . Par conseguirlo se ha añadido una puerta NOR de cuatro entradas para unir la señal de ECHO proveniente de cada sensor en una sola línea. De esta forma, además de liberar dos puertos, se liberan otros recursos, como los *timers*. Esto hace más más fácil la programación, ya que se asigna un sólo timer(TIMER_A2) a la entrada para medir el ancho de pulso de las tres señales.

Por último, para manejar estas dos señales con el MSP432P401R las tenemos que adaptar de 3.3v a 5v y viceversa, ya que el módulo HC-SR04 trabaja con una tensiones de 5v.

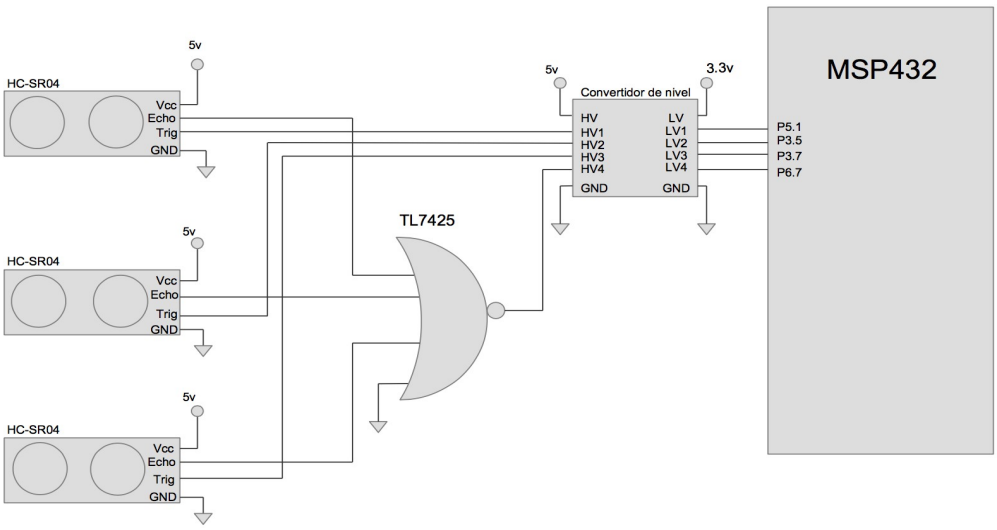


Ilustración 18: Esquema eléctrico del modulo de los sensores de ultrasonidos.

4.2.5 Esquema eléctrico del circuito para los motores DC.

Los motores usados para el desplazamiento del robot funcionan con una tensión entre 3v y 6v. Tienen un consumo de de unos 80mA. Aunque en un principio no es un consumo excesivamente alto, separar su alimentación de los circuitos de control, puede hacer más fácil la sustitución de los motores por otros más potentes en el futuro, contribuyendo a la escavilidad del sistema.

Ilustración 19: Motor DC.

Por otro lado, para lograr una buena protección entre los pines de salida de la placa MSP432P401R y el *driver* de potencia de los motores se usan opto-acopladores. De esta forma los dos circuitos de alimentación quedan eléctricamente aislados.

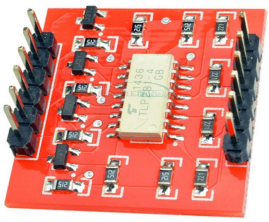


Ilustración 20: Opto acoplador

Además el módulo opto-acoplador tiene la ventaja de poder manejar directamente las señales de 3.3v, por lo que no hay necesidad de utilizar un conversor de nivel de tensión en este caso.

Después del modulo opto-acoplador se necesita un *driver* para manejar la potencia de los motores. Para esto se va a emplear el módulo MX1508 que va a permitir manejar dos motores con cambio de sentido. Este puede manejar hasta 1.5A por canal.

Ilustración 21: Driver Motores

Se va a conectar a la In1 la señal PWM para controlar la velocidad del motor, y seleccionamos el sentido de giro con la señal In2. La única pega de usar esta configuración es que cuando seleccionemos la marcha atrás la señal PWM, al hacer una and con la entrada In2, nos va a invertir el tiempo de ciclo a nivel alto por el de nivel bajo. Esto se arreglará mediante software.

In2	In1	Función
0	0	stop
0	1	adelante
1	0	atrás
1	1	stop

El esquema del circuito es el siguiente:

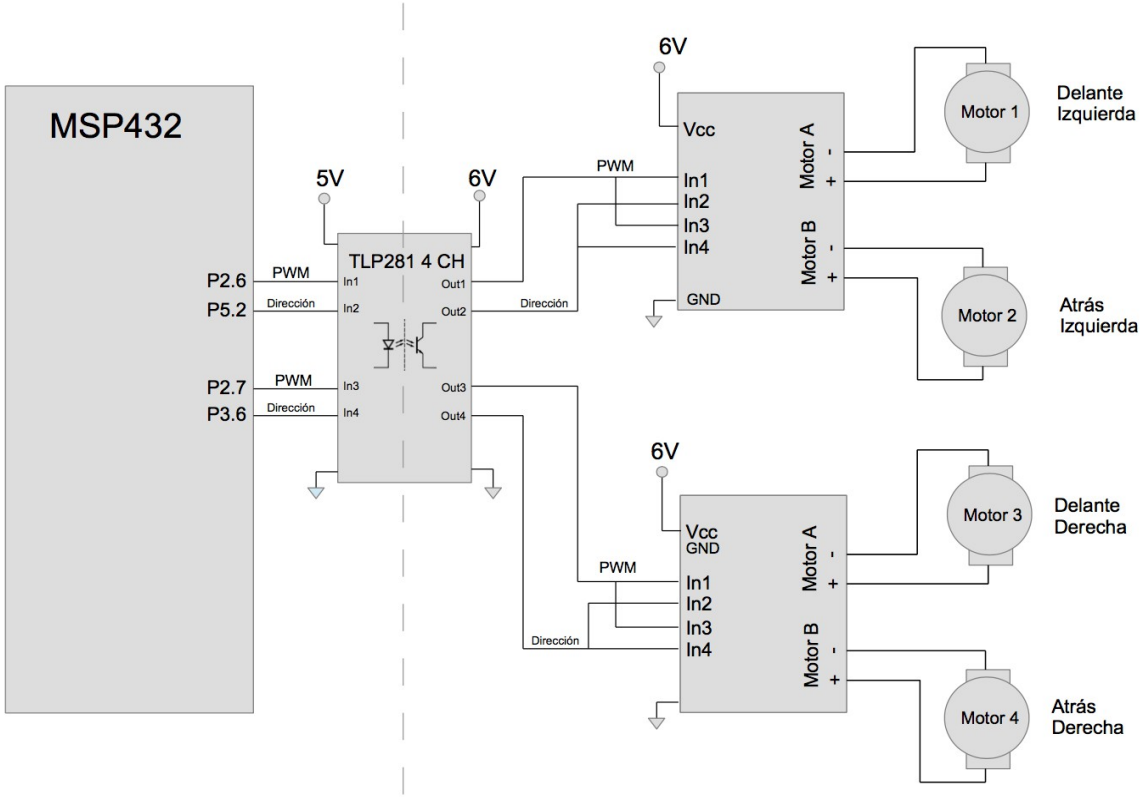


Ilustración 22: Esquema eléctrico para el control de los motores DC.

4.2.6 Esquema eléctrico del circuito para los servos de la cámara.

Para mover el encuadre de la cámara se usan dos servo-motores. En concreto el SG90 9G. Es un servo activo, muy pequeño y de sólo 9 gramos de peso. Trabaja con voltajes entre 4.2v y 6v, y su consumo es del muy bajo, aunque depende de la carga a la que esté sometido. En este caso no es un problema pues la cámara tiene un peso muy reducido.

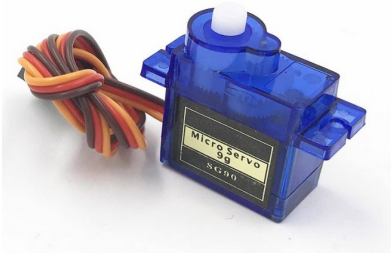


Ilustración 23: Servo SG90

El servo dispone de dos conexiones para la alimentación: positivo y negativo. Más una tercera por donde recibe la señal PWM para el control. La posición del servo dependerá del ancho de pulso de esta última. Para un ancho de 1,5 ms el servo se coloca en la posición central. Un ancho de 1ms -90º, y un ancho de 2ms 90º. El periodo de la señal PWM debe ser de 20ms.

Al tratarse de un servo activo, la corriente en la entrada de señal PWM va a ser muy pequeña. Incluso podríamos conectar la entrada de PWM directamente al pin del MSP432P401R, pues además los 3.3v son suficientes para manejarla. Sin embargo, como ya se ha comentado, se prefiere conectarlo a través de opto-acopladores para mayor seguridad y modificaciones futuras.

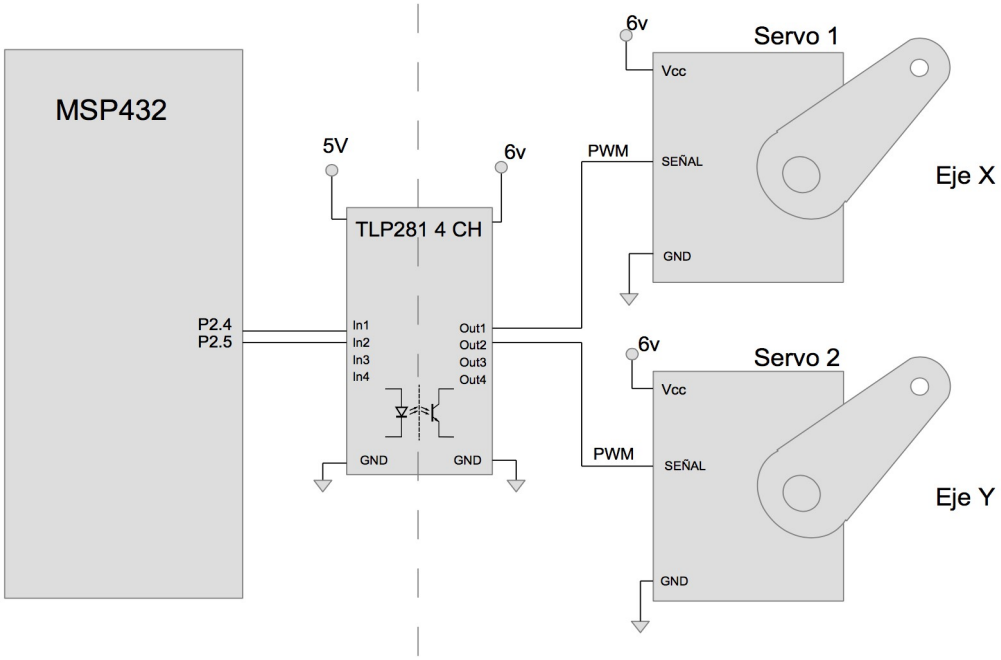


Ilustración 24: Esquema eléctrico del módulo para el control de los servos.

4.2.7 Esquema eléctrico del circuito para el encoder incremental.

Este módulo trabaja con señales de 3.3V o 5V. Conectamos la alimentación a uno de los pines de 3,3V que dispone el MSP432P401R, de esta forma la señal D0, que indica los pulsos que va leyendo el sensor, se puede llevar directamente al pin de entrada del MSP432P401R.

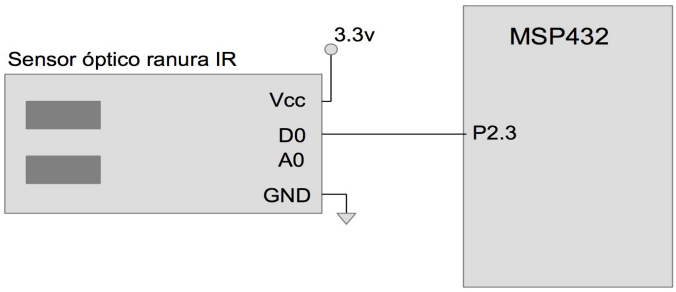


Ilustración 25: Circuito encoder incremental

4.2.8 Esquema del circuito de alimentación de la Raspberry.

Este circuito no se ha implementado. Se usaría un módulo que dispone de un relé y una etapa previa para excitar la bobina de éste.

El esquema de conexión es el siguiente:

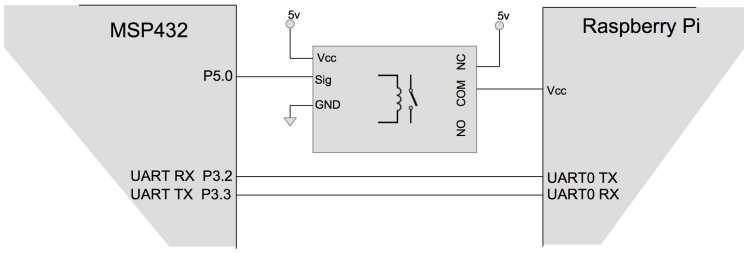


Ilustración 26: Circuito para el control de la alimentación de la Raspberry.

4.2.9 Conexión de la UART entre el MSP432P401R y la Raspberry

La conexiones UART entre los dos dispositivos se realizan mediante una conexión directa. Debido a que los dos manejan señales compatibles de 3.3v no se precisa de ninguna adaptación.

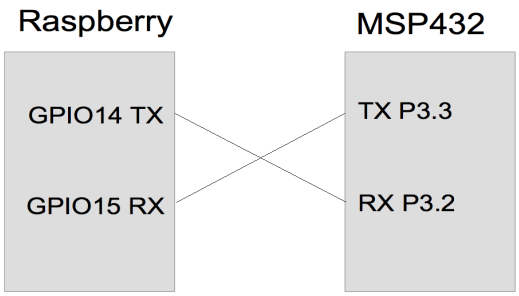


Ilustración 27: Conexión Puerto Serie

4.3 Descripción del software

El siguiente esquema da una idea general de las distintas capas de software sobre las que se sustentan los diferentes módulos del sistema.

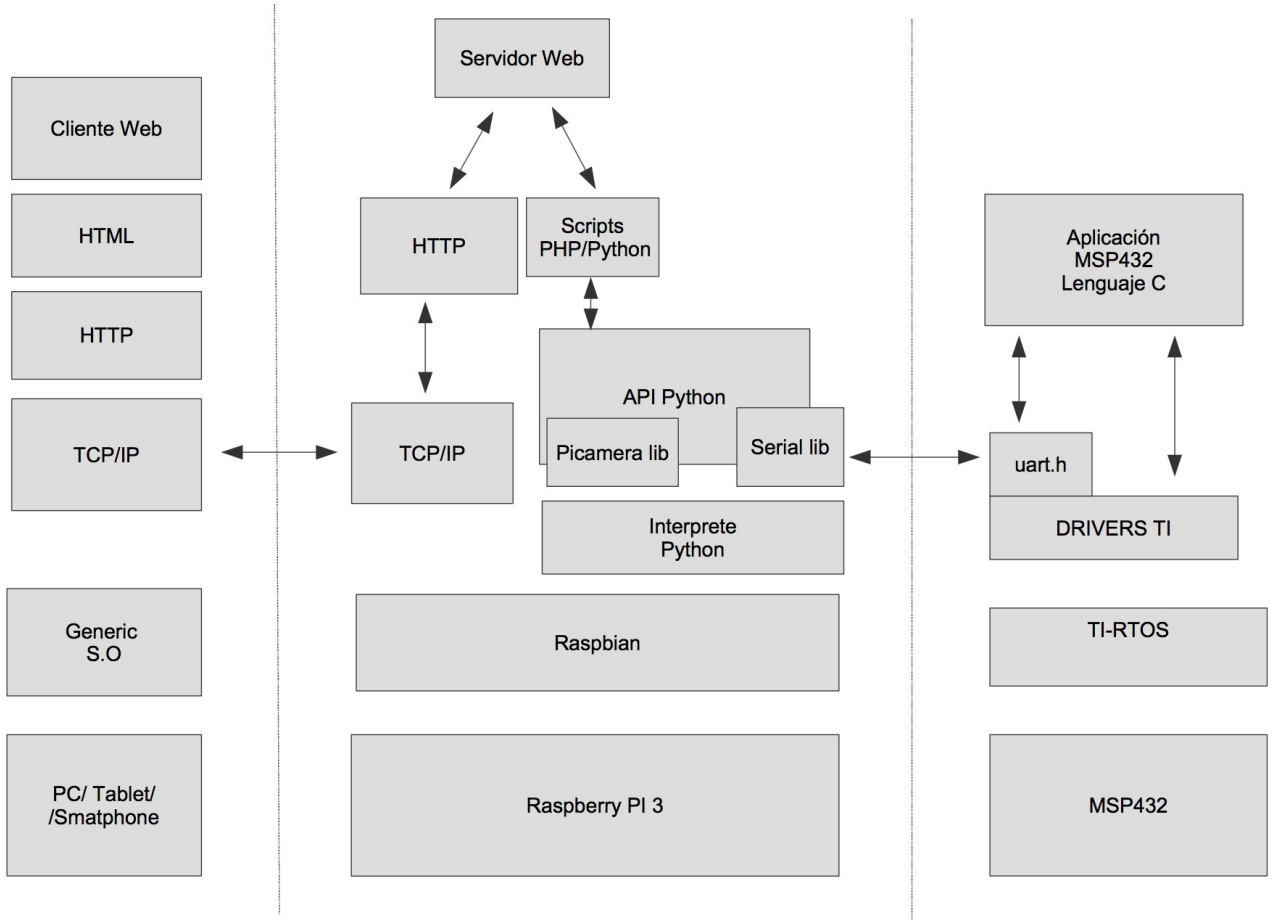


Ilustración 29: Capas Software.

En la figura se puede apreciar la separación entre el dispositivo del usuario, la Raspberry y el MSP432P401R.

4.3.1 Esquema general de los módulos software.

En el siguiente esquema se muestran los diferentes módulos software que se han desarrollado.

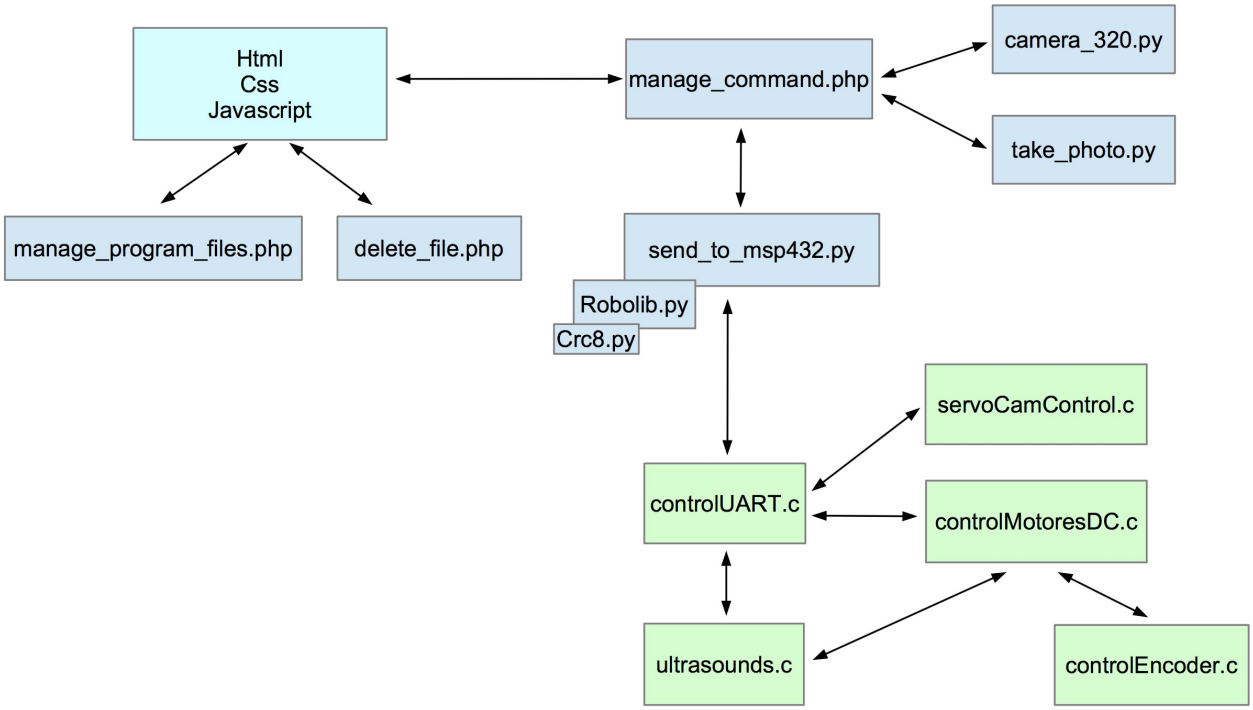


Ilustración 30: Esquema general módulos software .

Los módulos desarrollados en lenguaje C son threads que se ejecutan en el dispositivo MSP432P401R.

Los módulos desarrollados en PHP son ejecutados por la Raspberry, se lanzan a partir de las peticiones que recibe el servidor web Apache.

Los módulos Python se ejecutan en la Raspberry, estos son lanzados desde los módulos PHP por medio de una llamada exec.

Los módulos HTML,CSS y Javascript aunque residen en el servidor de la Raspberry, son ejecutados en el dispositivo del usuario.

4.3.2 Software del MSP432P401R

El siguiente esquema presenta en detalle las capas software sobre las que se desarrolla la aplicación que ejecuta el MSP432P401R.

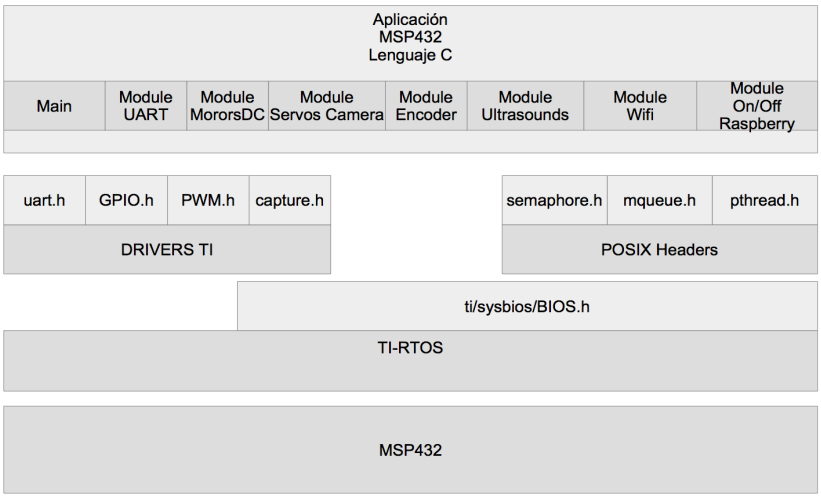


Ilustración 31: Capas software sobre las que se sustenta la aplicación del MSP432P401R.

4.3.2.1 Módulos ejecutados por el MSP432P401R

La aplicación está dividida en una serie de *threads*, todos ellos se ejecutan en paralelo, colaborando entre ellos para controlar los dispositivos conectados a esta placa.

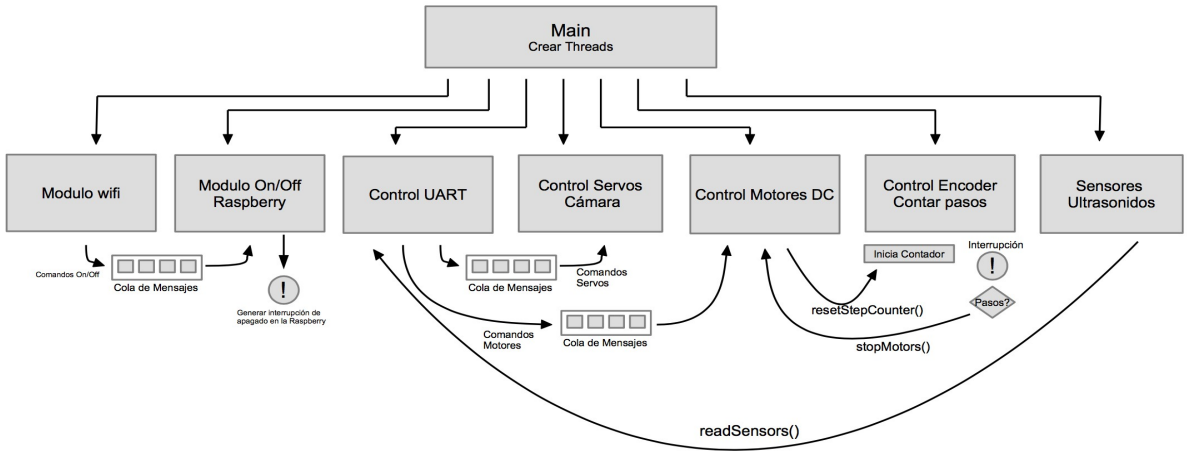


Ilustración 32: Threads ejecutados por el MSP432P401R.

Para lograr la coordinación de los distintos *threads* se usan los mecanismos de concurrencia que proporciona el sistema operativo TI RTOS. Se ha usado la capa POSIX para favorecer la portabilidad del código.

Colas de mensajes: El módulo Control UART envía los comandos a los módulos control Motores DC y Cotrol Servos Cámara. Se utiliza una cola de mensajes para en comunicarse con el thread que controla los motores y otra para el thread que controla los servos los servos. Estas colas son creadas por el módulo Main.

Semáforos: El módulo sensores ultrasonidos realiza operaciones de lectura y actualización del valor de cada uno de los sensores. Estas operaciones deben hacerse de forma atómica para evitar errores de lectura. También el módulo *encoder* usa un semáforo para acceder a una variable en exclusión mutua desde distintas partes de código que se ejecutan de forma asíncrona.

4.3.2.2 Módulo control UART

Este módulo ejecuta un thread que se encarga de todo el proceso de comunicación. Primero se abre el puerto serie y se ejecuta un bucle infinito. Dentro del bucle se espera a recibir una trama, tras comprobar que el CRC es correcto se decodifica el tipo de trama y se obtiene el comando para enviarlo al módulo correspondiente. Para la comunicación con los distintos módulos se emplean colas de mensajes. En estas se escriben los comandos que serán leídos por los otros módulos para materializar su ejecución.

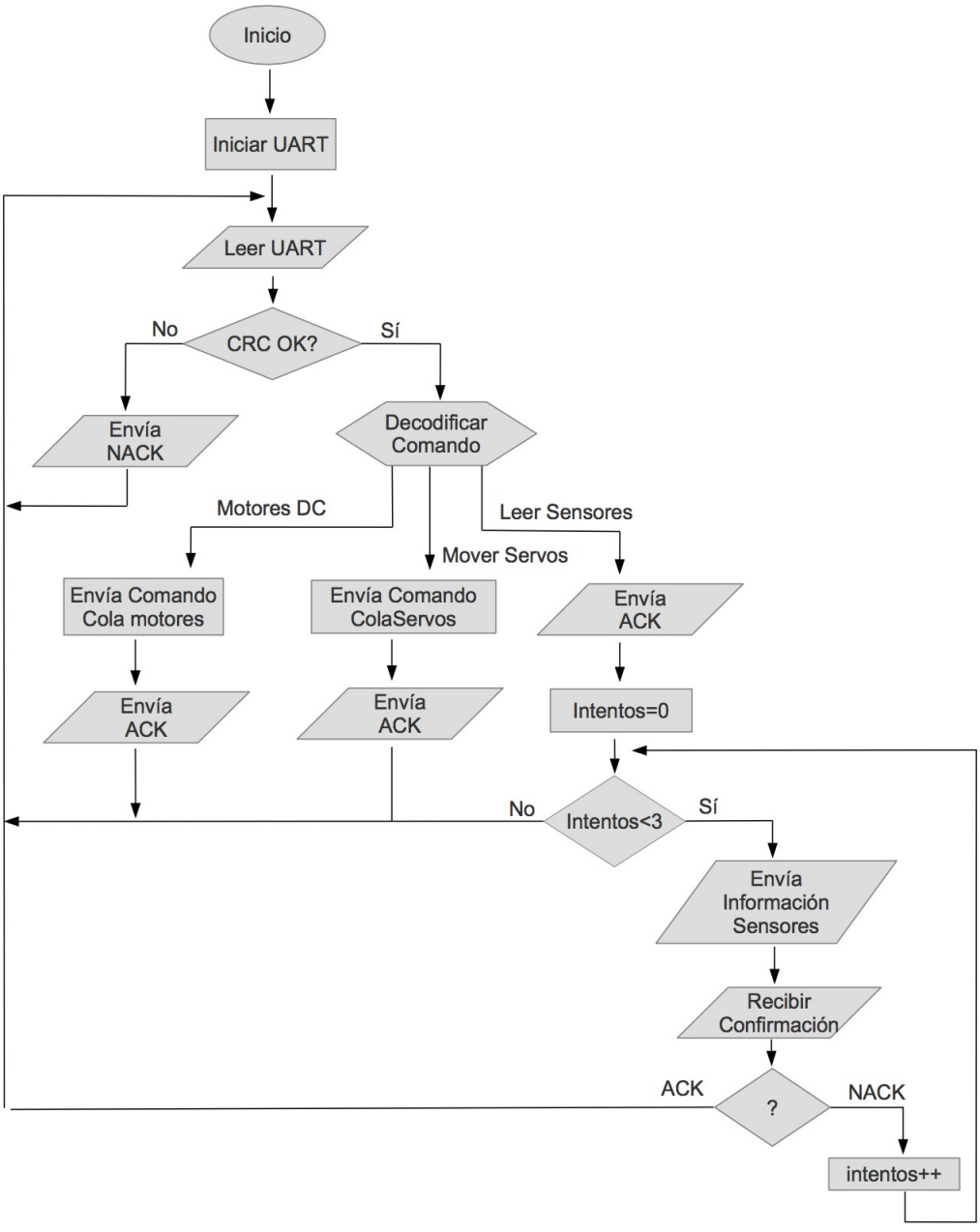


Ilustración 33: Diagrama de flujo. Módulo control UART

4.3.2.3 Módulo control de motores DC

Este módulo se encarga del control de los motores DC.

Para generar las señales PWM que controlan la velocidad de los motores este módulo utiliza el *driver* PWM de los Drivers TI. El período de la señal PWM es de 3ms.

Un tema a tener en cuenta es el de los *timers*. Vamos a usar señales PWM de distinta frecuencia. Los motores DC usan una frecuencia y los servos otra diferente. El *driver* PWM sólo se puede programar con los *timer* TA0 y TA1. Así que se ha movido el reloj del sistema(SYSBIOS), que por defecto trabaja con el TA0, al TA3 (El TA2 se usa para el *driver* Capture). Esta configuración se encuentra en el archivo *release.cfg* dentro del código del núcleo de TI-RTOS y es fundamental hacer ese cambio a la hora de compilar para que la aplicación funcione.

Los comandos que controlan los motores tienen el siguiente formato:

char	char	char	char	u_int 16
Es un carácter que indica la dirección de giro de los motores del lado Izquierdo. Puede ser : f: forward b:backware	Velocidad de los motores del lado lado izquierdo. Es un número de 1 a 5.	Es un carácter que indica la dirección de giro de los motores del lado Derecho. Puede ser : f: forward b:backware	Velocidad de los motores del lado lado izquierdo. Es un número de 1 a 5.	Distancia. Indica la distancia en centímetros que debe avanzar el robot.

Tabla 4: Formato de comando para el control de los motores DC

A partir del comando debemos generar las señales que se definieron para el control del los circuitos.

In2	In1	Función
0	0	stop
0	1	adelante
1	0	atrás
1	1	stop

Recordemos que la entrada In1 se usa para enviar la señal PWM que controla la velocidad del motor, y se usa In2 para seleccionar el sentido de giro.

En la programación de esta funciones debemos tener en cuenta que al invertir el sentido de giro, se invierte también el *duty cycle* ya que el circuíto internamente hace una AND entre las dos señales. Por ejemplo, si la señal PWM tiene un *duty cycle* del 70% con la señal In2 puesta a 1, tendrá un *duty cycle* del 30% con la señal In2 puesta a 0.

4.3.2.4 Módulo encoder.

Este módulo va a llevar la cuenta de los pulsos que genera el *encoder* incremental.

Se configura uno de los pines de entrada y se programa sobre él una interrupción hardware, la cual se activará cada vez que se detecte un flanco de subida.

Cada vez que se lanza la interrupción se consulta si se ha llegado al final y se actualiza el contador. Este proceso se debe llevar acabo en exclusión mutua. Ya que durante la cuenta se podría ordenar el reinicio del contador, por ejemplo si llega una nueva orden a los motores. Por este motivo se crea un semáforo para acceder de forma segura a las variables.

4.3.2.5 Servos

Este módulo también utiliza el *driver* PWM de los TI Drivers. En este caso el período de la señal PWM es de 20mS.

Recibe el comando a través de una cola de mensajes. El comando está formado por dos caracteres. El primero indica la posición del primer servo, eje X. Y el segundo la posición del segundo servo, eje Y. Los valores en las posiciones pueden tomar los siguientes valores.

Valor	Grados	Duty Cicle (us)
1	-90º	500
2	-65	750
3	-45	1000
4	-20	1250
5	0º Posición central	1500
6	20	1750
7	45	2000
8	65	2250
9	90º	2500

A partir de los parámetros que se reciben en el comando se activan las señales necesarias para re-posicionar los servos. Se hace corresponder cada una de las posiciones con un valor preestablecido para ajustar el *duty cycle* de la señal PWM.

4.3.2.6 Módulo ultrasonidos

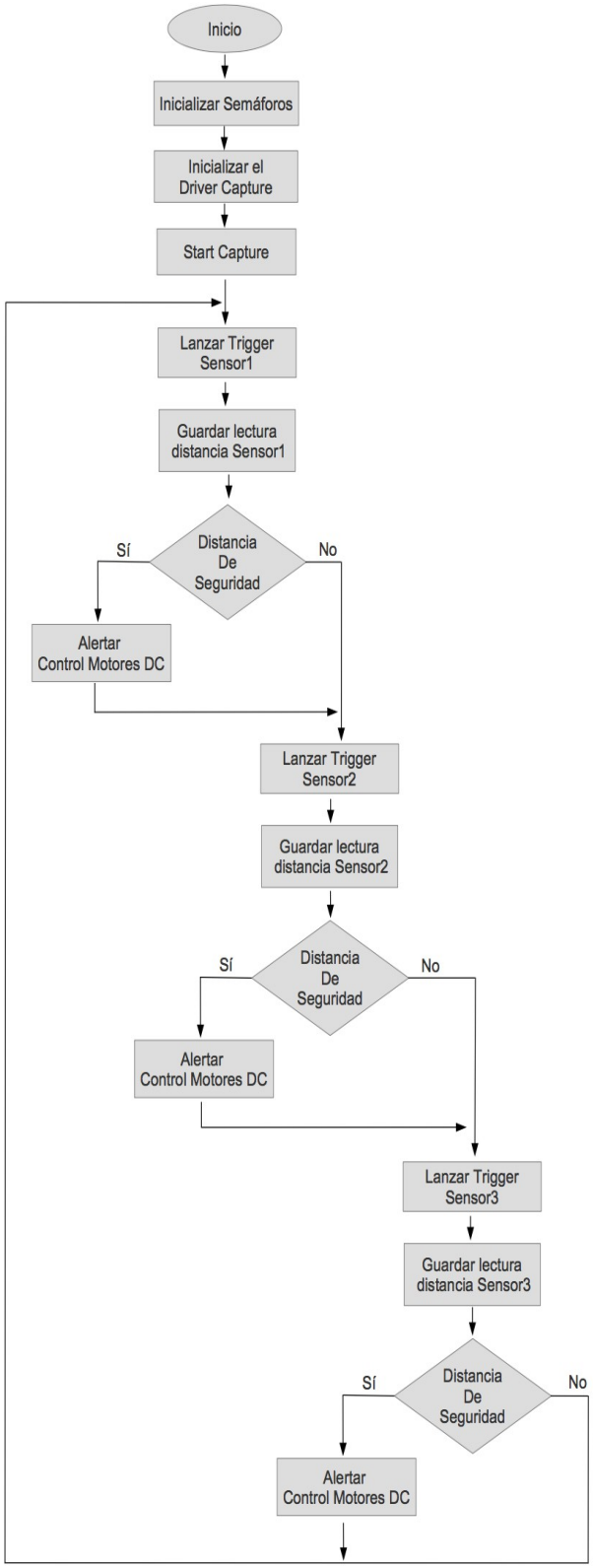


Ilustración 34: Diagrama de flujo. Módulo del sensor de ultrasonidos.

En el apartado 4.2.4 ya vimos vimos como funciona el sensor HC-SR04.

Básicamente, para cada sensor, necesitamos llevar a cabo dos acciones para determinar la distancia medida por un sensor de ultrasonidos.

- Disparar el Trigger, 10us.
- Medir el ancho del pulso de la señal Echo.

Para disparar los Triggers se usa un puerto con configuración de salida para cada uno de los sensores.

Las señales de Echo de cada sensor llegan todas por el mismo pin de entrada a la placa.

Una forma de medir el ancho de un pulso es programar una entrada para lanzar una interrupción hardware. Debemos configurar la interrupción de forma que se lance tanto en el flanco de subida como en el de bajada. Cogiendo los tiempos en cada uno de estos flancos para determinar el ancho del pulso.

Esta solución tiene un inconveniente. El pulso que queremos medir está en un rango de 150us a 25ms. Para coger los tiempos deberíamos emplear alguna función que nos de el tiempo del sistema. En TI RTOS el reloj del sistema está configurado por defecto a un periodo de 1ms, para superar esa resolución deberíamos disminuir ese periodo, pero esto podría tener consecuencias en el funcionamiento de todo el sistema. Además esta forma de medir los tiempos puede verse afectada por la planificación del procesador, por ejemplo un cambio de contexto podría falsear el instante concreto en que se pretende hacer la medida.

En su lugar se ha utilizado el *driver* Capture de la librería TI Drivers. Está pensado precisamente para resolver este tipo de situaciones. Este lanza una interrupción cada vez que se produce un flanco en el pin de entrada, y se usa un *timer* dedicado para medir los tiempos. En la interrupción se guarda el valor del *timer*, en lugar del tiempo del sistema, pero aquí no importa leer el valor un poco antes o después ya que el *timer* se encarga de coger y retener los tiempos medidos.

Par evitar que el robot choque de frente contra un obstáculo, cada vez que se lea un valor de uno de los sensores, se va a comprobar si este sobrepasa cierto limite de seguridad. En ese caso se llama a una función del módulo motores DC, el cual determinará la detención de los motores si el desplazamiento del robot es hacia delante.

Este módulo también proporciona un método para que otros módulos puedan consultar el valor de los sensores.

4.3.3 Software de la Raspberry.

4.3.3.1 Librería robolib.py

Esta es una librería desarrollada en Python que incluye todo lo necesario para enviar comandos desde la Raspberry al MSP432P401R.
Incluye las funciones para realizar las siguientes acciones:

- Configuración y apertura del puerto serie
- Envío y recepción de mensajes.
- Formateo de las tramas.
- Decodificación de los campos de las tramas.
- Comprobación del CRC.
- Bloqueo y desbloqueo del fichero usado para la exclusión mutua.
- Funciones para enviar comandos al robot.
    - Desplazamiento
    - Movimiento de los servos de la cámara
    - Lectura de los sensores de ultrasonidos.

4.3.3.2 Concurrencia en las operaciones de envío y recepción de mensajes.

El interprete de Python ejecutará los scripts que hacen uso de la librería robolib.py como procesos del sistema, por lo tanto pueden ejecutarse varios scripts de forma concurrente. Es evidente que debemos proporcionar un mecanismo de protección en el acceso a zonas críticas en el proceso de envío y recepción de mensajes por el puerto serie.

Para evitar que se se corrompan las tramas que circulan por el puerto, se debe asegurar que las operaciones de lectura y escritura se hacen en exclusión mutua. Por otro lado, el intercambio de mensajes definido por el protocolo de comunicación implica el envío de dos o más tramas, por ejemplo, se envía una trama que contiene el comando, y se recibe otra con el ACK . También debemos proteger que durante este proceso otro interlocutor envíe tramas ajenas a esa comunicación.

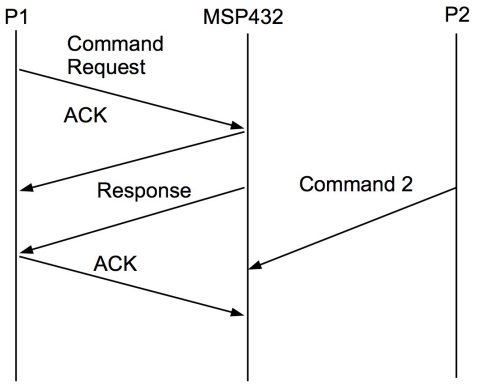


Ilustración 35: Interferencia de un comando sobre otro anterior

Este control es muy importante de cara al desarrollo de la aplicación web. La aplicación web ya de por sí puede generar muchos accesos concurrentes, pues se pueden conectar varios clientes al servidor apache al mismo tiempo y lanzar comandos contra el robot. Por otro lado, aunque asumamos que sólo un usuario debería manejar el robot, nos encontramos con la naturaleza asíncrona, por los eventos, propias de Javascript. Hay eventos que se producen por la interacción del usuario con algún objeto de la aplicación web, un botón por ejemplo. Pero también hay eventos que se lanzan periódicamente, o al actualizar una página, ir hacia atrás etc. Cada uno de estos eventos va a ser controlado por un proceso ejecutando el interprete de Python. Incluso podríamos lanzar uno de los programas subidos al robot y a la vez actuar sobre los controles manuales de este, de forma que se crearan varios procesos en el sistema que intentaran acceder a las funciones de la librería robolib.py concurrentemente.

Una vez identificado el problema pasemos a discutir la solución planteada.

En Python sólo existen mecanismos de concurrencia entre hilos de ejecución. Pero, como se ha comentado anteriormente, lo que se necesita son mecanismos de concurrencia entre procesos, Aunque hay algún proyecto que trabaja para la implementar concurrencia entre procesos en Python, me pareció que no estaba lo suficientemente maduro como para incorporalo en desarrollo de este proyecto.

La solución propuesta se basa en el bloqueo de ficheros del sistema. Usando la función *lockf* que sí tiene implementación en Python. Se ha incorporado un fichero de bloqueo para controlar los accesos. El fichero se guarda en el directorio /run/lock/ del sistema. Este directorio está implementado en un sistema de archivos tmpfs, lo que quiere decir que se ubica en memoria RAM, y no en el disco. Esto hace que el acceso sea más rápido. Aunque el fichero no persiste en un reinicio del sistema, tampoco lo necesitamos, se crea cuando se requiere el primer acceso. Además no viene mal evitar escrituras en la tarjeta SD.

Para bloquear el archivo un proceso debe abrir el fichero en modo de escritura “w”. Después ejecuta la función lockf con el parámetro LOCK_EX para obtener el recurso en exclusión mutua. Para liberar el recurso se vuelve a invocar a la función con el parámetro LOCK_UN.

En mismo archivo de bloqueo se escribe el PID del proceso que toma el bloqueo del archivo. Cuando el proceso hace el desbloqueo borra su número de proceso y escribe una marca para saber que se ha desbloqueado sin problema. Esta información puede ser útil a la hora de hacer una traza de lo que pasa en el sistema, o bien implementar un demonio que supervise su estado. Si un proceso deja bloqueado el archivo pero ya no existe en el sistema ningún otro proceso podrá acceder a la sección crítica.

4.3.3.3 Módulos de la Aplicación web

La siguiente ilustración muestra los diferentes actores que intervienen el la aplicación web y cómo se relacionan.

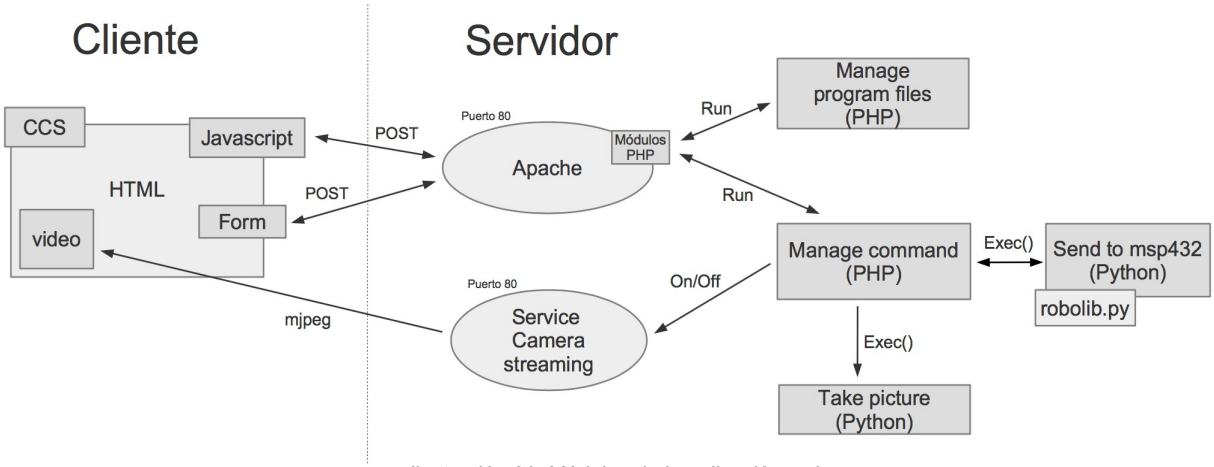


Ilustración 36: Módulos de la aplicación web

Desde la página web, un usuario lanza las peticiones contra el servidor, bien desde javascript por medio de un evento, o bien al accionar un botón incrustado en un formulario. En cualquiera de los casos se usa el método POST del protocolo HTTP. La petición llega al servidor apache que se está ejecutando en la Raspberry y esté lanza el módulo PHP correspondiente.

4.3.3.4 Módulo para manejar los archivos de programa. Manage program files.

El archivo manage_program_files.php es un script que genera una página HTML para gestionar los archivos de programa. Se encarga de subir los ficheros seleccionados por el usuario y de generar y actualizar la lista de los mismos. Se compone de un fichero en PHP y otro en Javascript.

En el fichero Javascript se encuentran las funciones que ejecutan el POST para borrar o ejecutar el archivo. También una función que actualiza la página cuando hay algún cambio en la lista de archivos.

4.3.3.5 Módulo para manejar los comandos.

El archivo manage_command.php es otro script en PHP. Se va a encargar de ejecutar un script dependiendo del comando a ejecutar. En este punto se crean diferentes procesos, mediante una llamada *exec,* que ejecutan el interprete de Python. Dependiendo del comando se lanzara uno de los siguientes

scripts:

- Comando a enviar a la Raspberry: Se lanza el script Send_to_MSP432.py. Este script envía los comandos por el puerto serie al MSP432P401R.
- Comando para tomar una fotografía: Se lanza el script Take picture.sh. Este script para el servidor de video, toma una fotografía y la guarda en el directorio pictures.
- Comando Para activar/desactivar el servidor de video: Se lanza un script bash que activa o desactiva el servicio.

4.3.1.1 El servidor de vídeo

Para el servidor de vídeo se ha creado un servicio en el sistema operativo para controlar la puesta en marcha de este servidor.

Para ello se ha creado el archivo */lib/systemd/system/camera_rpi.service*. Con esta configuración es posible lanzar el servidor de video como un servicio más del sistema.

Para darlo de alta en el sistema ejecutamos:

```
#systemctl daemon-reload
#systemctl enable camera_rpi.service
```

Para ponerlo en marcha

```
#systemctl start camera_rpi.service
```

Para pararlo

```
#systemctl stop camera_rpi.service
```

4.3.1.1 Diseño de la página web.

La imagen siguiente muestra la estructura principal de la página web.

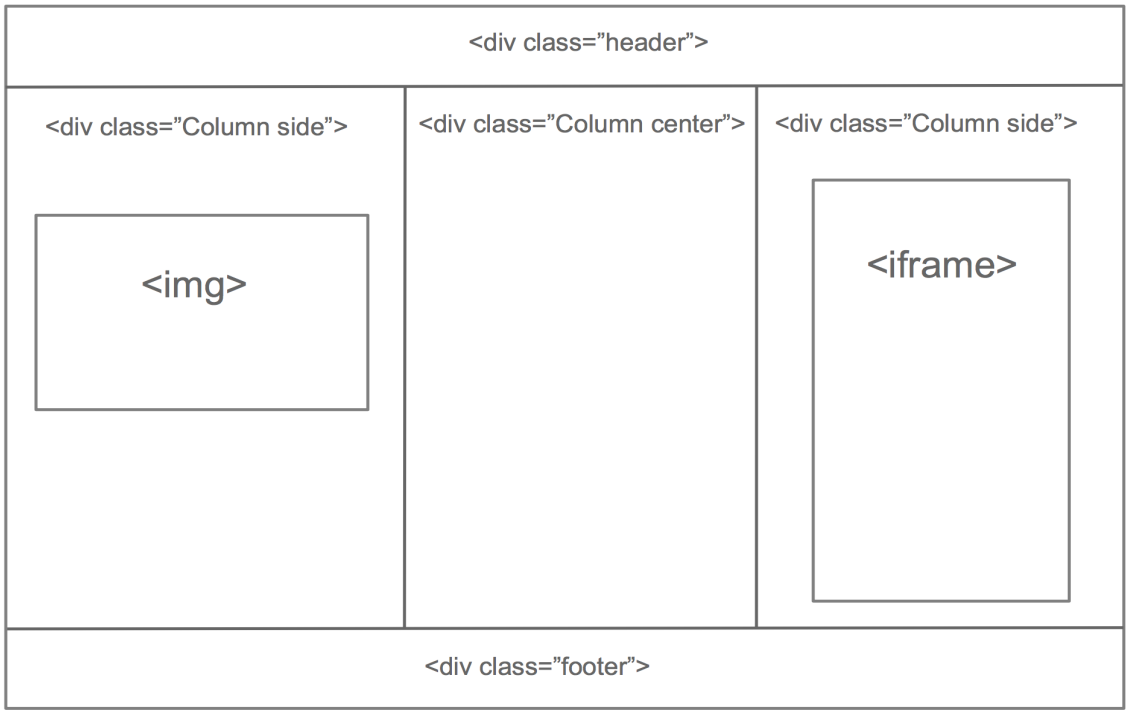


Ilustración 37: Estructura de los div de la página web.

En el div de la derecha se ha insertado una etiqueta <*iframe*>. En este se carga con el contenido de la página generada por el script manage_program_files.php. Esto se hace así para poder actualizar sólo esta parte de la web, de forma que no se pierdan los valores de los otros objetos en la página.

En la columna de la derecha hay una etiqueta <img>. Esta se usa tanto para mostrar el video en streaming, como para las fotografías.

4.4 Otros ficheros y configuraciones en el sistema

- Se ha instalado y configurado el servidor apache2.
- El directorio usado para publicar la aplicación web es */var/www/html/roboweb*
- Se ha instalado el interprete de PHP y el módulo para apache. Se ha configurado el archivo *php.ini* para permitir la subida de archivos al servidor.
- Se usa el fichero */run/lock/robot.lock* como archivo de bloqueo para implementar acceso a secciones críticas entre los procesos que ejecutan funciones de comunicación por el puerto serie.
- Los directorios *program_files y pictures* deben pertenecer al usuario *www-data* para que pueda escribir en ellos. El resto de ficheros basta con que tengan permiso de lectura.
- Se debe crear enlaces simbólicos en el directorio *program_files* hacia las librerías que están en el directorio *scripts.* En concreto los siguientes archivos *robolib.py, crc8.py y libcrc8.so.*

5 Viabilidad técnica

Antes de la compra de los materiales se hizo una investigación a fondo sobre el tipo de componentes a utilizar. Tratándose de un proyecto con fines educativos me aseguré de que fuese un material ampliamente usado, fácil de conseguir y económico.

Sobre el material de la asignatura no tenía experiencia, pues en el momento de cursar la asignatura de Sistemas Empotrados no se trabajaba con este material.

Para cada uno de los componentes me fijé en otros proyectos que encontré en la web, investigando cómo los usaban, sus posibilidades y características técnicas, tratando de encontrar similitudes y diferencias con mi proyecto.

Como resultado, todos los componentes son tecnologías probadas y se encuentra amplia documentación para facilitar su uso . No ha habido sorpresas en este apartado y se ha podido obtener los resultados esperados.

La totalidad del software es libre o gratuito.

Los componentes escogidos son capaces de hacer funcionar este proyecto.

6 Valoración económica

6.1 Coste de los materiales

Descripción	Cantidad	Precio por unidad	Portes	Precio en Euros
Raspberry Pi 3	1	35,99	incluido	35,99
Launchpad MSP432P401R Launchpad	1	11,8	7 (estimado)	18,8
Tarjeta SD 16 Gb	1	13,9	incluido	13,9
Sensores ultrasonidos HC-SR04	4	0,58	1,06	3,38
Kit Chasis del robot Motores DC	1	13,48	incluido	13,48
Servos sg90	2	0,98	0,87	2,83
Cámara para Raspberry	1	9,05	incluido	9,05
Soporte articulado Cámamara	1	0,52	1,65	0,57
Cable para la cámara 30cm	1	1,39	0,4	1,79
Módulo relé	1	2,03	incluido	2,03
Módulo opto-acopladores	2	0,59	0,73	1,91
Módulo *driver* potencia motores	2	0,41	1,25	2,07
Convertidor de nivel	1	0,54	0,85	1,39
Soportes sensor ultrasonidos	3	0,35	1,35	1,7
Caja para baterías	1	2,35	incluido	2,35
Power Bank	1	3,54	incluido	3,54
Batería de litio recargable 18650 3,7 v 3400 mah	6	14,18	incluido	14,18
Cables dupont 120 pc tipos y distintas medidas	1	3,58	incluido	3,58
Terminales 40 Pines 2,54 mm	2	0,21	0,67	1,09
			Total	133,63

Proveedores:

La Raspberry y la tarjeta SD se compraron en Amazon.

El resto de componentes se compraron en AliExpress.

6.2 Costes de industrialización

Aunque el proyecto tiene un enfoque meramente académico se hace una aproximación de los costes de industrialización.

6.2.1 Coste de desarrollo del prototipo.

Recurso	Horas	Precio/Hora	Gastos
Analista Programador	97	90,00 €	8.730,00 €
Programador Senior	210	60,00 €	12.600,00 €
Coste Materiales			133,63 €
		Total	21.463,63 €

Tabla 5: Coste del desarrollo del prototipo.

6.2.2 Costes de industrialización.

Descripción	Gastos
Prototipo	21.463,63 €
Promoción	6.000,00 €
Documentación Industrial(certificado CE)	20.000,00 €
Total	47.463,63 €

Tabla 6: Costes de industrialización.

6.2.3 Costes de fabricación.

Fabricación Unidades	Coste/ unidad	Precio venta	Gastos	Ganancia	Beneficios	Amortización
1000	99,00 €	230,00 €	146.463,63 €	230.000,00 €	83.536,37 €	637 unidades
2000	95,00 €	230,00 €	237.463,63 €	460.000,00 €	222.536,37 €	1032 unidades
5000	92,00 €	230,00 €	507.463,63 €	1.150.000,00 €	642.536,37 €	2207 unidades

Tabla 7: Costes de fabricación.

7 Conclusiones

A continuación se vuelve a dar una lista de los objetivos iniciales comentado en cada caso el grado de consecución de los mismos.

7.1.1 Objetivos principales

- **Detección de obstáculos y medida de distancias por medio de sensores de ultrasonidos:** Se ha alcanzado el objetivo. El único problema detectado es el inherente a los sensores de ultrasonidos comentado en el apartado 4.2.4
- **Desplazamiento del vehículo por medio de Motores DC:** Se ha alcanzado el objetivo. El vehículo se mueve con facilidad y precisión.
- **Movimiento de la cámara. Control con servos PWM:** Se ha alcanzado el objetivo. El movimiento de la cámara es correcto. Quizá lo que no pude controlar fue la velocidad de movimiento de los servos, que a veces tienen un movimiento algo brusco. En algunas ocasiones también se ha detectado una vibración que se retroalimenta. Creo que es resultado de usar componentes de baja calidad.
- **Cuantificar el desplazamiento del vehículo por medio de un encoder incremental:** Se ha alcanzado el objetivo. La medición es bastante buena.
- **Comunicación Raspberry – MSP432P401R mediante UART:** Se ha alcanzado el objetivo. Se debe trabajar más para asegurar la robustez en este apartado. Aunque es perfectamente funcional y no se han detectado fallos todavía.
- **Librería API para el control de los comandos del MSP432P401R mediante UART:** Se ha alcanzado el objetivo. Sería conveniente estudiar los métodos de empaquetado de librerías que se usan en Python.

- **Servidor WEB y página de acceso al robot:** Se ha alcanzado el objetivo. La web es bastante funcional.
- **Obtención de imágenes desde la cámara:** Se ha alcanzado el objetivo
- **Ejecución de programas subidos por el usuario a la plataforma:** Se ha alcanzado el objetivo
- **Conexión wifi MSP432P401R:** No se ha alcanzado este objetivo. El mayor problema fue la falta de tiempo y la complejidad en la puesta en marcha de la placa
- **Encendido/apagado de la Raspberry a través del MSP432P401R:** No se ha conseguido el objetivo. Por motivos de falta de tiempo.

7.1.2 Objetivos Secundarios

- **Gestión en remoto de los programas, subir, bajar, borrar, ver fichero de salida:** Objetivo conseguido.
- **Retransmisión de vídeo en *streaming:*** *Objetivo conseguido.*

7.2 Conclusiones del trabajo. Lecciones aprendidas.

En este proyecto he podido poner en práctica muchos de los conocimientos que he ido aprendiendo a lo largo de mis estudios. He comprobado que de la teoría a la práctica hay mucho camino por recorrer, y no hay que dar nada por sentado. Este trabajo me ha dado la oportunidad de aprender muchos temas prácticos que no se suelen abordar en el transcurso de una asignatura. También el integrar en un proyecto conocimientos varias asignaturas te da un punto de vista diferente sobre el trabajo de ingeniería.

7.3 Auto evaluación

7.3.1 Tiempo dedicado a cada PAC

- PAC0: 14 horas.
- PAC1: 25 horas.
- PAC2: 97 horas.
- PAC3: 95 horas.
- PAC4: 85 horas.
- Memoria: 32 horas.

- Total: 348 horas

7.3.2 Reflexión crítica

Por un lado me siento satisfecho del trabajo realizado. Sobre todo por haber sido capaz de realizar la mayor parte de los objetivos, de manera que el resultado es un sistema funcional. Por otro lado siento que me hubiese venido bien algo más de organización y previsión en los tiempos que iba a dedicar a cada tarea. He tenido que dedicar muchas horas más de las que inicialmente se planificaron. Respecto a la elaboración de esta memoria reconozco que subestimé el tiempo y esfuerzo que se debe dedicar. He hecho muchas más cosas de las que he podido documentar en esta memoria. Debería haberle dedicado tiempo desde el inicio del proyecto.

7.3.3 Análisis crítico de la planificación

La planificación de las diferentes fases fue demasiado optimista. Se iban cumpliendo los objetivos a costa de no hacer trabajos de documentación. El problema se agudizó en la tercera fase, no teniendo tiempo para cumplir con todos los objetivos. La memoria del proyecto también se ha visto perjudicada pues es muy complicado desarrollar un documento de esta envergadura al final del proyecto.

7.4 Líneas de trabajo futuro

- Implementación de gestión de usuarios.
- Carga de las baterías por inducción
- Incrementar el número de comandos: Por ejemplo para tener una realimentación del estado de cada periférico.
- Robustez del sistema.
- Seguridad de la red.
- Librerías en lenguaje C.
- Programación desde Scratch.

8 Glosario

[10] ADC: Convertidor analógico digital.

[40] Duty cycle: Porción del período que una señal PWM esta puesta a nivel alto.

[21] HTTP: Hypertext Transfer Protocol.

[41] Interrupción: Señal que avisa al procesador de un evento que debe ser atendido de forma asíncrona. Puede ser un evento producido por un elemento hardware o software.

[16] Microcontrolador: Circuito integrado programable que aúna en su interior una unidad central de proceso, memoria y periféricos de entrada y salida.

[32] PWM: Abreviatura de Pulse Width Modulation. PWM es una modulación digital generalmente usada para controlar la cantidad de energía que se entrega a un dispositivo.

[9] RTOS: Sistema operativo de tiempo real.

[10] Sistema embebido: Es un sistema computacional diseñado para realizar una tarea concreta.

[3] SSH: Secure Shell: protocolo para el acceso remoto a un servidor.

[22] TCP/IP Modelo de capas del conjunto de protocolos de Internet.

[9] TI: Texas Instruments. Es un fabricante de semiconductores.

[9] UART Universal Asynchronous Receiver-Transmitter.

9 Bibliografía

- TI-RTOS Kernel (SYS/BIOS)
  User's Guide
  Texas Instruments
  http://www.ti.com/lit/ug/spruex3u/spruex3u.pdf

- TI-RTOS 2.20
  User’s Guide
  Texas Instruments
  http://www.ti.com/lit/ug/spruhd4m/spruhd4m.pdf

- MSP432P401RSimpleLink™ MicrocontrollerLaunchPad™DevelopmentKit (MSP--EXP432P401R)
  User's Guide
  Texas Instrument
  http://www.ti.com/lit/ug/slau597f/slau597f.pdf

Recursos web:

- Raspbian
  https://www.raspbian.org/

- Raspberry
  https://www.raspberrypi.org/

- Apache
  https://httpd.apache.org/

- Raspberry video Straming
  https://randomnerdtutorials.com/video-streaming-with-raspberry-pi-camera/
  https://picamera.readthedocs.io/en/latest/recipes2.html

- CRC
  https://barrgroup.com/Embedded-Systems/How-To/CRC-Calculation-C-Code
  https://www.geeksforgeeks.org/cyclic-redundancy-check-python/
  https://rosettacode.org/wiki/CRC-32#Library

- Python Serial
  https://pimylifeup.com/raspberry-pi-serial/
  http://embeddedlaboratory.blogspot.com/2017/03/serial-communication-in-raspberry-pi.html
  https://www.teachmemicro.com/raspberry-pi-serial-uart-tutorial/